4 Bäume und Wälder

In diesem Kapitel beschäftigen wir uns mit dem Problem in einem Graphen mit Kantengewichten einen *aufspannenden Baum minimalen Gewichts* oder einen *Wald maximalen Gewichts* zu finden.

Definitionen

Ein Graph G heißt **zusammenhängend**, falls es zu jedem Knotenpaar $s, t \in V$ einen $[s,t]$-Weg in G gibt. Die **Komponenten** von G sind die bezüglich Kanteninklusion maximalen zusammenhängenden Teilgraphen von G. Wir bezeichnen mit $\zeta(G)$ die Anzahl der Komponenten von G. Ein Knoten v heißt **Artikulationsknoten** von G, falls $\zeta(G - \{v\}) > \zeta(G)$.

Wir nennen eine Kantenmenge B einen **Wald** in G, falls $(V(B), B)$ keinen Kreis enthält. Ist $(V(B), B)$ außerdem zusammenhängend, so heißt B ein **Baum**. Ein Baum B in einem Graphen heißt **aufspannend**, falls $V(B) = V$ ist. Ist B ein Baum und v ein Knoten mit $d_B(v) = 1$, so heißt v **Blatt von** B.
Beide Probleme sind in direkter Weise äquivalent, siehe Übung.

Algorithmen

Der folgende Algorithmus bestimmt einen maximalen (bzgl. Kanteninklusion) Wald minimalen Gewichts.

Algorithmus 4.1 (Kruskal, 1956 [1])

INPUT: Graph $G = (V, E)$, Gewichte c_e für $e \in E$.
OUTPUT: Maximaler Wald (bzgl. Kantenzahl) $T \subseteq E$ minimalen Gewichts.

(1) Sortiere die Kantengewichte in nicht-absteigender Reihenfolge:

$$c_{e_1} \leq c_{e_2} \leq \cdots \leq c_{e_m}$$

(2) Setze $T := \emptyset$.

(3) FOR $i = 1$ TO m DO

(4) IF $T \cup \{e_i\}$ enthält keinen Kreis THEN $T := T \cup \{e_i\}$

(5) Gib T aus.

Satz 4.2 Kruskals Algorithmus (4.1) ist korrekt.

Die Laufzeit beträgt: $\mathcal{O}(\underbrace{m \log m}_{\text{Sortieren}} + \overbrace{m}^{\text{Schleife}} \cdot \overbrace{n}^{\text{Kreisprüfung}}) = \mathcal{O}(m \cdot n)$.

Wir beweisen folgende Behauptung per Induktion über i.

> Nach dem i-ten Schritt gibt es einen maximalen (bzgl. Kanteninklusion) Wald $\overline{W}$ minimalen Gewichts mit $\overline{W} \cap \{e_1, \ldots, e_i\} = W \cap \{e_1, \ldots, e_i\}$

Beweis Induktion über i.

$\underline{i = 0}$: Für $i = 0$ ist die Behauptung offensichtlich erfüllt.

$\underline{i - 1 \to i}$: Wir unterscheiden zwei Fälle:

1. Fall: $W \cup \{e_i\}$ enthält einen Kreis, d.h. $e_i \notin W$.

Da $\overline{W} \cap \{e_1, \ldots, e_{i-1}\} = W \cap \{e_1, \ldots, e_{i-1}\}$ enthält auch $\overline{W} \cup \{e_i\}$ einen Kreis, somit $e_i \notin \overline{W}$.

2. Fall: $W \cup \{e_i\}$ enthält keinen Kreis, d.h. $e_i \in W$.

a) $e_i \in \overline{W}$, was die Behauptung ist.

b) $e_i \notin \overline{W}$. Damit enthält $\overline{W} \cup \{e_i\}$ einen Kreis C. Somit muss es ein $j > i$ geben mit $e_j \in \overline{W}, e_j \notin W$ und $c_{e_j} \geq c_{e_i}$, da $\overline{W} \cap \{e_1, \ldots, e_{i-1}\} = W \cap \{e_1, \ldots, e_{i-1}\}$ und $W \cup \{e_i\}$ keinen Kreis enthält. D.h. $\overline{W}' := \big(\overline{W} \setminus \{e_j\}\big) \cup \{e_i\}$ ist auch ein maximaler (bzgl. Kanteninklusion) Wald minimalen Gewichts, der die Behauptung erfüllt.

Für $i = m$ folgt die Behauptung. ■

Bemerkung 4.3

(a) Wenn G zusammenhängend ist, so ist ein Wald maximaler Kantenanzahl und minimalen Gewichts identisch mit einem minimal aufspannenden Baum.

(b) Der Kruskal-Algorithmus gehört zu der Klasse der Greedy-Algorithmen (engl. greedy = gefräßig, gierig). In jedem Schritt wird diejenige Kante ausgewählt, die hinsichtlich der Kostenfunktion bestmöglich ist. Siehe dazu auch ein späteres Kapitel.

(c) Der Kruskal-Algorithmus kann analog auch zur Bestimmung eines maximalen (bzgl. Gewichtsfunktion) Waldes benutzt werden, wobei man nur Kanten in Erwägung zieht, deren Gewichte positiv sind.

Ein gemeinsames Gerüst für viele Algorithmen ist der folgende

Algorithmus 4.4

INPUT: $G = (V, E)$ zusammenhängend, Gewichte c_e für $e \in E$.
OUTPUT: Aufspannender Baum $T \subseteq E$ minimalen Gewichts.

(1) Initialisierung: Für $i \in V$ setze $V_i = \{i\}$, $T_i := \emptyset$.

(2) Führe $|V| - 1$ mal aus:

  (3) Wähle nichtleeres V_i.

  (4) Wähle $uv \in \delta(V_i)$, $u \in V_i$ mit $c_{uv} \leq c_{pq}$ für alle $pq \in \delta(V_i)$.

  (5) Bestimme j, so dass $v \in V_j$.

  (6) Setze $V_i := V_i \cup V_j$, $V_j := \emptyset$ und $T_i := T_i \cup T_j \cup \{uv\}$, $T_j := \emptyset$.

(7) Gib T_i mit $T_i \neq \emptyset$ aus!

Satz 4.5 Der Algorithmus 4.4 arbeitet korrekt.

Beweis Wir zeigen durch Induktion über $k = \sum |T_i|$, dass G einen minimal aufspannenden Baum T mit $T_i \subseteq T$ enthält.

$\underline{k = 0:}$ Trivial.

$\underline{k - 1 \to k:}$ Sei uv die hinzugefügte Kante. Nach Annahme gibt es einen minimalen aufspannenden Baum T mit $T_i \subseteq T$ für alle bisher bestimmten Mengen T_i.

$uv \in T$**:** erfüllt die Behauptung.

$uv \notin T$**:** So muss $T \cup \{uv\}$ einen Kreis enthalten. Also gibt es eine Kante $rs \in T$ mit $r \in V_i$ und $s \in V \setminus V_i$. Nach Wahl der Kante in (4) gilt $c_{rs} \geq c_{uv}$. Also ist $(T \setminus \{rs\}) \cup \{uv\}$ ebenfalls ein minimal aufspannender Baum, der die Bedingung der Induktionsannahme erfüllt.

■

Ein Spezialfall von Algorithmus 4.4 ist der folgende Algorithmus, der auf Prim zurückgeht [2]. Er ist besonders geeignet für vollständige Graphen:

Algorithmus 4.6 (Algorithmus von Prim, 1957 [2])

INPUT: $G = (V, E)$ zusammenhängend, Gewichte c_e für $e \in E$.
OUTPUT: Aufspannender Baum $T \subseteq E$ minimalen Gewichts.

(1) Initialisierung: Wähle $w \in V$ beliebig und setze $T := \emptyset$, $W := \{w\}$ und $V' := V \setminus \{w\}$.

(2) IF $V' = \emptyset$ THEN **Stop** (gib T aus).

(3) Wähle $c_{uv} = \min \left\{ c_e \;\middle|\; e \in \delta(W) \right\}$ mit $u \in W$ und $v \in V'$.

(4) Setze $T := T \cup \{uv\}$, $W := W \cup \{v\}$ und $V' := V' \setminus \{v\}$.

(5) Gehe zu (2) .

Die Laufzeit von Algorithmus 4.6 ist $\mathcal{O}(n^2)$. Für vollständige Graphen ist dies bestmöglich.

Literatur

[1] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Am. Math. Soc*, 7:48 – 50, 1956.

[2] R.C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389 – 1401, 1957.