Algorithmische Diskrete Mathematik

Prof. Dr. Alexander Martin

SS 2008

Inhaltsverzeichnis

1 Endliche Mengen

Definition 1.1
Eine **Menge** ist eine Ansammlung verschiedener Objekte, **Elemente** genannt.

Beispiel:
Die Menge aller Haare auf menschlichen Köpfen
Die Menge aller Kanten eines Kantenstapels
Die Menge aller Atome im Weltall
Die Menge der reellen Zahlen

$\mathbb{N} = \{1, 2, 3, \ldots\}$ - natürliche Zahlen
$\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$
$\mathbb{Z} = \{0, 1, -1, 2, -2, \ldots\}$ - ganze Zahlen
$\mathbb{Q} = \left\{\frac{p}{q} : \; p, q \in \mathbb{Z}\right\}$ - rationale Zahlen
$\mathbb{R}$ - reelle Zahlen

Gibt man die Elemente einer Menge explizit an, so verwendet man geschweifte Klammern.

$$P = \{\text{Gustav}, \text{Gregor}, \text{Georg}, \text{Gunter}\}$$
$$M = \{1, 17, 39, 34, 18\}$$

1.1 Schreibweisen und Mengenoperationen

$\emptyset =$ leere Menge

Beachte den Unterschied $\emptyset$ versus $\{\emptyset\}$
: oder $|$ $\equiv$ mit der Eigenschaft
$\{x \in P |$ Name endet mit 'v'$\}$
$\{x \in \mathbb{Z} | \; x \geq 0\}$
$\in \equiv$ „ist ein Element von“
$\notin \equiv$ „ist nicht Element von“

Aufpassen „Barbier von Sevillia“
Schreibweise „|“ erlaubt nicht notwendige Weise eine Zuordnung oder nicht Zuordnung von Elementen zu Mengen.

Seien A, B, C Mengen. Dann bezeichnet

$$A \cap B = \{x \in A \text{ und } x \in B\}$$
$$A \cup B = \{x \in A \text{ oder } x \in B\}$$
$$A \setminus B = \{x \in A \text{ ohne } x \in B\}$$
$$A \subset B = A \text{ ist Teilmenge von } B$$
$$A = B \Leftrightarrow A \subset B \text{ und } B \subset A$$
$$|A| = \text{ Kardinalität von } A \text{ (Mächtigkeit von } A = \text{ die Anzahl Elemente in } A$$

Gilt $|A| < \infty$, so heisst A endlich.
Gilt $|A| = \infty$, so heisst A unendliche Menge.
Bei unendlichen Mengen wird noch weiter unterschieden in abzählbare, überabzählbare, ... Mengen.

$$A \Delta B = (A \setminus B) \cup (B \setminus A)$$

1.2 Rechenregeln

1.) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

Beweis:

I.) Sei $x \in A \cap (B \cup C) \Longrightarrow x \in A$ und $x \in B \cup C \Longrightarrow x \in A$ und $x \in B$ oder $x \in C$. Im ersten Fall ist damit $x \in A \cap B$, im zweiten Fall $x \in A \cap C$. Also insgesamt $x \in (A \cap B) \cup (A \cap C)$

II.) $x \in (A \cap B) \cup (A \cap C) \Longrightarrow$ a) $x \in A \cap B$ oder $x \in A \cap C \Longrightarrow x \in A$ und b.) $x \in B$ oder $x \in C \Longrightarrow x \in A \cap (B \cup C)$ □

Analogie: $\cap \equiv \cdot \quad \cup \equiv +$
1.) liest sich damit $A \cdot (B + C) = AB + BC$ (Distributivität)
weitere Regeln
2.) $A \cup B = B \cup A$, $A \cap B = B \cap A$ (Kommutativität)
3.) $(A \cup B) \cup C = A \cup (B \cup C)$, $(A \cap B) \cap C = A \cap (B \cap C)$ (Assoziativität)
4.) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ (Distributivität)
Achtung 4.) gilt nicht bei Zahlen $a + (b \cdot c) \neq (a + b)(b + c)$

1.3 Teilmengen und Anordnungen einer Menge

Satz 1.2
Eine n-elementige Menge besitzt 2^n Teilmengen.

Beweis:
Jedes Element ist entweder in der Menge enthalten, oder nicht. Es gibt also 2 Möglichkeiten pro Element und bei n Elementen insgesamt

$$\underbrace{2 \cdot 2 \cdot \ldots \cdot 2}_{n-\text{mal}} = 2^n$$

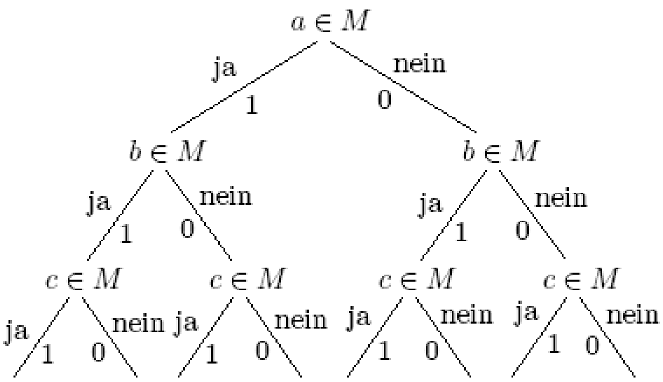
Beispiel:

M	Teilmengen von M
$\emptyset$	$\emptyset$
$\{a\}$	$\emptyset, \{a\}$
$\{a, b\}$	$\emptyset, \{a\}, \{b\}, \{a, b\}$
$\{a, b, c\}$	$\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}$

Darstellung über Binärzahlen mit n Stellen

$$\begin{array}{ccccc} \{a\} & \{b\} & \{c\} & & \\ 1 & 1 & 0 & \equiv & \{a, b\} \\ 0 & 0 & 0 & \equiv & \emptyset \\ 0 & 0 & 1 & \equiv & \{c\} \end{array}$$

Darstellung als Baum



$\{a, b\} \equiv 110$

Satz 1.3
Die Anzahl n-stelliger Strings, die aus k Elementen erzeugt werden können, beträgt k^n.

Beweis:
Für die erste Stelle gibt es k Möglichkeiten, für die Zweite $k \cdot k$, für die Dritte $k \cdot k \cdot k$, ..., für die n-te $\underbrace{k \cdot k \cdot k \cdot \ldots \cdot k}_{n\text{-mal}} = k^n$. □

Korollar 1.4
Stehen für Stelle i jeweils k_i Elemente zur Auswahl, so ist die Anzahl n-stelliger Strings $\prod_{i=0}^{n} k_i$.

1.4 Permutationen

Mengen sind an sich nicht geordnet, d.h.

$$\{a, b\} = \{b, a\}$$

Legt man Wert auf die Anordnung, d.h. die Reihenfolge, in der die Elemente aufgelistet werden, so spricht man von einer **Ordnung** oder geordneten Liste. Tauscht man die Reihenfolge so spricht man von einer **Permutation**.

Beispiel:
$M = \{1, 2, 3\}$
Permutationen: $123,\ 132,\ 213,\ 231,\ 231,\ 312,\ 321$

Satz 1.5
Die Anzahl Permutationen einer n-elementigen Menge beträgt $n!$.

Beweis:
Für das erste Element gibt es n mögliche Positionen, für das zweite $n-1$, für das dritte $n-2$, für das vierte $n-3$, ...,$3 \cdot 2 \cdot 1 = n!$ □

Möchte man nur k Elemente aus einer n-elementigen Menge anordnen, so ergibt sich

Korollar 1.6
Die Anzahl der geordnenten k-elementigen Teilmengen einer Menge mit n Elementen beträgt

$$n \cdot (n-1) \cdot \ldots \cdot (n-k+1) = \frac{n!}{(n-k)!}$$

Verzichtet man auf die Reihenfolge, so ergibt sich mit Satz 1.5

Satz 1.7
Die Anzahl der k-Teilmengen einer n-elementigen Menge ist $\frac{n!}{(n-k)! \cdot k!}$.

Bezeichnung

$$\binom{n}{k} := \frac{n!}{(n-k)! \cdot k!}$$

sprich „n über k“ oder „k aus n“ heißt **Binomialkoeffizient**.

Es gelten direkt folgende Bezeichnungen.

Satz 1.8

a.) $\binom{n}{k} = \binom{n}{n-k}$

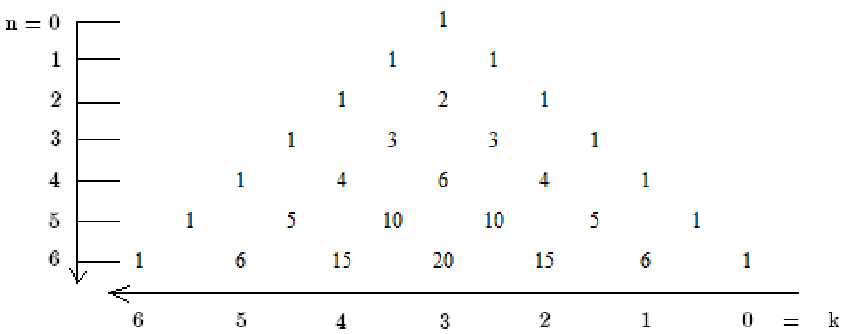
b.) $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$

Beweis:
a.) klar
b.) Eine k-elementige Teilmenge enthält entweder das n-te Element oder nicht. Im ersten Fall müssen aus $(n-1)$ Elementen noch $(k-1)$ ausgewählt werden, es gibt also $\binom{n-1}{k-1}$ Möglichkeiten, im zweiten Fall aus $(n-1)$ Elementen k, also $\binom{n-1}{k}$. Insgesamt also

$$\binom{n-1}{k} + \binom{n-1}{k-1} = \binom{n}{k}$$

Alternativ nachrechnen! □

1.5 Das Pascalsche Dreieck



Da eine Teilmenge einer n-elementigen Menge entweder $k = 0, 1, 2, 3, \ldots$ oder n Elemente enthält, ergibt sich mit Satz 1.2

Satz 1.9

$\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{n-1} + \binom{n}{n} = 2^n$

In Kurzschreibweise:

$$\boxed{\sum_{k=0}^{n} \binom{n}{k} = 2^n}$$

Exkurs Grundbegriffe der Aussagenlogik

Definition:
Eine Aussage ist ein Ausdruck, der WAHR oder FALSCH sein kann.

Beispiel:
„Der Ball ist rund“
„Das Papier ist weiß“

Aussagentafeln und Verknüpfungen:

- $\wedge$ und
- $\vee$ oder
- $\neg$ Verneinung
- $A \Rightarrow B$ „aus A folgt B“
- $A \Leftrightarrow B$ „A ist äquivalent zu B“ ($\equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$)

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \dot{\vee} B$	$A \Rightarrow B$	$\neg(A \Rightarrow B)$	$A \wedge \neg B$
0	0	0	0	1	0	1	0	0
0	1	0	1	1	1	1	0	0
1	0	0	1	0	1	0	1	1
1	1	1	1	0	0	1	0	0

2 Kombinatorische Werkzeuge

2.1 Vollständige Induktion

Behauptung
Es gilt eine Aussage $A(n)$ für alle $n \in \mathbb{N}$.

Beweis:

I.) Induktionsanfang beweisen!
$A(1)$ ist richtig

II.) Induktionsschritt beweisen!
Wenn $A(n-1)$ richtig ist, dann auch $A(n)$ für alle $n \in \mathbb{N}$.

Aus I.) und II.) folgt die Behauptung, denn

I.) $\Longrightarrow A(1)$ richtig
$\overset{II.)}{\Longrightarrow} A(2)$ richtig
$\overset{II.)}{\Longrightarrow} A(3)$ richtig
$\Longrightarrow \ldots$ usw. $\square$

Satz 2.1

$$\sum_{i=0}^{n} q^i = \frac{1-q^{n+1}}{1-q} \qquad \text{für } q \neq 1$$

Beweis:
$\underline{n=1}$: Induktionsanfang

$$\sum_{i=0}^{1} q^i = 1+q$$
$$\frac{1-q^2}{1-q} = \frac{(1+q)\cdot(1-q)}{1-q} = 1+q$$

$\underline{(n-1) \to n}$

$$\sum_{i=1}^{n} q^i = q^n + \sum_{i=1}^{n-1} q^i \overset{\text{Ind.ann.}}{=} q^n + \frac{1-q^n}{1-q}$$
$$= \frac{q^n \cdot (1-q) + 1 - q^n}{1-q} = \frac{q^n - q^{n+1} + 1 - q^n}{1-q} = \frac{1-q^{n+1}}{1-q}$$

$\square$

Satz 2.2
$2^n > n^2 \qquad \forall\, n \in \mathbb{N},\, n \geq 5$

Beweis:
$\underline{n = 5}$:

$$2^5 = 32 > 25 = 5^2$$

$\underline{(n-1) \to n}$:

$$2^n = 2 \cdot 2^{n-1} \overset{\text{Ind.ann.}}{>} 2 \cdot (n-1)^2 = 2n^2 - 4n + 1 = n^2 + n^2 - 4n + 1$$
$$\overset{n^2 > 4n \text{ (für } n > 4)}{>} n^2 + 1 > n^2$$

$\square$

Satz 2.3 (Bernoulli-Ungleichung)
Für $h \in \mathbb{R}$, $h \geq -1$ und $n \in \mathbb{N}$ gilt

$$(1+h)^n \geq 1 + n \cdot h$$

Beweis:
$\underline{n = 1}$:

$$(1+h)^1 = 1 + h \geq 1 + 1 \cdot h$$

$\underline{(n-1) \to n}$:

$$(1+h)^n = (1+h)^{n-1} \cdot \underbrace{(1+h)}_{\geq 0} \overset{\text{Ind.ann.}}{\geq} (1 + (n-1) \cdot h) \cdot (1+h)$$
$$= 1 + h + n \cdot h - h + n \cdot h^2 - h^2$$
$$= 1 + n \cdot h + \underbrace{(n-1)h^2}_{\geq 0} \geq 1 + n \cdot h$$

$\square$

2.2 Fibonacci-Zahlen

Motivation: Kaninchen-Beispiel von Fibonacci.
Ein Bauer züchtet Kaninchen. Jedes Kaninchen gebärt ein Junges jeden Monat, nachdem es 2 Monate alt ist.
Wir nehmen an, Kaninchen sterben nicht und wir ignorieren männliche Kaninchen. Wieviele Kaninchen wird der Bauer im n-ten Monat haben, wenn er mit einem beginnt?

$1, 1, 2, 3, 5, 8, \ldots$

Die Anzahl neuer Kaninchen ist die Anzahl der mindestens 2 Monate alten Kaninchen, allgemein

$$F_{n+1} = F_n + F_{n-1}$$

Die Zahlen F_n werden Fibonacci-Zahlen genannt.

Satz 2.4

$$F_n = \frac{1}{\sqrt{5}} \cdot \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right)$$

Die Zahl $\tau = \frac{1+\sqrt{5}}{2}$ heißt **goldener Schnitt**.

Beweis: (Induktion)
<u>$n = 1$</u>:

$$\frac{1}{\sqrt{5}} \cdot \left(\left(\frac{1+\sqrt{5}}{2} \right)^1 - \left(\frac{1-\sqrt{5}}{2} \right)^1 \right) = 1$$

<u>$n = 2$</u>

$$\frac{1}{\sqrt{5}} \cdot \left(\left(\frac{1+\sqrt{5}}{2} \right)^2 - \left(\frac{1-\sqrt{5}}{2} \right)^2 \right) = 1$$

<u>$n \to (n+1)$:</u>

$$F_{n+1} = F_n + F_{n-1} \overset{\text{Ind.ann.}}{=} \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right) + \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n-1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}\right)$$

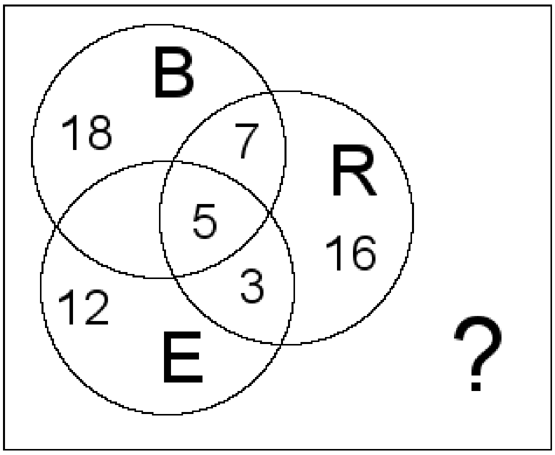
$$= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n-1} \cdot \underbrace{\left(\frac{1+\sqrt{5}}{2}+1\right)}_{=\frac{1+2\cdot\sqrt{5}+5}{4}=\left(\frac{1+\sqrt{5}}{2}\right)^2} - \frac{1}{\sqrt{5}} \cdot \left(\frac{1-\sqrt{5}}{2}\right)^{n-1} \cdot \underbrace{\left(\frac{1-\sqrt{5}}{2}+1\right)}_{=\left(\frac{1-\sqrt{5}}{2}\right)^2}$$

$$= \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right)$$

2.3 Das Inklusions-Exklusionsprinzip

Beispiel:
Schulklasse mit 40 Schülern mit unterschiedlichen Pop-Idolen:
Beatles, Rolling Stones und Elvis Presley

gemäß folgendem Bild



Wir wollen wissen wie viele Schüler gar keine Pop-Idole haben.

$$
\begin{array}{lccccccccc}
 & \text{0-Eigenschaften} & & \text{1-Eigenschaft} & & \text{2-Eigenschaft} & & \text{3-Eigenschaften} \\
40 = & x & + & (18+12+16) & - & (5+7+3) & + & 2 \\
40 \equiv & \binom{3}{0} & + & \binom{3}{1} & - & \binom{3}{2} & + & \binom{3}{3} \\
40 = & x & + & 46 & - & 15 & + & 2
\end{array}
$$

$\Longrightarrow x = 7$

Dieses Prinzip, das von einer Gesamtmenge die 1-elementigen (Elemente bezogen auf ihre Eigenschaften) Mengen abzieht, dann die 2-elementigen addiert, usw., nennt man **Inklusions-Exklusions-Prinzip**.

Satz 2.5

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

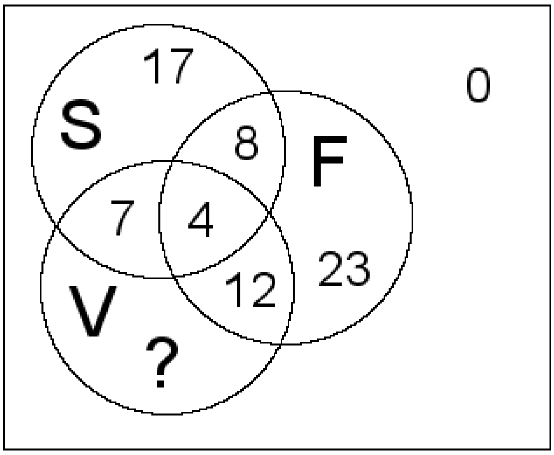
$$\boxed{\underbrace{\sum_{k=0}^{n} \binom{n}{k}}_{k \text{ gerade}} = \underbrace{\sum_{k=0}^{n} \binom{n}{k}}_{k \text{ ungerade}}}$$

Beweis:
Wir erinnern an die Darstellung von Teilmengen als Binärzahlen.

$$
\begin{aligned}
&\{a, b, c\} \\
&101 = \{a, c\} \\
&000 = \emptyset
\end{aligned}
$$

Jede k-elementige Teilmenge entspricht einem String der Länge n mit k Einsen. Da es genauso viele Strings mit einer geraden Anzahl von Einsen gibt wie mit einer ungeraden Anzahl, folgt die Behauptung. $\square$

Beispiel:
Schulklasse mit 40 Mädchen, die gerne Schach, Fußball und Volleyball spielen gemäß folgenden Interessen:



Wir interessieren uns für die Anzahl an Schülerinnen, die nur Volleyball spielen?

$$\begin{array}{rccccccccc}
 & \binom{3}{0} & + & \binom{3}{1} & - & \binom{3}{2} & + & \binom{3}{3} \\[2ex]
40 = & 0 & + & (V+17+23) & - & (8+7+12) & + & 4 \\[2ex]
40 = & V & + & 40 & - & 27 & + & 4
\end{array}$$

$\Longrightarrow V = 23$

2.4 Das Taubenschlag-Prinzip (engl. pigeon-hole principle oder auch Schubfachprinzip)

Beobachtung 2.6
Wenn n Objekte auf k Schachteln verteilt werden sollen mit $k < n$, dann gibt es eine Schachtel mit mindestens 2 Objekten.

Beispiel:

1.) Mindestens 2 Hessen haben gleich viele Haare auf dem Kopf

2.) 50 Schuss auf ein 70cm $\times$ 70cm Quadrat. Dann gibt es 2 Treffer, die höchstens 15cm Abstand haben.

3.) Betrachte Menge A mit $n+1$ Zahlen aus $\{1, 2, \ldots, 2n\}$. Dann gibt es 2 Zahlen, von denen die eine die andere teilt:

Beispiel:
$\underline{n = 7}$ $\quad \{1, 2, 3, \ldots, 14\}$
$2, 3, 5, 7, 11, 13$

Beweis:
Jedes $a \in A$ schreibe in der Form

$$a = 2^k \cdot m \quad k \in \mathbb{N}_0 \text{ und } m \text{ ungerade}$$

$2 = 2^1 \cdot 1 \quad 7 = 2^0 \cdot 7.$
In $\{1, 2, \ldots, 2n\}$ stecken n ungerade Zahlen.
$\overset{\text{Beobachtung 2.6}}{\Longrightarrow}$ Es muss 2 Zahlen aus A geben mit demselben m. Eine von diesen teilt die andere. $\square$

4.) Betrachte n Zahlen $a_1, \ldots, a_n$. Dann gibt es eine Reihe aufeinanderfolgende Zahlen $a_{k+1}, a_{k+2} \ldots, a_l$ mit $l > k$, deren Summe $\sum_{i=k+1}^{l} a_i$ durch n teilbar ist.
$\underline{n = 3}$ $\quad 2, 2, ?$
$\underline{n = 4}$ $\quad 2, 1, 2, ?$

Beweis:
$N = \{0, a_1, a_1 + s_2, a_1 + a_2 + a_3, \ldots, a_1 + a_2 + \ldots + a_n\}$
$|N| = n + 1$
Jedes $a \in N$ teile durch n mit Rest $\in R = \{0, 1, 2, \ldots, n - 1\}$
$|R| = n$
$\overset{\text{Beobachtung 2.6}}{\Longrightarrow}$ es gibt 2 Elemente aus H mit demselben Rest

$$a_1 + \ldots + a_k, \; a_1 + a_2 + \ldots + d_l \quad k < l$$

$$\Longrightarrow \quad \underbrace{\sum_{i=1}^{k} a_i - \sum_{i=1}^{k} a_i}_{\sum_{i=k+1}^{l} a_i \text{ ist durch } n \text{ teilbar}} \qquad \text{hat den Rest } 0$$
$\square$

5.) Geburtstagswette
„Von 50 haben 2 am gleichen Tag Geburtstag"

Beweis:
Ich verliere, wenn alle an verschiedenen Tagen Geburtstag haben. Die Wahrscheinlichkeit dafür ist

$$\frac{365 \cdot 364 \cdot 363 \cdot \ldots \cdot 316}{365 \cdot 365 \cdot \ldots \cdot 365} = \frac{\%}{365^{50}} \approx 0,0296\%$$

23 hätten gereicht um eine Gewinnwahrscheinlichkeit $> 50\%$ zu haben. $\square$

2.5 Binomialkoeffizienten

Satz 2.7 (Binomialsatz)

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} (x^k \cdot y^{n-k}) \qquad x, y \in \mathbb{R},\ n \in \mathbb{N}$$

Beispiel:
$n = 2$

$$\begin{aligned}(x+y)^2 &= x^2 + 2xy + y^2 \\ &= \binom{2}{0} x^0 y^2 + \binom{2}{1} x^1 y^1 + \binom{2}{2} x^2 y^0 \\ &= y^2 + 2xy + x^2\end{aligned}$$

<u>**Beweisskizze:**</u>

$$\begin{aligned}(x+y)^n &= \underbrace{(x+y) \cdot (x+y) \cdot \ldots \cdot (x+y)}_{n-\text{mal}} \\ &= \sum_{k=0}^{n} \binom{n}{k} x^k \cdot y^{n-k}\end{aligned}$$

Korollar 2.8 (Satz 1.9)

$$x = y = 1 \Longrightarrow 2^n = \sum_{k=0}^{n} \binom{n}{k}$$

Korollar 2.9 (Satz 2.5)

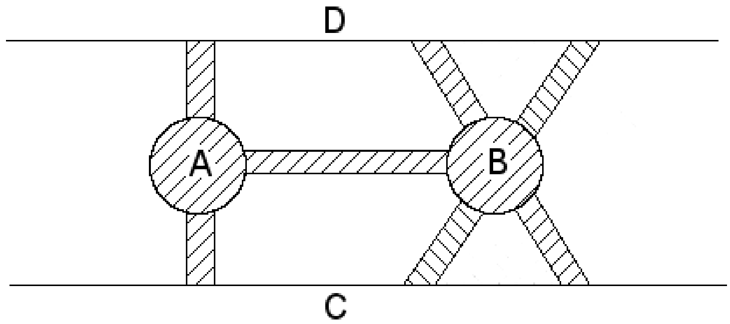
$$x = -1,\ y = 1 \Longrightarrow 0 = \sum_{k=0}^{n} (-1)^k \cdot \binom{n}{k}$$

3 Graphen-Grundlagen

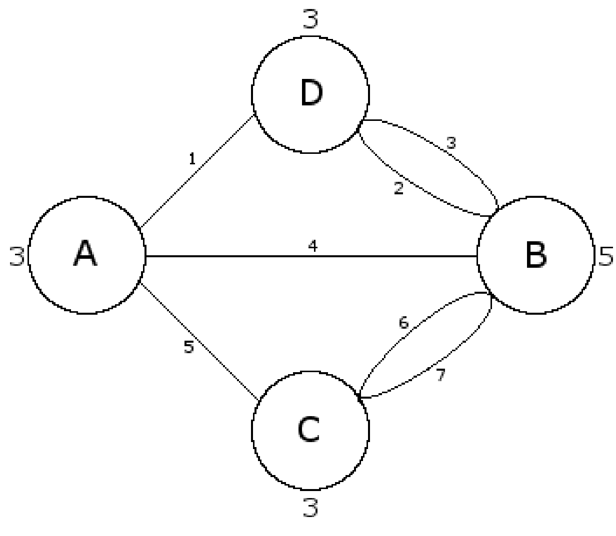
3.1 Grundlagen

Beispiel:

1.) Königsberger Brückenproblem (Euler, 1736)



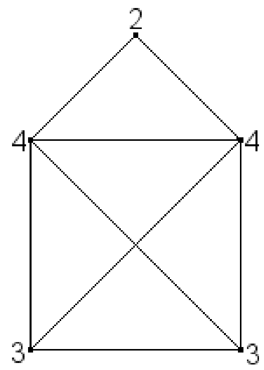
Frage: Gibt es eine Rundreise durch Königsberg, die jede Brücke genau einmal benutzt?



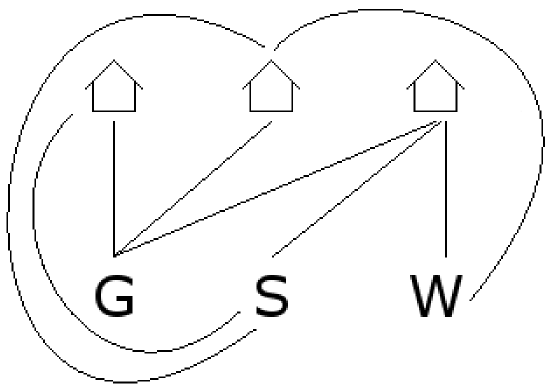
abstrahierte Darstellung des Brückenproblems

Nein, da mehr als 2 Knoten ungeraden Grad haben.

2.) Das Haus vom Nikolaus

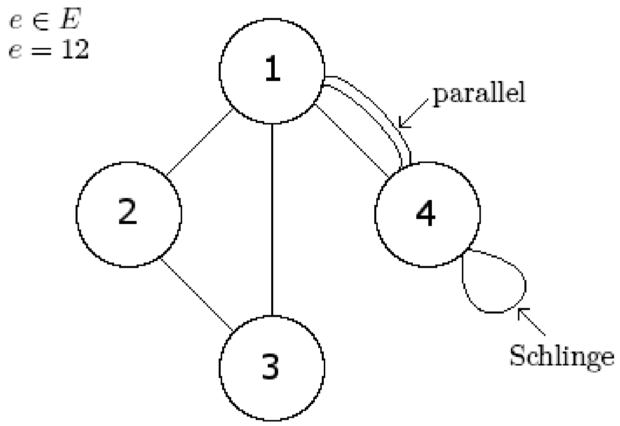


3.) Ein Grundversorgungsproblem



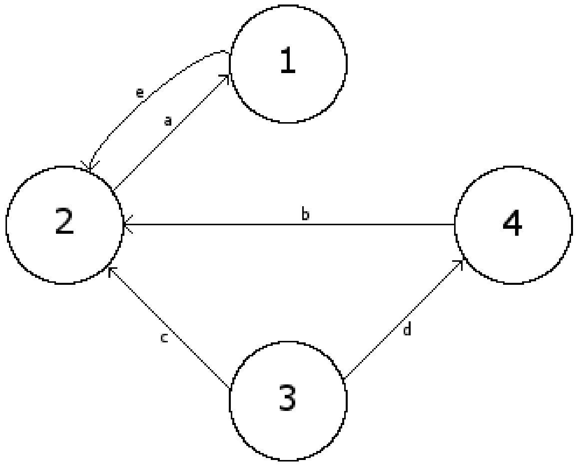
Definition 3.1
Ein ungerichteter Graph ist ein Tripel (V, E, Ψ) mit
V nicht-leere Menge, **Knoten**
E Menge, **Kanten**
$\Psi : E \to V \times V$
$V = \{A, B, C, D\}$, $E = \{1, 2, \ldots, 7\}$
$\Psi(1) = \{A, D\} = AD = [A, D]$
$\Psi(7) = \{B, C\} = BC = [B, C]$
$G = (V, E)$, Kante BC

G heisst **endlich**, falls $|V|$, $|E| < \infty$ andernfalls **unendlich**



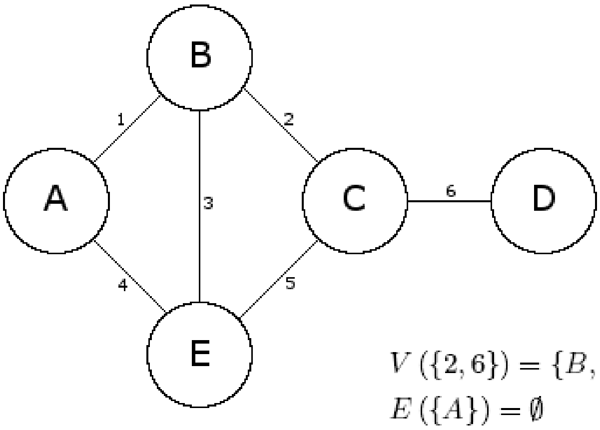
1 und 2 sind **inzident** (liegen auf) zu e
1 und 2 sind **adjazent** („benachbart" oder verbunden)
Ein Graph ohne Schlingen und parallele Kanten heisst **einfach**

Ein gerichteter Graph $D = (V, A, \Psi)$ ist ein Tripel mit $V \neq \emptyset$ und E Mengen und $\Psi : E \to V \times V$



$\Psi(a) = (2, 1)$
$\Psi(e) = (1, 2)$
$\Psi(a) \neq \Psi(e)$
$\Psi(a) = \Psi(e)$
sind gerichtet und
$\{2, 1\}$ und $\{1, 2\}$ sind ungerichtet.
A arc (engl.) Bogen
E edge (engl.) Kante
V vertex (engl.) Knoten

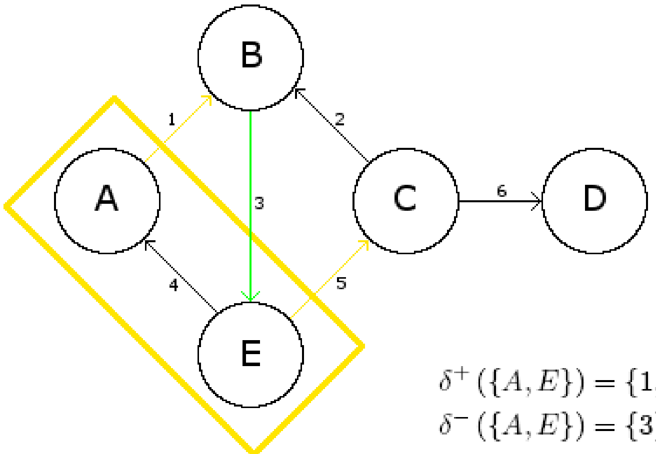
$F \subseteq E \quad V(F) =$ Menge aller Knoten inzident zu einem $f \in F$
$W \subseteq V \quad E(W) =$ Menge aller Kanten mit beiden Endknoten in W



$V(\{2,6\}) = \{B, C, D\}$
$E(\{A\}) = \emptyset$
$E(\{A, B, C\}) = \{1, 2\}$

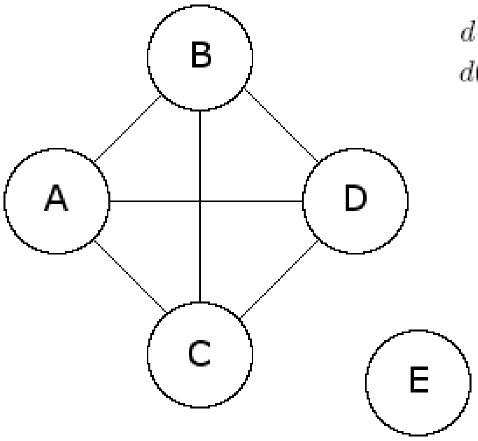
$U, W \subseteq V$
$\underbrace{\delta(U, W)}_{=[U:W]} =$ Menge aller Kanten mit einem Endknoten in U und einem in W.
$\delta(\{C\}, \{B, D\}) = \{2, 6\}$
$\delta(W, V \setminus W) = \delta(W)$
$\delta(\{A, E\}, \{B, C, D\}) = \{1, 3, 5\}$

Eine Kantenmenge $F \subseteq E$ für die es $W \subseteq V$ mit $\delta(W) = F$ gibt, heisst **(von W induzierter) Schnitt**



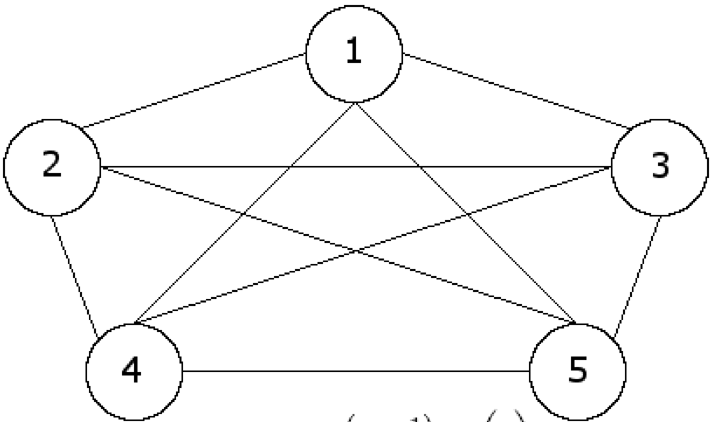
$\delta^+(\{A, E\}) = \{1, 5\}$
$\delta^-(\{A, E\}) = \{3\}$

$d(v) = |\delta(\{v\})|$ heisst **Grad** von $v \in V$



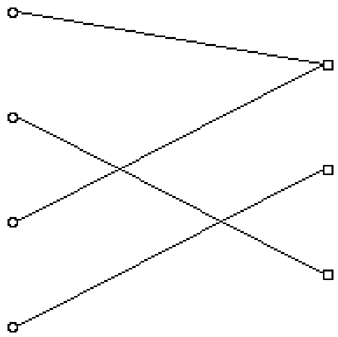
$d(A) = 3$
$d(E) = 0$

$d(v) = 0$ dann heisst v **isoliert**.
$G = (V, E)$ heisst **vollständig**, falls jeder Knoten mit jedem anderen verbunden ist.



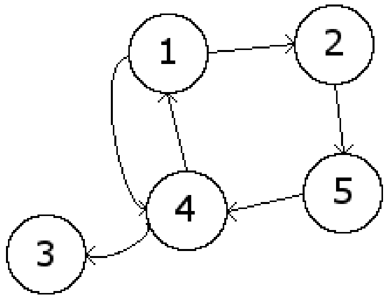
$$|E(K_n)| = \frac{n(n-1)}{2} = \binom{n}{2}$$

G heisst **bipartit**, falls $V = V_1 \odot V_2$ mit $E \subseteq [V_1 : V_2]$
$|V_1| = n \qquad n = |V_2| \qquad V_1 \cap V_2 = \emptyset$

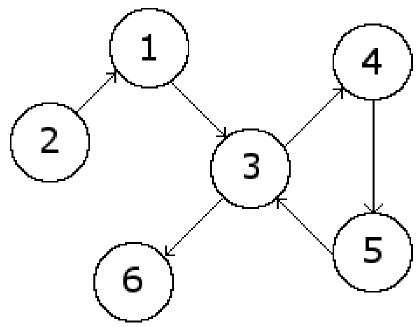


Ist E alle Kanten zwischen V_1 und V_2 so heisst er **vollständig** bipartit $K_{n,m}$

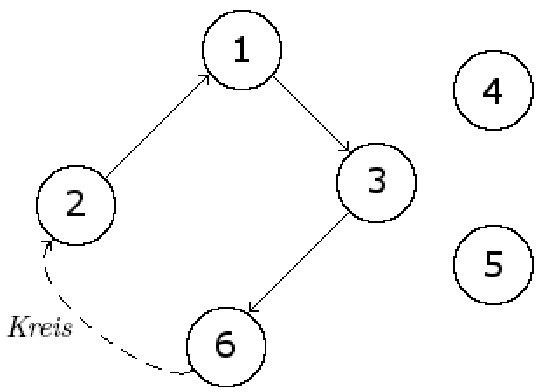
Eine endliche Menge $(v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$, $k \geq 0$ heisst **Kette**, falls e_i Endknoten v_{i-1} und v_i hat



Kette, kein Pfad, kein Weg



Kette, Pfad, kein Weg



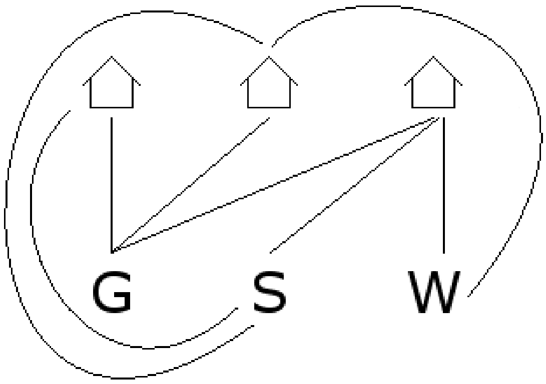
Kette, Pfad, Weg

Ein Pfad, der jede Kante genau einmal enthält, heisst **Eulerpfad**.
Wenn der Pfad geschlossen ist, dann **Eulertour**.

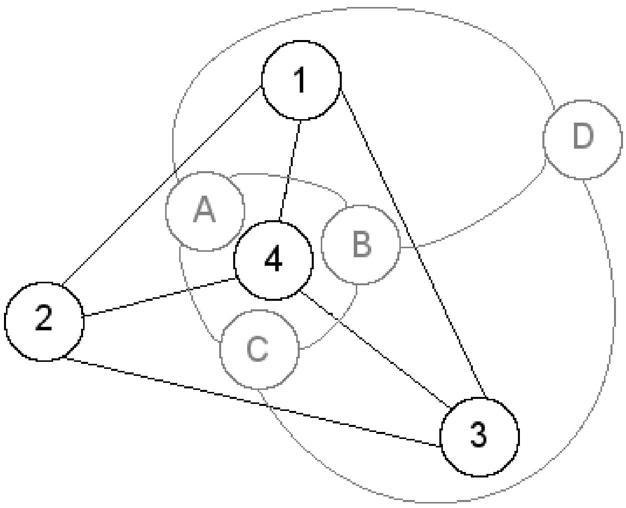
Ein Pfad, der jede Knoten genau einmal enthält, heisst **Hamiltonweg**.
Wenn der Pfad geschlossen ist, dann **Hamiltontour**.

Ein Graph, der so in der Ebene zeichenbar ist, dass sich keine 2 Kanten kreuzen, nennt man **planar**.

Ist $K_{3,3}$ planar?



oder

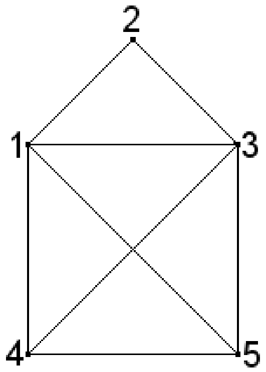


dualer Graph

<u>**Exkurs**</u> Algorithmen

Ein Problem ist eine Fragestellung mit offenen Parametern und einer Spezifikation, wie eine Lösung aussieht.

Beispiel:
$G = (V, E)$, enthält G eine Eulertour?



Lösung: $1, 2, 3, 4, 5, 3$
$1, 5, 4, 1$

$V = \{1, 2, 3, 4, 5\}$
$E = \{12, 13, 13, \ldots\}$

$n \in \mathbb{N}_0 \quad \log_2 n$
$\langle n \rangle := \lceil \log_2(n+1) \rceil + 1$ für $n \in \mathbb{Z}$
Kodierungslänge von n
$\langle r \rangle := \langle p \rangle + \langle q \rangle$ für $r = \frac{p}{q} \in \mathbb{Q}$ teilerfremd.

Wenn alle Parameter festgelegt sind, spricht man von einem **Problembeispiel**.
Die „Größe“ eines Problembeispiels ist die Kodierungslänge der festgelegten Eingabeparameter
Beispiel:

$$\begin{aligned}\langle G \rangle : &= \sum_{i \in V} \langle i \rangle + \sum_{i,j \in E} \langle i \rangle + \langle j \rangle \\ &\approx \underbrace{|V|}_{n} + \underbrace{|E|}_{m}\end{aligned}$$

Ein **Algorithmus** ist eine Anleitung zur schrittweise Lösung eines Problems. Wir sagen ein Algorithmus löst ein Problem Π, falls für alle Problembeispiele $I \in \Pi$ A eine Lösung in endlicher Zeit findet.
Ein **Schritt** ist eine elementare Operation:
Addieren, Subtrahieren, Vergleichen, $\begin{bmatrix} \text{Multiplikation} \\ \text{Division} \end{bmatrix}$

Laufzeit eines Algorithmus ist die Anzahl Schritte, die notwendig sind sind zur Lösung des Problembeispiels.

$$\begin{aligned}&I \in \Pi,\ l = \langle I \rangle \\ &T(l) = \text{Laufzeit des Algorithmus} = \text{Anzahl der Schritte}\end{aligned}$$

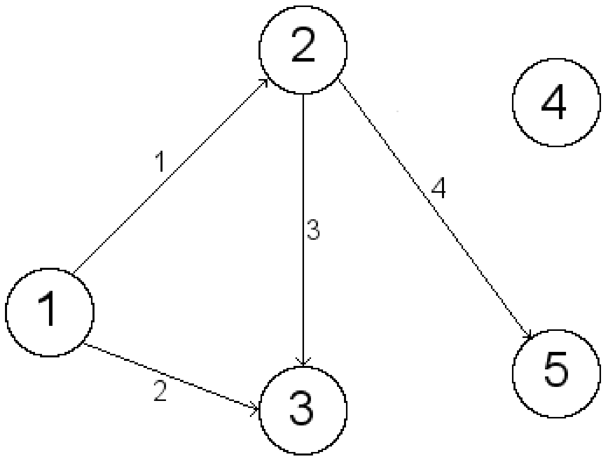
Ein Algorithmus A läuft in Polynomialzeit, falls es ein Polynom p gibt

$$\boxed{T(l) \leq p(l)} \qquad \forall\, l = \langle I \rangle, I \in \mathit{\Pi}$$

Die Menge der Probleme, die in Polynomialzeit lösbar sind, bezeichnen wir mit $\mathcal{P}$.

Definition 3.2 (Größenordnung von Funktionen)
$M = \{f \mid f : \mathbb{N} \to \mathbb{R}\}$ = Menge der reellwertigen Funktionen (z.B. $T \in M$). Sei $g \in M$:
$\mathcal{O}(g) = \{f \in M \mid \exists\, c, n_0 \in \mathbb{N} : f(n) \leq c \cdot g(n), n \geq n_0\}$
$g(n) = n \quad c = 6 \quad n_0 = 7$
$f(n) \ = 5n + 3 \in \mathcal{O}(g),\ 5n + 3 \leq 6n$ für $n \geq 7$
$\bigotimes(g) = 5n + 3 \in \mathcal{O}(g),\ 5n + 3 \leq 6n$ für $n \geq 7$
$\Theta(g) \ = \mathcal{O}(g) \cup \bigotimes(g)$

Speicherung von Graphen



(1) Kanten-Bogen-Liste:

$n, m, a_1, e_1, a_2, e_2, \ldots, a_m, e_m$ wobei a_1 der Anfangsknoten sei und e_1 der Endknoten.
$5, 4, 2, 1, 2, 1, 3, 2, 3, 2, 5$

+ Effiziente Speicherung $\mathcal{O}(m)$
− Zugriff = $\mathcal{O}(m)$

(2) Adjazenzmatrizen:

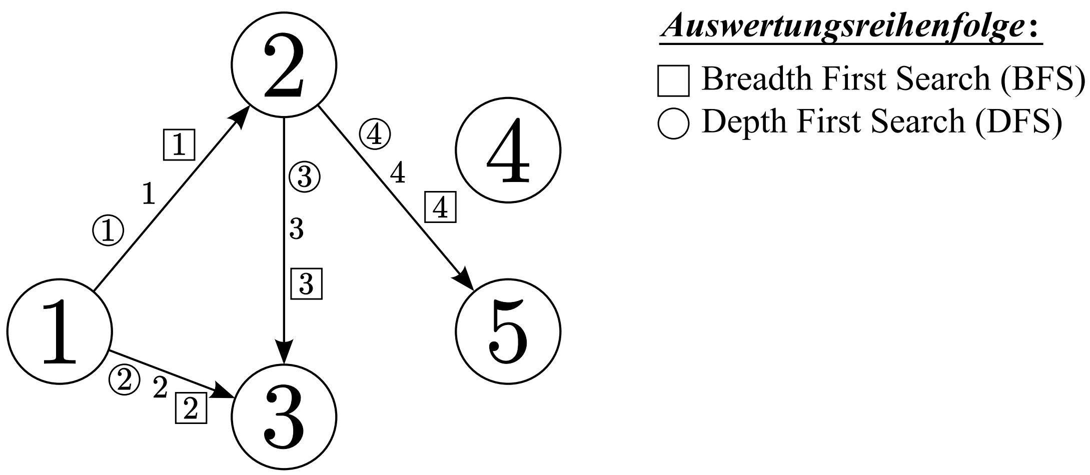
$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

+ Zugriff $\mathcal{O}(1)$
− Speicher $\mathcal{O}(n^2)$

(3) Adjazenzlisten:
Anzahl Knoten
Anzahl Kanten
Für jeden Knoten, den Grad und die inzidenten Kanten
5
4
1,2,1,2,1,3
2,3,1,2,2,3,2,5
3,2,1,3,2,3
4,0
5,1,2,5

Speicheraufwand: $\mathcal{O}(n+m)$
Zugriff: $\mathcal{O}(n)$

Gibt es einen Weg von 1 nach 4?



Programmiercodes des BFS und DFS Algorithmus für die Programmiersprache C zum testen und verstehen.

Breadth-First-Search Algorithmus

Breadth-First-Search arbeitet nach dem FIFO-Prinzip (First-In-First-Out).

Graph $G = (V, E)$, Liste L Anzahl der Zusammenhangskomponenten und die Menge der Knoten jeder Zusammenhangskomponente

1. Initialisierung:
   MARK(v) := -1 für alle $v \in V$
   NEXT := 1
2. FOR $v \in V$
   a) Füge v ans Ende der Liste L an
   b) WHILE $L \neq \emptyset$
      i. Wähle v vom Anfang der Liste und entferne v aus L
      ii. IF MARK(v) < 0 THEN
         A. MARK(v) := NEXT
      iii. END IF
      iv. FOR $w \in \delta(v)$ DO
         A. IF MARK(w) < 0 THEN
            - MARK(w) := NEXT
            - Füge w am Ende der Liste an
         B. END IF
      v. END FOR
   c) END WHILE
   d) $NEXT = NEXT + 1$
3. END FOR

$NEXT$ gibt die Anzahl der Zusammenhangskomponenten an. Die Knoten einer Zusammenhangskomponente sind mit demselben Wert markiert.

Depth-First-Search Algorithmus

Depth-First-Search arbeitet nach dem LIFO-Prinzip (Last-In-First-Out).

Graph $G = (V, E)$ Anzahl der Zusammenhangskomponenten und die Menge der Knoten jeder Zusammenhangskomponente

1. Initialisierung:
   MARK(v) := -1 für alle $v \in V$
   NEXT := 1
2. For $v \in V$ Do
   a) If MARK(v) < 0 Then
      i. MARK(v) := NEXT
      ii. For $w \in \delta(v)$ Do
         A. check_nachbar(w, NEXT)
      iii. NEXT := NEXT + 1
   b) End If
3. End For
4. Gib NEXT aus

check_nachbar(w, NEXT)
Input: $G = (V, E)$, $w \in V$, NEXT, MARK($\cdot$)

(1) If MARK(w) < 0 Do
   (2) MARK(w) = NEXT
   (3) For $u \in \delta^+(w)$ Do
      (4) check_nachbar(u, NEXT)
   (5) End For
(6) End If