Graphen und Algorithmen

Vorlesung #7: Matchingtheorie

Dr. Armin Fügenschuh
Technische Universität Darmstadt
WS 2007/2008

Übersicht

Übersicht

* Matchings und erweiternde Wege

Übersicht

* Matchings und erweiternde Wege
* Satz von Berge

Übersicht

* Matchings und erweiternde Wege
* Satz von Berge
* Das Heiratsproblem und der Heiratssatz von Hall

Übersicht

* Matchings und erweiternde Wege
* Satz von Berge
* Das Heiratsproblem und der Heiratssatz von Hall
* Matchings in bipartiten Graphen

Übersicht

* Matchings und erweiternde Wege
* Satz von Berge
* Das Heiratsproblem und der Heiratssatz von Hall
* Matchings in bipartiten Graphen
* Der Ungarische Algorithmus

Übersicht

* Matchings und erweiternde Wege
* Satz von Berge
* Das Heiratsproblem und der Heiratssatz von Hall
* Matchings in bipartiten Graphen
* Der Ungarische Algorithmus
* Der Kuhn-Munkres-Algorithmus

Übersicht

* Matchings und erweiternde Wege
* Satz von Berge
* Das Heiratsproblem und der Heiratssatz von Hall
* Matchings in bipartiten Graphen
* Der Ungarische Algorithmus
* Der Kuhn-Munkres-Algorithmus
* Lösung des Chinesischen Postbotenproblems

Übersicht

* Matchings und erweiternde Wege
* Satz von Berge
* Das Heiratsproblem und der Heiratssatz von Hall
* Matchings in bipartiten Graphen
* Der Ungarische Algorithmus
* Der Kuhn-Munkres-Algorithmus
* Lösung des Chinesischen Postbotenproblems
* 1/2-Approximation des TSP von Christofides

Matchings (Paarungen, Korrespondenzen)

Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.

Matchings (Paarungen, Korrespondenzen)

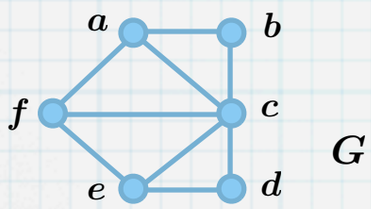
* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.

Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:

Matchings (Paarungen, Korrespondenzen)

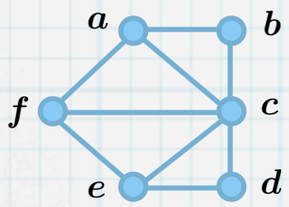
* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:



G

Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:

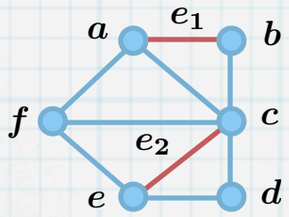


$M_1 := \{e_1, e_2\}$

G

Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:

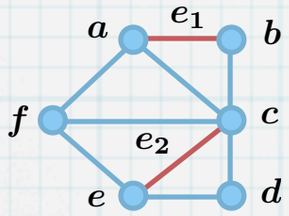


$M_1 := \{e_1, e_2\}$

G

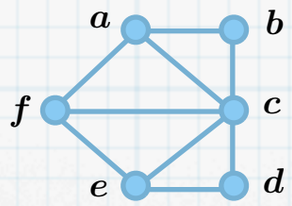
Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:



$M_1 := \{e_1, e_2\}$

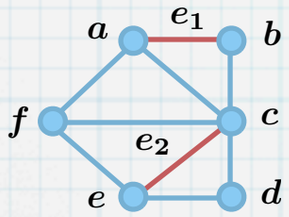
G



G

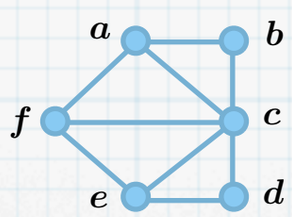
Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:



$M_1 := \{e_1, e_2\}$

G

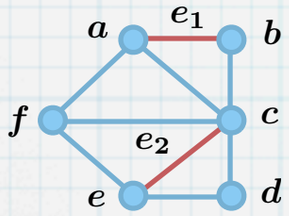


$M_2 := \{e_1, e_3, e_4\}$

G

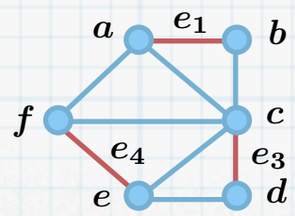
Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:



$M_1 := \{e_1, e_2\}$

G

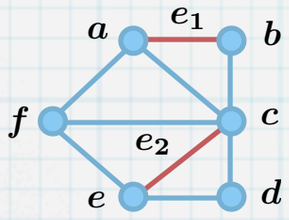


$M_2 := \{e_1, e_3, e_4\}$

G

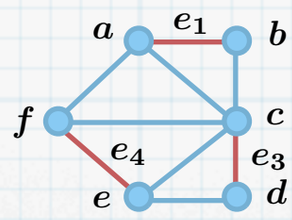
Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:



$M_1 := \{e_1, e_2\}$

G



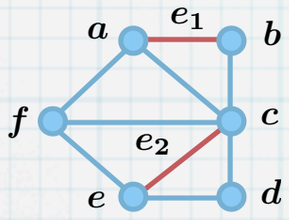
$M_2 := \{e_1, e_3, e_4\}$

G

* **Definition 2**:
  Ist ein Knoten v des Graphen G ein Endknoten einer Kante im Matching M, dann wird v als M**–gesättigt** bezeichnet. Man sagt auch, dass v **durch** M **gesättigt** wird.

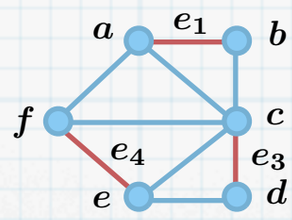
Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:



$M_1 := \{e_1, e_2\}$

G



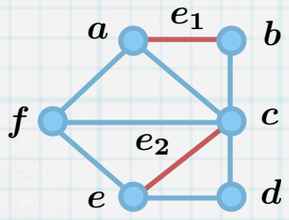
$M_2 := \{e_1, e_3, e_4\}$

G

* **Definition 2**:
  Ist ein Knoten v des Graphen G ein Endknoten einer Kante im Matching M, dann wird v als M**–gesättigt** bezeichnet. Man sagt auch, dass v **durch** M **gesättigt** wird.
  Andernfalls ist v M**–ungesättigt.**

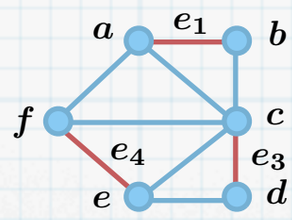
Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:



$M_1 := \{e_1, e_2\}$

G



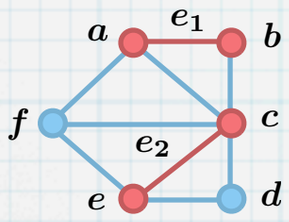
$M_2 := \{e_1, e_3, e_4\}$

G

* **Definition 2**:
  Ist ein Knoten v des Graphen G ein Endknoten einer Kante im Matching M, dann wird v als M**–gesättigt** bezeichnet. Man sagt auch, dass v **durch** M **gesättigt** wird.
  Andernfalls ist v M**–ungesättigt**.
* Beispiel (Forts.):
  Die Knoten a, b, c, e sind M_1–gesättigt, f und d sind M_1–ungesättigt.

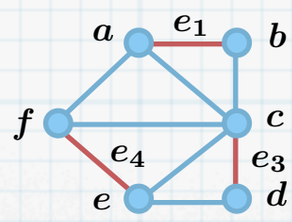
Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:



$M_1 := \{e_1, e_2\}$

G



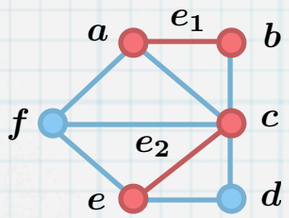
$M_2 := \{e_1, e_3, e_4\}$

G

* **Definition 2**:
  Ist ein Knoten v des Graphen G ein Endknoten einer Kante im Matching M, dann wird v als M**–gesättigt** bezeichnet. Man sagt auch, dass v **durch** M **gesättigt** wird.
  Andernfalls ist v M**–ungesättigt**.
* Beispiel (Forts.):
  Die Knoten a, b, c, e sind M_1–gesättigt, f und d sind M_1–ungesättigt.

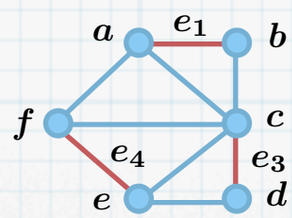
Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:



$M_1 := \{e_1, e_2\}$

G



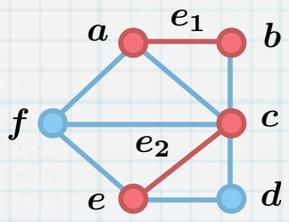
$M_2 := \{e_1, e_3, e_4\}$

G

* **Definition 2**:
  Ist ein Knoten v des Graphen G ein Endknoten einer Kante im Matching M, dann wird v als M**–gesättigt** bezeichnet. Man sagt auch, dass v **durch** M **gesättigt** wird.
  Andernfalls ist v M**–ungesättigt**.
* Beispiel (Forts.):
  Die Knoten a, b, c, e sind M_1–gesättigt, f und d sind M_1–ungesättigt.
  Jeder Knoten ist M_2–gesättigt.

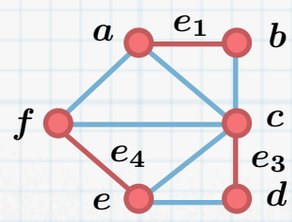
Matchings (Paarungen, Korrespondenzen)

* Generalvoraussetzung: In dieser Vorlesung sei G stets ein **schlingenfreier** (Multi–) Graph mit Knotenmenge $V = V(G)$ und Kantenmenge $E = E(G)$.
* **Definition 1**:
  Eine Kantenteilmenge $M \subseteq E$ heißt **Matching** von G, wenn keine zwei Kanten aus M einen gemeinsamen Endknoten haben.
* Beispiel:



$M_1 := \{e_1, e_2\}$

G



$M_2 := \{e_1, e_3, e_4\}$

G

* **Definition 2**:
  Ist ein Knoten v des Graphen G ein Endknoten einer Kante im Matching M, dann wird v als M**–gesättigt** bezeichnet. Man sagt auch, dass v **durch** M **gesättigt** wird.
  Andernfalls ist v M**–ungesättigt**.
* Beispiel (Forts.):
  Die Knoten a, b, c, e sind M_1–gesättigt, f und d sind M_1–ungesättigt.
  Jeder Knoten ist M_2–gesättigt.

Perfekte und maximale Matchings

Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M-gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.

Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.
* Beispiel (Forts.):
  M_2 ist ein perfektes Matching.

Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.
* Beispiel (Forts.):
  M_2 ist ein perfektes Matching.

  M_1 ist ein inklusionsmaximales Matching.

Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.
* Beispiel (Forts.):
  M_2 ist ein perfektes Matching.

  M_1 ist ein inklusionsmaximales Matching.
* Bemerkung:

Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.
* Beispiel (Forts.):
  M_2 ist ein perfektes Matching.

  M_1 ist ein inklusionsmaximales Matching.
* Bemerkung:

  Jedes perfekte Matching ist ein maximales Matching.

Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.
* Beispiel (Forts.):
  M_2 ist ein perfektes Matching.

  M_1 ist ein inklusionsmaximales Matching.
* Bemerkung:

  Jedes perfekte Matching ist ein maximales Matching.

  Die Umkehrung davon ist nicht wahr.

Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.
* Beispiel (Forts.):
  M_2 ist ein perfektes Matching.

  M_1 ist ein inklusionsmaximales Matching.
* Bemerkung:
  Jedes perfekte Matching ist ein maximales Matching.
  Die Umkehrung davon ist nicht wahr.
* Beispiel: Zwei maximale, nicht-perfekte Matchings.

Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

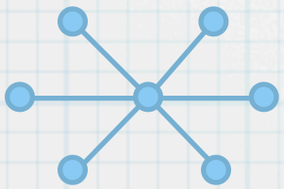
  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.
* Beispiel (Forts.):
  M_2 ist ein perfektes Matching.

  M_1 ist ein inklusionsmaximales Matching.
* Bemerkung:

  Jedes perfekte Matching ist ein maximales Matching.

  Die Umkehrung davon ist nicht wahr.
* Beispiel: Zwei maximale, nicht-perfekte Matchings.



Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

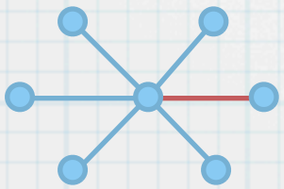
  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.
* Beispiel (Forts.):
  M_2 ist ein perfektes Matching.

  M_1 ist ein inklusionsmaximales Matching.
* Bemerkung:

  Jedes perfekte Matching ist ein maximales Matching.

  Die Umkehrung davon ist nicht wahr.
* Beispiel: Zwei maximale, nicht-perfekte Matchings.



Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

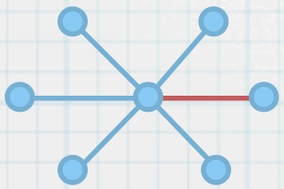
  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

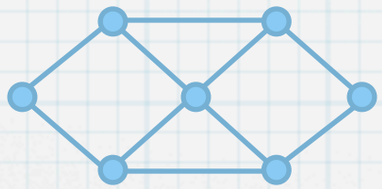
  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.
* Beispiel (Forts.):
  M_2 ist ein perfektes Matching.

  M_1 ist ein inklusionsmaximales Matching.
* Bemerkung:

  Jedes perfekte Matching ist ein maximales Matching.

  Die Umkehrung davon ist nicht wahr.
* Beispiel: Zwei maximale, nicht-perfekte Matchings.





Perfekte und maximale Matchings

* **Definition 3**:
  Ist M ein Matching von G, in dem jeder Knoten von G M–gesättigt ist, dann wird M als **perfektes Matching** bezeichnet.

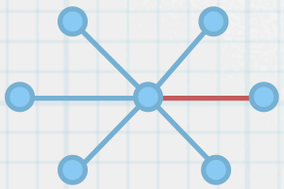
  Ein Matching M heißt **maximal**, wenn G kein Matching M' enthält, welches mehr Kanten als M hat.

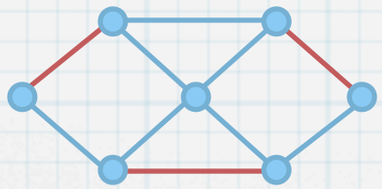
  Ein Matching M heißt **inklusionsmaximal**, wenn M nicht durch Hinzufügen einer weiteren Kante von G vergrößert werden kann.
* Beispiel (Forts.):
  M_2 ist ein perfektes Matching.

  M_1 ist ein inklusionsmaximales Matching.
* Bemerkung:

  Jedes perfekte Matching ist ein maximales Matching.

  Die Umkehrung davon ist nicht wahr.
* Beispiel: Zwei maximale, nicht-perfekte Matchings.





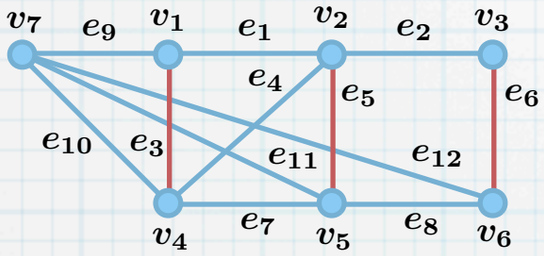
Alternierende und erweiternde Wege

Alternierende und erweiterndе Wege

* **Definition 4**:
  Sei M ein Matching von G. Ein M-**alternierender Weg** in G ist ein Weg, dessen Kanten abwechselnd zu M und $E \setminus M$ gehören.

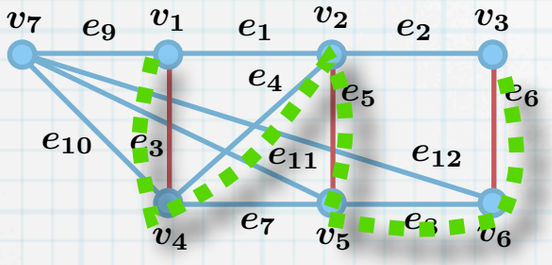
Alternierende und erweiterndе Wege

* **Definition 4**:
  Sei M ein Matching von G. Ein M-**alternierender Weg** in G ist ein Weg, dessen Kanten abwechselnd zu M und $E \setminus M$ gehören.
* Beispiel: $P := (v_1, e_3, v_4, e_4, v_2, e_5, v_5, e_8, v_6, e_6, v_3)$ ist ein $\{e_3, e_5, e_6\}$-alternierender Weg.



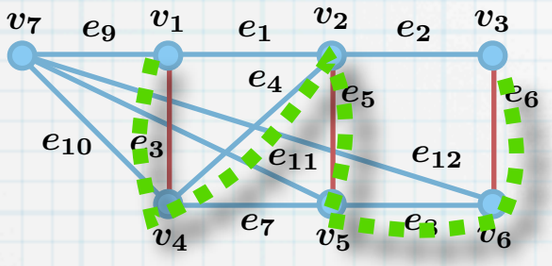
Alternierende und erweiterndе Wege

* **Definition 4**:
  Sei M ein Matching von G. Ein M-**alternierender Weg** in G ist ein Weg, dessen Kanten abwechselnd zu M und $E \setminus M$ gehören.
* Beispiel: $P := (v_1, e_3, v_4, e_4, v_2, e_5, v_5, e_8, v_6, e_6, v_3)$ ist ein $\{e_3, e_5, e_6\}$-alternierender Weg.



Alternierende und erweiterndе Wege

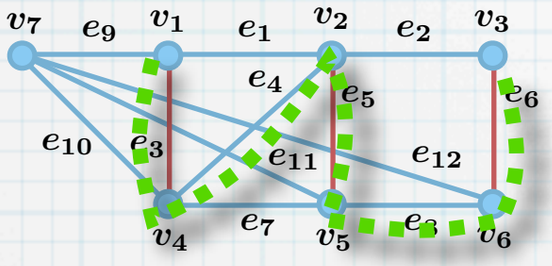
* **Definition 4**:
  Sei M ein Matching von G. Ein M-**alternierender Weg** in G ist ein Weg, dessen Kanten abwechselnd zu M und $E \setminus M$ gehören.
* Beispiel: $P := (v_1, e_3, v_4, e_4, v_2, e_5, v_5, e_8, v_6, e_6, v_3)$ ist ein $\{e_3, e_5, e_6\}$-alternierender Weg.



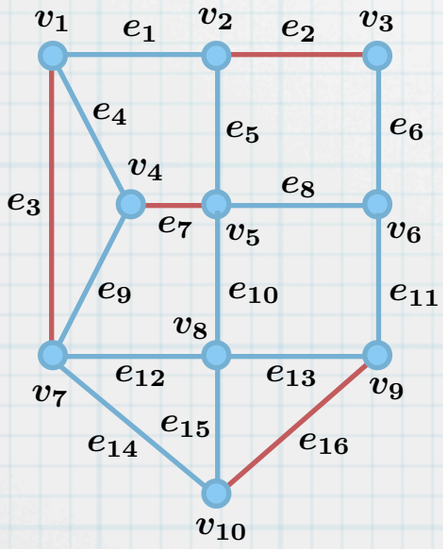
* **Definition 5**:
  Ein M-alternierender Weg, dessen Anfangs- und Endknoten M-ungesättigt sind, heißt M-**erweiternder Weg**.

Alternierende und erweiterndе Wege

* **Definition 4**:
  Sei M ein Matching von G. Ein M-**alternierender Weg** in G ist ein Weg, dessen Kanten abwechselnd zu M und $E \setminus M$ gehören.
* Beispiel: $P := (v_1, e_3, v_4, e_4, v_2, e_5, v_5, e_8, v_6, e_6, v_3)$ ist ein $\{e_3, e_5, e_6\}$-alternierender Weg.

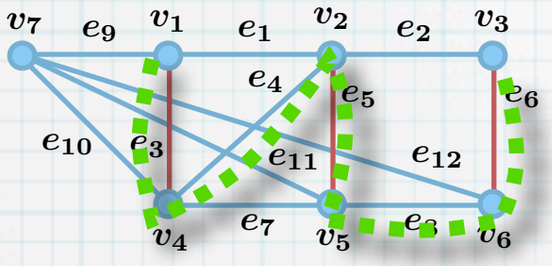


* **Definition 5**:
  Ein M-alternierender Weg, dessen Anfangs- und Endknoten M-ungesättigt sind, heißt M-**erweiternder Weg**.
* Beispiel: $P = (v_6, e_6, v_3, e_2, v_2, e_5, v_5, e_7, v_4, e_4, v_1, e_3, v_7, e_{12}, v_8)$ ist ein $\{e_2, e_3, e_7, e_{16}\}$-erweiternder Weg

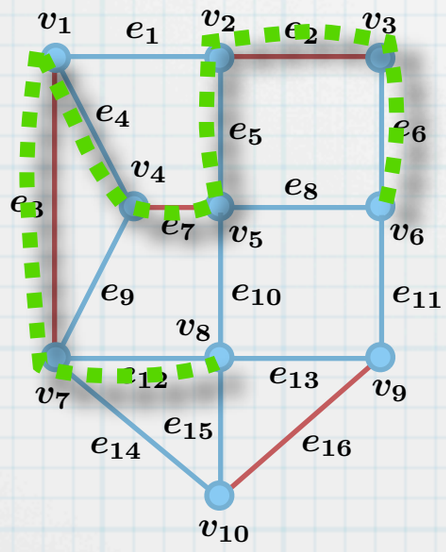


Alternierende und erweiterndе Wege

* **Definition 4**:
  Sei M ein Matching von G. Ein M-**alternierender Weg** in G ist ein Weg, dessen Kanten abwechselnd zu M und $E \setminus M$ gehören.
* Beispiel: $P := (v_1, e_3, v_4, e_4, v_2, e_5, v_5, e_8, v_6, e_6, v_3)$ ist ein $\{e_3, e_5, e_6\}$-alternierender Weg.



* **Definition 5**:
  Ein M-alternierender Weg, dessen Anfangs- und Endknoten M-ungesättigt sind, heißt M-**erweiternder Weg**.
* Beispiel: $P = (v_6, e_6, v_3, e_2, v_2, e_5, v_5, e_7, v_4, e_4, v_1, e_3, v_7, e_{12}, v_8)$ ist ein $\{e_2, e_3, e_7, e_{16}\}$-erweiternder Weg



Lemma von der symmetrischen Differenz

Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen,

Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen,

  oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.

Lemma von der symmetrischen Differenz

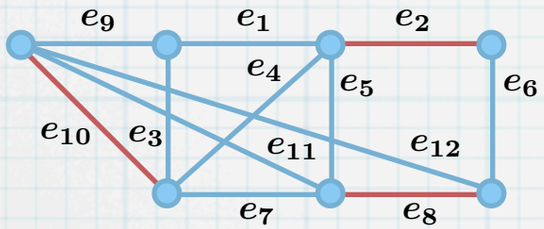
* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:

Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



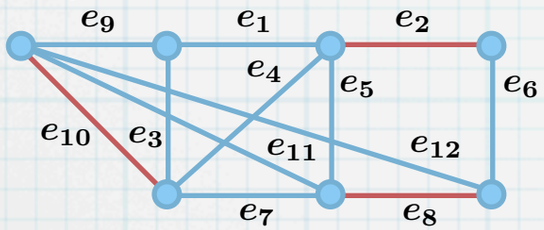
$M_1 := \{e_2, e_8, e_{10}\}$

Lemma von der symmetrischen Differenz

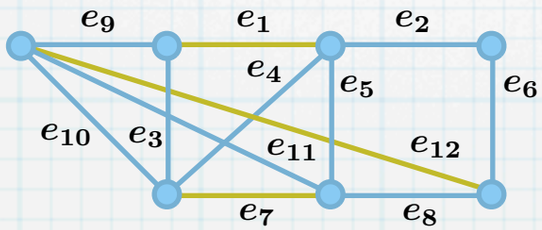
* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen,

  oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$

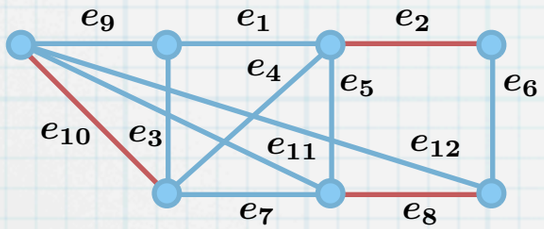


$M_2 := \{e_1, e_7, e_{12}\}$

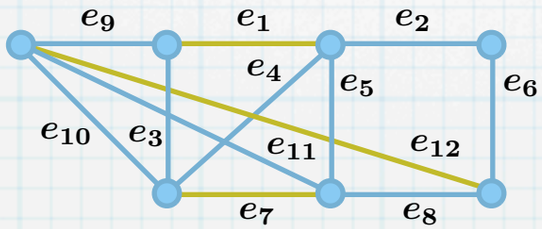
Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

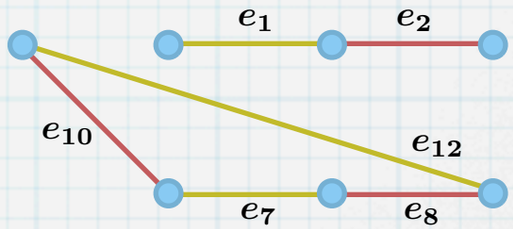
  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$



$M_2 := \{e_1, e_7, e_{12}\}$

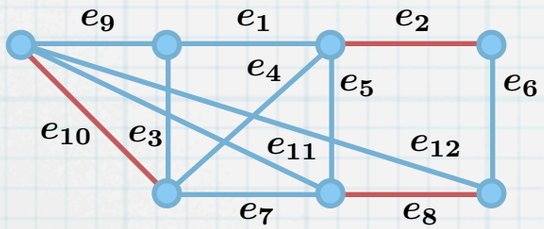


$M_1 \triangle M_2 := \{e_1, e_2, e_7, e_8, e_{10}, e_{12}\}$

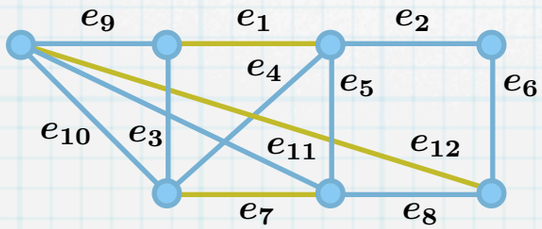
Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

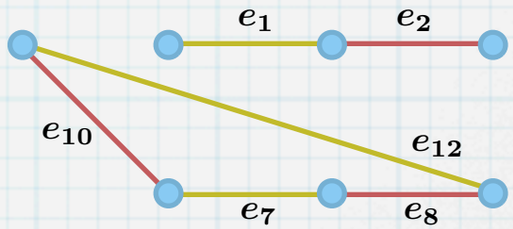
  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$



$M_2 := \{e_1, e_7, e_{12}\}$



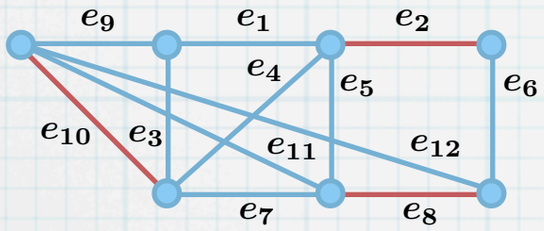
$M_1 \triangle M_2 := \{e_1, e_2, e_7, e_8, e_{10}, e_{12}\}$

* Beweis (von Lemma 6):

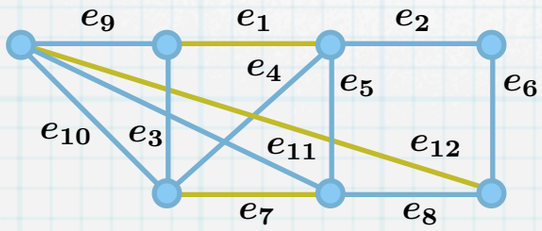
Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

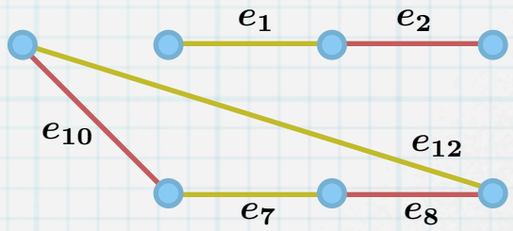
  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$



$M_2 := \{e_1, e_7, e_{12}\}$



$M_1 \triangle M_2 := \{e_1, e_2, e_7, e_8, e_{10}, e_{12}\}$

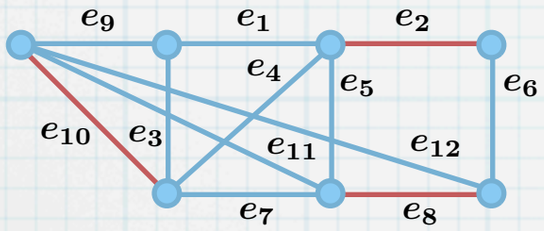
* Beweis (von Lemma 6):

  Sei v ein Knoten von H.

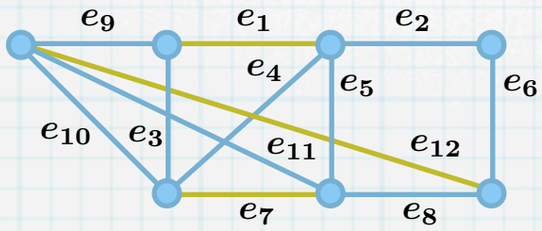
Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

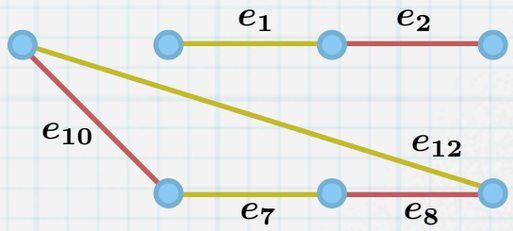
  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$



$M_2 := \{e_1, e_7, e_{12}\}$



$M_1 \triangle M_2 := \{e_1, e_2, e_7, e_8, e_{10}, e_{12}\}$

* Beweis (von Lemma 6):

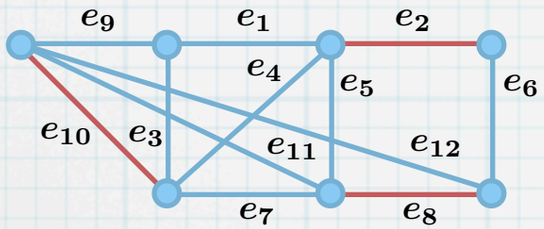
  Sei v ein Knoten von H.

  Da M_1 Matching, gibt es höchstens eine Kante in M_1, die v als Endknoten hat.

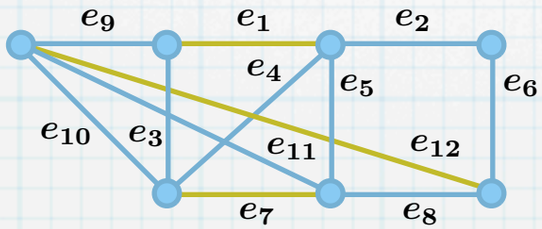
Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

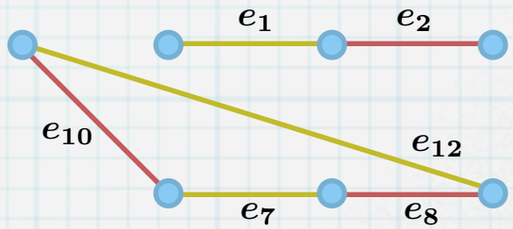
  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$



$M_2 := \{e_1, e_7, e_{12}\}$



$M_1 \triangle M_2 := \{e_1, e_2, e_7, e_8, e_{10}, e_{12}\}$

* Beweis (von Lemma 6):

  Sei v ein Knoten von H.

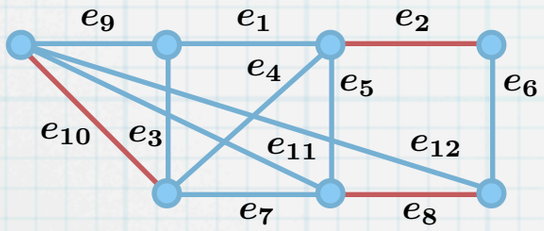
  Da M_1 Matching, gibt es höchstens eine Kante in M_1, die v als Endknoten hat.

  Ebenso gibt es höchstens eine Kante in M_2, die v als Endknoten hat.

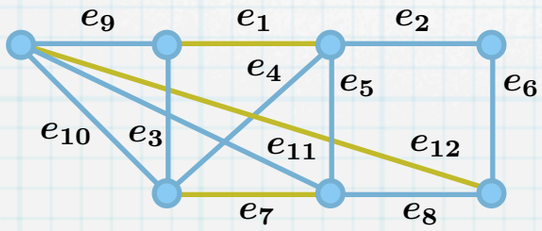
Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

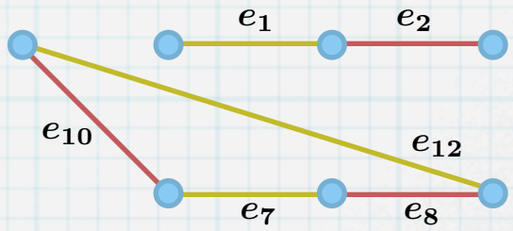
  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$



$M_2 := \{e_1, e_7, e_{12}\}$



$M_1 \triangle M_2 := \{e_1, e_2, e_7, e_8, e_{10}, e_{12}\}$

* Beweis (von Lemma 6):

  Sei v ein Knoten von H.

  Da M_1 Matching, gibt es höchstens eine Kante in M_1, die v als Endknoten hat.

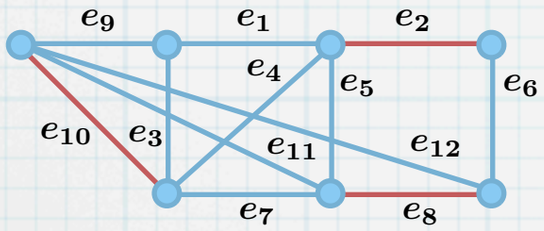
  Ebenso gibt es höchstens eine Kante in M_2, die v als Endknoten hat.

  1. Fall, v ist Endknoten einer Kante in $M_1 \setminus M_2$ und auch einer Kante in $M_2 \setminus M_1$.

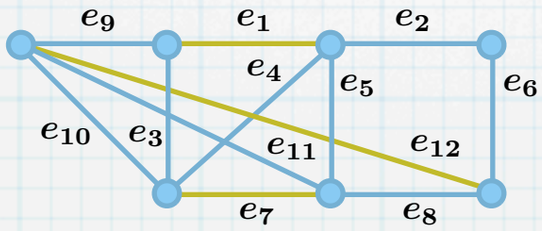
Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

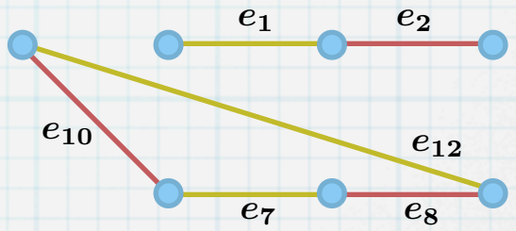
  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$



$M_2 := \{e_1, e_7, e_{12}\}$



$M_1 \triangle M_2 := \{e_1, e_2, e_7, e_8, e_{10}, e_{12}\}$

* Beweis (von Lemma 6):

  Sei v ein Knoten von H.

  Da M_1 Matching, gibt es höchstens eine Kante in M_1, die v als Endknoten hat.

  Ebenso gibt es höchstens eine Kante in M_2, die v als Endknoten hat.

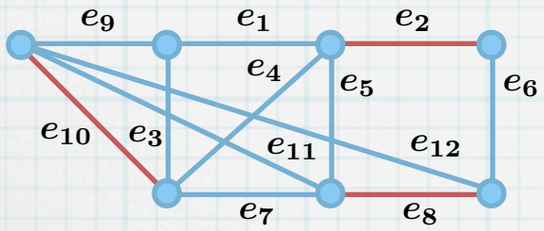
  1. Fall, v ist Endknoten einer Kante in $M_1 \setminus M_2$ und auch einer Kante in $M_2 \setminus M_1$.

     Dann ist $\deg_H(v) = 2$.

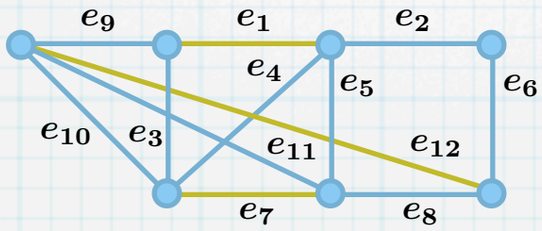
Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

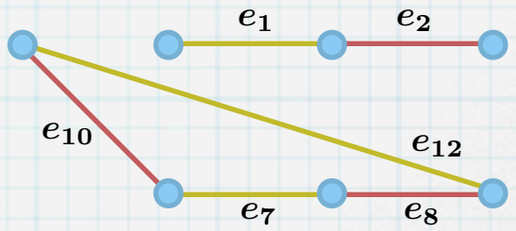
  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$



$M_2 := \{e_1, e_7, e_{12}\}$



$M_1 \triangle M_2 := \{e_1, e_2, e_7, e_8, e_{10}, e_{12}\}$

* Beweis (von Lemma 6):

  Sei v ein Knoten von H.

  Da M_1 Matching, gibt es höchstens eine Kante in M_1, die v als Endknoten hat.

  Ebenso gibt es höchstens eine Kante in M_2, die v als Endknoten hat.

  1. Fall, v ist Endknoten einer Kante in $M_1 \setminus M_2$ und auch einer Kante in $M_2 \setminus M_1$.

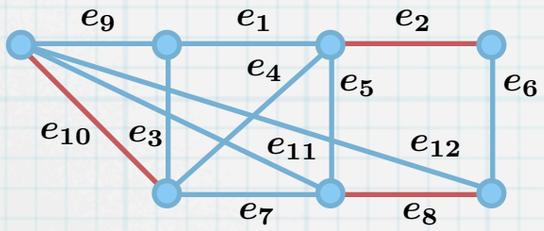
     Dann ist $\deg_H(v) = 2$.

  2. Fall, v ist Endknoten einer Kante in $M_1 \setminus M_2$ oder in $M_2 \setminus M_1$, aber nicht in beiden.

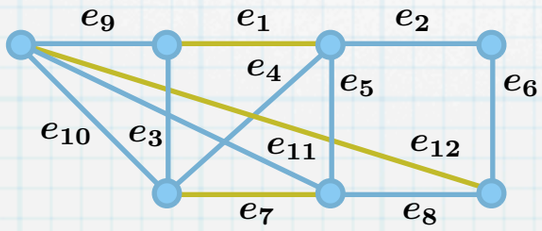
Lemma von der symmetrischen Differenz

* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

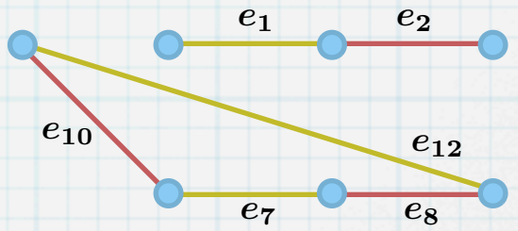
  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen, oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$



$M_2 := \{e_1, e_7, e_{12}\}$



$M_1 \triangle M_2 := \{e_1, e_2, e_7, e_8, e_{10}, e_{12}\}$

* Beweis (von Lemma 6):

  Sei v ein Knoten von H.

  Da M_1 Matching, gibt es höchstens eine Kante in M_1, die v als Endknoten hat.

  Ebenso gibt es höchstens eine Kante in M_2, die v als Endknoten hat.

  1. Fall, v ist Endknoten einer Kante in $M_1 \setminus M_2$ und auch einer Kante in $M_2 \setminus M_1$.

     Dann ist $\deg_H(v) = 2$.

  2. Fall, v ist Endknoten einer Kante in $M_1 \setminus M_2$ oder in $M_2 \setminus M_1$, aber nicht in beiden.

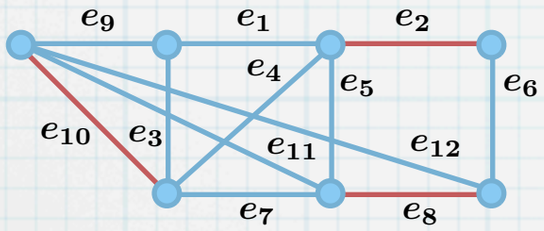
     Dann ist $\deg_H(v) = 1$.

Lemma von der symmetrischen Differenz

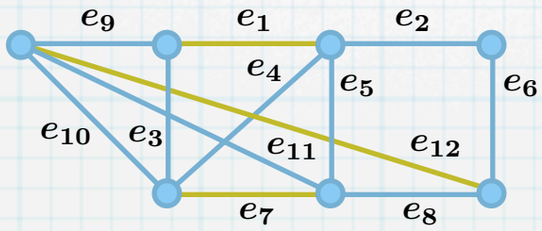
* **Lemma 6**:
  Seien M_1, M_2 zwei Matchings in einem schlichten Graphen G. Sei H der Untergraph von G, der durch die Kantenmenge $M_1 \triangle M_2 := (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$ gegeben ist, d.h. durch die symmetrische Differenz der beiden Matchings. Sei K eine Zusammenhangskomponente von H. Dann gilt:

  Entweder ist K ein Zyklus gerader Länge, dessen Kanten abwechselnd in M_1 und M_2 liegen,

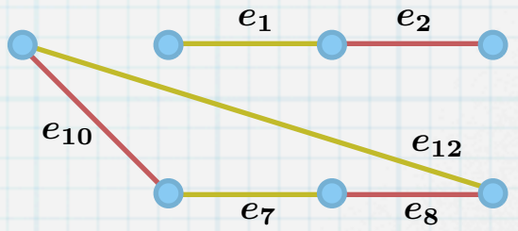
  oder K ist ein Weg, dessen Kanten abwechselnd in M_1 und M_2 liegen und dessen Endknoten in einem der beiden Matchings ungesättigt sind.
* Beispiel:



$M_1 := \{e_2, e_8, e_{10}\}$



$M_2 := \{e_1, e_7, e_{12}\}$



$M_1 \triangle M_2 := \{e_1, e_2, e_7, e_8, e_{10}, e_{12}\}$

* Beweis (von Lemma 6):

  Sei v ein Knoten von H.

  Da M_1 Matching, gibt es höchstens eine Kante in M_1, die v als Endknoten hat.

  Ebenso gibt es höchstens eine Kante in M_2, die v als Endknoten hat.

  1. Fall, v ist Endknoten einer Kante in $M_1 \setminus M_2$ und auch einer Kante in $M_2 \setminus M_1$.

     Dann ist $\deg_H(v) = 2$.

  2. Fall, v ist Endknoten einer Kante in $M_1 \setminus M_2$ oder in $M_2 \setminus M_1$, aber nicht in beiden.

     Dann ist $\deg_H(v) = 1$.

  Die Zusammenhangskomponente von v ist dann entweder ein Weg oder ein Zyklus.

Umfärbungen entlang von Wegen

Umfärbungen entlang von Wegen

* Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.

Umfärbungen entlang von Wegen

* Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
* Sei P ein M-alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.

Umfärbungen entlang von Wegen

* Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
* Sei P ein M-alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
* Ist P zudem M-erweiternd, dann sind die Endknoten ungesättigt.

Umfärbungen entlang von Wegen

- Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
- Sei P ein M-alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
- Ist P zudem M-erweiternd, dann sind die Endknoten ungesättigt.
- Also sind die erste und die letzte Kante von P blau.

Umfärbungen entlang von Wegen

* Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
* Sei P ein M-alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
* Ist P zudem M-erweiternd, dann sind die Endknoten ungesättigt.
* Also sind die erste und die letzte Kante von P blau.
* Damit ist P eine alternierende Folge der Form blau, rot, blau, ..., rot, blau.

Umfärbungen entlang von Wegen

* Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
* Sei P ein M-alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
* Ist P zudem M-erweiternd, dann sind die Endknoten ungesättigt.
* Also sind die erste und die letzte Kante von P blau.
* Damit ist P eine alternierende Folge der Form blau, rot, blau, ..., rot, blau.
* Somit hat P eine ungerade Anzahl von Kanten, $2m + 1$, davon m rote und $m + 1$ blaue.

Umfärbungen entlang von Wegen

* Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
* Sei P ein M-alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
* Ist P zudem M-erweiternd, dann sind die Endknoten ungesättigt.
* Also sind die erste und die letzte Kante von P blau.
* Damit ist P eine alternierende Folge der Form blau, rot, blau, ..., rot, blau.
* Somit hat P eine ungerade Anzahl von Kanten, $2m + 1$, davon m rote und $m + 1$ blaue.
* Wir definieren eine Kantenmenge M', welche aus den Kanten von M minus den roten Kanten von P plus den blauen Kanten von P besteht.

Umfärbungen entlang von Wegen

* Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
* Sei P ein M-alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
* Ist P zudem M-erweiternd, dann sind die Endknoten ungesättigt.
* Also sind die erste und die letzte Kante von P blau.
* Damit ist P eine alternierende Folge der Form blau, rot, blau, ..., rot, blau.
* Somit hat P eine ungerade Anzahl von Kanten, $2m + 1$, davon m rote und $m + 1$ blaue.
* Wir definieren eine Kantenmenge M', welche aus den Kanten von M minus den roten Kanten von P plus den blauen Kanten von P besteht.
* Die Endknoten von P sind nicht durch M gesättigt, und alle anderen Knoten von P sind bereits durch M gesättigt.

Umfärbungen entlang von Wegen

- Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
- Sei P ein M-alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
- Ist P zudem M-erweiternd, dann sind die Endknoten ungesättigt.
- Also sind die erste und die letzte Kante von P blau.
- Damit ist P eine alternierende Folge der Form blau, rot, blau, ..., rot, blau.
- Somit hat P eine ungerade Anzahl von Kanten, $2m+1$, davon m rote und $m+1$ blaue.
- Wir definieren eine Kantenmenge M', welche aus den Kanten von M minus den roten Kanten von P plus den blauen Kanten von P besteht.
- Die Endknoten von P sind nicht durch M gesättigt, und alle anderen Knoten von P sind bereits durch M gesättigt.
- Also ist M' wiederum ein Matching, welches eine Kante mehr als M hat.

Umfärbungen entlang von Wegen

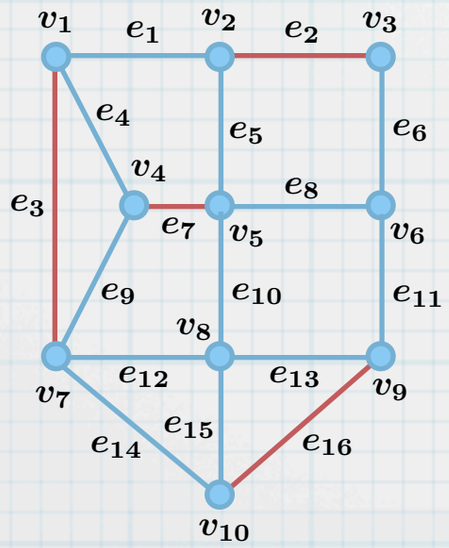
- Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
- Sei P ein M-alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
- Ist P zudem M-erweiternd, dann sind die Endknoten ungesättigt.
- Also sind die erste und die letzte Kante von P blau.
- Damit ist P eine alternierende Folge der Form blau, rot, blau, ..., rot, blau.
- Somit hat P eine ungerade Anzahl von Kanten, $2m + 1$, davon m rote und $m + 1$ blaue.
- Wir definieren eine Kantenmenge M', welche aus den Kanten von M minus den roten Kanten von P plus den blauen Kanten von P besteht.
- Die Endknoten von P sind nicht durch M gesättigt, und alle anderen Knoten von P sind bereits durch M gesättigt.
- Also ist M' wiederum ein Matching, welches eine Kante mehr als M hat.
- **Definition 7**:
  Die oben beschriebene Operation, die ein gegebenes Matching M in ein größeres Matching M' unter Verwendung eines M-erweiternden Weges P umwandelt, wird als **Umfärbung entlang des erweiternden Weges** P bezeichnet.

Umfärbungen entlang von Wegen

- Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
- Sei P ein M-alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
- Ist P zudem M-erweiternd, dann sind die Endknoten ungesättigt.
- Also sind die erste und die letzte Kante von P blau.
- Damit ist P eine alternierende Folge der Form blau, rot, blau, ..., rot, blau.
- Somit hat P eine ungerade Anzahl von Kanten, $2m + 1$, davon m rote und $m + 1$ blaue.
- Wir definieren eine Kantenmenge M', welche aus den Kanten von M minus den roten Kanten von P plus den blauen Kanten von P besteht.
- Die Endknoten von P sind nicht durch M gesättigt, und alle anderen Knoten von P sind bereits durch M gesättigt.
- Also ist M' wiederum ein Matching, welches eine Kante mehr als M hat.
- **Definition 7**:
  Die oben beschriebene Operation, die ein gegebenes Matching M in ein größeres Matching M' unter Verwendung eines M-erweiternden Weges P umwandelt, wird als **Umfärbung entlang des erweiternden Weges** P bezeichnet.
- Beispiel:

Umfärbungen entlang von Wegen

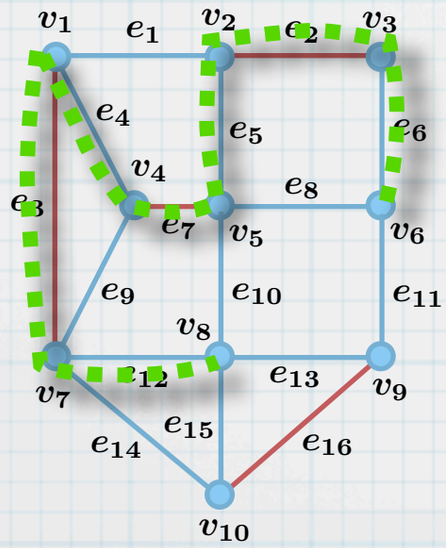
- Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
- Sei P ein M–alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
- Ist P zudem M–erweiternd, dann sind die Endknoten ungesättigt.
- Also sind die erste und die letzte Kante von P blau.
- Damit ist P eine alternierende Folge der Form blau, rot, blau, ..., rot, blau.
- Somit hat P eine ungerade Anzahl von Kanten, $2m + 1$, davon m rote und $m + 1$ blaue.
- Wir definieren eine Kantenmenge M', welche aus den Kanten von M minus den roten Kanten von P plus den blauen Kanten von P besteht.
- Die Endknoten von P sind nicht durch M gesättigt, und alle anderen Knoten von P sind bereits durch M gesättigt.
- Also ist M' wiederum ein Matching, welches eine Kante mehr als M hat.
- **Definition 7**:
  Die oben beschriebene Operation, die ein gegebenes Matching M in ein größeres Matching M' unter Verwendung eines M–erweiternden Weges P umwandelt, wird als **Umfärbung entlang des erweiternden Weges** P bezeichnet.
- Beispiel:



$$M = \{e_2, e_3, e_7, e_{16}\}$$

Umfärbungen entlang von Wegen

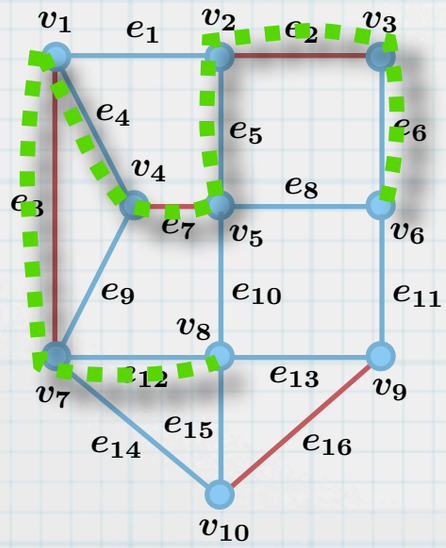
- Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
- Sei P ein M–alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
- Ist P zudem M–erweiternd, dann sind die Endknoten ungesättigt.
- Also sind die erste und die letzte Kante von P blau.
- Damit ist P eine alternierende Folge der Form blau, rot, blau, ..., rot, blau.
- Somit hat P eine ungerade Anzahl von Kanten, $2m + 1$, davon m rote und $m + 1$ blaue.
- Wir definieren eine Kantenmenge M', welche aus den Kanten von M minus den roten Kanten von P plus den blauen Kanten von P besteht.
- Die Endknoten von P sind nicht durch M gesättigt, und alle anderen Knoten von P sind bereits durch M gesättigt.
- Also ist M' wiederum ein Matching, welches eine Kante mehr als M hat.
- **Definition 7**:
  Die oben beschriebene Operation, die ein gegebenes Matching M in ein größeres Matching M' unter Verwendung eines M–erweiternden Weges P umwandelt, wird als **Umfärbung entlang des erweiternden Weges** P bezeichnet.
- Beispiel:



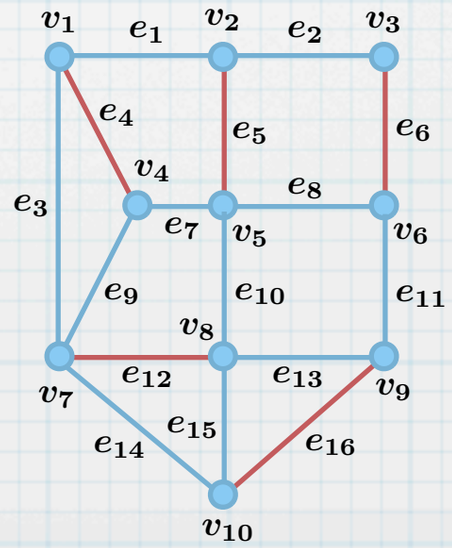
$$M = \{e_2, e_3, e_7, e_{16}\}$$
$$P = (e_6, e_2, e_5, e_7, e_4, e_3, e_{12})$$

Umfärbungen entlang von Wegen

* Sei M ein Matching von G. Wir bezeichnen die Kanten in M als „rot“, alle anderen als „blau“.
* Sei P ein M–alternierender Weg in G, so dass die Kanten abwechselnd rot und blau sind.
* Ist P zudem M–erweiternd, dann sind die Endknoten ungesättigt.
* Also sind die erste und die letzte Kante von P blau.
* Damit ist P eine alternierende Folge der Form blau, rot, blau, ..., rot, blau.
* Somit hat P eine ungerade Anzahl von Kanten, $2m+1$, davon m rote und $m+1$ blaue.
* Wir definieren eine Kantenmenge M', welche aus den Kanten von M minus den roten Kanten von P plus den blauen Kanten von P besteht.
* Die Endknoten von P sind nicht durch M gesättigt, und alle anderen Knoten von P sind bereits durch M gesättigt.
* Also ist M' wiederum ein Matching, welches eine Kante mehr als M hat.
* **Definition 7**:
Die oben beschriebene Operation, die ein gegebenes Matching M in ein größeres Matching M' unter Verwendung eines M–erweiternden Weges P umwandelt, wird als **Umfärbung entlang des erweiternden Weges** P bezeichnet.
* Beispiel:



$$M = \{e_2, e_3, e_7, e_{16}\}$$
$$P = (e_6, e_2, e_5, e_7, e_4, e_3, e_{12})$$



$$M' = \{e_4, e_5, e_6, e_{12}, e_{16}\}$$

Satz von Berge

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

  Sei H der Untergraph von G mit Kantenmenge $M \triangle M' = (M \backslash M') \cup (M' \backslash M)$.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

  Sei H der Untergraph von G mit Kantenmenge $M \triangle M' = (M \backslash M') \cup (M' \backslash M)$.

  Für die Zusammenhangskomponenten von H gilt nach Lemma 6:

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

  Sei H der Untergraph von G mit Kantenmenge $M \triangle M' = (M \backslash M') \cup (M' \backslash M)$.

  Für die Zusammenhangskomponenten von H gilt nach Lemma 6:

  Entweder sind es gerade Zyklen mit Kanten abwechselnd in M und M', oder

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

  Sei H der Untergraph von G mit Kantenmenge $M \triangle M' = (M \backslash M') \cup (M' \backslash M)$.

  Für die Zusammenhangskomponenten von H gilt nach Lemma 6:

  Entweder sind es gerade Zyklen mit Kanten abwechselnd in M und M', oder

  es sind Wege mit Kanten abwechselnd in M und M', deren Endknoten in einem der beiden Matchings ungesättigt sind.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

  Sei H der Untergraph von G mit Kantenmenge $M \triangle M' = (M \backslash M') \cup (M' \backslash M)$.

  Für die Zusammenhangskomponenten von H gilt nach Lemma 6:

  Entweder sind es gerade Zyklen mit Kanten abwechselnd in M und M', oder

  es sind Wege mit Kanten abwechselnd in M und M', deren Endknoten in einem der beiden Matchings ungesättigt sind.

  Wenn eine Komponente ein Weg ist, dann ist dieser gerade.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

  Sei H der Untergraph von G mit Kantenmenge $M \triangle M' = (M \backslash M') \cup (M' \backslash M)$.

  Für die Zusammenhangskomponenten von H gilt nach Lemma 6:

  Entweder sind es gerade Zyklen mit Kanten abwechselnd in M und M', oder

  es sind Wege mit Kanten abwechselnd in M und M', deren Endknoten in einem der beiden Matchings ungesättigt sind.

  Wenn eine Komponente ein Weg ist, dann ist dieser gerade.

  Ein Weg ungerader Länge hätte Endknoten, die im selben Matching M (oder M') ungesättigt sind. Dann gibt es einen M– (oder M'–) erweiternden Weg.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

  Sei H der Untergraph von G mit Kantenmenge $M \triangle M' = (M \backslash M') \cup (M' \backslash M)$.

  Für die Zusammenhangskomponenten von H gilt nach Lemma 6:

  Entweder sind es gerade Zyklen mit Kanten abwechselnd in M und M', oder

  es sind Wege mit Kanten abwechselnd in M und M', deren Endknoten in einem der beiden Matchings ungesättigt sind.

  Wenn eine Komponente ein Weg ist, dann ist dieser gerade.

  Ein Weg ungerader Länge hätte Endknoten, die im selben Matching M (oder M') ungesättigt sind. Dann gibt es einen M- (oder M'-) erweiternden Weg.

  M-erweiternde Wege gibt es nicht (nach Voraussetzung).

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

  Sei H der Untergraph von G mit Kantenmenge $M \triangle M' = (M \backslash M') \cup (M' \backslash M)$.

  Für die Zusammenhangskomponenten von H gilt nach Lemma 6:

  Entweder sind es gerade Zyklen mit Kanten abwechselnd in M und M', oder

  es sind Wege mit Kanten abwechselnd in M und M', deren Endknoten in einem der beiden Matchings ungesättigt sind.

  Wenn eine Komponente ein Weg ist, dann ist dieser gerade.

  Ein Weg ungerader Länge hätte Endknoten, die im selben Matching M (oder M') ungesättigt sind. Dann gibt es einen M- (oder M'-) erweiternden Weg.

  M-erweiternde Wege gibt es nicht (nach Voraussetzung).

  M'-erweiternde Wege gibt es nicht (da M' maximal, siehe $\Rightarrow$-Teil des Beweises).

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

  Sei H der Untergraph von G mit Kantenmenge $M \triangle M' = (M \backslash M') \cup (M' \backslash M)$.

  Für die Zusammenhangskomponenten von H gilt nach Lemma 6:

  Entweder sind es gerade Zyklen mit Kanten abwechselnd in M und M', oder

  es sind Wege mit Kanten abwechselnd in M und M', deren Endknoten in einem der beiden Matchings ungesättigt sind.

  Wenn eine Komponente ein Weg ist, dann ist dieser gerade.

  Ein Weg ungerader Länge hätte Endknoten, die im selben Matching M (oder M') ungesättigt sind. Dann gibt es einen M- (oder M'-) erweiternden Weg.

  M-erweiternde Wege gibt es nicht (nach Voraussetzung).

  M'-erweiternde Wege gibt es nicht (da M' maximal, siehe $\Rightarrow$-Teil des Beweises).

  Also sind alle Komponenten von H gerade Wege oder gerade Zyklen, welche die selbe Zahl von Kanten in M wie in M' haben.

Satz von Berge

* **Satz 8** (Berge, 1957):
  Ein Matching M in einem Graphen G ist genau dann ein maximales Matching, wenn G keinen M-erweiternden Weg enthält.
* Beweis:

  ($\Rightarrow$) Sei M maximales Matching von G und sei P ein M-erweiternder Weg.

  Durch Umfärbung entlang P entsteht ein Matching M' von G, welches eine Kante mehr enthält als M, im Widerspruch zur Maximalität von M.

  ($\Leftarrow$) Sei M ein Matching von G, so dass es keinen M-erweiternden Weg gibt.

  Sei M' ein maximales Matching von G. Zu zeigen ist: $|M| = |M'|$.

  Es ist $|M| = |M \backslash M'| + |M \cap M'|$ und $|M'| = |M' \backslash M| + |M' \cap M|$.

  Also bleibt zu zeigen: $|M \backslash M'| = |M' \backslash M|$.

  Sei H der Untergraph von G mit Kantenmenge $M \triangle M' = (M \backslash M') \cup (M' \backslash M)$.

  Für die Zusammenhangskomponenten von H gilt nach Lemma 6:

  Entweder sind es gerade Zyklen mit Kanten abwechselnd in M und M', oder

  es sind Wege mit Kanten abwechselnd in M und M', deren Endknoten in einem der beiden Matchings ungesättigt sind.

  Wenn eine Komponente ein Weg ist, dann ist dieser gerade.

  Ein Weg ungerader Länge hätte Endknoten, die im selben Matching M (oder M') ungesättigt sind. Dann gibt es einen M- (oder M'-) erweiternden Weg.

  M-erweiternde Wege gibt es nicht (nach Voraussetzung).

  M'-erweiternde Wege gibt es nicht (da M' maximal, siehe $\Rightarrow$-Teil des Beweises).

  Also sind alle Komponenten von H gerade Wege oder gerade Zyklen, welche die selbe Zahl von Kanten in M wie in M' haben.

  Somit ist $|M \backslash M'| = |M' \backslash M|$.

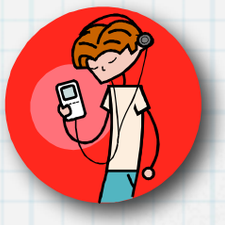
Das Heiratsproblem

Das Heiratsproblem

* Gegeben sei eine endliche Menge Jungen.

Das Heiratsproblem

* Gegeben sei eine endliche Menge Jungen.

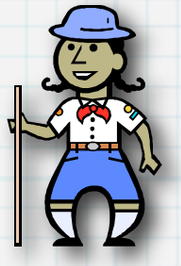


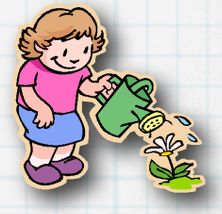
Das Heiratsproblem

* Gegeben sei eine endliche Menge Jungen.
* Jeder Junge hat mehrere Freundinnen.

Das Heiratsproblem

* Gegeben sei eine endliche Menge Jungen.
* Jeder Junge hat mehrere Freundinnen.









Das Heiratsproblem

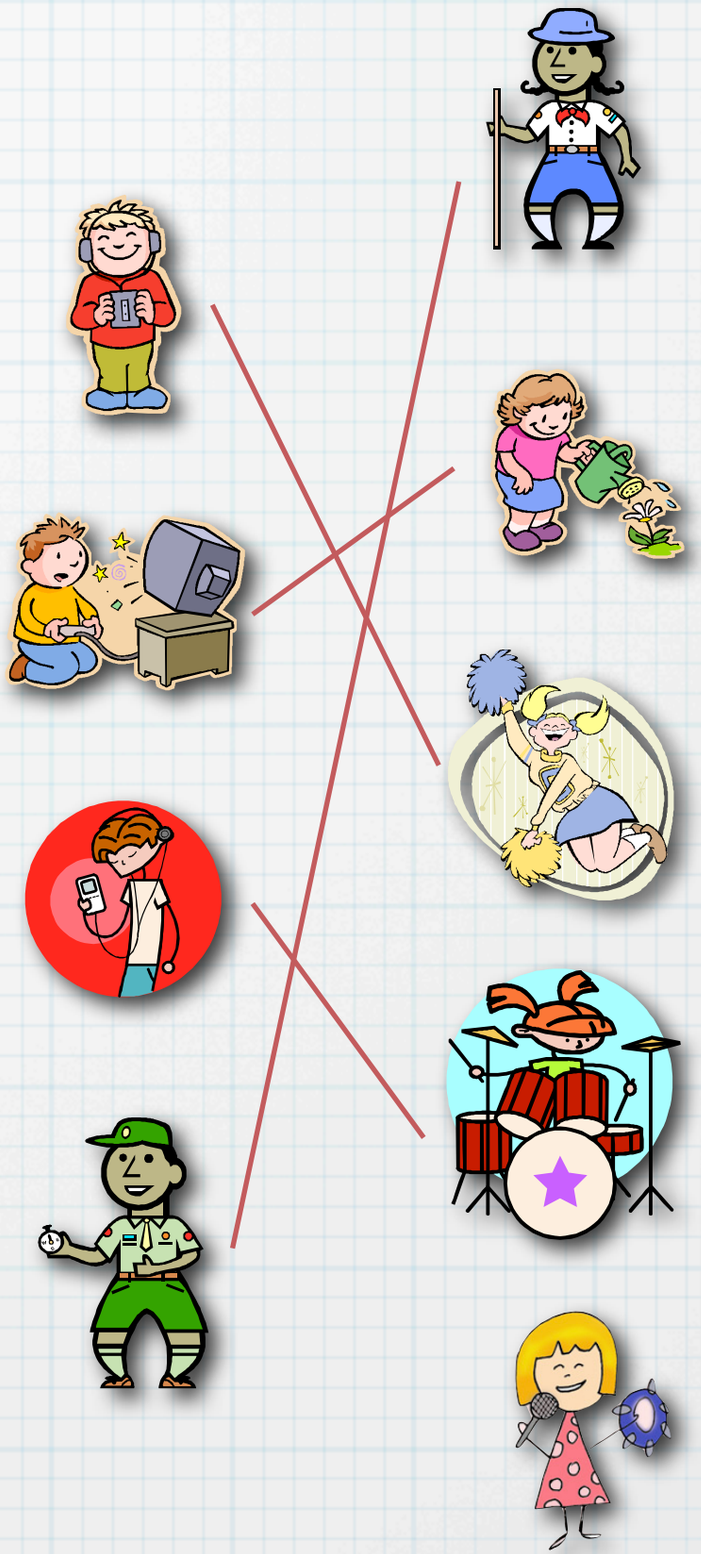
* Gegeben sei eine endliche Menge Jungen.
* Jeder Junge hat mehrere Freundinnen.

Das Heiratsproblem

- Gegeben sei eine endliche Menge Jungen.
- Jeder Junge hat mehrere Freundinnen.
- Können alle Jungen so verheiratet werden,
  dass jeder Junge (genau) eine Freundin heiraten kann?

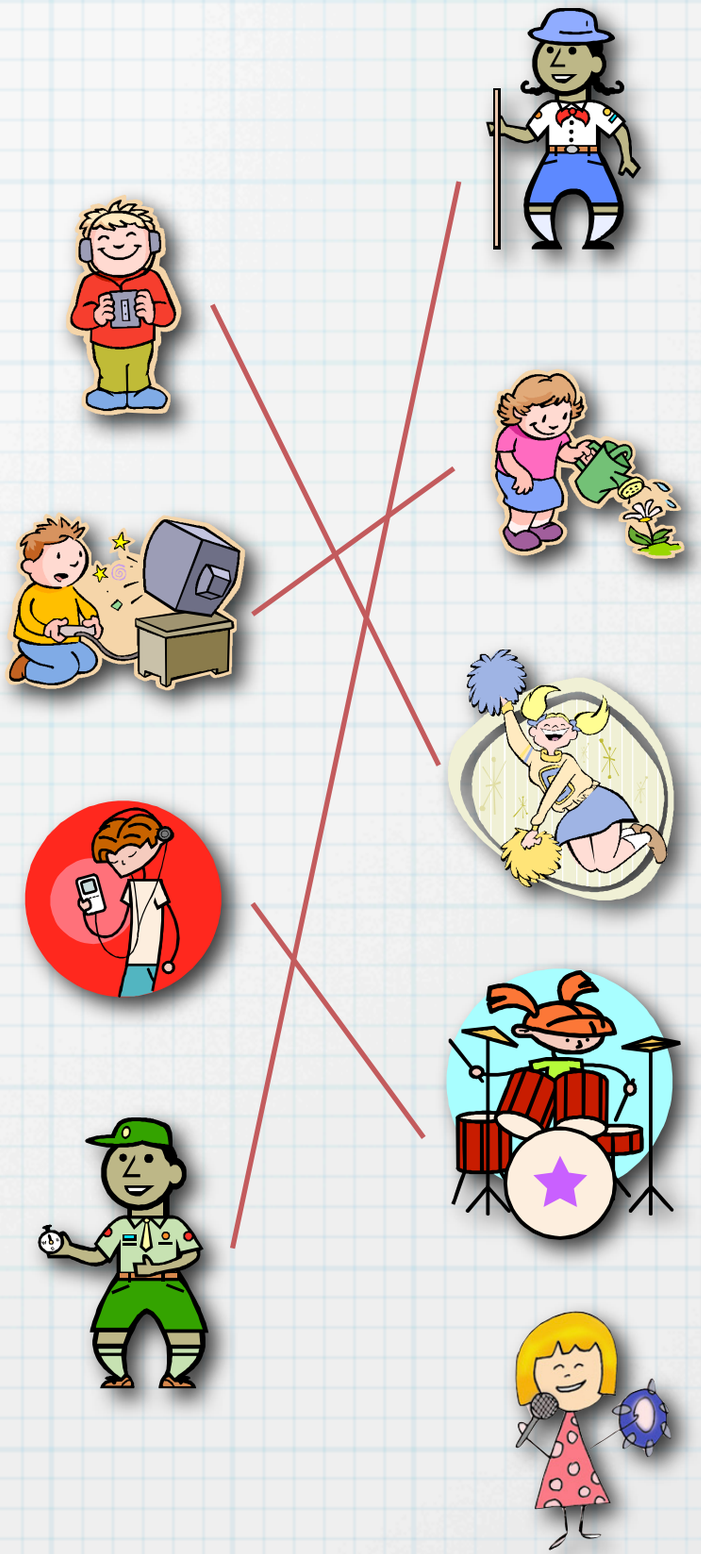
Das Heiratsproblem

- Gegeben sei eine endliche Menge Jungen.
- Jeder Junge hat mehrere Freundinnen.
- Können alle Jungen so verheiratet werden,
  dass jeder Junge (genau) eine Freundin heiraten kann?



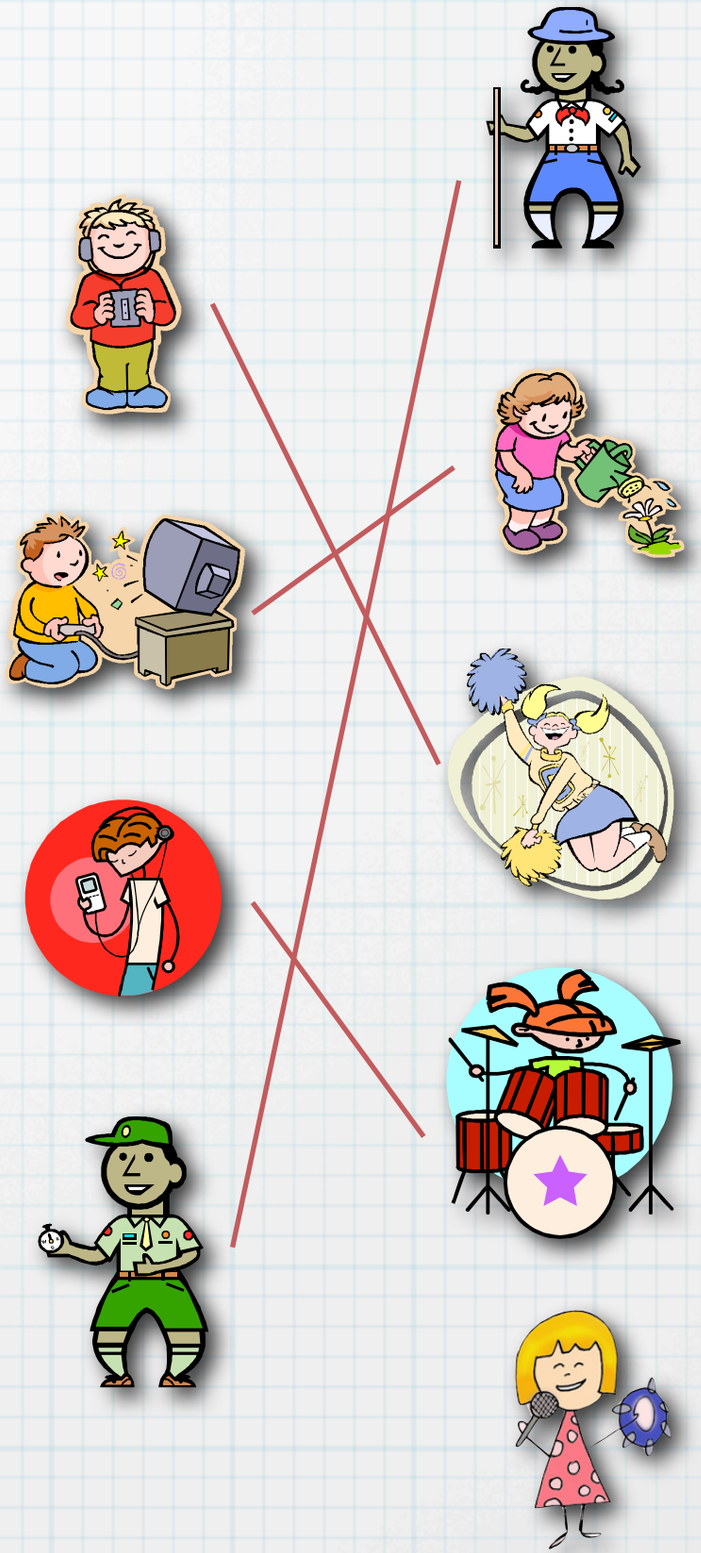
Das Heiratsproblem

- Gegeben sei eine endliche Menge Jungen.
- Jeder Junge hat mehrere Freundinnen.
- Können alle Jungen so verheiratet werden,
  dass jeder Junge (genau) eine Freundin heiraten kann?
- Als graphentheoretisches Problem formuliert:



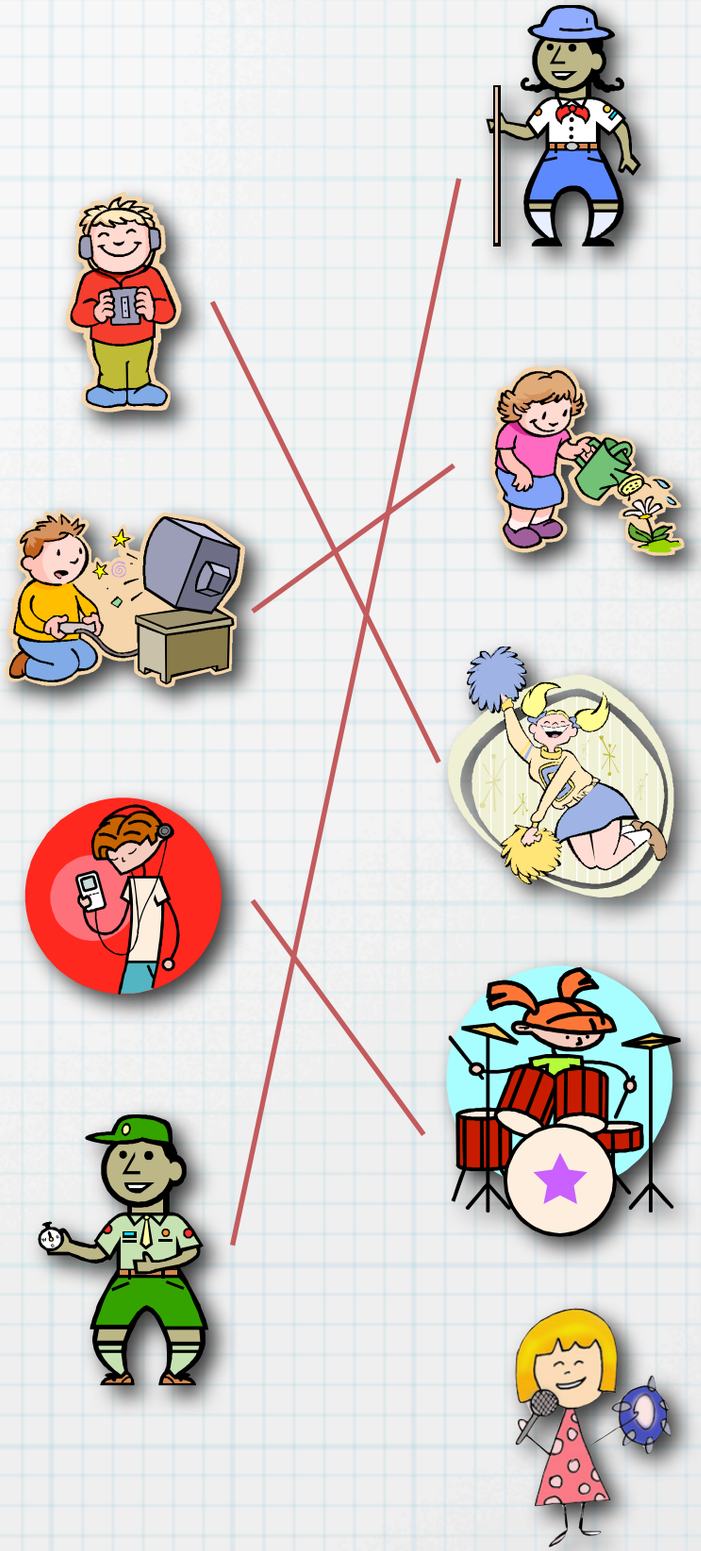
Das Heiratsproblem

- Gegeben sei eine endliche Menge Jungen.
- Jeder Junge hat mehrere Freundinnen.
- Können alle Jungen so verheiratet werden,
  dass jeder Junge (genau) eine Freundin heiraten kann?
- Als graphentheoretisches Problem formuliert:
  - Gegeben sei ein bipartiter Graph $G = (V, E)$ mit Bipartition $V = X \cup Y$.



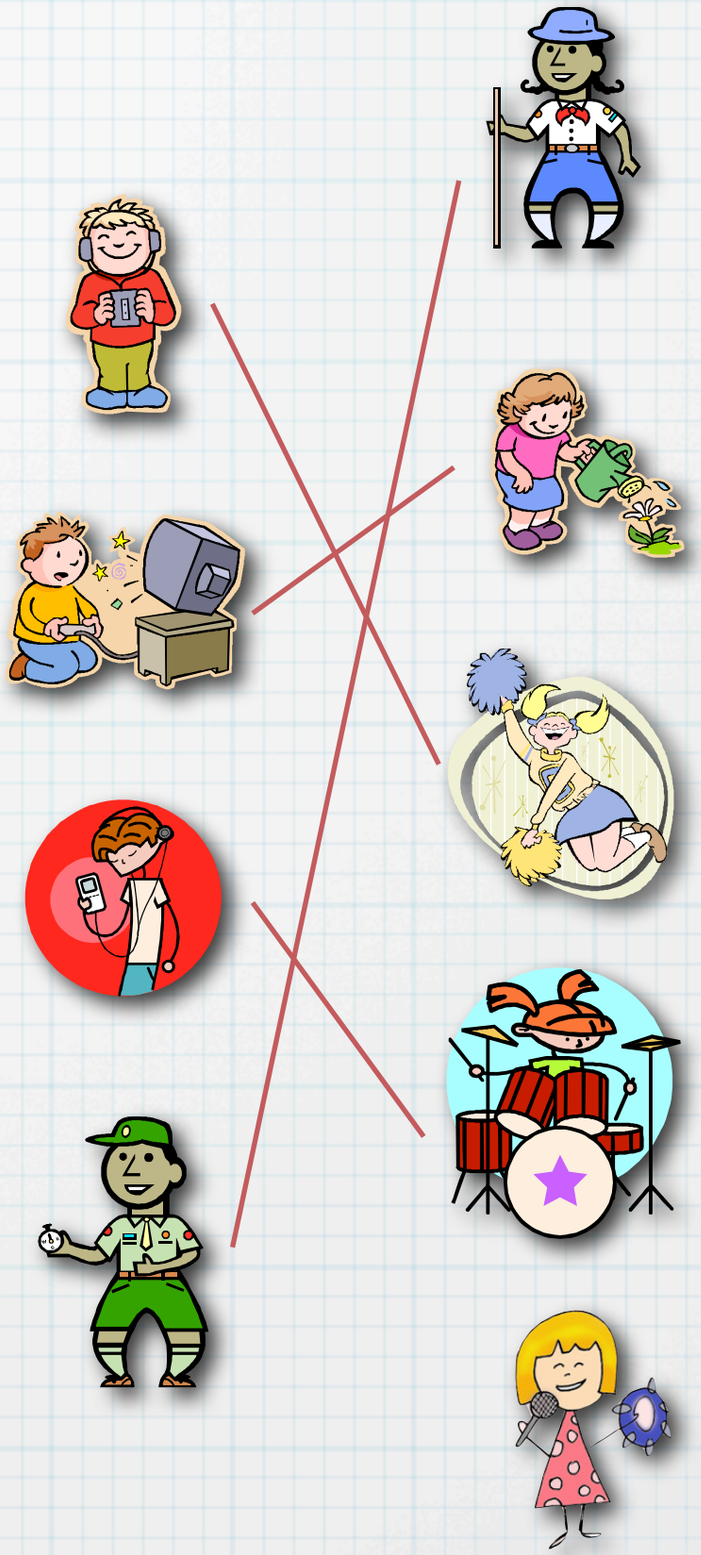
Das Heiratsproblem

- Gegeben sei eine endliche Menge Jungen.
- Jeder Junge hat mehrere Freundinnen.
- Können alle Jungen so verheiratet werden, dass jeder Junge (genau) eine Freundin heiraten kann?
- Als graphentheoretisches Problem formuliert:
  - Gegeben sei ein bipartiter Graph $G = (V, E)$ mit Bipartition $V = X \cup Y$.
  - Welche Bedingungen sind notwendig bzw. hinreichend, damit ein Matching in G existiert, welches jeden Knoten von X sättigt.



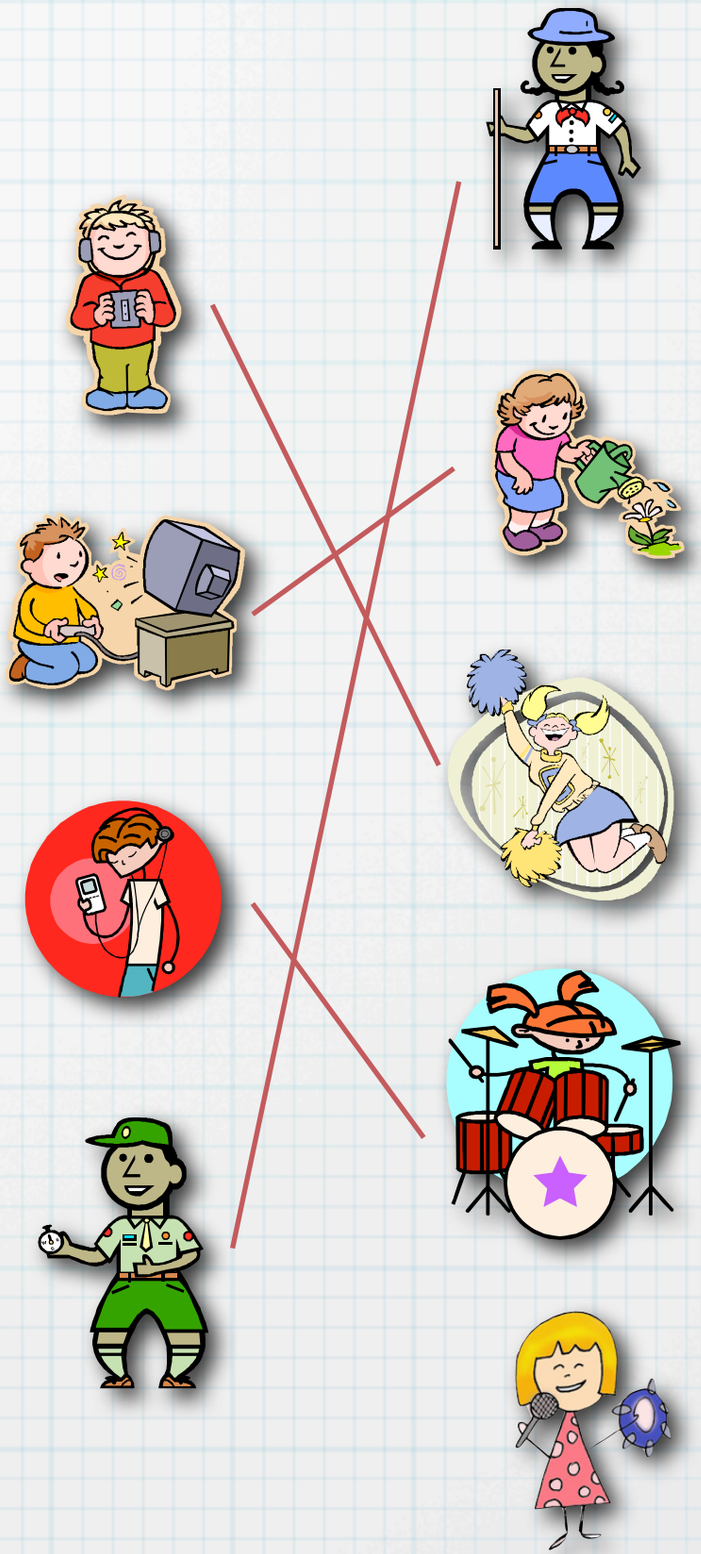
Das Heiratsproblem

- Gegeben sei eine endliche Menge Jungen.
- Jeder Junge hat mehrere Freundinnen.
- Können alle Jungen so verheiratet werden,
  dass jeder Junge (genau) eine Freundin heiraten kann?
- Als graphentheoretisches Problem formuliert:
  - Gegeben sei ein bipartiter Graph $G = (V, E)$ mit Bipartition $V = X \cup Y$.
  - Welche Bedingungen sind notwendig bzw. hinreichend, damit ein Matching in G existiert, welches jeden Knoten von X sättigt.
- Philip Hall (engl. Mathematiker, 1904–1982) zeigte, dass das Heiratsproblem für n Jungen genau dann lösbar ist, wenn für jedes k, $1 \leq k \leq n$, jede Menge von k Jungen zusammen mindestens k Freundinnen haben.



Das Heiratsproblem

- Gegeben sei eine endliche Menge Jungen.
- Jeder Junge hat mehrere Freundinnen.
- Können alle Jungen so verheiratet werden, dass jeder Junge (genau) eine Freundin heiraten kann?
- Als graphentheoretisches Problem formuliert:
  - Gegeben sei ein bipartiter Graph $G = (V, E)$ mit Bipartition $V = X \cup Y$.
  - Welche Bedingungen sind notwendig bzw. hinreichend, damit ein Matching in G existiert, welches jeden Knoten von X sättigt.
- Philip Hall (engl. Mathematiker, 1904–1982) zeigte, dass das Heiratsproblem für n Jungen genau dann lösbar ist, wenn für jedes k, $1 \leq k \leq n$, jede Menge von k Jungen zusammen mindestens k Freundinnen haben.
- **Definition 9**:
  Sei S eine Knotenmenge in G. Die benachbarte Menge $N(S)$ von S in G ist die Menge aller Knoten, die adjazent zu Knoten in S ist.



Der Heiratssatz von Hall

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

  Sei M^* ein maximales Matching von G. Auch durch M^* wird nicht ganz X gesättigt.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

  Sei M^* ein maximales Matching von G. Auch durch M^* wird nicht ganz X gesättigt.

  Sei x_0 ein Knoten, der nicht durch M^* gesättigt wird.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

  Sei M^* ein maximales Matching von G. Auch durch M^* wird nicht ganz X gesättigt.

  Sei x_0 ein Knoten, der nicht durch M^* gesättigt wird.

  Da M^* maximal, gibt es nach dem Satz von Berge keinen M^*-erweiternden Weg in G.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

  Sei M^* ein maximales Matching von G. Auch durch M^* wird nicht ganz X gesättigt.

  Sei x_0 ein Knoten, der nicht durch M^* gesättigt wird.

  Da M^* maximal, gibt es nach dem Satz von Berge keinen M^*-erweiternden Weg in G.

  Wenn also P ein M^*-alternierender Weg in G ist, dessen Endknoten x_0 ist, dann muss das andere Ende von P ein M^*-gesättigter Knoten sein.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

  Sei M^* ein maximales Matching von G. Auch durch M^* wird nicht ganz X gesättigt.

  Sei x_0 ein Knoten, der nicht durch M^* gesättigt wird.

  Da M^* maximal, gibt es nach dem Satz von Berge keinen M^*–erweiternden Weg in G.

  Wenn also P ein M^*–alternierender Weg in G ist, dessen Endknoten x_0 ist, dann muss das andere Ende von P ein M^*–gesättigter Knoten sein.

  Es ist entweder $P = (x_0, y_1, x_1, y_2, x_2, \ldots, y_n, x_n)$ oder $P = (x_0, y_1, x_1, \ldots, x_{n-1}, y_n)$, mit $x_i \in X, y_i \in Y$ (G bipart.), $\{y_1, x_1\}, \{y_2, x_2\}, \ldots \in M^*, \{x_0, y_1\}, \{x_1, y_2\}, \ldots \notin M^*$.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

  Sei M^* ein maximales Matching von G. Auch durch M^* wird nicht ganz X gesättigt.

  Sei x_0 ein Knoten, der nicht durch M^* gesättigt wird.

  Da M^* maximal, gibt es nach dem Satz von Berge keinen M^*–erweiternden Weg in G.

  Wenn also P ein M^*–alternierender Weg in G ist, dessen Endknoten x_0 ist, dann muss das andere Ende von P ein M^*–gesättigter Knoten sein.

  Es ist entweder $P = (x_0, y_1, x_1, y_2, x_2, \ldots, y_n, x_n)$ oder $P = (x_0, y_1, x_1, \ldots, x_{n-1}, y_n)$, mit $x_i \in X, y_i \in Y$ (G bipart.), $\{y_1, x_1\}, \{y_2, x_2\}, \ldots \in M^*, \{x_0, y_1\}, \{x_1, y_2\}, \ldots \notin M^*$.

  Im zweiten Fall wird y_n durch M^* gesättigt. Aber $\{x_{n-1}, y_n\}$ ist keine Kante in M^*.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

  Sei M^* ein maximales Matching von G. Auch durch M^* wird nicht ganz X gesättigt.

  Sei x_0 ein Knoten, der nicht durch M^* gesättigt wird.

  Da M^* maximal, gibt es nach dem Satz von Berge keinen M^*–erweiternden Weg in G.

  Wenn also P ein M^*–alternierender Weg in G ist, dessen Endknoten x_0 ist, dann muss das andere Ende von P ein M^*–gesättigter Knoten sein.

  Es ist entweder $P = (x_0, y_1, x_1, y_2, x_2, \ldots, y_n, x_n)$ oder $P = (x_0, y_1, x_1, \ldots, x_{n-1}, y_n)$, mit $x_i \in X, y_i \in Y$ (G bipart.), $\{y_1, x_1\}, \{y_2, x_2\}, \ldots \in M^*, \{x_0, y_1\}, \{x_1, y_2\}, \ldots \notin M^*$.

  Im zweiten Fall wird y_n durch M^* gesättigt. Aber $\{x_{n-1}, y_n\}$ ist keine Kante in M^*.

  Also gibt es noch eine weitere Kante $\{y_n, x_n\}$, um die P verlängert werden kann.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

  Sei M^* ein maximales Matching von G. Auch durch M^* wird nicht ganz X gesättigt.

  Sei x_0 ein Knoten, der nicht durch M^* gesättigt wird.

  Da M^* maximal, gibt es nach dem Satz von Berge keinen M^*–erweiternden Weg in G.

  Wenn also P ein M^*–alternierender Weg in G ist, dessen Endknoten x_0 ist, dann muss das andere Ende von P ein M^*–gesättigter Knoten sein.

  Es ist entweder $P = (x_0, y_1, x_1, y_2, x_2, \ldots, y_n, x_n)$ oder $P = (x_0, y_1, x_1, \ldots, x_{n-1}, y_n)$, mit $x_i \in X, y_i \in Y$ (G bipart.), $\{y_1, x_1\}, \{y_2, x_2\}, \ldots \in M^*, \{x_0, y_1\}, \{x_1, y_2\}, \ldots \notin M^*$.

  Im zweiten Fall wird y_n durch M^* gesättigt. Aber $\{x_{n-1}, y_n\}$ ist keine Kante in M^*.

  Also gibt es noch eine weitere Kante $\{y_n, x_n\}$, um die P verlängert werden kann.

  Somit kommt o.B.d.A. nur der erste (linke) Fall für P vor.

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

  Sei M^* ein maximales Matching von G. Auch durch M^* wird nicht ganz X gesättigt.

  Sei x_0 ein Knoten, der nicht durch M^* gesättigt wird.

  Da M^* maximal, gibt es nach dem Satz von Berge keinen M^*–erweiternden Weg in G.

  Wenn also P ein M^*–alternierender Weg in G ist, dessen Endknoten x_0 ist, dann muss das andere Ende von P ein M^*–gesättigter Knoten sein.

  Es ist entweder $P = (x_0, y_1, x_1, y_2, x_2, \ldots, y_n, x_n)$ oder $P = (x_0, y_1, x_1, \ldots, x_{n-1}, y_n)$, mit $x_i \in X, y_i \in Y$ (G bipart.), $\{y_1, x_1\}, \{y_2, x_2\}, \ldots \in M^*, \{x_0, y_1\}, \{x_1, y_2\}, \ldots \notin M^*$.

  Im zweiten Fall wird y_n durch M^* gesättigt. Aber $\{x_{n-1}, y_n\}$ ist keine Kante in M^*.

  Also gibt es noch eine weitere Kante $\{y_n, x_n\}$, um die P verlängert werden kann.

  Somit kommt o.B.d.A. nur der erste (linke) Fall für P vor.

  Seien S' bzw. T die Menge aller Knoten in X bzw. Y, die durch einen M^*–alternierenden Weg von x_0 erreichbar sind (wobei wir x_0 aus S' ausschließen).

Der Heiratssatz von Hall

* **Satz 10** (Hall, 1935):
  Sei $G = (V, E)$ ein bipartiter Graph mit $V = X \cup Y$. G hat genau dann ein Matching, welches jeden Knoten von X sättigt, wenn $|N(S)| \geq |S|$ für alle Teilmengen S von X.
* Beweis:

  ($\Rightarrow$) Sei M ein Matching, welches jeden Knoten in X sättigt.

  $S \subseteq X$ kann nur dann gesättigt werden, wenn sie mindestens $|S|$ Nachbarn hat.

  Also ist $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  ($\Leftarrow$) Sei $|N(S)| \geq |S|$ für alle Teilmengen S von X.

  Angenommen, es gibt kein Matching, welches jeden Knoten von X sättigt.

  Sei M^* ein maximales Matching von G. Auch durch M^* wird nicht ganz X gesättigt.

  Sei x_0 ein Knoten, der nicht durch M^* gesättigt wird.

  Da M^* maximal, gibt es nach dem Satz von Berge keinen M^*-erweiternden Weg in G.

  Wenn also P ein M^*-alternierender Weg in G ist, dessen Endknoten x_0 ist, dann muss das andere Ende von P ein M^*-gesättigter Knoten sein.

  Es ist entweder $P = (x_0, y_1, x_1, y_2, x_2, \ldots, y_n, x_n)$ oder $P = (x_0, y_1, x_1, \ldots, x_{n-1}, y_n)$, mit $x_i \in X, y_i \in Y$ (G bipart.), $\{y_1, x_1\}, \{y_2, x_2\}, \ldots \in M^*, \{x_0, y_1\}, \{x_1, y_2\}, \ldots \notin M^*$.

  Im zweiten Fall wird y_n durch M^* gesättigt. Aber $\{x_{n-1}, y_n\}$ ist keine Kante in M^*.

  Also gibt es noch eine weitere Kante $\{y_n, x_n\}$, um die P verlängert werden kann.

  Somit kommt o.B.d.A. nur der erste (linke) Fall für P vor.

  Seien S' bzw. T die Menge aller Knoten in X bzw. Y, die durch einen M^*-alternierenden Weg von x_0 erreichbar sind (wobei wir x_0 aus S' ausschließen).

  Da jeder Knoten in X mit einem Knoten in Y gepaart ist, gilt $|S'| = |T|$.

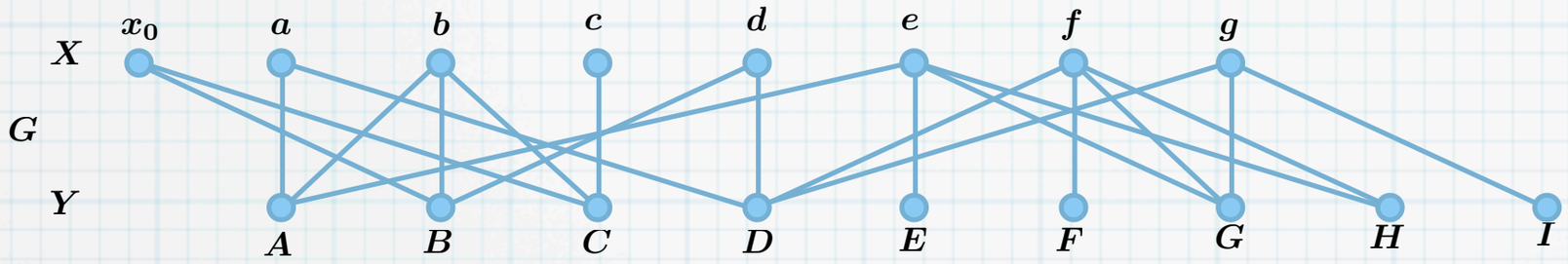
Der Heiratssatz von Hall (Forts.)

Der Heiratssatz von Hall (Forts.)

* Beispiel:

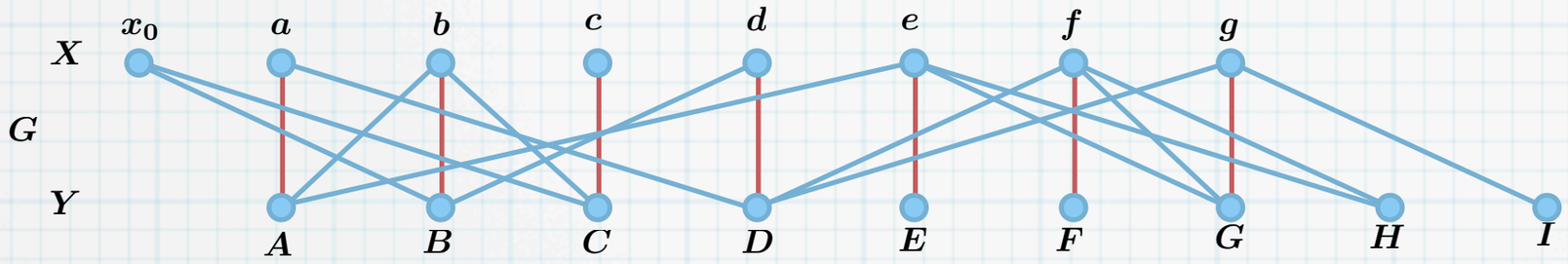
Der Heiratssatz von Hall (Forts.)

* Beispiel:



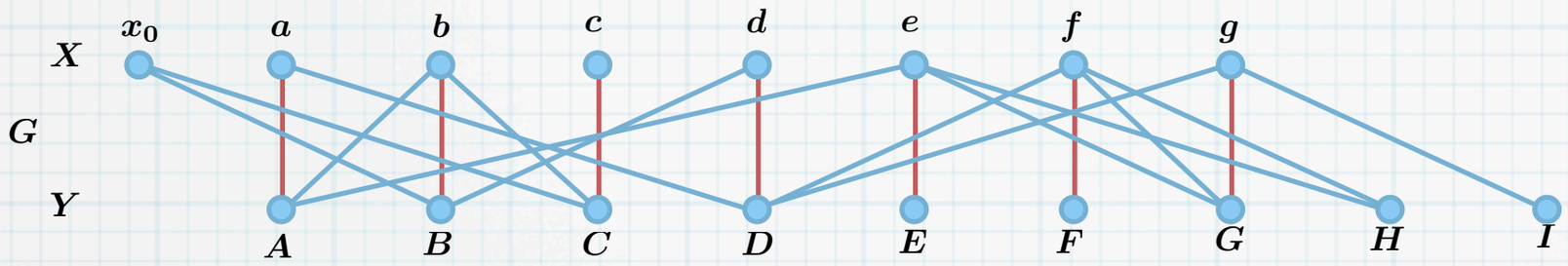
Der Heiratssatz von Hall (Forts.)

* Beispiel:



Der Heiratssatz von Hall (Forts.)

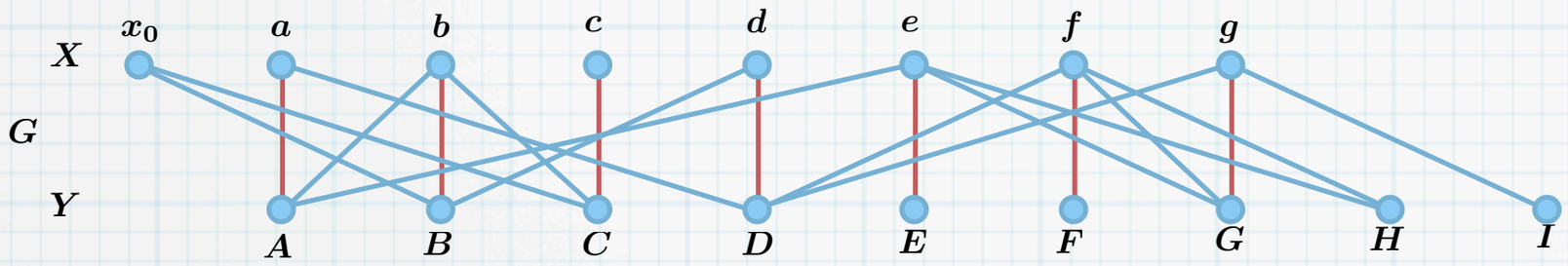
* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$

Der Heiratssatz von Hall (Forts.)

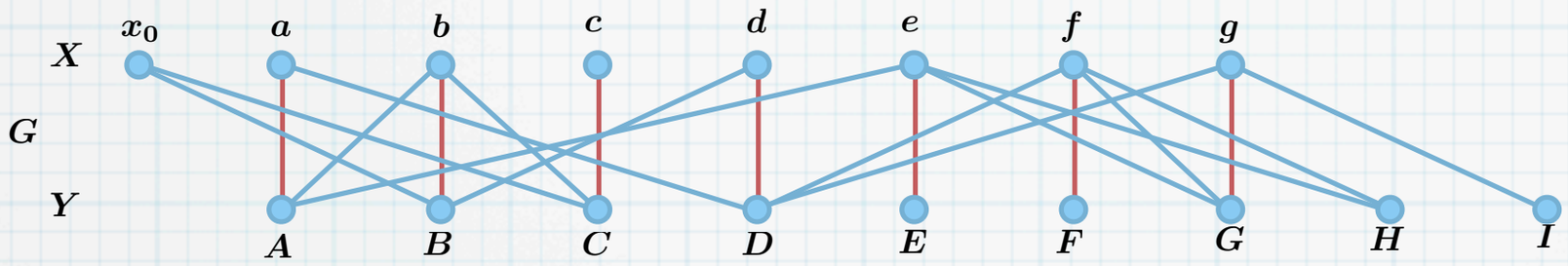
* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$

Der Heiratssatz von Hall (Forts.)

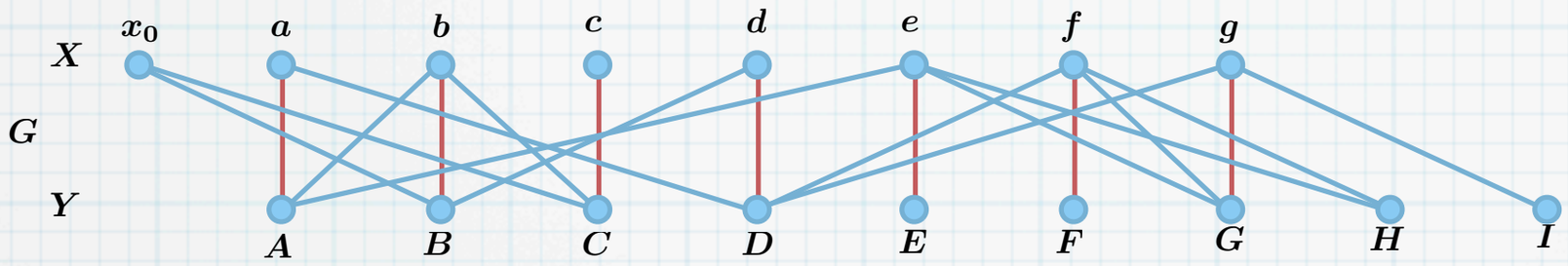
* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$

Der Heiratssatz von Hall (Forts.)

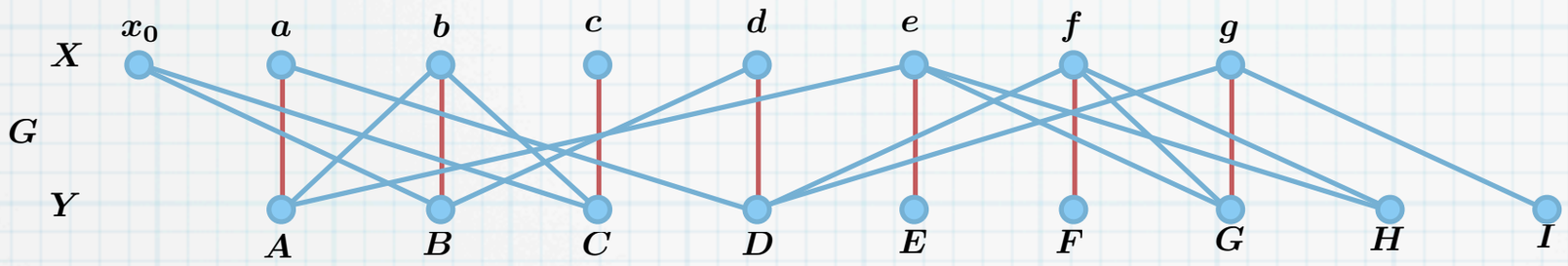
* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$

Der Heiratssatz von Hall (Forts.)

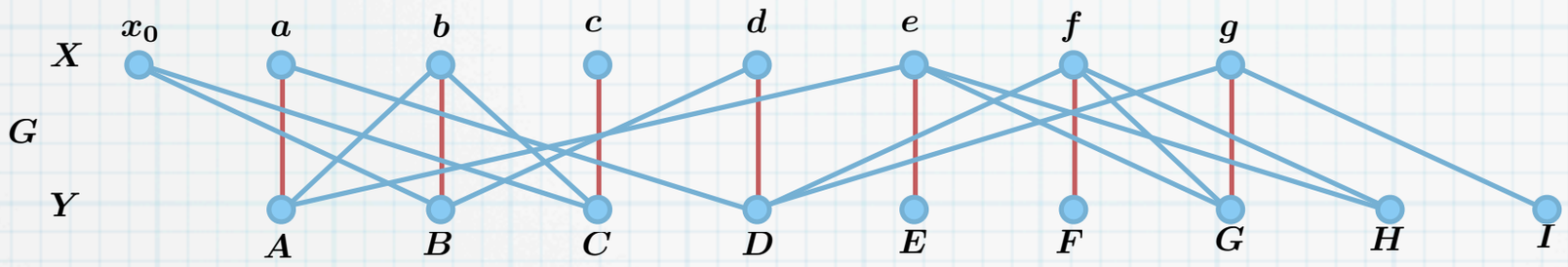
* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)

Der Heiratssatz von Hall (Forts.)

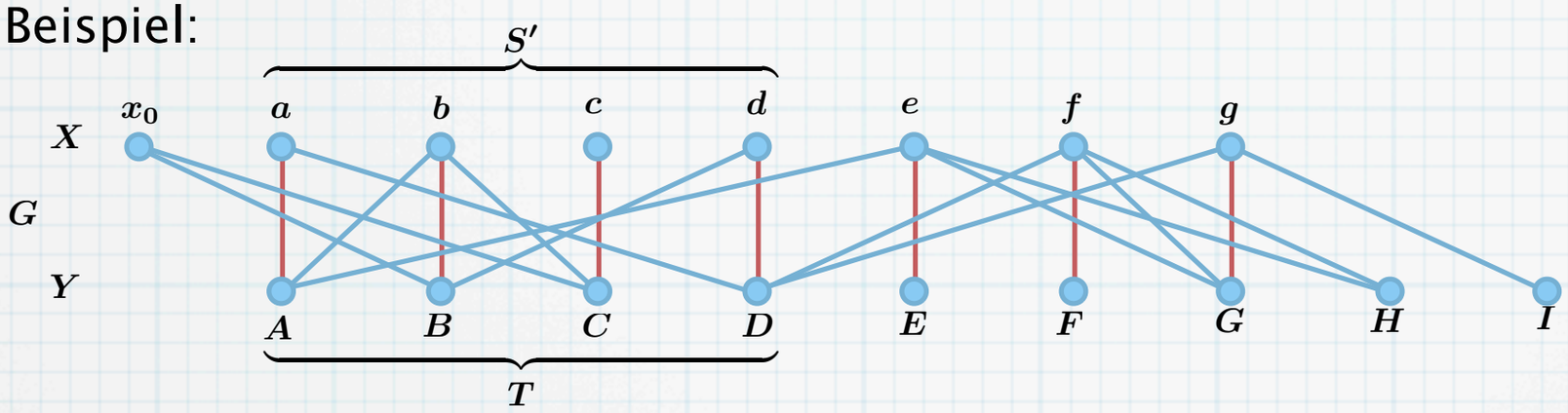
* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

Der Heiratssatz von Hall (Forts.)

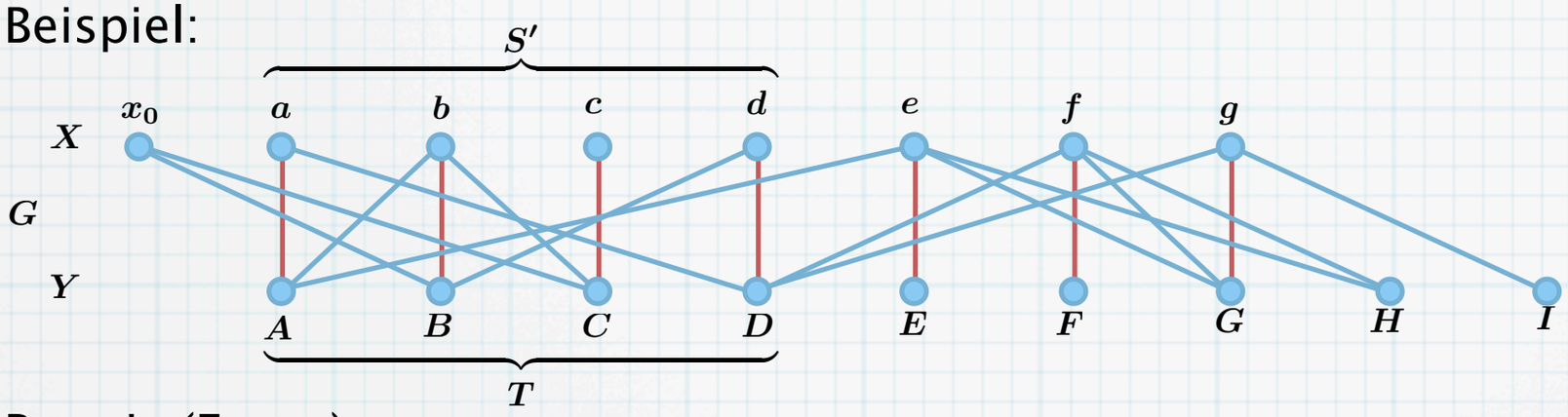
* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

Der Heiratssatz von Hall (Forts.)

* Beispiel:

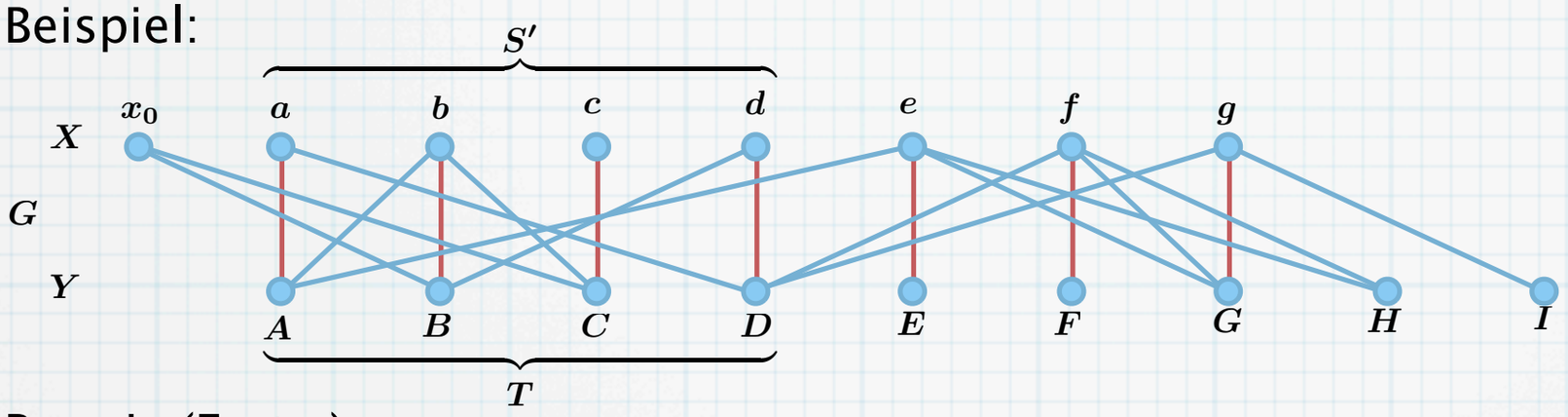


Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Der Heiratssatz von Hall (Forts.)

* Beispiel:



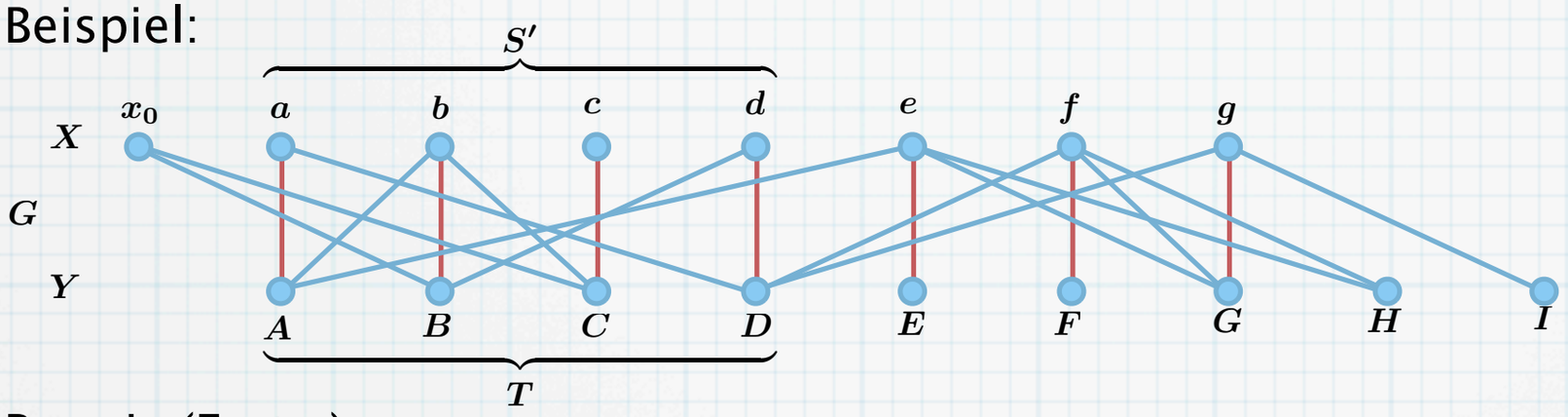
Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

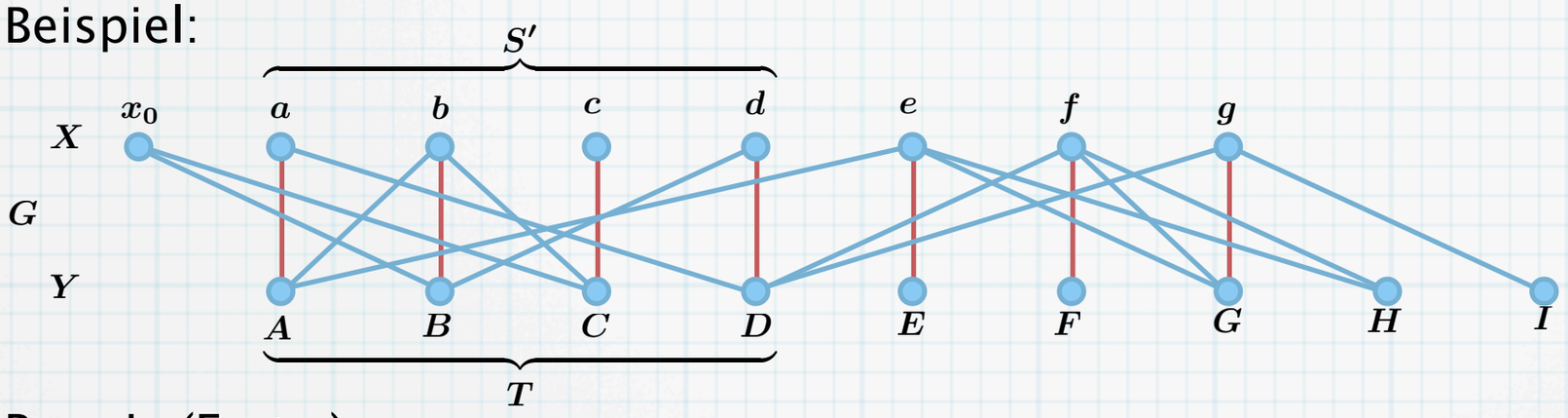
* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

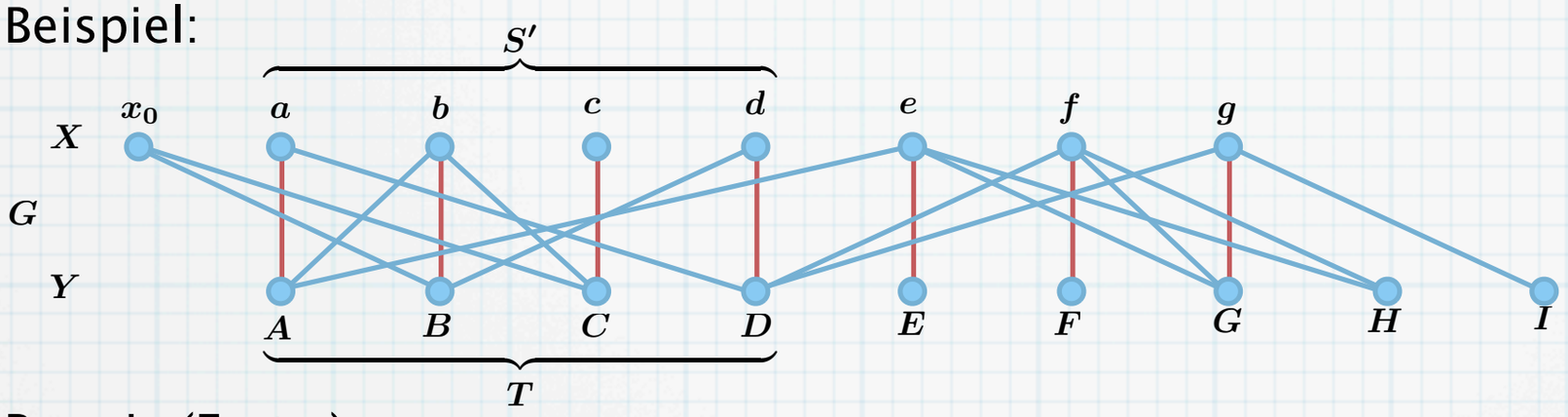
Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*-alternierenden Weges der Länge 1.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

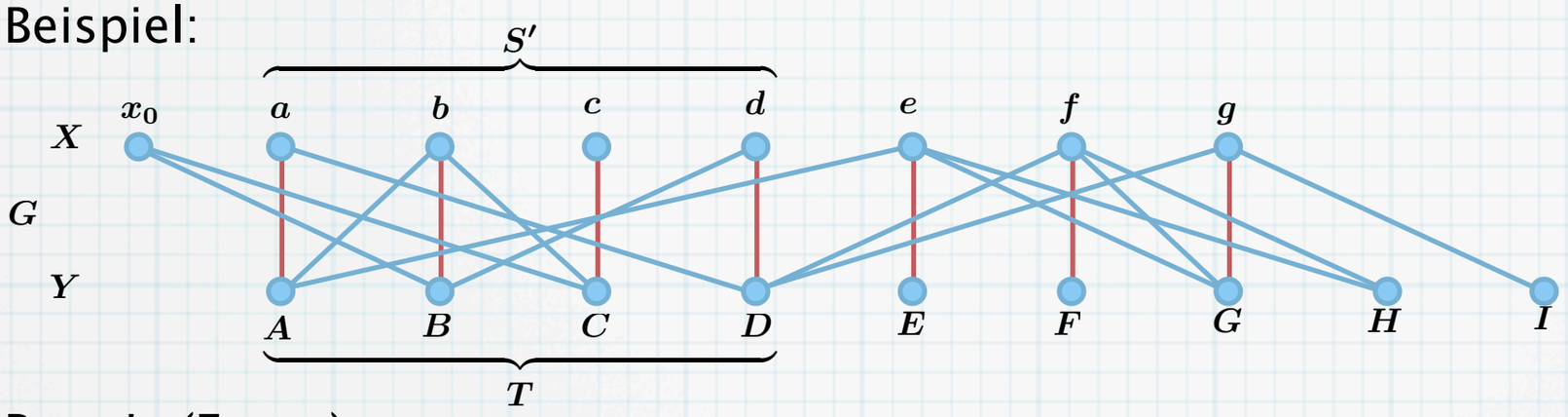
Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*-alternierenden Weges der Länge 1.
Also ist $y \in T$.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

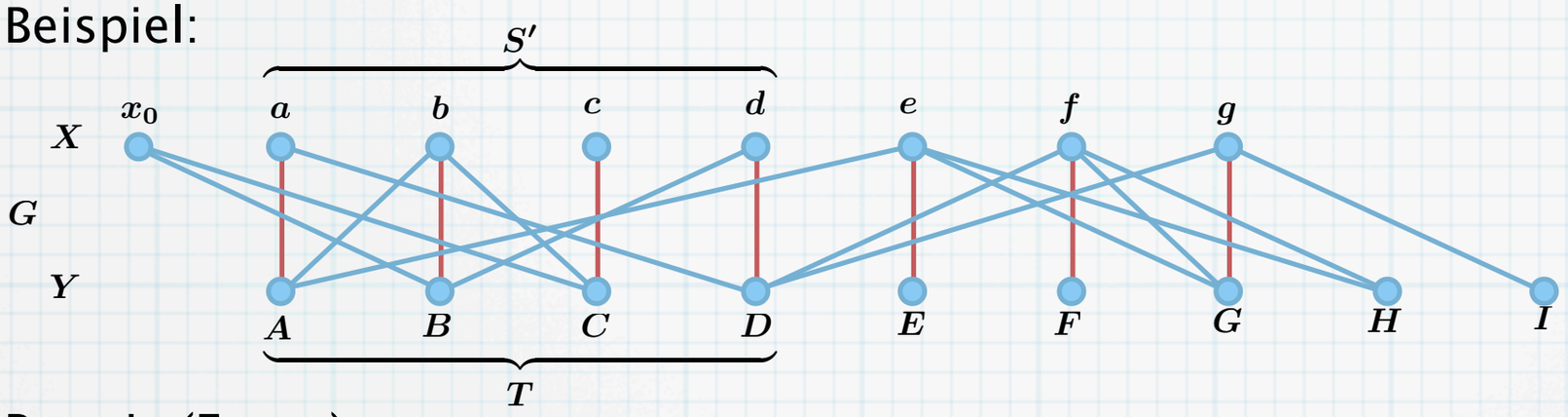
Dann ist y Endknoten eines M^*–alternierenden Weges der Länge 1.

Also ist $y \in T$.

2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*–alternierenden Weg P mit Endknoten x_0 vorkommt.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*–alternierenden Weges der Länge 1.

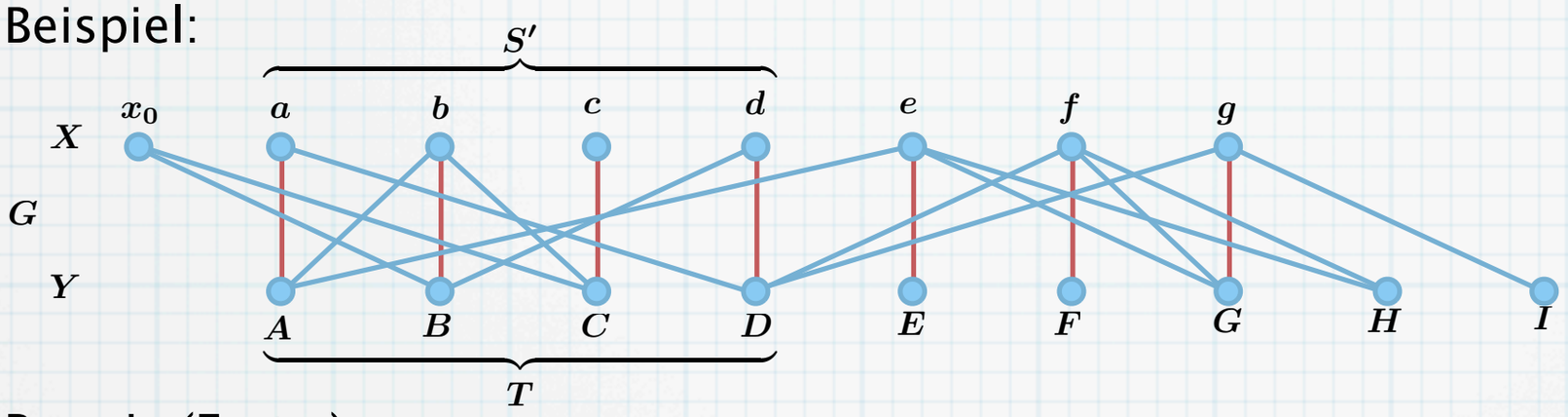
Also ist $y \in T$.

2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*–alternierenden Weg P mit Endknoten x_0 vorkommt.

1. Fall, y ist bereits ein Knoten in P.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*-alternierenden Weges der Länge 1.

Also ist $y \in T$.

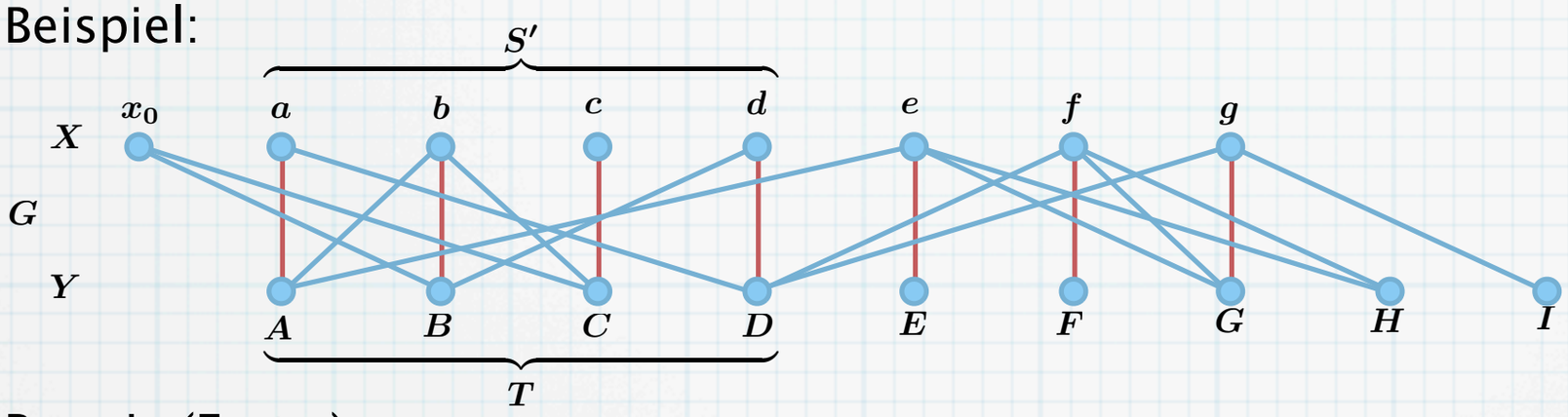
2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*-alternierenden Weg P mit Endknoten x_0 vorkommt.

1. Fall, y ist bereits ein Knoten in P.

2. Fall, P kann durch Hinzufügen der Kante $\{x_i, y\}$ zu einem längeren M^*-alternierenden Weg P' erweitert werden.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*–alternierenden Weges der Länge 1.

Also ist $y \in T$.

2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*–alternierenden Weg P mit Endknoten x_0 vorkommt.

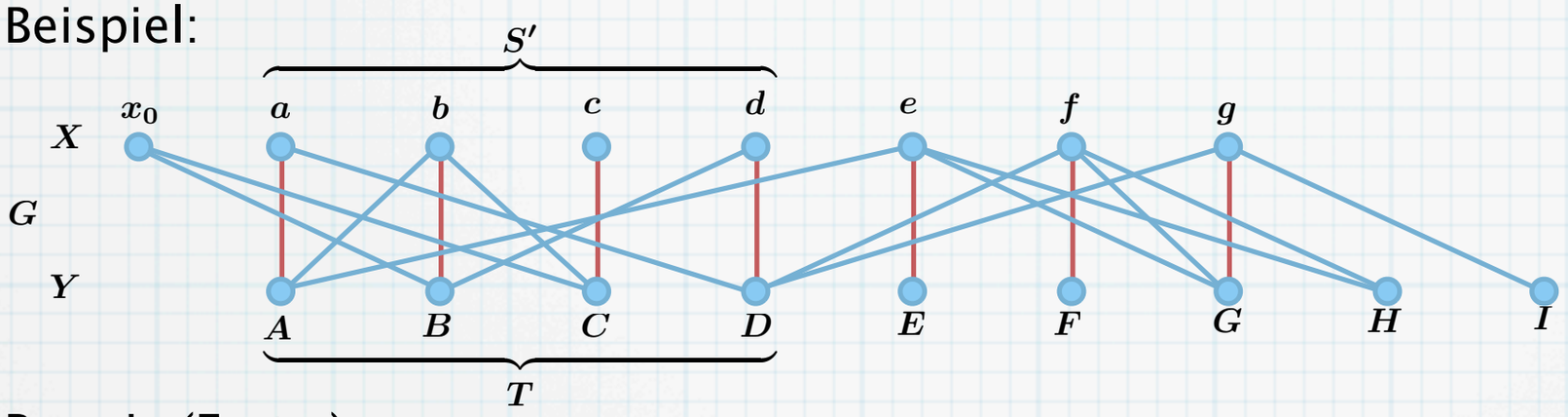
1. Fall, y ist bereits ein Knoten in P.

2. Fall, P kann durch Hinzufügen der Kante $\{x_i, y\}$ zu einem längeren M^*–alternierenden Weg P' erweitert werden.

In beiden Unterfällen ist $y \in T$.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*-alternierenden Weges der Länge 1.

Also ist $y \in T$.

2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*-alternierenden Weg P mit Endknoten x_0 vorkommt.

1. Fall, y ist bereits ein Knoten in P.

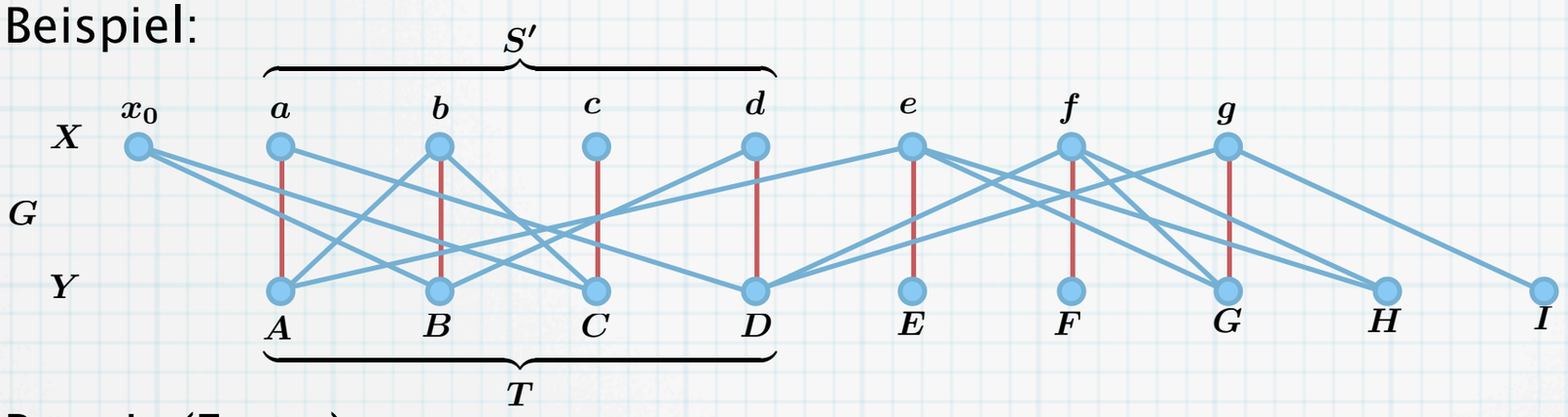
2. Fall, P kann durch Hinzufügen der Kante $\{x_i, y\}$ zu einem längeren M^*-alternierenden Weg P' erweitert werden.

In beiden Unterfällen ist $y \in T$.

Da in beiden Fällen stets $y \in T$, ist $N(S) \subseteq T$.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*–alternierenden Weges der Länge 1.

Also ist $y \in T$.

2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*–alternierenden Weg P mit Endknoten x_0 vorkommt.

1. Fall, y ist bereits ein Knoten in P.

2. Fall, P kann durch Hinzufügen der Kante $\{x_i, y\}$ zu einem längeren M^*–alternierenden Weg P' erweitert werden.

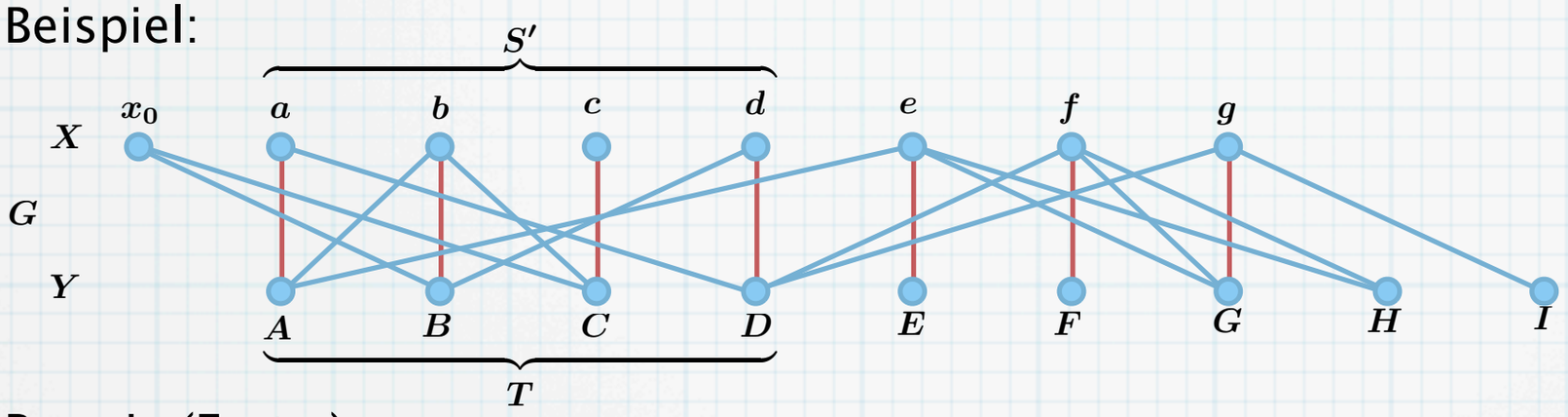
In beiden Unterfällen ist $y \in T$.

Da in beiden Fällen stets $y \in T$, ist $N(S) \subseteq T$.

Offensichtlich gilt $T \subseteq N(S)$. Also ist $N(S) = T$.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*–alternierenden Weges der Länge 1.

Also ist $y \in T$.

2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*–alternierenden Weg P mit Endknoten x_0 vorkommt.

1. Fall, y ist bereits ein Knoten in P.

2. Fall, P kann durch Hinzufügen der Kante $\{x_i, y\}$ zu einem längeren M^*–alternierenden Weg P' erweitert werden.

In beiden Unterfällen ist $y \in T$.

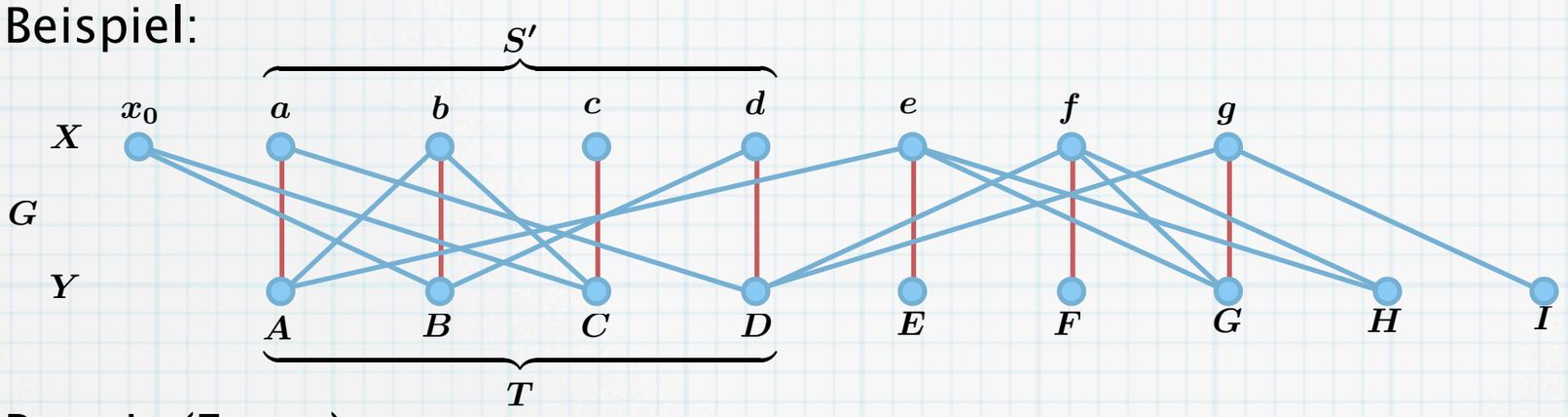
Da in beiden Fällen stets $y \in T$, ist $N(S) \subseteq T$.

Offensichtlich gilt $T \subseteq N(S)$. Also ist $N(S) = T$.

Anderseits ist $|S| = |S'| + 1$

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*–alternierenden Weges der Länge 1.

Also ist $y \in T$.

2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*–alternierenden Weg P mit Endknoten x_0 vorkommt.

1. Fall, y ist bereits ein Knoten in P.

2. Fall, P kann durch Hinzufügen der Kante $\{x_i, y\}$ zu einem längeren M^*–alternierenden Weg P' erweitert werden.

In beiden Unterfällen ist $y \in T$.

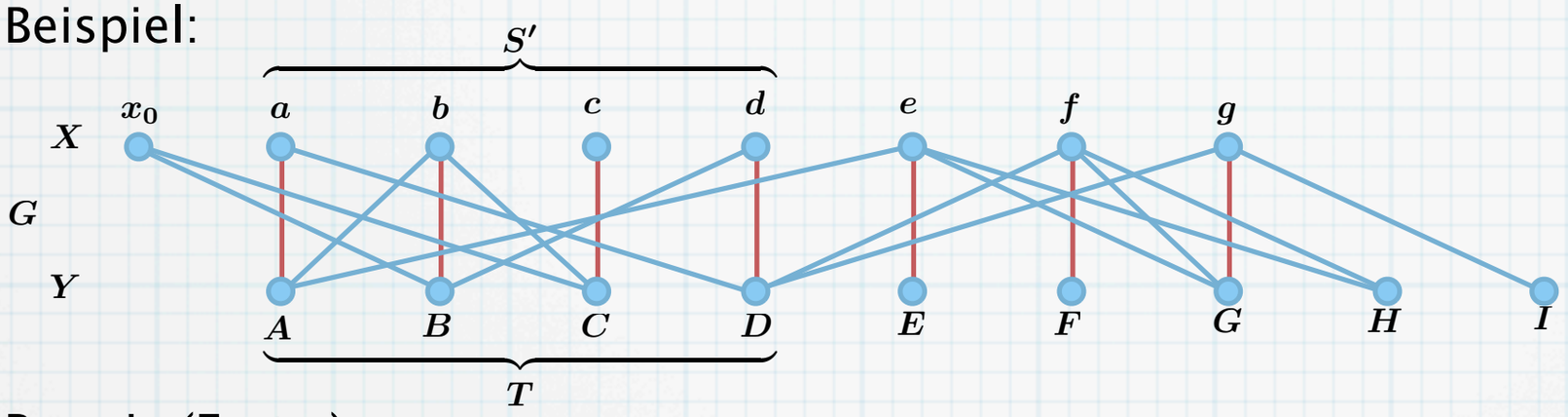
Da in beiden Fällen stets $y \in T$, ist $N(S) \subseteq T$.

Offensichtlich gilt $T \subseteq N(S)$. Also ist $N(S) = T$.

Anderseits ist $|S| = |S'| + 1 = |T| + 1$

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*–alternierenden Weges der Länge 1.

Also ist $y \in T$.

2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*–alternierenden Weg P mit Endknoten x_0 vorkommt.

1. Fall, y ist bereits ein Knoten in P.

2. Fall, P kann durch Hinzufügen der Kante $\{x_i, y\}$ zu einem längeren M^*–alternierenden Weg P' erweitert werden.

In beiden Unterfällen ist $y \in T$.

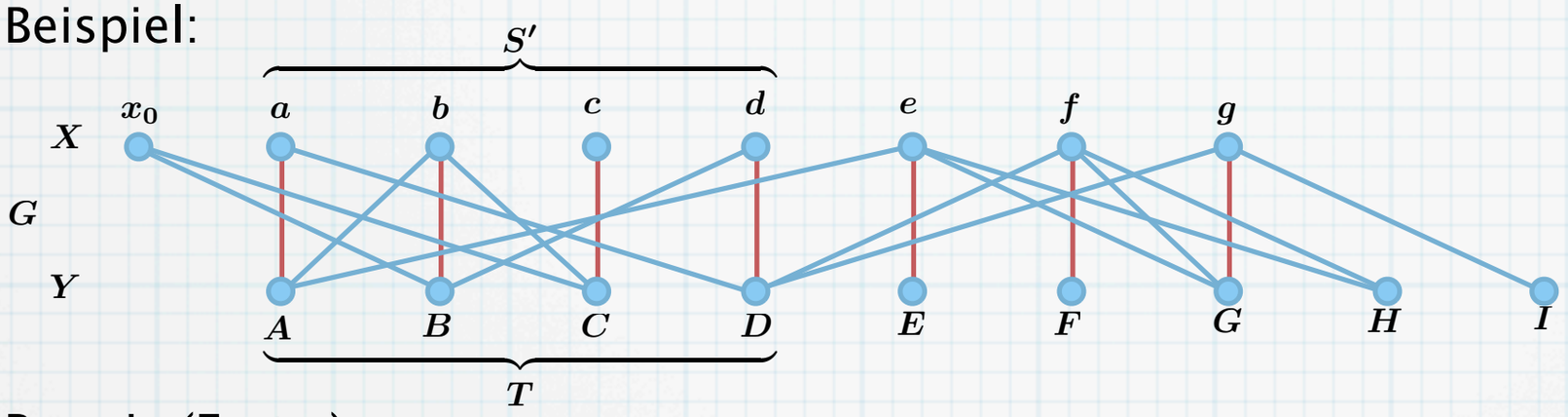
Da in beiden Fällen stets $y \in T$, ist $N(S) \subseteq T$.

Offensichtlich gilt $T \subseteq N(S)$. Also ist $N(S) = T$.

Anderseits ist $|S| = |S'| + 1 = |T| + 1 = |N(S)| + 1$.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*–alternierenden Weges der Länge 1.

Also ist $y \in T$.

2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*–alternierenden Weg P mit Endknoten x_0 vorkommt.

1. Fall, y ist bereits ein Knoten in P.

2. Fall, P kann durch Hinzufügen der Kante $\{x_i, y\}$ zu einem längeren M^*–alternierenden Weg P' erweitert werden.

In beiden Unterfällen ist $y \in T$.

Da in beiden Fällen stets $y \in T$, ist $N(S) \subseteq T$.

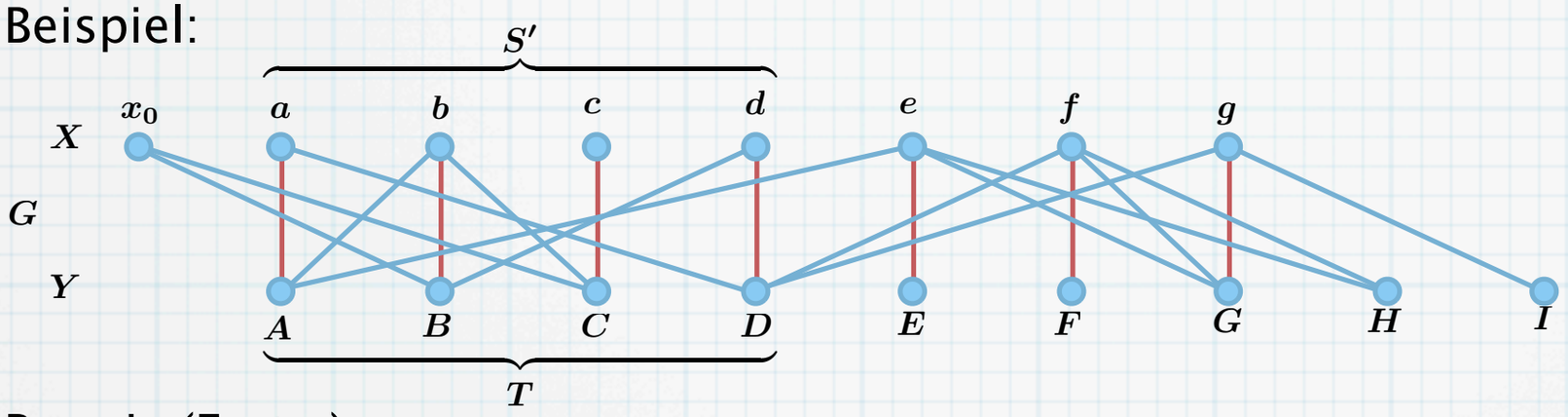
Offensichtlich gilt $T \subseteq N(S)$. Also ist $N(S) = T$.

Anderseits ist $|S| = |S'| + 1 = |T| + 1 = |N(S)| + 1$.

Also ist $|N(S)| < |S|$, im Widerspruch zur Voraussetzung.

Der Heiratssatz von Hall (Forts.)

* Beispiel:



Länge 1: $(x_0, B), (x_0, C)$
Länge 2: $(x_0, B, b), (x_0, C, c)$
Länge 3: $(x_0, B, b, A), (x_0, B, b, C)$
Länge 4: $(x_0, B, b, A, a), (x_0, B, b, C, c)$
Länge 5: (x_0, B, b, A, a, D)
Länge 6: (x_0, B, b, A, a, D, d)

* Beweis (Forts.):

Sei $S := S' \cup \{x_0\}$ und $y \in Y$ adjazent zu einem Knoten in S, d.h. $y \in N(S)$.

1. Fall, y ist adjazent zu x_0.

Dann ist y Endknoten eines M^*-alternierenden Weges der Länge 1.

Also ist $y \in T$.

2. Fall, y ist adjazent zu einem Knoten $x_i \in X$, der auf einem M^*-alternierenden Weg P mit Endknoten x_0 vorkommt.

1. Fall, y ist bereits ein Knoten in P.

2. Fall, P kann durch Hinzufügen der Kante $\{x_i, y\}$ zu einem längeren M^*-alternierenden Weg P' erweitert werden.

In beiden Unterfällen ist $y \in T$.

Da in beiden Fällen stets $y \in T$, ist $N(S) \subseteq T$.

Offensichtlich gilt $T \subseteq N(S)$. Also ist $N(S) = T$.

Anderseits ist $|S| = |S'| + 1 = |T| + 1 = |N(S)| + 1$.

Also ist $|N(S)| < |S|$, im Widerspruch zur Voraussetzung.

Die Annahme war also falsch, und es gibt ein Matching, welches alle Knoten von X sättigt.

Perfekte Matchings in regulären bipartiten Graphen

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k-regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k-regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.

  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.

  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.

  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k-regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.

  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.

  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.

  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.

  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.

  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.

  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.

  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.

  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.

  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.

  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.

  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.

  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.

  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.

  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.

  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.

  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.

  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.

  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.
  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.
  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.
  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.
  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.
  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.
  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.
  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.
  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.
  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.
  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.
  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.
  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.
  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.
  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.
  Da $N(S)$ die Menge der mit S benachbarten Knoten ist, gilt $E_1 \subseteq E_2$.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.
  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.
  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.
  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.
  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.
  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.
  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.
  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.
  Da $N(S)$ die Menge der mit S benachbarten Knoten ist, gilt $E_1 \subseteq E_2$.
  Folglich $|E_1| \leq |E_2|$.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.
  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.
  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.
  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.
  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.
  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.
  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.
  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.
  Da $N(S)$ die Menge der mit S benachbarten Knoten ist, gilt $E_1 \subseteq E_2$.
  Folglich $|E_1| \leq |E_2|$.
  Aufgrund der k–Regularität gilt $|E_2| = k \cdot |N(S)|$.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.
  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.
  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.
  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.
  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.
  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.
  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.
  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.
  Da $N(S)$ die Menge der mit S benachbarten Knoten ist, gilt $E_1 \subseteq E_2$.
  Folglich $|E_1| \leq |E_2|$.
  Aufgrund der k–Regularität gilt $|E_2| = k \cdot |N(S)|$.
  Also ist $k \cdot |N(S)| = |E_2|$

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.
  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.
  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.
  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.
  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.
  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.
  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.
  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.
  Da $N(S)$ die Menge der mit S benachbarten Knoten ist, gilt $E_1 \subseteq E_2$.
  Folglich $|E_1| \leq |E_2|$.
  Aufgrund der k–Regularität gilt $|E_2| = k \cdot |N(S)|$.
  Also ist $k \cdot |N(S)| = |E_2| \geq |E_1|$

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.
  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.
  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.
  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.
  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.
  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.
  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.
  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.
  Da $N(S)$ die Menge der mit S benachbarten Knoten ist, gilt $E_1 \subseteq E_2$.
  Folglich $|E_1| \leq |E_2|$.
  Aufgrund der k–Regularität gilt $|E_2| = k \cdot |N(S)|$.
  Also ist $k \cdot |N(S)| = |E_2| \geq |E_1| = k \cdot |S|$.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.
  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.
  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.
  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.
  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.
  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.
  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.
  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.
  Da $N(S)$ die Menge der mit S benachbarten Knoten ist, gilt $E_1 \subseteq E_2$.
  Folglich $|E_1| \leq |E_2|$.
  Aufgrund der k–Regularität gilt $|E_2| = k \cdot |N(S)|$.
  Also ist $k \cdot |N(S)| = |E_2| \geq |E_1| = k \cdot |S|$.
  Durch Kürzen ($k > 0$) folgt dann $|N(S)| \geq |S|$.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.
  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.
  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.
  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.
  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.
  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.
  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.
  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.
  Da $N(S)$ die Menge der mit S benachbarten Knoten ist, gilt $E_1 \subseteq E_2$.
  Folglich $|E_1| \leq |E_2|$.
  Aufgrund der k–Regularität gilt $|E_2| = k \cdot |N(S)|$.
  Also ist $k \cdot |N(S)| = |E_2| \geq |E_1| = k \cdot |S|$.
  Durch Kürzen ($k > 0$) folgt dann $|N(S)| \geq |S|$.
  Da S eine beliebige Knotenteilmenge war, folgt aus dem Heiratssatz von Hall, dass G ein Matching M enthält, welches jeden Knoten von X sättigt.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.

  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.

  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.

  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.

  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.

  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.

  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.

  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.

  Da $N(S)$ die Menge der mit S benachbarten Knoten ist, gilt $E_1 \subseteq E_2$.

  Folglich $|E_1| \leq |E_2|$.

  Aufgrund der k–Regularität gilt $|E_2| = k \cdot |N(S)|$.

  Also ist $k \cdot |N(S)| = |E_2| \geq |E_1| = k \cdot |S|$.

  Durch Kürzen ($k > 0$) folgt dann $|N(S)| \geq |S|$.

  Da S eine beliebige Knotenteilmenge war, folgt aus dem Heiratssatz von Hall, dass G ein Matching M enthält, welches jeden Knoten von X sättigt.

  Da $|X| = |Y|$, wird auch jeder Knoten in Y durch dieses Matching M gesättigt.

Perfekte Matchings in regulären bipartiten Graphen

* **Satz 11**:
  Sei G ein k–regulärer bipartiter Graph (mit $k > 0$). Dann hat G ein perfektes Matching.
* Das heißt: Wenn jeder Junge k Freundinnen hat und jedes Mädchen k Freunde, dann kann jeder Junge eine seiner Freundinnen und jedes Mädchen einen ihrer Freunde heiraten.
* Beweis (von Satz 11):

  Sei $X \cup Y = V$ die Bipartition von $G = (V, E)$.

  Es gibt $|X|$ Knoten in X, und jeder Knoten hat k Kanten.

  Somit gibt es $k \cdot |X|$ Kanten zwischen X und Y.

  Mit dem gleichen Argument gibt es $k \cdot |Y|$ Kanten zwischen Y und X.

  Aus $k \cdot |X| = k \cdot |Y|$ folgt durch Kürzen ($k > 0$) dann $|X| = |Y|$.

  Sei $S \subseteq X$ und E_1 die Teilmenge der Kanten, die mit Knoten aus S inzidieren.

  Aufgrund der k–Regularität gilt $|E_1| = k \cdot |S|$.

  Sei E_2 die Teilmenge der Kanten, die mit Knoten aus $N(S)$ inzidieren.

  Da $N(S)$ die Menge der mit S benachbarten Knoten ist, gilt $E_1 \subseteq E_2$.

  Folglich $|E_1| \leq |E_2|$.

  Aufgrund der k–Regularität gilt $|E_2| = k \cdot |N(S)|$.

  Also ist $k \cdot |N(S)| = |E_2| \geq |E_1| = k \cdot |S|$.

  Durch Kürzen ($k > 0$) folgt dann $|N(S)| \geq |S|$.

  Da S eine beliebige Knotenteilmenge war, folgt aus dem Heiratssatz von Hall, dass G ein Matching M enthält, welches jeden Knoten von X sättigt.

  Da $|X| = |Y|$, wird auch jeder Knoten in Y durch dieses Matching M gesättigt.

  Damit ist M ein perfektes Matching.

Skizze eines Matchingalgorithmus für bipartite Graphen

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

(a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** matching

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** matching

(2) wähle ein beliebiges Matching M

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) <u>entweder</u> ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** matching

(2) wähle ein beliebiges Matching M

(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** matching
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) **if** es gibt einen M–erweiternden Weg P mit Endknoten x_0 **then**

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** matching
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) **if** es gibt einen M–erweiternden Weg P mit Endknoten x_0 **then**
(5) bestimme Matching M' durch Umfärbung von M entlang P

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm matching
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      if es gibt einen M–erweiternden Weg P mit Endknoten x_0 then
(5)        bestimme Matching M' durch Umfärbung von M entlang P
(6)        setze M := M'
```

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm matching
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      if es gibt einen M–erweiternden Weg P mit Endknoten x_0 then
(5)        bestimme Matching M' durch Umfärbung von M entlang P
(6)        setze M := M'
(7)      else
```

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm matching
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      if es gibt einen M–erweiternden Weg P mit Endknoten x_0 then
(5)        bestimme Matching M' durch Umfärbung von M entlang P
(6)        setze M := M'
(7)      else
(8)        bestimme S wie im Beweis des Heiratssatzes
```

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm matching
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      if es gibt einen M–erweiternden Weg P mit Endknoten x_0 then
(5)        bestimme Matching M' durch Umfärbung von M entlang P
(6)        setze M := M'
(7)      else
(8)        bestimme S wie im Beweis des Heiratssatzes
(9)        gib S aus
```

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm matching
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      if es gibt einen M–erweiternden Weg P mit Endknoten x_0 then
(5)        bestimme Matching M' durch Umfärbung von M entlang P
(6)        setze M := M'
(7)      else
(8)        bestimme S wie im Beweis des Heiratssatzes
(9)        gib S aus
(10)       goto (14)
```

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm matching
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      if es gibt einen M–erweiternden Weg P mit Endknoten x_0 then
(5)        bestimme Matching M' durch Umfärbung von M entlang P
(6)        setze M := M'
(7)      else
(8)        bestimme S wie im Beweis des Heiratssatzes
(9)        gib S aus
(10)       goto (14)
(11)     end if
```

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm matching
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      if es gibt einen M–erweiternden Weg P mit Endknoten x_0 then
(5)        bestimme Matching M' durch Umfärbung von M entlang P
(6)        setze M := M'
(7)      else
(8)        bestimme S wie im Beweis des Heiratssatzes
(9)        gib S aus
(10)       goto (14)
(11)     end if
(12)   end while
```

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm matching
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      if es gibt einen M–erweiternden Weg P mit Endknoten x_0 then
(5)        bestimme Matching M' durch Umfärbung von M entlang P
(6)        setze M := M'
(7)      else
(8)        bestimme S wie im Beweis des Heiratssatzes
(9)        gib S aus
(10)       goto (14)
(11)     end if
(12)   end while
(13)   gib M aus
```

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm matching
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      if es gibt einen M–erweiternden Weg P mit Endknoten x_0 then
(5)        bestimme Matching M' durch Umfärbung von M entlang P
(6)        setze M := M'
(7)      else
(8)        bestimme S wie im Beweis des Heiratssatzes
(9)        gib S aus
(10)       goto (14)
(11)     end if
(12)   end while
(13)   gib M aus
(14) end algorithm
```

Skizze eines Matchingalgorithmus für bipartite Graphen

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) entweder ein Matching M, durch das alle Knoten in X gesättigt werden,

  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm matching
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      if es gibt einen M–erweiternden Weg P mit Endknoten x_0 then
(5)        bestimme Matching M' durch Umfärbung von M entlang P
(6)        setze M := M'
(7)      else
(8)        bestimme S wie im Beweis des Heiratssatzes
(9)        gib S aus
(10)       goto (14)
(11)     end if
(12)   end while
(13)   gib M aus
(14) end algorithm
```

* Die algorithmische Schwierigkeit ist dabei die Bestimmung eines M–erweiternden Weges mit Endknoten x_0 bzw. der Nachweis, dass es keinen solchen Weg gibt.

Alternierende Bäume

Alternierende Bäume

* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:

Alternierende Bäume

* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:

  (a) H ist ein Baum,

Alternierende Bäume

* **Definition 12**:
Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:

(a) H ist ein Baum,

(b) $x_0 \in V(H)$,

Alternierende Bäume

* **Definition 12**:
Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:

  (a) H ist ein Baum,

  (b) $x_0 \in V(H)$,

  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,

Alternierende Bäume

* **Definition 12**:
Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:

  (a) H ist ein Baum,

  (b) $x_0 \in V(H)$,

  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,

  wird als M-**alternierender Baum mit Wurzel** x_0 bezeichnet.

Alternierende Bäume

* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:

  (a) H ist ein Baum,

  (b) $x_0 \in V(H)$,

  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,

  wird als M-**alternierender Baum mit Wurzel** x_0 bezeichnet.
* Beispiel:

Alternierende Bäume

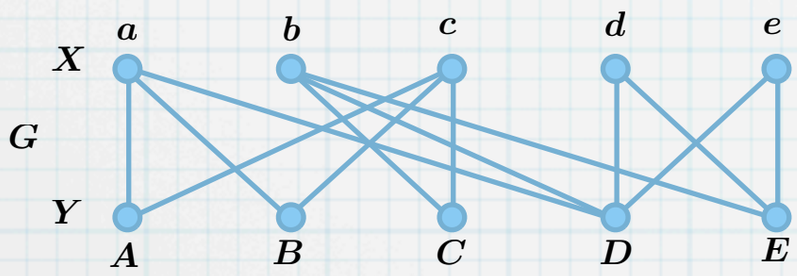
* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:

  (a) H ist ein Baum,

  (b) $x_0 \in V(H)$,

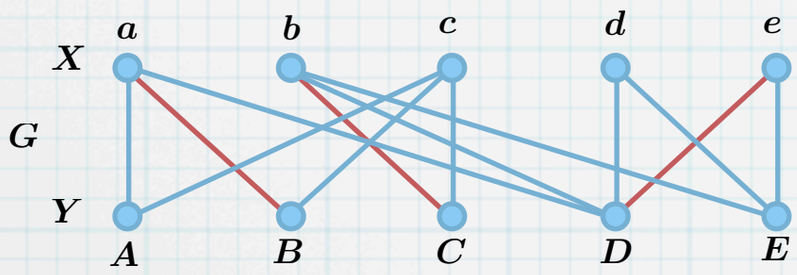
  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,

  wird als M-**alternierender Baum mit Wurzel** x_0 bezeichnet.
* Beispiel:



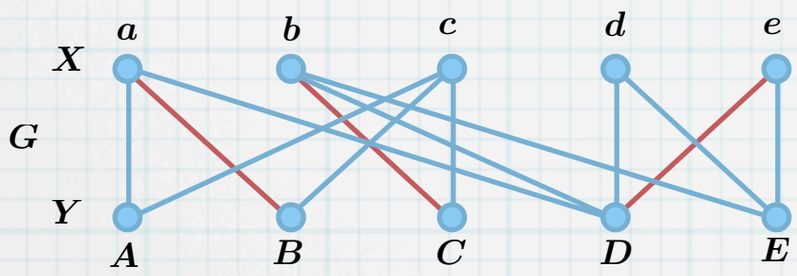
Alternierende Bäume

* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:
  (a) H ist ein Baum,
  (b) $x_0 \in V(H)$,
  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,
  wird als M-**alternierender Baum mit Wurzel** x_0 bezeichnet.
* Beispiel:



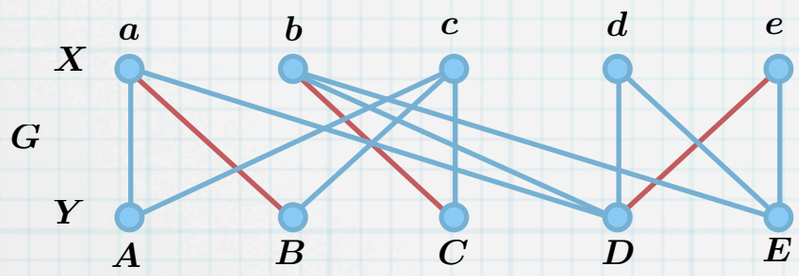
Alternierende Bäume

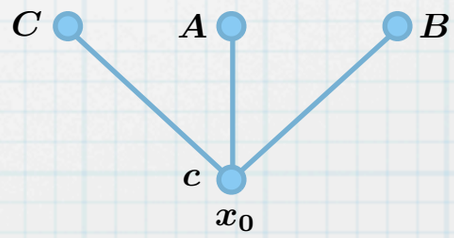
* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:
  (a) H ist ein Baum,
  (b) $x_0 \in V(H)$,
  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,
  wird als M-**alternierender Baum mit Wurzel** x_0 bezeichnet.
* Beispiel:



Alternierende Bäume

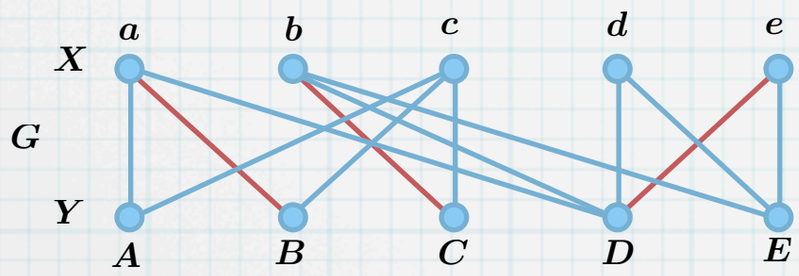
* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:
  (a) H ist ein Baum,
  (b) $x_0 \in V(H)$,
  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,
  wird als M-**alternierender Baum mit Wurzel** x_0 bezeichnet.
* Beispiel:

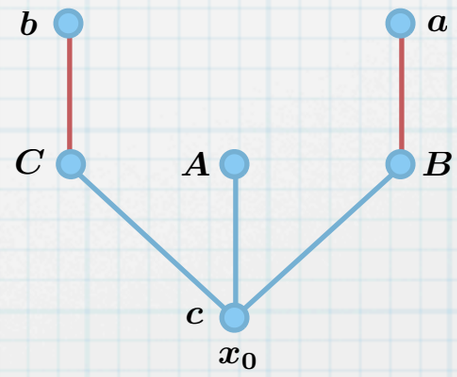




Alternierende Bäume

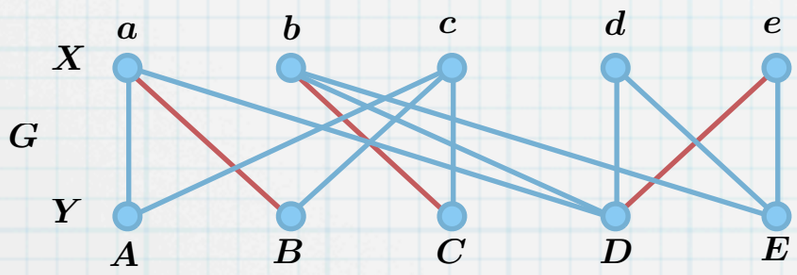
* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:
  (a) H ist ein Baum,
  (b) $x_0 \in V(H)$,
  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,
  wird als M-**alternierender Baum mit Wurzel** x_0 bezeichnet.
* Beispiel:

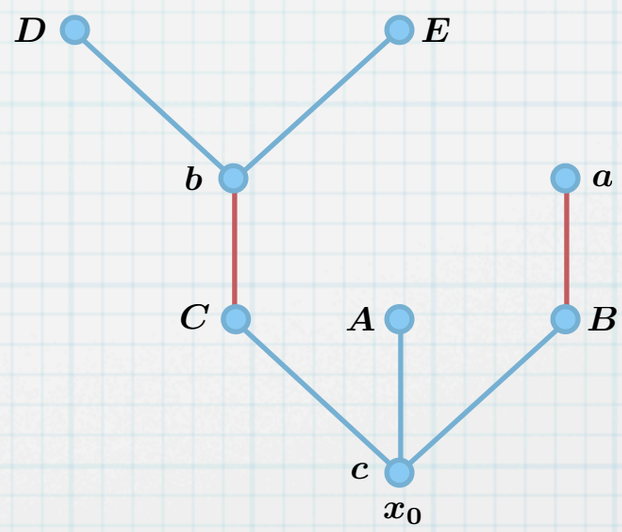




Alternierende Bäume

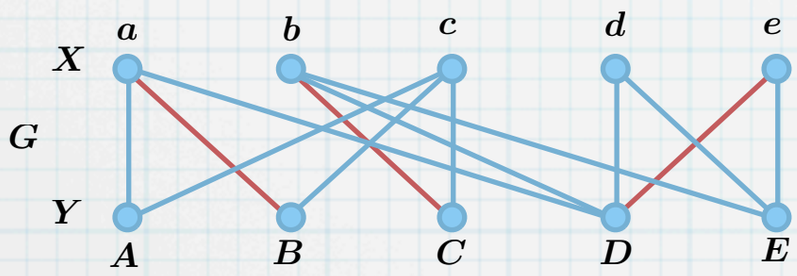
* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:
  (a) H ist ein Baum,
  (b) $x_0 \in V(H)$,
  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,
  wird als M-**alternierender Baum mit Wurzel** x_0 bezeichnet.
* Beispiel:

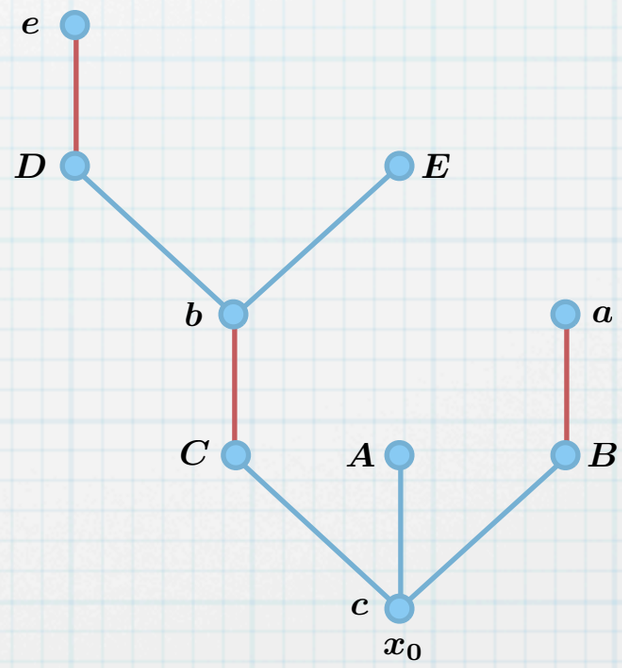




Alternierende Bäume

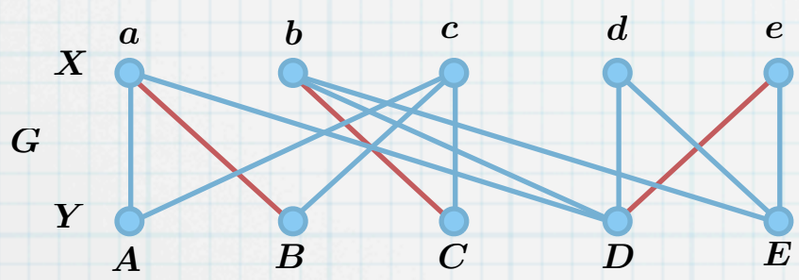
* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:
  (a) H ist ein Baum,
  (b) $x_0 \in V(H)$,
  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,
  wird als M-**alternierender Baum mit Wurzel** x_0 bezeichnet.
* Beispiel:

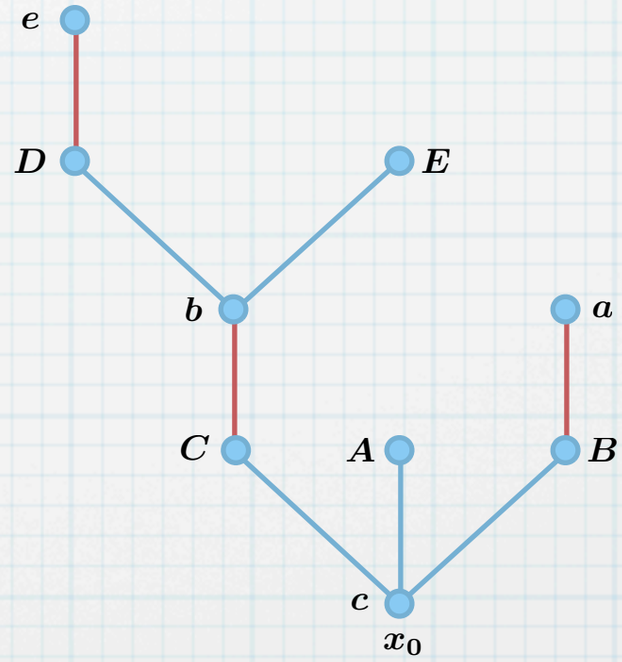




Alternierende Bäume

* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:
  (a) H ist ein Baum,
  (b) $x_0 \in V(H)$,
  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,
  wird als M-**alternierender Baum mit Wurzel** x_0 bezeichnet.
* Beispiel:

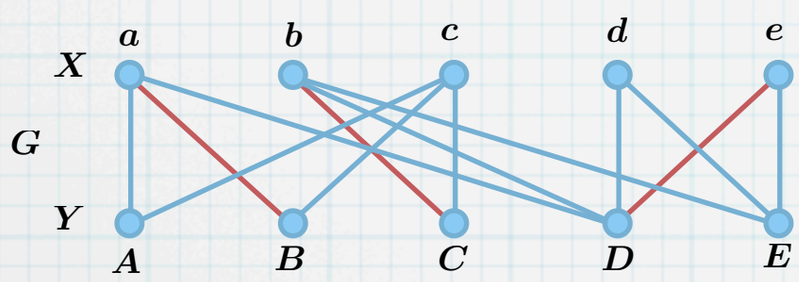


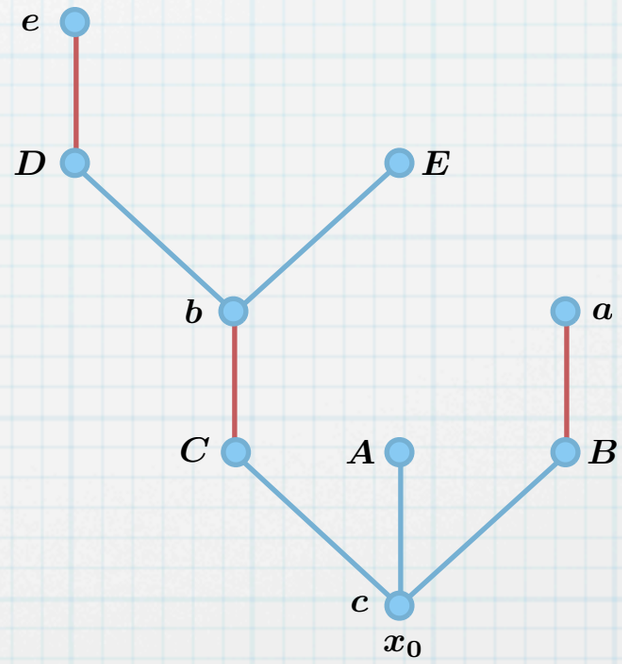


* Ein M-erweiternder Weg mit Endknoten x_0 kann konstruiert werden, indem man einen M-alternierenden Baum mit Wurzel x_0 „wachsen“ lässt.

Alternierende Bäume

* **Definition 12**:
  Sei M ein Matching des bipartiten Graphen $G = (V, E)$ mit $V = X \cup Y$. Sei x_0 ein M-ungesättigter Knoten in X. Ein Untergraph H von G mit den folgenden Eigenschaften:
  (a) H ist ein Baum,
  (b) $x_0 \in V(H)$,
  (c) für jeden Knoten $v \in V(H)$ ist der (eindeutige) Weg zwischen x_0 und v in H ein M-alternierender Weg,
  wird als **M-alternierender Baum mit Wurzel x_0** bezeichnet.
* Beispiel:





* Ein M-erweiternder Weg mit Endknoten x_0 kann konstruiert werden, indem man einen M-alternierenden Baum mit Wurzel x_0 „wachsen“ lässt.
  Dieses Verfahren wird **Ungarische Methode** genannt (nach D. König und J. Egerváry, 1931).

So wachsen alternierende Bäume

So wachsen alternierende Bäume

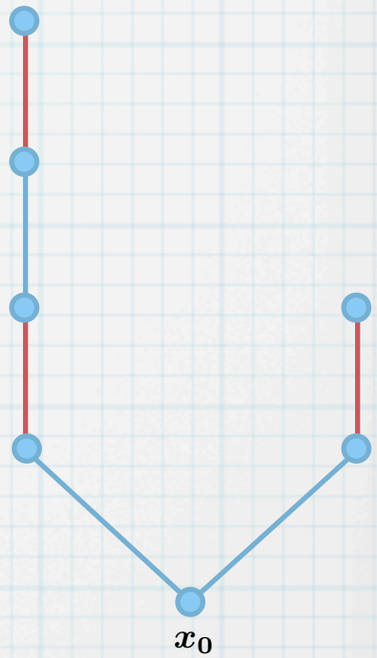
* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:

So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,

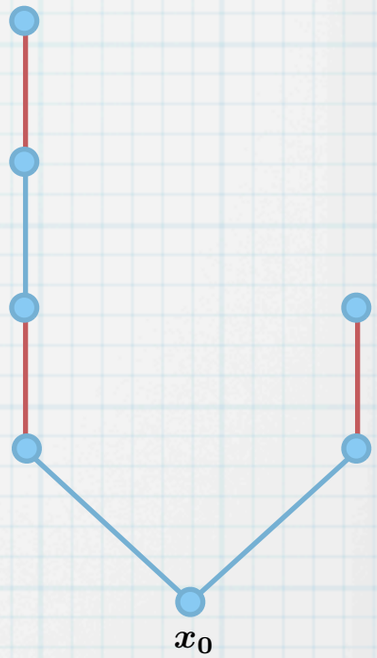
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,



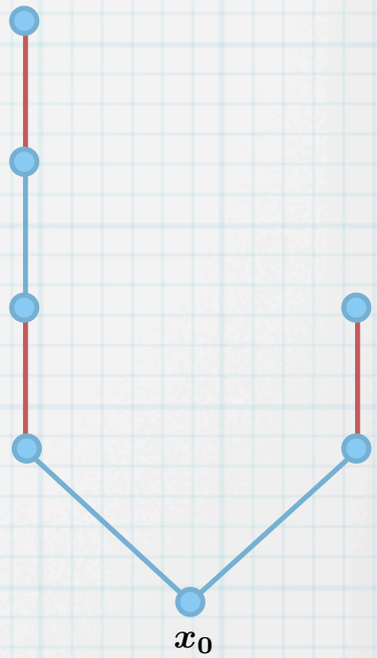
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.



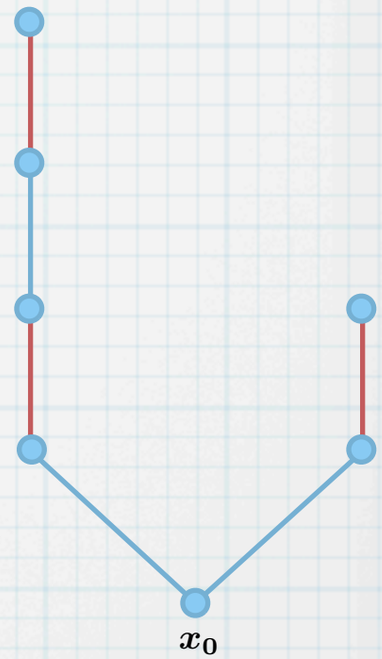
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.



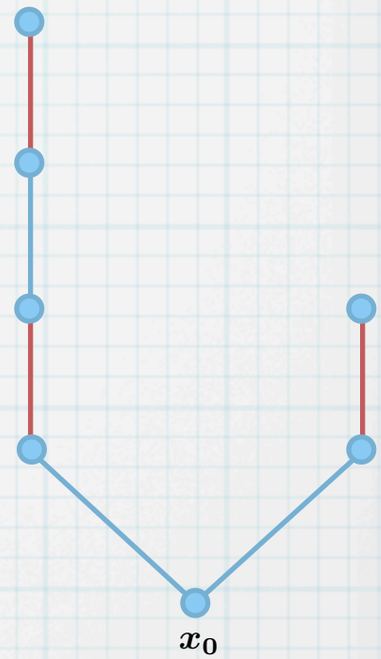
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.



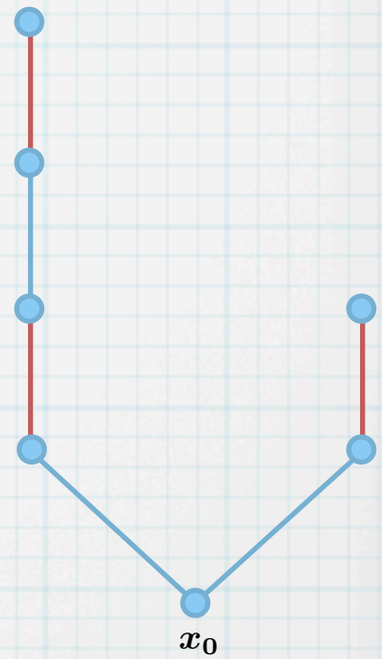
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.



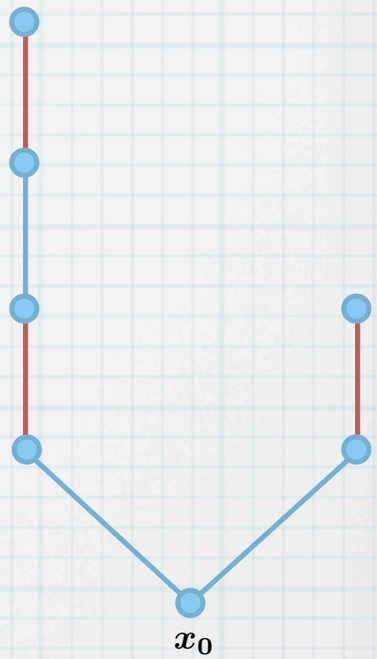
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.



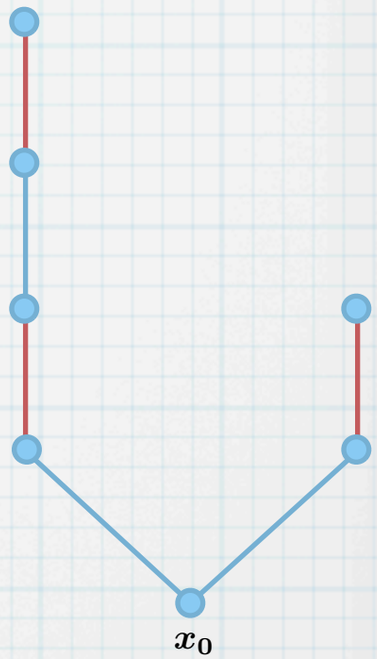
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.



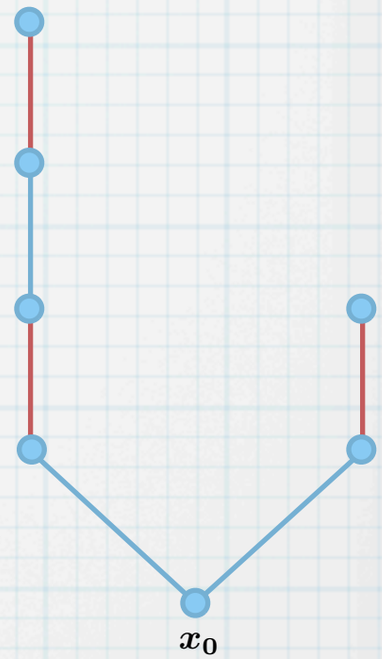
So wachsen alternierende Bäume

- Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
- Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.



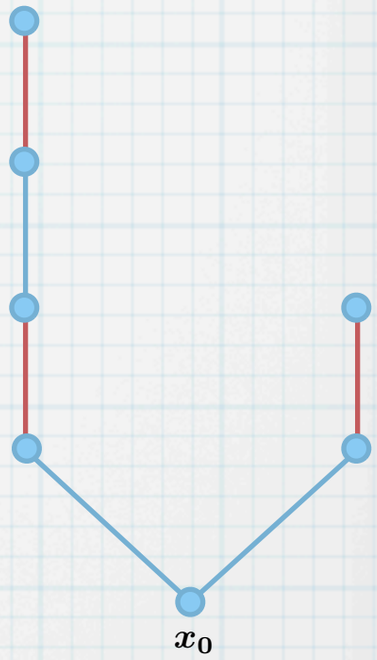
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.



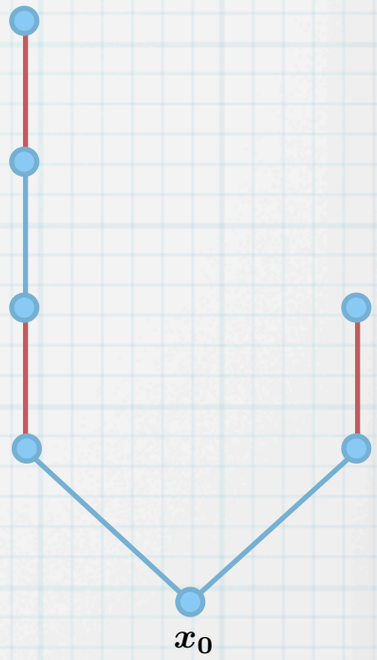
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.



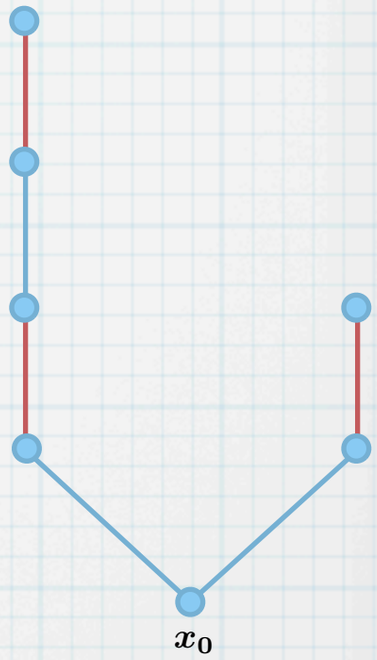
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.



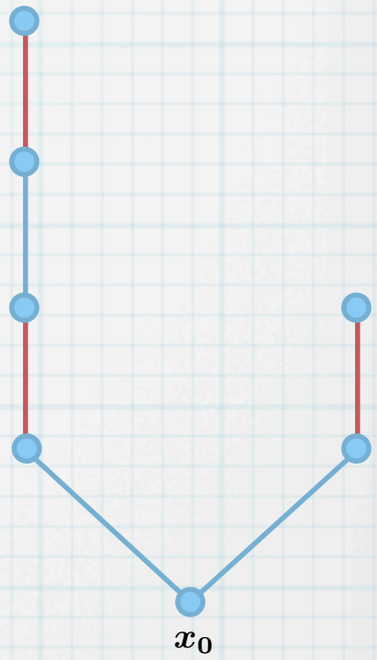
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.
     Damit ist x entweder ungesättigt oder bereits gepaart.



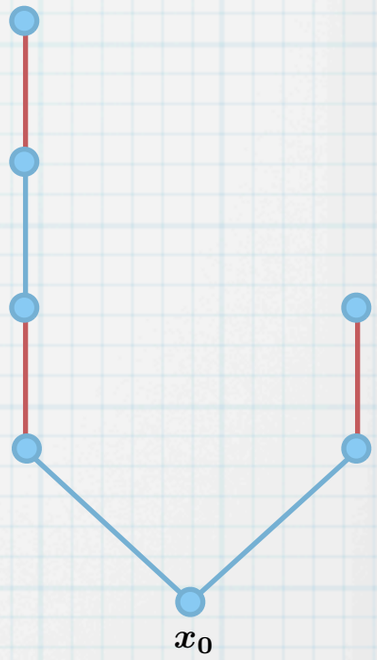
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.
     Damit ist x entweder ungesättigt oder bereits gepaart.
     Also ist die Kante $\{x, y\}$ nicht in M.



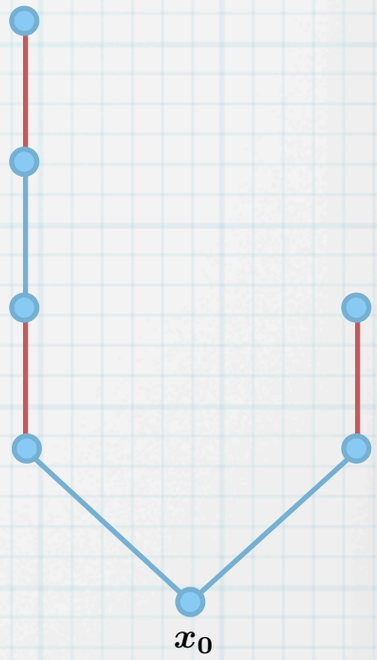
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.
     Damit ist x entweder ungesättigt oder bereits gepaart.
     Also ist die Kante $\{x, y\}$ nicht in M.
     Wird y durch M gesättigt, gibt es eine weitere Kante $\{y, z\} \in M$.



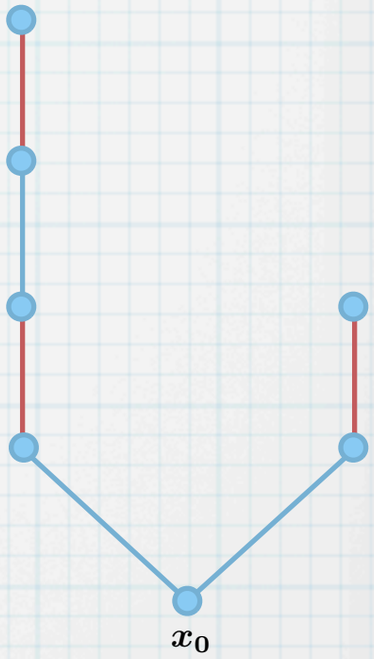
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.
     Damit ist x entweder ungesättigt oder bereits gepaart.
     Also ist die Kante $\{x, y\}$ nicht in M.
     Wird y durch M gesättigt, gibt es eine weitere Kante $\{y, z\} \in M$.
     Der Baum wächst dann durch Hinzufügen von $\{x, y\}$ und $\{y, z\}$.



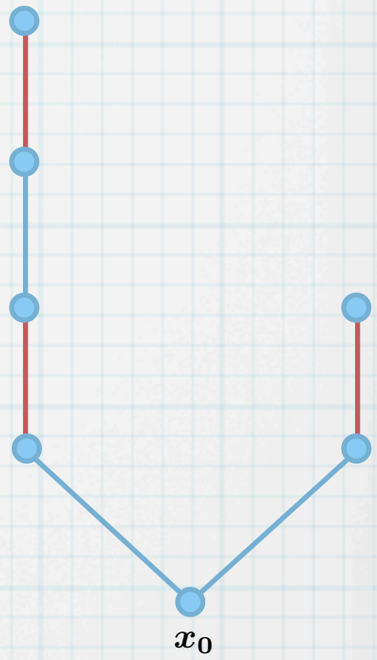
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.
     Damit ist x entweder ungesättigt oder bereits gepaart.
     Also ist die Kante $\{x, y\}$ nicht in M.
     Wird y durch M gesättigt, gibt es eine weitere Kante $\{y, z\} \in M$.
     Der Baum wächst dann durch Hinzufügen von $\{x, y\}$ und $\{y, z\}$.
     Ist y ungesättigt, tritt Fall (b) ein (siehe unten).



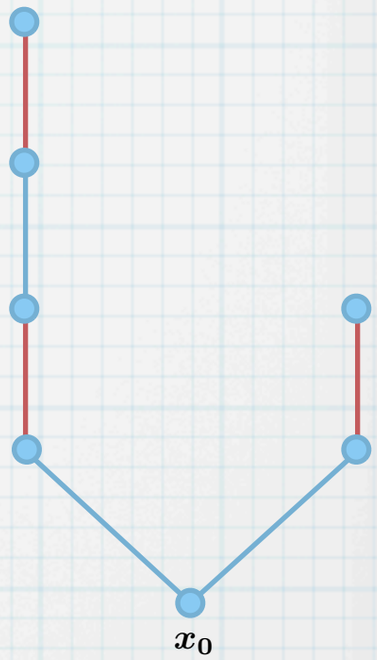
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.
     Damit ist x entweder ungesättigt oder bereits gepaart.
     Also ist die Kante $\{x, y\}$ nicht in M.
     Wird y durch M gesättigt, gibt es eine weitere Kante $\{y, z\} \in M$.
     Der Baum wächst dann durch Hinzufügen von $\{x, y\}$ und $\{y, z\}$.
     Ist y ungesättigt, tritt Fall (b) ein (siehe unten).
  2. Fall, $T = N(S)$.



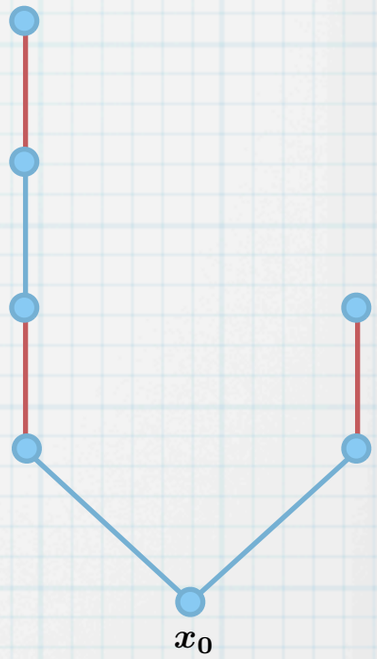
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.
     Damit ist x entweder ungesättigt oder bereits gepaart.
     Also ist die Kante $\{x, y\}$ nicht in M.
     Wird y durch M gesättigt, gibt es eine weitere Kante $\{y, z\} \in M$.
     Der Baum wächst dann durch Hinzufügen von $\{x, y\}$ und $\{y, z\}$.
     Ist y ungesättigt, tritt Fall (b) ein (siehe unten).
  2. Fall, $T = N(S)$.
     Dann gilt (siehe Beweis von Hall) $|N(S)| = |T| = |S'| = |S| - 1$, also $|N(S)| < |S|$.



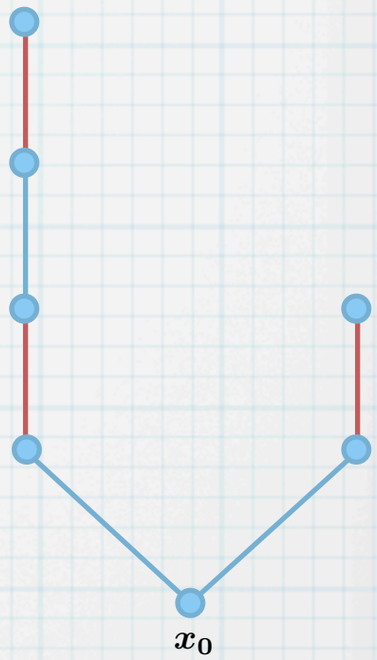
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.
     Damit ist x entweder ungesättigt oder bereits gepaart.
     Also ist die Kante $\{x, y\}$ nicht in M.
     Wird y durch M gesättigt, gibt es eine weitere Kante $\{y, z\} \in M$.
     Der Baum wächst dann durch Hinzufügen von $\{x, y\}$ und $\{y, z\}$.
     Ist y ungesättigt, tritt Fall (b) ein (siehe unten).
  2. Fall, $T = N(S)$.
     Dann gilt (siehe Beweis von Hall) $|N(S)| = |T| = |S'| = |S| - 1$, also $|N(S)| < |S|$.
     Somit enthält der Graph kein Matching, welches jeden Knoten in X sättigt.



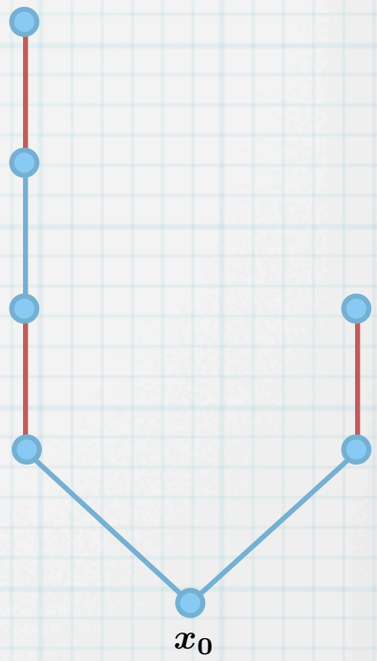
So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.
     Damit ist x entweder ungesättigt oder bereits gepaart.
     Also ist die Kante $\{x, y\}$ nicht in M.
     Wird y durch M gesättigt, gibt es eine weitere Kante $\{y, z\} \in M$.
     Der Baum wächst dann durch Hinzufügen von $\{x, y\}$ und $\{y, z\}$.
     Ist y ungesättigt, tritt Fall (b) ein (siehe unten).
  2. Fall, $T = N(S)$.
     Dann gilt (siehe Beweis von Hall) $|N(S)| = |T| = |S'| = |S| - 1$, also $|N(S)| < |S|$.
     Somit enthält der Graph kein Matching, welches jeden Knoten in X sättigt.
* Fall (b): Der Baum hat einen Weg P, dessen beide Endknoten x_0 und v M–ungesättigt sind.



So wachsen alternierende Bäume

* Alternierende Bäume wachsen so, dass stets eine der beiden Bedingungen erfüllt ist:
  (a) <u>entweder</u> sind alle Knoten von H, mit Ausnahme von x_0, M–gesättigt und in M gepaart,
  (b) <u>oder</u> H enthält einen M–ungesättigten Knoten $v \neq x_0$.
* Fall (a): Bezeichne mit S' die Knoten in H mit gerader Entfernung ($\neq 0$) zu x_0.
  Bezeichne mit T die Knoten in H mit ungerader Entfernung zu x_0.
  Per Induktion sieht man ein, dass dann $|T| = |S'|$ gilt.
  Setze $S := S' \cup \{x_0\}$.
  Bemerke: S, S', T sind wie im Heiratssatz von Hall definiert.
  Es gilt $T \subseteq N(S)$.
  1. Fall, $T \subset N(S)$.
     Dann gibt es einen Knoten $y \in V(G), y \notin T$,
     der adjazent mit einem Knoten $x \in S$ ist.
     Dann ist entweder $x = x_0$ oder $x \in S'$.
     Damit ist x entweder ungesättigt oder bereits gepaart.
     Also ist die Kante $\{x, y\}$ nicht in M.
     Wird y durch M gesättigt, gibt es eine weitere Kante $\{y, z\} \in M$.
     Der Baum wächst dann durch Hinzufügen von $\{x, y\}$ und $\{y, z\}$.
     Ist y ungesättigt, tritt Fall (b) ein (siehe unten).
  2. Fall, $T = N(S)$.
     Dann gilt (siehe Beweis von Hall) $|N(S)| = |T| = |S'| = |S| - 1$, also $|N(S)| < |S|$.
     Somit enthält der Graph kein Matching, welches jeden Knoten in X sättigt.
* Fall (b): Der Baum hat einen Weg P, dessen beide Endknoten x_0 und v M–ungesättigt sind.
  Durch Umfärben entlang P erhält man ein vergrößertes Matching.



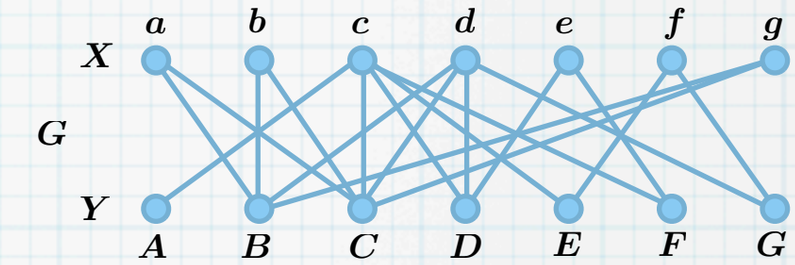
Der Ungarische Algorithmus

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.

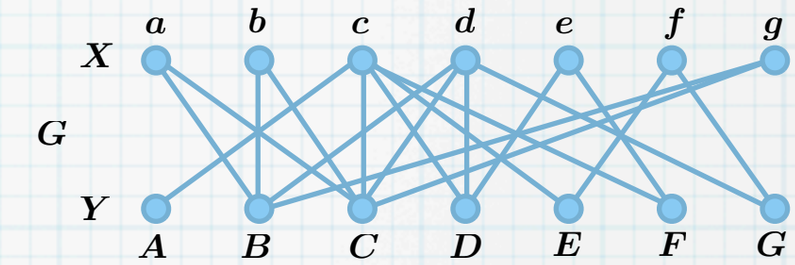
Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.



Der Ungarische Algorithmus

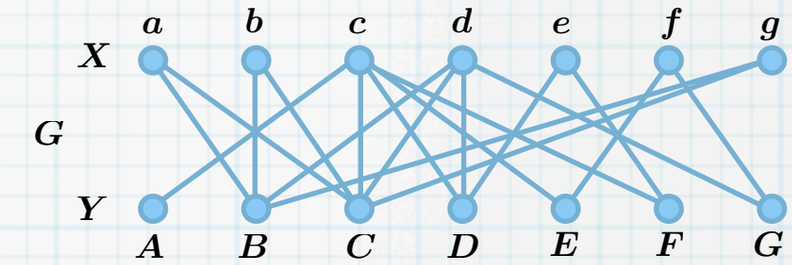
* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:



Der Ungarische Algorithmus

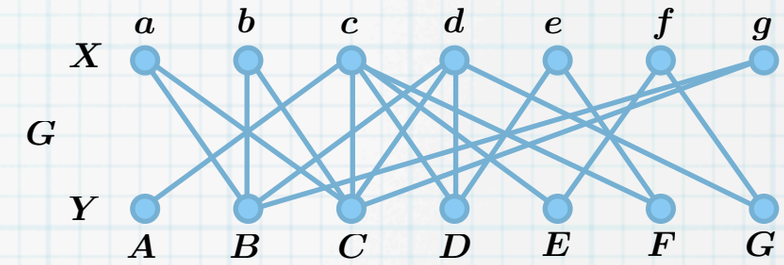
* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:

  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,



Der Ungarische Algorithmus

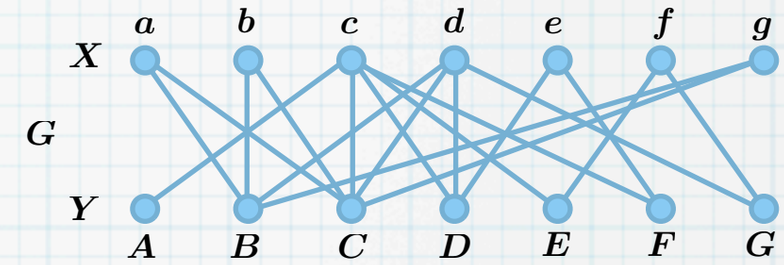
* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

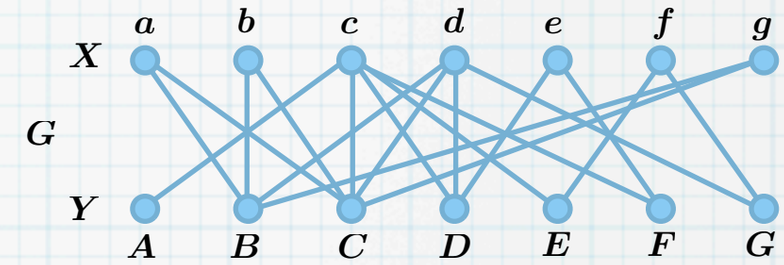
(1) **algorithm** hungarian



Der Ungarische Algorithmus

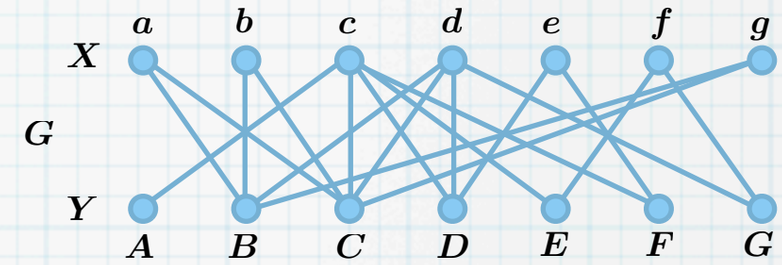
* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

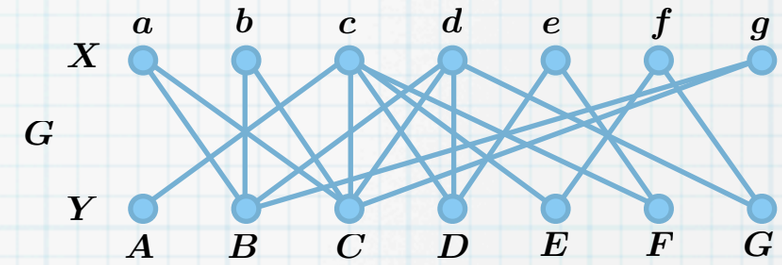


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

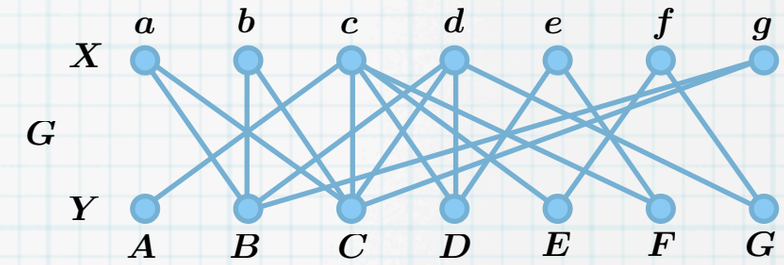
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M

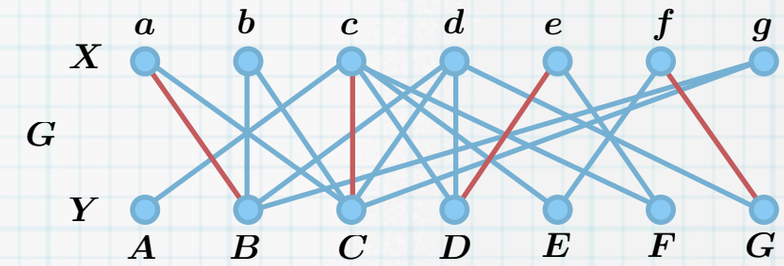


$M := \{aB, cC, eD, fG\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M

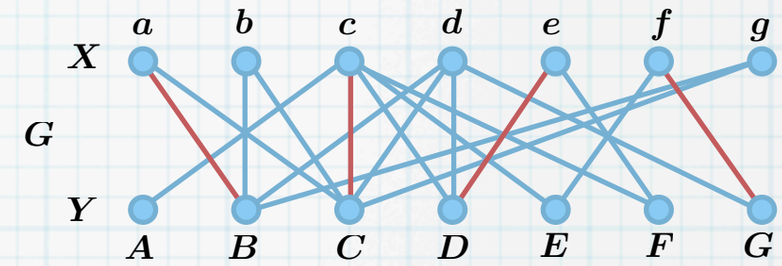


$M := \{aB, cC, eD, fG\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**

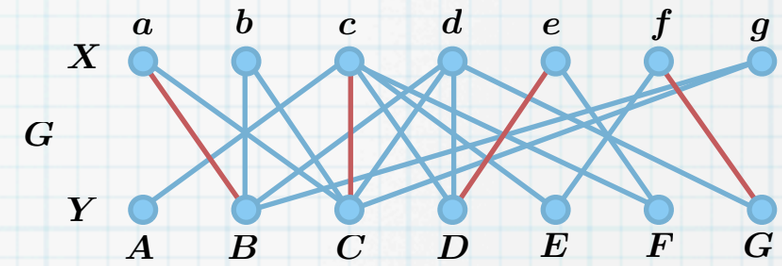


$M := \{aB, cC, eD, fG\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**

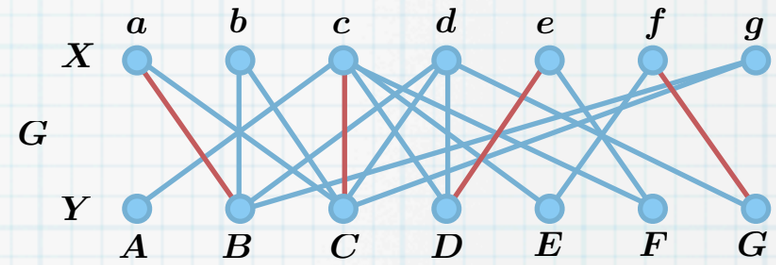


$M := \{aB, cC, eD, fG\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**



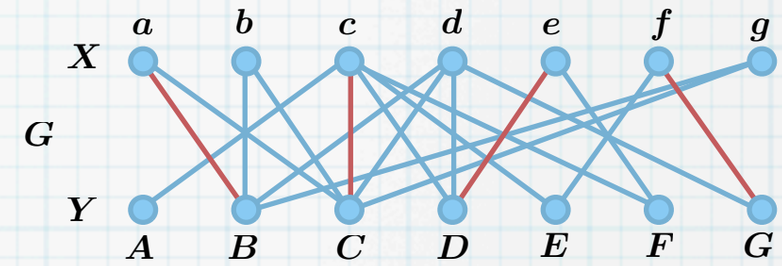
$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$



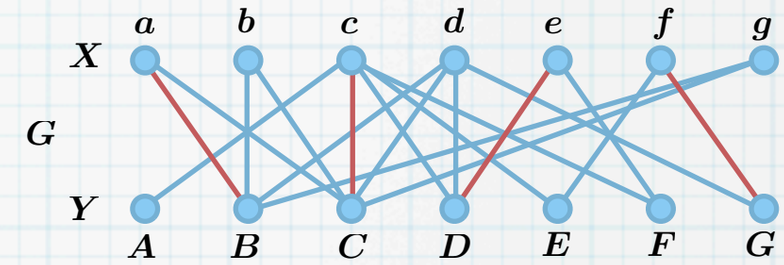
$M := \{aB, cC, eD, fG\}$

$x_0 = b$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$



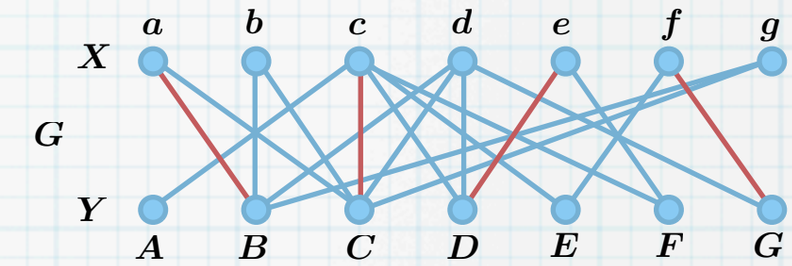
$M := \{aB, cC, eD, fG\}$

$x_0 = b$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$



$M := \{aB, cC, eD, fG\}$

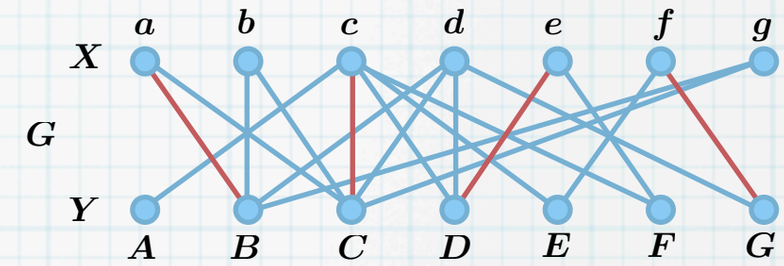
$x_0 = b$

$S = \{b\}, T = \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$



$M := \{aB, cC, eD, fG\}$

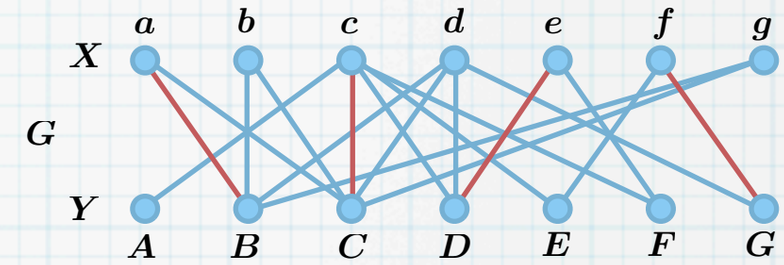
$x_0 = b$

$S = \{b\}, T = \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**



$M := \{aB, cC, eD, fG\}$

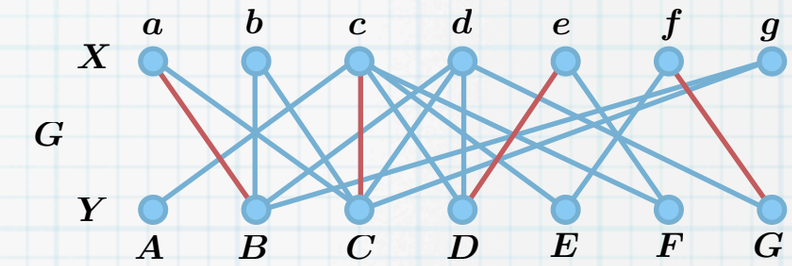
$x_0 = b$

$S = \{b\}, T = \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**



$M := \{aB, cC, eD, fG\}$

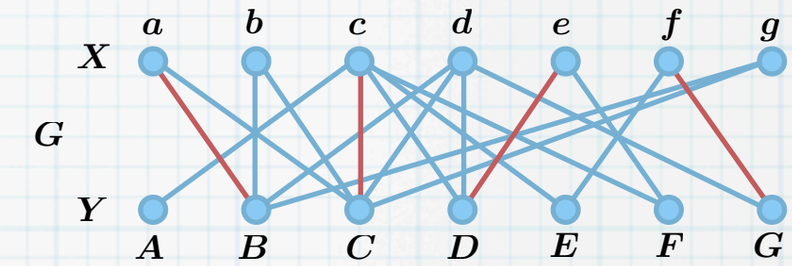
$x_0 = b$

$S = \{b\}, T = \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**



$M := \{aB, cC, eD, fG\}$

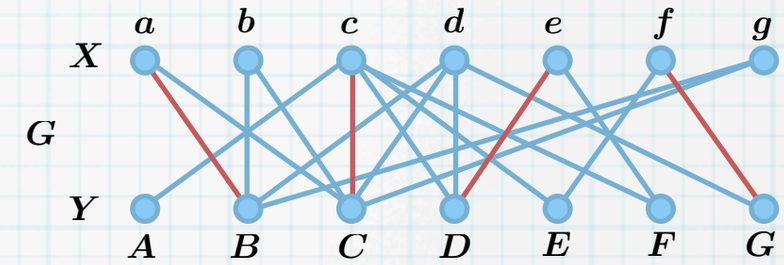
$x_0 = b$

$S = \{b\}, T = \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**



$M := \{aB, cC, eD, fG\}$

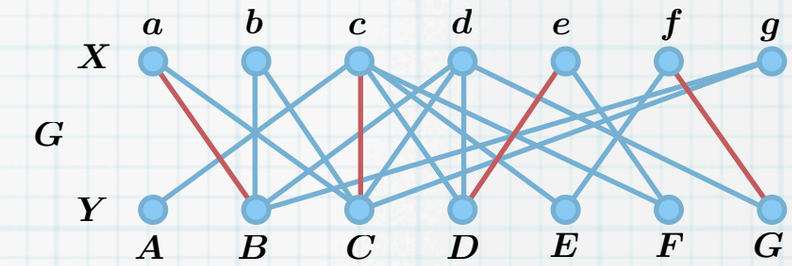
$x_0 = b$

$S = \{b\}, T = \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**



$M := \{aB, cC, eD, fG\}$

$x_0 = b$

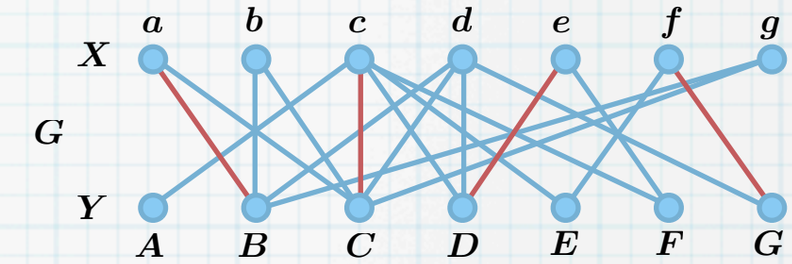
$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus



$M := \{aB, cC, eD, fG\}$

$x_0 = b$

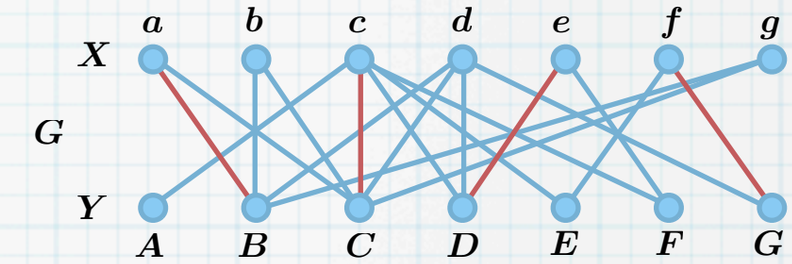
$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)



$M := \{aB, cC, eD, fG\}$

$x_0 = b$

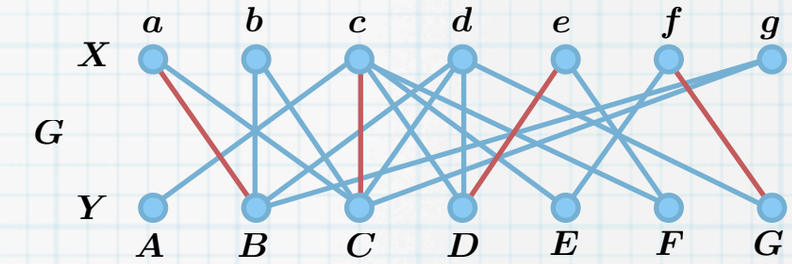
$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**



$M := \{aB, cC, eD, fG\}$

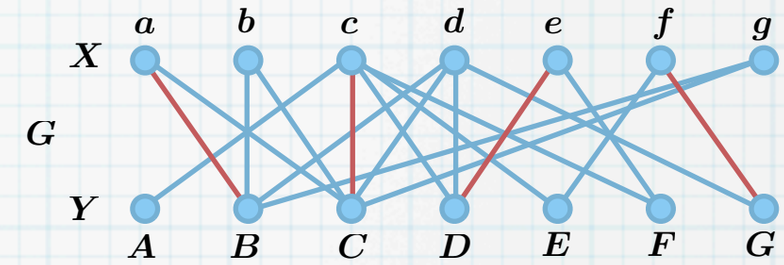
$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

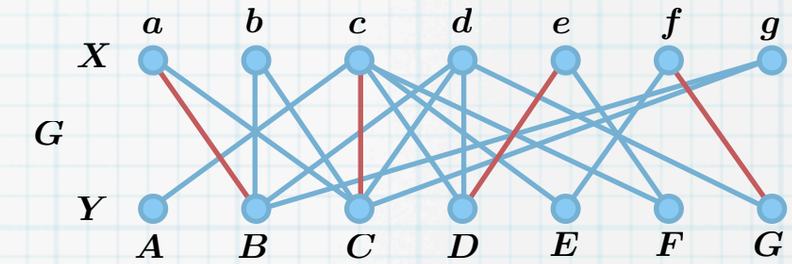
$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$



$M := \{aB, cC, eD, fG\}$

$x_0 = b$

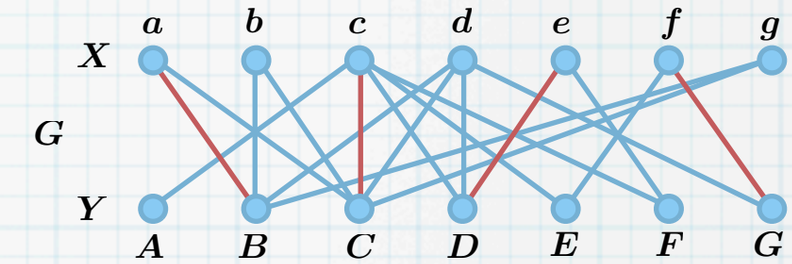
$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$



$M := \{aB, cC, eD, fG\}$

$x_0 = b$

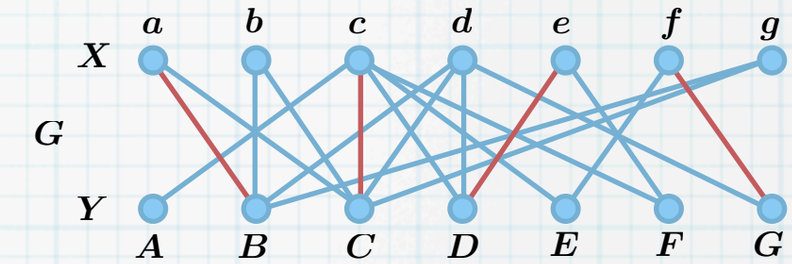
$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$



$M := \{aB, cC, eD, fG\}$

$x_0 = b$

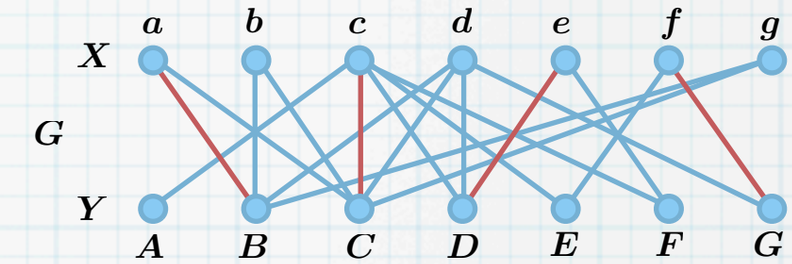
$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

$y = B$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

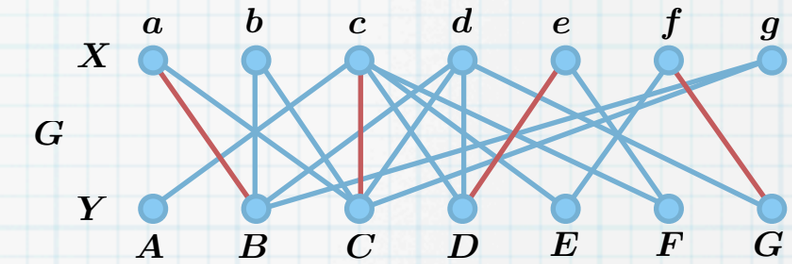
$\{B, C\} \neq \{\}$

$y = B$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$



$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

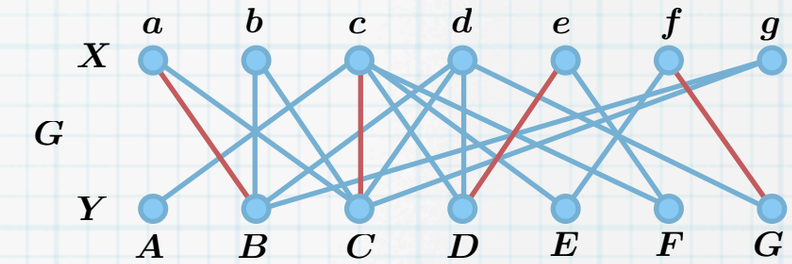
$\{B, C\} \neq \{\}$

$y = B$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$



$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

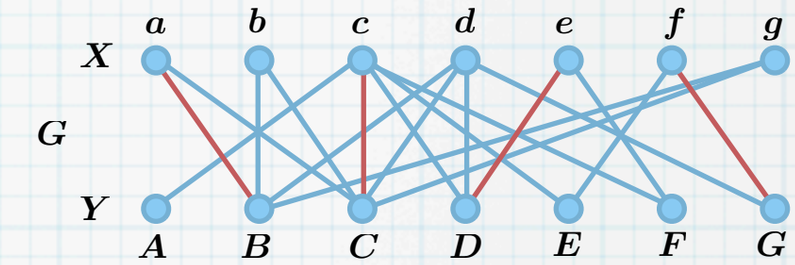
$\{B, C\} \neq \{\}$

$y = B$

$T = \{B\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



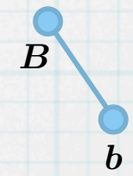
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

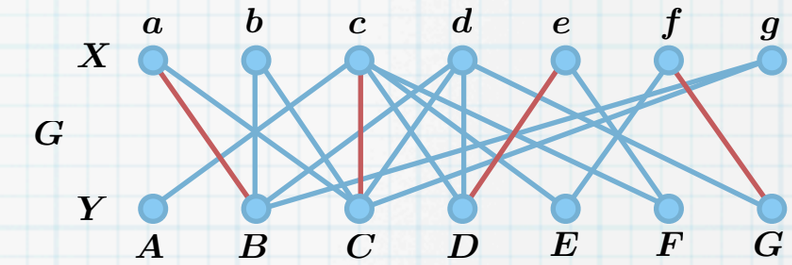


$y = B$

$T = \{B\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



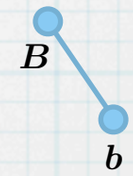
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

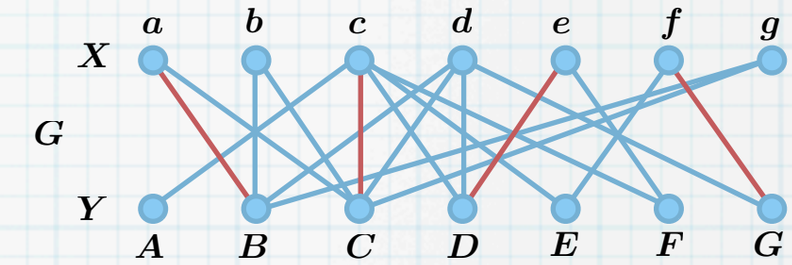


$y = B$

$T = \{B\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



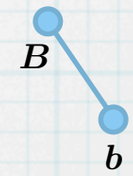
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$

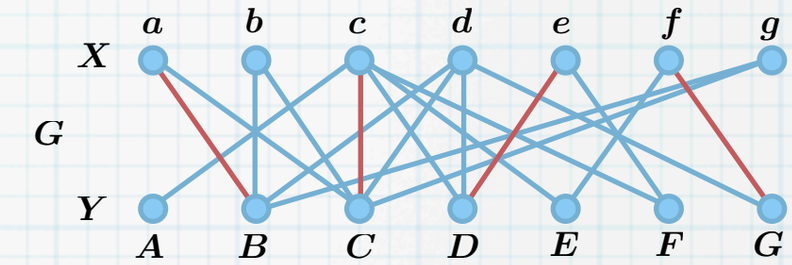


$y = B$

$T = \{B\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



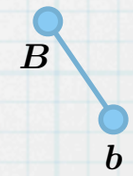
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



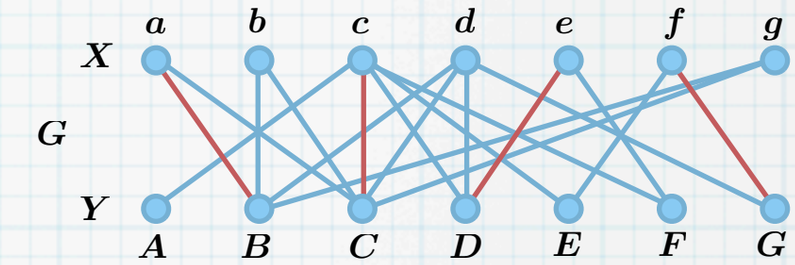
$y = B$

$T = \{B\}$

$z = a$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



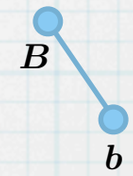
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



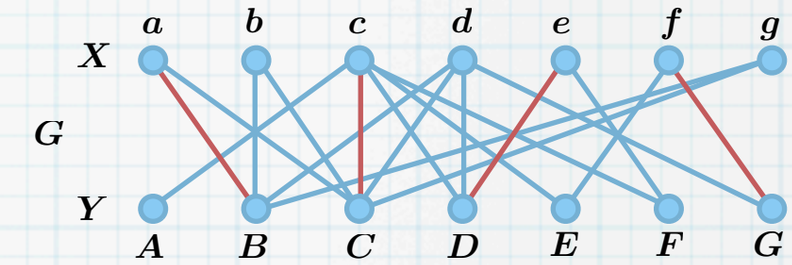
$y = B$

$T = \{B\}$

$z = a$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



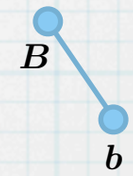
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



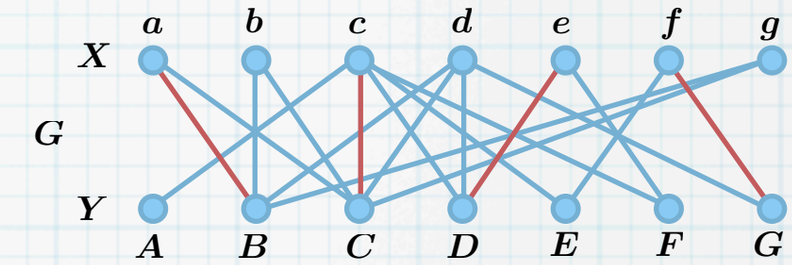
$y = B$

$T = \{B\}$

$z = a$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



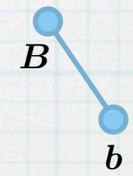
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



$y = B$

$T = \{B\}$

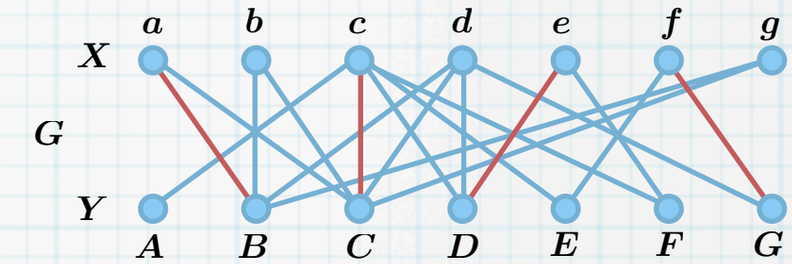
$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$

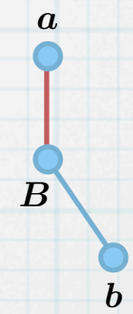


$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



$y = B$

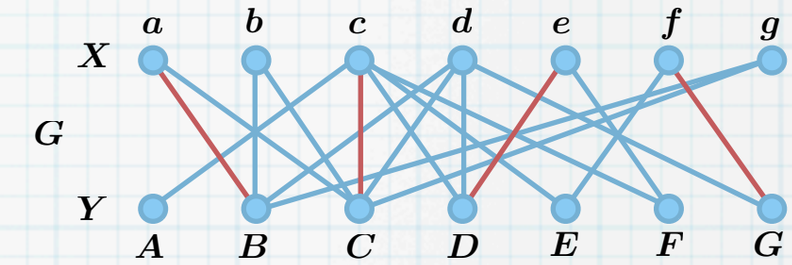
$T = \{B\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



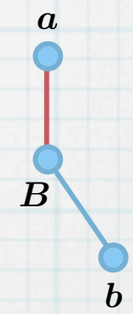
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



$y = B$

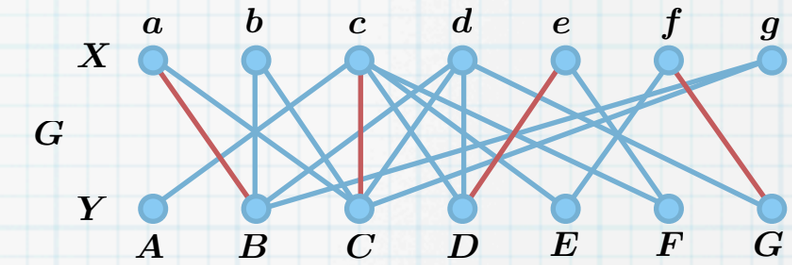
$T = \{B\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



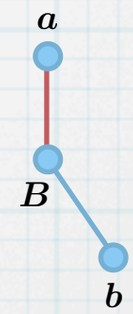
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



$y = B$

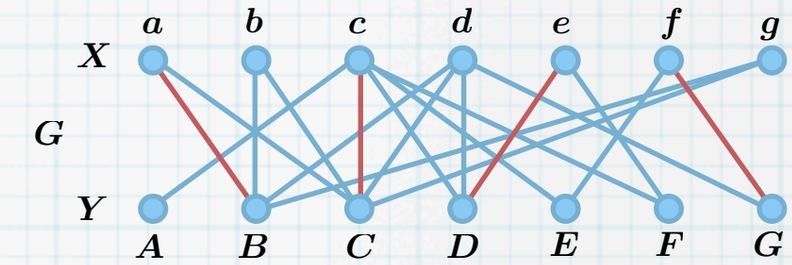
$T = \{B\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



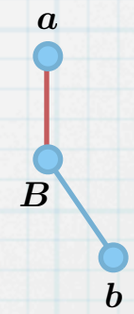
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



$y = B$

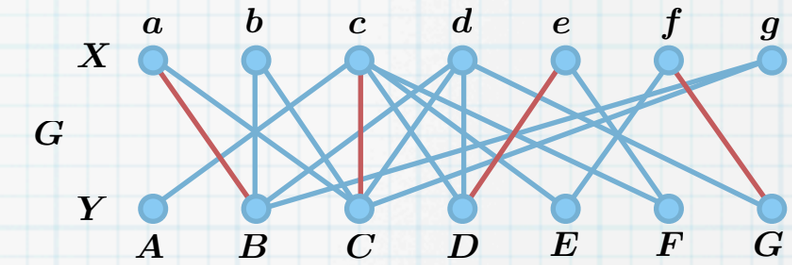
$T = \{B\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



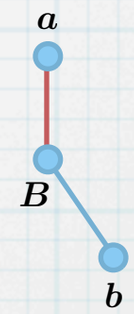
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



$y = B$

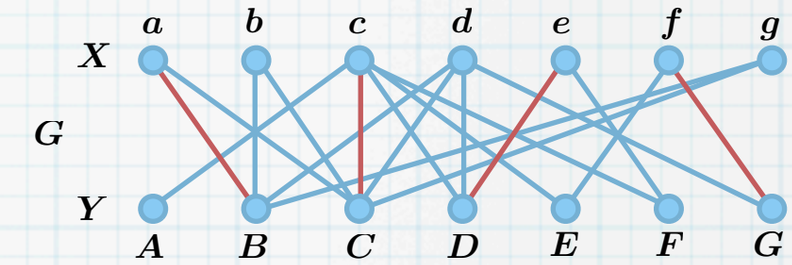
$T = \{B\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



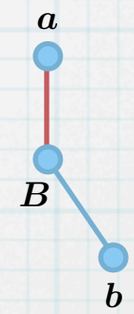
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



$y = B$

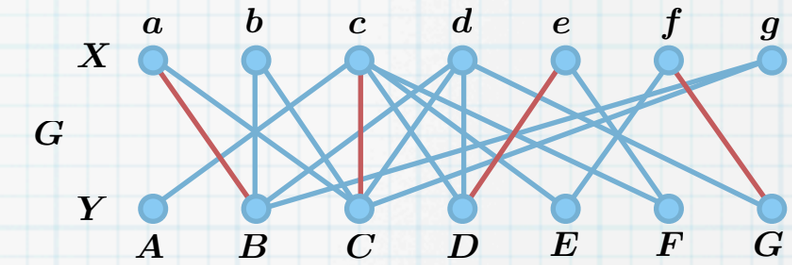
$T = \{B\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



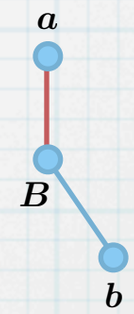
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



$y = B$

$T = \{B\}$

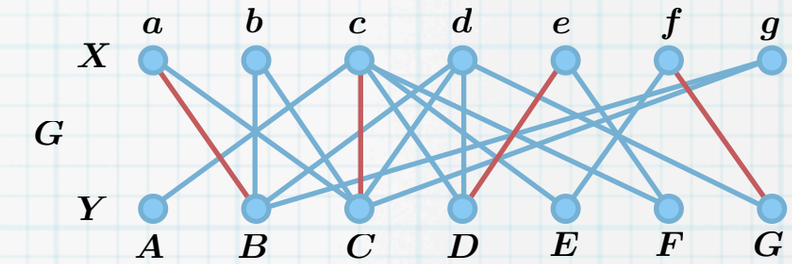
$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

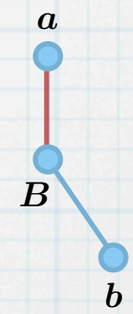


$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



$y = B$

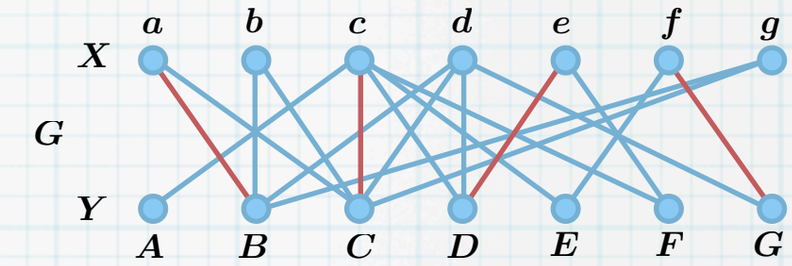
$T = \{B\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



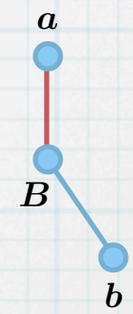
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{\}$



$y = B$

$T = \{B\}$

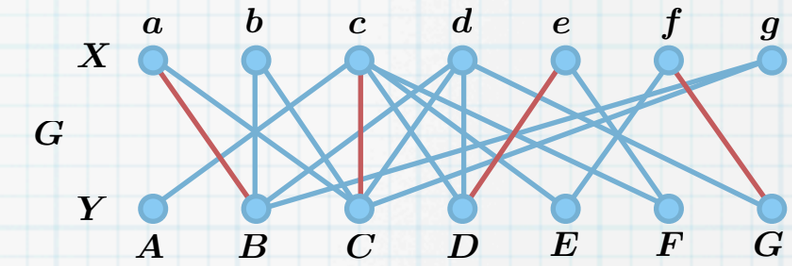
$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

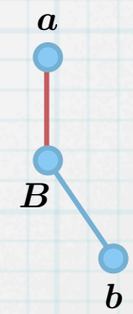


$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$



$$y = B$$

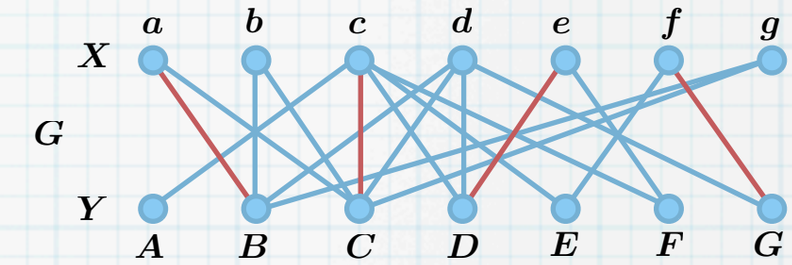
$$T = \{B\}$$

$$z = a$$

$$S = \{a, b\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



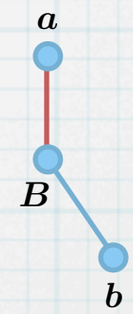
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = B$

$T = \{B\}$

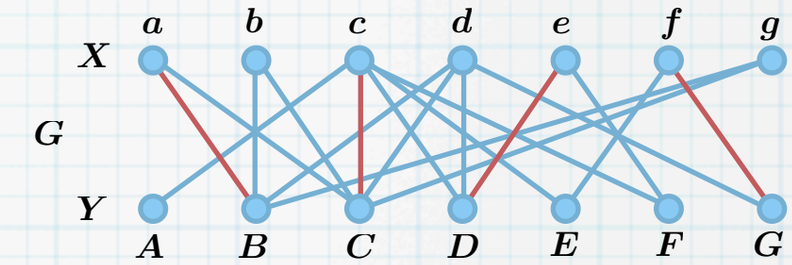
$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

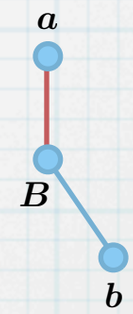


$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$



$$y = B$$

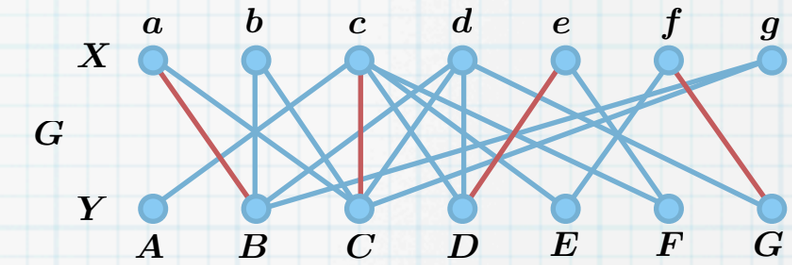
$$T = \{B\}$$

$$z = a$$

$$S = \{a, b\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



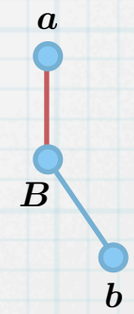
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

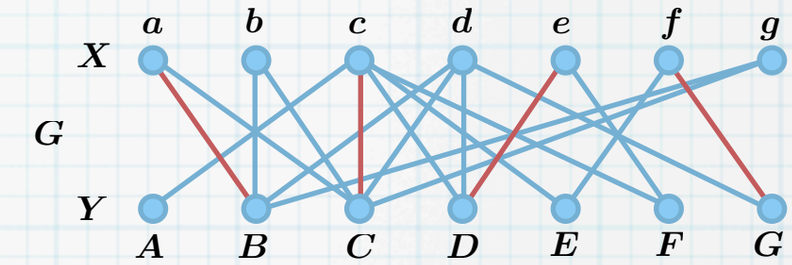
$T = \{B\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



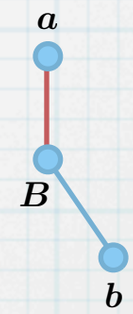
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

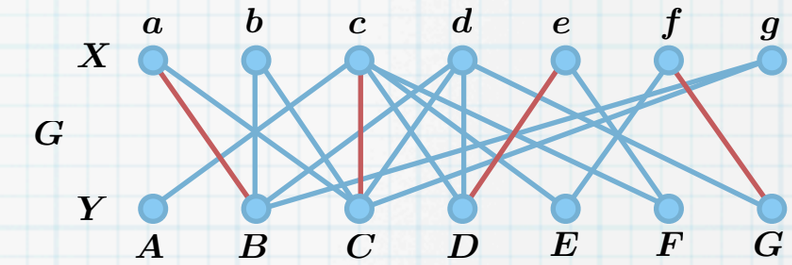
$T = \{B\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



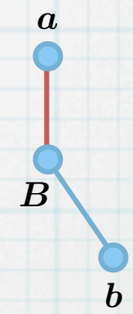
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

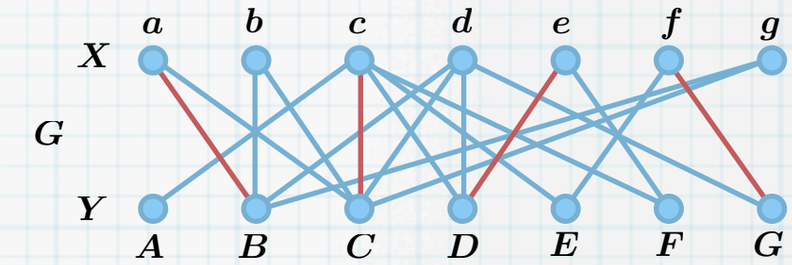
$T = \{B, C\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



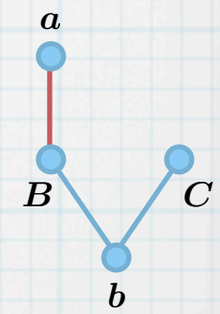
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$



$$y = C$$

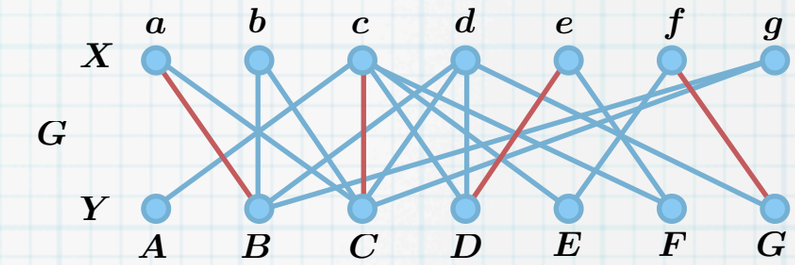
$$T = \{B, C\}$$

$$z = a$$

$$S = \{a, b\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



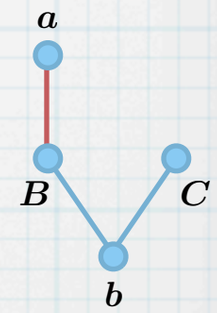
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

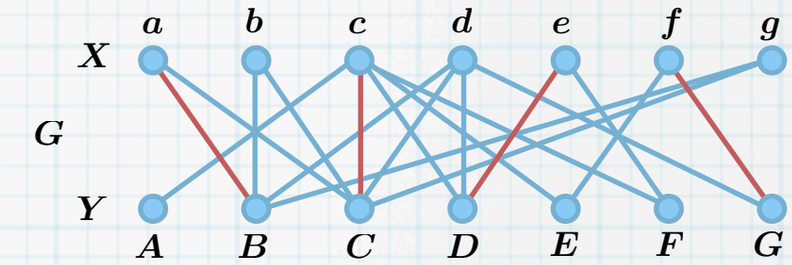
$T = \{B, C\}$

$z = a$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



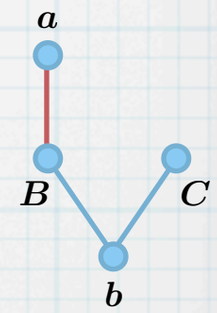
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

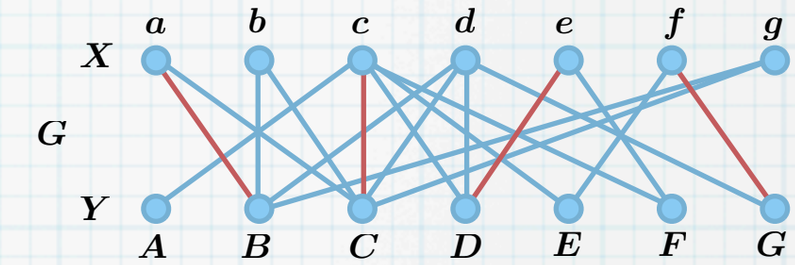
$T = \{B, C\}$

$z = c$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



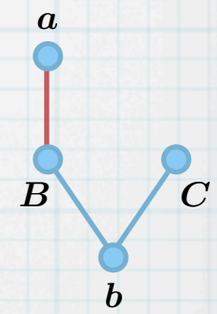
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

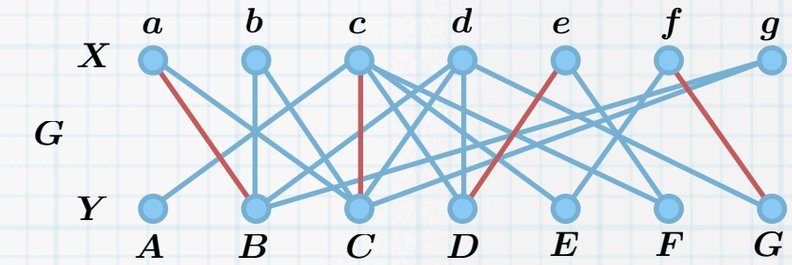
$T = \{B, C\}$

$z = c$

$S = \{a, b\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



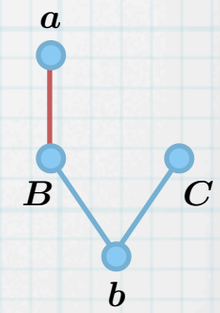
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

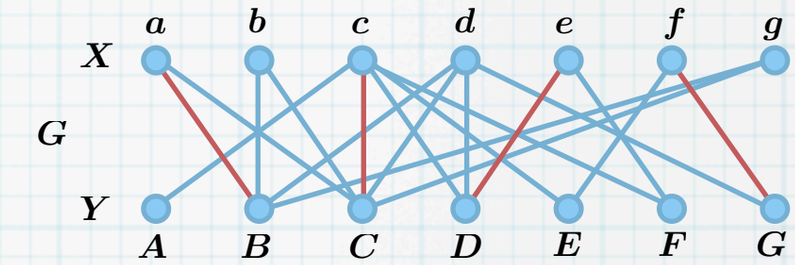
$T = \{B, C\}$

$z = c$

$S = \{a, b, c\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



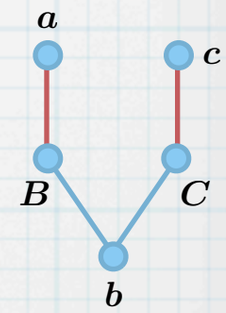
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

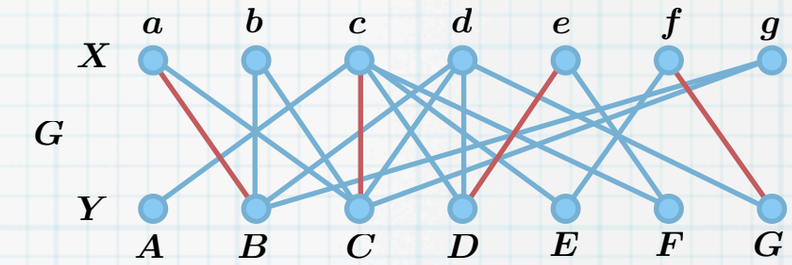
$T = \{B, C\}$

$z = c$

$S = \{a, b, c\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



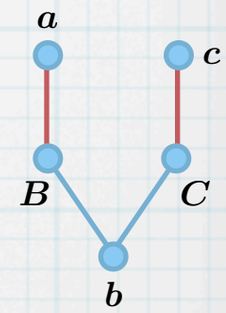
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

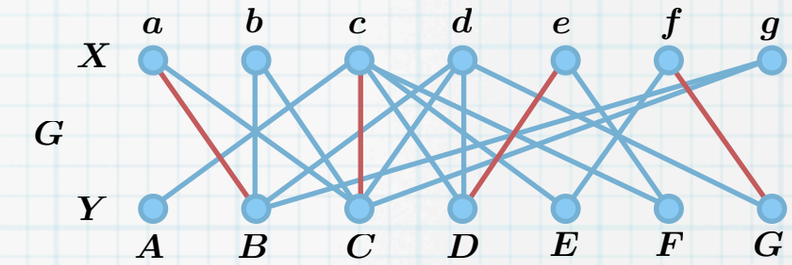
$T = \{B, C\}$

$z = c$

$S = \{a, b, c\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



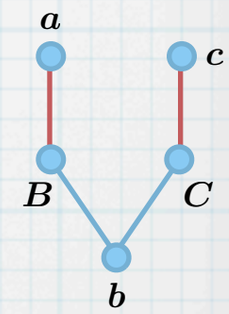
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$



$$y = C$$

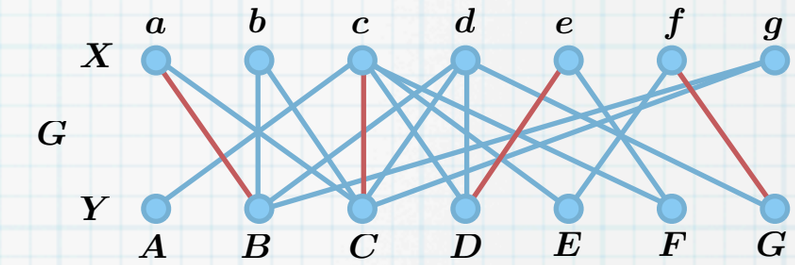
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



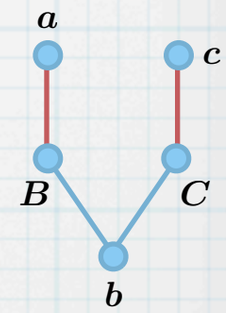
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

$T = \{B, C\}$

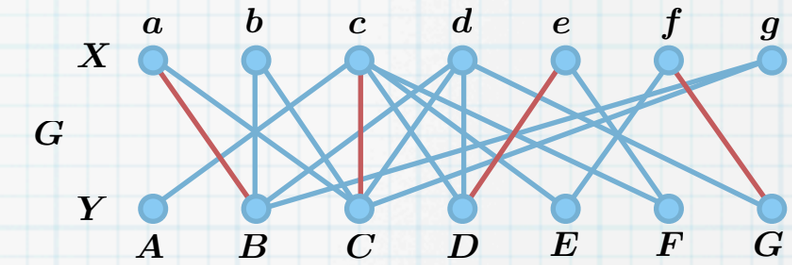
$z = c$

$S = \{a, b, c\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

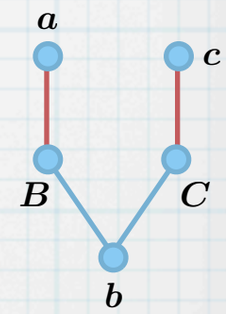


$M := \{aB, cC, eD, fG\}$

$x_0 = b$

$S = \{b\}, T = \{\}$

$\{B, C\} \neq \{B\}$



$y = C$

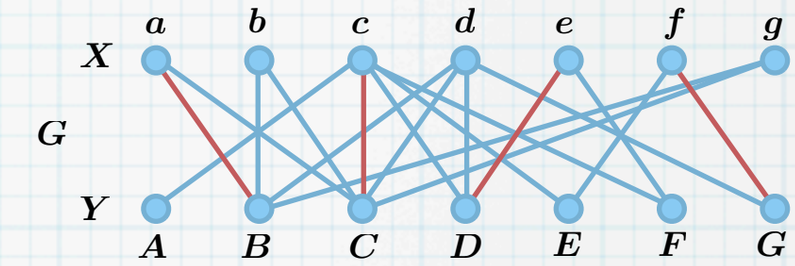
$T = \{B, C\}$

$z = c$

$S = \{a, b, c\}$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



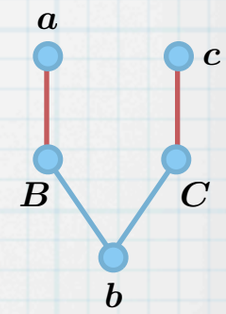
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$



$$y = C$$

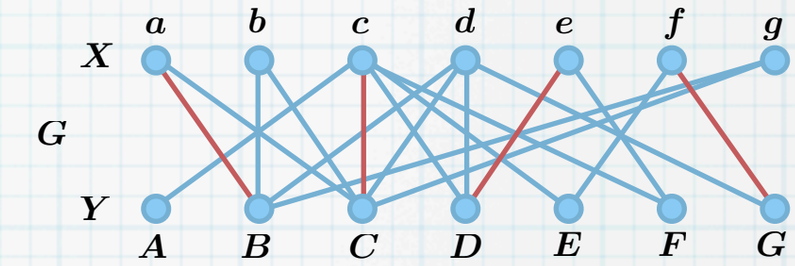
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



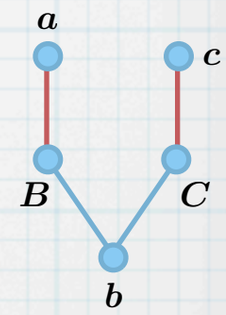
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = C$$

$$T = \{B, C\}$$

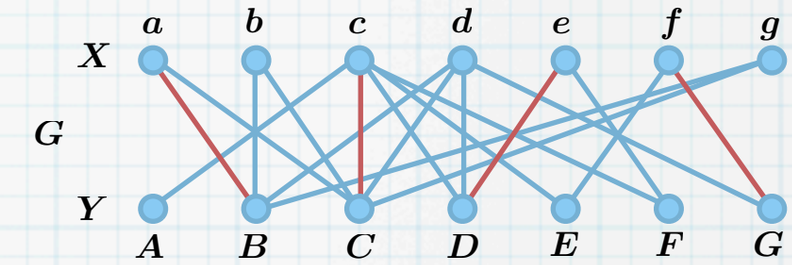
$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

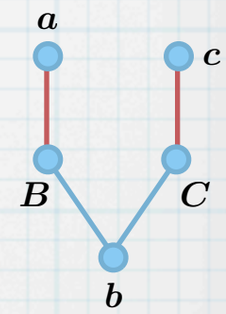


$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = C$$

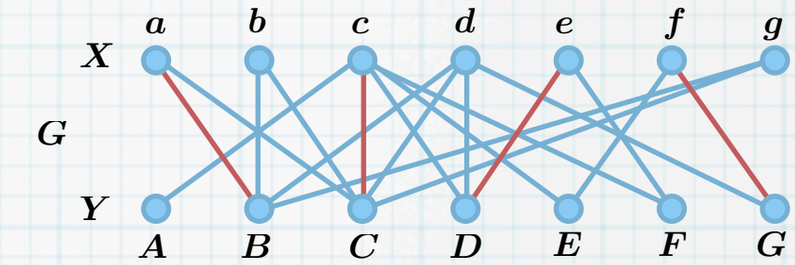
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



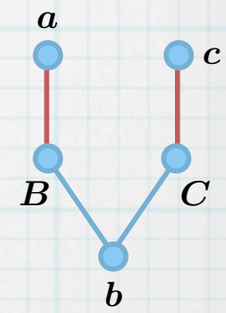
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = C$$

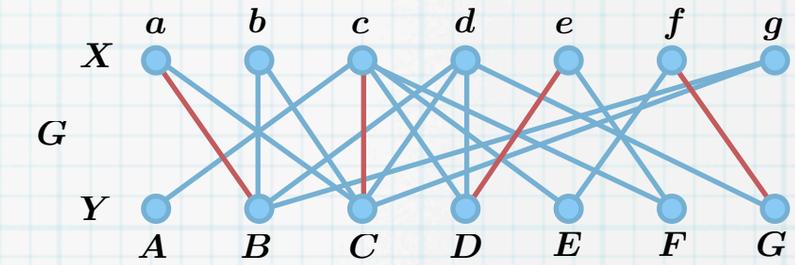
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



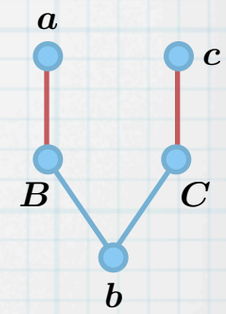
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

$$T = \{B, C\}$$

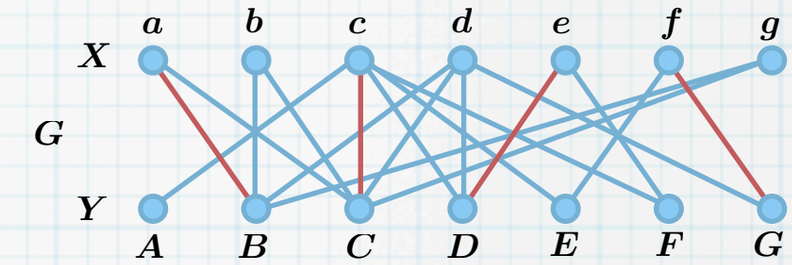
$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

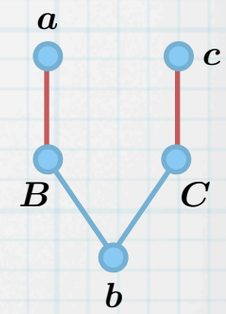


$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

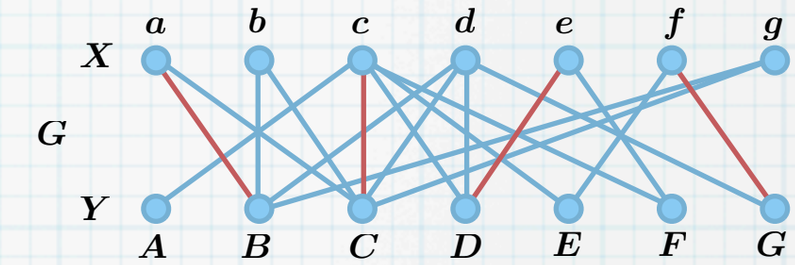
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



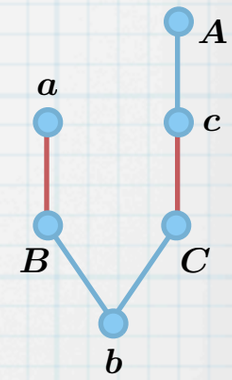

```
(1)  algorithm hungarian
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      S := {x_0}, T := ∅
(5)      repeat
(6)        if N(S) = T then
(7)          gib S aus
(8)          goto (19)
(9)        else
(10)         wähle y ∈ N(S)\T
(11)         T := T ∪ {y}
(12)         if es gibt ein z mit {y, z} ∈ M then
(13)           S := S ∪ {z}
(14)         end if
(15)       end if
(16)     until y ungesättigt
```

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

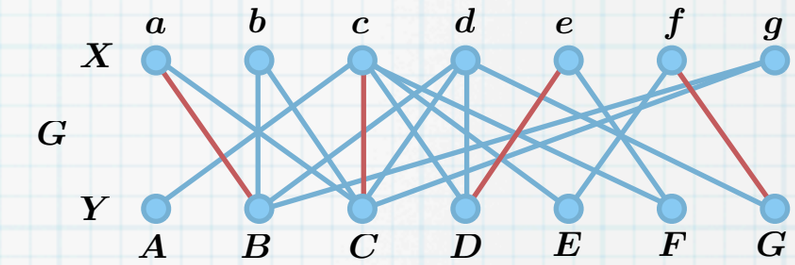
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



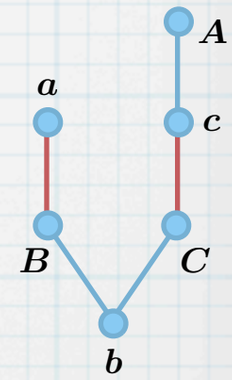
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

$$T = \{B, C\}$$

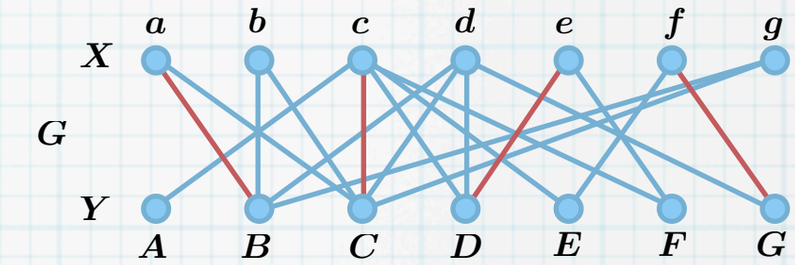
$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

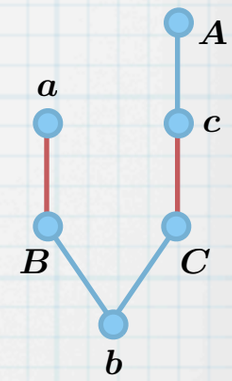


$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

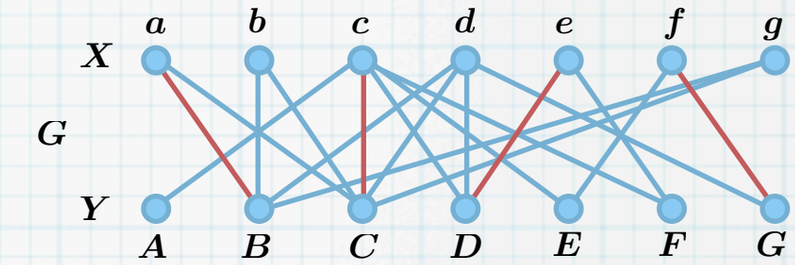
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



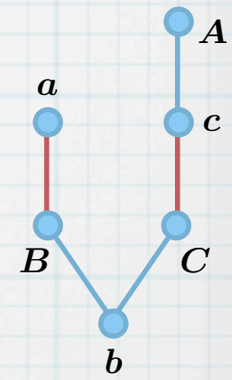
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

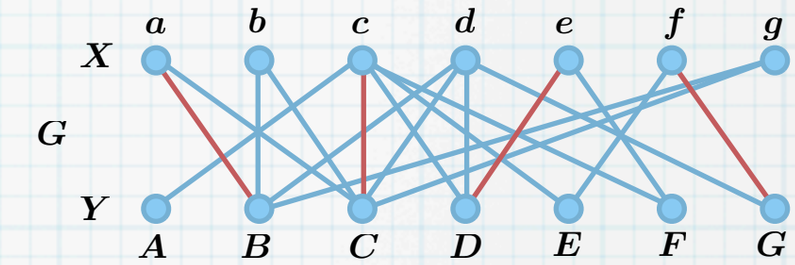
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



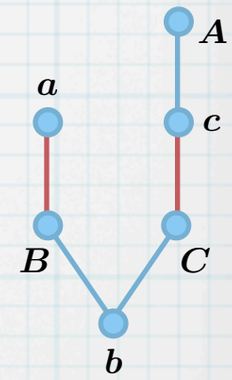

```
(1)  algorithm hungarian
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      S := {x_0}, T := ∅
(5)      repeat
(6)        if N(S) = T then
(7)          gib S aus
(8)          goto (19)
(9)        else
(10)         wähle y ∈ N(S)\T
(11)         T := T ∪ {y}
(12)         if es gibt ein z mit {y, z} ∈ M then
(13)           S := S ∪ {z}
(14)         end if
(15)       end if
(16)     until y ungesättigt
```

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

$$T = \{B, C\}$$

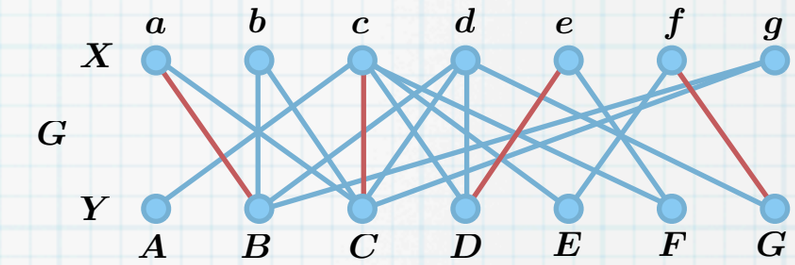
$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um

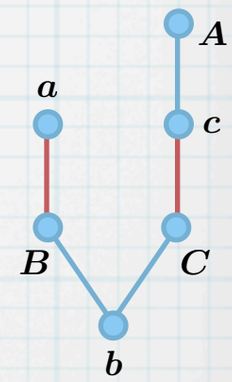


$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

$$T = \{B, C\}$$

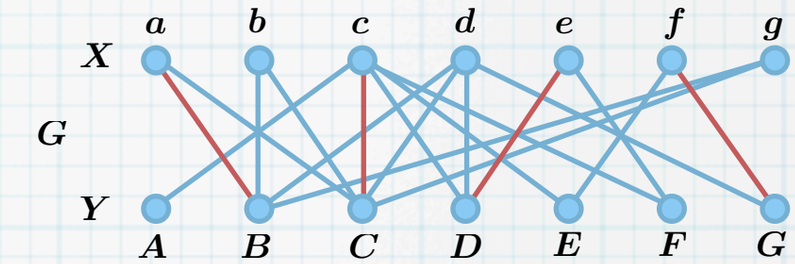
$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um

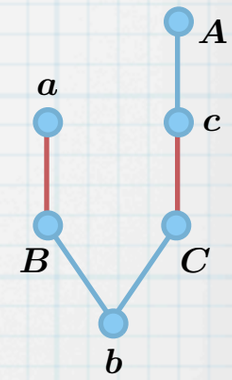


$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

$$T = \{B, C\}$$

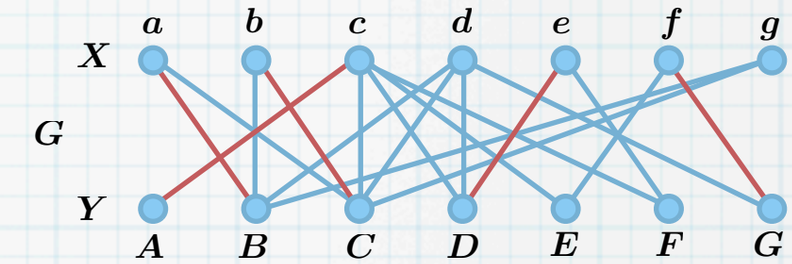
$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

```
(1)  algorithm hungarian
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      S := {x_0}, T := ∅
(5)      repeat
(6)        if N(S) = T then
(7)          gib S aus
(8)          goto (19)
(9)        else
(10)         wähle y ∈ N(S)\T
(11)         T := T ∪ {y}
(12)         if es gibt ein z mit {y, z} ∈ M then
(13)           S := S ∪ {z}
(14)         end if
(15)       end if
(16)     until y ungesättigt
(17)     färbe M entlang des Weges von x_0 nach y um
```

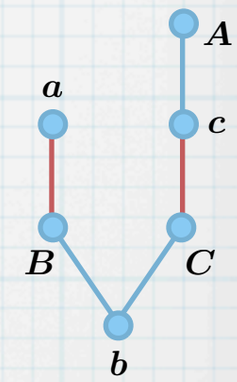


$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

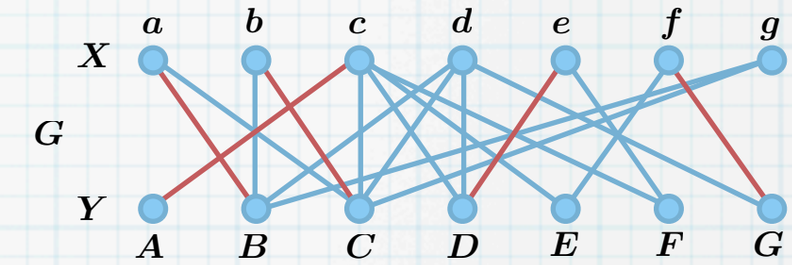
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



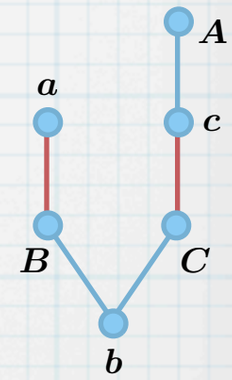
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

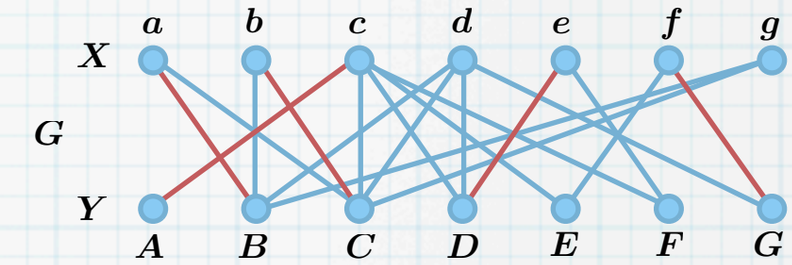
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



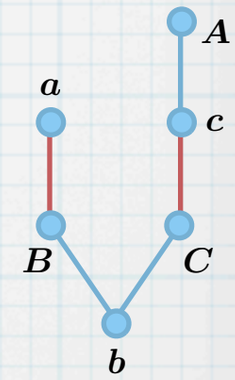

```
(1)  algorithm hungarian
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      S := {x_0}, T := ∅
(5)      repeat
(6)        if N(S) = T then
(7)          gib S aus
(8)          goto (19)
(9)        else
(10)         wähle y ∈ N(S)\T
(11)         T := T ∪ {y}
(12)         if es gibt ein z mit {y, z} ∈ M then
(13)           S := S ∪ {z}
(14)         end if
(15)       end if
(16)     until y ungesättigt
(17)     färbe M entlang des Weges von x_0 nach y um
(18)   end while
```

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

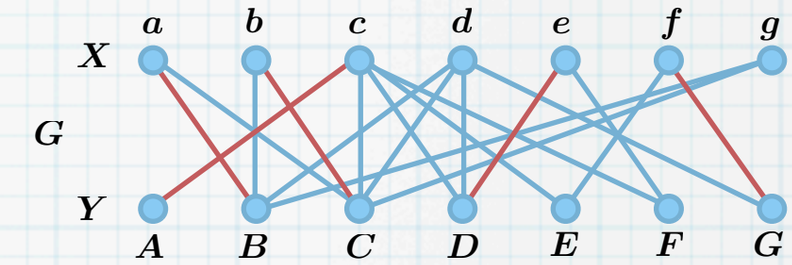
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



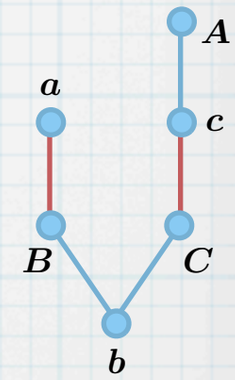

```
(1)  algorithm hungarian
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      S := {x_0}, T := ∅
(5)      repeat
(6)        if N(S) = T then
(7)          gib S aus
(8)          goto (19)
(9)        else
(10)         wähle y ∈ N(S)\T
(11)         T := T ∪ {y}
(12)         if es gibt ein z mit {y, z} ∈ M then
(13)           S := S ∪ {z}
(14)         end if
(15)       end if
(16)     until y ungesättigt
(17)     färbe M entlang des Weges von x_0 nach y um
(18)   end while
```

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = b$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

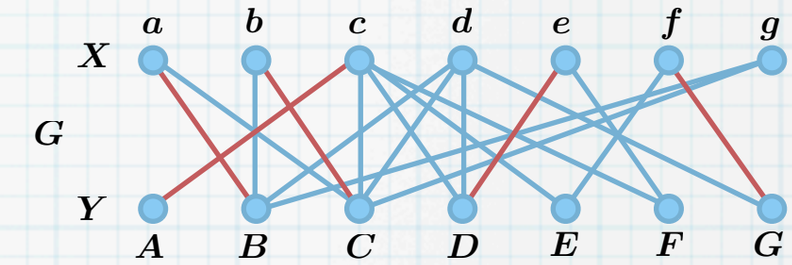
$$T = \{B, C\}$$

$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



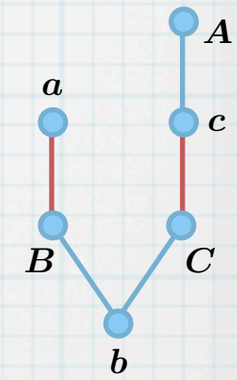

```
(1)  algorithm hungarian
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      S := {x_0}, T := ∅
(5)      repeat
(6)        if N(S) = T then
(7)          gib S aus
(8)          goto (19)
(9)        else
(10)         wähle y ∈ N(S)\T
(11)         T := T ∪ {y}
(12)         if es gibt ein z mit {y, z} ∈ M then
(13)           S := S ∪ {z}
(14)         end if
(15)       end if
(16)     until y ungesättigt
(17)     färbe M entlang des Weges von x_0 nach y um
(18)   end while
```

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

$$T = \{B, C\}$$

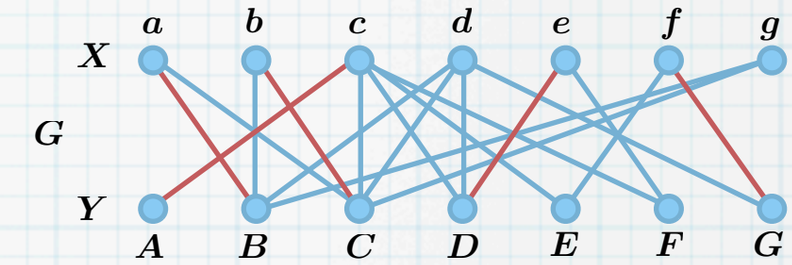
$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

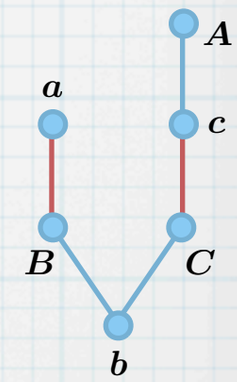


$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{b\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

$$T = \{B, C\}$$

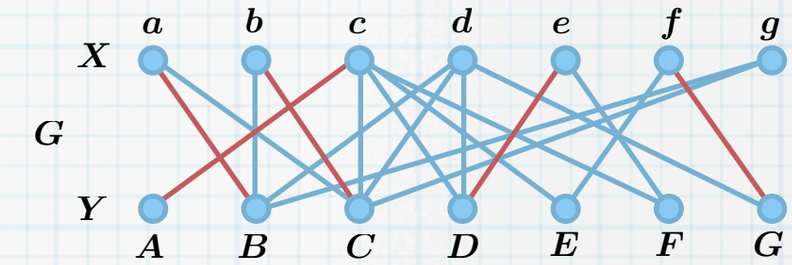
$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

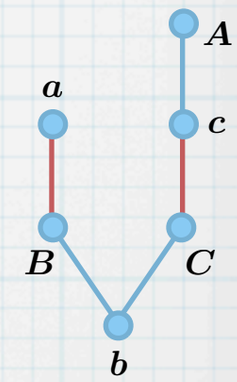


$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$



$$y = A$$

$$T = \{B, C\}$$

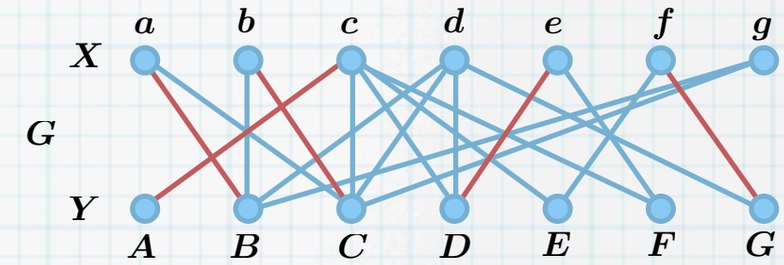
$$z = c$$

$$S = \{a, b, c\}$$

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$

$$y = A$$

$$T = \{B, C\}$$

$$z = c$$

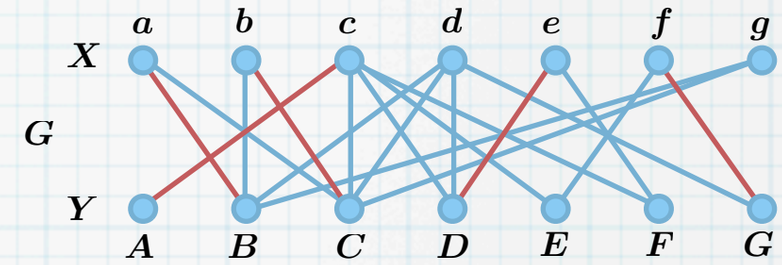
$$S = \{a, b, c\}$$

d

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$

$$y = A$$

$$T = \{B, C\}$$

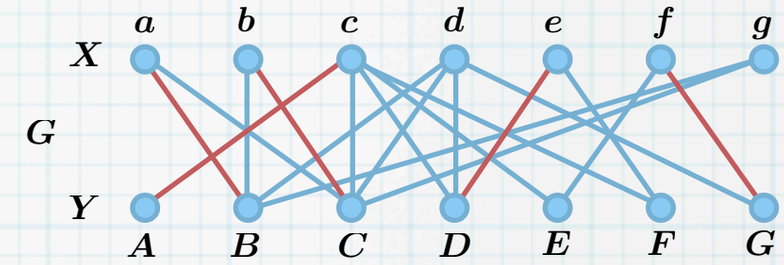
$$z = c$$

$$S = \{a, b, c\}$$

d

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.




```
(1)  algorithm hungarian
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      S := {x_0}, T := ∅
(5)      repeat
(6)        if N(S) = T then
(7)          gib S aus
(8)          goto (19)
(9)        else
(10)         wähle y ∈ N(S)\T
(11)         T := T ∪ {y}
(12)         if es gibt ein z mit {y, z} ∈ M then
(13)           S := S ∪ {z}
(14)         end if
(15)       end if
(16)     until y ungesättigt
(17)     färbe M entlang des Weges von x_0 nach y um
(18)   end while
```

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

$$\{A, B, C, D, E, F\} \neq \{B, C\}$$

$$y = A$$

$$T = \{B, C\}$$

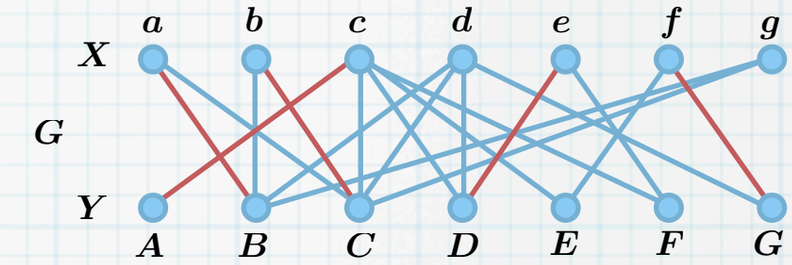
$$z = c$$

$$S = \{a, b, c\}$$

d

Der Ungarische Algorithmus

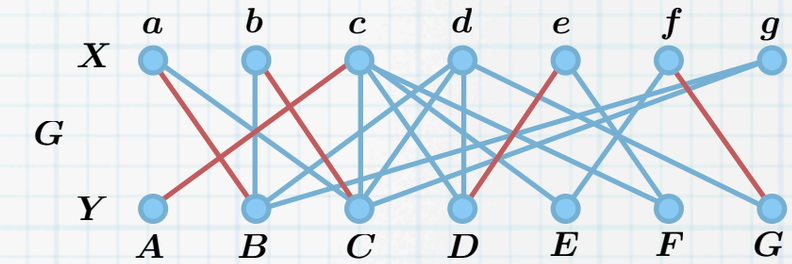
* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = A$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = c$
(13) $S := S \cup \{z\}$ — $S = \{a, b, c\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

Der Ungarische Algorithmus

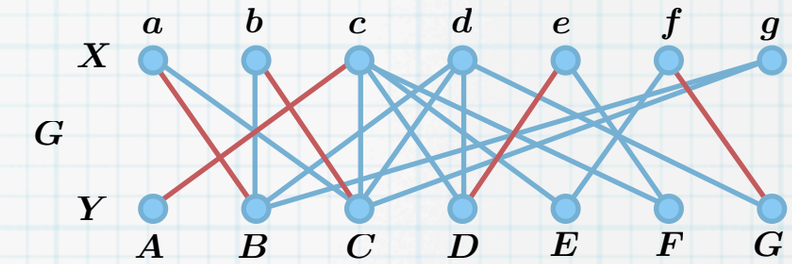
* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = A$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = c$
(13) $S := S \cup \{z\}$ — $S = \{a, b, c\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

$\{B, C, D, G\} \neq \{\}$

$y = A$

$T = \{B, C\}$

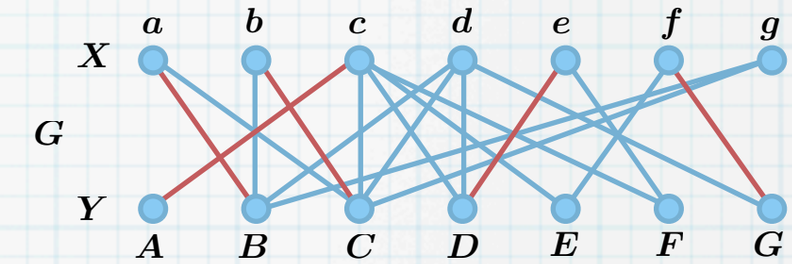
$z = c$

$S = \{a, b, c\}$

d

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

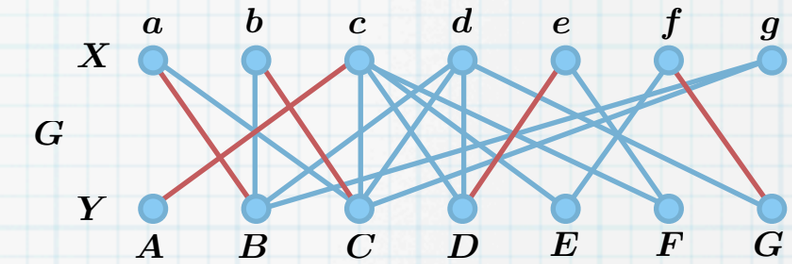


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = c$
(13) $S := S \cup \{z\}$ — $S = \{a, b, c\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

d

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.

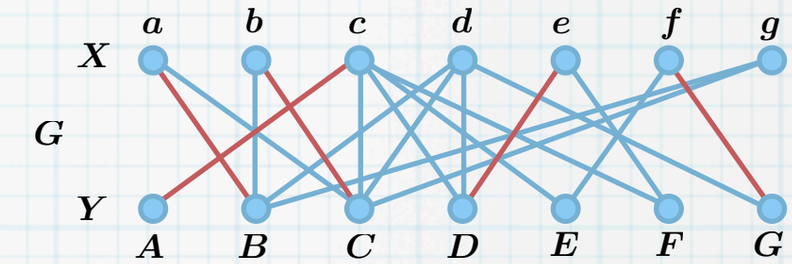


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = c$
(13) $S := S \cup \{z\}$ — $S = \{a, b, c\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

d

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

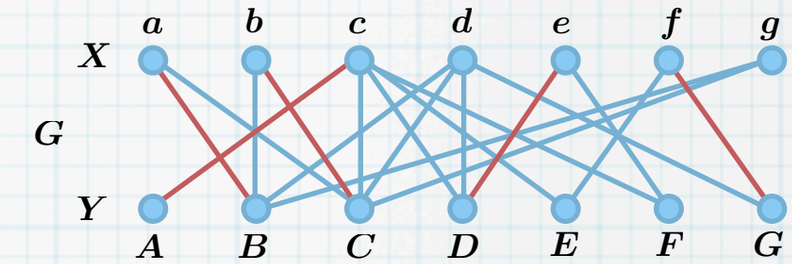


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = c$
(13) $S := S \cup \{z\}$ — $S = \{a, b, c\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

d

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.




```
(1)  algorithm hungarian
(2)    wähle ein beliebiges Matching M
(3)    while es gibt einen M–ungesättigten Knoten x_0 do
(4)      S := {x_0}, T := ∅
(5)      repeat
(6)        if N(S) = T then
(7)          gib S aus
(8)          goto (19)
(9)        else
(10)         wähle y ∈ N(S)\T
(11)         T := T ∪ {y}
(12)         if es gibt ein z mit {y, z} ∈ M then
(13)           S := S ∪ {z}
(14)         end if
(15)       end if
(16)     until y ungesättigt
(17)     färbe M entlang des Weges von x_0 nach y um
(18)   end while
```

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

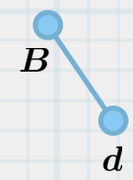
$\{B, C, D, G\} \neq \{\}$

$y = B$

$T = \{B\}$

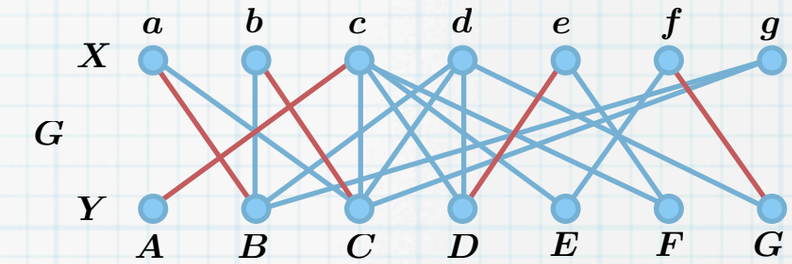
$z = c$

$S = \{a, b, c\}$

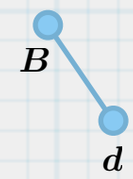


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



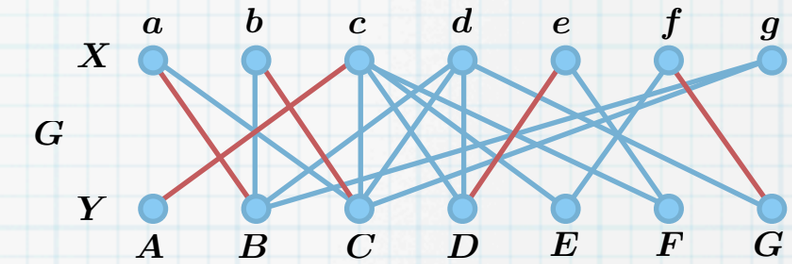
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = c$
(13) $S := S \cup \{z\}$ — $S = \{a, b, c\}$



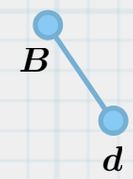
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



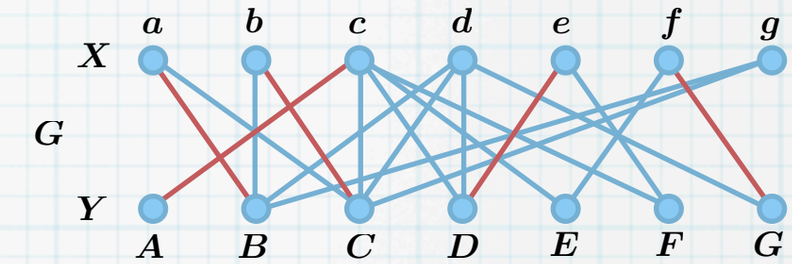
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, b, c\}$



(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

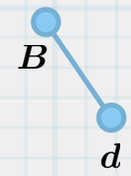
$\{B, C, D, G\} \neq \{\}$

$y = B$

$T = \{B\}$

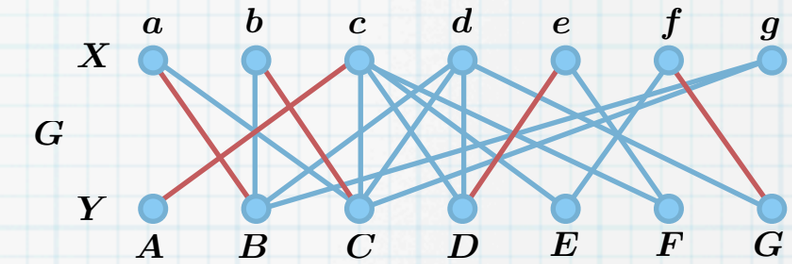
$z = a$

$S = \{a, b, c\}$

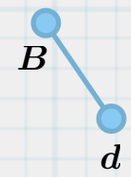


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



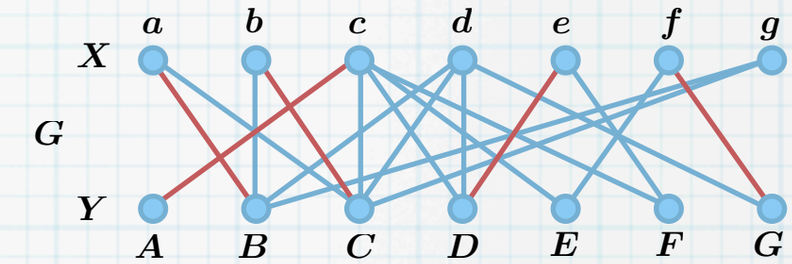
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, d\}$



(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

Der Ungarische Algorithmus

- Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
- Ausgabe:
  - (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  - (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

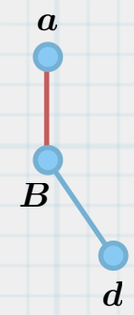
$\{B, C, D, G\} \neq \{\}$

$y = B$

$T = \{B\}$

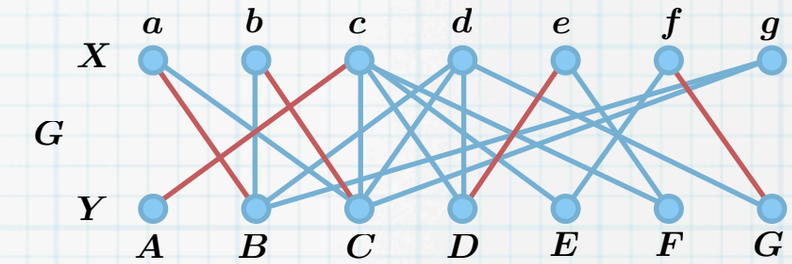
$z = a$

$S = \{a, d\}$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

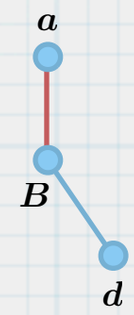
$\{B, C, D, G\} \neq \{\}$

$y = B$

$T = \{B\}$

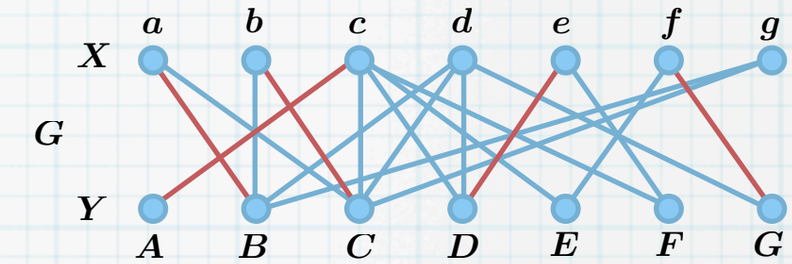
$z = a$

$S = \{a, d\}$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

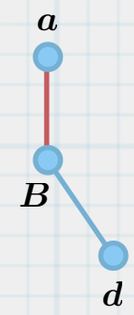
$\{B, C, D, G\} \neq \{\}$

$y = B$

$T = \{B\}$

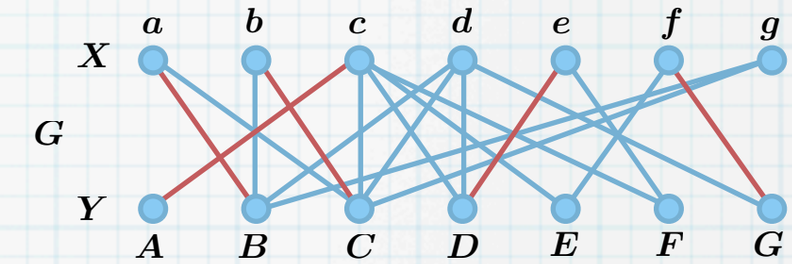
$z = a$

$S = \{a, d\}$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

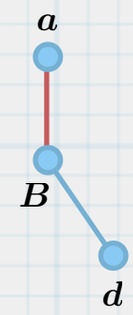
$\{B, C, D, G\} \neq \{\}$

$y = B$

$T = \{B\}$

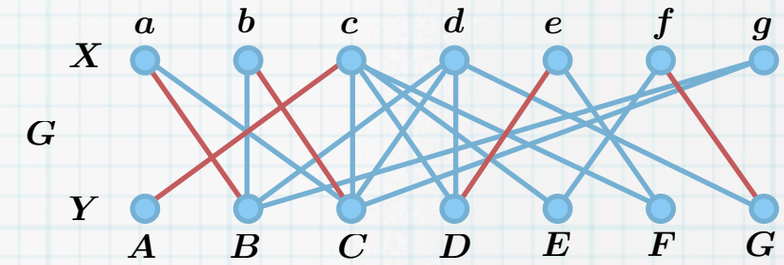
$z = a$

$S = \{a, d\}$

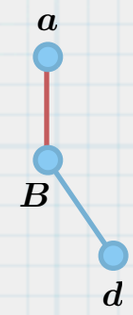


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

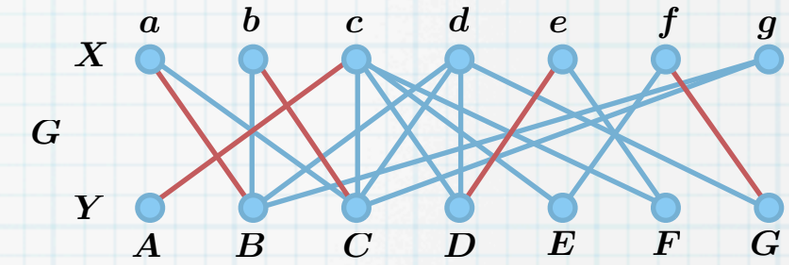


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, d\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

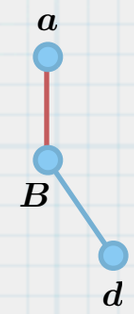
$\{B, C, D, G\} \neq \{\}$

$y = B$

$T = \{B\}$

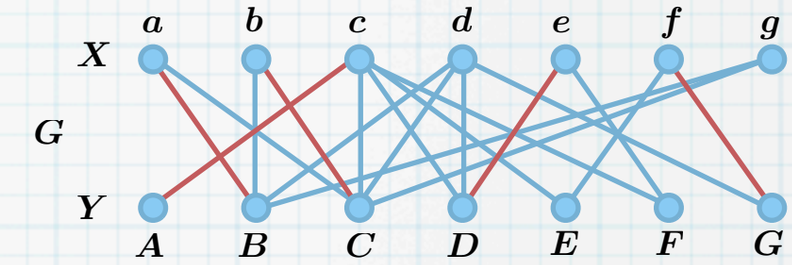
$z = a$

$S = \{a, d\}$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

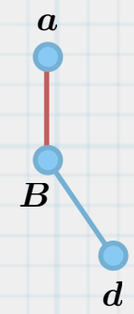
$\{B, C, D, G\} \neq \{B\}$

$y = B$

$T = \{B\}$

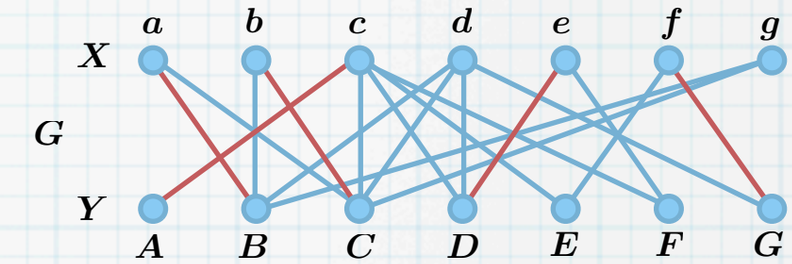
$z = a$

$S = \{a, d\}$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

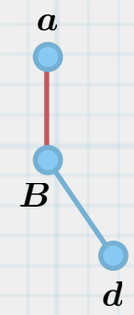
$$\{B, C, D, G\} \neq \{B\}$$

$$y = B$$

$$T = \{B\}$$

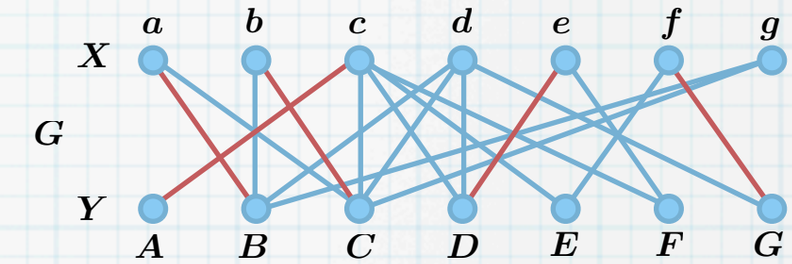
$$z = a$$

$$S = \{a, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

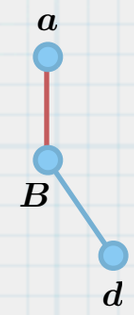
$\{B, C, D, G\} \neq \{B\}$

$y = B$

$T = \{B\}$

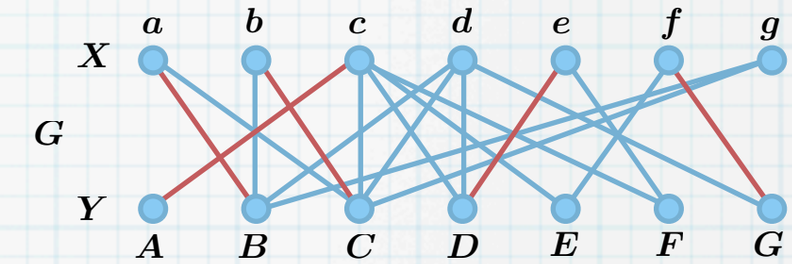
$z = a$

$S = \{a, d\}$

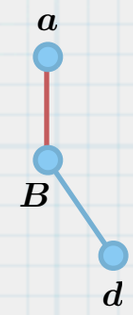


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



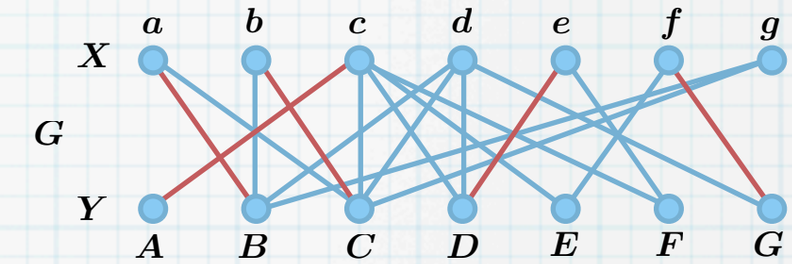
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{B\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, d\}$



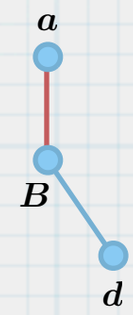
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

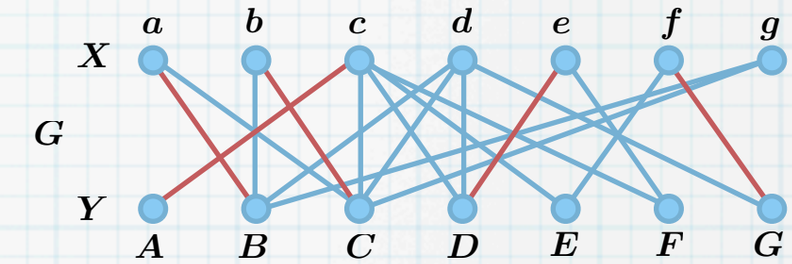


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = d$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{d\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, G\} \neq \{B\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, d\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = d$

$S = \{d\}, T = \{\}$

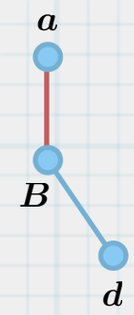
$\{B, C, D, G\} \neq \{B\}$

$y = C$

$T = \{B, C\}$

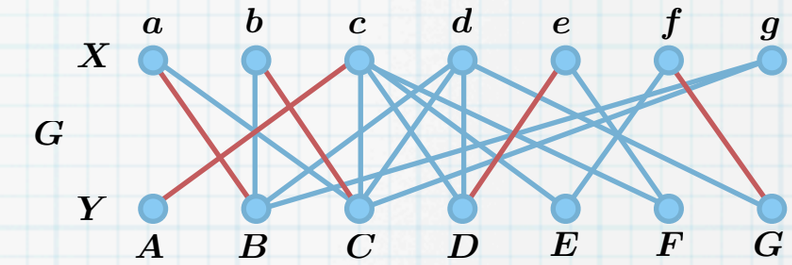
$z = a$

$S = \{a, d\}$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

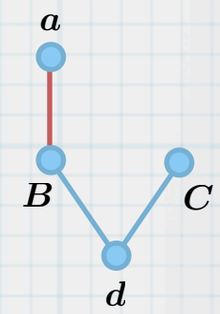
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

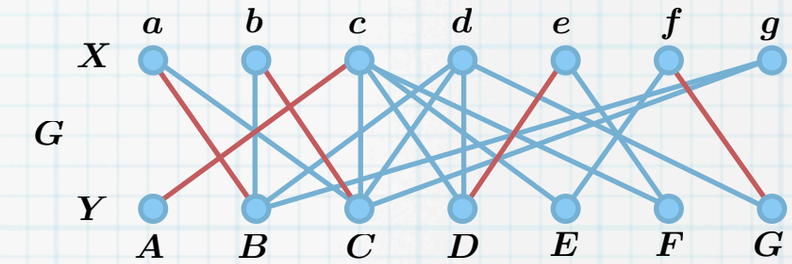
$$z = a$$

$$S = \{a, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

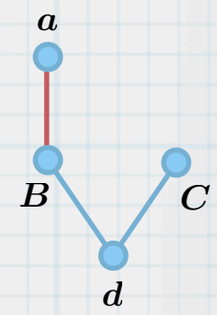
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

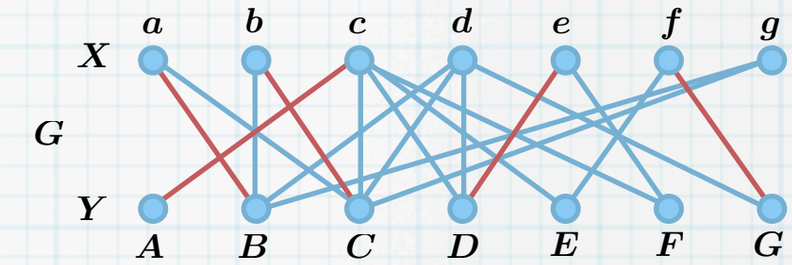
$$z = a$$

$$S = \{a, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

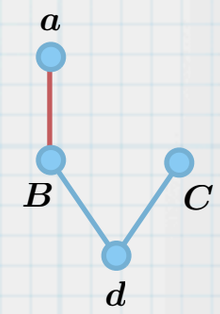
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

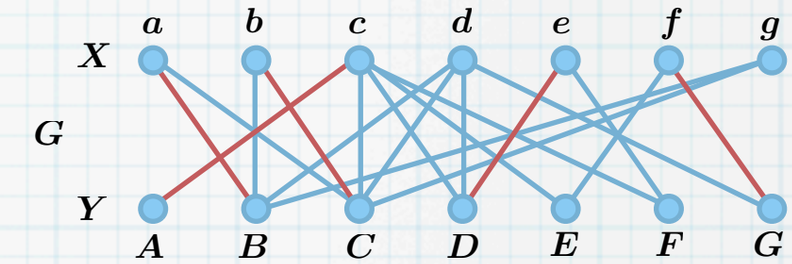
$$z = b$$

$$S = \{a, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

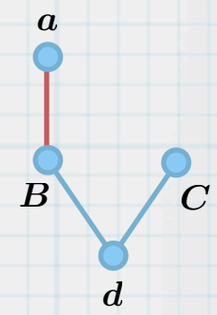
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

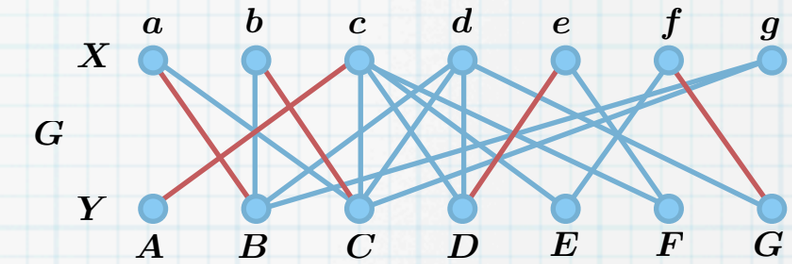
$$z = b$$

$$S = \{a, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

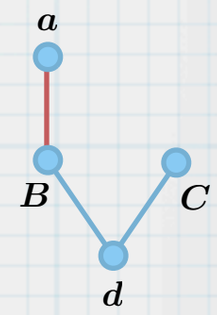
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

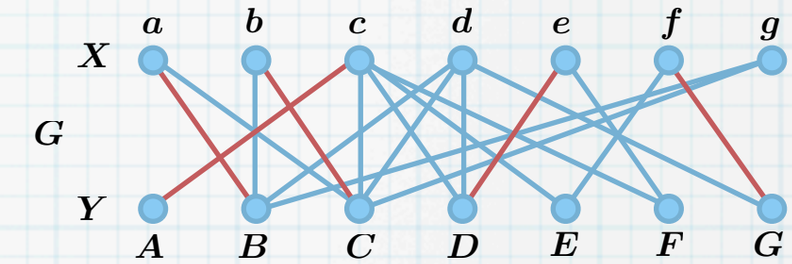
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

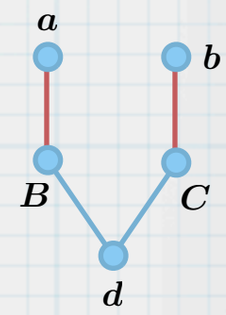
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

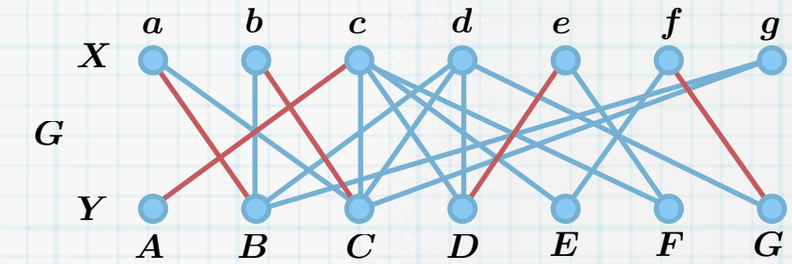
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

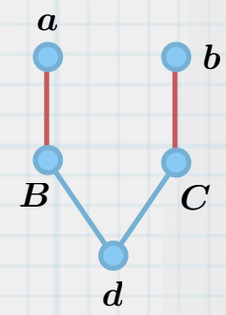
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

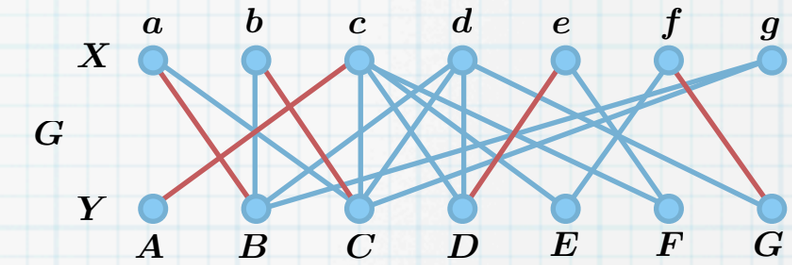
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

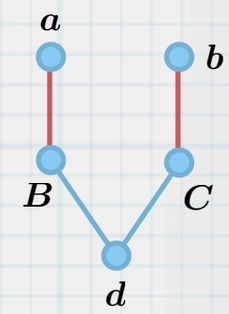
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

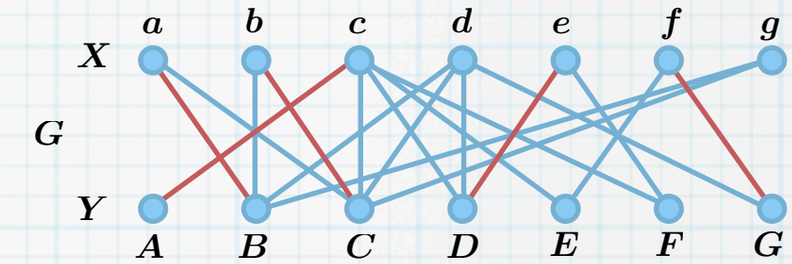
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

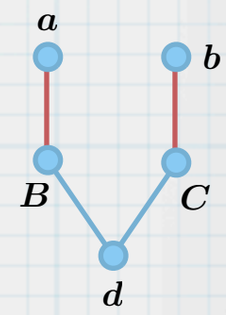
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

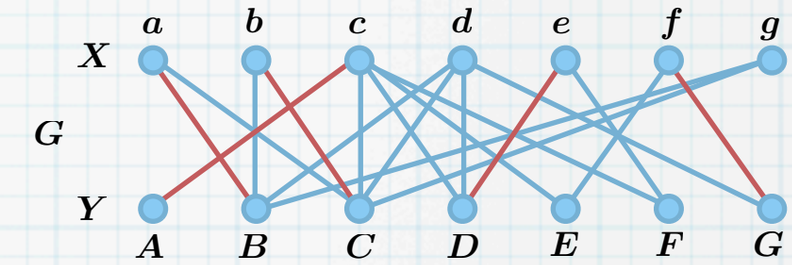
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

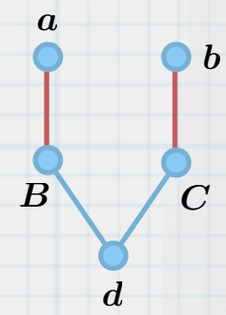
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

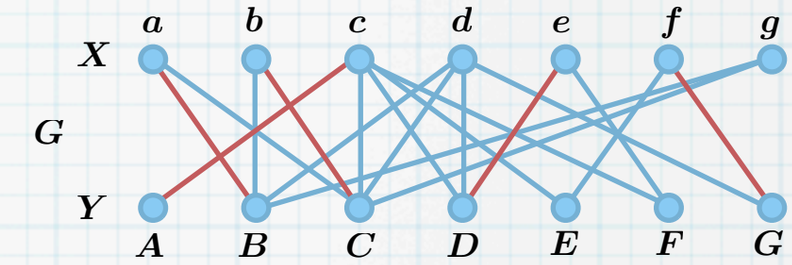
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

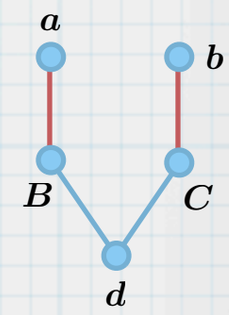
$$\{B, C, D, G\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

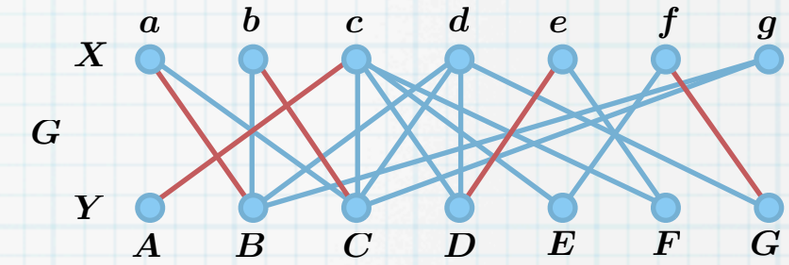
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

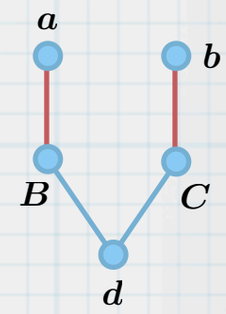
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = C$$

$$T = \{B, C\}$$

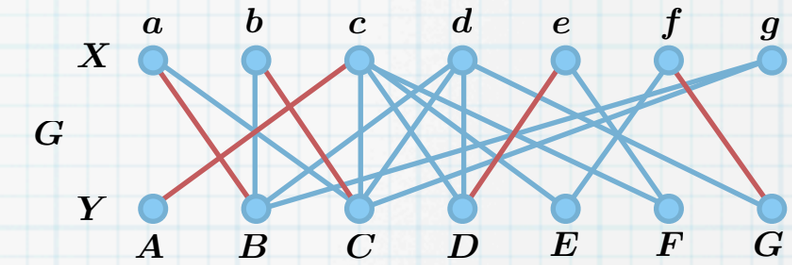
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

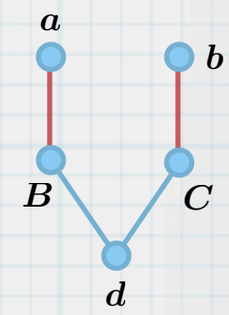
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = C$$

$$T = \{B, C\}$$

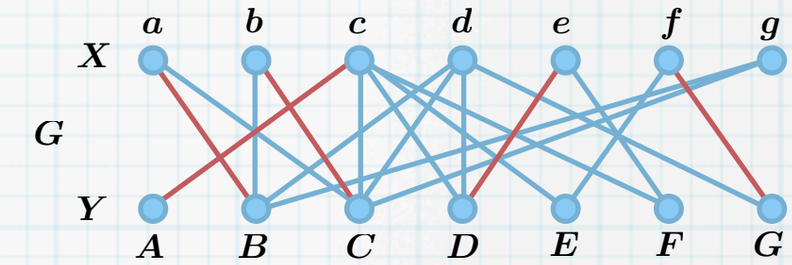
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

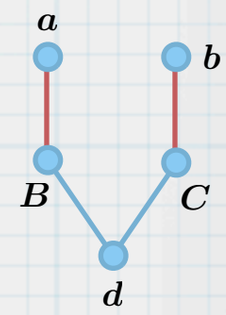
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = C$$

$$T = \{B, C\}$$

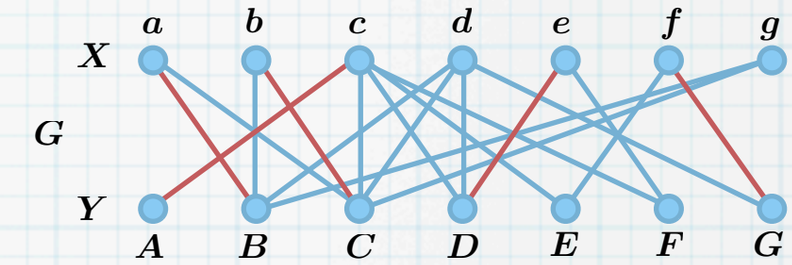
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

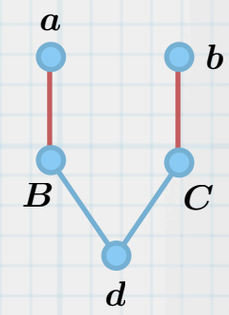
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C\}$$

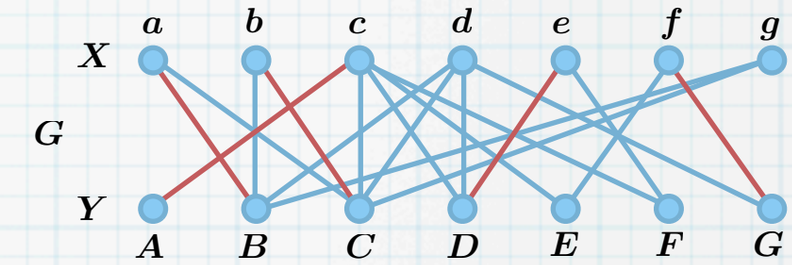
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

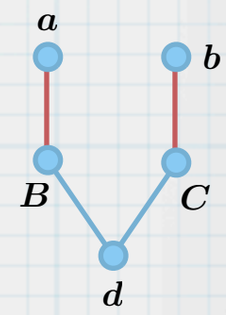
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C\}$$

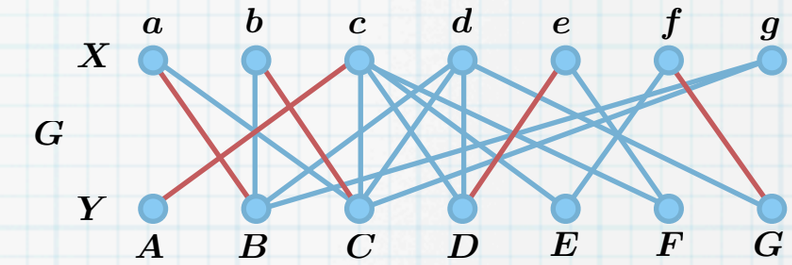
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

- Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
- Ausgabe:
  - (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  - (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

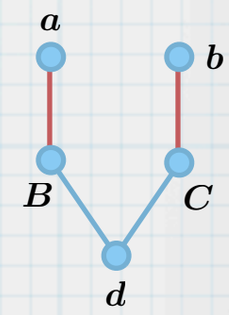
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

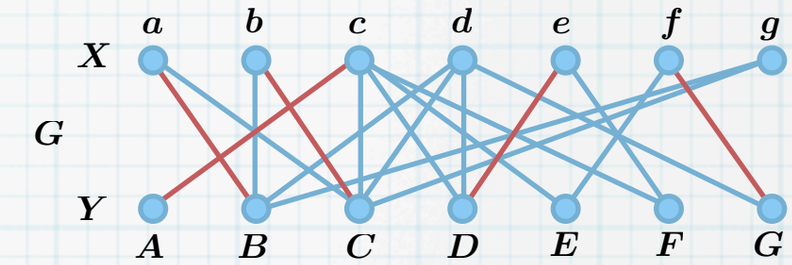
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

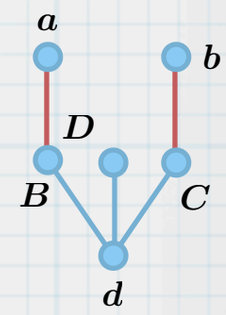
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

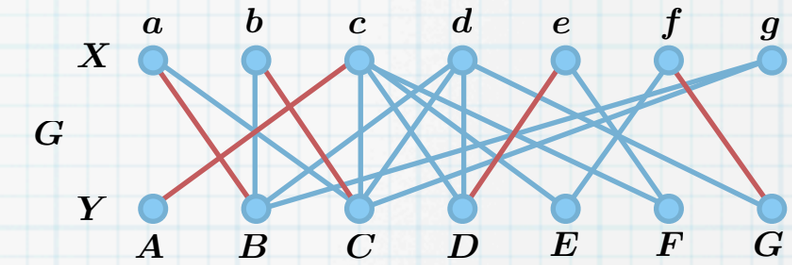
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

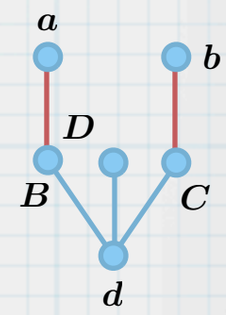
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

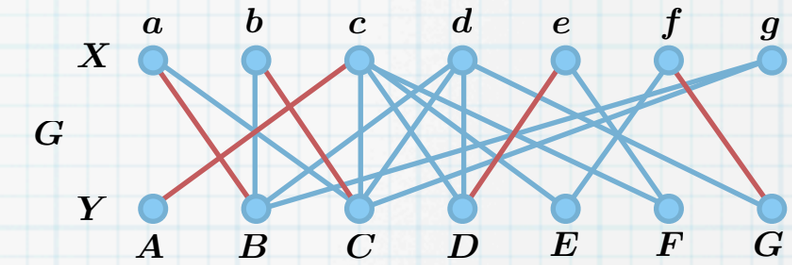
$$z = b$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

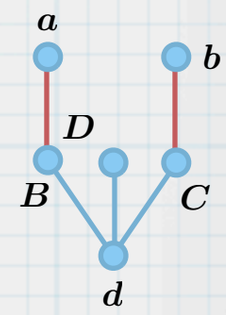
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

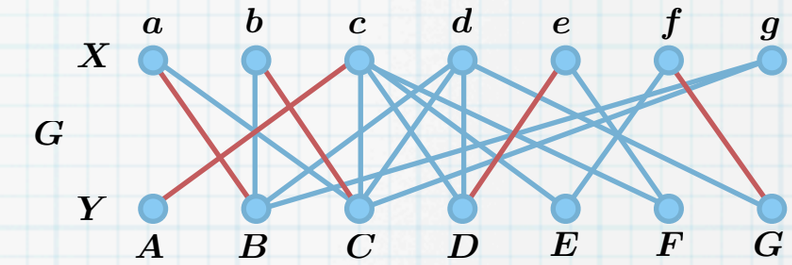
$$z = e$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

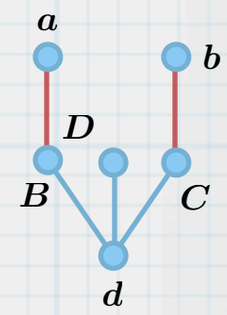
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

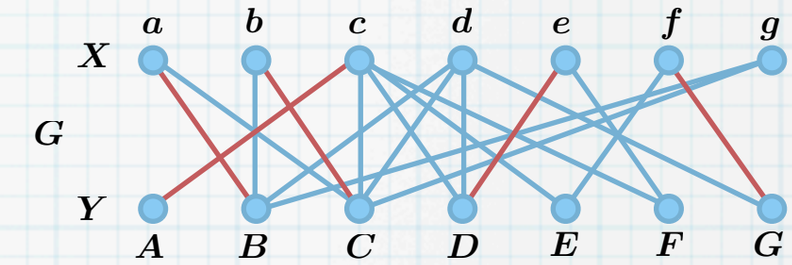
$$z = e$$

$$S = \{a, b, d\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

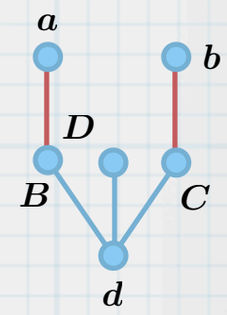
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

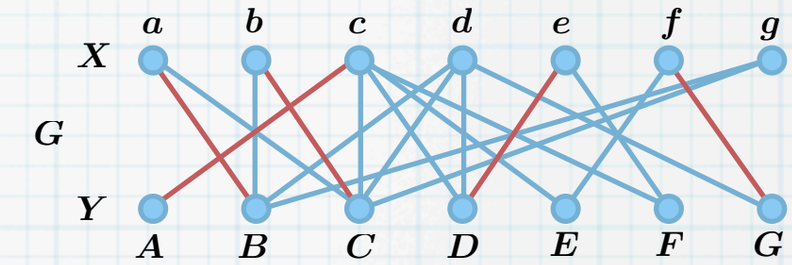
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

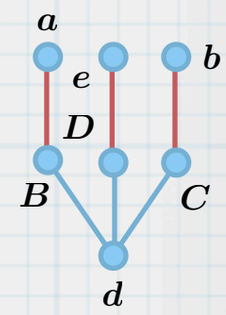
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

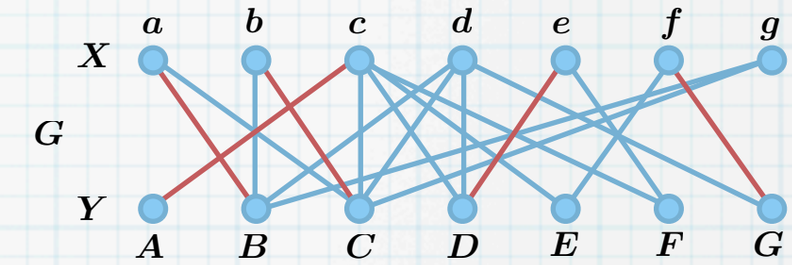
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

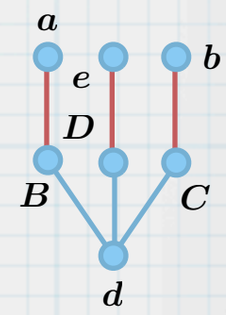
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

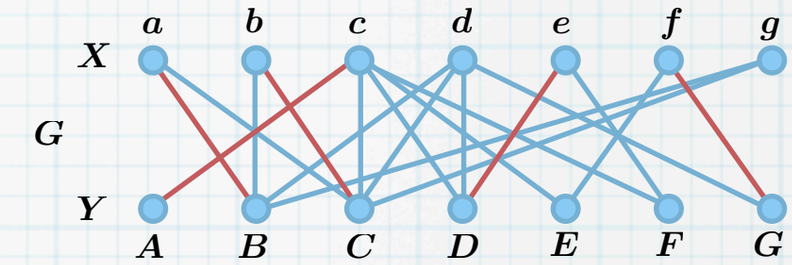
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

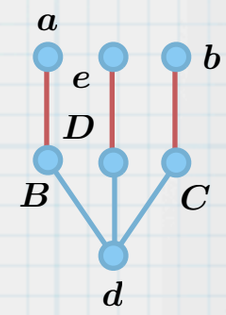
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

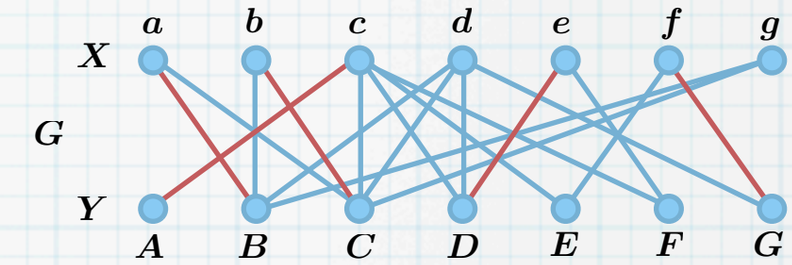
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

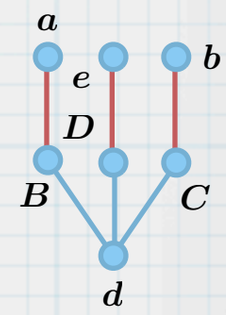
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

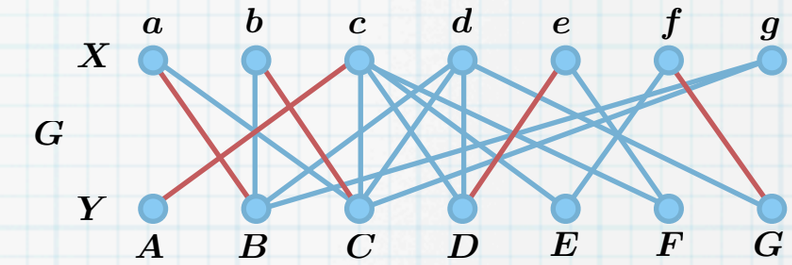
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) <u>entweder</u> ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) <u>oder</u> eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

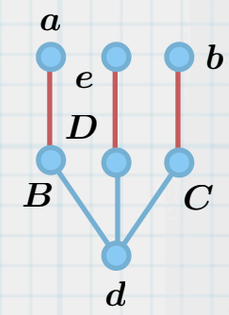
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

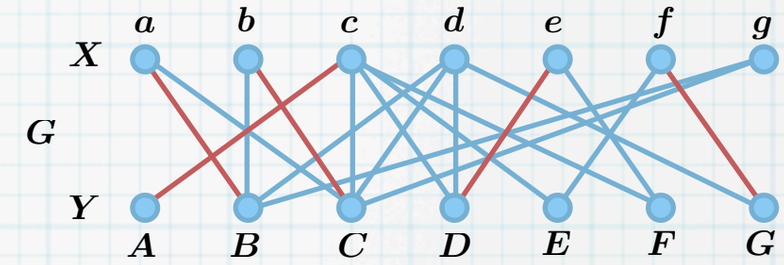
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

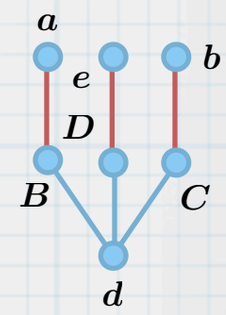
$$\{B, C, D, G\} \neq \{B, C\}$$

$$y = D$$

$$T = \{B, C, D\}$$

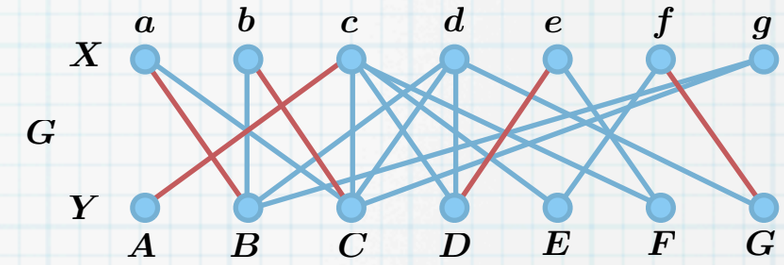
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

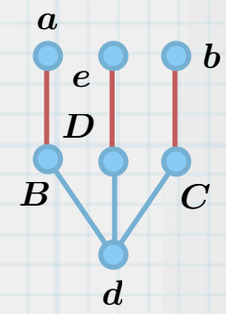
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = D$$

$$T = \{B, C, D\}$$

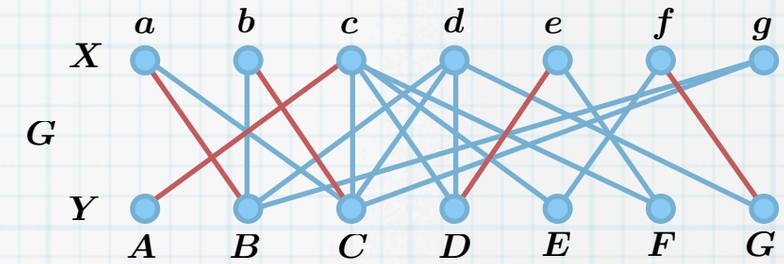
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

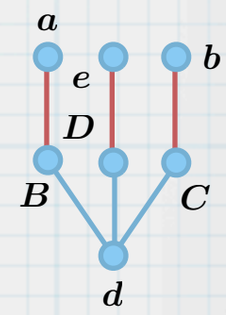
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = D$$

$$T = \{B, C, D\}$$

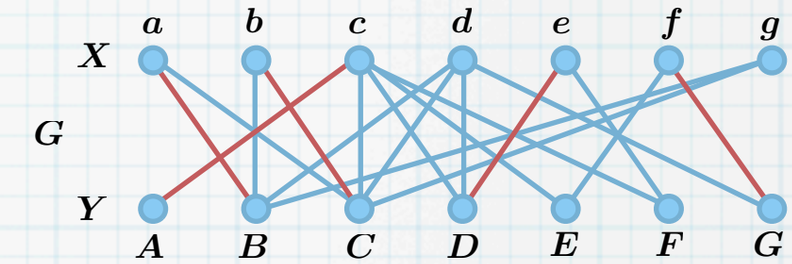
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

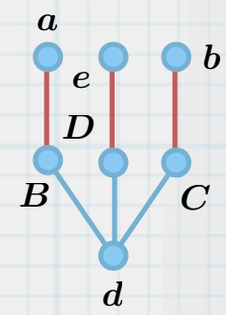
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = D$$

$$T = \{B, C, D\}$$

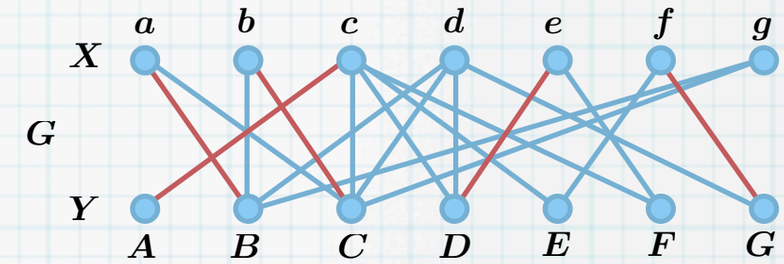
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

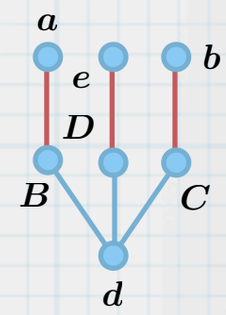
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D\}$$

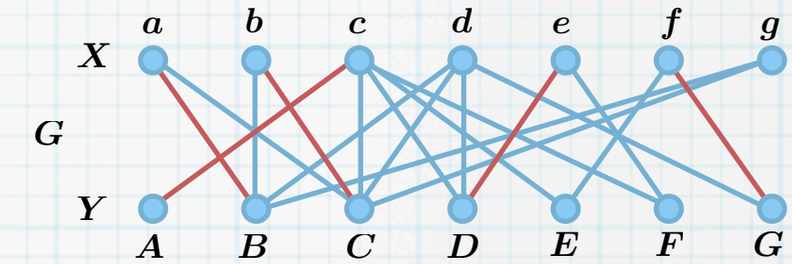
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

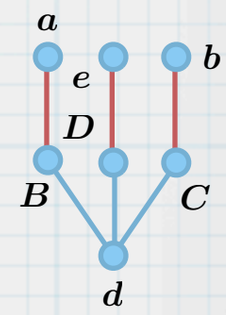
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D\}$$

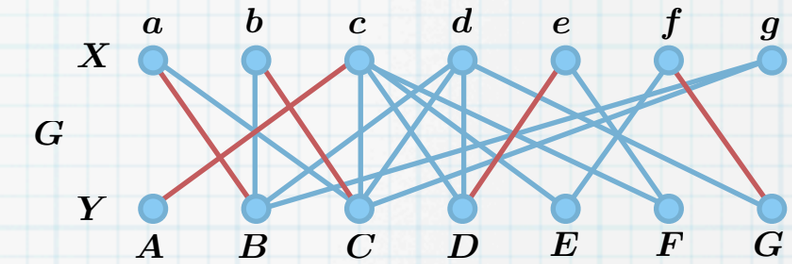
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

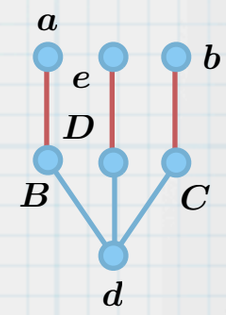
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

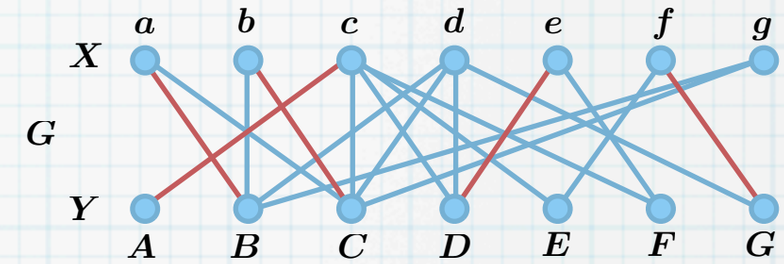
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

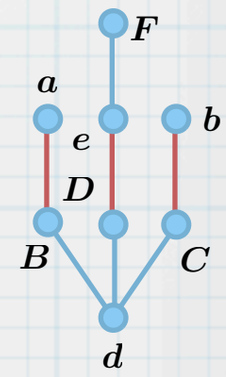
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

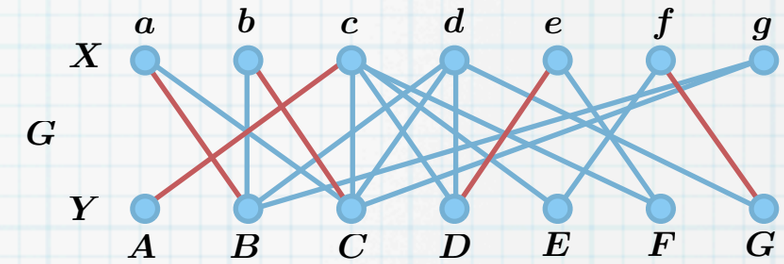
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

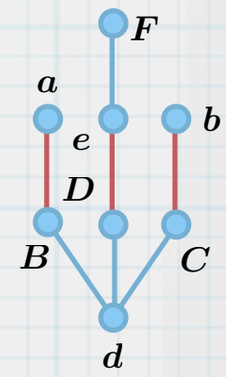
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

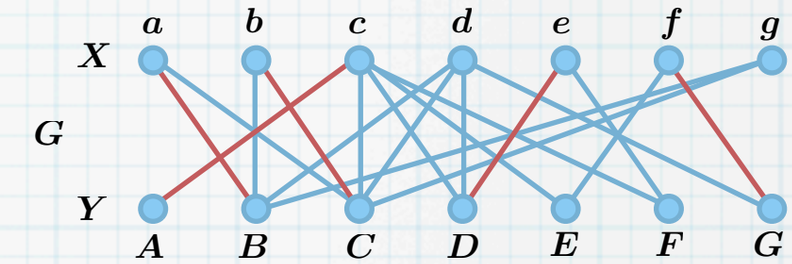
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

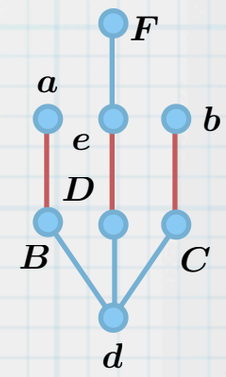
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

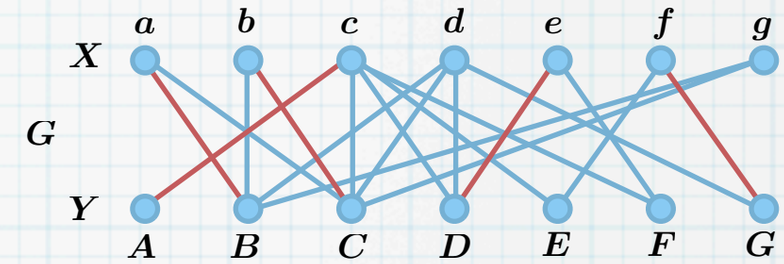
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

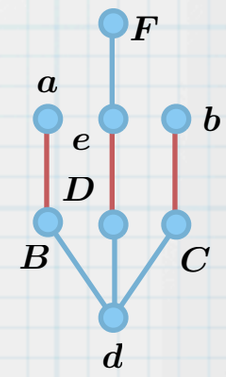
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

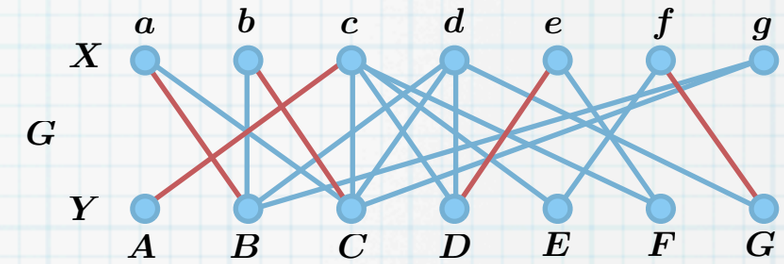
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

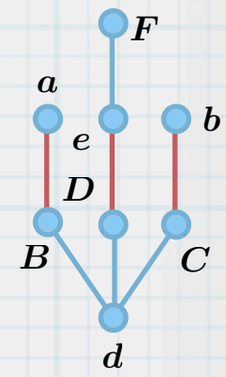
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

$$z = e$$

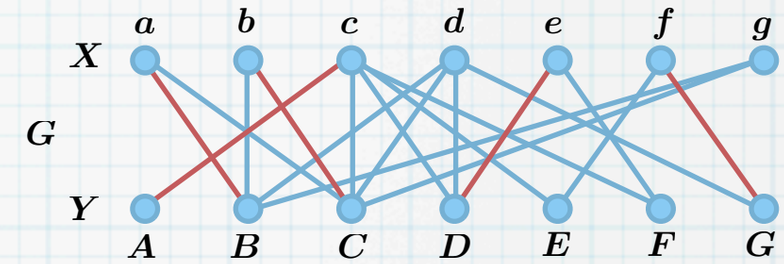
$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

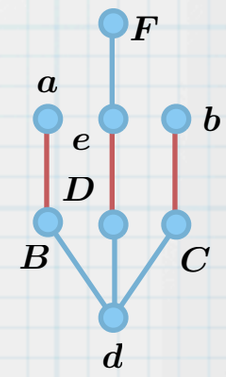
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

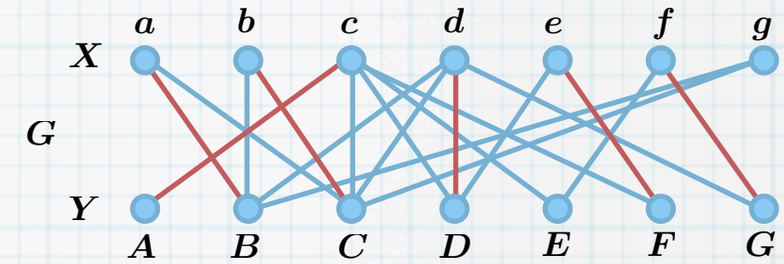
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

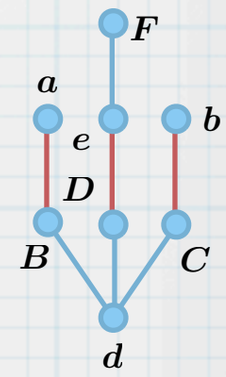
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

$$z = e$$

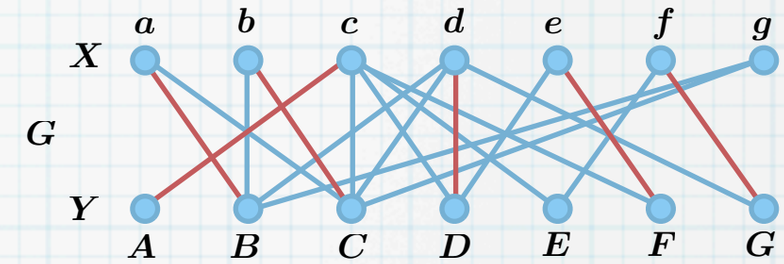
$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

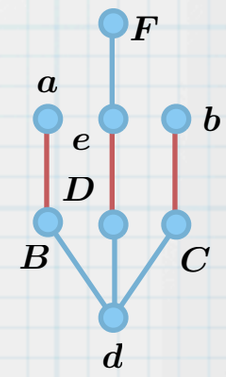
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

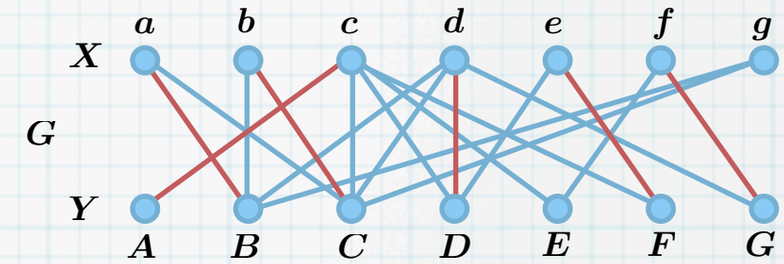
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = d$$

$$S = \{d\}, T = \{\}$$

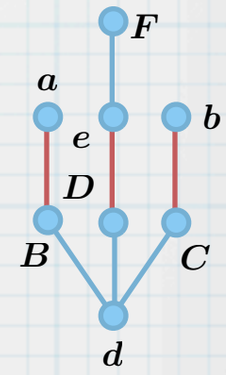
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

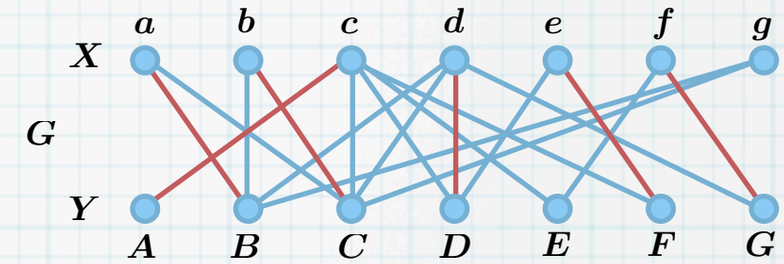
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = g$$

$$S = \{d\}, T = \{\}$$

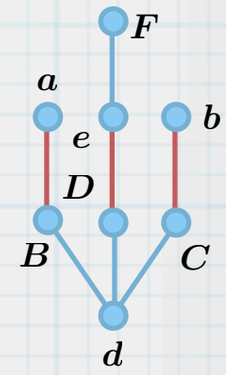
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

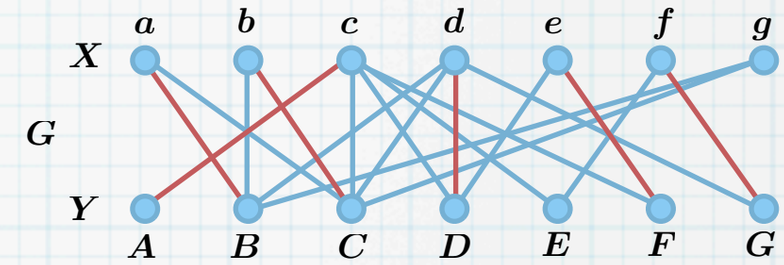
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = g$$

$$S = \{d\}, T = \{\}$$

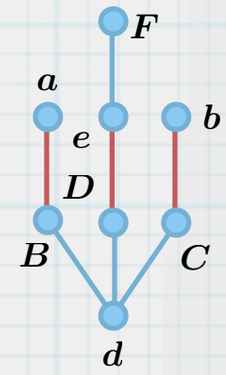
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

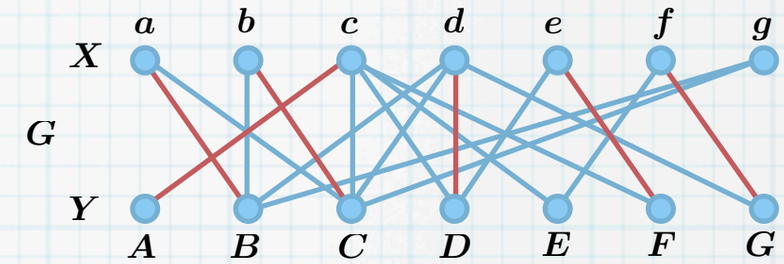
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = g$$

$$S = \{g\}, T = \{\}$$

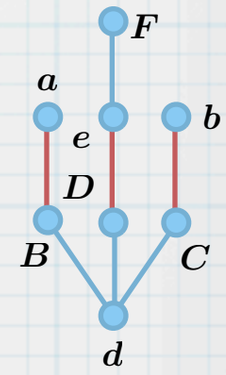
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$

$$T = \{B, C, D, F\}$$

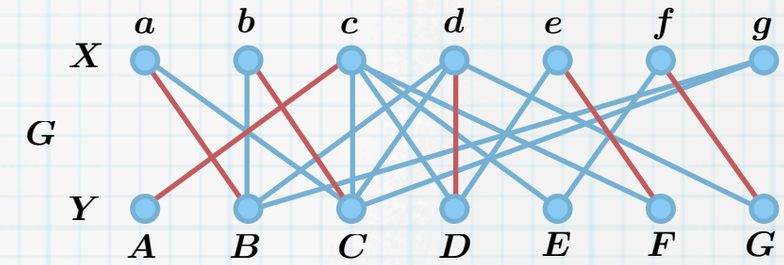
$$z = e$$

$$S = \{a, b, d, e\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

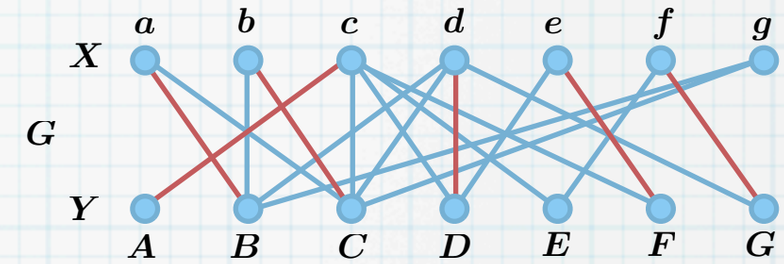
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e\}$$

g

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

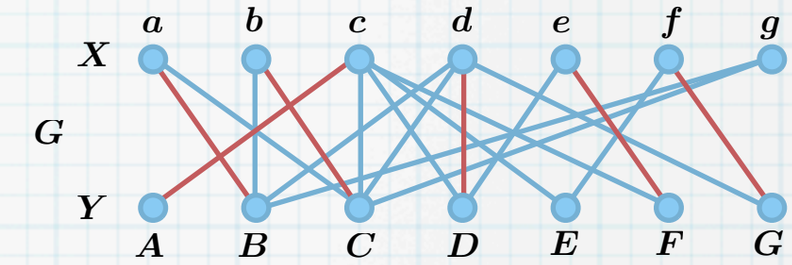


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, F, G\} \neq \{B, C, D\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = F$
(11) $T := T \cup \{y\}$ — $T = \{B, C, D, F\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = e$
(13) $S := S \cup \{z\}$ — $S = \{a, b, d, e\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

g

Der Ungarische Algorithmus

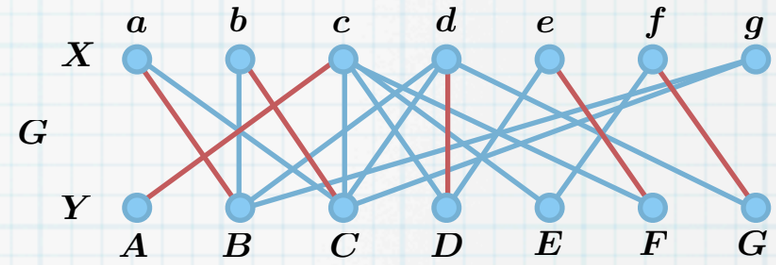
* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C, D, F, G\} \neq \{B, C, D\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = F$
(11) $T := T \cup \{y\}$ — $T = \{B, C, D, F\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = e$
(13) $S := S \cup \{z\}$ — $S = \{a, b, d, e\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

Der Ungarische Algorithmus

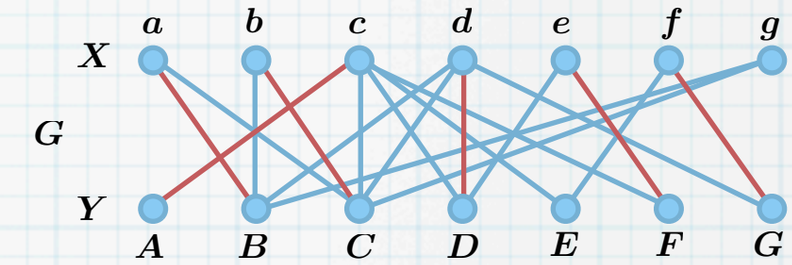
* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = F$
(11) $T := T \cup \{y\}$ — $T = \{B, C, D, F\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = e$
(13) $S := S \cup \{z\}$ — $S = \{a, b, d, e\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

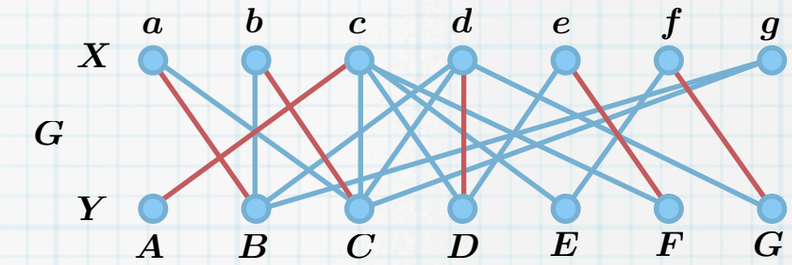


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = F$
(11) $T := T \cup \{y\}$ — $T = \{B, C, D, F\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = e$
(13) $S := S \cup \{z\}$ — $S = \{a, b, d, e\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

g

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

$\{B, C\} \neq \{\}$

$y = F$

$T = \{B, C, D, F\}$

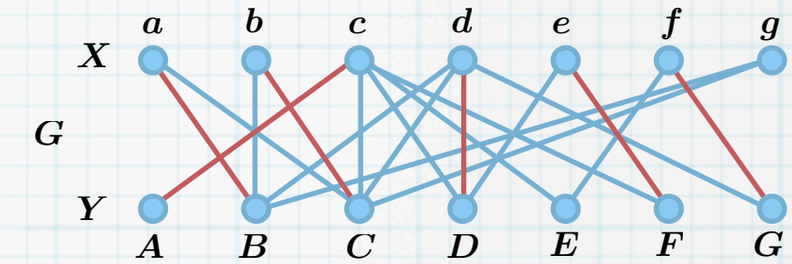
$z = e$

$S = \{a, b, d, e\}$

g

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

$\{B, C\} \neq \{\}$

$y = B$

$T = \{B, C, D, F\}$

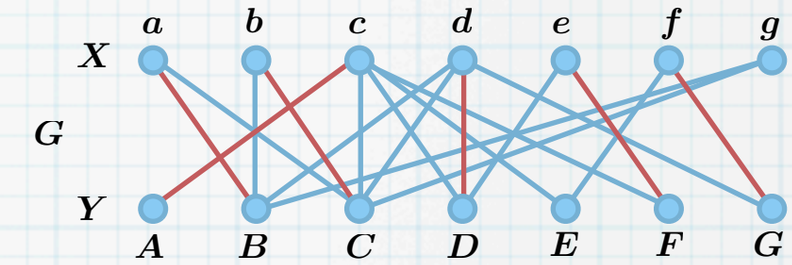
$z = e$

$S = \{a, b, d, e\}$

g

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

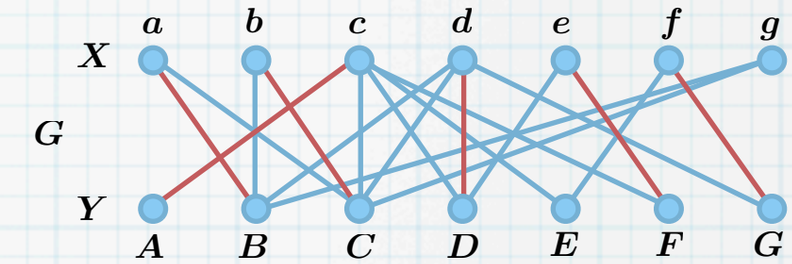


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B, C, D, F\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = e$
(13) $S := S \cup \{z\}$ — $S = \{a, b, d, e\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

g

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

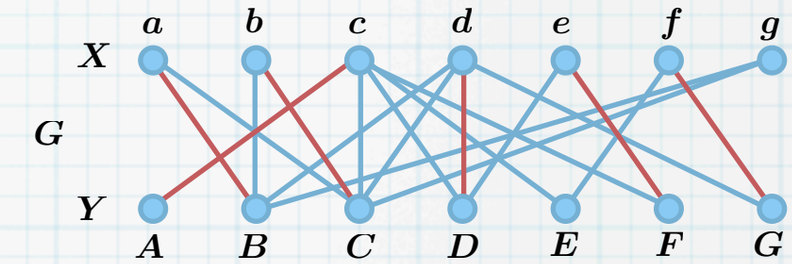


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = e$
(13) $S := S \cup \{z\}$ — $S = \{a, b, d, e\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

g

Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

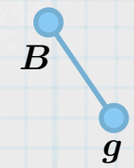
$\{B, C\} \neq \{\}$

$y = B$

$T = \{B\}$

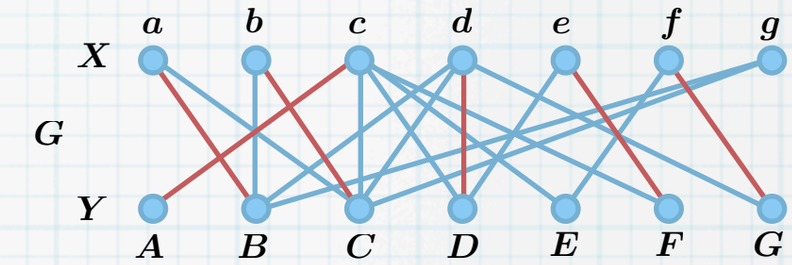
$z = e$

$S = \{a, b, d, e\}$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

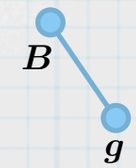
$\{B, C\} \neq \{\}$

$y = B$

$T = \{B\}$

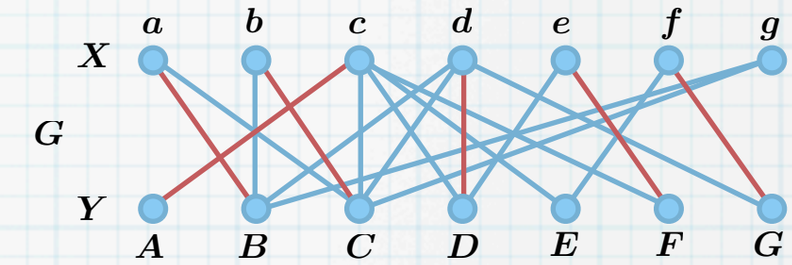
$z = e$

$S = \{a, b, d, e\}$

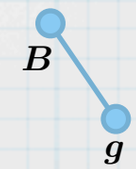


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

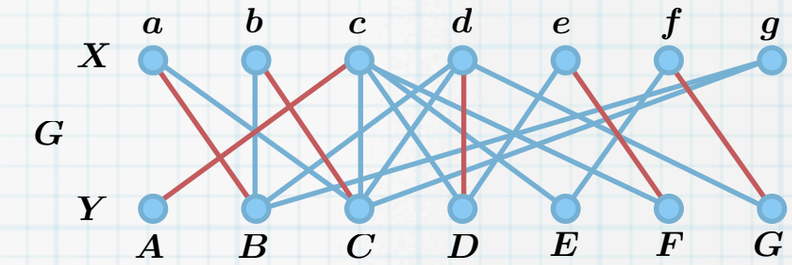


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, b, d, e\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

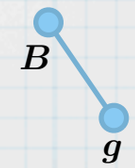
$\{B, C\} \neq \{\}$

$y = B$

$T = \{B\}$

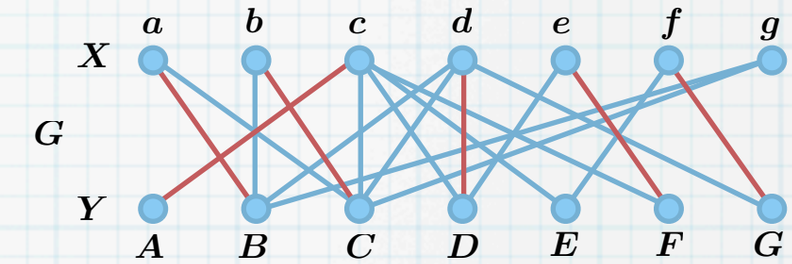
$z = a$

$S = \{a, b, d, e\}$

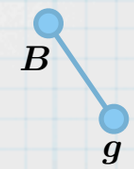


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

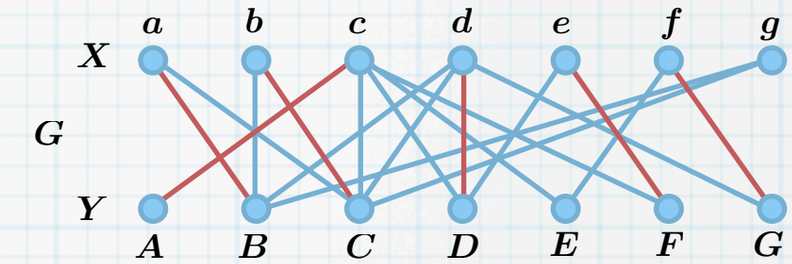


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

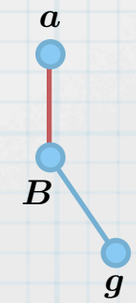
$\{B, C\} \neq \{\}$

$y = B$

$T = \{B\}$

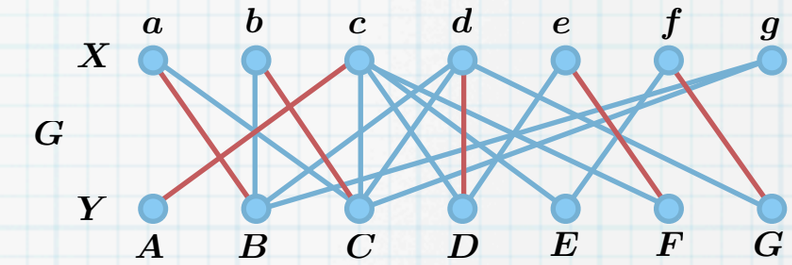
$z = a$

$S = \{a, g\}$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

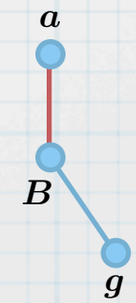
$\{B, C\} \neq \{\}$

$y = B$

$T = \{B\}$

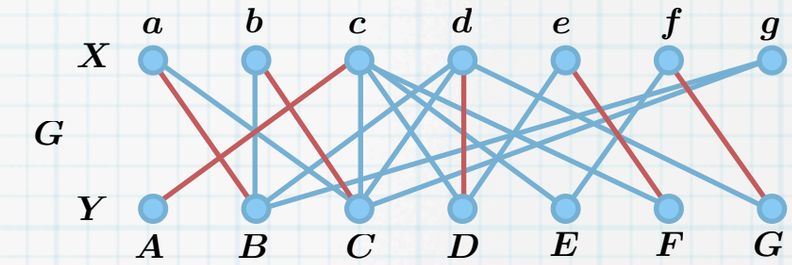
$z = a$

$S = \{a, g\}$

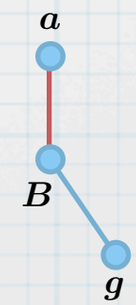


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

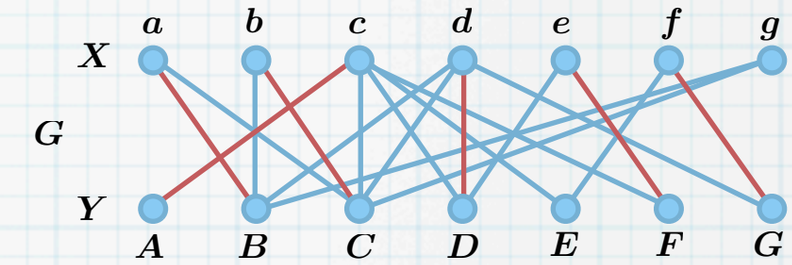


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



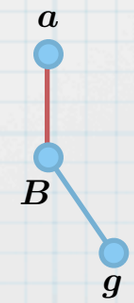
Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



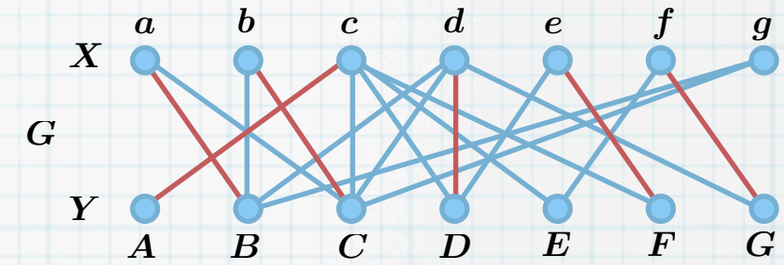
(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$
$$\{B, C\} \neq \{\}$$
$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$

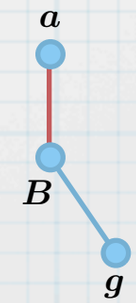


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

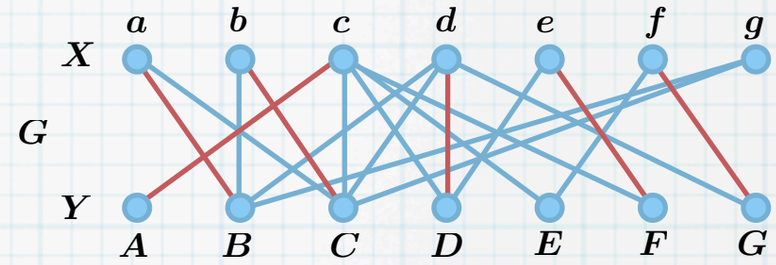


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

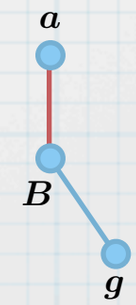


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

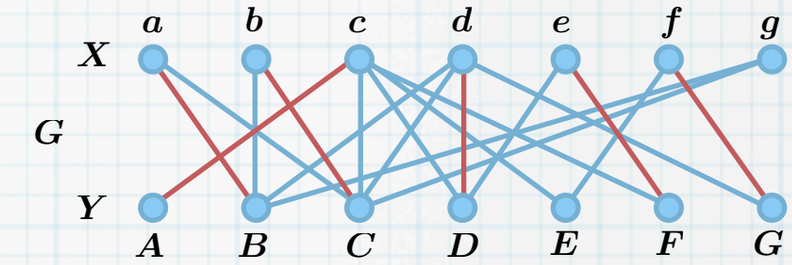


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

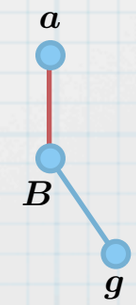


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

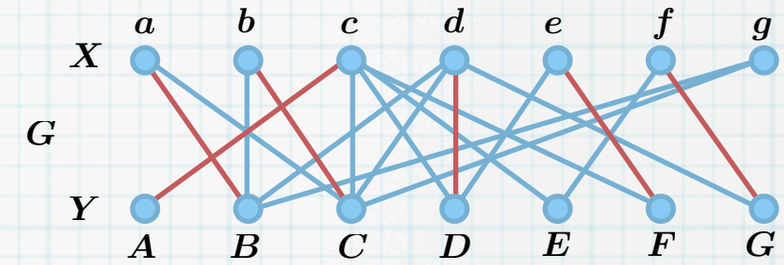


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{B\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = B$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = g$$

$$S = \{g\}, T = \{\}$$

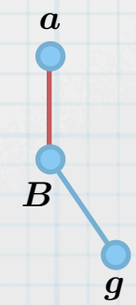
$$\{B, C\} \neq \{B\}$$

$$y = B$$

$$T = \{B\}$$

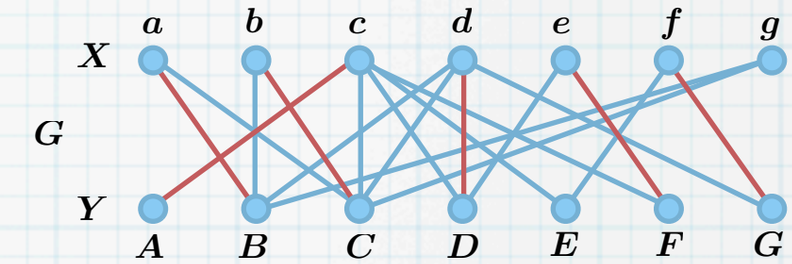
$$z = a$$

$$S = \{a, g\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

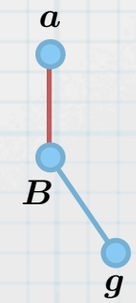
$\{B, C\} \neq \{B\}$

$y = B$

$T = \{B\}$

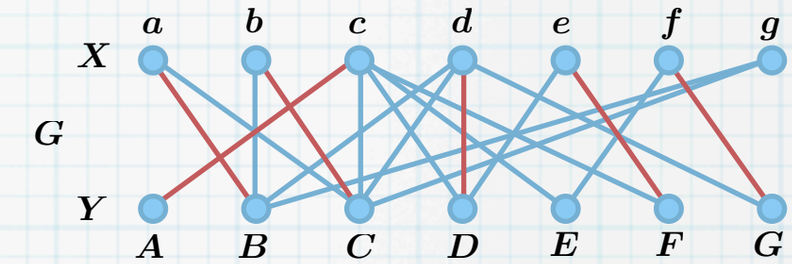
$z = a$

$S = \{a, g\}$

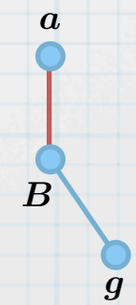


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

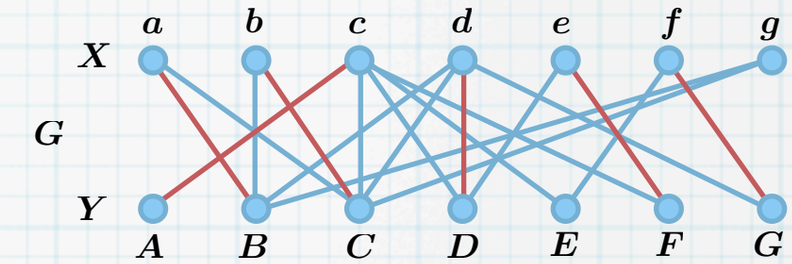


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{B\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

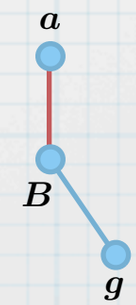
$\{B, C\} \neq \{B\}$

$y = C$

$T = \{B\}$

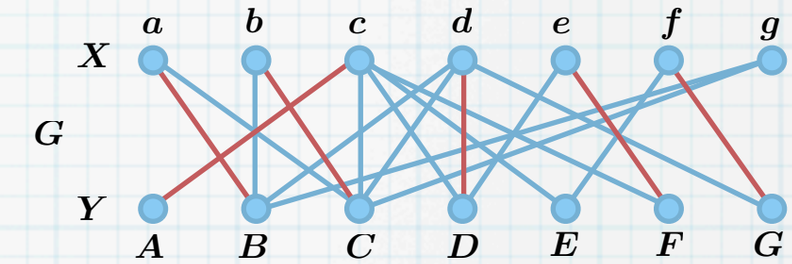
$z = a$

$S = \{a, g\}$

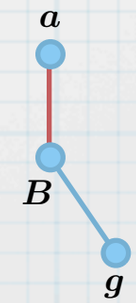


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

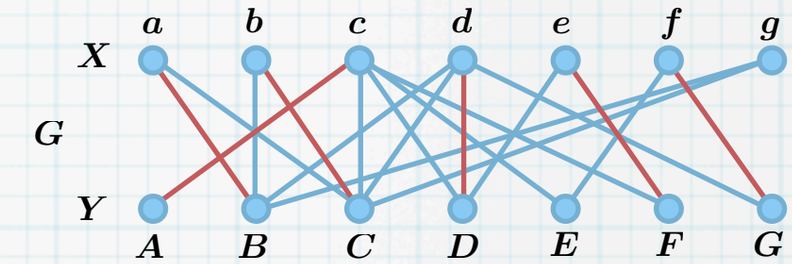


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{B\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

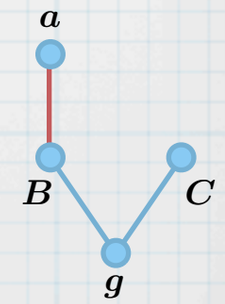
$\{B, C\} \neq \{B\}$

$y = C$

$T = \{B, C\}$

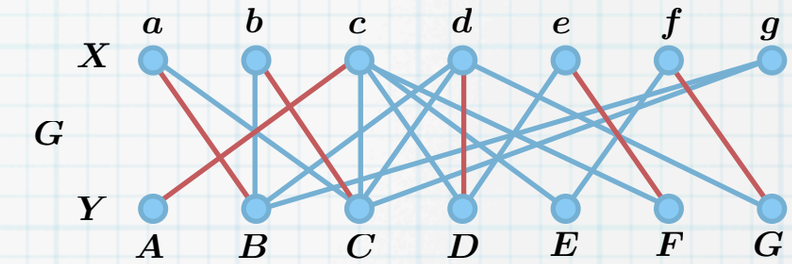
$z = a$

$S = \{a, g\}$

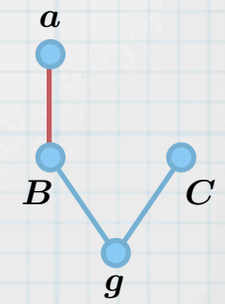


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

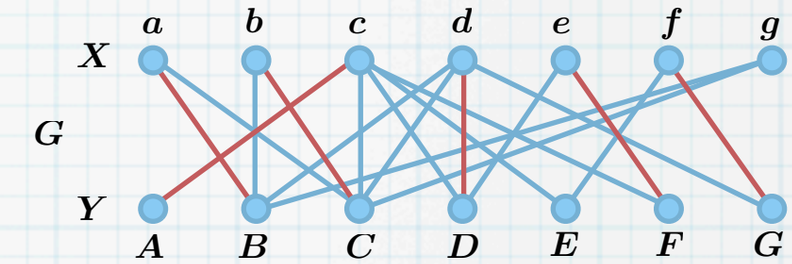


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{B\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = a$
(13) $S := S \cup \{z\}$ — $S = \{a, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

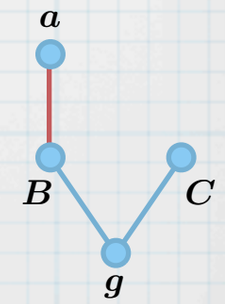
$\{B, C\} \neq \{B\}$

$y = C$

$T = \{B, C\}$

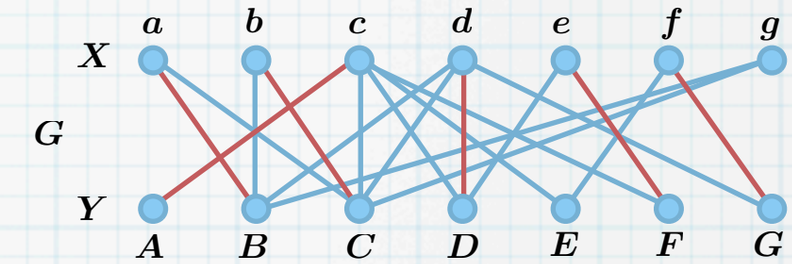
$z = b$

$S = \{a, g\}$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

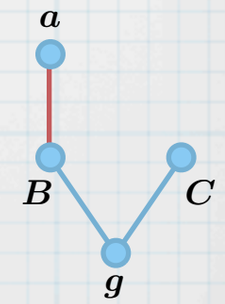
$\{B, C\} \neq \{B\}$

$y = C$

$T = \{B, C\}$

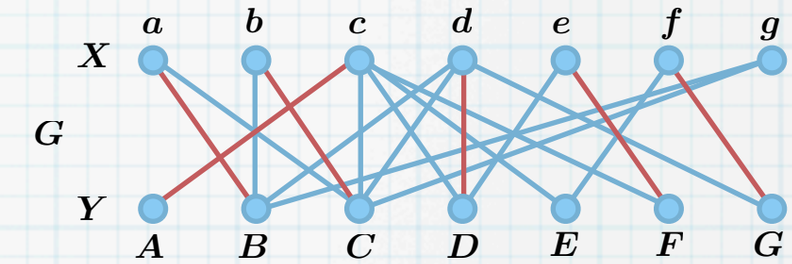
$z = b$

$S = \{a, g\}$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

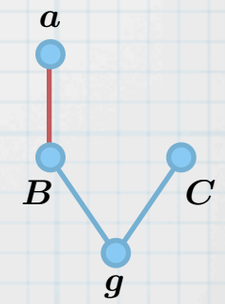
$\{B, C\} \neq \{B\}$

$y = C$

$T = \{B, C\}$

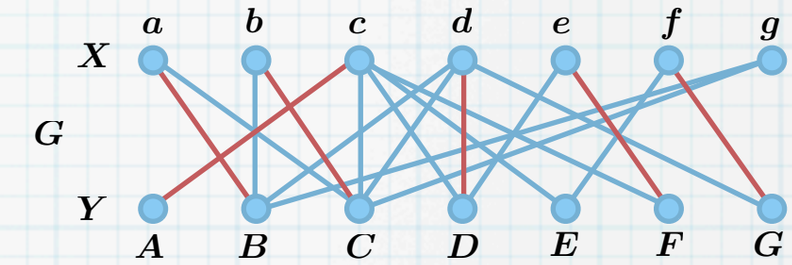
$z = b$

$S = \{a, b, g\}$

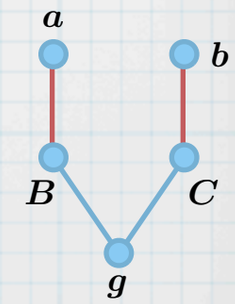


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

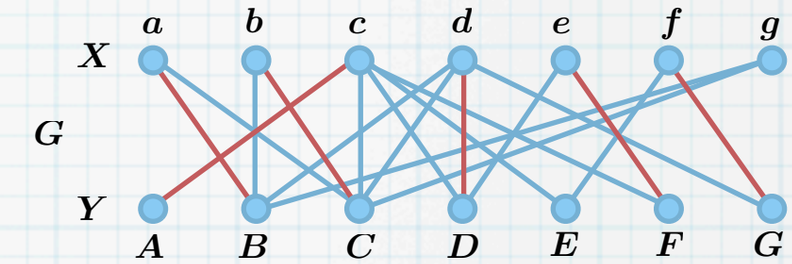


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{B\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = b$
(13) $S := S \cup \{z\}$ — $S = \{a, b, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = g$$

$$S = \{g\}, T = \{\}$$

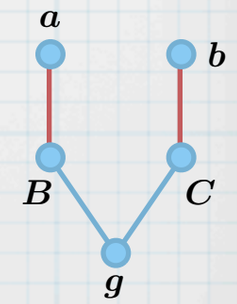
$$\{B, C\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

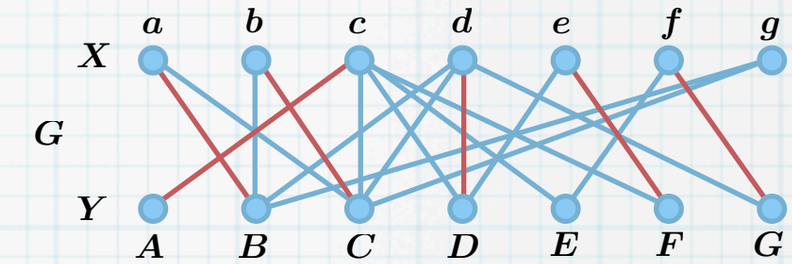
$$z = b$$

$$S = \{a, b, g\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = g$$

$$S = \{g\}, T = \{\}$$

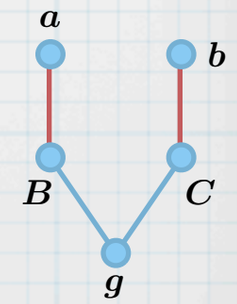
$$\{B, C\} \neq \{B\}$$

$$y = C$$

$$T = \{B, C\}$$

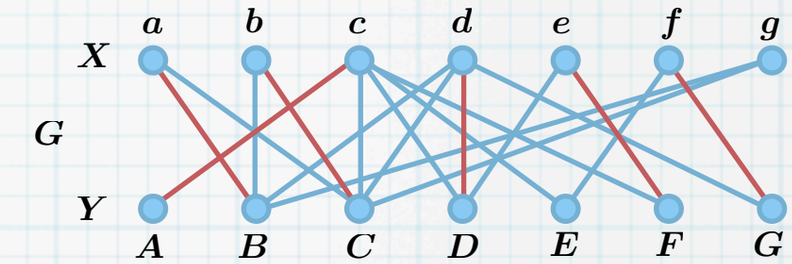
$$z = b$$

$$S = \{a, b, g\}$$

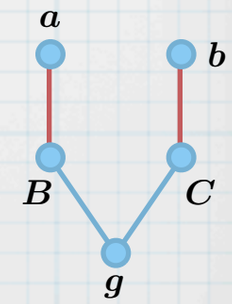


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

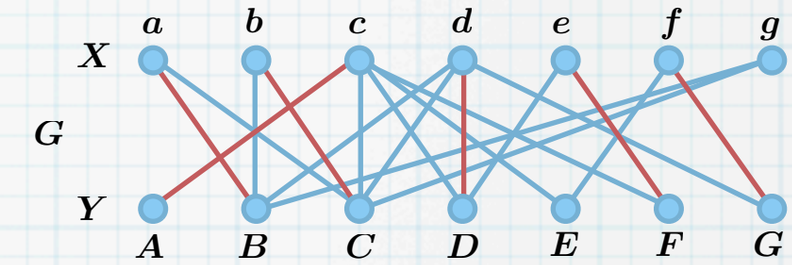


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{B\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = b$
(13) $S := S \cup \{z\}$ — $S = \{a, b, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

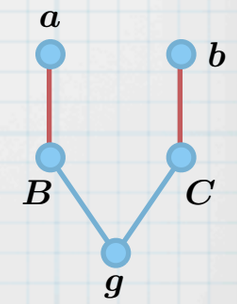


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

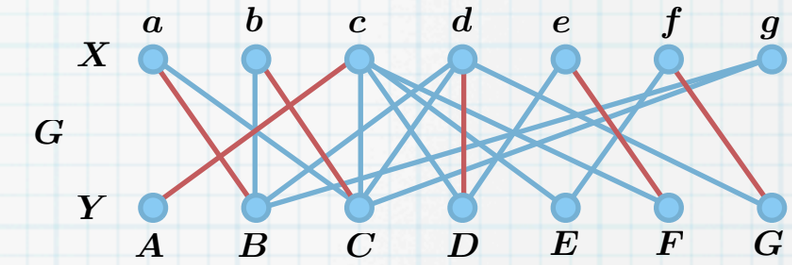


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{B\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = b$
(13) $S := S \cup \{z\}$ — $S = \{a, b, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

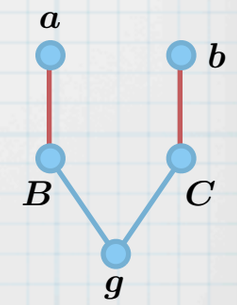


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

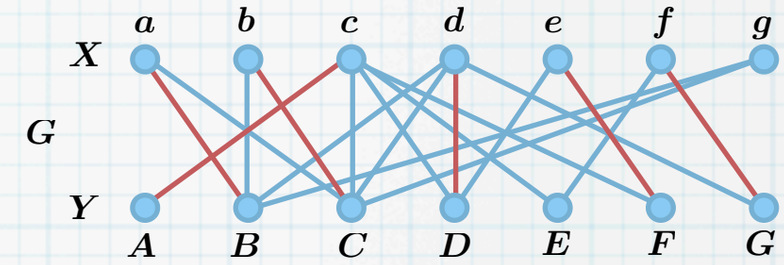


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} \neq \{B\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = b$
(13) $S := S \cup \{z\}$ — $S = \{a, b, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

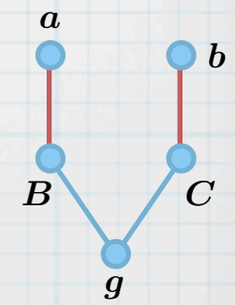


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

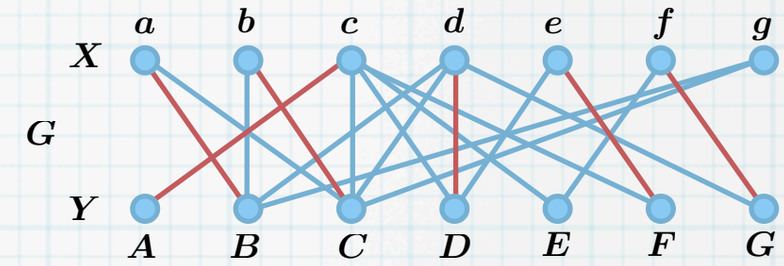


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} = \{B, C\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = b$
(13) $S := S \cup \{z\}$ — $S = \{a, b, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

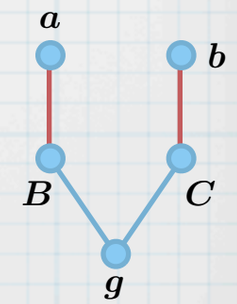


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.

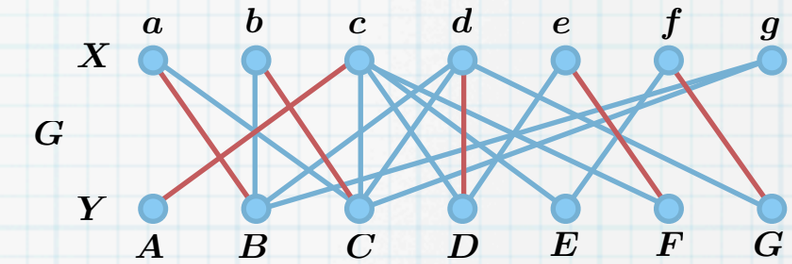


(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} = \{B, C\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = b$
(13) $S := S \cup \{z\}$ — $S = \{a, b, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$M := \{aB, cC, eD, fG\}$$

$$x_0 = g$$

$$S = \{g\}, T = \{\}$$

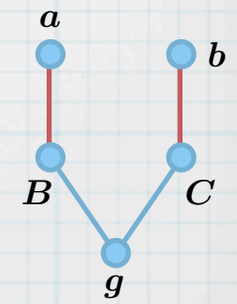
$$\{B, C\} = \{B, C\}$$

$$y = C$$

$$T = \{B, C\}$$

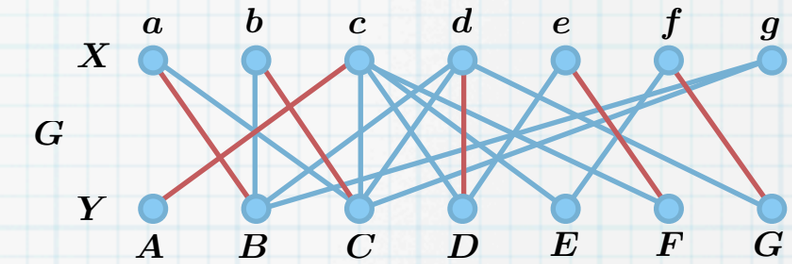
$$z = b$$

$$S = \{a, b, g\}$$



Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do**
(4) $S := \{x_0\}, T := \emptyset$
(5) **repeat**
(6) **if** $N(S) = T$ **then**
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$
(11) $T := T \cup \{y\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(13) $S := S \cup \{z\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**
(19) **end algorithm**

$M := \{aB, cC, eD, fG\}$

$x_0 = g$

$S = \{g\}, T = \{\}$

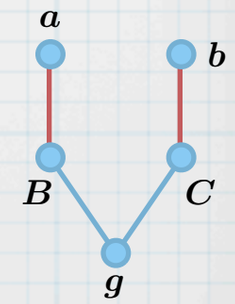
$\{B, C\} = \{B, C\}$

$y = C$

$T = \{B, C\}$

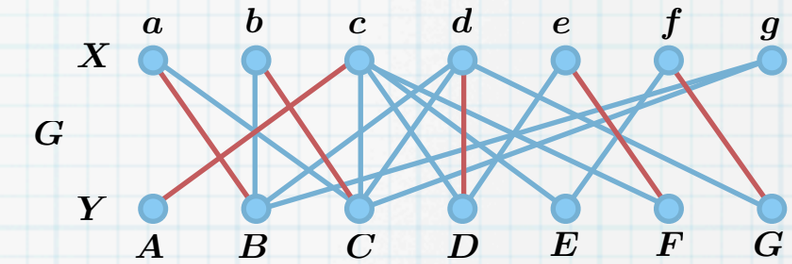
$z = b$

$S = \{a, b, g\}$

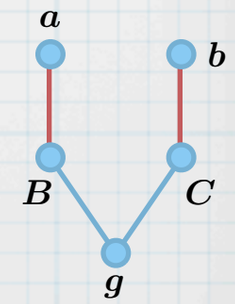


Der Ungarische Algorithmus

* Eingabe: bipartiter Graph $G = (V, E)$ mit $V = X \cup Y$.
* Ausgabe:
  (a) entweder ein Matching M durch das alle Knoten in X gesättigt werden,
  (b) oder eine Knotenteilmenge S mit $|N(S)| < |S|$.



(1) **algorithm** hungarian
(2) wähle ein beliebiges Matching M — $M := \{aB, cC, eD, fG\}$
(3) **while** es gibt einen M–ungesättigten Knoten x_0 **do** — $x_0 = g$
(4) $S := \{x_0\}, T := \emptyset$ — $S = \{g\}, T = \{\}$
(5) **repeat**
(6) **if** $N(S) = T$ **then** — $\{B, C\} = \{B, C\}$
(7) gib S aus
(8) **goto** (19)
(9) **else**
(10) wähle $y \in N(S) \backslash T$ — $y = C$
(11) $T := T \cup \{y\}$ — $T = \{B, C\}$
(12) **if** es gibt ein z mit $\{y, z\} \in M$ **then** — $z = b$
(13) $S := S \cup \{z\}$ — $S = \{a, b, g\}$
(14) **end if**
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**
(19) **end algorithm**



Korrektheitsbeweis

Korrektheitsbeweis

* **Satz 13**:
  Der ungarische Algorithmus arbeitet korrekt.

Korrektheitsbeweis

* **Satz 13**:
  Der ungarische Algorithmus arbeitet korrekt.
* Beweis:

Korrektheitsbeweis

* **Satz 13**:
  Der ungarische Algorithmus arbeitet korrekt.
* Beweis:

  Zu Beginn jeder äußeren Iteration, in Schritt (4), hat S ein Element mehr als T.

Korrektheitsbeweis

* **Satz 13**:
  Der ungarische Algorithmus arbeitet korrekt.
* Beweis:

  Zu Beginn jeder äußeren Iteration, in Schritt (4), hat S ein Element mehr als T.

  Werden weitere gesättigte Knoten angehängt, dann wird ein Element zu S und ein Element zu T hinzugefügt, vgl. Schritt (11) und (13).

Korrektheitsbeweis

* **Satz 13**:
  Der ungarische Algorithmus arbeitet korrekt.
* Beweis:

  Zu Beginn jeder äußeren Iteration, in Schritt (4), hat S ein Element mehr als T.

  Werden weitere gesättigte Knoten angehängt, dann wird ein Element zu S und ein Element zu T hinzugefügt, vgl. Schritt (11) und (13).

  Also hat S stets ein Element mehr als T, d.h. $|T| = |S| - 1$.

Korrektheitsbeweis

* **Satz 13**:
  Der ungarische Algorithmus arbeitet korrekt.
* Beweis:

  Zu Beginn jeder äußeren Iteration, in Schritt (4), hat S ein Element mehr als T.

  Werden weitere gesättigte Knoten angehängt, dann wird ein Element zu S und ein Element zu T hinzugefügt, vgl. Schritt (11) und (13).

  Also hat S stets ein Element mehr als T, d.h. $|T| = |S| - 1$.

  Terminiert der Algorithmus aufgrund von $N(S) = T$, so ist $|N(S)| = |T| = |S| - 1 < |S|$.

Korrektheitsbeweis

* **Satz 13**:
  Der ungarische Algorithmus arbeitet korrekt.
* Beweis:

  Zu Beginn jeder äußeren Iteration, in Schritt (4), hat S ein Element mehr als T.

  Werden weitere gesättigte Knoten angehängt, dann wird ein Element zu S und ein Element zu T hinzugefügt, vgl. Schritt (11) und (13).

  Also hat S stets ein Element mehr als T, d.h. $|T| = |S| - 1$.

  Terminiert der Algorithmus aufgrund von $N(S) = T$, so ist $|N(S)| = |T| = |S| - 1 < |S|$.

  Wegen des Heiratssatzes von Hall ist die Ausgabe, dass es in diesem Fall kein Matching gibt, welches jeden Knoten von X sättigt, korrekt.

Korrektheitsbeweis

* **Satz 13**:
  Der ungarische Algorithmus arbeitet korrekt.
* Beweis:

  Zu Beginn jeder äußeren Iteration, in Schritt (4), hat S ein Element mehr als T.

  Werden weitere gesättigte Knoten angehängt, dann wird ein Element zu S und ein Element zu T hinzugefügt, vgl. Schritt (11) und (13).

  Also hat S stets ein Element mehr als T, d.h. $|T| = |S| - 1$.

  Terminiert der Algorithmus aufgrund von $N(S) = T$, so ist $|N(S)| = |T| = |S| - 1 < |S|$.

  Wegen des Heiratssatzes von Hall ist die Ausgabe, dass es in diesem Fall kein Matching gibt, welches jeden Knoten von X sättigt, korrekt.

  Andernfalls terminiert der Algorithmus erst, wenn jeder Knoten von X gesättigt ist.

Optimale Matchings

Optimale Matchings

* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist.
  Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.

Optimale Matchings

* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.

Optimale Matchings

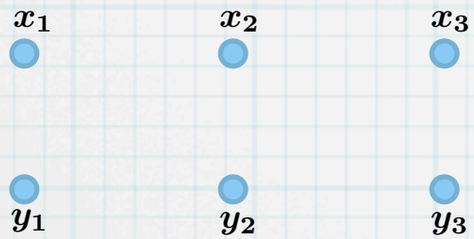
* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.

Optimale Matchings

* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:

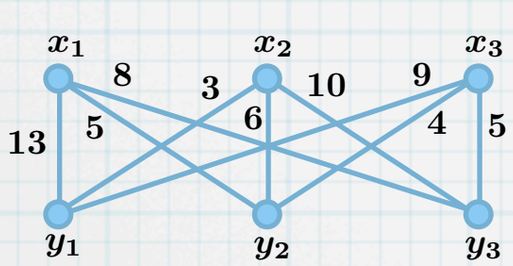
Optimale Matchings

* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:



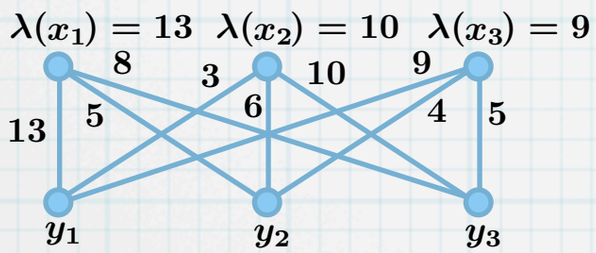
Optimale Matchings

* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:



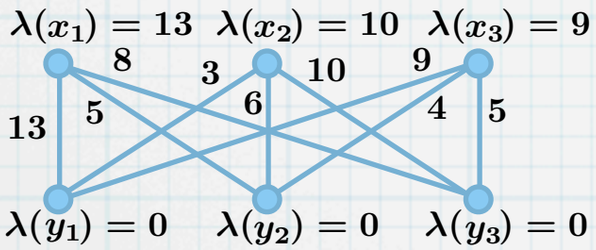
Optimale Matchings

* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:



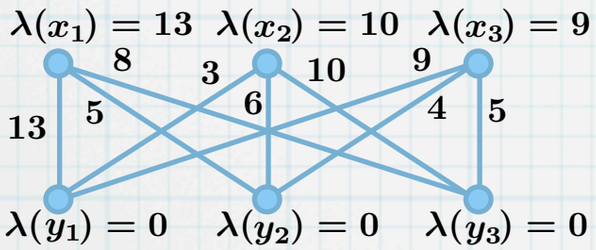
Optimale Matchings

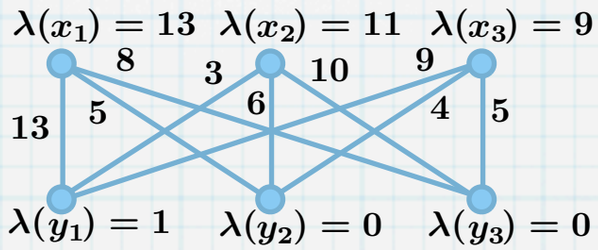
* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:



Optimale Matchings

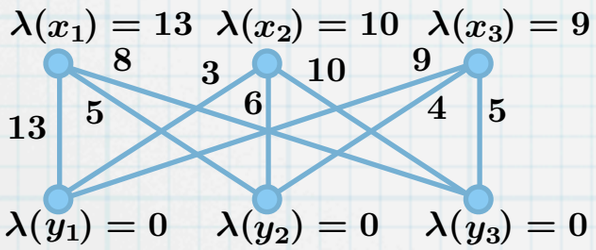
* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:

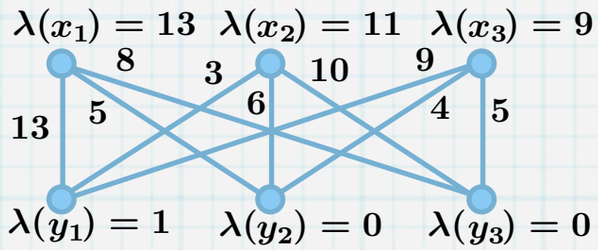




Optimale Matchings

* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:

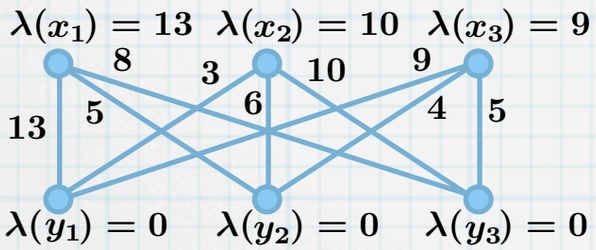


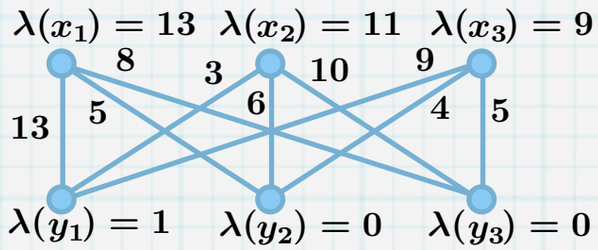


* **Definition 16**:
  Sei λ eine zulässige Knotenkennzeichnung von G und $E_\lambda := \{\{x_i, y_j\} : \lambda(x_i) + \lambda(y_j) = w_{i,j}\}$. Dann wird $G_\lambda := (V, E_\lambda)$ als der **Gleichheitsuntergraph** bzgl. λ bezeichnet.

Optimale Matchings

* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:

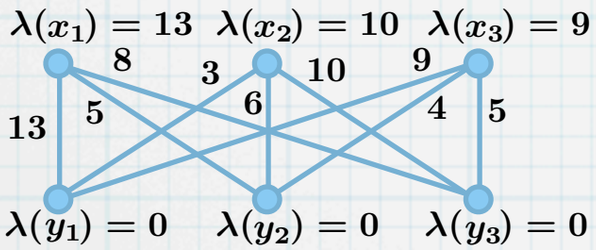


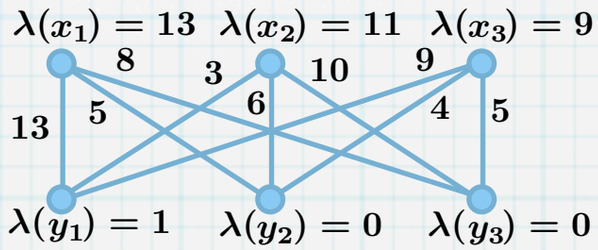


* **Definition 16**:
  Sei λ eine zulässige Knotenkennzeichnung von G und $E_\lambda := \{\{x_i, y_j\} : \lambda(x_i) + \lambda(y_j) = w_{i,j}\}$. Dann wird $G_\lambda := (V, E_\lambda)$ als der **Gleichheitsuntergraph** bzgl. λ bezeichnet.
* Beispiel (Forts.):

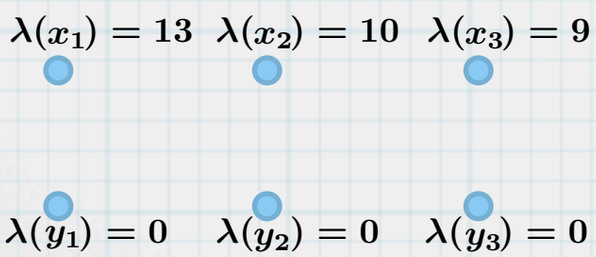
Optimale Matchings

* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:



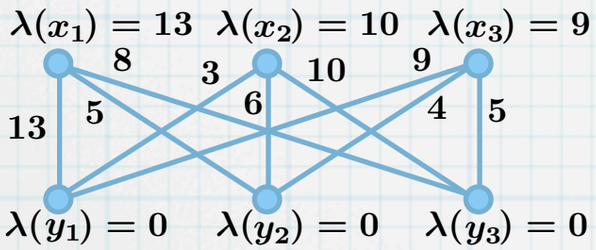


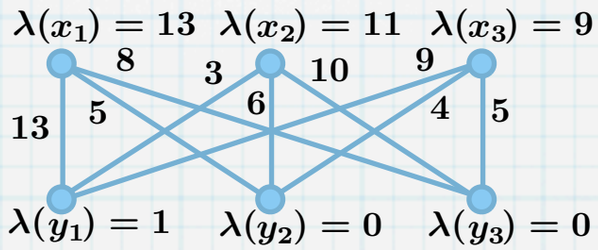
* **Definition 16**:
  Sei λ eine zulässige Knotenkennzeichnung von G und $E_\lambda := \{\{x_i, y_j\} : \lambda(x_i) + \lambda(y_j) = w_{i,j}\}$. Dann wird $G_\lambda := (V, E_\lambda)$ als der **Gleichheitsuntergraph** bzgl. λ bezeichnet.
* Beispiel (Forts.):



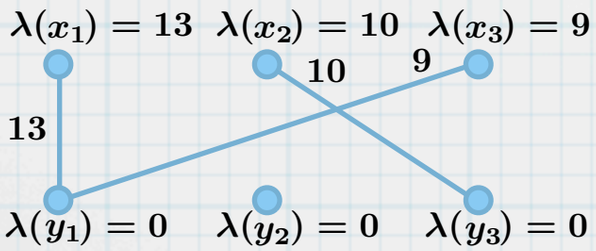
Optimale Matchings

* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:



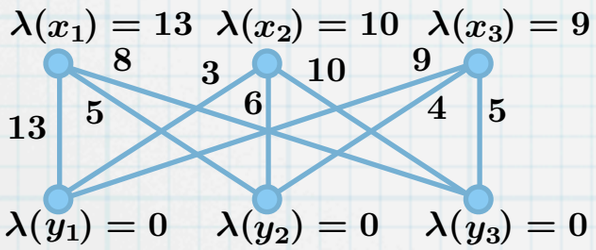


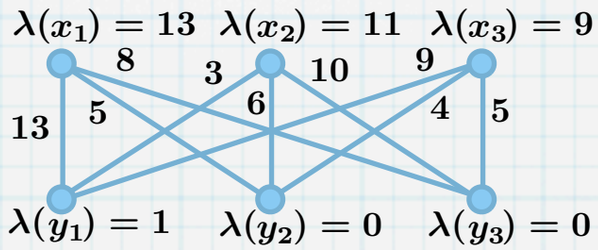
* **Definition 16**:
  Sei λ eine zulässige Knotenkennzeichnung von G und $E_\lambda := \{\{x_i, y_j\} : \lambda(x_i) + \lambda(y_j) = w_{i,j}\}$. Dann wird $G_\lambda := (V, E_\lambda)$ als der **Gleichheitsuntergraph** bzgl. λ bezeichnet.
* Beispiel (Forts.):



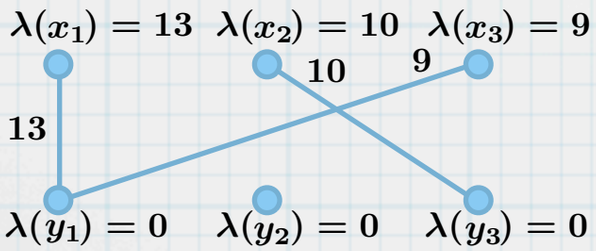
Optimale Matchings

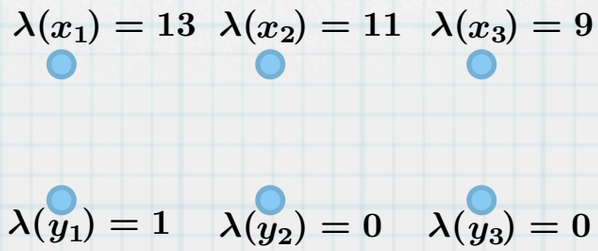
* **Definition 14**:
  Sei $G = (V, E)$ ein bewerteter vollständiger bipartiter Graph mit Bipartition $V = X \cup Y$, wobei $X = \{x_1, \ldots, x_n\}$ und $Y = \{y_1, \ldots, y_n\}$ und $w_{i,j}$ die Bewertung der Kante $\{x_i, y_j\}$ ist. Ein **optimales Matching** ist ein maximal-bewertetes perfektes Matching in G.
* **Definition 15**:
  Sei G wie zuvor. Eine Abbildung $\lambda : V \to \mathbb{R}$ heißt **zulässige Knotenkennzeichnung** von G, wenn $\lambda(x_i) + \lambda(y_j) \geq w_{i,j}$ für alle $x_i \in X, y_j \in Y$ gilt.
* Eine zulässige Knotenkennzeichnung kann stets erzielt werden durch die Setzung $\lambda(x_i) := \max\{w_{i,j} : 1 \leq j \leq n\}$ für alle $x_i \in X$ und $\lambda(y_j) := 0$ für alle $y_j \in Y$.
* Beispiel:





* **Definition 16**:
  Sei λ eine zulässige Knotenkennzeichnung von G und $E_\lambda := \{\{x_i, y_j\} : \lambda(x_i) + \lambda(y_j) = w_{i,j}\}$. Dann wird $G_\lambda := (V, E_\lambda)$ als der **Gleichheitsuntergraph** bzgl. λ bezeichnet.
* Beispiel (Forts.):





Gleichheitsuntergraphen und optimale Matchings

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

  Sei M^* ein perfektes Matching für G_λ.

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

  Sei M^* ein perfektes Matching für G_λ.

  Da G_λ die gleichen Knoten wie G enthält, ist M^* ein perfektes Matching von G.

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

  Sei M^* ein perfektes Matching für G_λ.

  Da G_λ die gleichen Knoten wie G enthält, ist M^* ein perfektes Matching von G.

  Es ist $w(M^*) = \sum_{\{x_i, y_j\} \in M^*} w_{i,j}$.

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

  Sei M^* ein perfektes Matching für G_λ.

  Da G_λ die gleichen Knoten wie G enthält, ist M^* ein perfektes Matching von G.

  Es ist $w(M^*) = \sum_{\{x_i, y_j\} \in M^*} w_{i,j}$.

  Für jede Kante $\{x_i, y_j\} \in M^*$ ist $\{x_i, y_j\} \in G_\lambda$, also $w_{i,j} = \lambda(x_i) + \lambda(y_j)$.

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

  Sei M^* ein perfektes Matching für G_λ.

  Da G_λ die gleichen Knoten wie G enthält, ist M^* ein perfektes Matching von G.

  Es ist $w(M^*) = \sum_{\{x_i, y_j\} \in M^*} w_{i,j}$.

  Für jede Kante $\{x_i, y_j\} \in M^*$ ist $\{x_i, y_j\} \in G_\lambda$, also $w_{i,j} = \lambda(x_i) + \lambda(y_j)$.

  Da das Matching M^* perfekt ist, verbinden seine Kanten alle Knoten von G genau ein Mal.

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

  Sei M^* ein perfektes Matching für G_λ.

  Da G_λ die gleichen Knoten wie G enthält, ist M^* ein perfektes Matching von G.

  Es ist $w(M^*) = \sum_{\{x_i,y_j\} \in M^*} w_{i,j}$.

  Für jede Kante $\{x_i, y_j\} \in M^*$ ist $\{x_i, y_j\} \in G_\lambda$, also $w_{i,j} = \lambda(x_i) + \lambda(y_j)$.

  Da das Matching M^* perfekt ist, verbinden seine Kanten alle Knoten von G genau ein Mal.

  Damit ist $w(M^*) = \sum_{\{x_i,y_j\} \in M^*} w_{i,j} = \sum_{i=1}^{n} (\lambda(x_i) + \lambda(y_i))$.

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

  Sei M^* ein perfektes Matching für G_λ.

  Da G_λ die gleichen Knoten wie G enthält, ist M^* ein perfektes Matching von G.

  Es ist $w(M^*) = \sum_{\{x_i,y_j\} \in M^*} w_{i,j}$.

  Für jede Kante $\{x_i, y_j\} \in M^*$ ist $\{x_i, y_j\} \in G_\lambda$, also $w_{i,j} = \lambda(x_i) + \lambda(y_j)$.

  Da das Matching M^* perfekt ist, verbinden seine Kanten alle Knoten von G genau ein Mal.

  Damit ist $w(M^*) = \sum_{\{x_i,y_j\} \in M^*} w_{i,j} = \sum_{i=1}^{n} (\lambda(x_i) + \lambda(y_i))$.

  Sei nun M ein beliebiges perfektes Matching von G.

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

  Sei M^* ein perfektes Matching für G_λ.

  Da G_λ die gleichen Knoten wie G enthält, ist M^* ein perfektes Matching von G.

  Es ist $w(M^*) = \sum_{\{x_i,y_j\} \in M^*} w_{i,j}$.

  Für jede Kante $\{x_i, y_j\} \in M^*$ ist $\{x_i, y_j\} \in G_\lambda$, also $w_{i,j} = \lambda(x_i) + \lambda(y_j)$.

  Da das Matching M^* perfekt ist, verbinden seine Kanten alle Knoten von G genau ein Mal.

  Damit ist $w(M^*) = \sum_{\{x_i,y_j\} \in M^*} w_{i,j} = \sum_{i=1}^{n} (\lambda(x_i) + \lambda(y_i))$.

  Sei nun M ein beliebiges perfektes Matching von G.

  Dann gilt $w(M) = \sum_{\{x_i,y_j\} \in M} w_{i,j} \leq \sum_{i=1}^{n} (\lambda(x_i) + \lambda(y_i))$.

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

  Sei M^* ein perfektes Matching für G_λ.

  Da G_λ die gleichen Knoten wie G enthält, ist M^* ein perfektes Matching von G.

  Es ist $w(M^*) = \sum_{\{x_i,y_j\} \in M^*} w_{i,j}$.

  Für jede Kante $\{x_i, y_j\} \in M^*$ ist $\{x_i, y_j\} \in G_\lambda$, also $w_{i,j} = \lambda(x_i) + \lambda(y_j)$.

  Da das Matching M^* perfekt ist, verbinden seine Kanten alle Knoten von G genau ein Mal.

  Damit ist $w(M^*) = \sum_{\{x_i,y_j\} \in M^*} w_{i,j} = \sum_{i=1}^{n} (\lambda(x_i) + \lambda(y_i))$.

  Sei nun M ein beliebiges perfektes Matching von G.

  Dann gilt $w(M) = \sum_{\{x_i,y_j\} \in M} w_{i,j} \leq \sum_{i=1}^{n} (\lambda(x_i) + \lambda(y_i))$.

  Es folgt $w(M^*) \geq w(M)$.

Gleichheitsuntergraphen und optimale Matchings

* **Satz 17**:
  Sei λ eine zulässige Knotenkennzeichnung für den bewerteten vollständigen bipartiten Graphen G. Wenn der Gleichheitsuntergraph G_λ ein perfektes Matching M^* enthält, dann ist M^* ein optimales Matching für G.
* Beweis:

  Sei M^* ein perfektes Matching für G_λ.

  Da G_λ die gleichen Knoten wie G enthält, ist M^* ein perfektes Matching von G.

  Es ist $w(M^*) = \sum_{\{x_i,y_j\} \in M^*} w_{i,j}$.

  Für jede Kante $\{x_i, y_j\} \in M^*$ ist $\{x_i, y_j\} \in G_\lambda$, also $w_{i,j} = \lambda(x_i) + \lambda(y_j)$.

  Da das Matching M^* perfekt ist, verbinden seine Kanten alle Knoten von G genau ein Mal.

  Damit ist $w(M^*) = \sum_{\{x_i,y_j\} \in M^*} w_{i,j} = \sum_{i=1}^{n} (\lambda(x_i) + \lambda(y_i))$.

  Sei nun M ein beliebiges perfektes Matching von G.

  Dann gilt $w(M) = \sum_{\{x_i,y_j\} \in M} w_{i,j} \leq \sum_{i=1}^{n} (\lambda(x_i) + \lambda(y_i))$.

  Es folgt $w(M^*) \geq w(M)$.

  Da M ein beliebiges Matching von G war, ist M^* daher ein optimales Matching von G.

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.
  1. Fall, die ungarische Methode liefert ein perfektes Matching M.*

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.
  1. Fall, die ungarische Methode liefert ein perfektes Matching M^*.
     Nach Satz 17 ist M^* dann ein optimales Matching für G. Fertig.

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.
  1. Fall, die ungarische Methode liefert ein perfektes Matching M^*.
     Nach Satz 17 ist M^* dann ein optimales Matching für G. Fertig.
  2. Fall, sie liefert ein nicht-perfektes Matching M', das nicht weiter vergrößert werden kann.

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.
  1. Fall, die ungarische Methode liefert ein perfektes Matching M^*.
     Nach Satz 17 ist M^* dann ein optimales Matching für G. Fertig.
  2. Fall, sie liefert ein nicht-perfektes Matching M', das nicht weiter vergrößert werden kann.
     Dann gibt es einen M'-alternierenden Baum H, der keinen M'-erweiternden Weg hat.

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.
  1. Fall, die ungarische Methode liefert ein perfektes Matching M^*.
     Nach Satz 17 ist M^* dann ein optimales Matching für G. Fertig.
  2. Fall, sie liefert ein nicht-perfektes Matching M', das nicht weiter vergrößert werden kann.
     Dann gibt es einen M'-alternierenden Baum H, der keinen M'-erweiternden Weg hat.
     Dieser Baum kann in G_λ nicht mehr weiter wachsen.

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.
  1. Fall, die ungarische Methode liefert ein perfektes Matching M^*.
     Nach Satz 17 ist M^* dann ein optimales Matching für G. Fertig.
  2. Fall, sie liefert ein nicht-perfektes Matching M', das nicht weiter vergrößert werden kann.
     Dann gibt es einen M'-alternierenden Baum H, der keinen M'-erweiternden Weg hat.
     Dieser Baum kann in G_λ nicht mehr weiter wachsen.
     Wir verändern nun λ zu λ', so dass

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.
  1. Fall, die ungarische Methode liefert ein perfektes Matching M^*.
     Nach Satz 17 ist M^* dann ein optimales Matching für G. Fertig.
  2. Fall, sie liefert ein nicht-perfektes Matching M', das nicht weiter vergrößert werden kann.
     Dann gibt es einen M'-alternierenden Baum H, der keinen M'-erweiternden Weg hat.
     Dieser Baum kann in G_λ nicht mehr weiter wachsen.
     Wir verändern nun λ zu λ', so dass
     λ' ebenso eine zulässige Knotenkennzeichnung ist,

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.
  1. Fall, die ungarische Methode liefert ein perfektes Matching M^*.
     Nach Satz 17 ist M^* dann ein optimales Matching für G. Fertig.
  2. Fall, sie liefert ein nicht-perfektes Matching M', das nicht weiter vergrößert werden kann.
     Dann gibt es einen M'-alternierenden Baum H, der keinen M'-erweiternden Weg hat.
     Dieser Baum kann in G_λ nicht mehr weiter wachsen.
     Wir verändern nun λ zu λ', so dass
     λ' ebenso eine zulässige Knotenkennzeichnung ist,
     der Gleichheitsuntergraph $G_{\lambda'}$ sowohl M' als auch den M'-alternierenden Baum H enthält,

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.
  1. Fall, die ungarische Methode liefert ein perfektes Matching M^*.
     Nach Satz 17 ist M^* dann ein optimales Matching für G. Fertig.
  2. Fall, sie liefert ein nicht-perfektes Matching M', das nicht weiter vergrößert werden kann.
     Dann gibt es einen M'-alternierenden Baum H, der keinen M'-erweiternden Weg hat.
     Dieser Baum kann in G_λ nicht mehr weiter wachsen.
     Wir verändern nun λ zu λ', so dass
     λ' ebenso eine zulässige Knotenkennzeichnung ist,
     der Gleichheitsuntergraph $G_{\lambda'}$ sowohl M' als auch den M'-alternierenden Baum H enthält,
     der M'-alternierende Baum H in $G_{\lambda'}$ weiter wachsen kann.

Ein Algorithmus für optimale Matchings in vollständigen bipartiten Graphen

* Beginne mit einer zulässigen Knotenkennzeichnung λ für G.
  Bestimme den Gleichheitsuntergraphen G_λ.
  Wähle ein beliebiges Matching M in G_λ.
  Wende die ungarische Methode auf G_λ und M an.
  1. Fall, die ungarische Methode liefert ein perfektes Matching M^*.
     Nach Satz 17 ist M^* dann ein optimales Matching für G. Fertig.
  2. Fall, sie liefert ein nicht-perfektes Matching M', das nicht weiter vergrößert werden kann.
     Dann gibt es einen M'-alternierenden Baum H, der keinen M'-erweiternden Weg hat.
     Dieser Baum kann in G_λ nicht mehr weiter wachsen.
     Wir verändern nun λ zu λ', so dass
     - λ' ebenso eine zulässige Knotenkennzeichnung ist,
     - der Gleichheitsuntergraph $G_{\lambda'}$ sowohl M' als auch den M'-alternierenden Baum H enthält,
     - der M'-alternierende Baum H in $G_{\lambda'}$ weiter wachsen kann.
* Bemerke, dass der M'-alternierende Baum H so erzeugt wurde, dass es Untermengen $S \subseteq X$ und $T \subseteq Y$ gibt mit $N_{G_\lambda}(S) = T$.

Defekte

Defekte

* **Definition 18**:
Seien $S \subseteq X$ und $T \subseteq Y$ die durch die ungarische Methode erzeugten Untermengen mit $N_{G_\lambda}(S) = T$. Dann ist $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$ der **Defekt** von λ.

Defekte

* **Definition 18**:
  Seien $S \subseteq X$ und $T \subseteq Y$ die durch die ungarische Methode erzeugten Untermengen mit $N_{G_\lambda}(S) = T$. Dann ist $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$ der **Defekt** von λ.
* Für einen Knoten v in G setze

$$\lambda'(v) := \begin{cases} \lambda(v) - d_\lambda & ; \quad v \in S, \\ \lambda(v) + d_\lambda & ; \quad v \in T, \\ \lambda(v) & ; \quad v \notin S \cup T. \end{cases}$$

Defekte

* **Definition 18**:
  Seien $S \subseteq X$ und $T \subseteq Y$ die durch die ungarische Methode erzeugten Untermengen mit $N_{G_\lambda}(S) = T$. Dann ist $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$ der **Defekt** von λ.
* Für einen Knoten v in G setze
  $$\lambda'(v) := \begin{cases} \lambda(v) - d_\lambda & ; \quad v \in S, \\ \lambda(v) + d_\lambda & ; \quad v \in T, \\ \lambda(v) & ; \quad v \notin S \cup T. \end{cases}$$
* Dieses λ' erfüllt die vorhin geforderten Eigenschaften:
  λ' ist eine zulässige Knotenkennzeichnung,
  der Gleichheitsuntergraph $G_{\lambda'}$ enthält M' und den M'–alternierenden Baum H,
  der M'–alternierende Baum H kann in $G_{\lambda'}$ weiter wachsen.

Defekte

* **Definition 18**:
  Seien $S \subseteq X$ und $T \subseteq Y$ die durch die ungarische Methode erzeugten Untermengen mit $N_{G_\lambda}(S) = T$. Dann ist $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$ der **Defekt** von λ.
* Für einen Knoten v in G setze
$$\lambda'(v) := \begin{cases} \lambda(v) - d_\lambda & ; \quad v \in S, \\ \lambda(v) + d_\lambda & ; \quad v \in T, \\ \lambda(v) & ; \quad v \notin S \cup T. \end{cases}$$
* Dieses λ' erfüllt die vorhin geforderten Eigenschaften:
  λ' ist eine zulässige Knotenkennzeichnung,
  der Gleichheitsuntergraph $G_{\lambda'}$ enthält M' und den M'–alternierenden Baum H,
  der M'–alternierende Baum H kann in $G_{\lambda'}$ weiter wachsen.
* Dann wird das Verfahren mit $G_{\lambda'}$ (an Stelle von G_λ) fortgesetzt.

Defekte

* **Definition 18**:
  Seien $S \subseteq X$ und $T \subseteq Y$ die durch die ungarische Methode erzeugten Untermengen mit $N_{G_\lambda}(S) = T$. Dann ist $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$ der **Defekt** von λ.
* Für einen Knoten v in G setze
$$\lambda'(v) := \begin{cases} \lambda(v) - d_\lambda & ; \quad v \in S, \\ \lambda(v) + d_\lambda & ; \quad v \in T, \\ \lambda(v) & ; \quad v \notin S \cup T. \end{cases}$$
* Dieses λ' erfüllt die vorhin geforderten Eigenschaften:
  λ' ist eine zulässige Knotenkennzeichnung,
  der Gleichheitsuntergraph $G_{\lambda'}$ enthält M' und den M'–alternierenden Baum H,
  der M'–alternierende Baum H kann in $G_{\lambda'}$ weiter wachsen.
* Dann wird das Verfahren mit $G_{\lambda'}$ (an Stelle von G_λ) fortgesetzt.
* Die Knotenkennzeichnung wird so lange geändert, bis ein perfektes Matching im Gleichheitsuntergraphen entsteht. Dieses ist nach Satz 17 dann ein optimales Matching.

Defekte

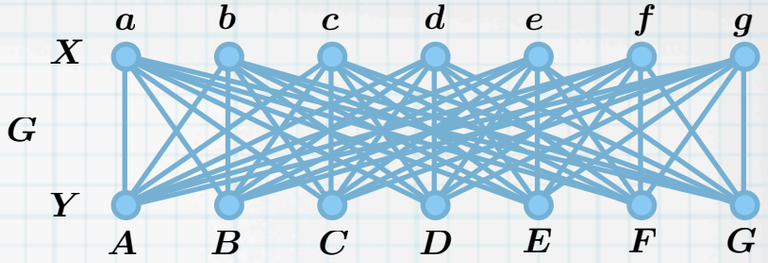
* **Definition 18**:
  Seien $S \subseteq X$ und $T \subseteq Y$ die durch die ungarische Methode erzeugten Untermengen mit $N_{G_\lambda}(S) = T$. Dann ist $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$ der **Defekt** von λ.
* Für einen Knoten v in G setze
$$\lambda'(v) := \begin{cases} \lambda(v) - d_\lambda & ; \quad v \in S, \\ \lambda(v) + d_\lambda & ; \quad v \in T, \\ \lambda(v) & ; \quad v \notin S \cup T. \end{cases}$$
* Dieses λ' erfüllt die vorhin geforderten Eigenschaften:
  λ' ist eine zulässige Knotenkennzeichnung,
  der Gleichheitsuntergraph $G_{\lambda'}$ enthält M' und den M'–alternierenden Baum H,
  der M'–alternierende Baum H kann in $G_{\lambda'}$ weiter wachsen.
* Dann wird das Verfahren mit $G_{\lambda'}$ (an Stelle von G_λ) fortgesetzt.
* Die Knotenkennzeichnung wird so lange geändert, bis ein perfektes Matching im Gleichheitsuntergraphen entsteht. Dieses ist nach Satz 17 dann ein optimales Matching.
* Dieses Verfahren ist nach Kuhn (1955) und Munkres (1957) benannt.

Der Algorithmus von Kuhn-Munkres

Der Algorithmus von Kuhn-Munkres

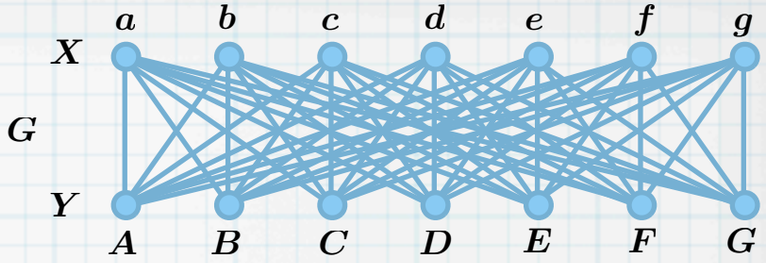
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.

Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.

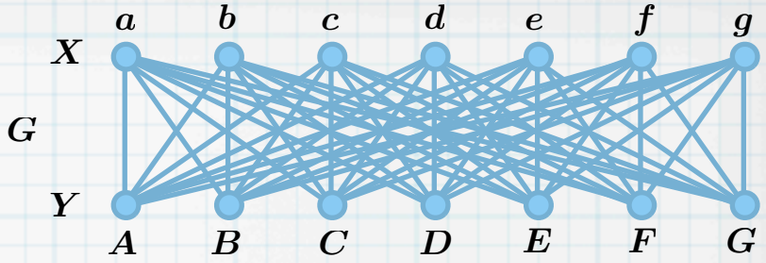
Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix}$$

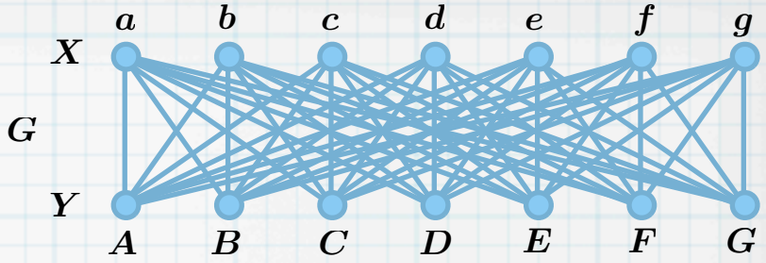
Der Algorithmus von Kuhn-Munkres



- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix}$$

Der Algorithmus von Kuhn-Munkres

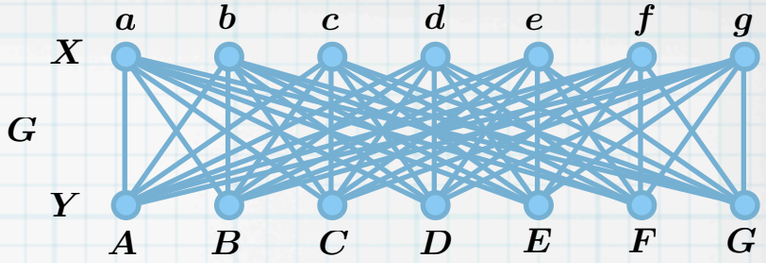


$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres

Der Algorithmus von Kuhn-Munkres

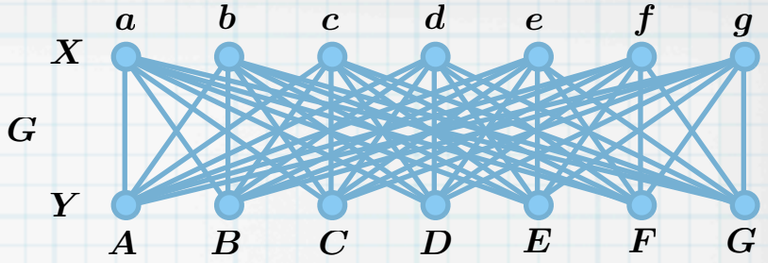


- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix}$$

Der Algorithmus von Kuhn-Munkres



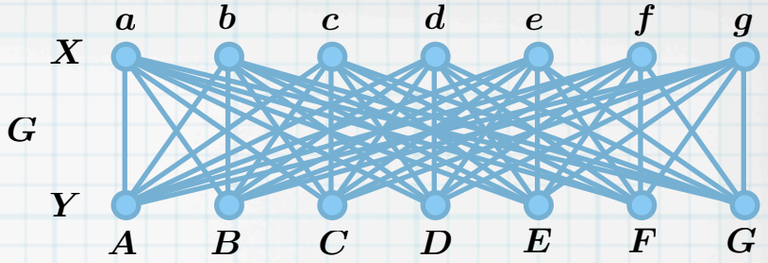
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres

(2) wähle eine zul. Knotenkennzeichnung λ

Der Algorithmus von Kuhn-Munkres



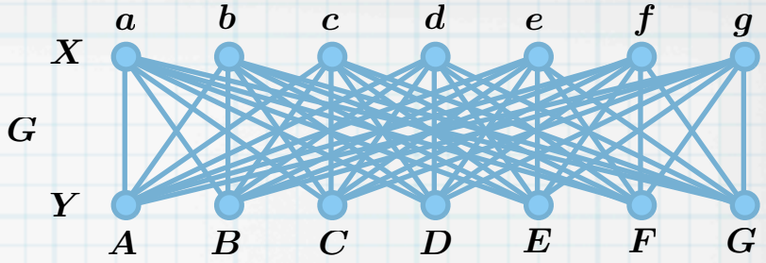
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres

(2) wähle eine zul. Knotenkennzeichung λ

Der Algorithmus von Kuhn-Munkres



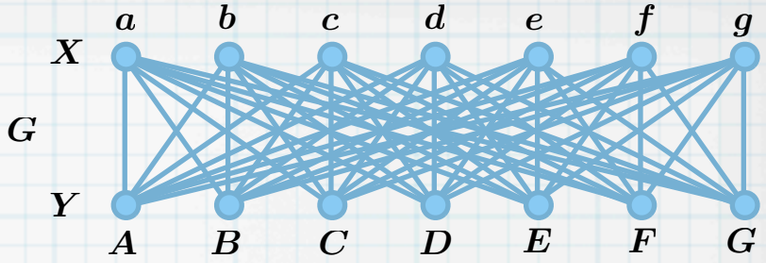
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres

(2) wähle eine zul. Knotenkennzeichnung λ

Der Algorithmus von Kuhn-Munkres



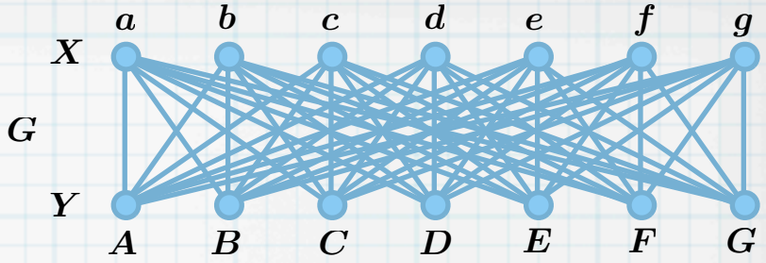
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres

(2) wähle eine zul. Knotenkennzeichnung λ

Der Algorithmus von Kuhn-Munkres

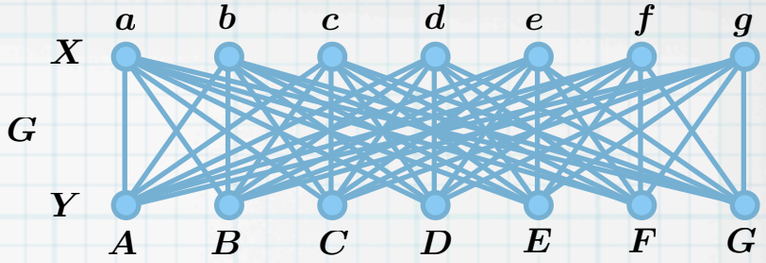


$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ

Der Algorithmus von Kuhn-Munkres

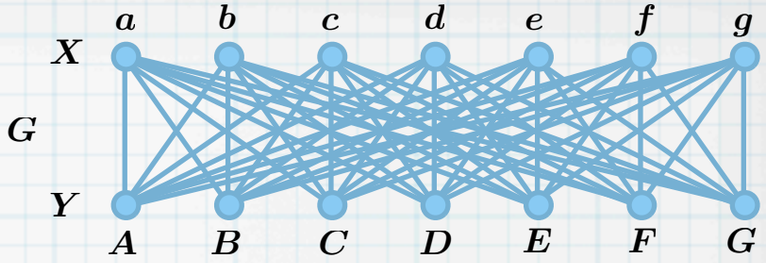


$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ

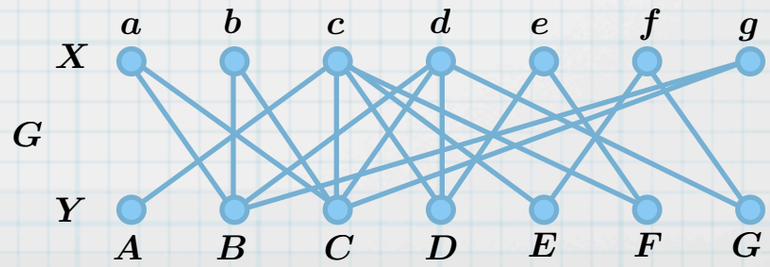
Der Algorithmus von Kuhn-Munkres



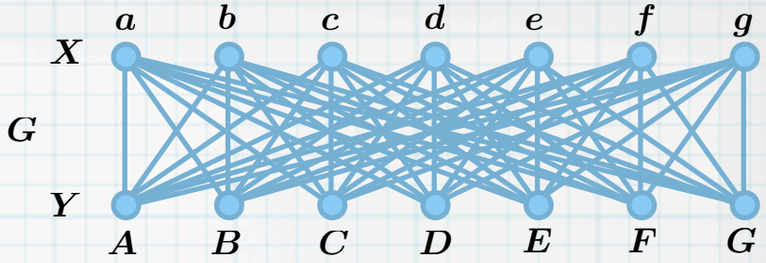
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ



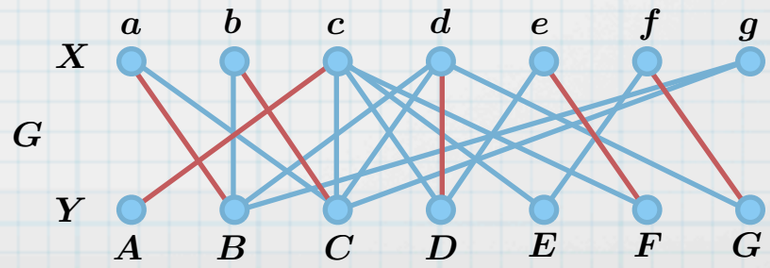
Der Algorithmus von Kuhn-Munkres



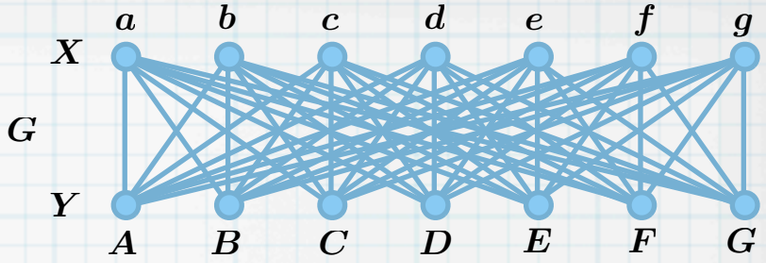
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ



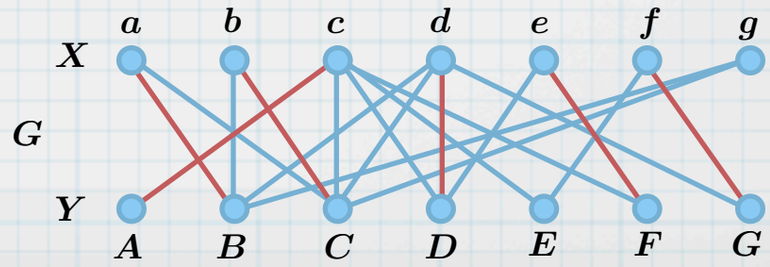
Der Algorithmus von Kuhn-Munkres



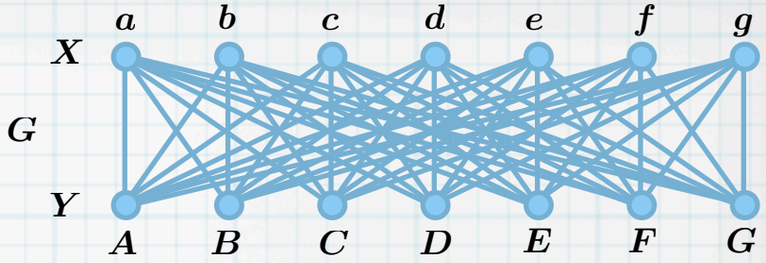
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**



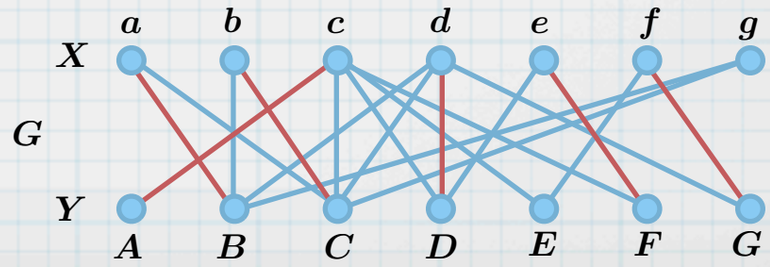
Der Algorithmus von Kuhn-Munkres



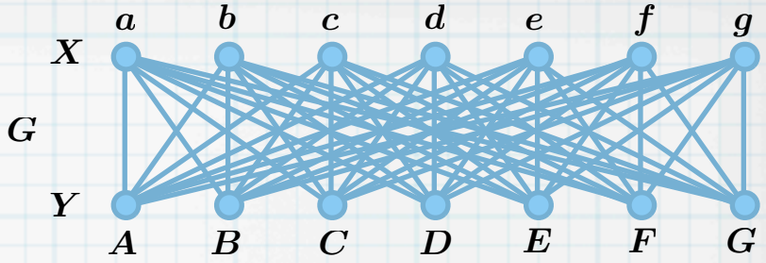
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M-ungesättigten Knoten x_0 **do**



Der Algorithmus von Kuhn-Munkres

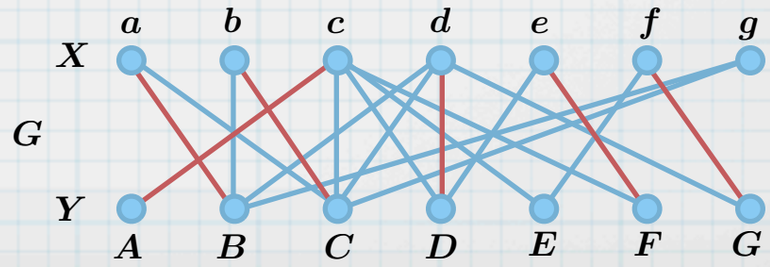


$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

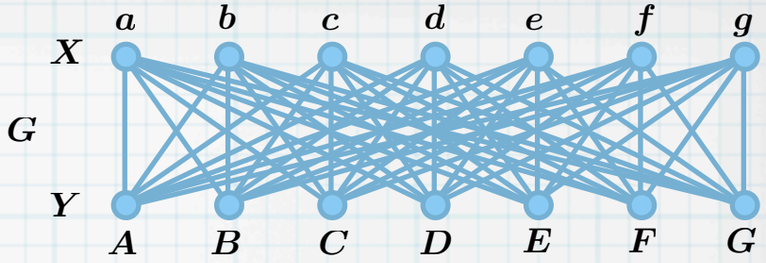
$x_0 = g$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**



Der Algorithmus von Kuhn-Munkres

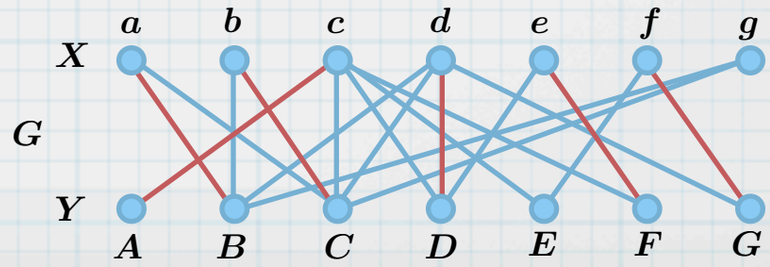


$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

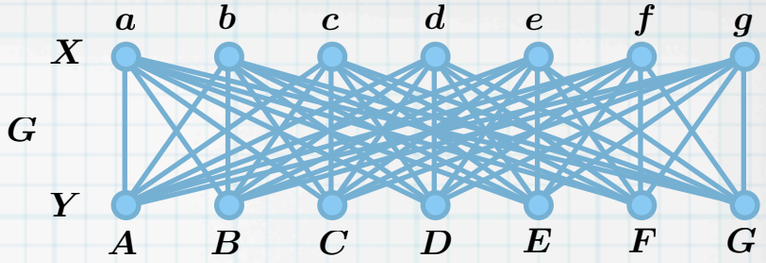
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$

$x_0 = g$



Der Algorithmus von Kuhn-Munkres

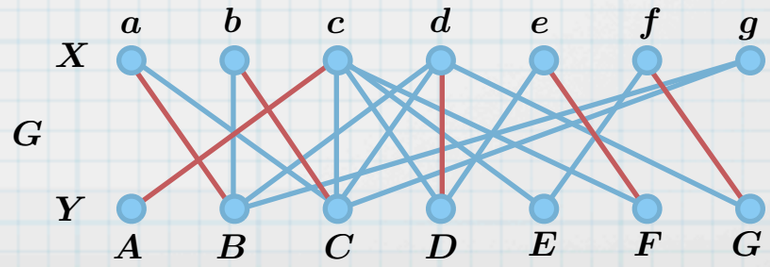


$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

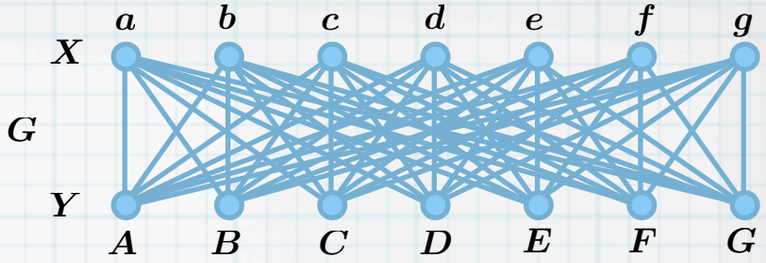
$x_0 = g$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$



Der Algorithmus von Kuhn-Munkres

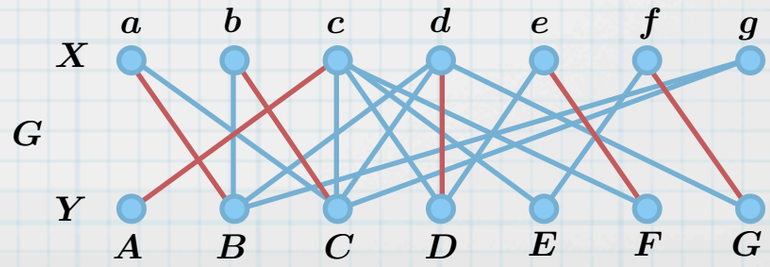


$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

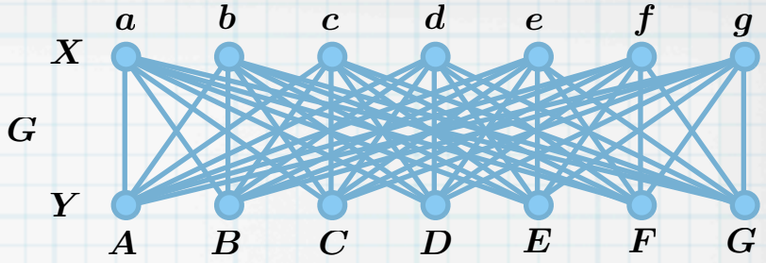
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$



Der Algorithmus von Kuhn-Munkres



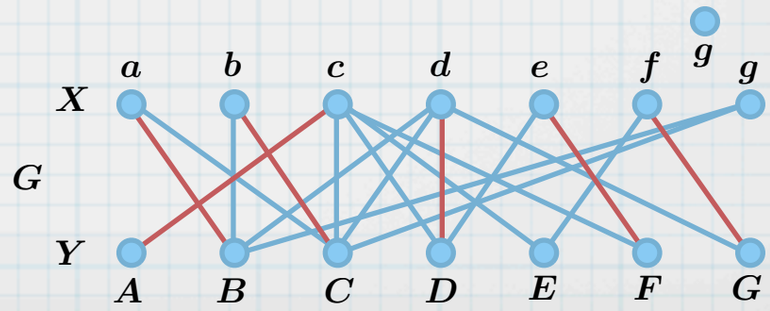
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

$$x_0 = g$$

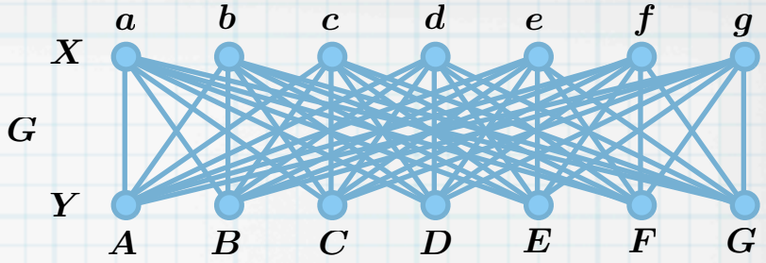
$$S = \{g\}, T = \{\}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$



Der Algorithmus von Kuhn-Munkres



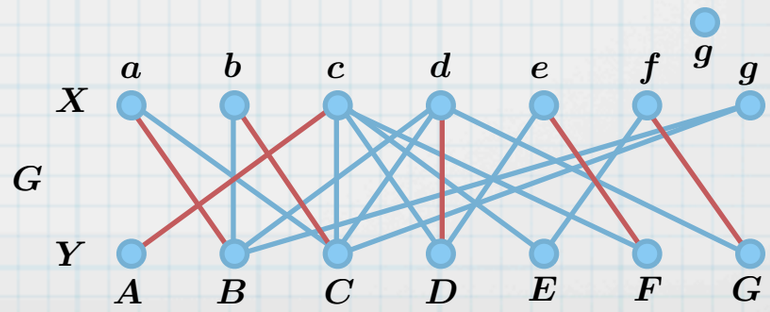
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

$$x_0 = g$$

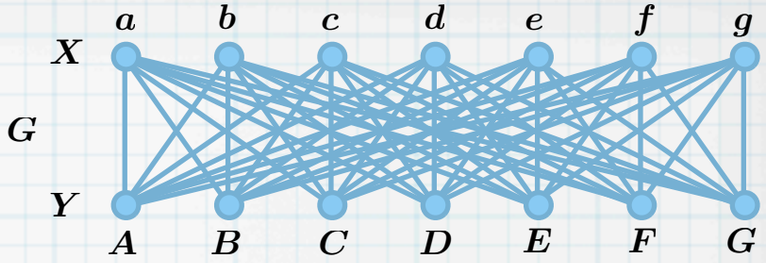
$$S = \{g\}, T = \{\}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**



Der Algorithmus von Kuhn-Munkres



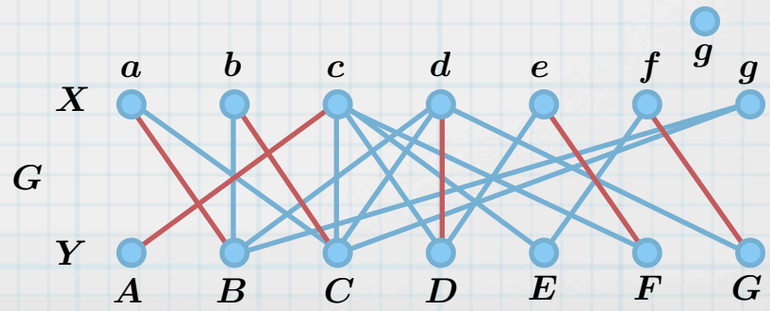
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

$$x_0 = g$$

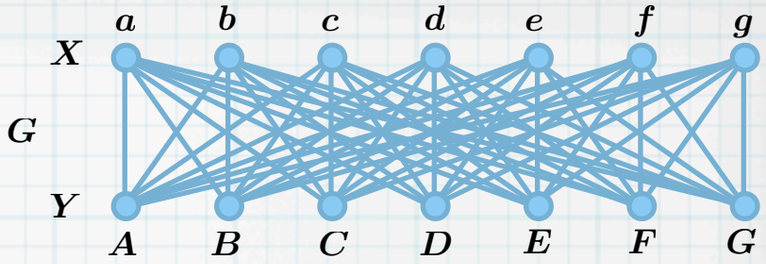
$$S = \{g\}, T = \{\}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**



Der Algorithmus von Kuhn-Munkres



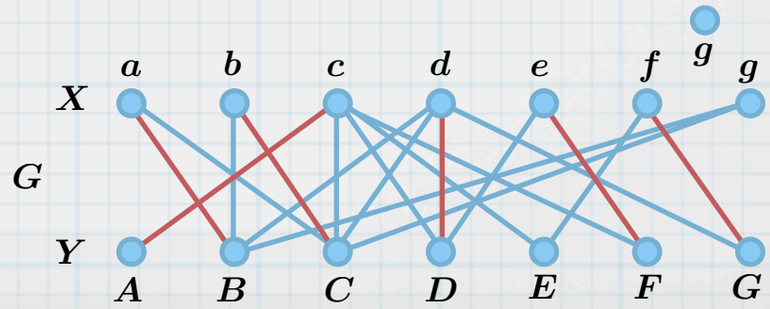
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

$$x_0 = g$$

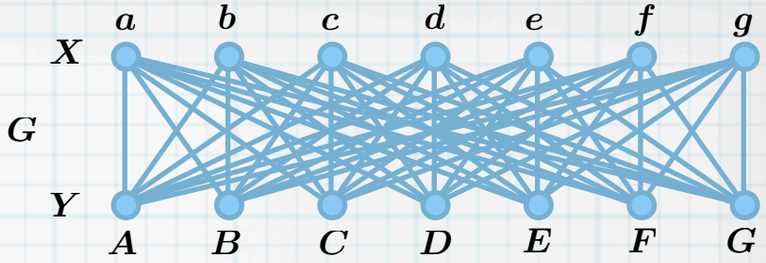
$$S = \{g\}, T = \{\}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**



Der Algorithmus von Kuhn-Munkres



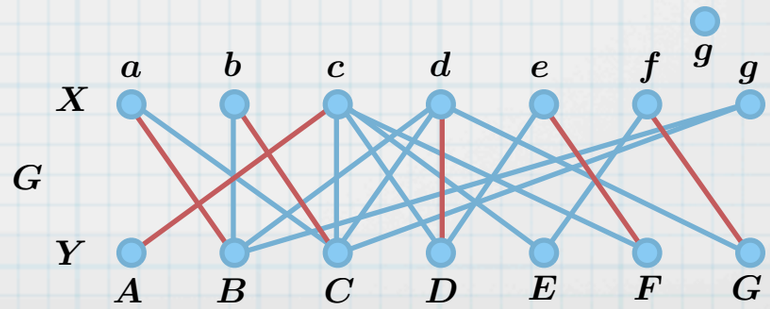
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**

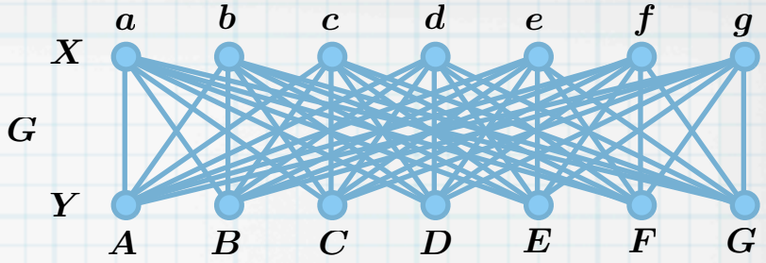
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

$$x_0 = g$$

$$S = \{g\}, T = \{\}$$



Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

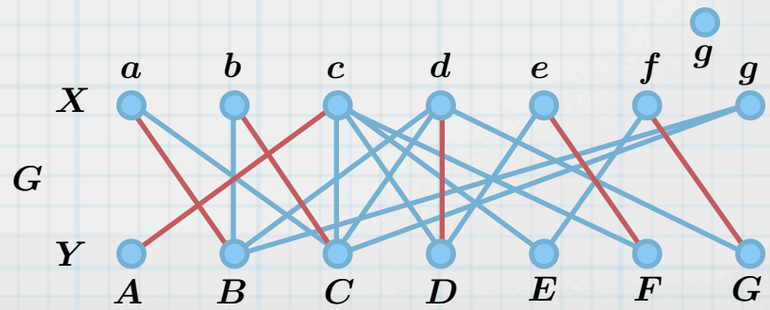
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**

$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

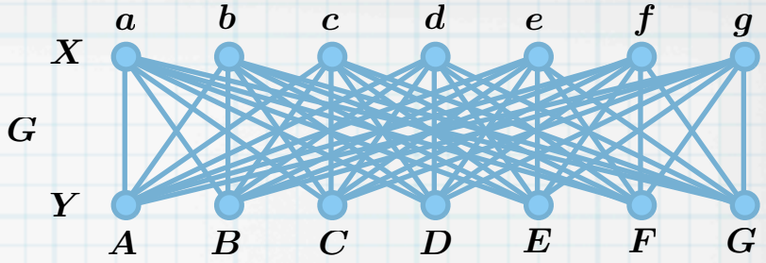
$$x_0 = g$$

$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$



Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

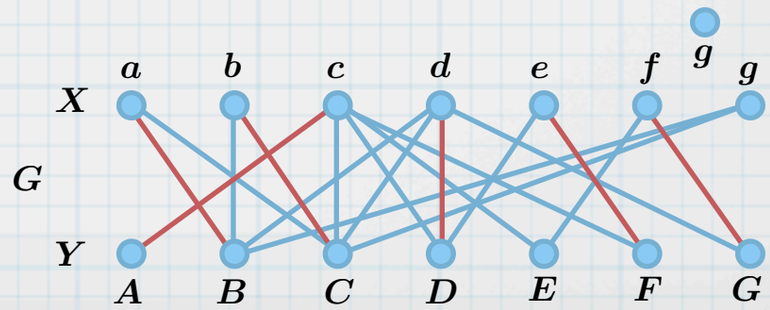
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$

$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

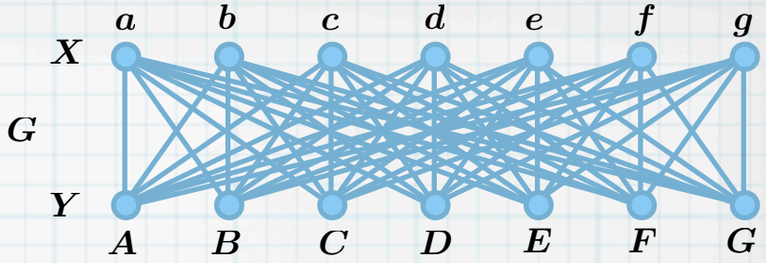
$$x_0 = g$$

$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$



Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

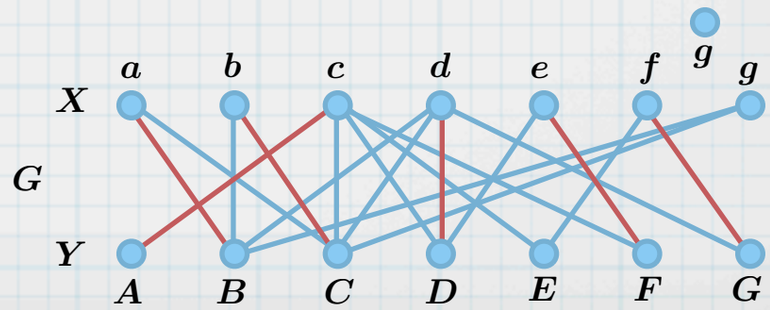
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

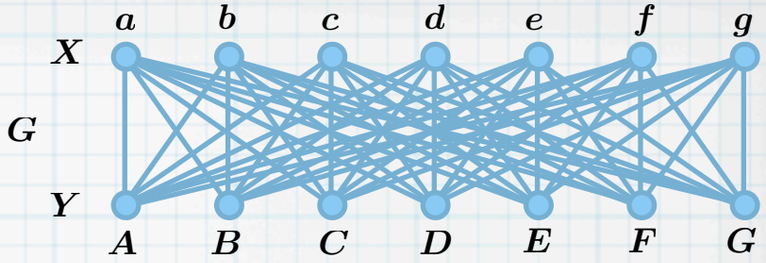
$$x_0 = g$$

$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$



Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

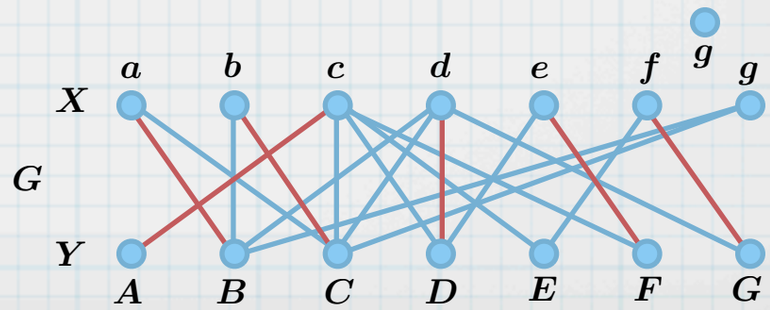
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

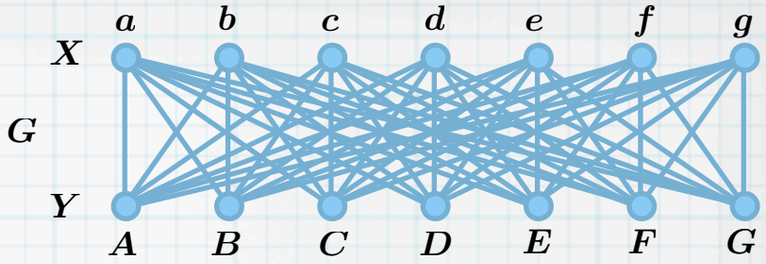
$$x_0 = g$$

$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$



Der Algorithmus von Kuhn-Munkres



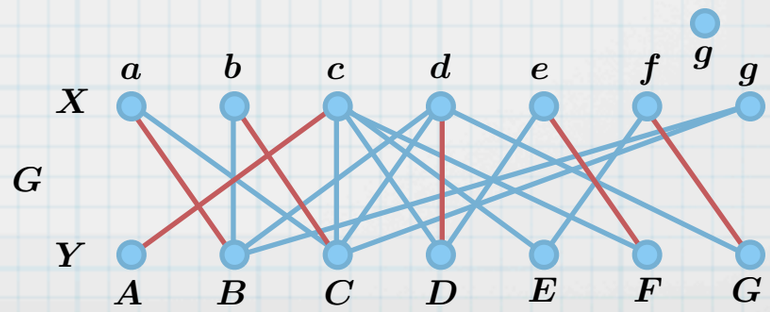
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

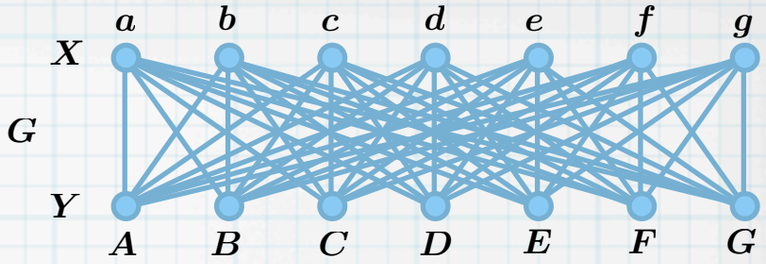
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$



Der Algorithmus von Kuhn-Munkres



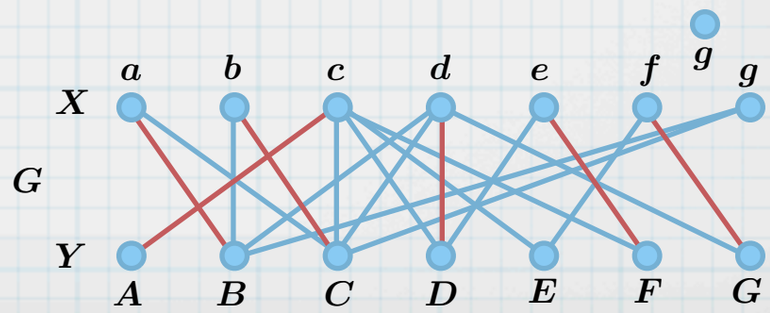
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

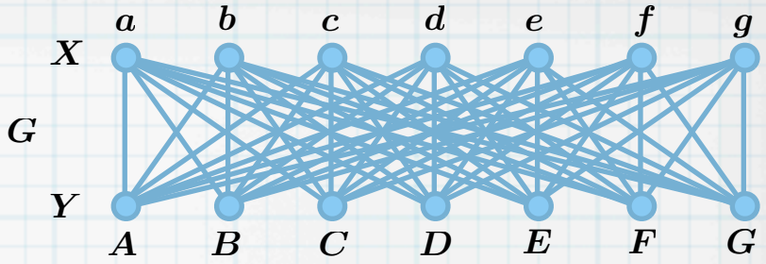
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$



Der Algorithmus von Kuhn-Munkres



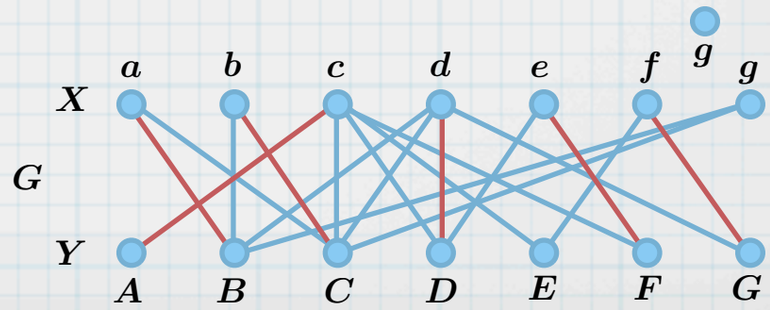
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

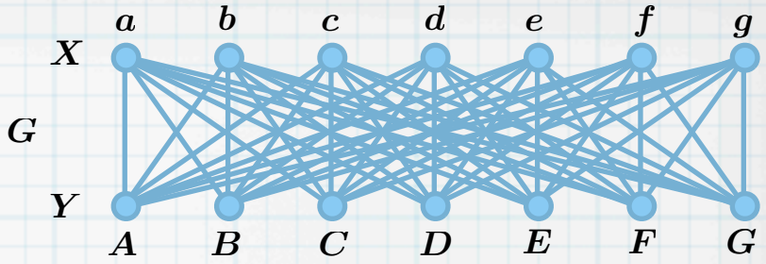
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

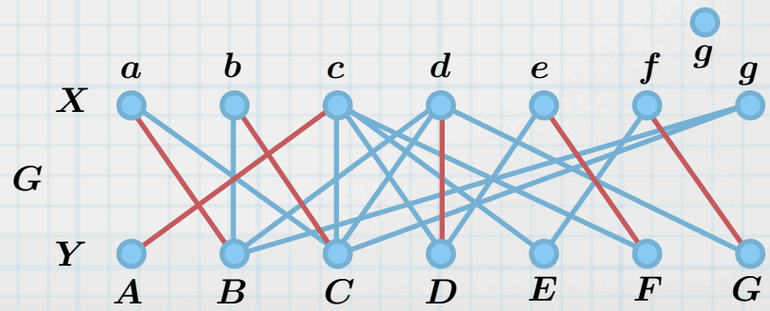
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$

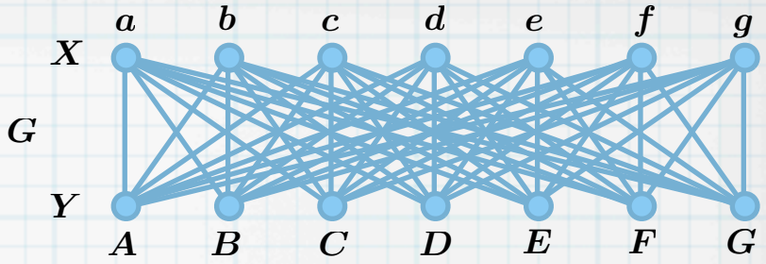
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

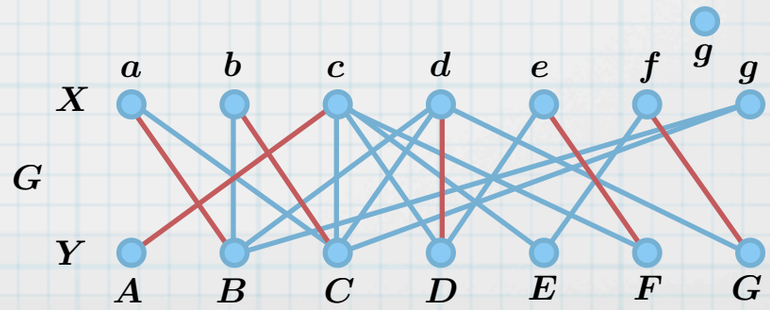
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$

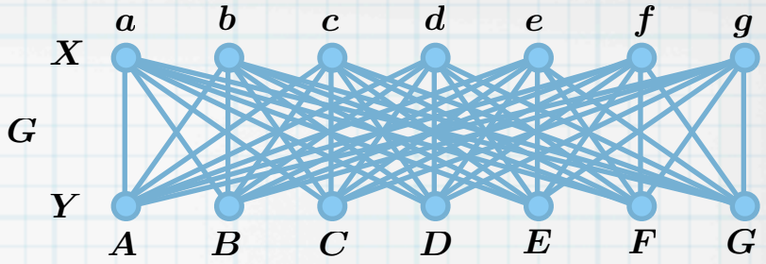
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

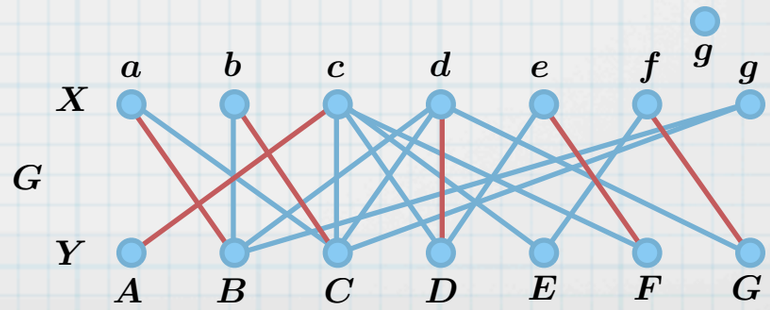
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$

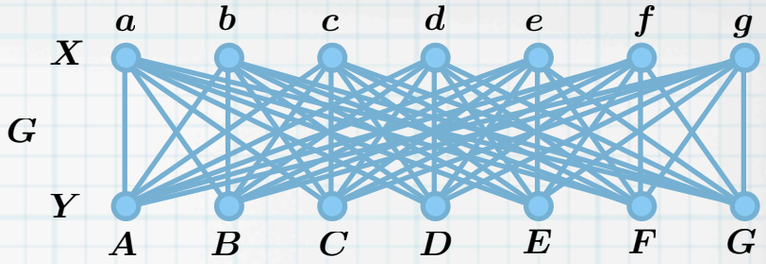
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

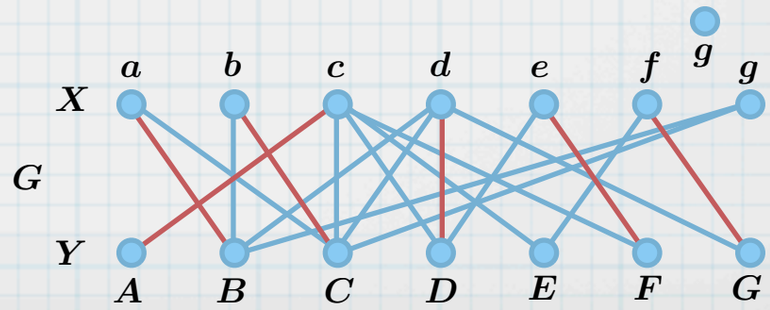
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$

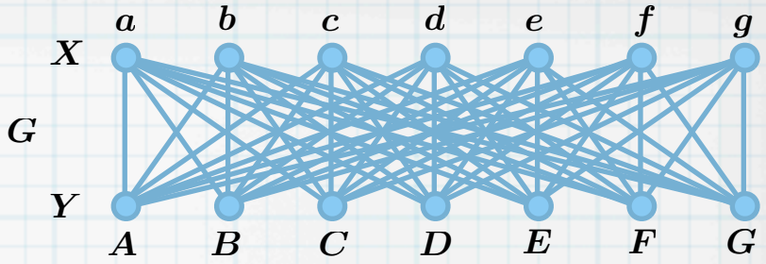
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

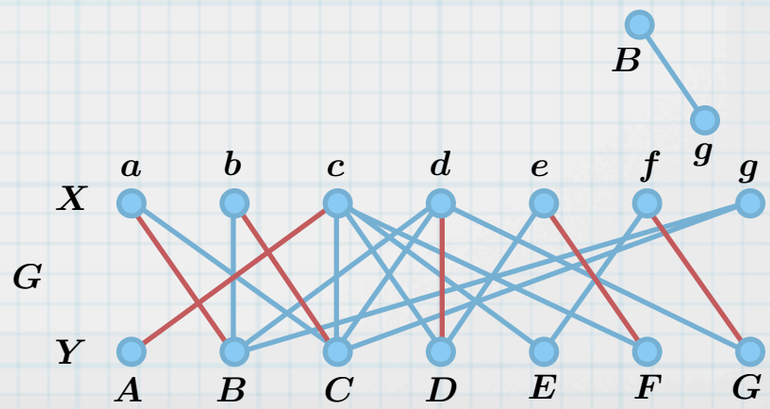
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$

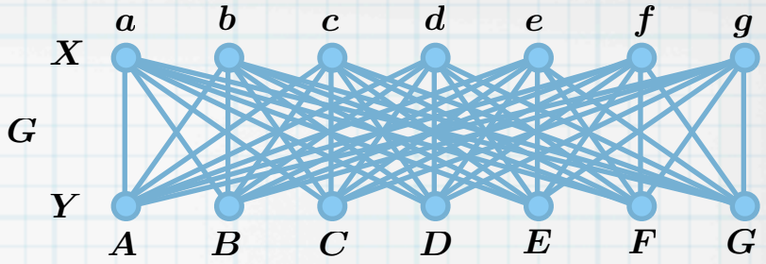
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

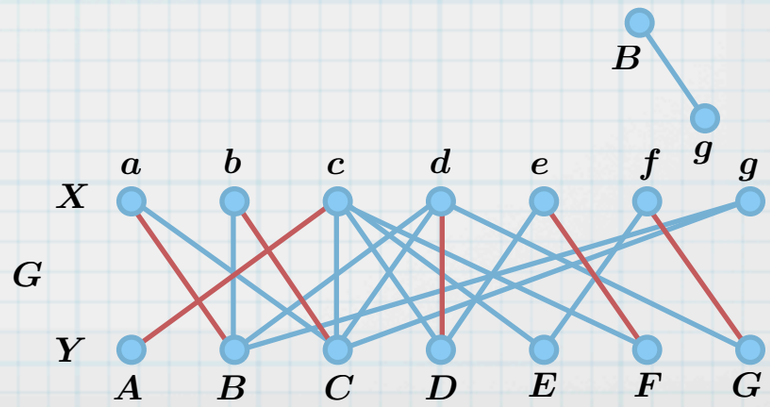
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**

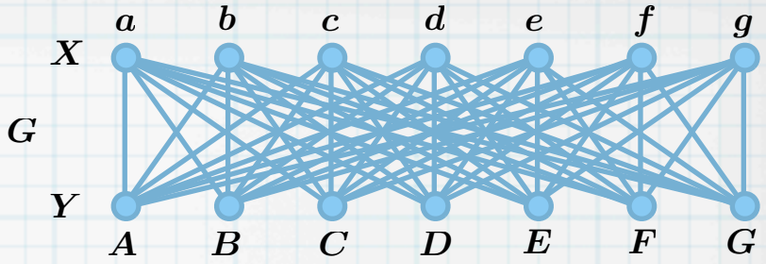
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

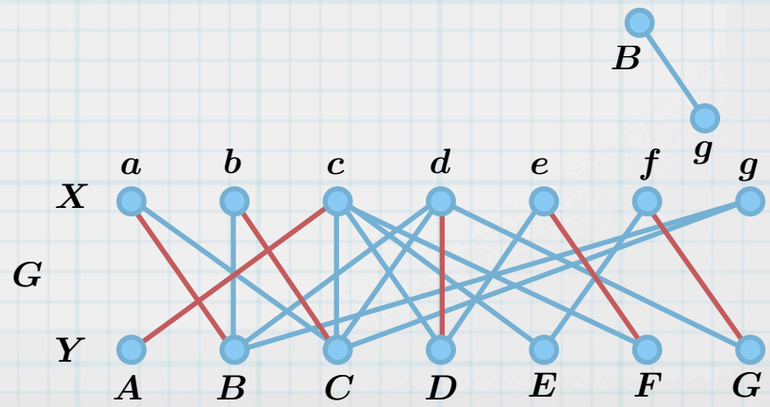
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**

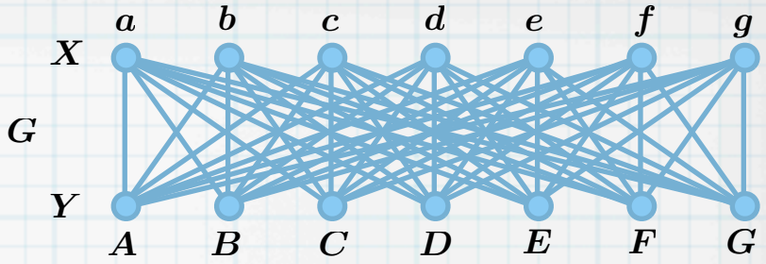
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

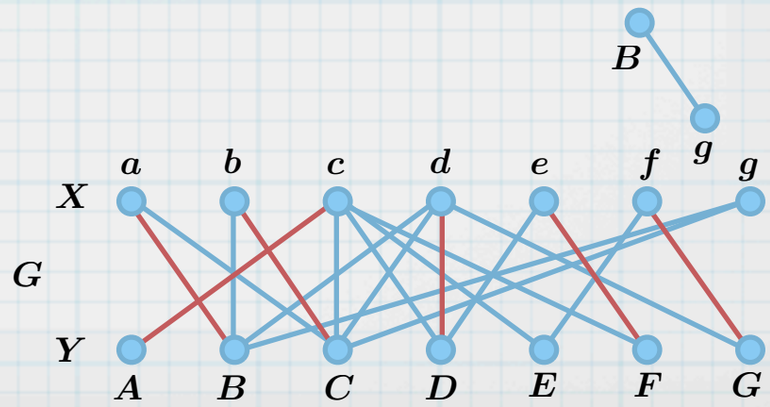
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**

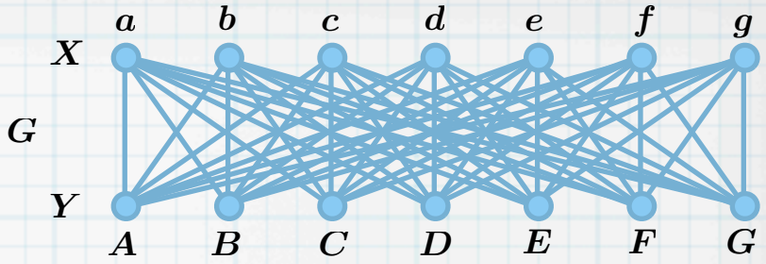
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

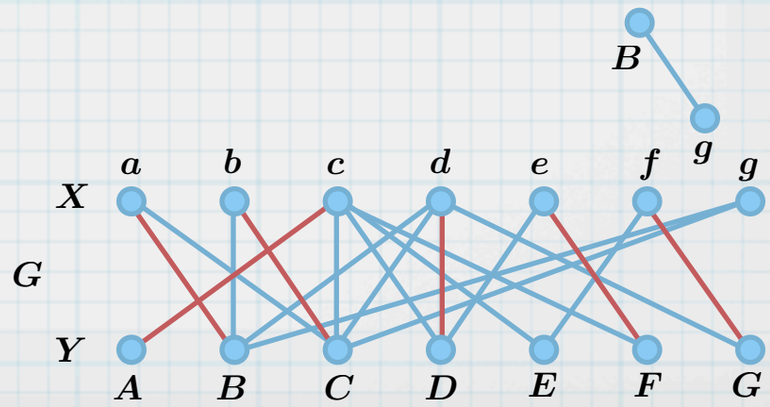
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$

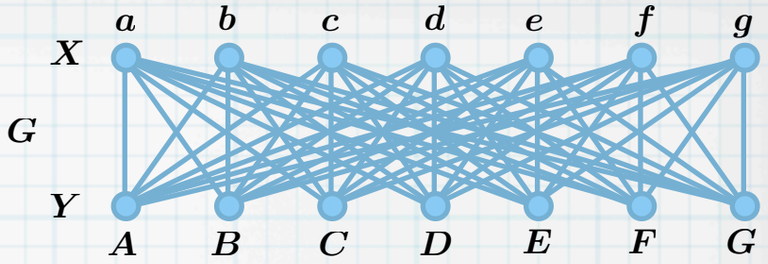
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

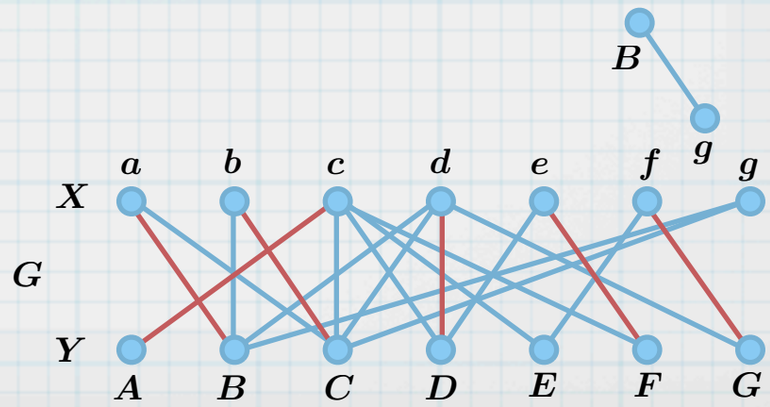
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$

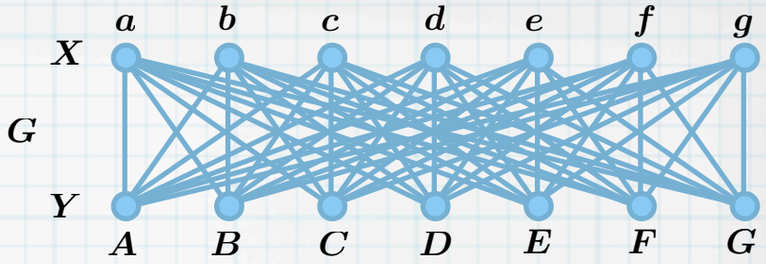
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

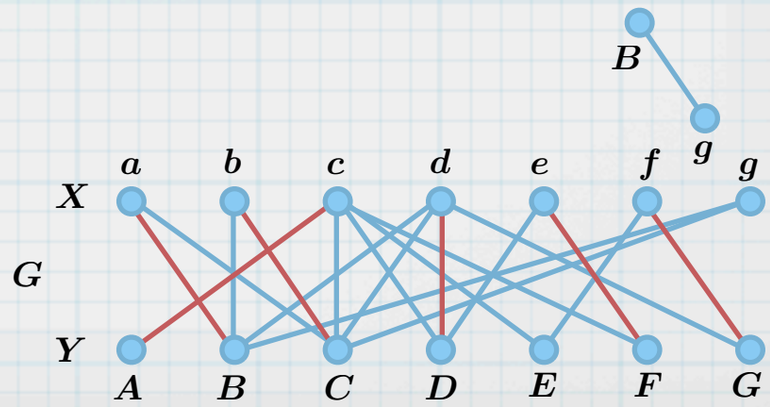
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$

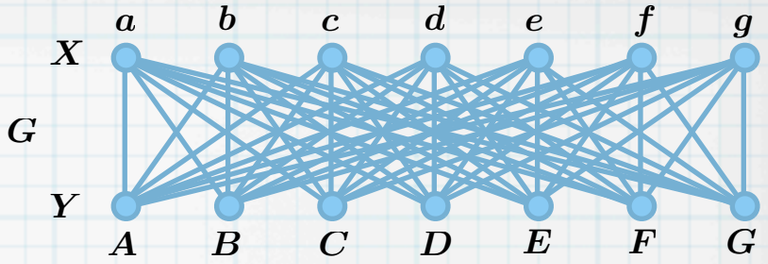
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

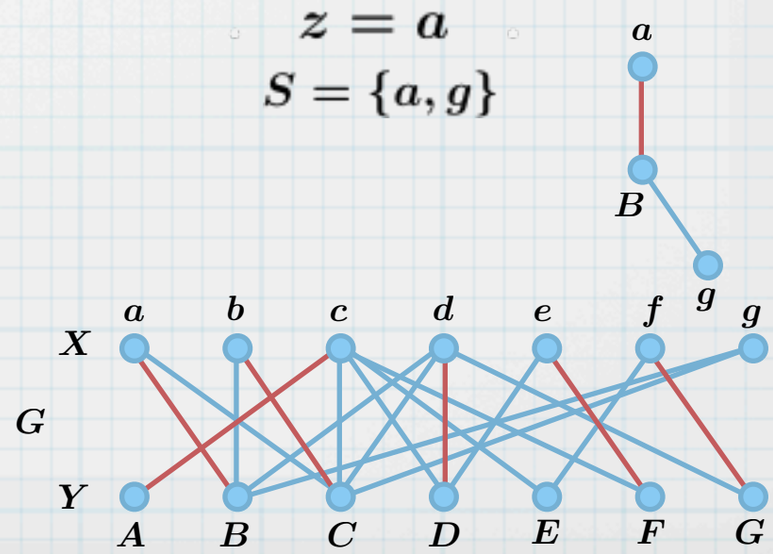
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

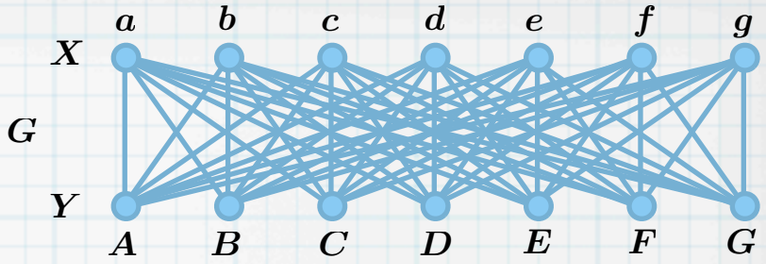
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

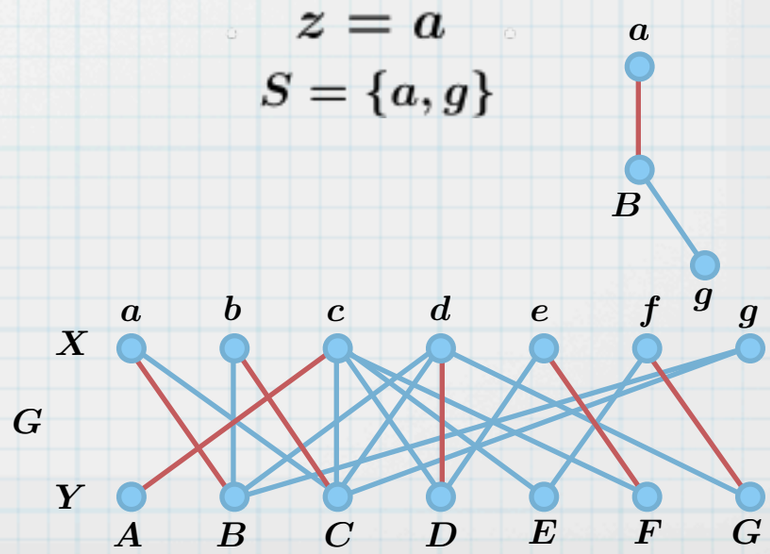
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**

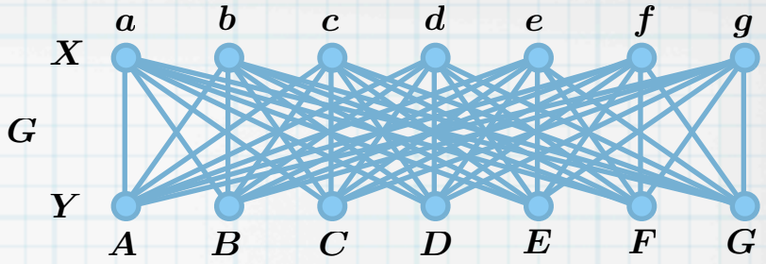
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

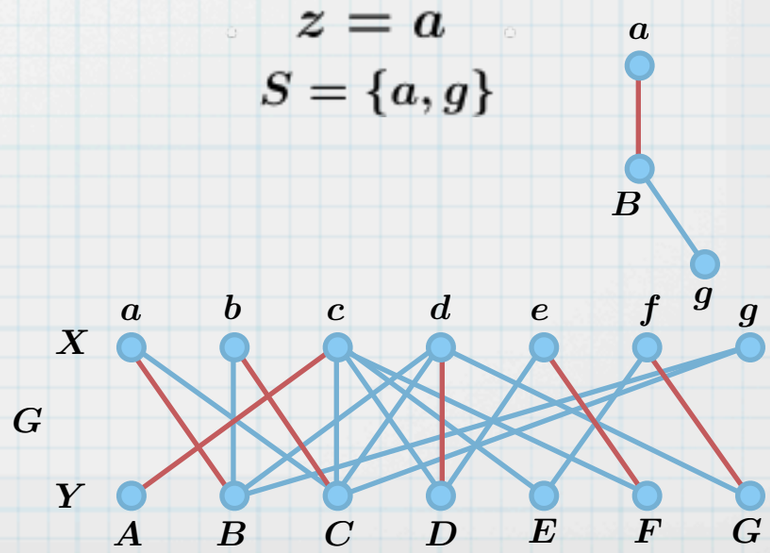
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**

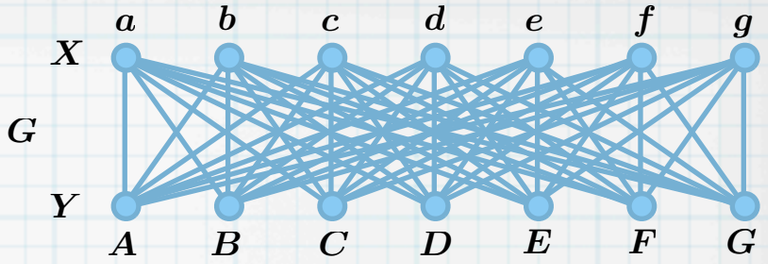
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

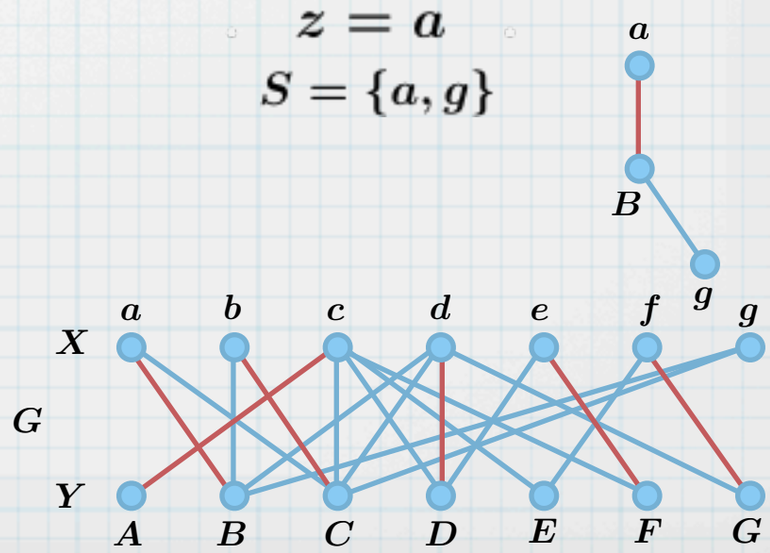
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

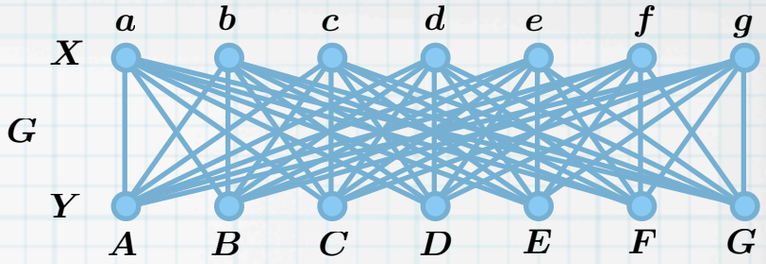
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

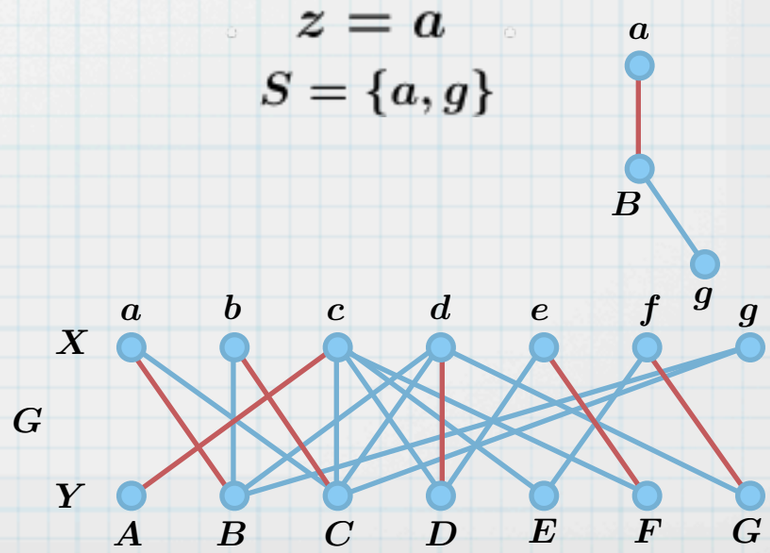
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

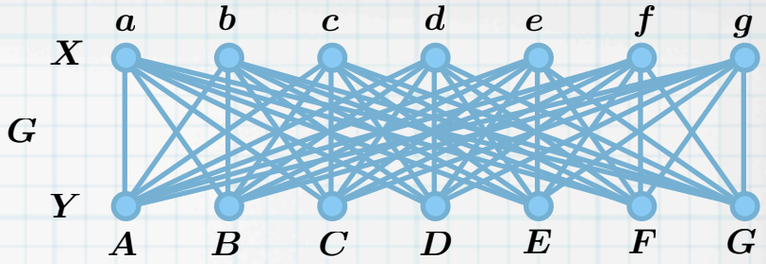
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

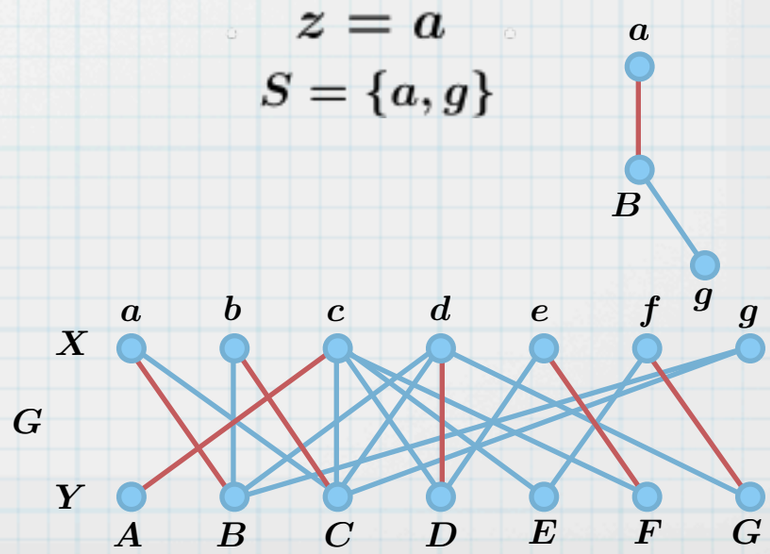
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

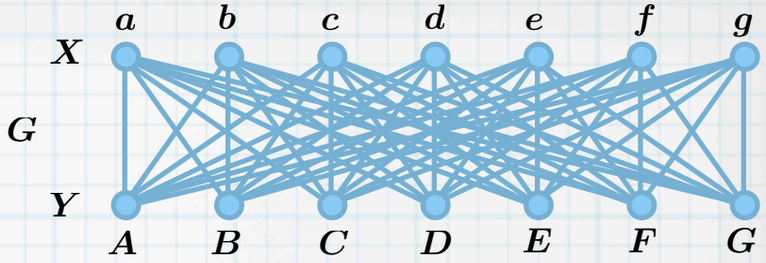
$x_0 = g$
$S = \{g\}, T = \{\}$

$\{B, C\} \neq \{\}$

$y = B$
$T = \{B\}$
$z = a$
$S = \{a, g\}$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

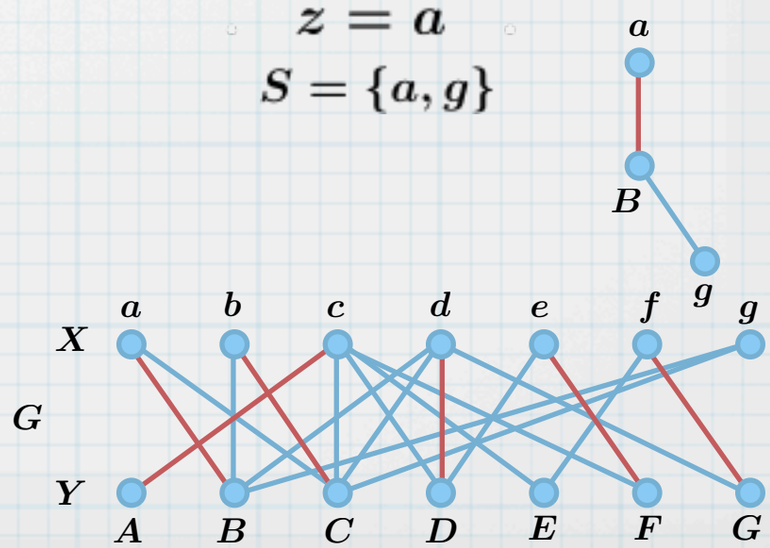
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

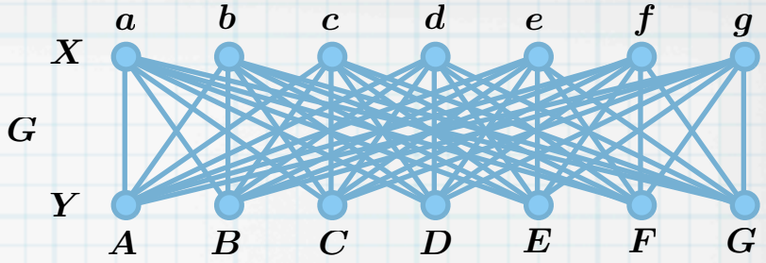
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

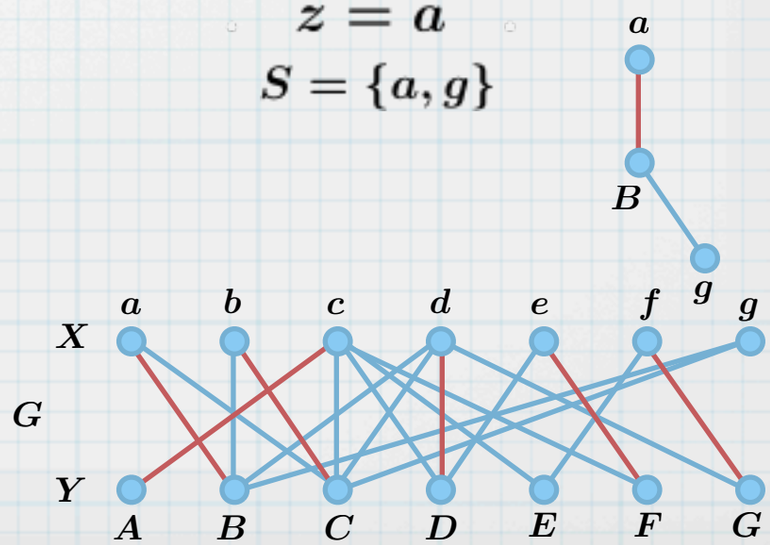
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

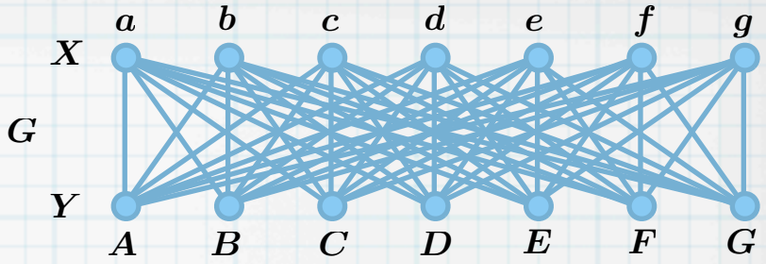
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

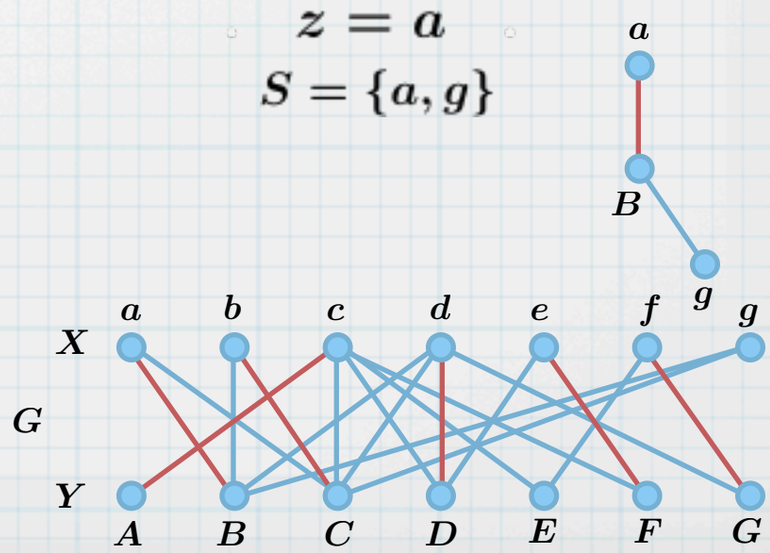
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

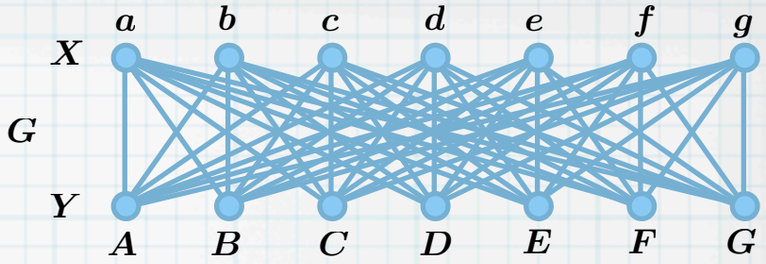
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

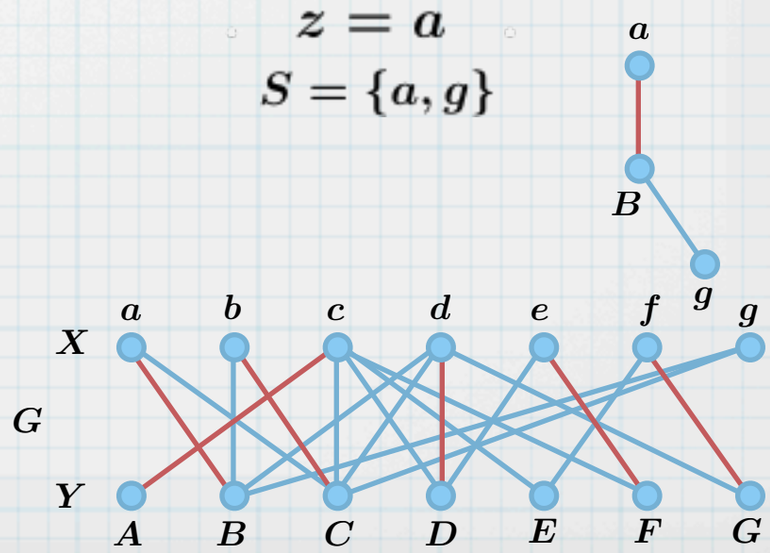
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

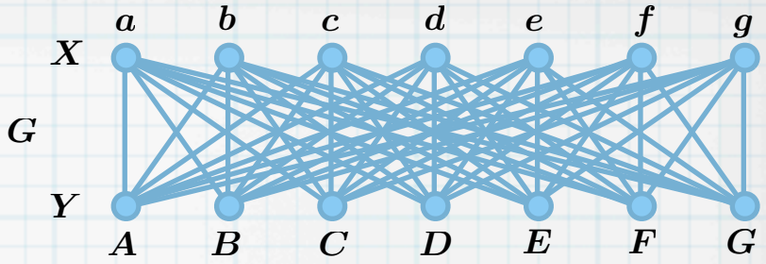
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$

$$y = B$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

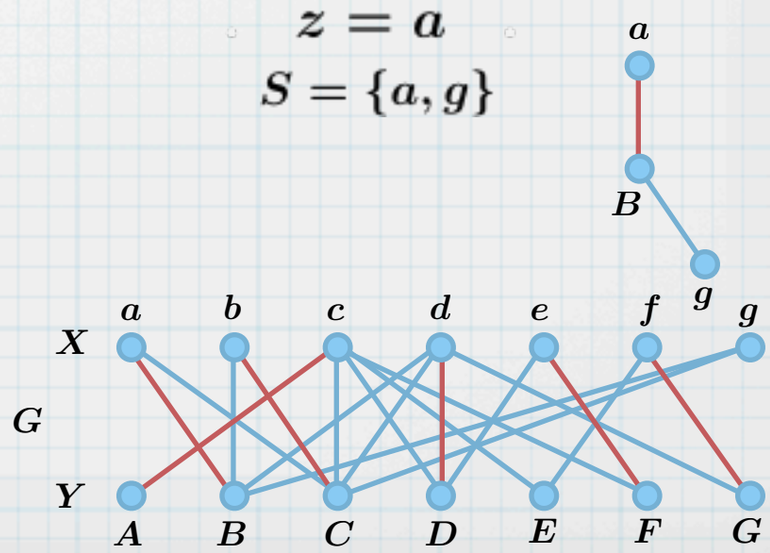
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

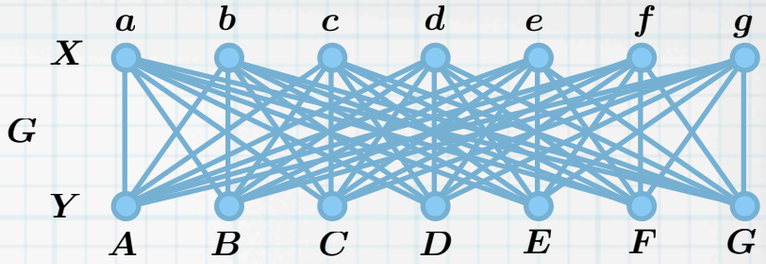
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

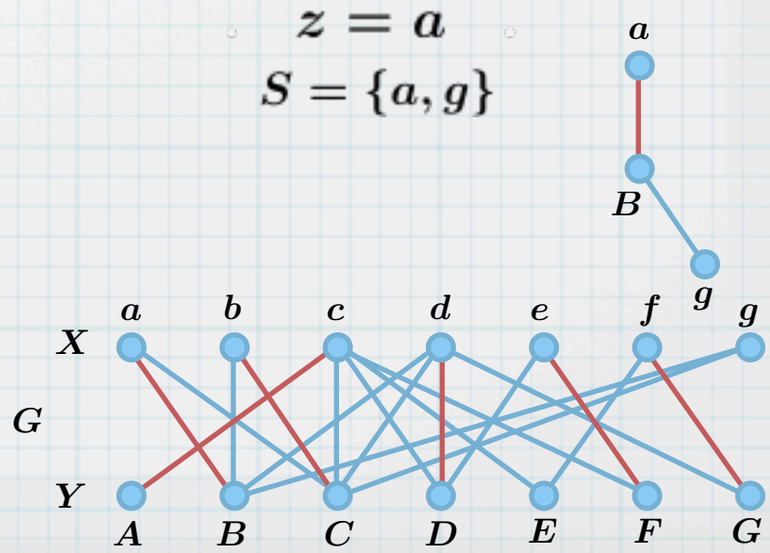
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

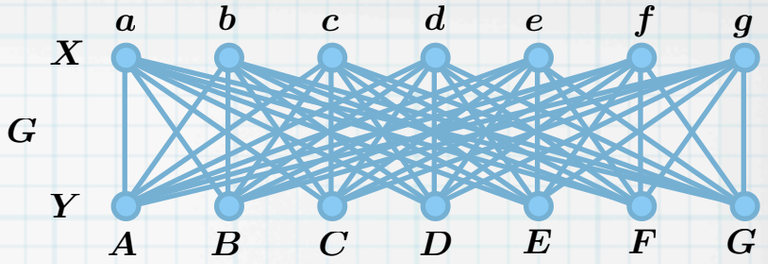
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

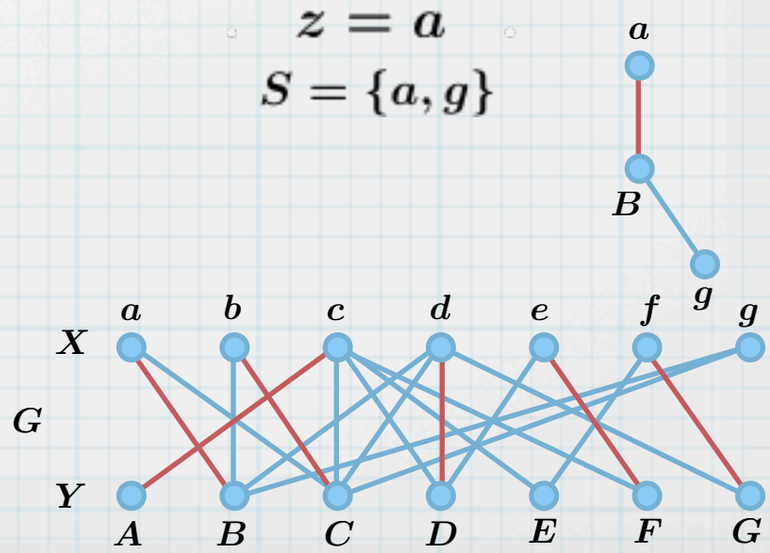
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

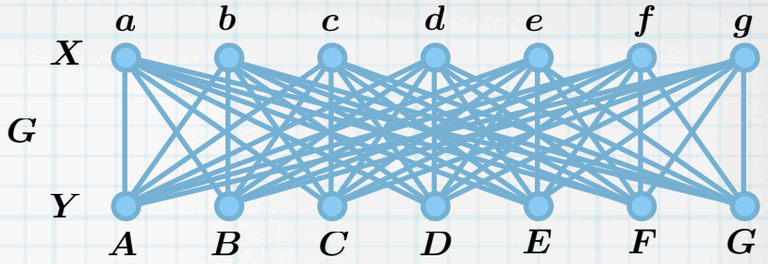
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = a$$
$$S = \{a, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

$$x_0 = g$$

$$S = \{g\}, T = \{\}$$

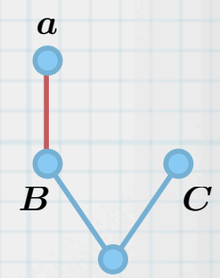
$$\{B, C\} \neq \{B\}$$

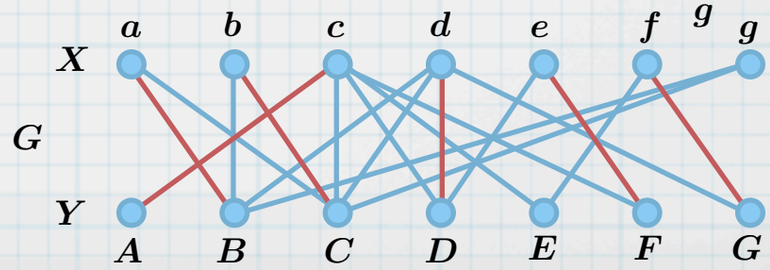
$$y = C$$

$$T = \{B, C\}$$

$$z = a$$

$$S = \{a, g\}$$

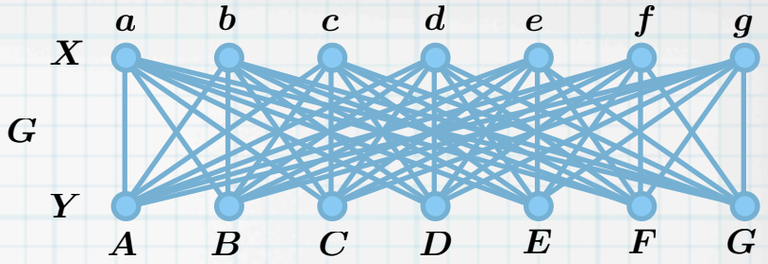




* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

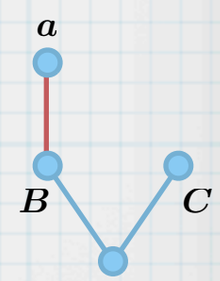
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

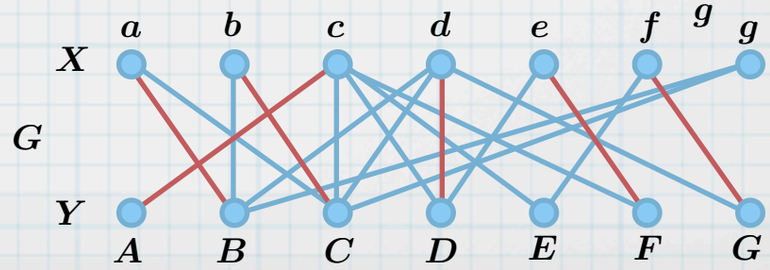
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

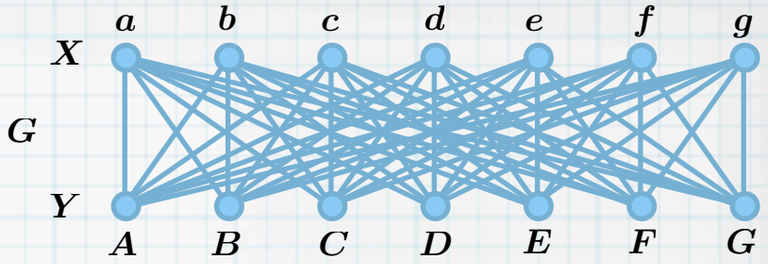
$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = a$$
$$S = \{a, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

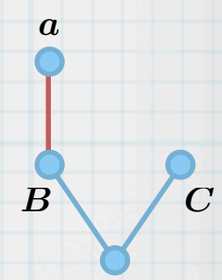
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

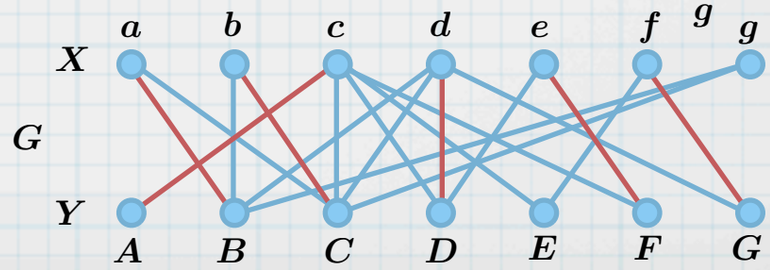
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

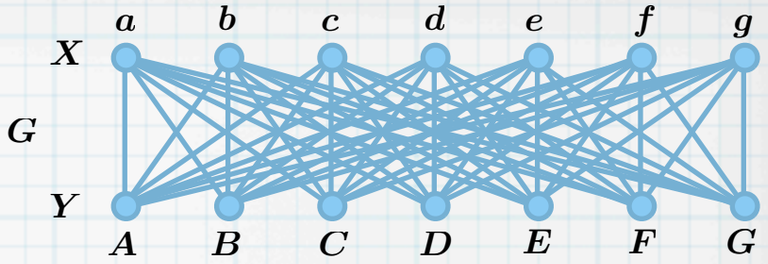
$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

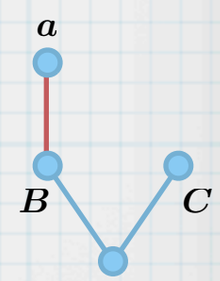
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

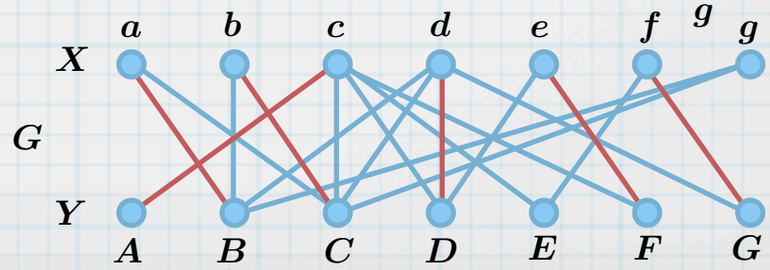
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

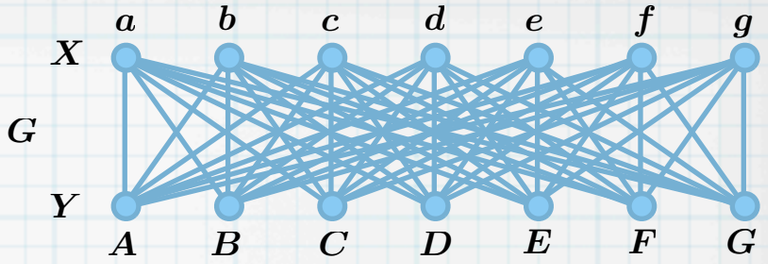
$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

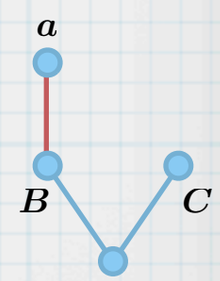
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

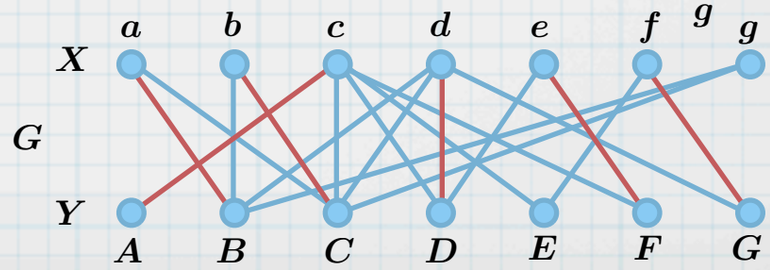
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

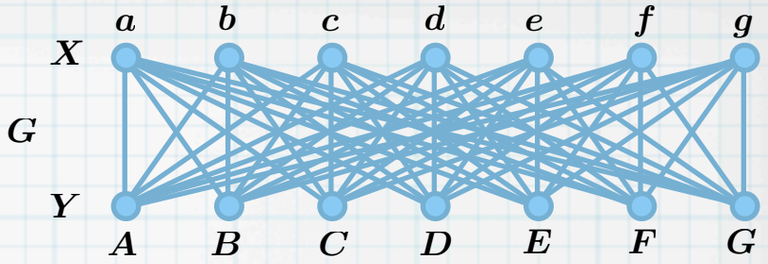
$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

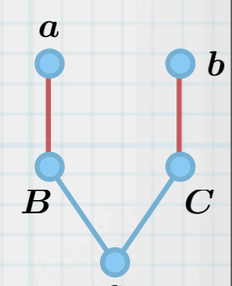
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

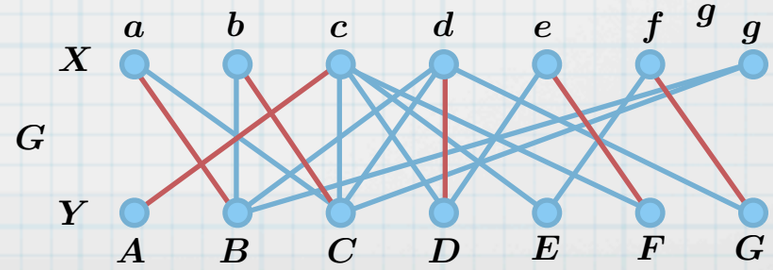
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

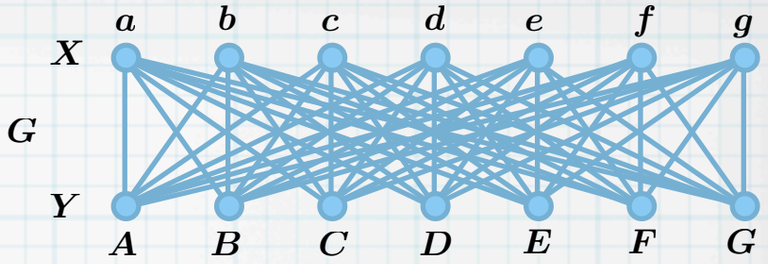
$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

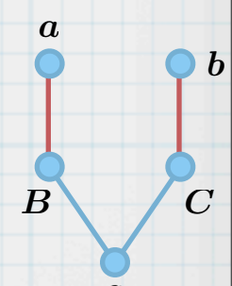
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

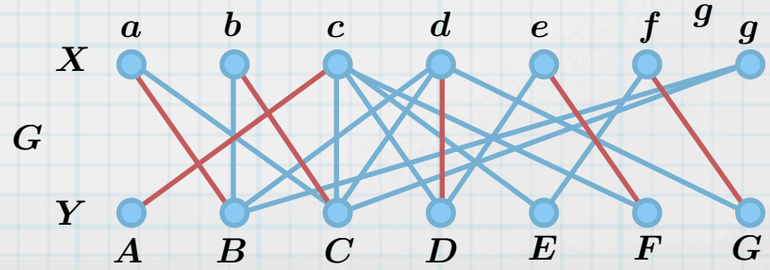
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

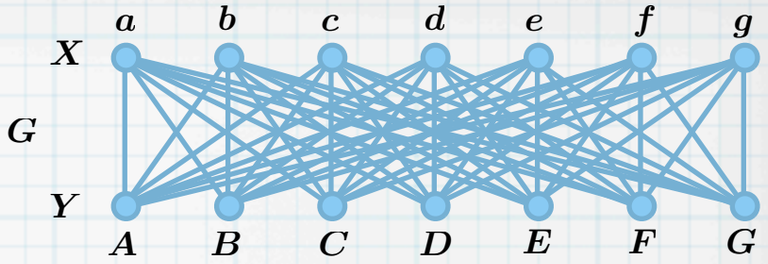
$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

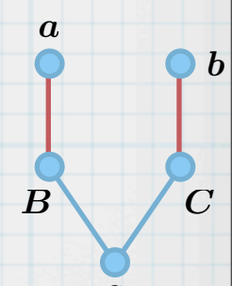
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

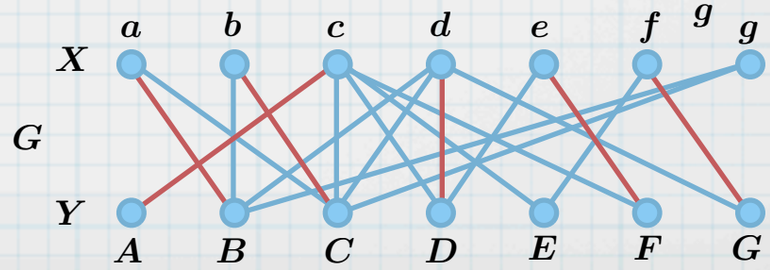
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

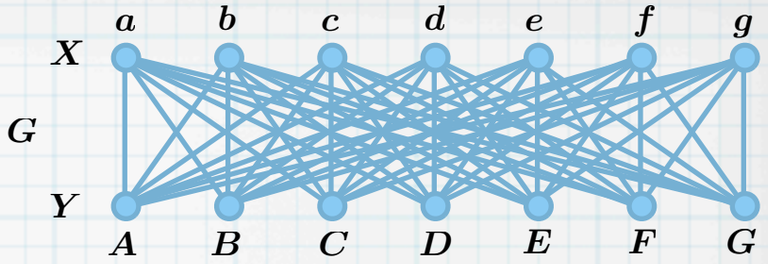
$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

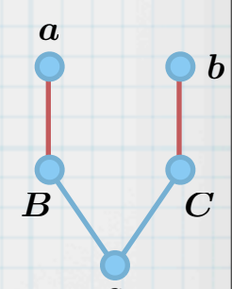
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

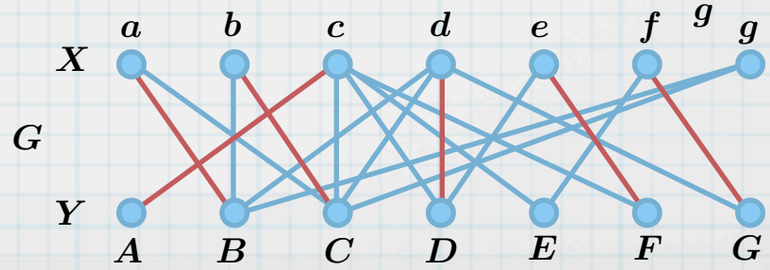
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

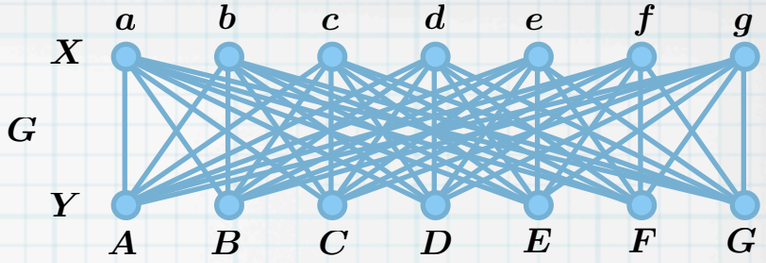
$$\{B, C\} \neq \{B\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

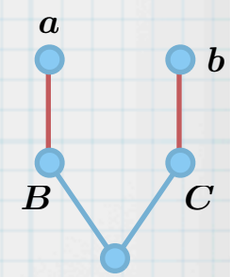
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

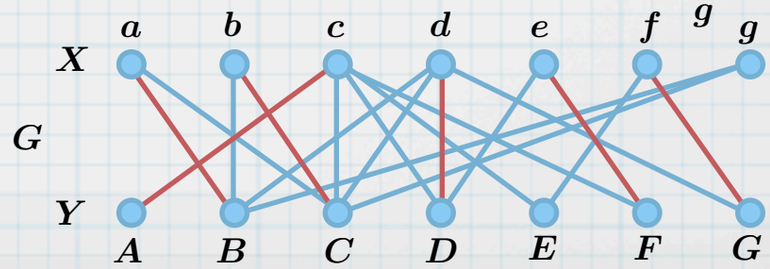
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C\} \neq \{B\}$$

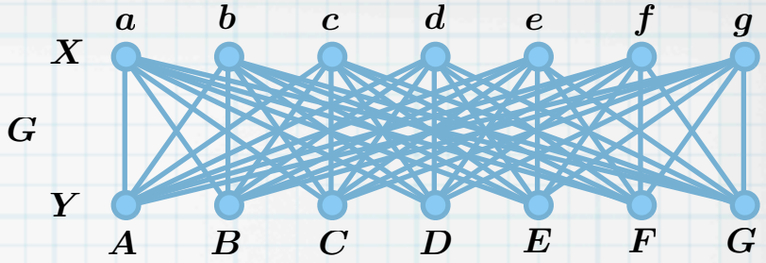
$$y = C$$
$$T = \{B, C\}$$

$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

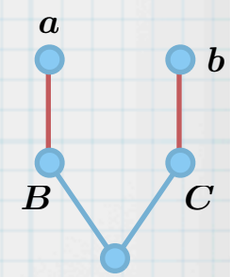
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

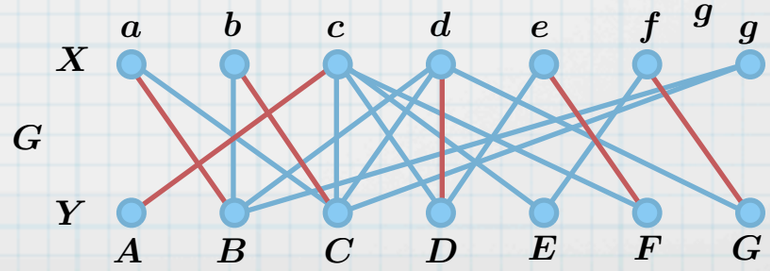
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

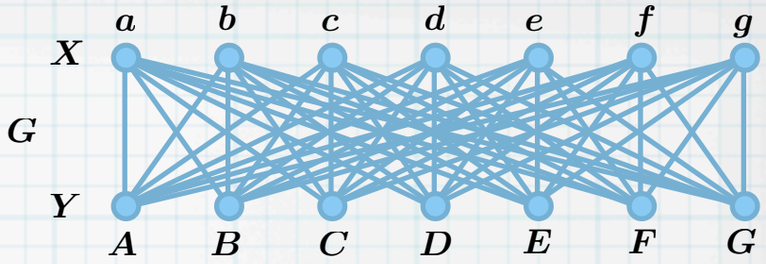
$$\{B, C\} = \{B, C\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

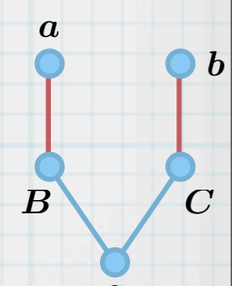
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

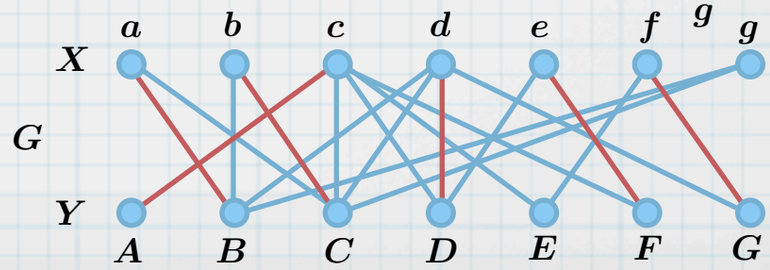
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

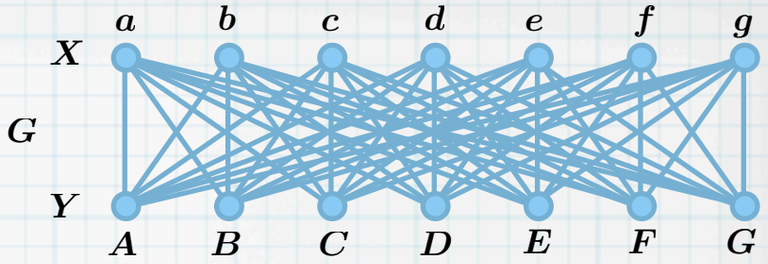
$$\{B, C\} = \{B, C\}$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

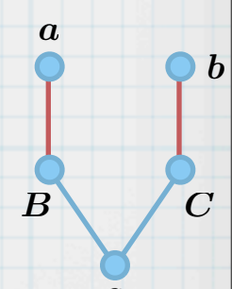
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

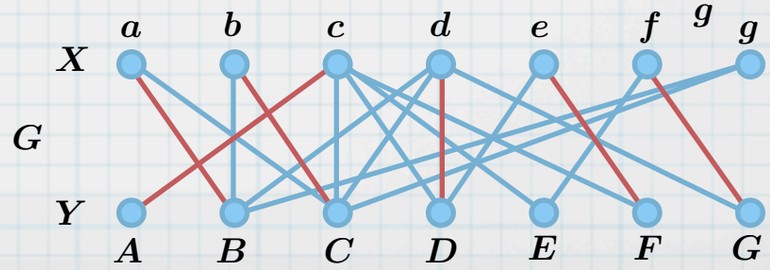
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

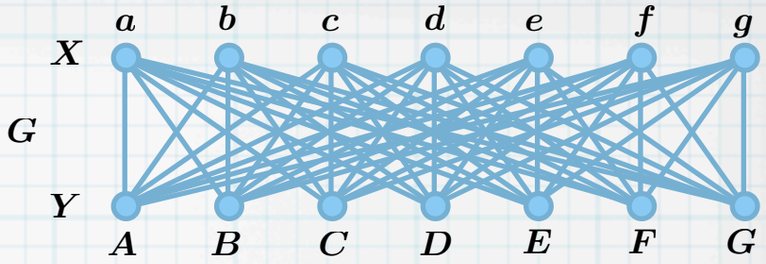
$$\{B, C\} = \{B, C\}$$
$$d_\lambda = 1$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 2 \\ 4 \\ 3 \\ 8 \\ 4 \end{matrix}$$

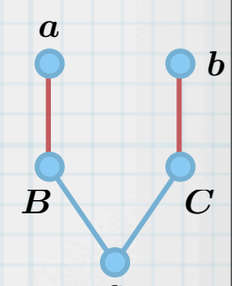
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

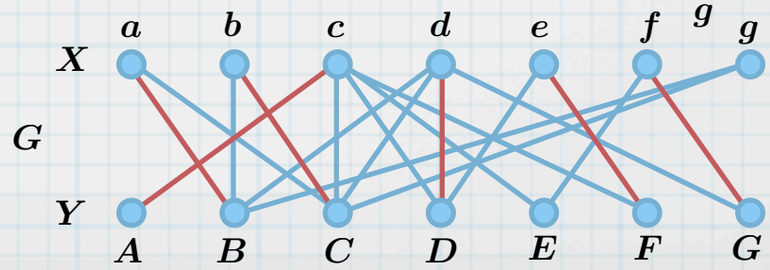
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

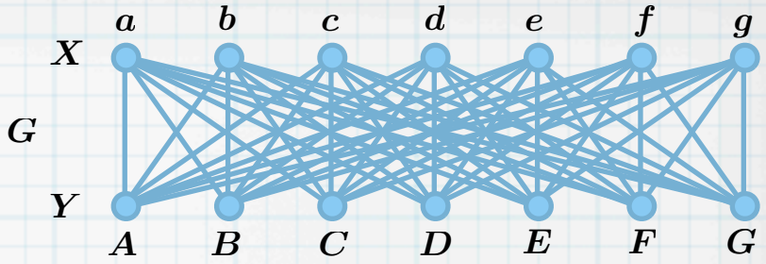
$$\{B, C\} = \{B, C\}$$
$$d_\lambda = 1$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

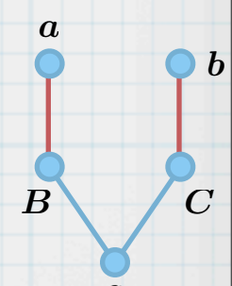
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

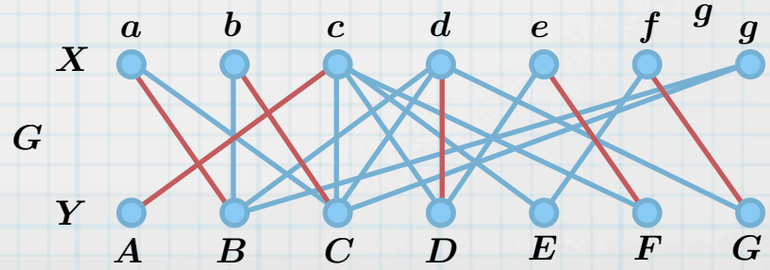
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

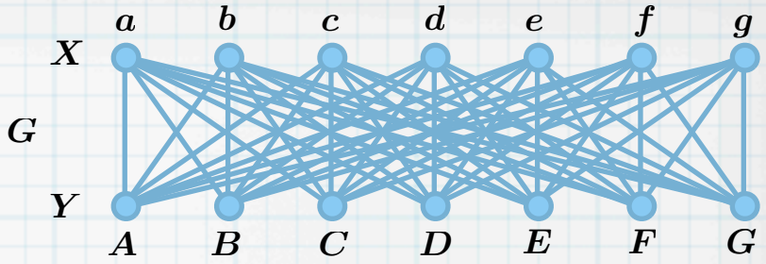
$$\{B, C\} = \{B, C\}$$
$$d_\lambda = 1$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

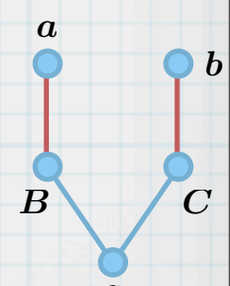
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

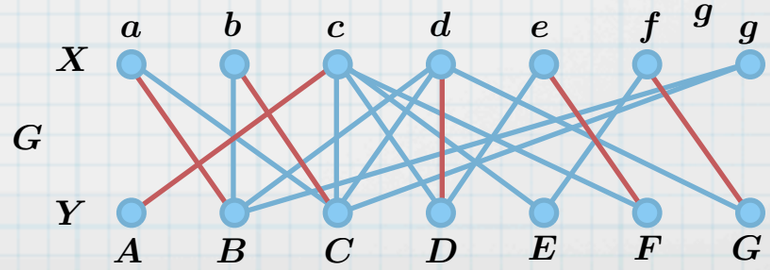
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \ \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \ \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

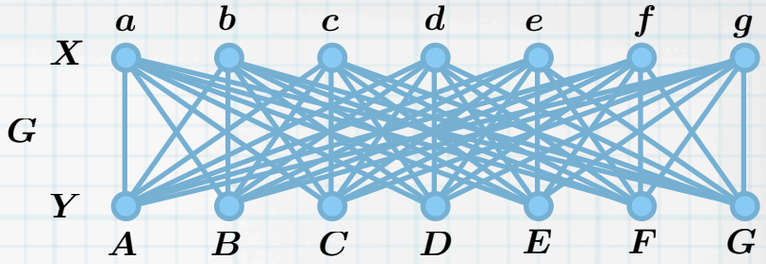
$$\{B, C\} = \{B, C\}$$
$$d_\lambda = 1$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

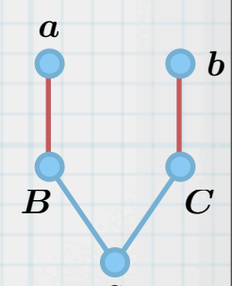
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

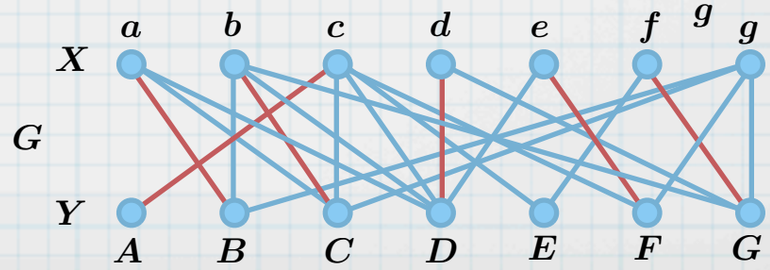
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

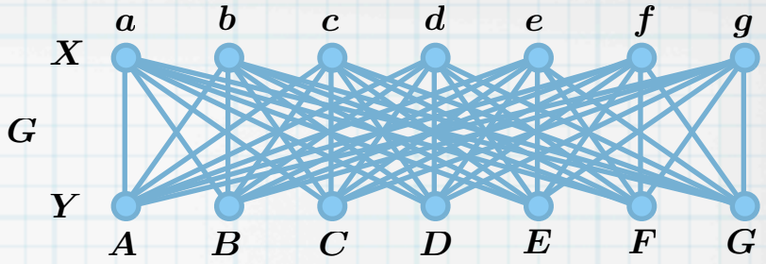
$$\{B, C\} = \{B, C\}$$
$$d_\lambda = 1$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

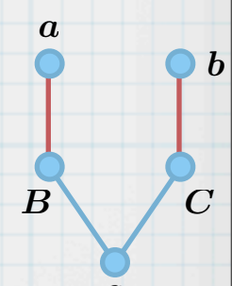
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

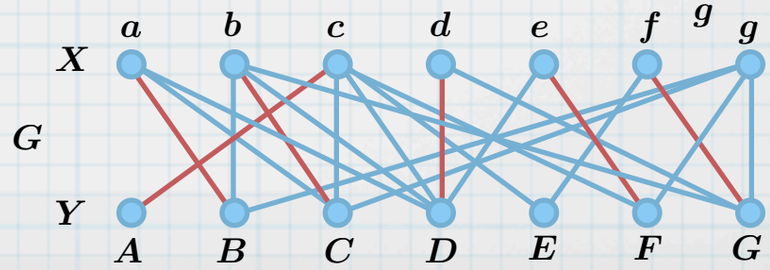
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \ \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \ \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

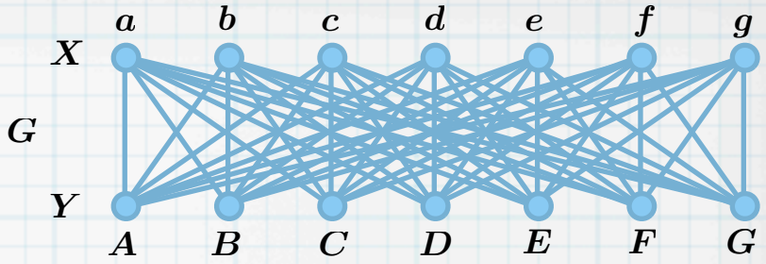
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

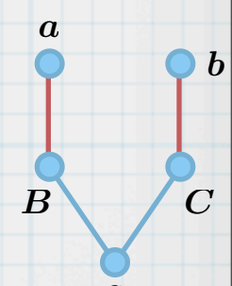
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \ \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \ \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

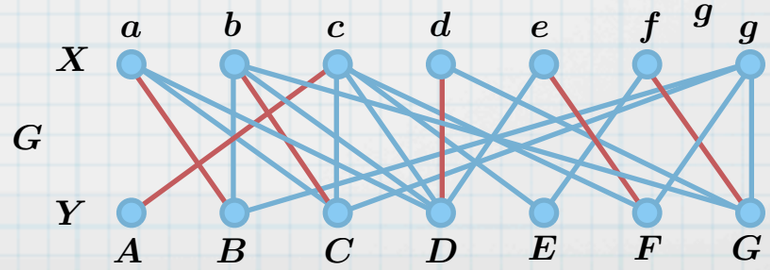
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

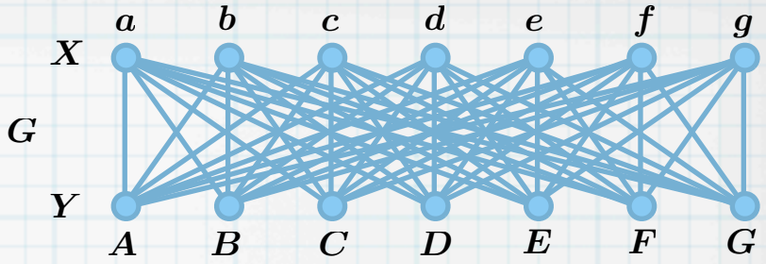
$$y = C$$
$$T = \{B, C\}$$

$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

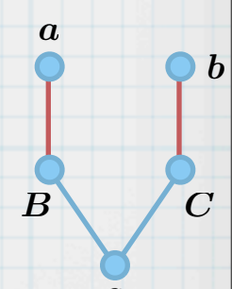
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

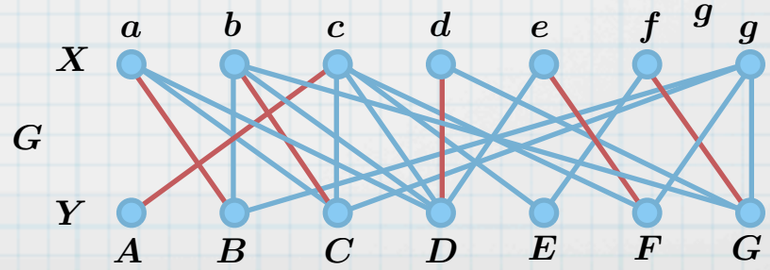
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

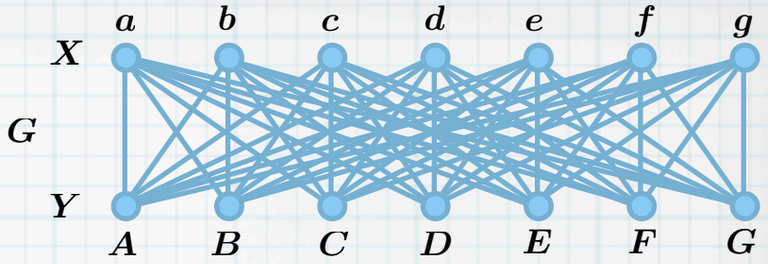
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = C$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

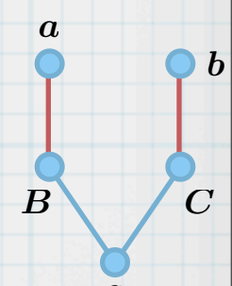
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

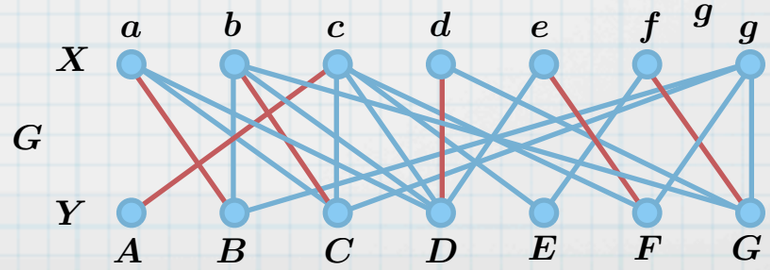
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \ \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \ \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

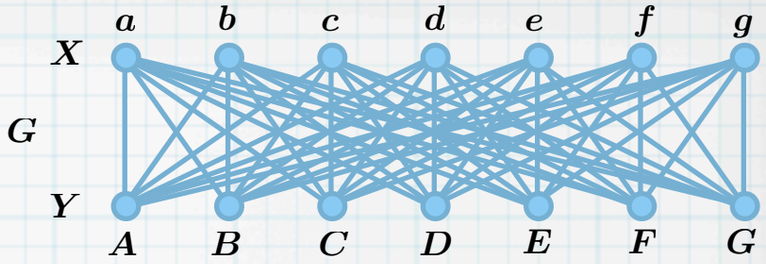
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

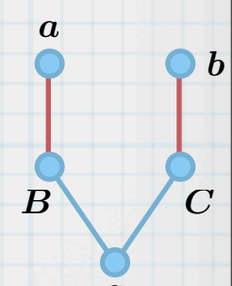
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

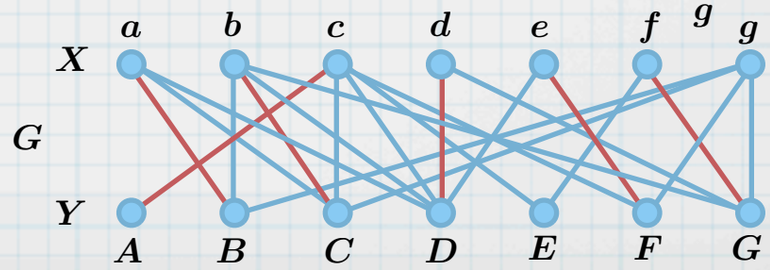
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

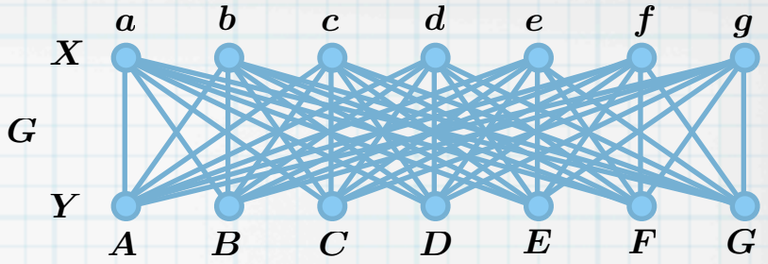
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

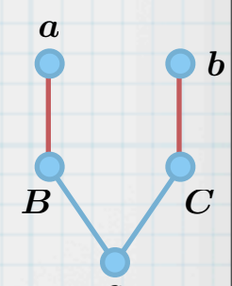
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \ \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \ \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

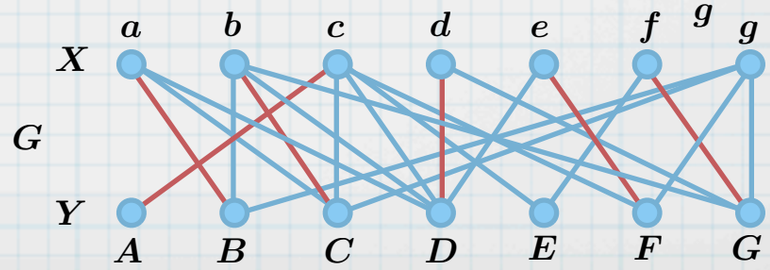
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

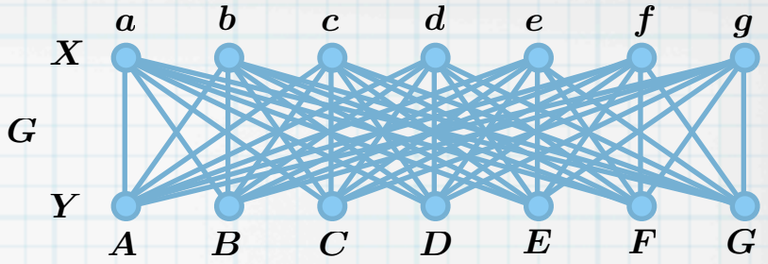
$$y = D$$
$$T = \{B, C, D\}$$

$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

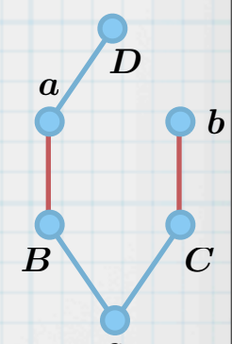
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

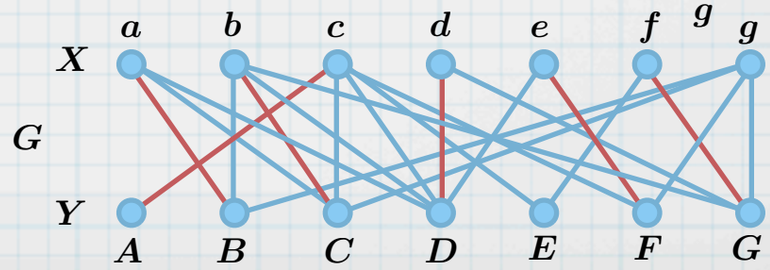
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \ \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \ \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

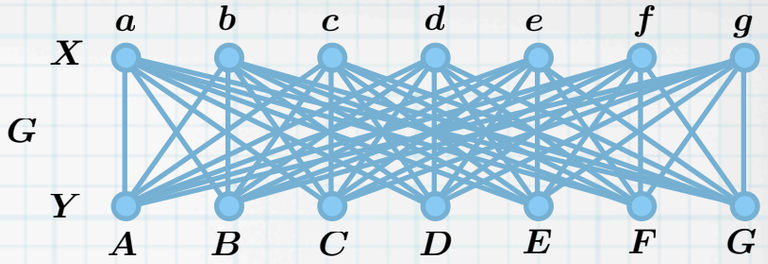
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

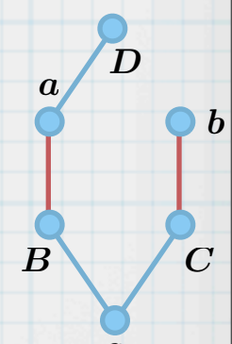
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

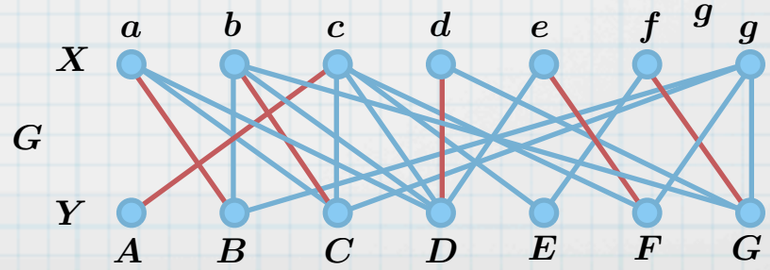
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

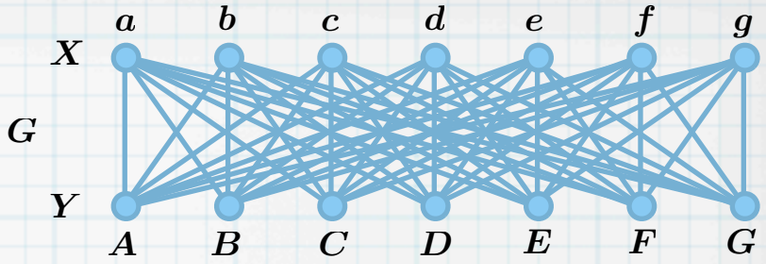
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = b$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

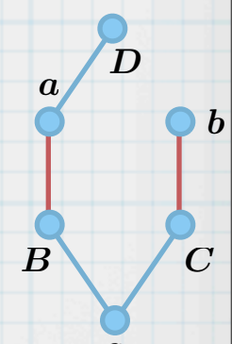
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

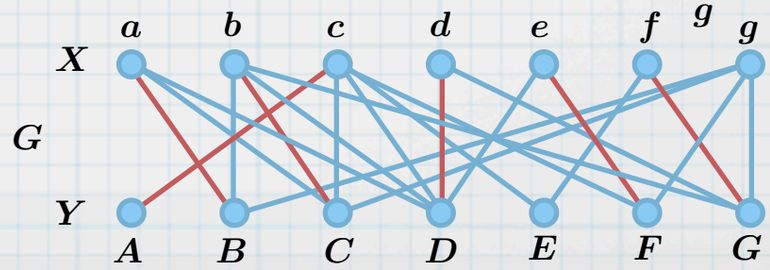
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

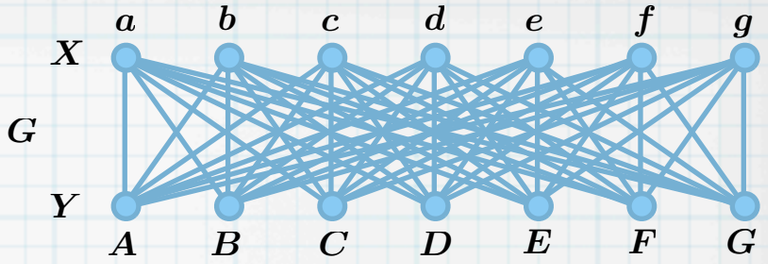
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

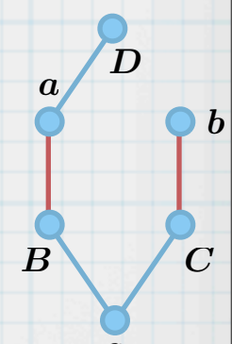
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

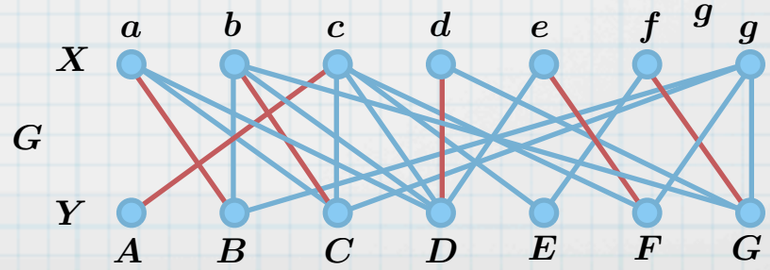
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \ \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \ \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

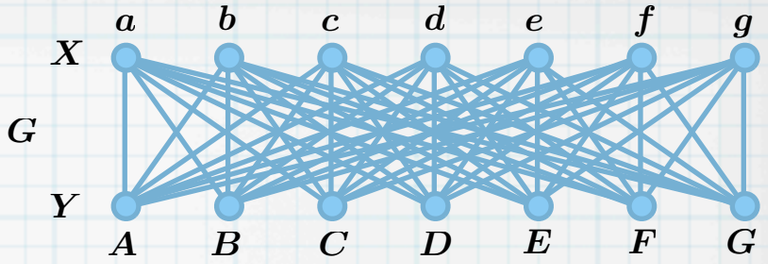
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

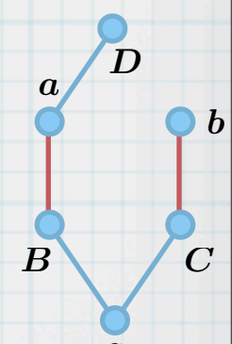
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

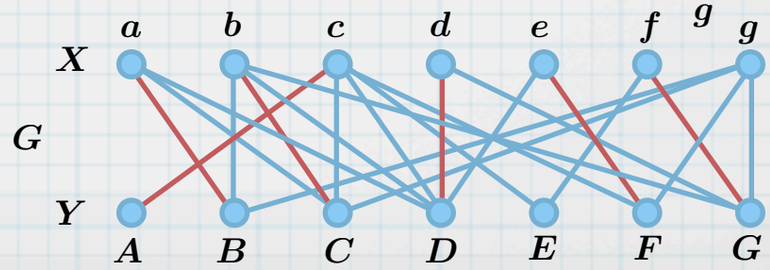
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

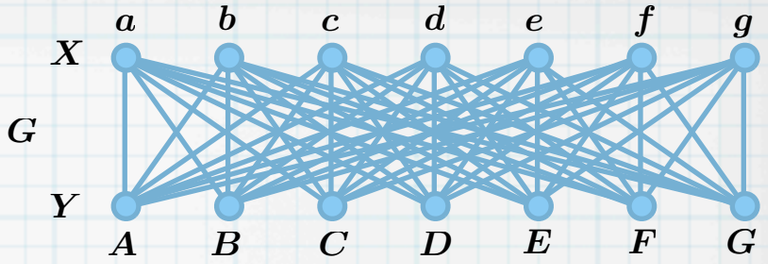
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

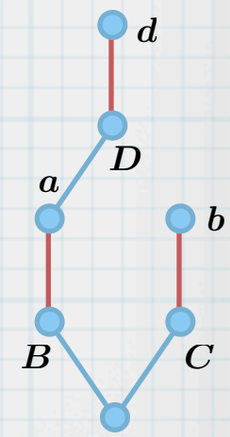
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

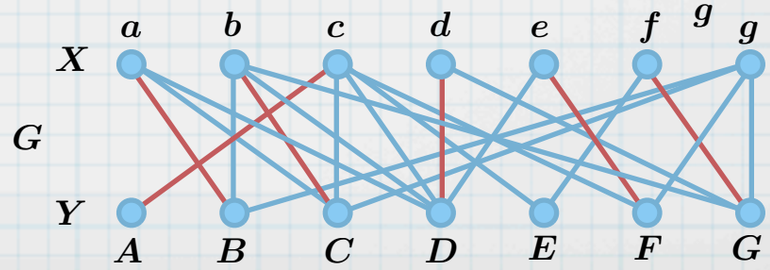
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \ \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \ \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

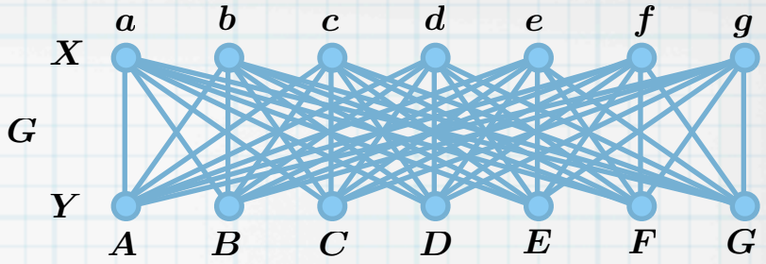
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

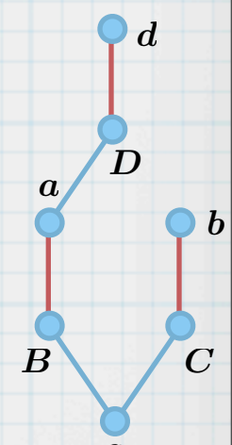
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

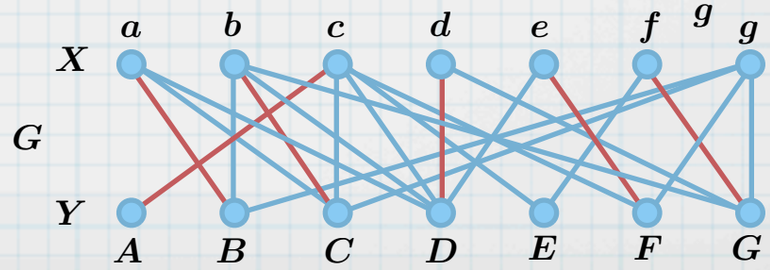
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

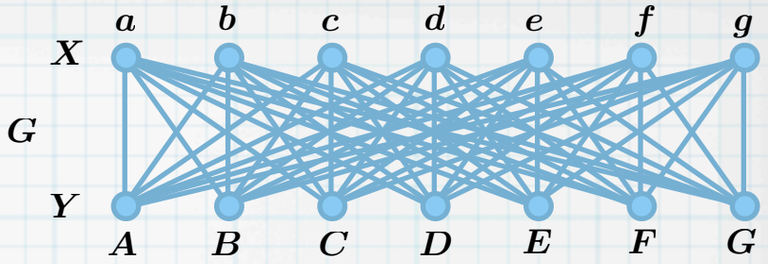
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

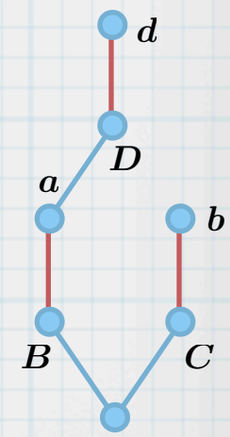
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

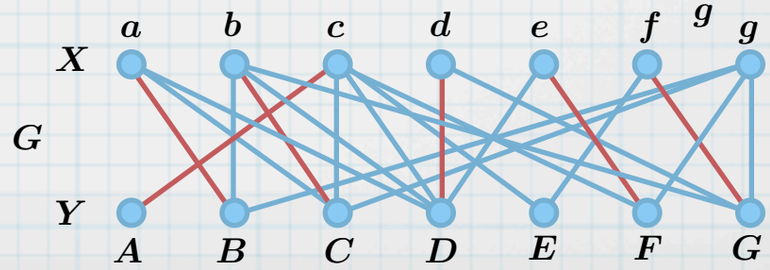
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \ \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \ \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

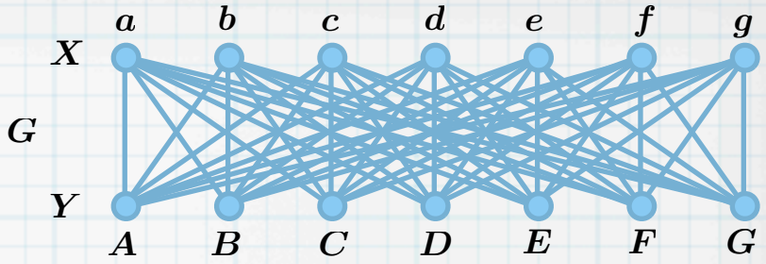
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

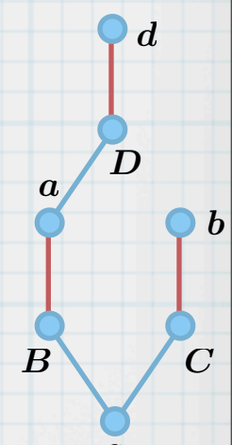
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

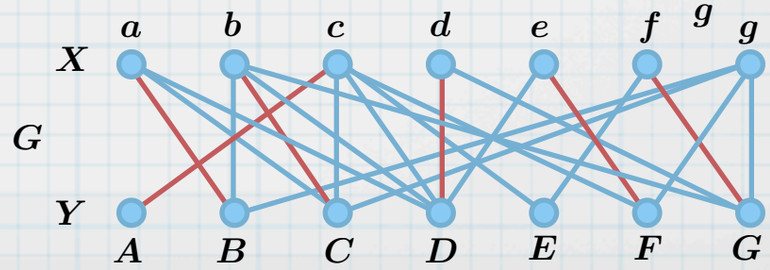
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

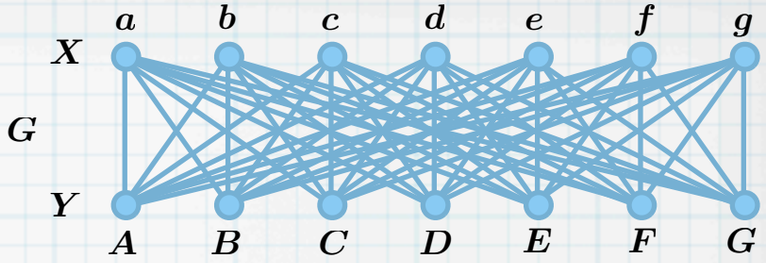
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

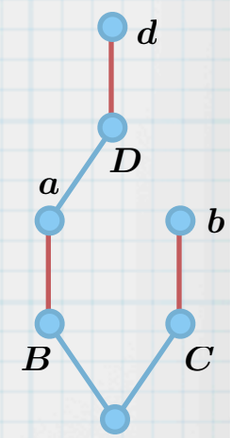
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

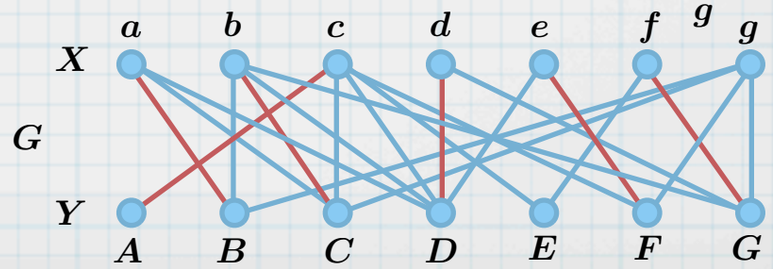
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

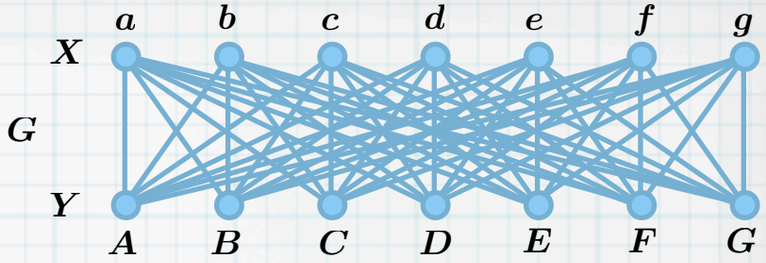
$$\{B, C, D, F, G\} \neq \{B, C\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

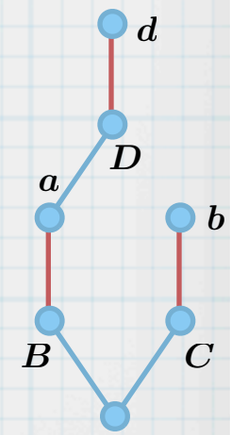
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

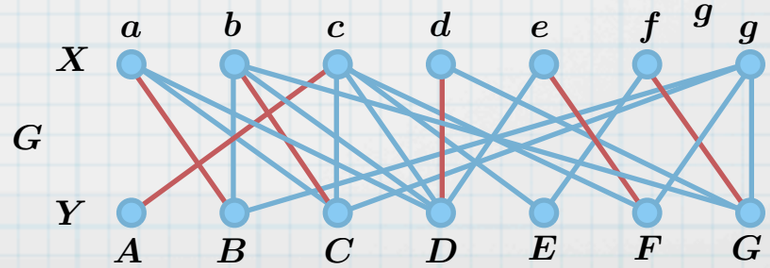
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

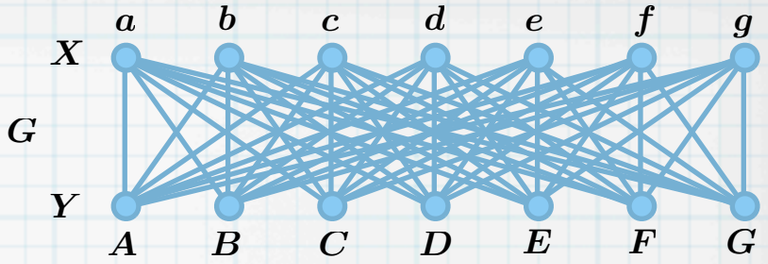
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

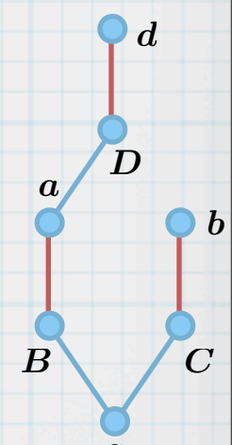
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

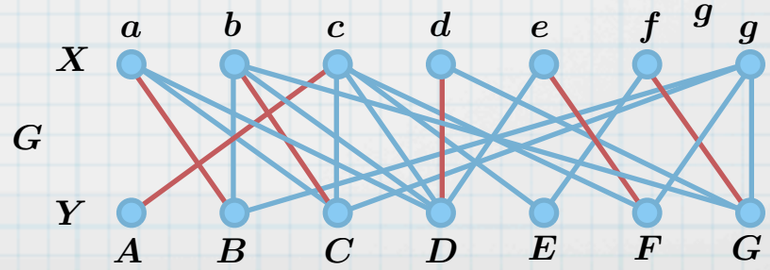
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

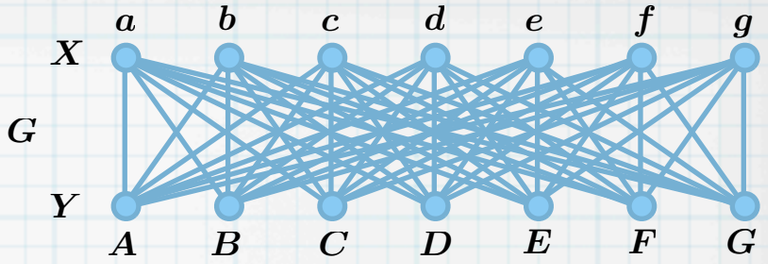
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

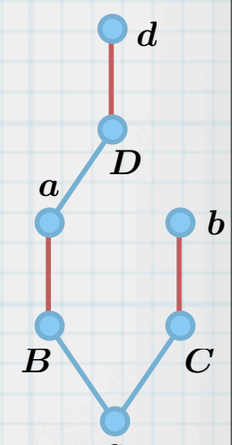
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

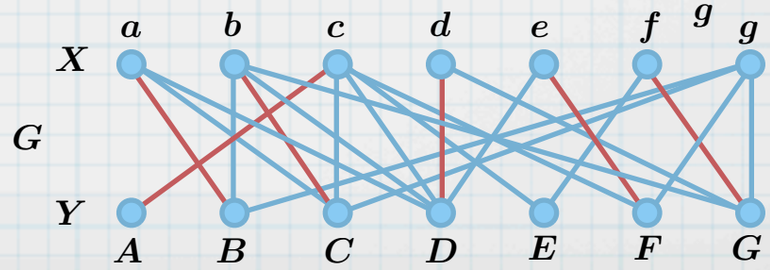
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

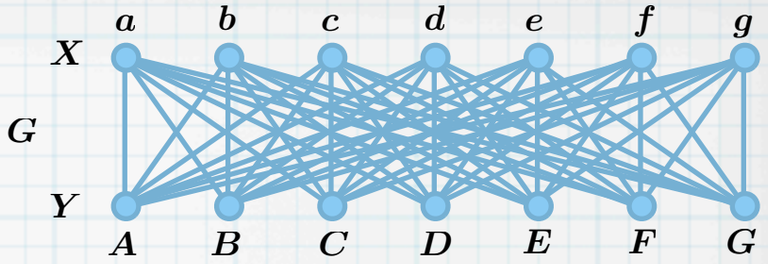
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = D$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

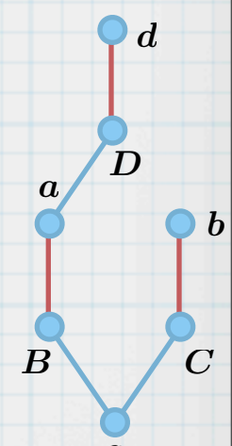
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

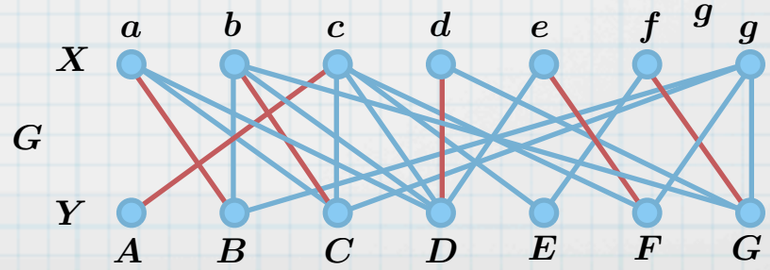
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

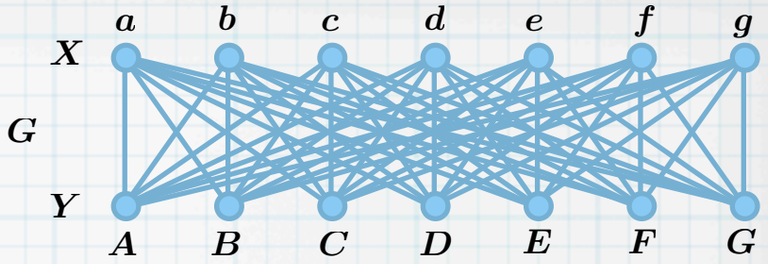
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

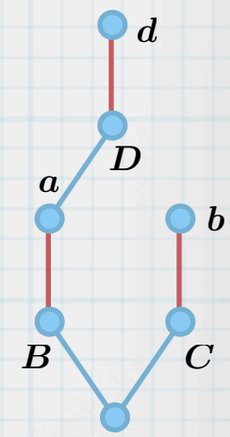
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

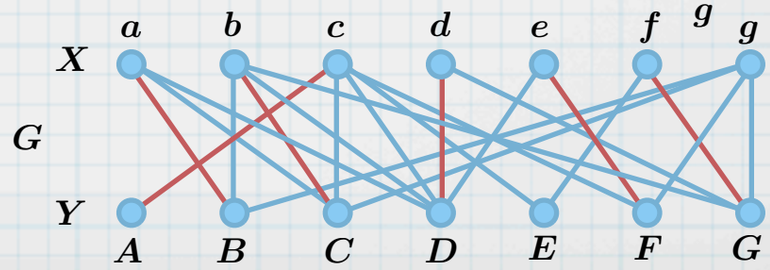
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

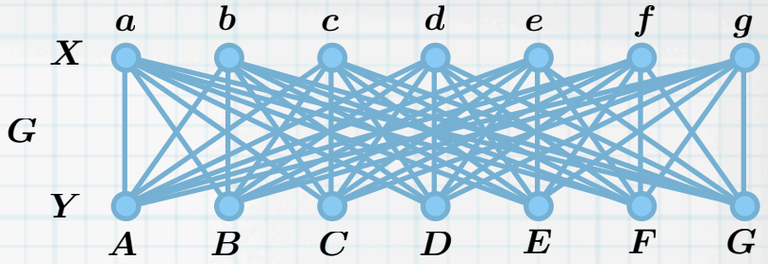
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

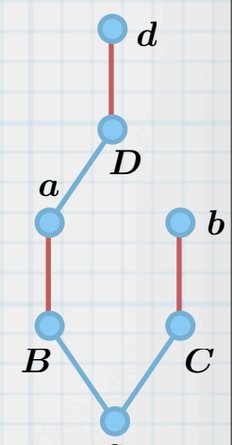
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

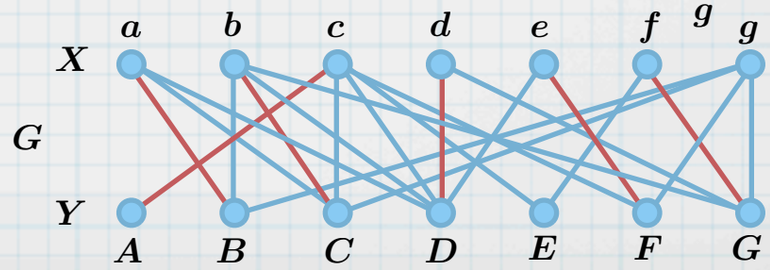
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

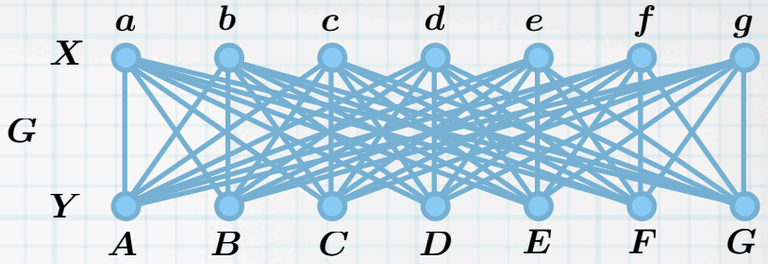
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

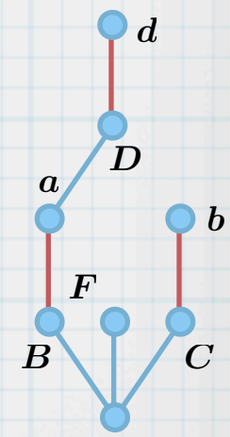
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

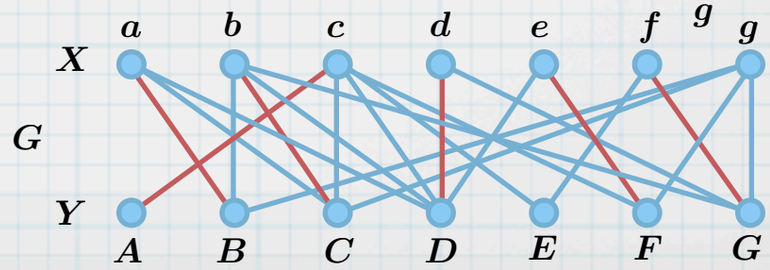
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

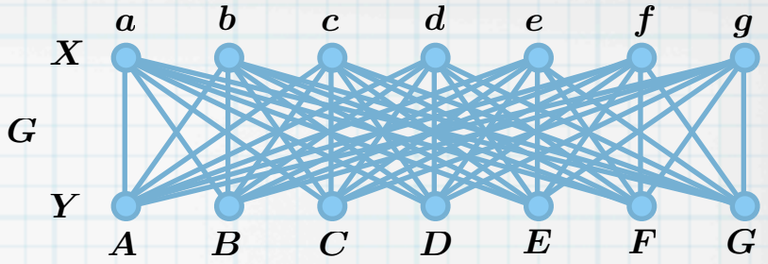
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$0\ 1\ 1\ 0\ 0\ 0\ 0$$

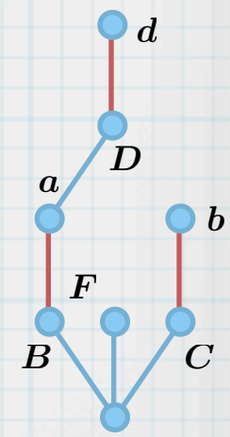
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

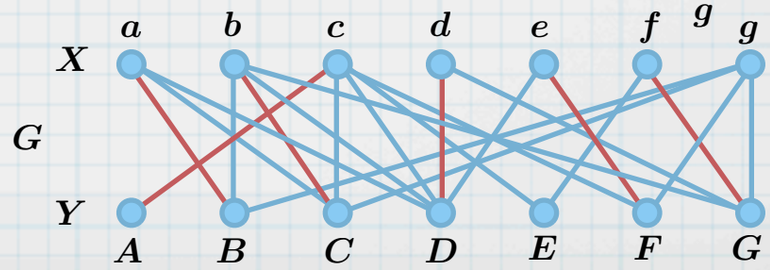
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

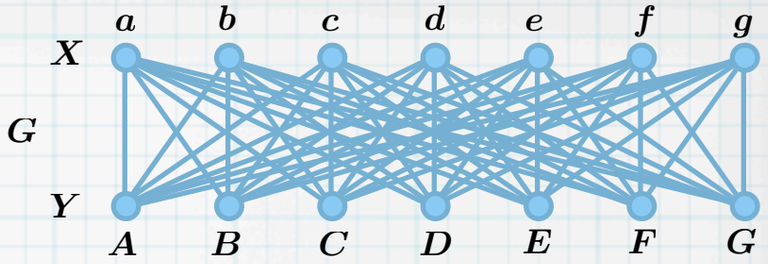
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = d$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

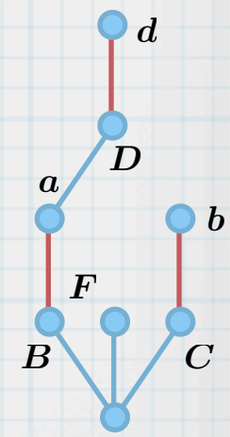
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

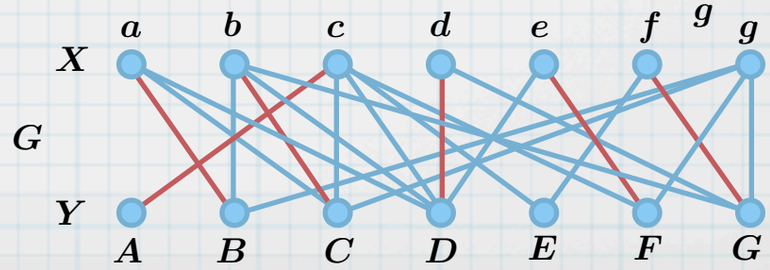
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

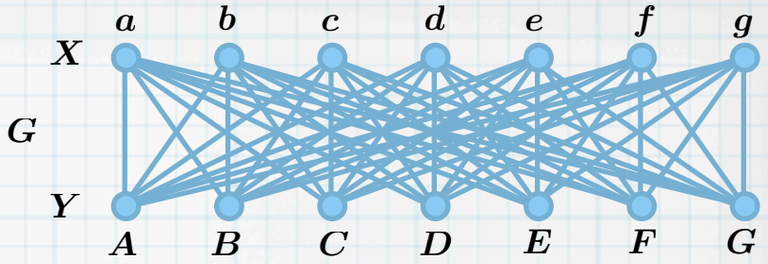
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

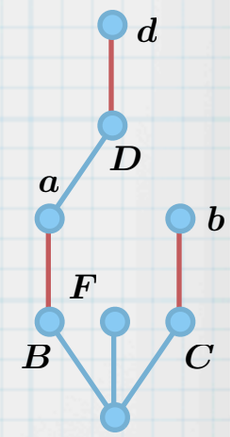
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

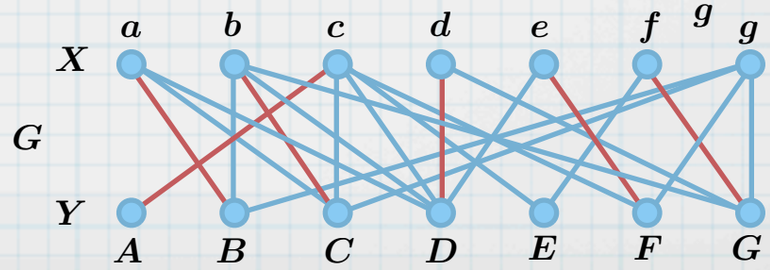
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

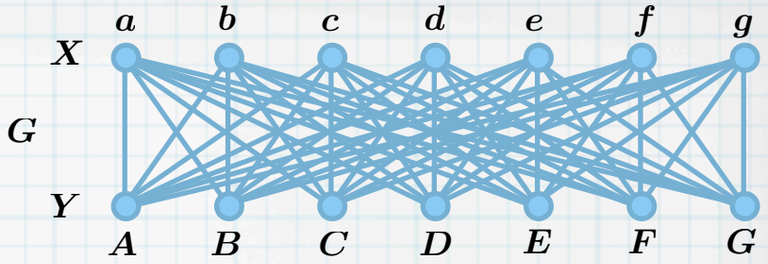
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

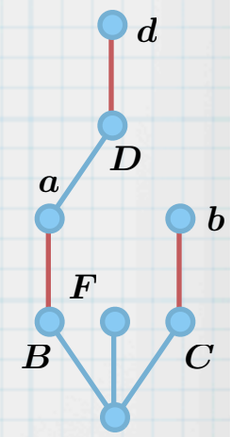
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

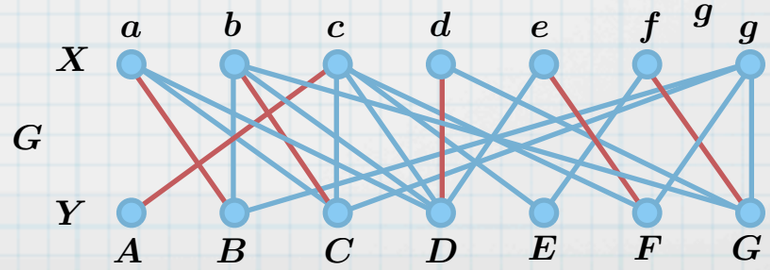
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

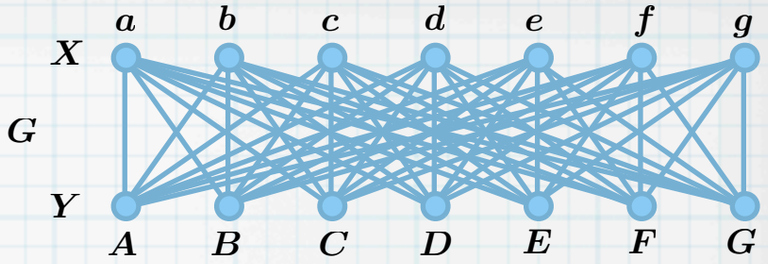
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

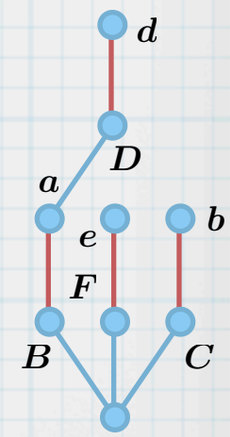
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

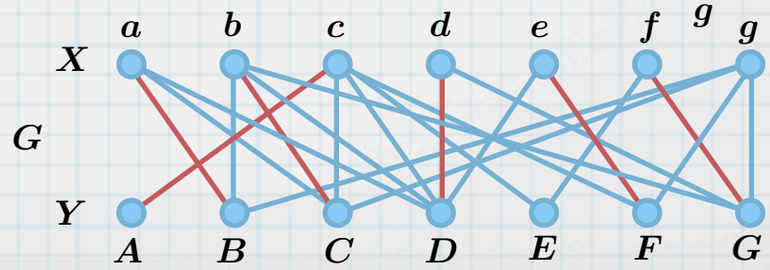
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

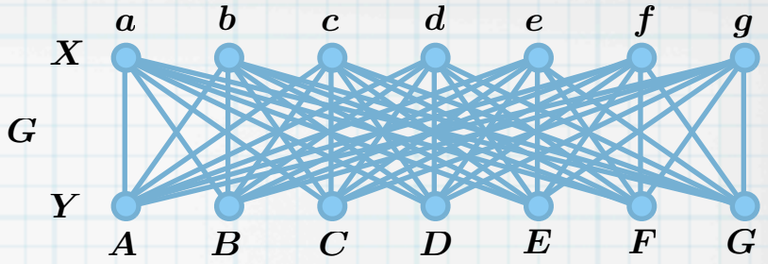
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

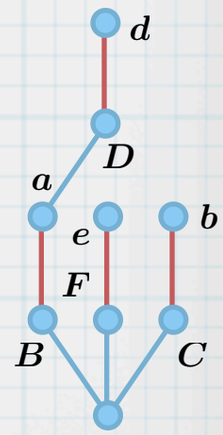
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

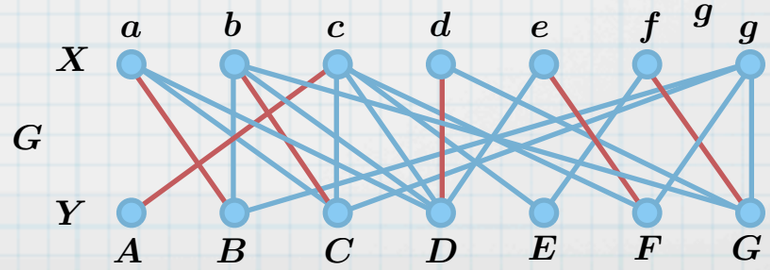
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

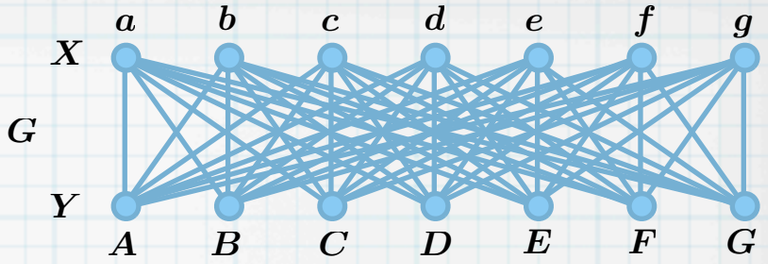
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

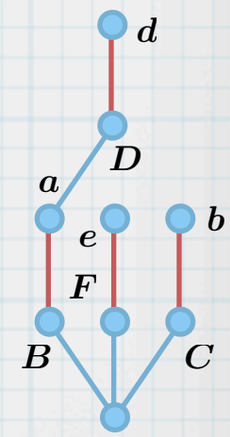
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

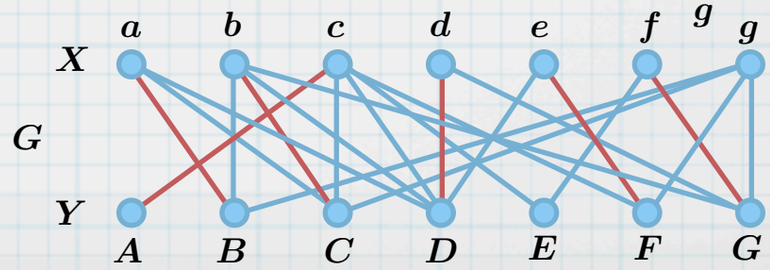
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

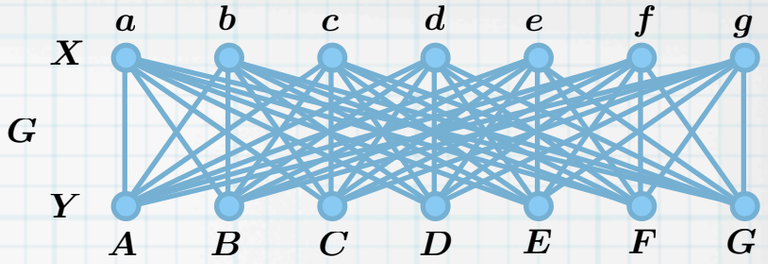
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

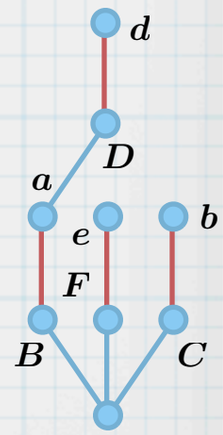
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

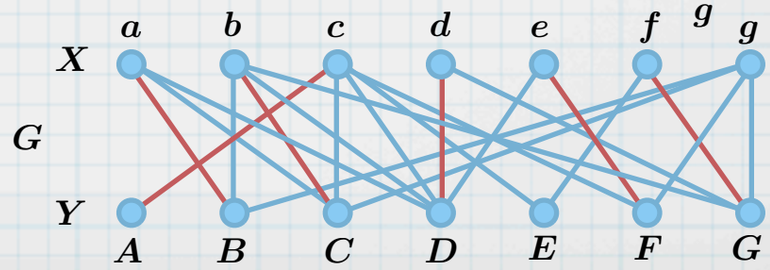
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

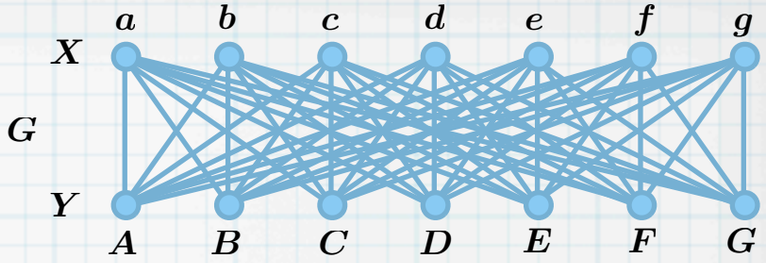
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

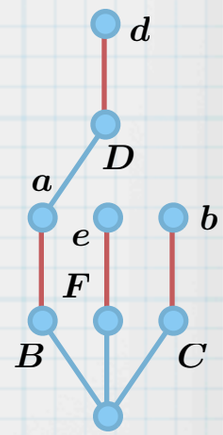
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

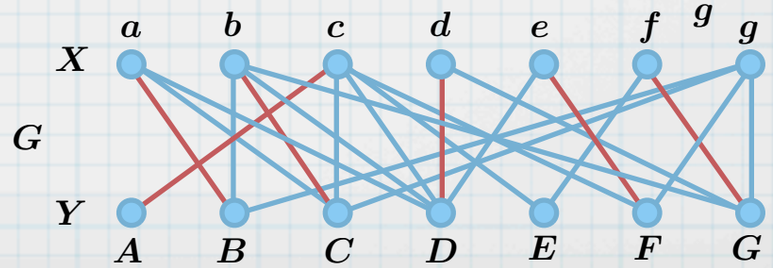
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

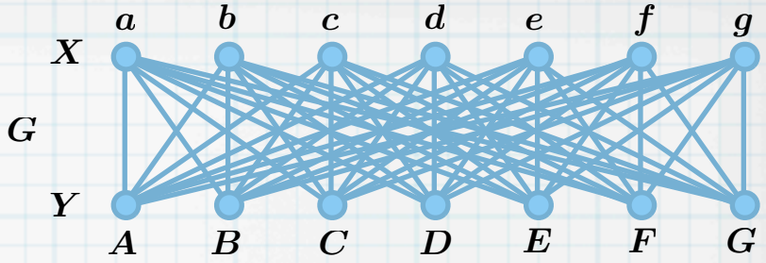
$$\{B, C, D, F, G\} \neq \{B, C, D\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

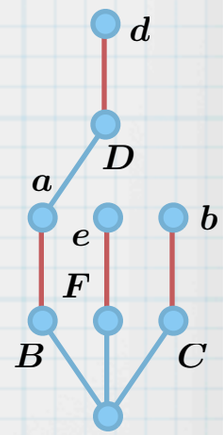
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

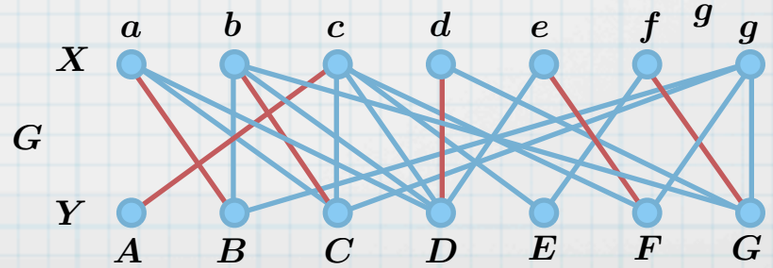
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

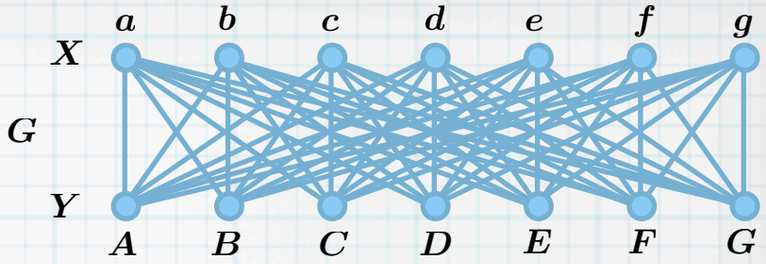
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

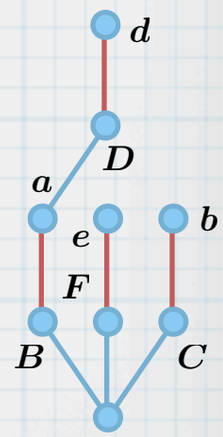
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

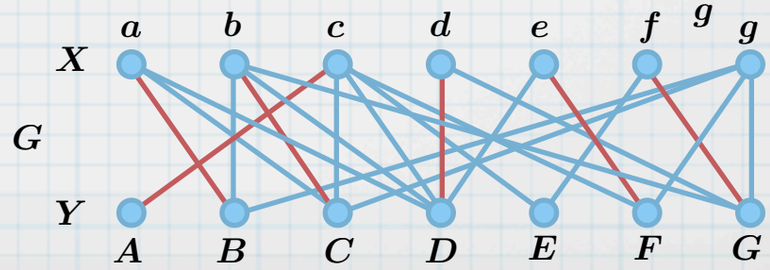
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

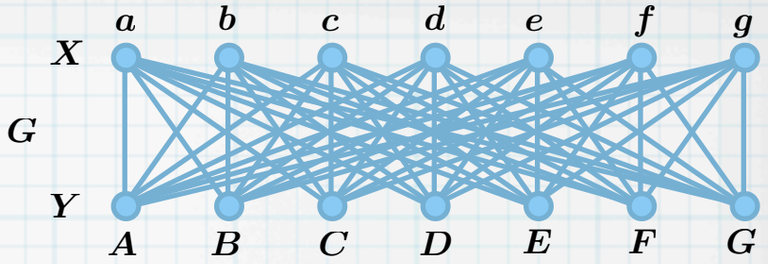
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

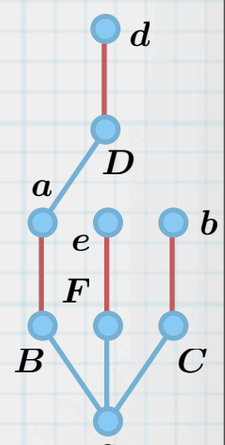
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

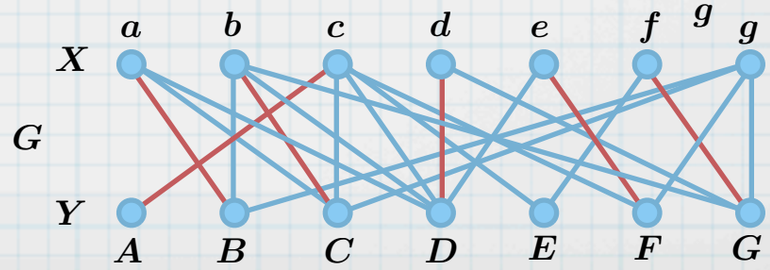
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

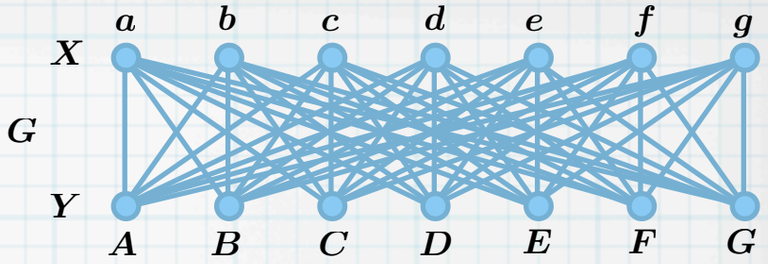
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$

$$y = F$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

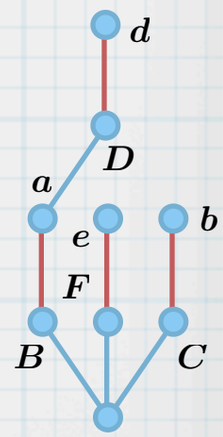
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

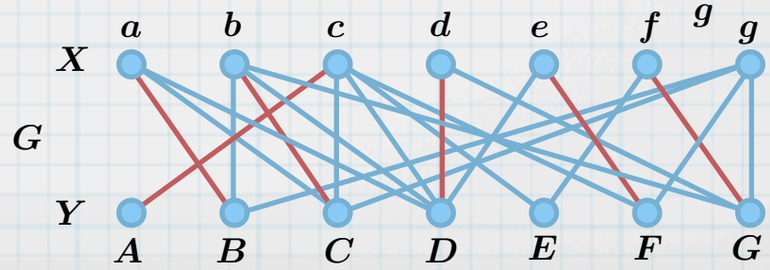
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

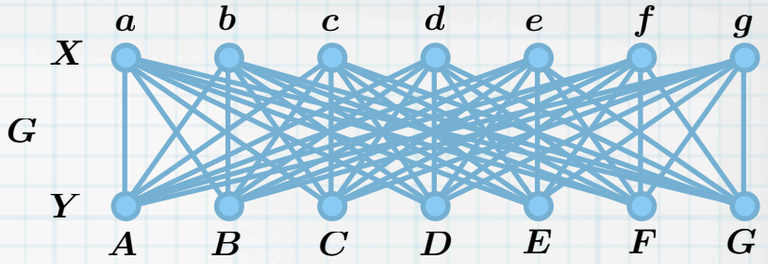
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$

$$y = G$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

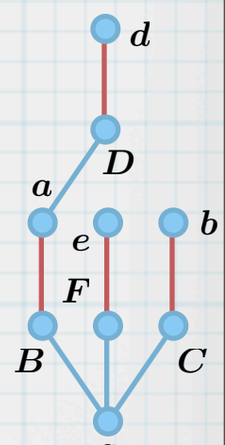
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

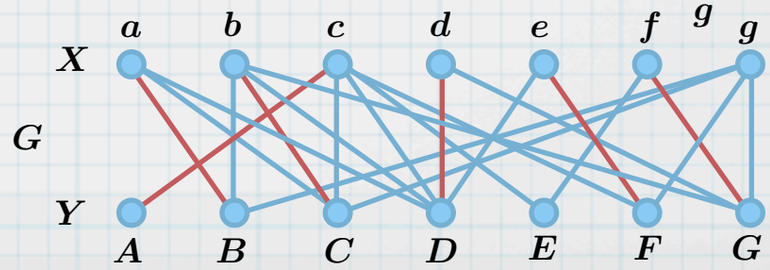
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

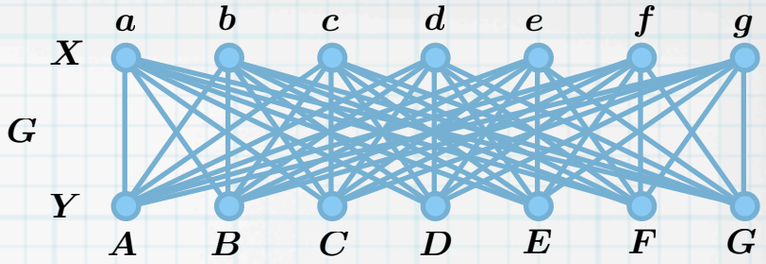
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$

$$y = G$$
$$T = \{B, C, D, F\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

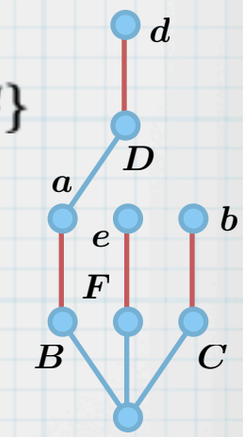
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

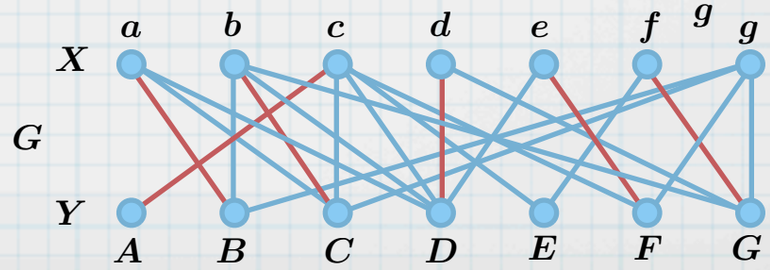
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

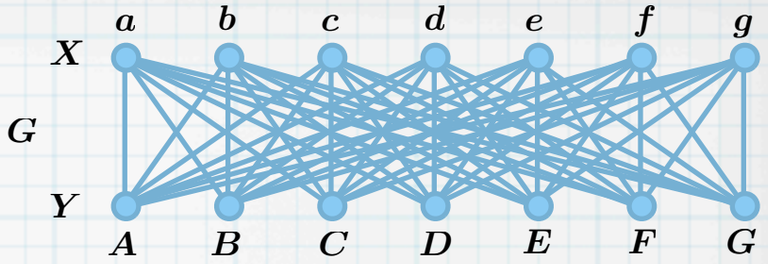
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$

$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



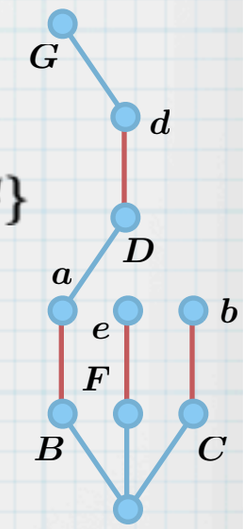
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

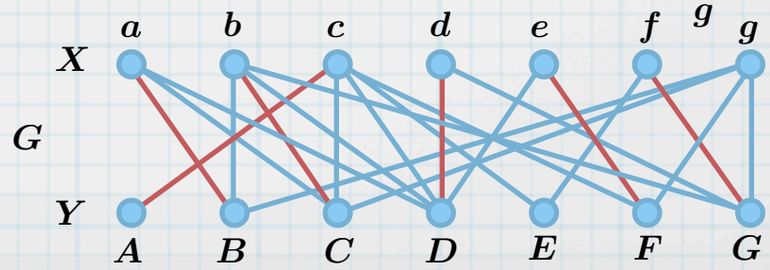
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

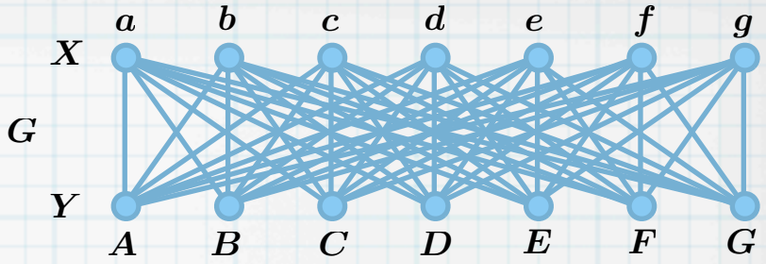
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$



$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$



Der Algorithmus von Kuhn-Munkres



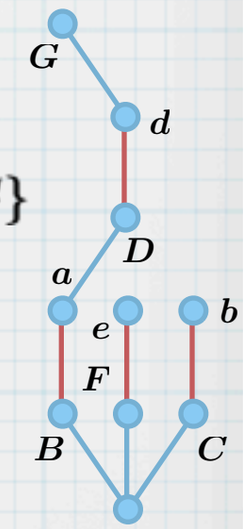
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

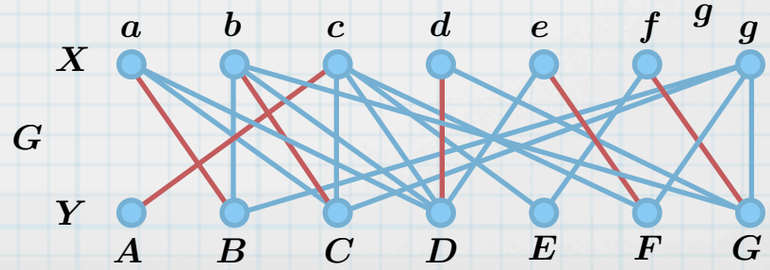
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

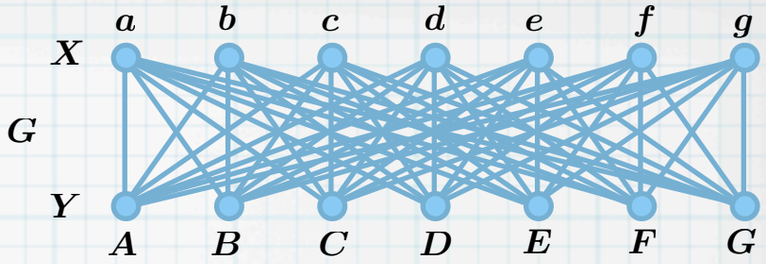
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$



$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = e$$
$$S = \{a, b, d, e, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

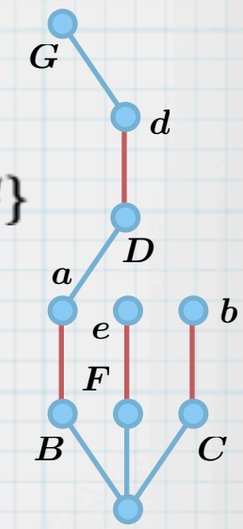
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

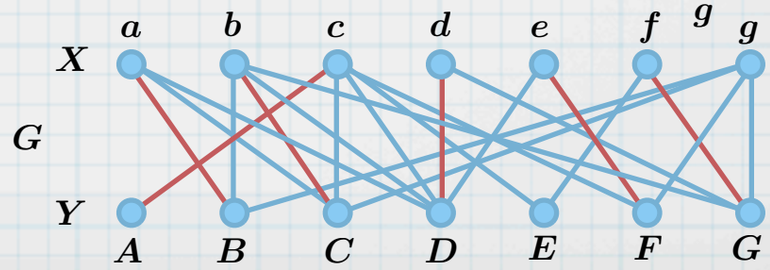
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

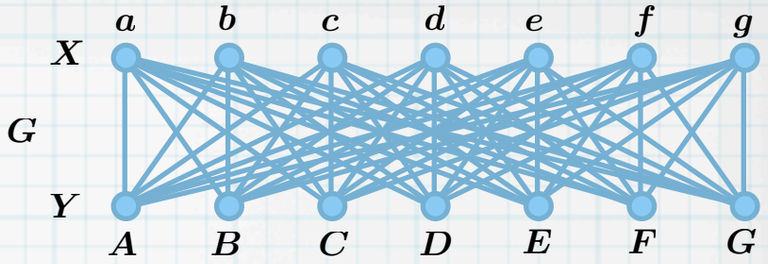
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$

$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

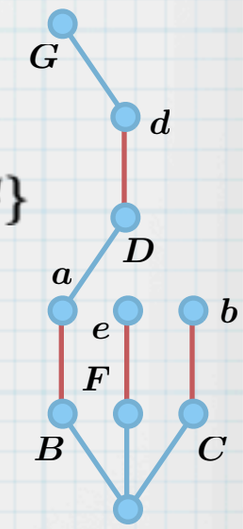
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

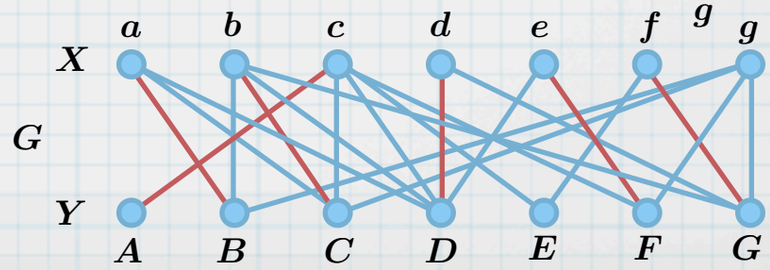
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

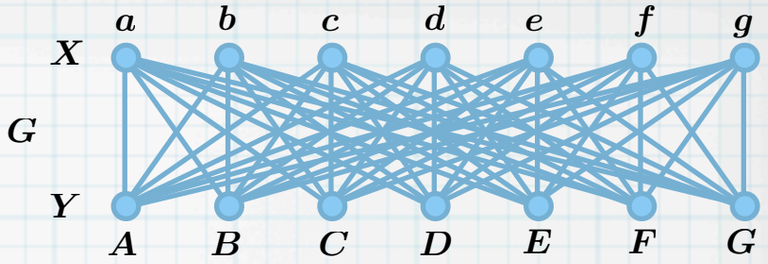
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$

$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, g\}$$





Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

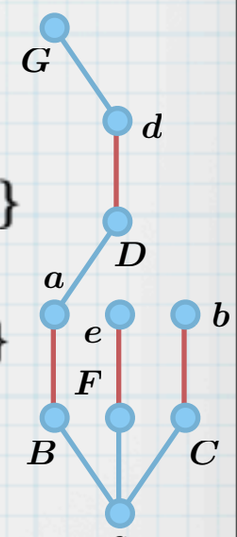
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

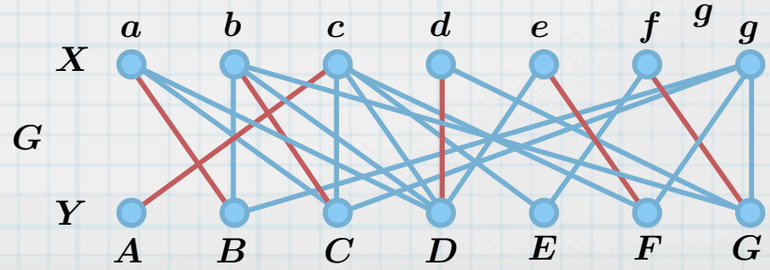
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

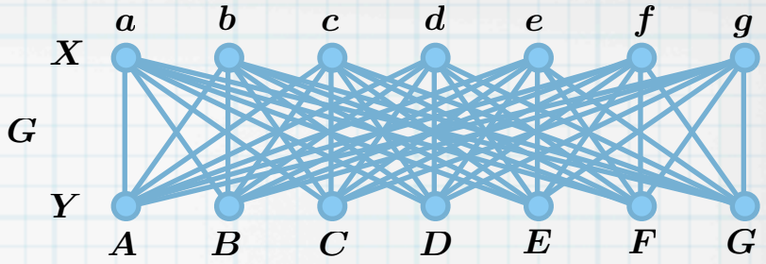
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$

$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$





Der Algorithmus von Kuhn-Munkres



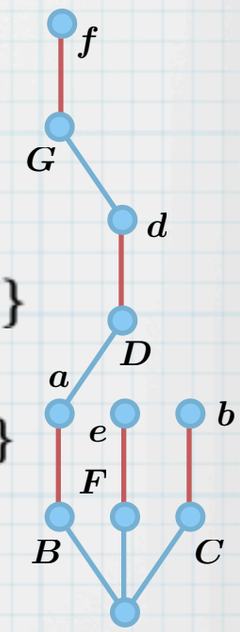
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

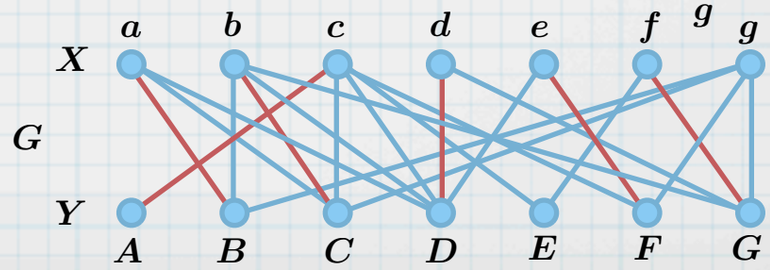
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

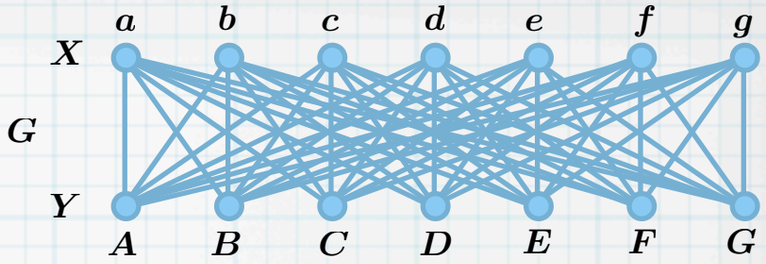
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$



$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

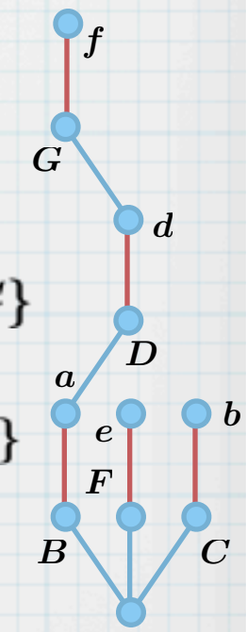
- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

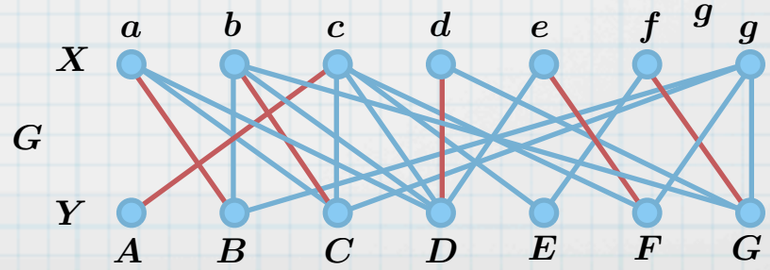
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$

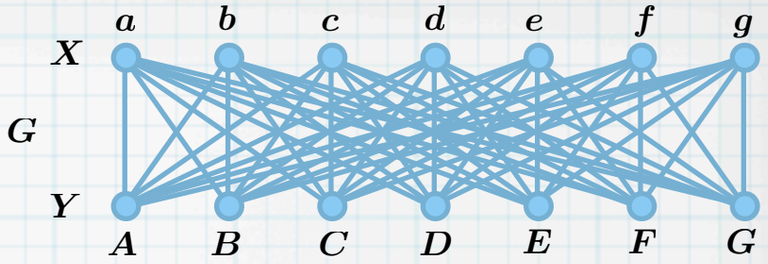
$$d_\lambda = 1$$



$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



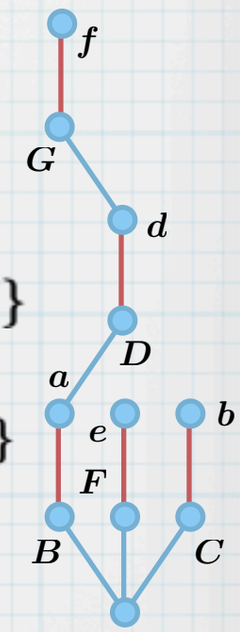
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

- Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
- Ausgabe: optimales Matching M.

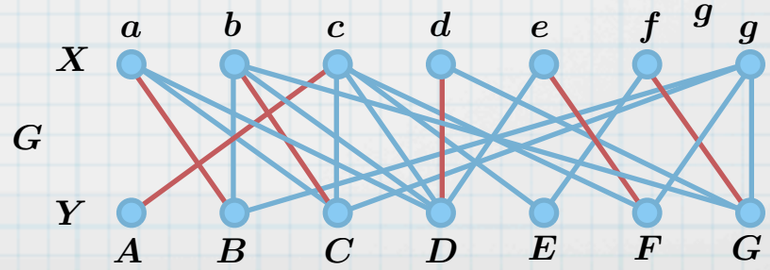
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

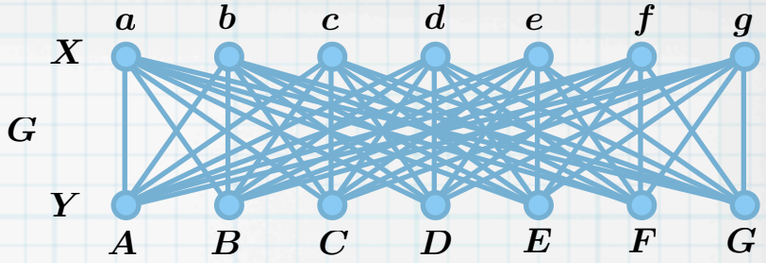
$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$
$$d_\lambda = 1$$



$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

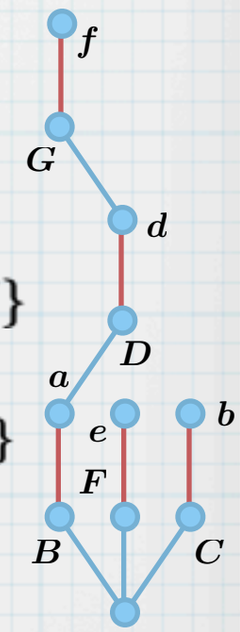
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

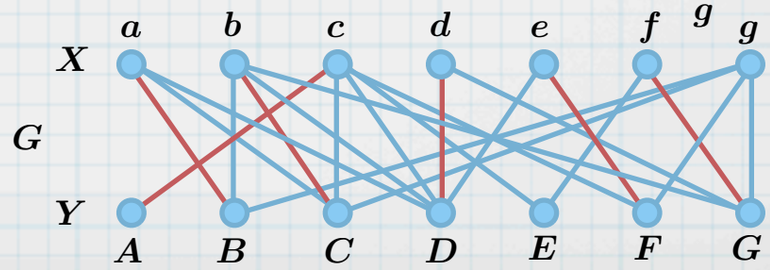
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$

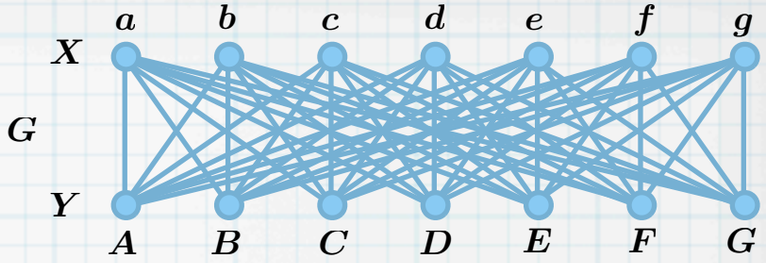
$$d_\lambda = 1$$



$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

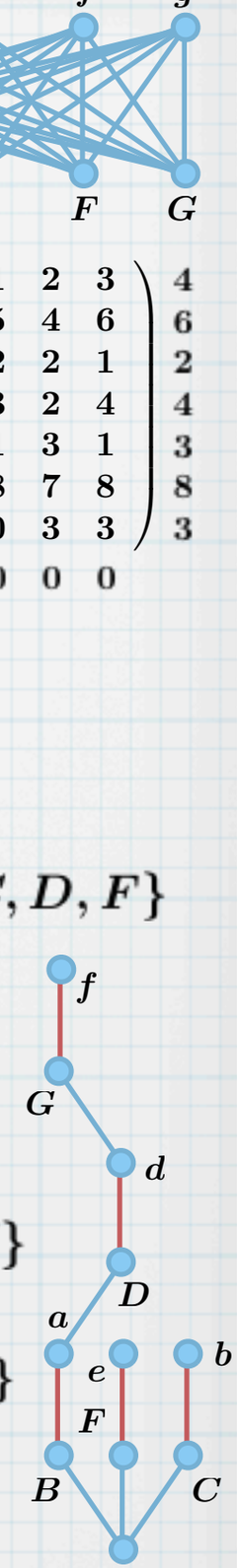
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

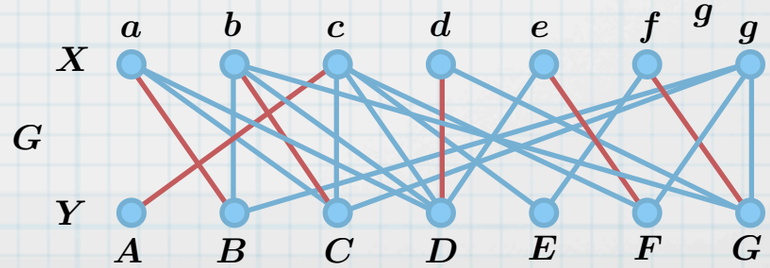
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, F, G\} \neq \{B, C, D, F\}$$

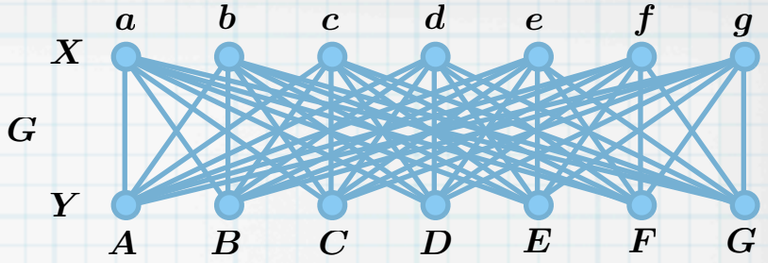
$$d_\lambda = 1$$



$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

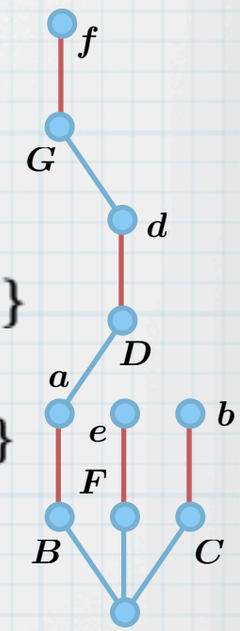
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

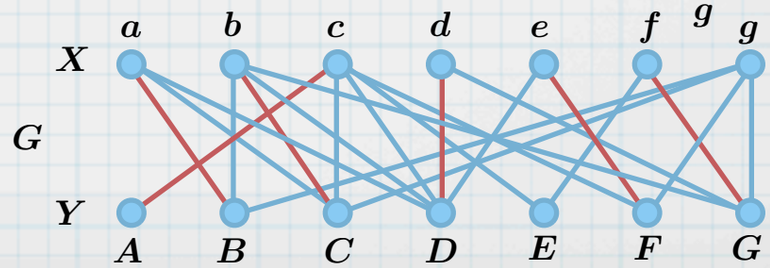
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$

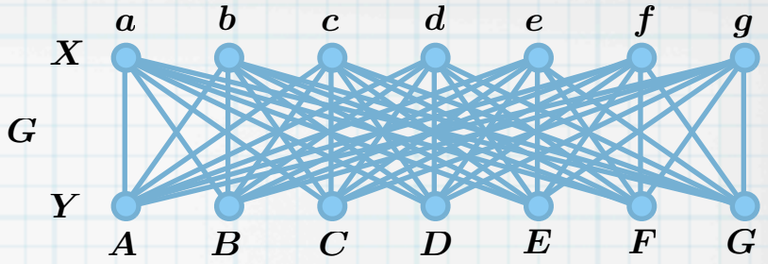
$$d_\lambda = 1$$



$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

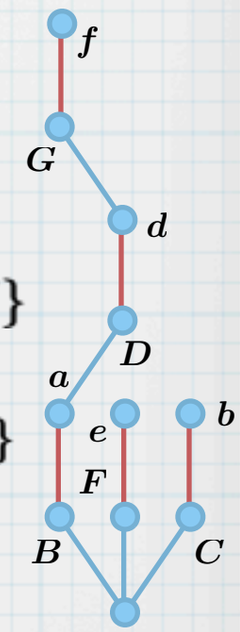
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

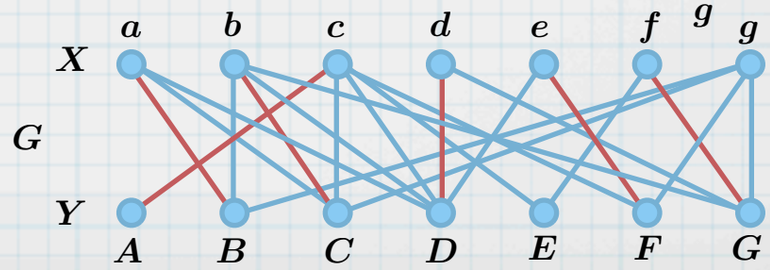
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$

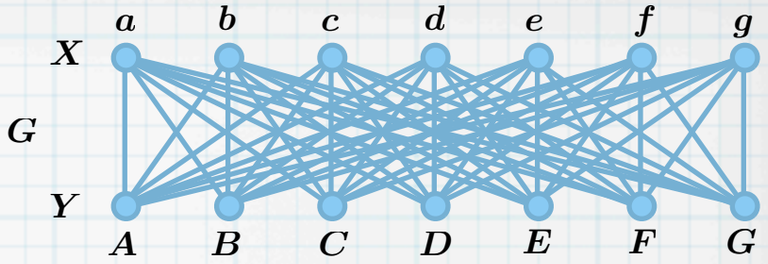
$$d_\lambda = 1$$



$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

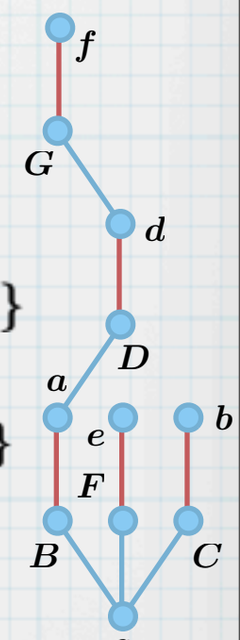
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

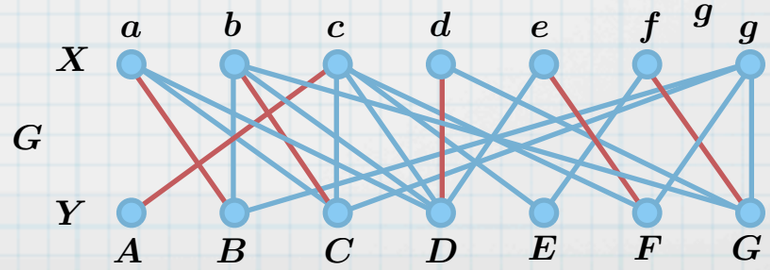
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$

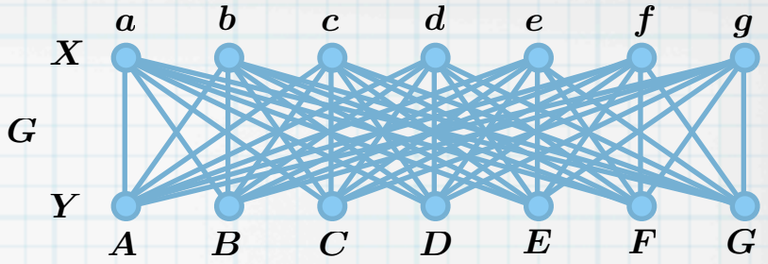
$$d_\lambda = 1$$

$$y = G$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$





Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

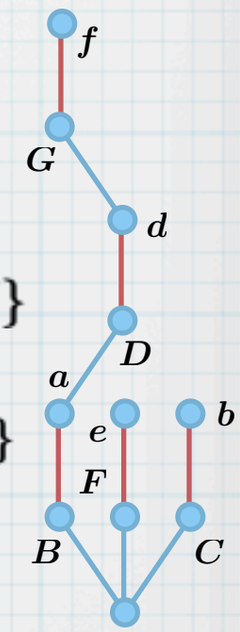
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

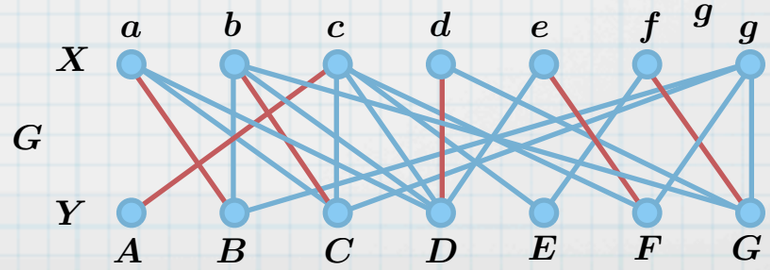
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$

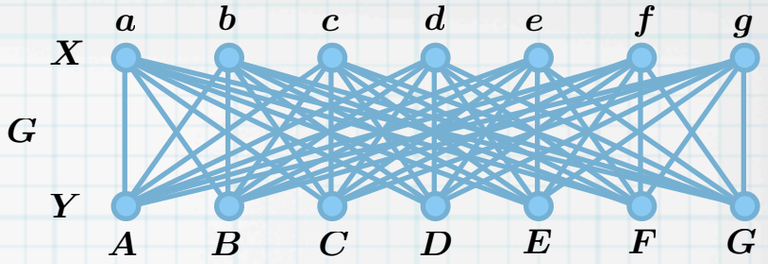
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

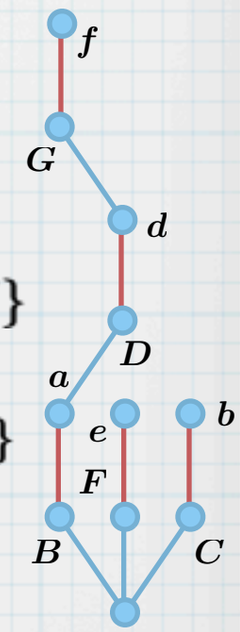
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

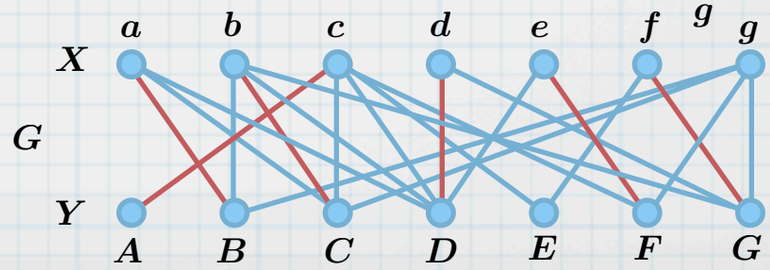
$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$

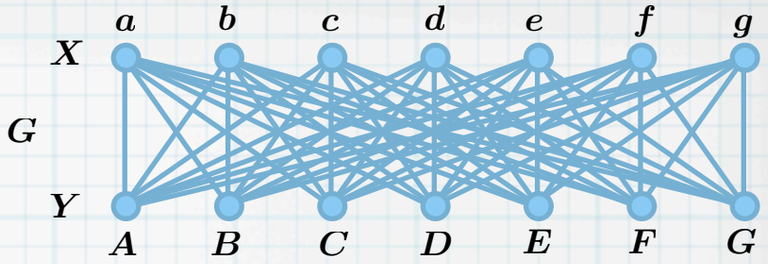
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



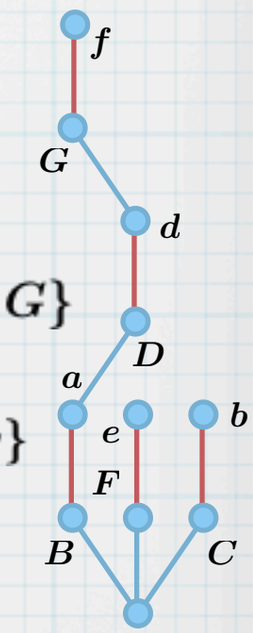
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

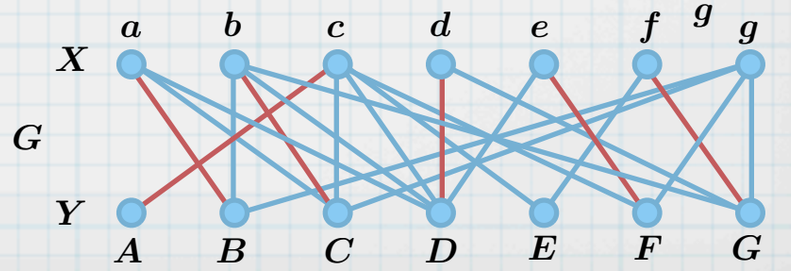
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

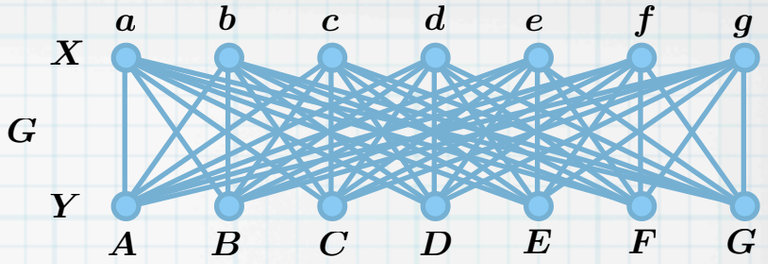
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



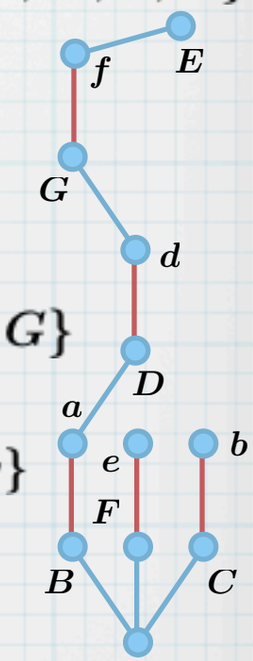
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

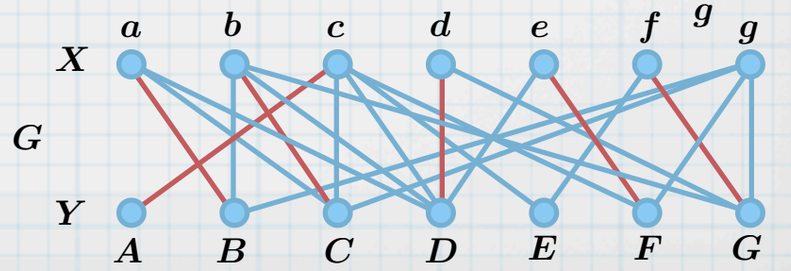
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

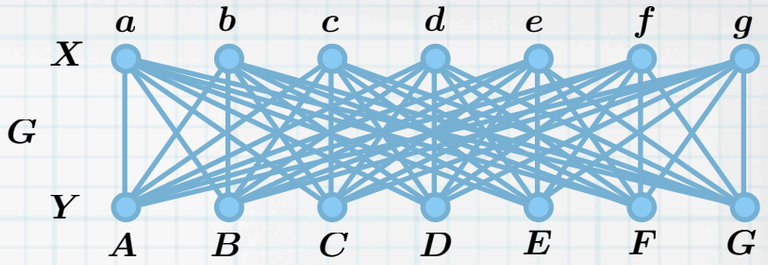
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



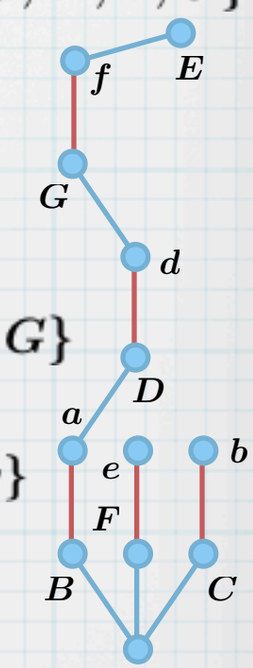
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

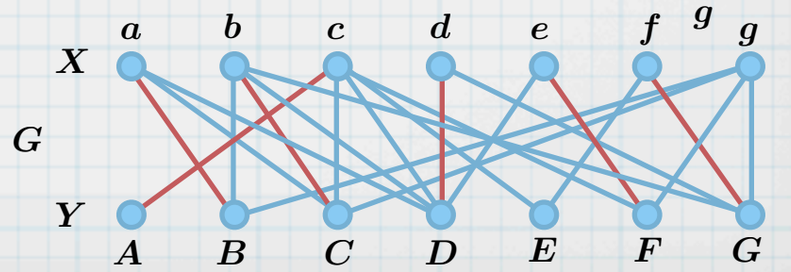
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

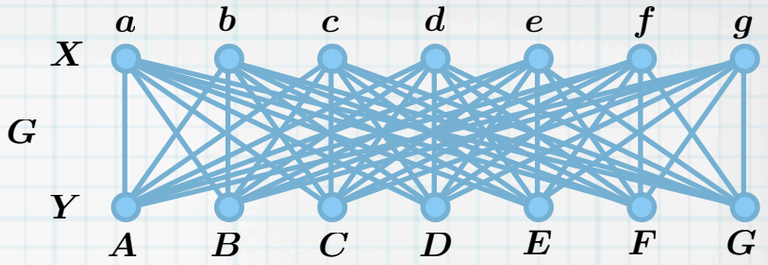
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



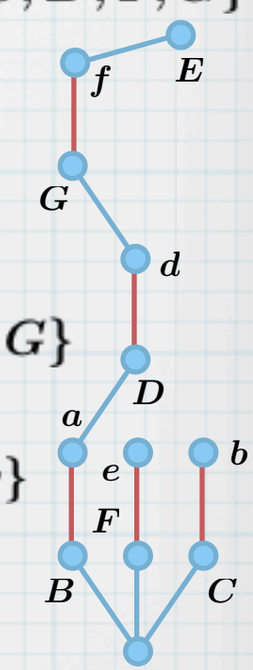
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

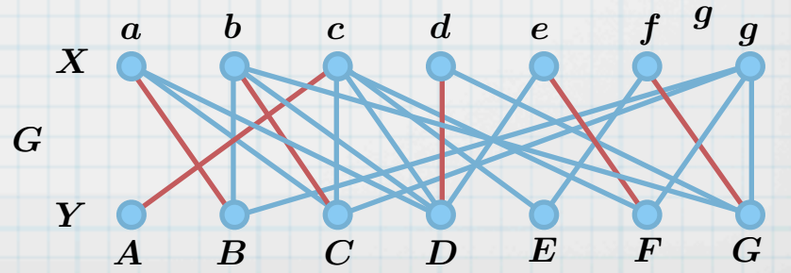
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

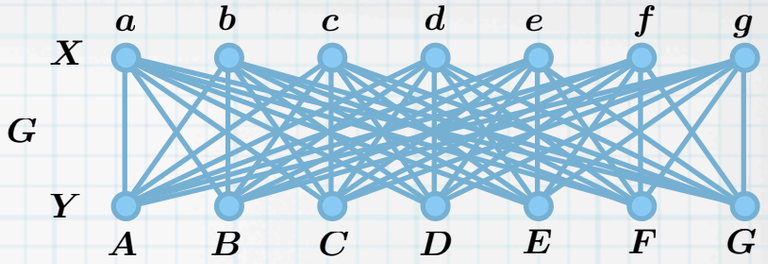
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



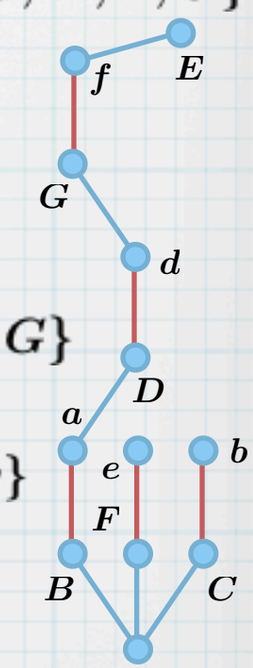
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt

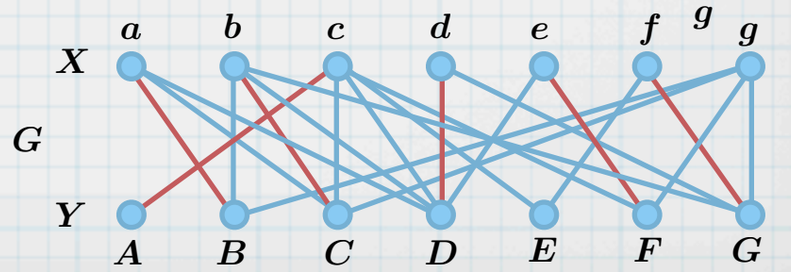
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

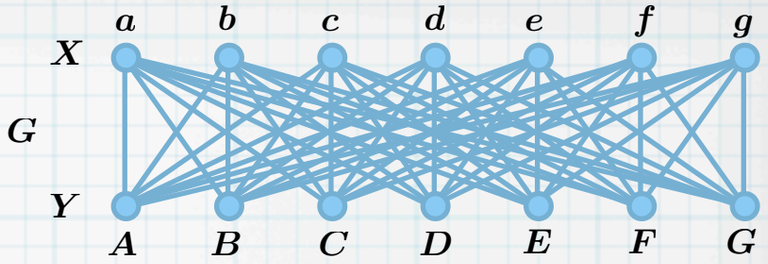
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



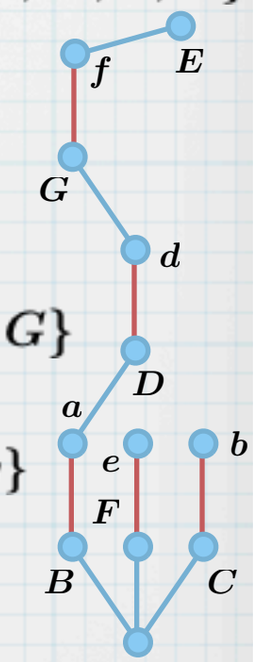
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um

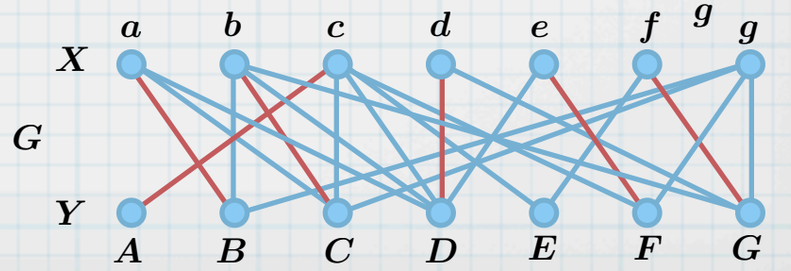
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

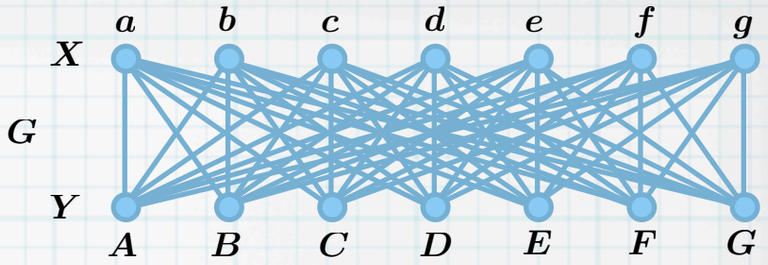
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



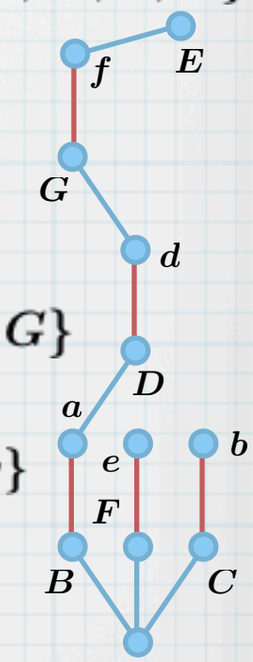
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um

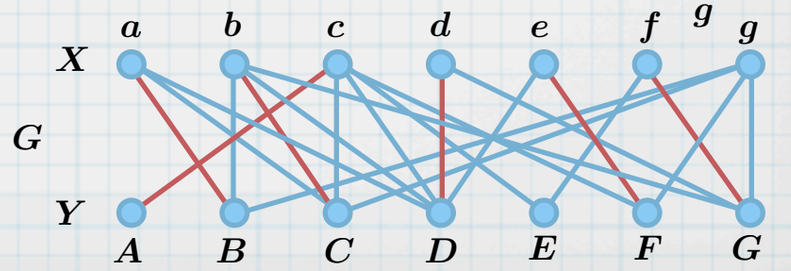
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

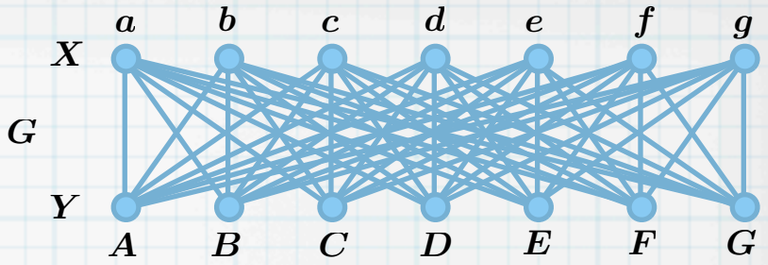
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



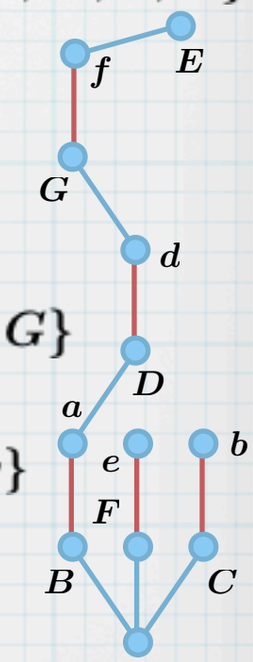
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

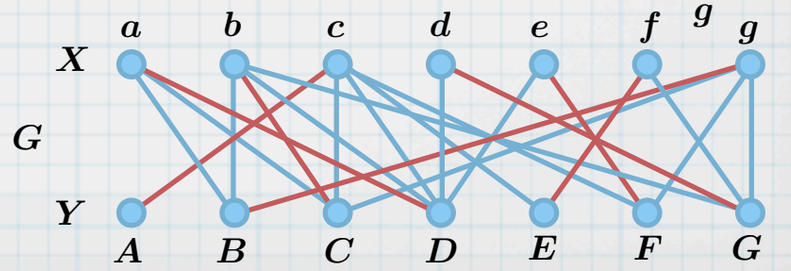
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

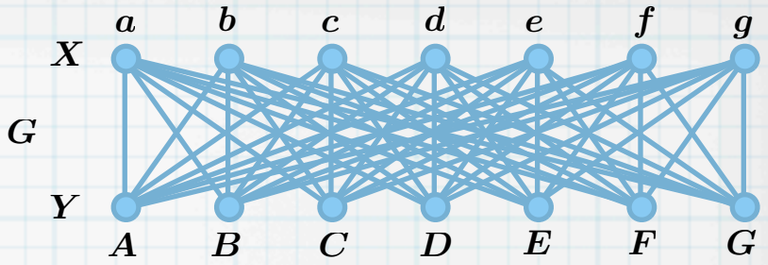
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



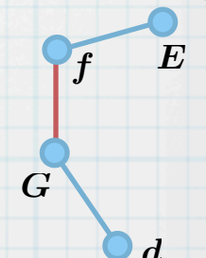
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

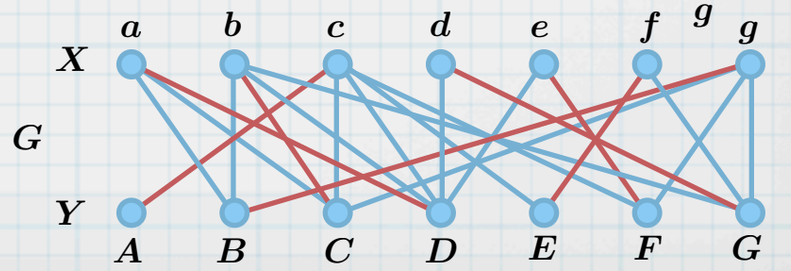
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

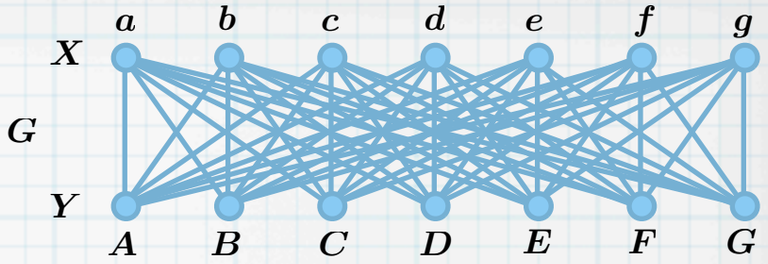
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



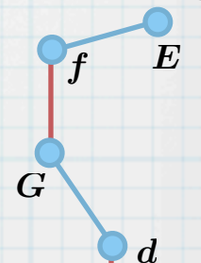
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

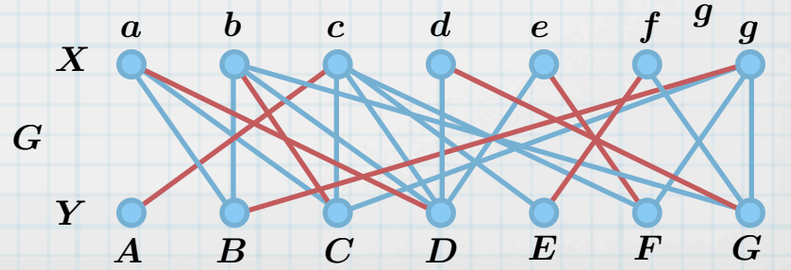
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

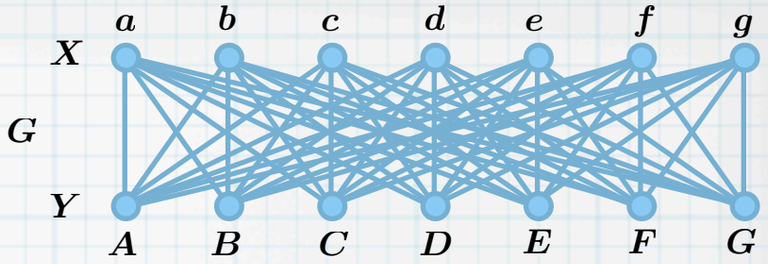
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



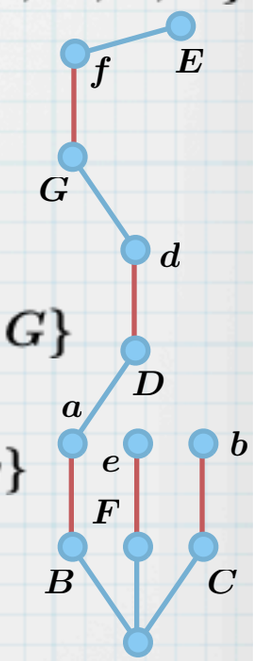
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

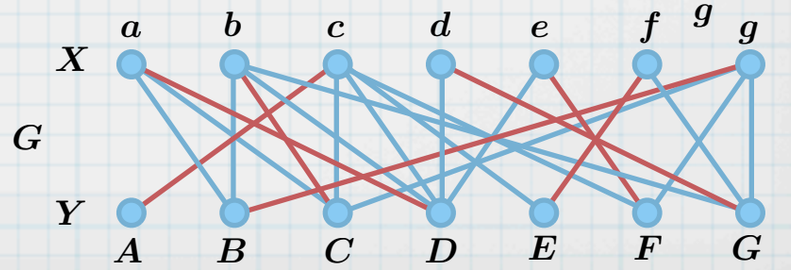
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M-ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

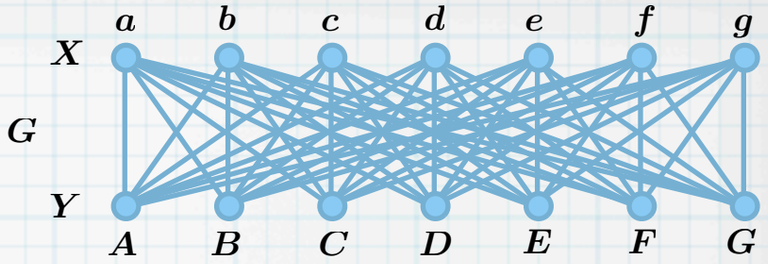
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



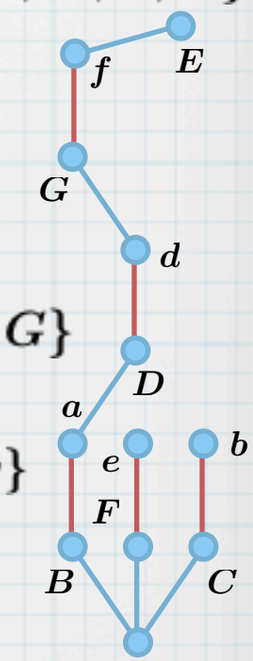
$$(w_{x,y})_{\substack{x\in X\\ y\in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

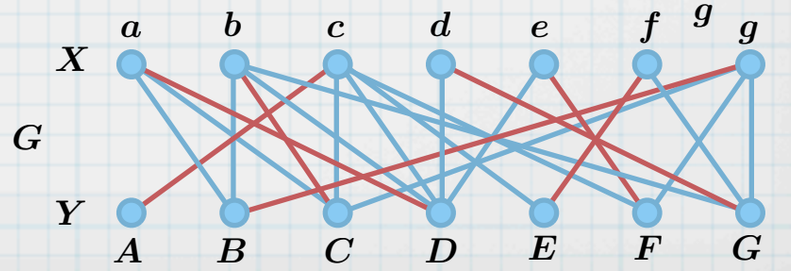
(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichnung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M-ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall\, v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall\, v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**
(19) **end algorithm**

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

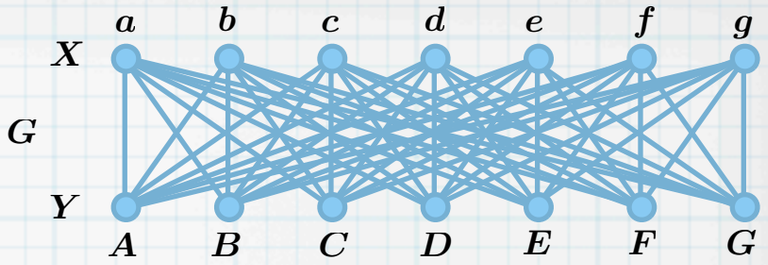
$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Der Algorithmus von Kuhn-Munkres



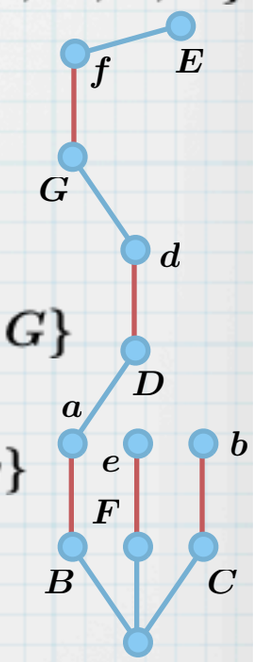
* Eingabe: vollständiger bipartiter Graph $G = K_{n,n} = (V, E)$ mit Bewertung $w : E \to \mathbb{R}$.
* Ausgabe: optimales Matching M.

(1) **algorithm** kuhnMunkres
(2) wähle eine zul. Knotenkennzeichung λ
(3) wähle ein beliebiges Matching M von G_λ
(4) **while** es gibt M–ungesättigten Knoten x_0 **do**
(5) $S := \{x_0\}, T := \emptyset$
(6) **repeat**
(7) **if** $N_{G_\lambda}(S) = T$ **then**
(8) $d_\lambda := \min\{\lambda(x) + \lambda(y) - w_{x,y} : x \in S, y \notin T\}$
(9) $\lambda(v) := \lambda(v) - d_\lambda \; \forall v \in S, \lambda(v) := \lambda(v) + d_\lambda \; \forall v \in T$
(10) **end if**
(11) wähle $y \in N_{G_\lambda}(S) \backslash T$
(12) $T := T \cup \{y\}$
(13) **if** es gibt ein z mit $\{y, z\} \in M$ **then**
(14) $S := S \cup \{z\}$
(15) **end if**
(16) **until** y ungesättigt
(17) färbe M entlang des Weges von x_0 nach y um
(18) **end while**
(19) **end algorithm**

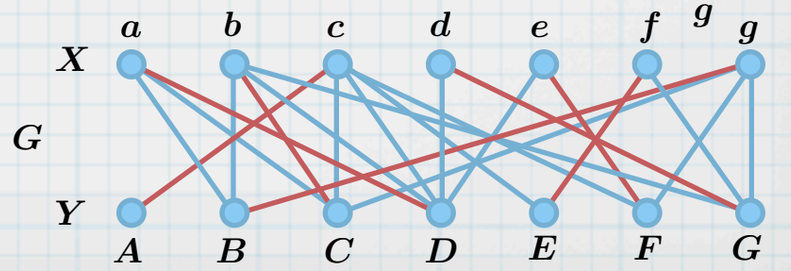
$$(w_{x,y})_{\substack{x \in X \\ y \in Y}} = \begin{pmatrix} 3 & 5 & 5 & 4 & 1 & 2 & 3 \\ 5 & 7 & 7 & 6 & 5 & 4 & 6 \\ 2 & 0 & 2 & 2 & 2 & 2 & 1 \\ 2 & 4 & 4 & 4 & 3 & 2 & 4 \\ 1 & 2 & 1 & 3 & 1 & 3 & 1 \\ 6 & 8 & 8 & 5 & 8 & 7 & 8 \\ 2 & 4 & 4 & 1 & 0 & 3 & 3 \end{pmatrix} \begin{matrix} 4 \\ 6 \\ 2 \\ 4 \\ 3 \\ 8 \\ 3 \end{matrix}$$
$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

$$x_0 = g$$
$$S = \{g\}, T = \{\}$$

$$\{B, C, D, E, F, G\} \neq \{B, C, D, F, G\}$$
$$d_\lambda = 1$$



$$y = E$$
$$T = \{B, C, D, E, F, G\}$$
$$z = f$$
$$S = \{a, b, d, e, f, g\}$$



Lösung des chinesischen Briefträgerproblems

Lösung des chinesischen Briefträgerproblems

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn

  der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn

  der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder

  der Graph genau zwei Knoten ungeraden Grades hat.

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn

  der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder

  der Graph genau zwei Knoten ungeraden Grades hat.

* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn

  der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder

  der Graph genau zwei Knoten ungeraden Grades hat.

* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:

  Sei U die Menge der ungeraden Knoten in G.

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn

  der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder

  der Graph genau zwei Knoten ungeraden Grades hat.

* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:

  Sei U die Menge der ungeraden Knoten in G.

  Für jedes Paar ungerader Knoten $u, v \in U$ berechne man ihre Entfernung $d_{u,v}$ im Graphen G bzgl. der Bewertung c (z.B. mittels des Algorithmus von Dijkstra oder Floyd-Warshal).

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn

  der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder

  der Graph genau zwei Knoten ungeraden Grades hat.

* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:

  Sei U die Menge der ungeraden Knoten in G.

  Für jedes Paar ungerader Knoten $u, v \in U$ berechne man ihre Entfernung $d_{u,v}$ im Graphen G bzgl. der Bewertung c (z.B. mittels des Algorithmus von Dijkstra oder Floyd–Warshal).

  Wir definieren einen vollständigen bewerteteten Graphen (K_U, d).

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn

  der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder

  der Graph genau zwei Knoten ungeraden Grades hat.

* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:

  Sei U die Menge der ungeraden Knoten in G.

  Für jedes Paar ungerader Knoten $u, v \in U$ berechne man ihre Entfernung $d_{u,v}$ im Graphen G bzgl. der Bewertung c (z.B. mittels des Algorithmus von Dijkstra oder Floyd–Warshal).

  Wir definieren einen vollständigen bewerteteten Graphen (K_U, d).

  Bemerke, dass die Anzahl ungerader Knoten gerade ist.

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn

  der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder

  der Graph genau zwei Knoten ungeraden Grades hat.

* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:

  Sei U die Menge der ungeraden Knoten in G.

  Für jedes Paar ungerader Knoten $u, v \in U$ berechne man ihre Entfernung $d_{u,v}$ im Graphen G bzgl. der Bewertung c (z.B. mittels des Algorithmus von Dijkstra oder Floyd–Warshal).

  Wir definieren einen vollständigen bewerteteten Graphen (K_U, d).

  Bemerke, dass die Anzahl ungerader Knoten gerade ist.

  Also hat K_U ein perfektes Matching.

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn
  - der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder
  - der Graph genau zwei Knoten ungeraden Grades hat.
* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:
  - Sei U die Menge der ungeraden Knoten in G.
  - Für jedes Paar ungerader Knoten $u, v \in U$ berechne man ihre Entfernung $d_{u,v}$ im Graphen G bzgl. der Bewertung c (z.B. mittels des Algorithmus von Dijkstra oder Floyd–Warshal).
  - Wir definieren einen vollständigen bewerteteten Graphen (K_U, d).
  - Bemerke, dass die Anzahl ungerader Knoten gerade ist.
  - Also hat K_U ein perfektes Matching.
  - Es gibt effiziente Algorithmen (Edmonds, 1965) zur Bestimmung eines minimalen perfekten Matchings M^* in (K_U, d).

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn
  - der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder
  - der Graph genau zwei Knoten ungeraden Grades hat.
* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:
  - Sei U die Menge der ungeraden Knoten in G.
  - Für jedes Paar ungerader Knoten $u, v \in U$ berechne man ihre Entfernung $d_{u,v}$ im Graphen G bzgl. der Bewertung c (z.B. mittels des Algorithmus von Dijkstra oder Floyd–Warshal).
  - Wir definieren einen vollständigen bewerteteten Graphen (K_U, d).
  - Bemerke, dass die Anzahl ungerader Knoten gerade ist.
  - Also hat K_U ein perfektes Matching.
  - Es gibt effiziente Algorithmen (Edmonds, 1965) zur Bestimmung eines minimalen perfekten Matchings M^* in (K_U, d).
  - Die Wege in G, die den Kanten in M^* entsprechen, werden in G verdoppelt.

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn
  - der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder
  - der Graph genau zwei Knoten ungeraden Grades hat.
* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:
  - Sei U die Menge der ungeraden Knoten in G.
  - Für jedes Paar ungerader Knoten $u, v \in U$ berechne man ihre Entfernung $d_{u,v}$ im Graphen G bzgl. der Bewertung c (z.B. mittels des Algorithmus von Dijkstra oder Floyd–Warshal).
  - Wir definieren einen vollständigen bewerteteten Graphen (K_U, d).
  - Bemerke, dass die Anzahl ungerader Knoten gerade ist.
  - Also hat K_U ein perfektes Matching.
  - Es gibt effiziente Algorithmen (Edmonds, 1965) zur Bestimmung eines minimalen perfekten Matchings M^* in (K_U, d).
  - Die Wege in G, die den Kanten in M^* entsprechen, werden in G verdoppelt.
  - Dadurch entsteht ein Eulerscher Obergraph G' von G.

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn
  - der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder
  - der Graph genau zwei Knoten ungeraden Grades hat.
* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:
  - Sei U die Menge der ungeraden Knoten in G.
  - Für jedes Paar ungerader Knoten $u, v \in U$ berechne man ihre Entfernung $d_{u,v}$ im Graphen G bzgl. der Bewertung c (z.B. mittels des Algorithmus von Dijkstra oder Floyd–Warshal).
  - Wir definieren einen vollständigen bewerteteten Graphen (K_U, d).
  - Bemerke, dass die Anzahl ungerader Knoten gerade ist.
  - Also hat K_U ein perfektes Matching.
  - Es gibt effiziente Algorithmen (Edmonds, 1965) zur Bestimmung eines minimalen perfekten Matchings M^* in (K_U, d).
  - Die Wege in G, die den Kanten in M^* entsprechen, werden in G verdoppelt.
  - Dadurch entsteht ein Eulerscher Obergraph G' von G.
  - In diesem kann mittels des Algorithmus von Fleury eine Eulertour bestimmt werden.

Lösung des chinesischen Briefträgerproblems

* Aus dem letzten Kapitel (#6) haben wir gesehen, wie das chinesische Briefträgerproblem (CPP) zu lösen ist, wenn
  - der zu Grunde liegende Graph $G = (V, E)$ eulersch ist, oder
  - der Graph genau zwei Knoten ungeraden Grades hat.
* In der allgemeinen Situation (mehr als zwei ungerade Knoten) kann eine Lösung wie folgt ermittelt werden:
  - Sei U die Menge der ungeraden Knoten in G.
  - Für jedes Paar ungerader Knoten $u, v \in U$ berechne man ihre Entfernung $d_{u,v}$ im Graphen G bzgl. der Bewertung c (z.B. mittels des Algorithmus von Dijkstra oder Floyd–Warshal).
  - Wir definieren einen vollständigen bewerteteten Graphen (K_U, d).
  - Bemerke, dass die Anzahl ungerader Knoten gerade ist.
  - Also hat K_U ein perfektes Matching.
  - Es gibt effiziente Algorithmen (Edmonds, 1965) zur Bestimmung eines minimalen perfekten Matchings M^* in (K_U, d).
  - Die Wege in G, die den Kanten in M^* entsprechen, werden in G verdoppelt.
  - Dadurch entsteht ein Eulerscher Obergraph G' von G.
  - In diesem kann mittels des Algorithmus von Fleury eine Eulertour bestimmt werden.
  - Diese ist dann die Lösung des chinesischen Briefträgerproblems.

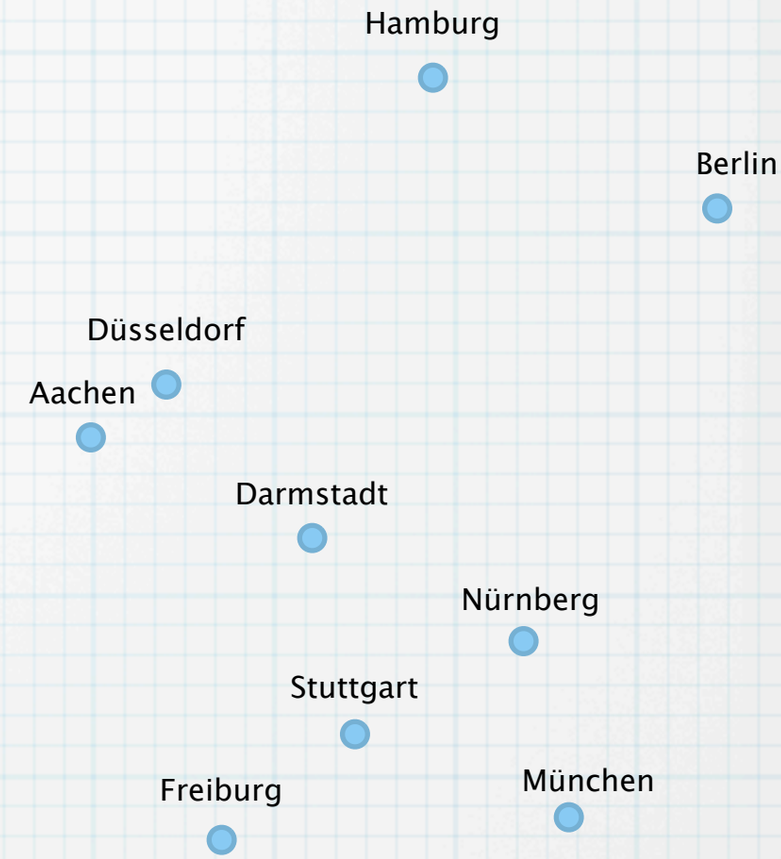
Der Algorithmus von Christofides (1976)

Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP

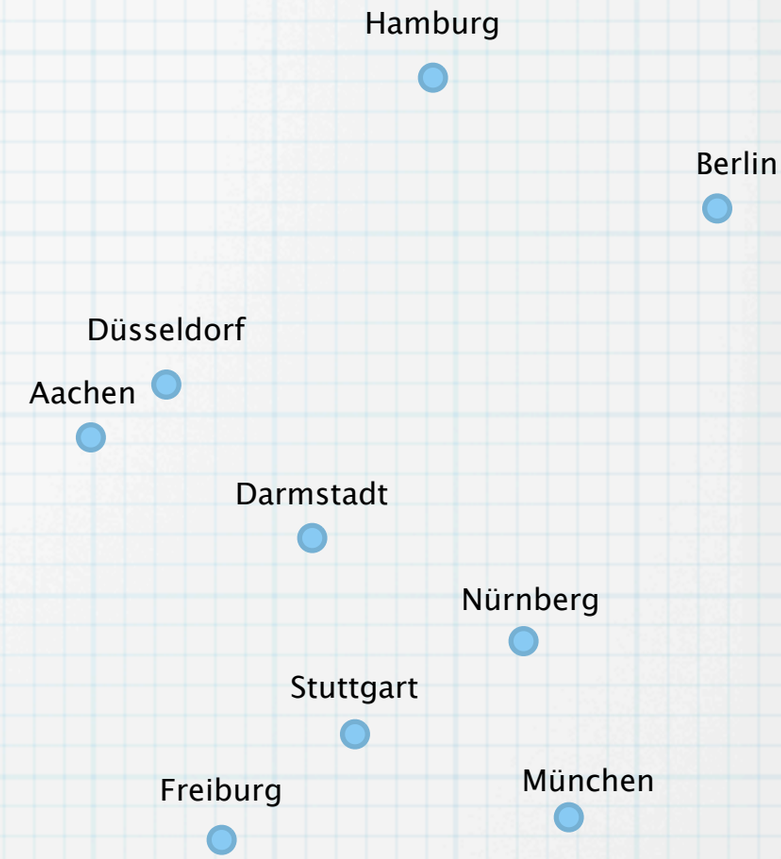
Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP



Der Algorithmus von Christofides (1976)

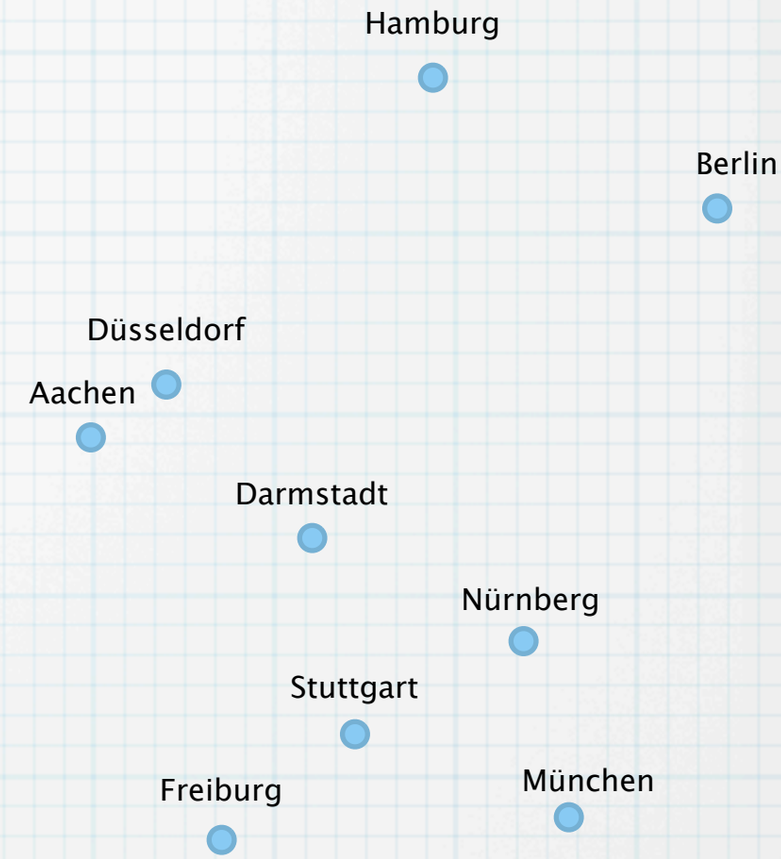
* Eingabe: Instanz (K_n, c) für mTSP



	Aa	*Be*	*Da*	*Dü*	*Fr*	*Ha*	*Mü*	*Nü*	*St*
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
Dü	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
Mü	64	60	40	63	37	80	0	17	22
Nü	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

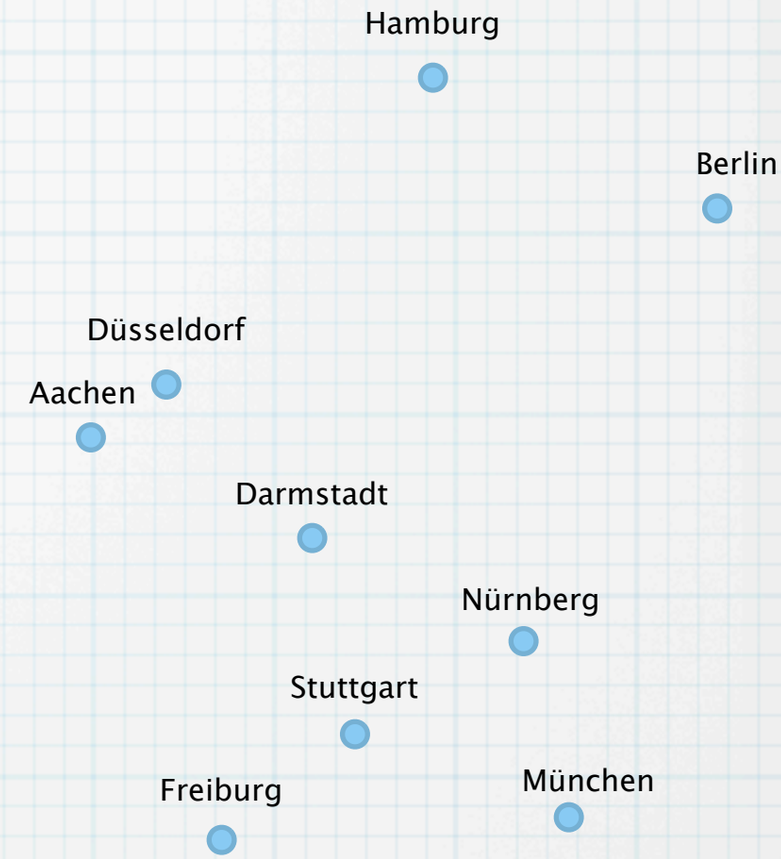


	Aa	*Be*	*Da*	*Dü*	*Fr*	*Ha*	*Mü*	*Nü*	*St*
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
Dü	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
Mü	64	60	40	63	37	80	0	17	22
Nü	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides



	Aa	*Be*	*Da*	*Dü*	*Fr*	*Ha*	*Mü*	*Nü*	*St*
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
Dü	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
Mü	64	60	40	63	37	80	0	17	22
Nü	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

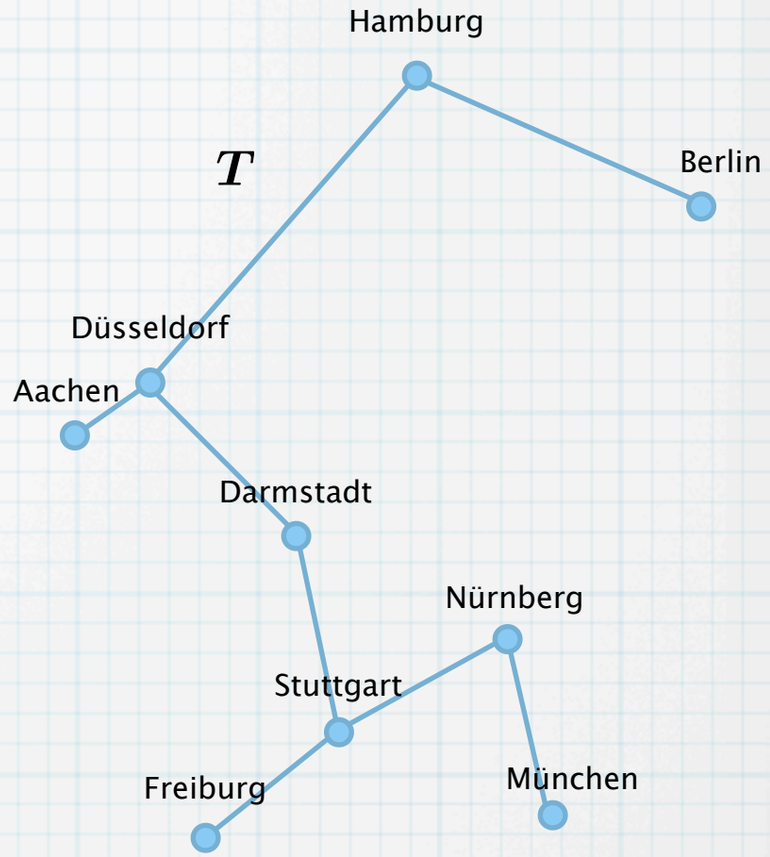
(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)

	Aa	*Be*	*Da*	*Dü*	*Fr*	*Ha*	*Mü*	*Nü*	*St*
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
Dü	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
Mü	64	60	40	63	37	80	0	17	22
Nü	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)

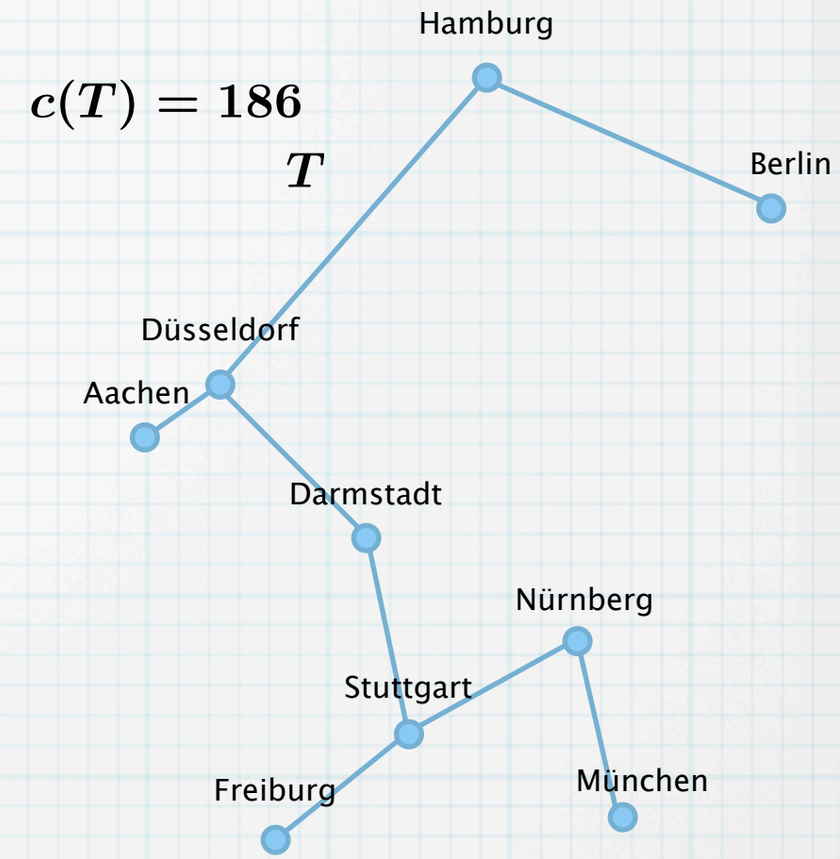


	Aa	*Be*	*Da*	*Dü*	*Fr*	*Ha*	*Mü*	*Nü*	*St*
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
Dü	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
Mü	64	60	40	63	37	80	0	17	22
Nü	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

Der Algorithmus von Christofides (1976)

- Eingabe: Instanz (K_n, c) für mTSP
- Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)

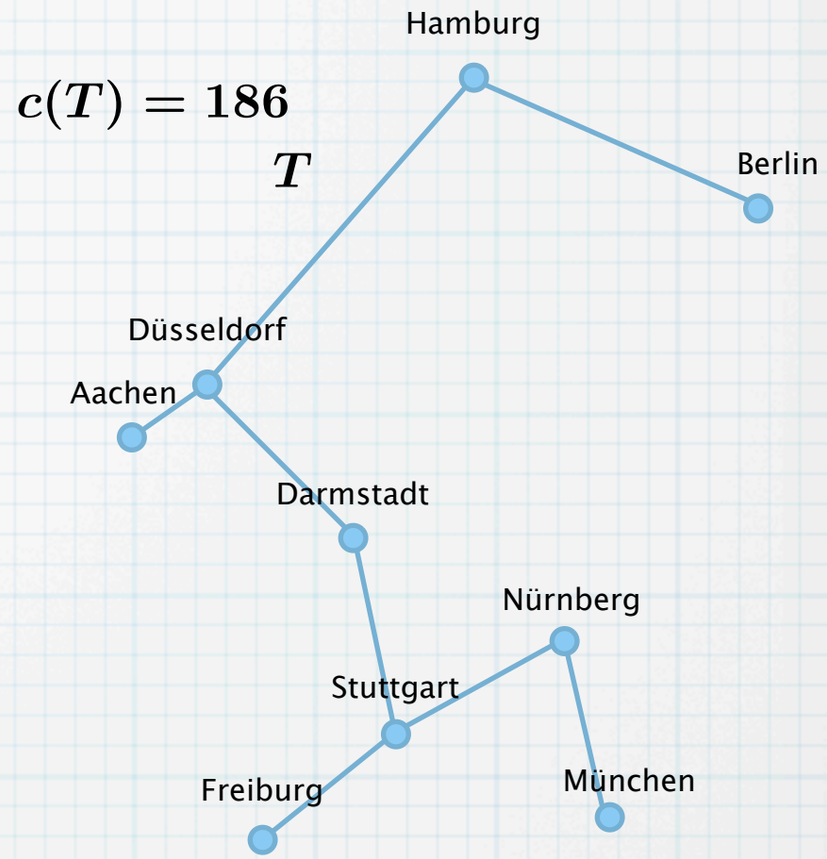


	Aa	*Be*	*Da*	*Dü*	*Fr*	*Ha*	*Mü*	*Nü*	*St*
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
Dü	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
Mü	64	60	40	63	37	80	0	17	22
Nü	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T

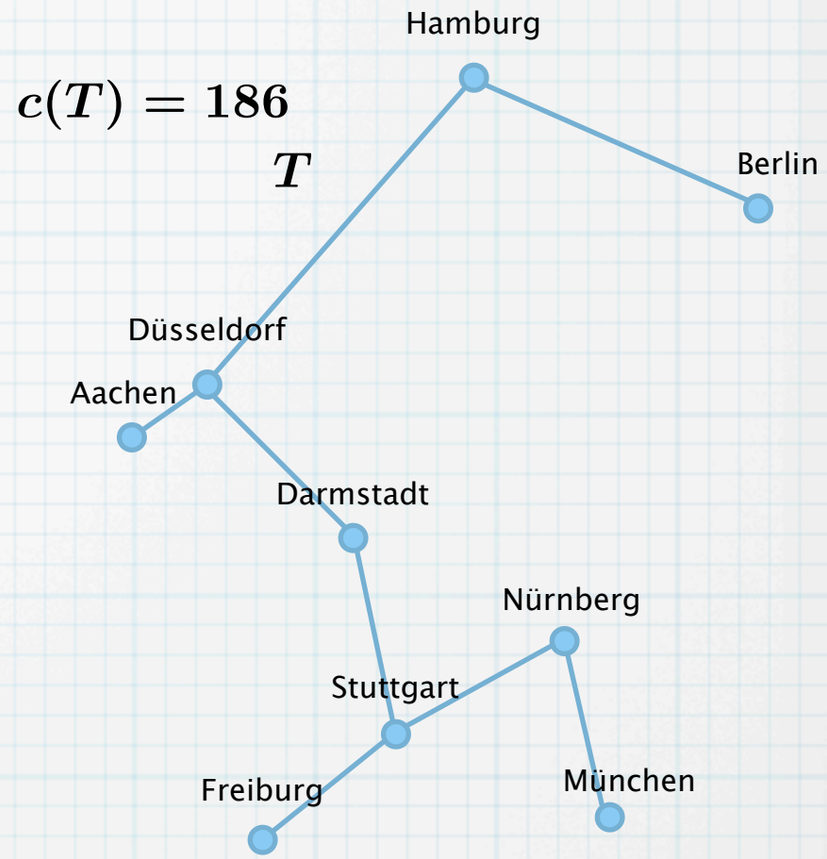


	Aa	*Be*	*Da*	*Dü*	*Fr*	*Ha*	*Mü*	*Nü*	*St*
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
Dü	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
Mü	64	60	40	63	37	80	0	17	22
Nü	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T



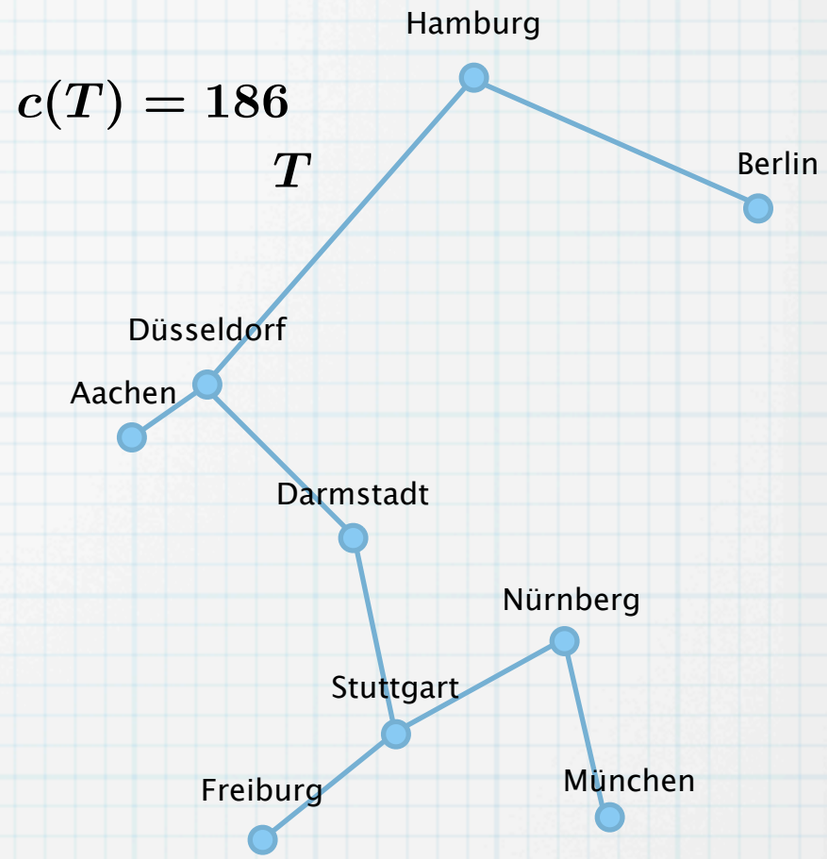
$U = \{Aa, Be, Dü, Fr, Mü, St\}$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T



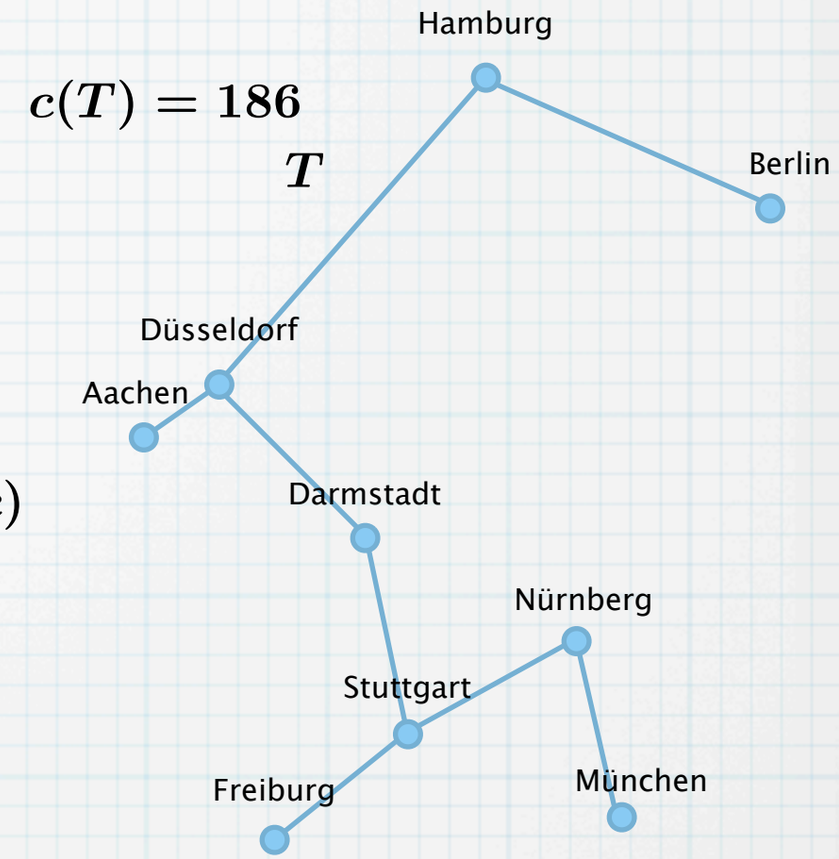
$U = \{Aa, Be, Dü, Fr, Mü, St\}$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

Der Algorithmus von Christofides (1976)

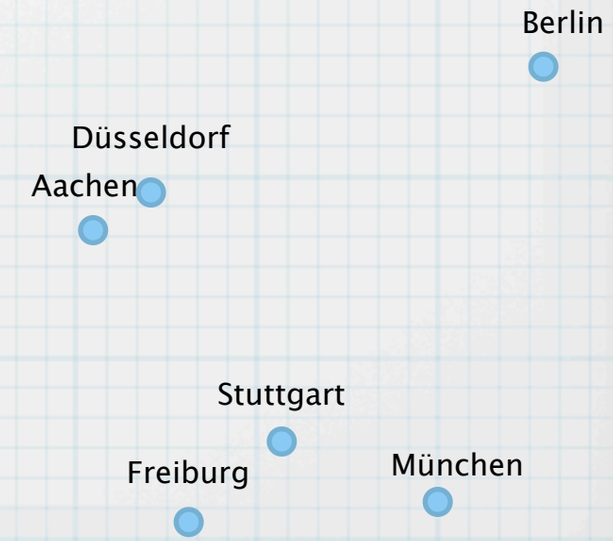
* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)



$U = \{Aa, Be, Dü, Fr, Mü, St\}$

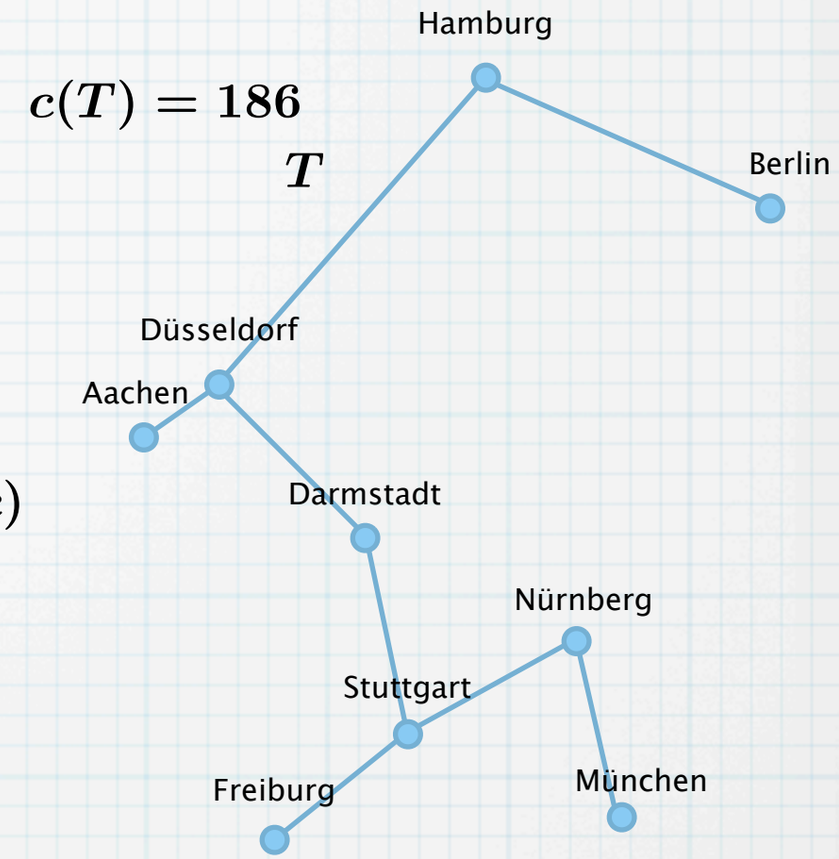
	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0



Der Algorithmus von Christofides (1976)

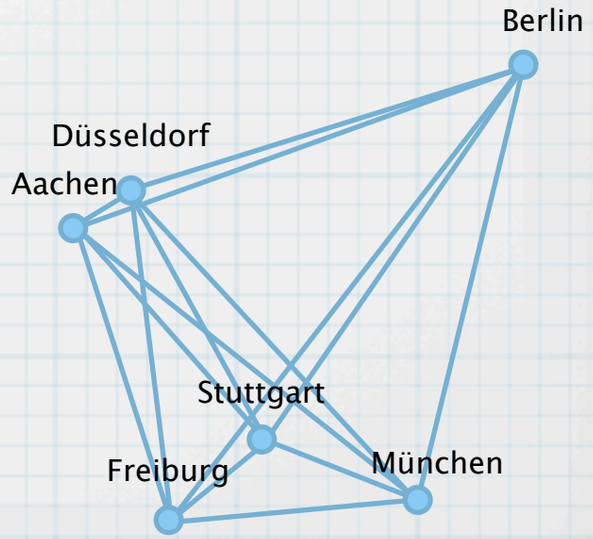
* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)



$$U = \{Aa, Be, Dü, Fr, Mü, St\}$$

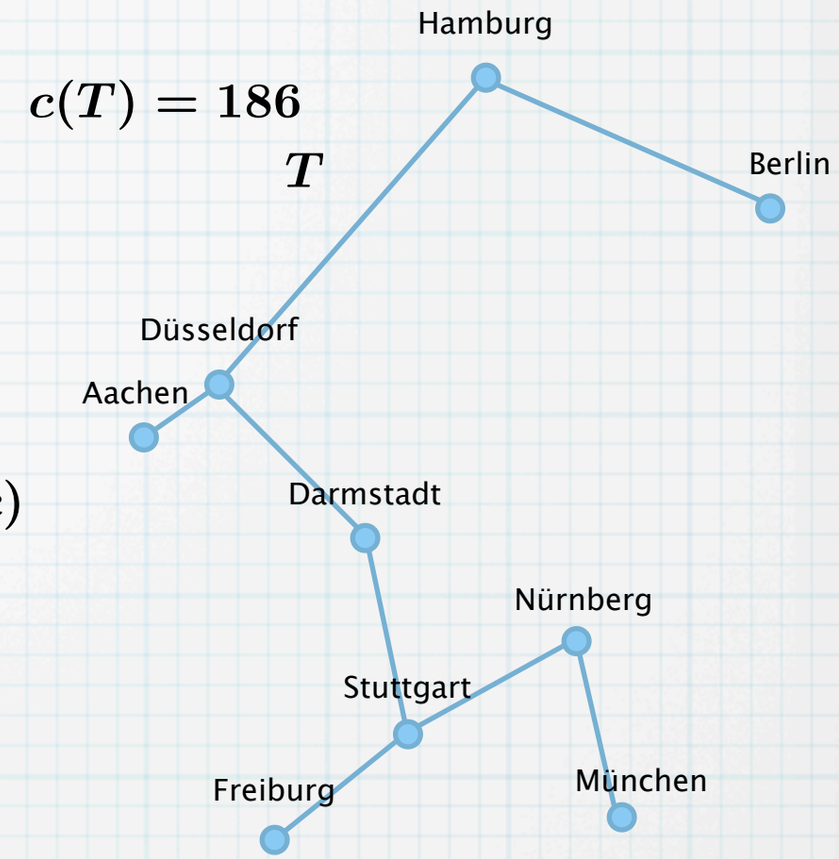
	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

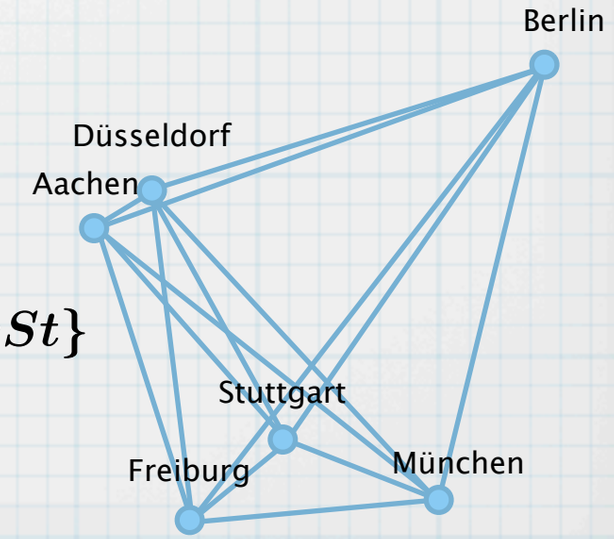
(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)



$$U = \{Aa, Be, Dü, Fr, Mü, St\}$$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

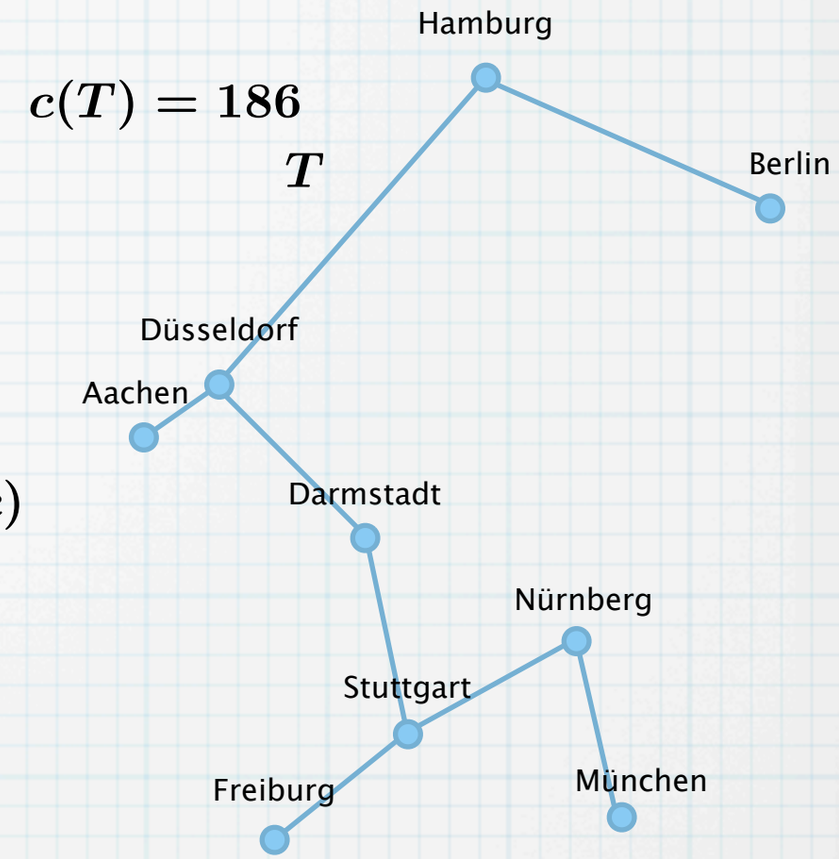
$$M^* = \{AaDü, BeMü, FrSt\}$$



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

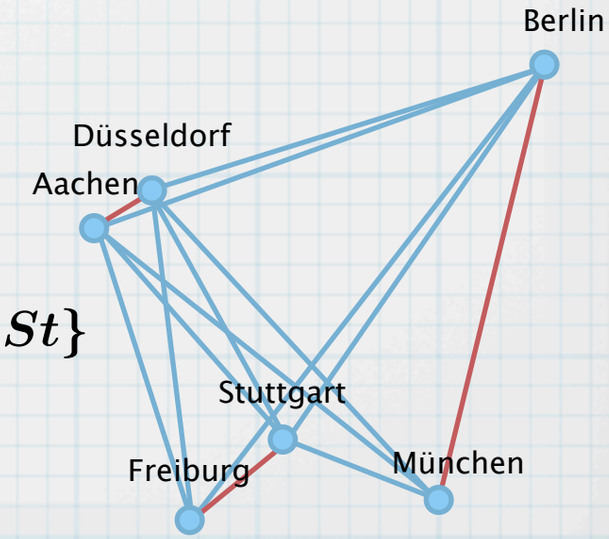
(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)



$$U = \{Aa, Be, Dü, Fr, Mü, St\}$$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

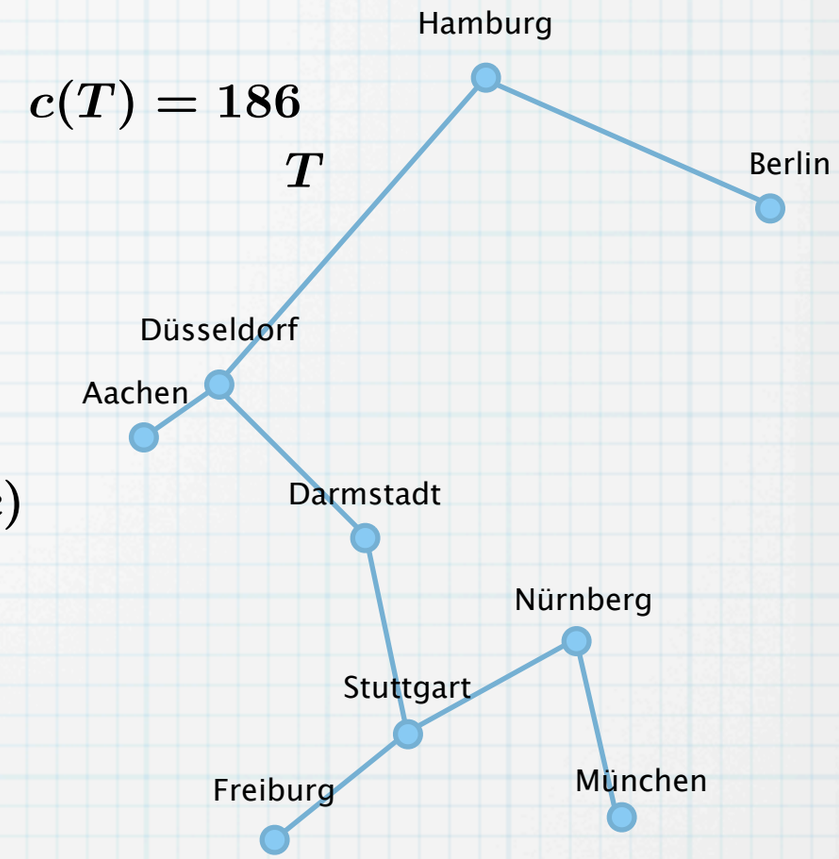
$$M^* = \{AaDü, BeMü, FrSt\}$$



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

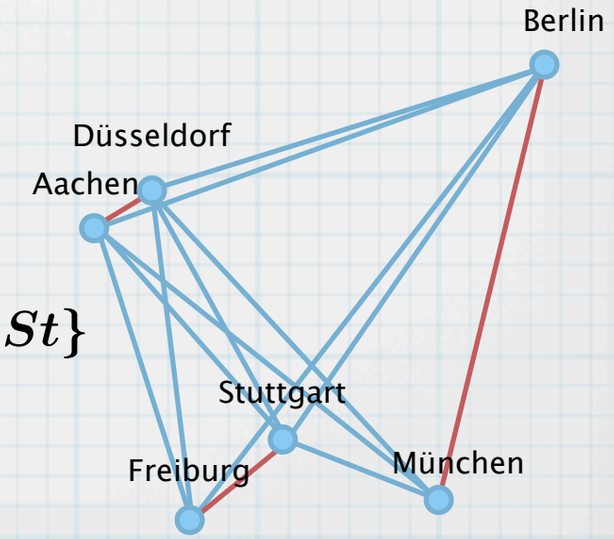
(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)



$$U = \{Aa, Be, Dü, Fr, Mü, St\}$$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

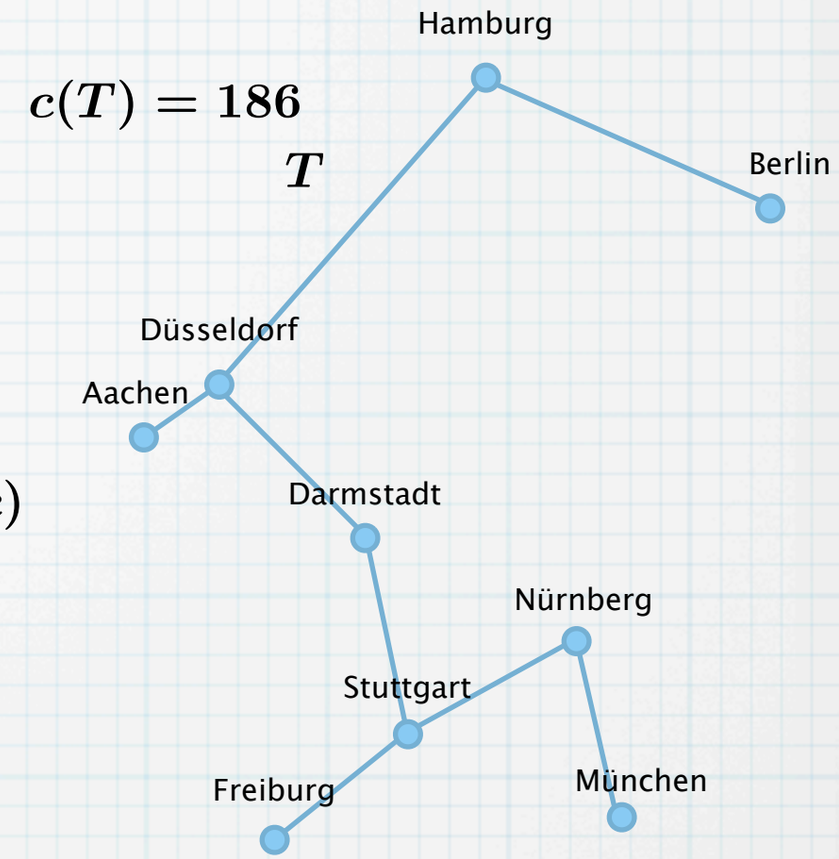
$$M^* = \{AaDü, BeMü, FrSt\}$$
$$c(M^*) = 95$$



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)
(5) sei G der eulersche Multigraph, der aus T durch Hinzufügen der Kanten von M^* entsteht.



$c(T) = 186$

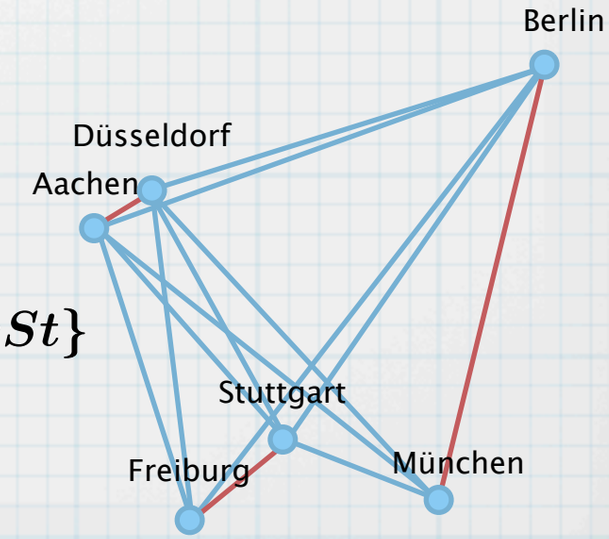
T

$U = \{Aa, Be, Dü, Fr, Mü, St\}$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

$M^* = \{AaDü, BeMü, FrSt\}$

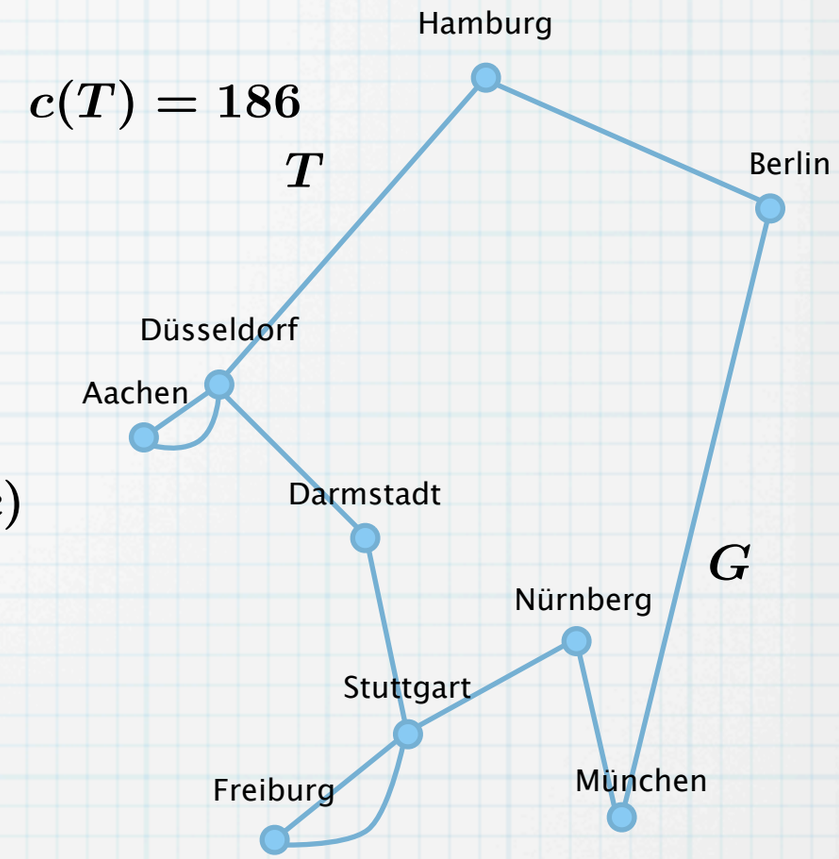
$c(M^*) = 95$



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

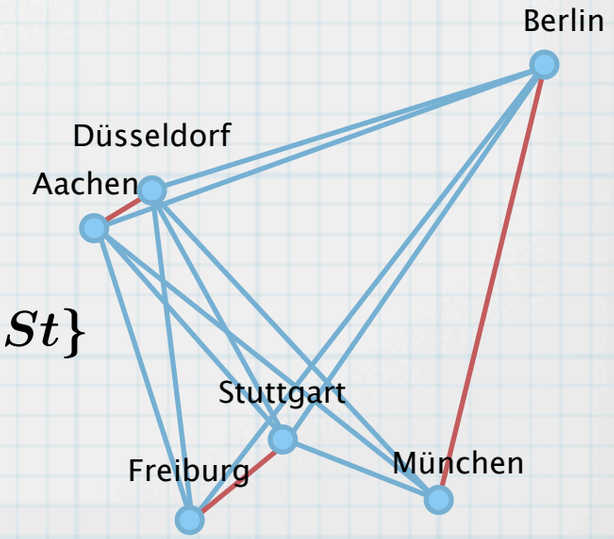
(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)
(5) sei G der eulersche Multigraph, der aus T durch Hinzufügen der Kanten von M^* entsteht.



$$U = \{Aa, Be, Dü, Fr, Mü, St\}$$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

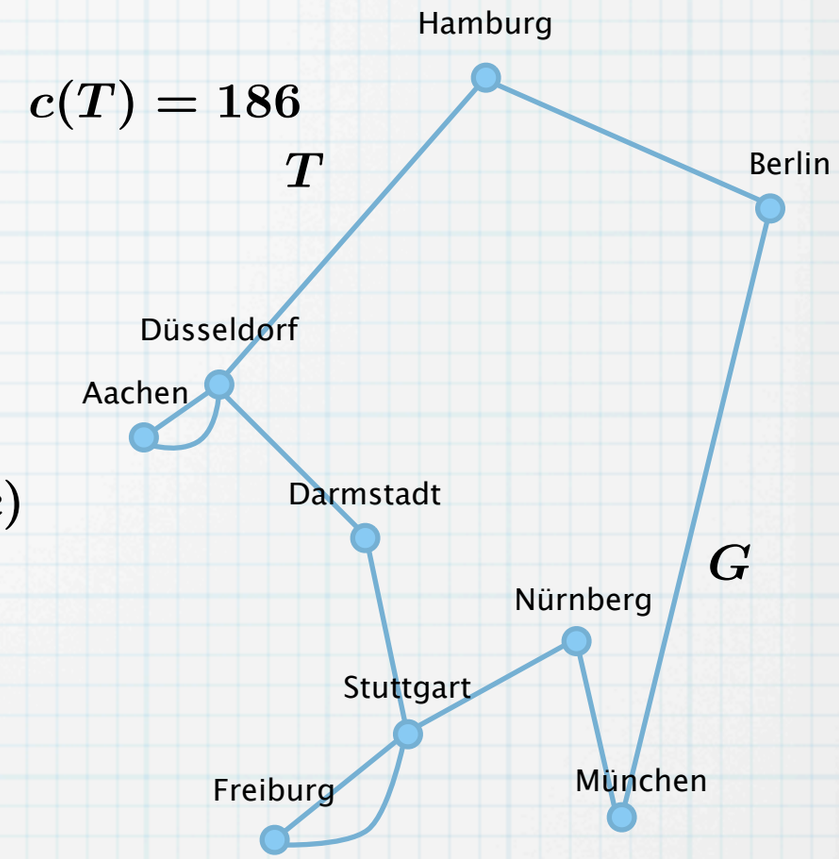
$$M^* = \{AaDü, BeMü, FrSt\}$$
$$c(M^*) = 95$$



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)
(5) sei G der eulersche Multigraph, der aus T durch Hinzufügen der Kanten von M^* entsteht.
(6) bestimme eine Eulertour K für G



$c(T) = 186$

T

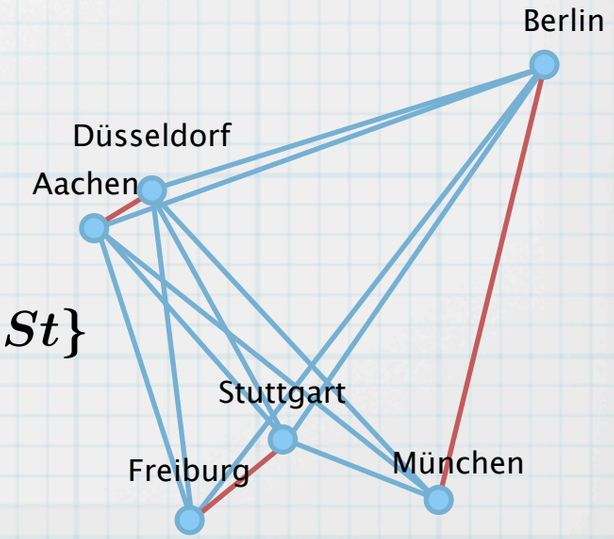
G

$U = \{Aa, Be, Dü, Fr, Mü, St\}$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

$M^* = \{AaDü, BeMü, FrSt\}$

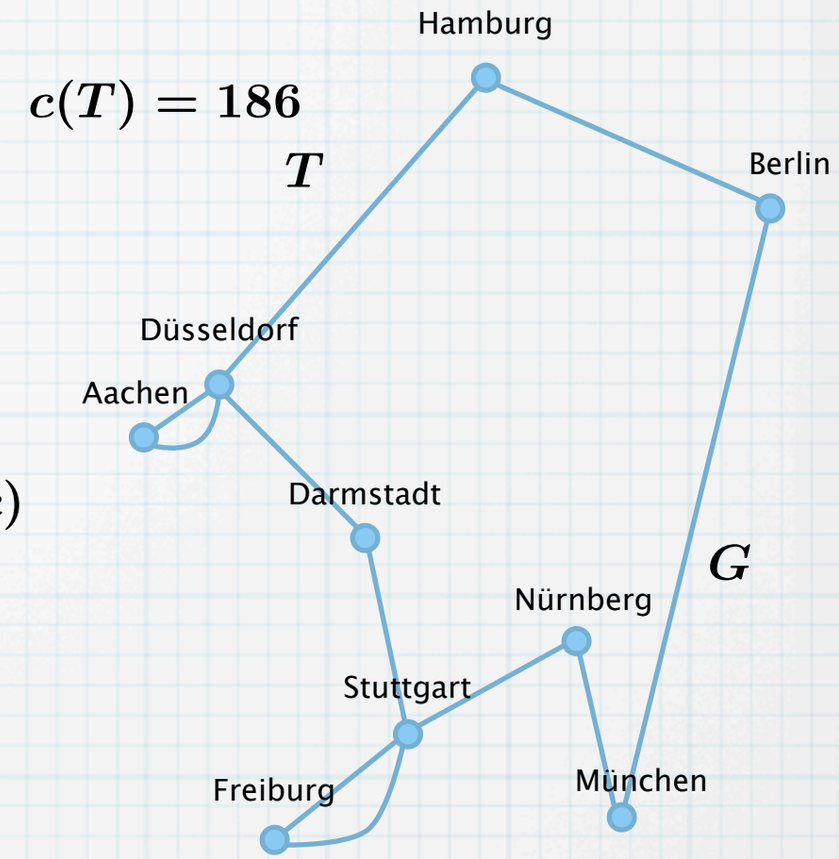
$c(M^*) = 95$



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)
(5) sei G der eulersche Multigraph, der aus T durch Hinzufügen der Kanten von M^* entsteht.
(6) bestimme eine Eulertour K für G



$c(T) = 186$

T

G

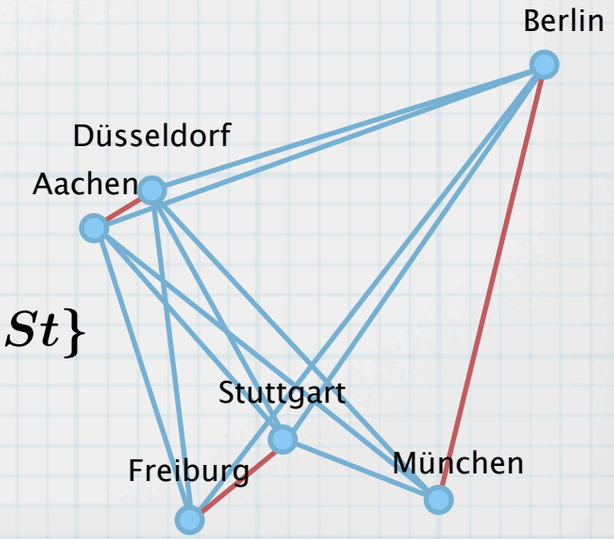
$U = \{Aa, Be, Dü, Fr, Mü, St\}$

$K = (Be, Mü, Nü, St, Fr, St, Da, Dü, Aa, Dü, Ha, Be)$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

$M^* = \{AaDü, BeMü, FrSt\}$

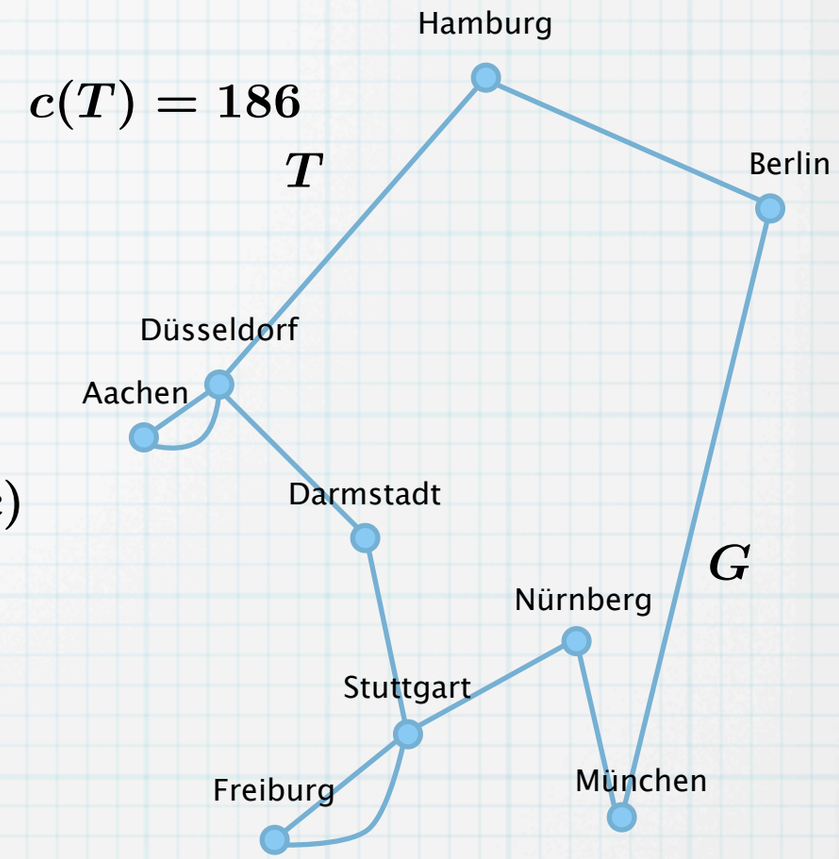
$c(M^*) = 95$



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)
(5) sei G der eulersche Multigraph, der aus T durch Hinzufügen der Kanten von M^* entsteht.
(6) bestimme eine Eulertour K für G
(7) wähle eine in K enthaltene Tour C



$c(T) = 186$

T

G

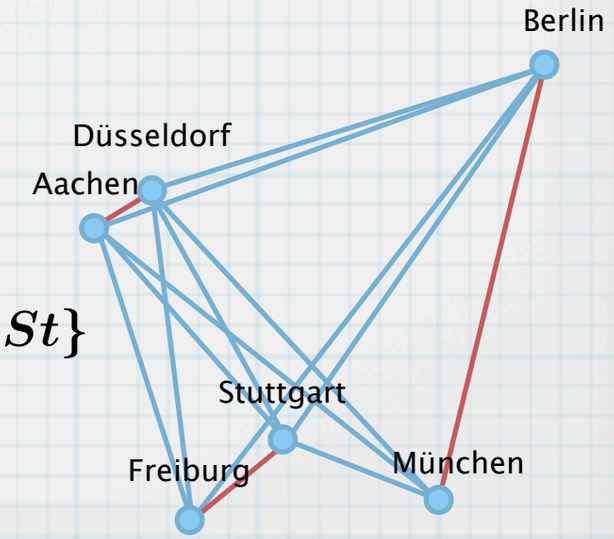
$$U = \{Aa, Be, Dü, Fr, Mü, St\}$$

$$K = (Be, Mü, Nü, St, Fr, St, Da, Dü, Aa, Dü, Ha, Be)$$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

$$M^* = \{AaDü, BeMü, FrSt\}$$

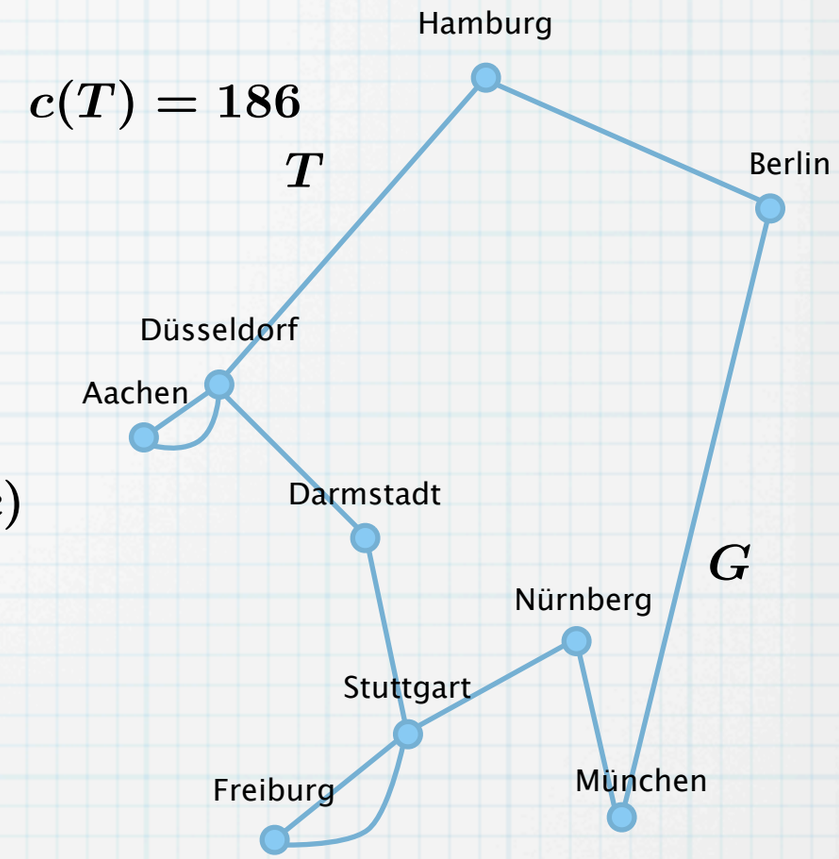
$$c(M^*) = 95$$



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)
(5) sei G der eulersche Multigraph, der aus T durch Hinzufügen der Kanten von M^* entsteht.
(6) bestimme eine Eulertour K für G
(7) wähle eine in K enthaltene Tour C



$c(T) = 186$

T

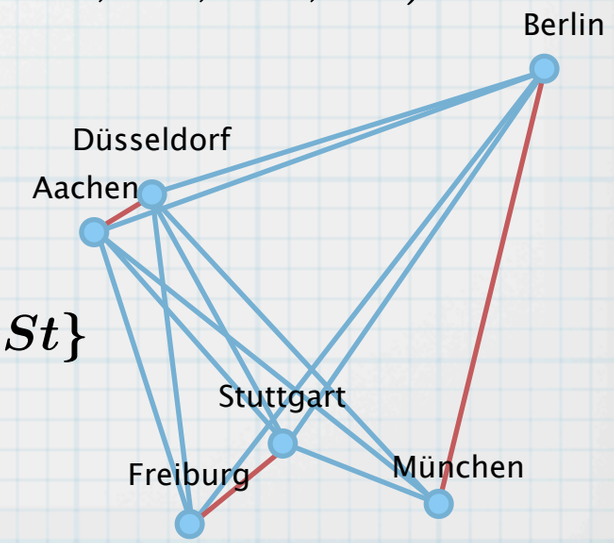
G

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

$U = \{Aa, Be, Dü, Fr, Mü, St\}$

$K = (Be, Mü, Nü, St, Fr, St, Da, Dü, Aa, Dü, Ha, Be)$

$C = (Be, Mü, Nü, St, Fr, Da, Dü, Aa, Ha, Be)$



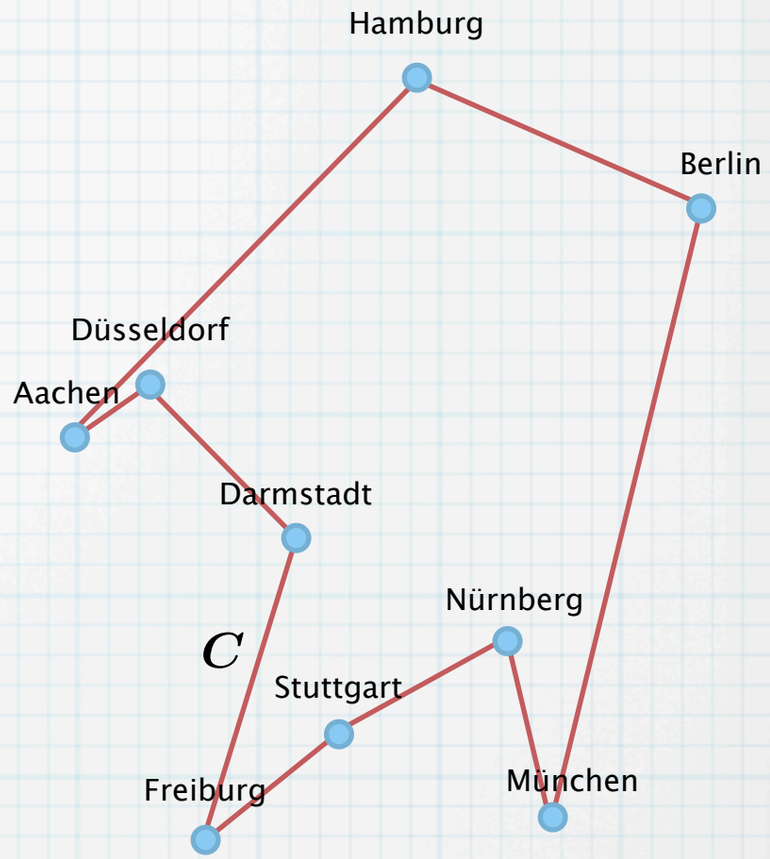
$M^* = \{AaDü, BeMü, FrSt\}$

$c(M^*) = 95$

Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)
(5) sei G der eulersche Multigraph, der aus T durch Hinzufügen der Kanten von M^* entsteht.
(6) bestimme eine Eulertour K für G
(7) wähle eine in K enthaltene Tour C



$$U = \{Aa, Be, Dü, Fr, Mü, St\}$$

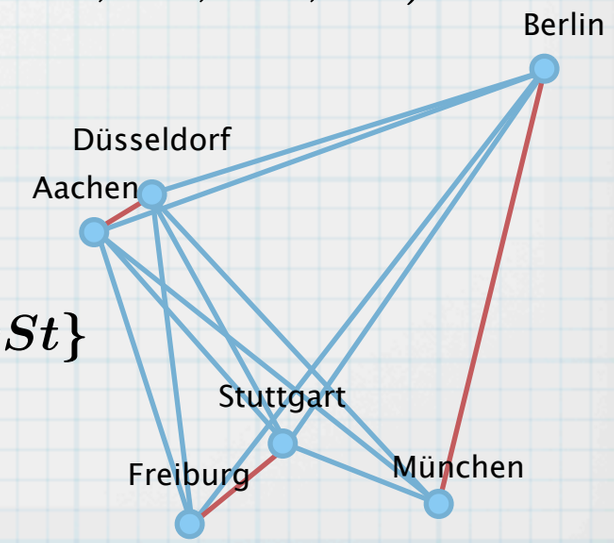
$$K = (Be, Mü, Nü, St, Fr, St, Da, Dü, Aa, Dü, Ha, Be)$$

$$C = (Be, Mü, Nü, St, Fr, Da, Dü, Aa, Ha, Be)$$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

$$M^* = \{AaDü, BeMü, FrSt\}$$

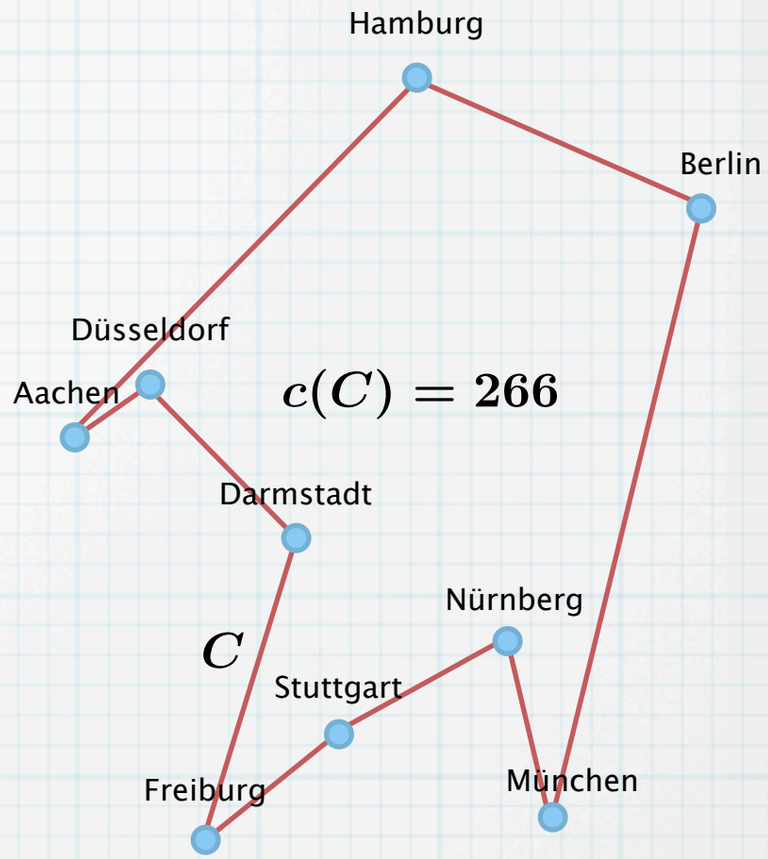
$$c(M^*) = 95$$



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)
(5) sei G der eulersche Multigraph, der aus T durch Hinzufügen der Kanten von M^* entsteht.
(6) bestimme eine Eulertour K für G
(7) wähle eine in K enthaltene Tour C



$$U = \{Aa, Be, Dü, Fr, Mü, St\}$$

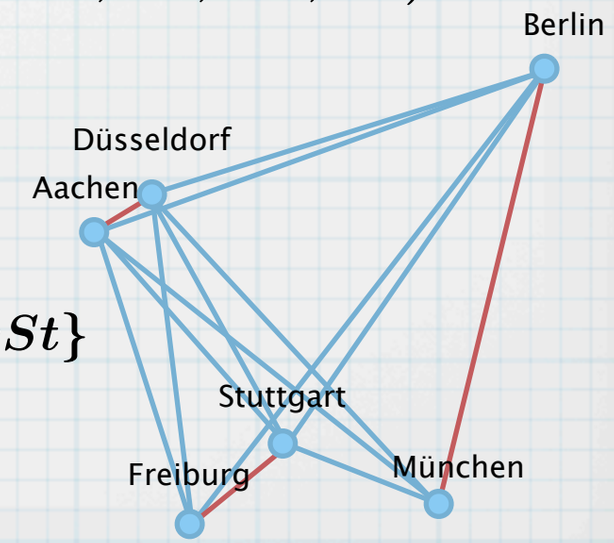
$$K = (Be, Mü, Nü, St, Fr, St, Da, Dü, Aa, Dü, Ha, Be)$$

$$C = (Be, Mü, Nü, St, Fr, Da, Dü, Aa, Ha, Be)$$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

$$M^* = \{AaDü, BeMü, FrSt\}$$

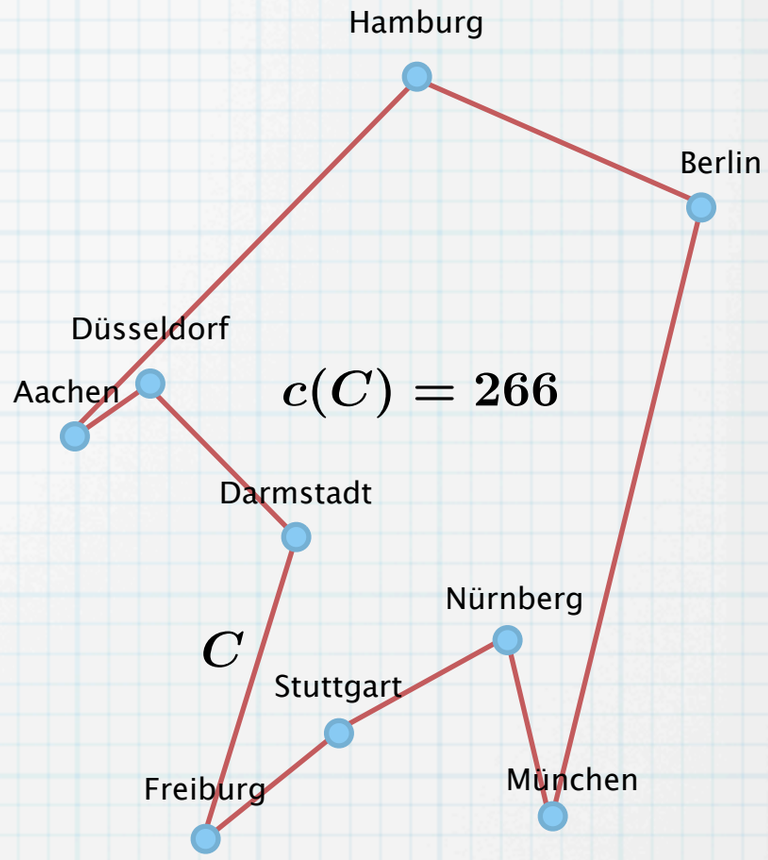
$$c(M^*) = 95$$



Der Algorithmus von Christofides (1976)

* Eingabe: Instanz (K_n, c) für mTSP
* Ausgabe: 1/2-approximative Tour C

(1) **algorithm** christofides
(2) bestimme minimalen Spannbaum T für (K_n, c)
(3) sei U die Menge der Knoten mit ungeradem Grad in T
(4) bestimme ein minimales perfektes Matching M^* in (K_U, c)
(5) sei G der eulersche Multigraph, der aus T durch Hinzufügen der Kanten von M^* entsteht.
(6) bestimme eine Eulertour K für G
(7) wähle eine in K enthaltene Tour C
(8) **end algorithm**



$c(C) = 266$

C

$$U = \{Aa, Be, Dü, Fr, Mü, St\}$$

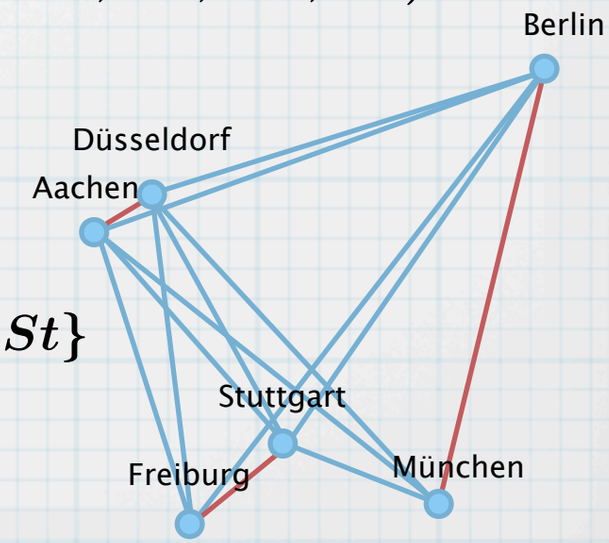
$$K = (Be, Mü, Nü, St, Fr, St, Da, Dü, Aa, Dü, Ha, Be)$$

$$C = (Be, Mü, Nü, St, Fr, Da, Dü, Aa, Ha, Be)$$

	Aa	Be	Da	$Dü$	Fr	Ha	$Mü$	$Nü$	St
Aa	0	64	26	8	57	49	64	47	46
Be	64	0	56	57	88	29	60	44	63
Da	26	56	0	23	34	50	40	22	20
$Dü$	8	57	23	0	54	43	63	44	41
Fr	57	88	34	54	0	83	37	43	27
Ha	49	29	50	43	83	0	80	63	70
$Mü$	64	60	40	63	37	80	0	17	22
$Nü$	47	44	22	44	43	63	17	0	19
St	46	63	20	41	27	70	22	19	0

$$M^* = \{AaDü, BeMü, FrSt\}$$

$$c(M^*) = 95$$



Beweis der Approximationsgüte

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E)$

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

  U aus Schritt (3) hat eine gerade Anzahl von Knoten, $|U| = 2m$.

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

  U aus Schritt (3) hat eine gerade Anzahl von Knoten, $|U| = 2m$.

  Sei $(i_1, i_2, \ldots, i_{2m})$ die Reihenfolge, in der die Knoten von U in C^* besucht werden.

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

  U aus Schritt (3) hat eine gerade Anzahl von Knoten, $|U| = 2m$.

  Sei $(i_1, i_2, \ldots, i_{2m})$ die Reihenfolge, in der die Knoten von U in C^* besucht werden.

  Wir definieren zwei perfekte Matchings M_1, M_2 von K_U durch
  $M_1 := \{i_1 i_2, i_3 i_4, \ldots, i_{2m-1} i_{2m}\}, M_2 := \{i_2 i_3, i_4 i_5, \ldots, i_{2m} i_1\}$.

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

  U aus Schritt (3) hat eine gerade Anzahl von Knoten, $|U| = 2m$.

  Sei $(i_1, i_2, \ldots, i_{2m})$ die Reihenfolge, in der die Knoten von U in C^* besucht werden.

  Wir definieren zwei perfekte Matchings M_1, M_2 von K_U durch
  $M_1 := \{i_1 i_2, i_3 i_4, \ldots, i_{2m-1} i_{2m}\}, M_2 := \{i_2 i_3, i_4 i_5, \ldots, i_{2m} i_1\}$.
  Da c die Dreicksungleichung erfüllt und M^* minimal ist, gilt
  $c(C^*) \geq c_{i_1 i_2} + c_{i_2 i_3} + c_{i_3 i_4} + \ldots + c_{i_{2m-1} i_{2m}} + c_{i_{2m} i_1}$

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

  U aus Schritt (3) hat eine gerade Anzahl von Knoten, $|U| = 2m$.

  Sei $(i_1, i_2, \ldots, i_{2m})$ die Reihenfolge, in der die Knoten von U in C^* besucht werden.

  Wir definieren zwei perfekte Matchings M_1, M_2 von K_U durch
  $M_1 := \{i_1 i_2, i_3 i_4, \ldots, i_{2m-1} i_{2m}\}, M_2 := \{i_2 i_3, i_4 i_5, \ldots, i_{2m} i_1\}$.
  Da c die Dreicksungleichung erfüllt und M^* minimal ist, gilt
  $c(C^*) \geq c_{i_1 i_2} + c_{i_2 i_3} + c_{i_3 i_4} + \ldots + c_{i_{2m-1} i_{2m}} + c_{i_{2m} i_1} = c(M_1) + c(M_2)$

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

  U aus Schritt (3) hat eine gerade Anzahl von Knoten, $|U| = 2m$.

  Sei $(i_1, i_2, \ldots, i_{2m})$ die Reihenfolge, in der die Knoten von U in C^* besucht werden.

  Wir definieren zwei perfekte Matchings M_1, M_2 von K_U durch
  $M_1 := \{i_1 i_2, i_3 i_4, \ldots, i_{2m-1} i_{2m}\}, M_2 := \{i_2 i_3, i_4 i_5, \ldots, i_{2m} i_1\}$.
  Da c die Dreicksungleichung erfüllt und M^* minimal ist, gilt
  $c(C^*) \geq c_{i_1 i_2} + c_{i_2 i_3} + c_{i_3 i_4} + \ldots + c_{i_{2m-1} i_{2m}} + c_{i_{2m} i_1} = c(M_1) + c(M_2) \geq 2c(M^*)$.

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

  U aus Schritt (3) hat eine gerade Anzahl von Knoten, $|U| = 2m$.

  Sei $(i_1, i_2, \ldots, i_{2m})$ die Reihenfolge, in der die Knoten von U in C^* besucht werden.

  Wir definieren zwei perfekte Matchings M_1, M_2 von K_U durch
  $M_1 := \{i_1 i_2, i_3 i_4, \ldots, i_{2m-1} i_{2m}\}, M_2 := \{i_2 i_3, i_4 i_5, \ldots, i_{2m} i_1\}$.
  Da c die Dreicksungleichung erfüllt und M^* minimal ist, gilt
  $c(C^*) \geq c_{i_1 i_2} + c_{i_2 i_3} + c_{i_3 i_4} + \ldots + c_{i_{2m-1} i_{2m}} + c_{i_{2m} i_1} = c(M_1) + c(M_2) \geq 2c(M^*)$.
  Also ist $c(M^*) \leq c(C^*)/2$.

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

  U aus Schritt (3) hat eine gerade Anzahl von Knoten, $|U| = 2m$.

  Sei $(i_1, i_2, \ldots, i_{2m})$ die Reihenfolge, in der die Knoten von U in C^* besucht werden.

  Wir definieren zwei perfekte Matchings M_1, M_2 von K_U durch
  $M_1 := \{i_1 i_2, i_3 i_4, \ldots, i_{2m-1} i_{2m}\}, M_2 := \{i_2 i_3, i_4 i_5, \ldots, i_{2m} i_1\}$.
  Da c die Dreicksungleichung erfüllt und M^* minimal ist, gilt
  $c(C^*) \geq c_{i_1 i_2} + c_{i_2 i_3} + c_{i_3 i_4} + \ldots + c_{i_{2m-1} i_{2m}} + c_{i_{2m} i_1} = c(M_1) + c(M_2) \geq 2c(M^*)$.
  Also ist $c(M^*) \leq c(C^*)/2$.
  Ferner ist $c(T) \leq c(C^*)$.

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

  U aus Schritt (3) hat eine gerade Anzahl von Knoten, $|U| = 2m$.

  Sei $(i_1, i_2, \ldots, i_{2m})$ die Reihenfolge, in der die Knoten von U in C^* besucht werden.

  Wir definieren zwei perfekte Matchings M_1, M_2 von K_U durch
  $M_1 := \{i_1 i_2, i_3 i_4, \ldots, i_{2m-1} i_{2m}\}, M_2 := \{i_2 i_3, i_4 i_5, \ldots, i_{2m} i_1\}$.
  Da c die Dreicksungleichung erfüllt und M^* minimal ist, gilt
  $c(C^*) \geq c_{i_1 i_2} + c_{i_2 i_3} + c_{i_3 i_4} + \ldots + c_{i_{2m-1} i_{2m}} + c_{i_{2m} i_1} = c(M_1) + c(M_2) \geq 2c(M^*)$.
  Also ist $c(M^*) \leq c(C^*)/2$.
  Ferner ist $c(T) \leq c(C^*)$.
  Somit gilt $c(C) \leq c(T) + c(M^*) \leq c(C^*) + c(C^*)/2 = 3/2 c(C^*)$.

Beweis der Approximationsgüte

* **Satz 19**:
  Der Algorithmus von Christofides ist 1/2-approximativ.
* Beweis:

  Sei C^* eine optimale Tour.

  Nach Lemma 30 / VL 6 gilt für die in Schritt (8) konstruierte Tour C die Abschätzung $c(C) \leq c(E) = c(T) + c(M^*)$.

  U aus Schritt (3) hat eine gerade Anzahl von Knoten, $|U| = 2m$.

  Sei $(i_1, i_2, \ldots, i_{2m})$ die Reihenfolge, in der die Knoten von U in C^* besucht werden.

  Wir definieren zwei perfekte Matchings M_1, M_2 von K_U durch
  $M_1 := \{i_1 i_2, i_3 i_4, \ldots, i_{2m-1} i_{2m}\}, M_2 := \{i_2 i_3, i_4 i_5, \ldots, i_{2m} i_1\}$.
  Da c die Dreicksungleichung erfüllt und M^* minimal ist, gilt
  $c(C^*) \geq c_{i_1 i_2} + c_{i_2 i_3} + c_{i_3 i_4} + \ldots + c_{i_{2m-1} i_{2m}} + c_{i_{2m} i_1} = c(M_1) + c(M_2) \geq 2c(M^*)$.
  Also ist $c(M^*) \leq c(C^*)/2$.
  Ferner ist $c(T) \leq c(C^*)$.
  Somit gilt $c(C) \leq c(T) + c(M^*) \leq c(C^*) + c(C^*)/2 = 3/2 c(C^*)$.
  Nach Definition ist der Algorithmus daher 1/2-approximativ.

Literaturquellen

Literaturquellen

* J. Clark, D.A. Holton, **Graphentheorie**, Spektrum Akademischer Verlag, Heidelberg, 1994. (Kapitel 4, Seite 133–172)

Literaturquellen

* J. Clark, D.A. Holton, **Graphentheorie**, Spektrum Akademischer Verlag, Heidelberg, 1994. (Kapitel 4, Seite 133–172)
* D. Jungnickel: **Graphen, Netzwerke und Algorithmen**, BI Wissenschaftsverlag, Mannheim, 1994. (Kapitel 12–14, Seite 435–570)

Literaturquellen

* J. Clark, D.A. Holton, **Graphentheorie**, Spektrum Akademischer Verlag, Heidelberg, 1994. (Kapitel 4, Seite 133–172)
* D. Jungnickel: **Graphen, Netzwerke und Algorithmen**, BI Wissenschaftsverlag, Mannheim, 1994. (Kapitel 12–14, Seite 435–570)
* B. Korte, J. Vygen: **Combinatorial Optimization – Theory and Algorithms**, 2nd Edition, Springer Verlag, Berlin, 2001. (Kapitel 10&11, Seite 205–260)