Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Übungsblatt 2 - Lösung

Vorlesung Algorithmentechnik im WS 08/09

Ausgabe 04. November 2008
Abgabe 18. November, 15:30 Uhr (im Kasten vor Zimmer 319, Informatik-Hauptgebäude, 3. OG)

Bitte schreiben Sie nur Ihren Namen und keine Matrikelnummer auf Ihr Übungsblatt. Die Bearbeitung in Zweiergruppen ist ausdrücklich erwünscht.

Problem 1: Boruvka-MST

Sei $G = (V, E)$ ein einfacher, ungerichteter, zusammenhängender Graph mit **injektiver** Kostenfunktion $w : E \to \mathbb{N}^+$. Wir schreiben (u, v) für eine ungerichtete Kante $\{u, v\}$. Wir bezeichnen eine Kante (u, v) als *lokal minimal* wenn sie für u oder v unter allen Kanten, die mit diesem Knoten inzidieren, das kleinste Gewicht hat.

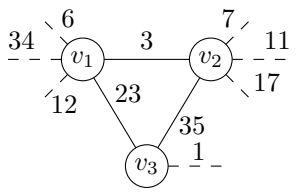
(a) Zeigen sie: Jede lokal minimale Kante gehört zu allen MST von G.

(b) Die Menge aller lokal minimalen Kanten eines Graphen ist ein Wald.

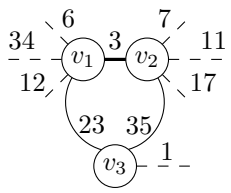
Eine *Boruvka-Phase* ist ein Algorithmus, der Folgendes leistet:

Schritt (1) Er bestimmt die Menge $L(G)$ aller lokal minimalen Kanten von G.

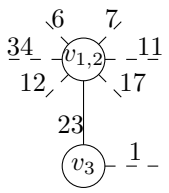
Schritt (2) Er berechnet den kontrahierten Graphen $G/L(G)$, wie in der Abbildung unten illustriert. Falls Mehrfachkanten entstünden, wird nur die Kante mit geringsten Gewicht behalten, und die anderen entfernt.



Der Graph vor der Boruvka Phase. Kante mit Gewicht 3 ist lokal minimal.



Knoten v_1 und v_2 werden entlang Kante mit Gewicht 3 kontrahiert.



Knoten v_1 und v_2 kontrahiert; teurere, doppelte Kante zu v_3 gelöscht.

Der so aus G entstehende Graph werde mit $B(G)$ bezeichnet.

(c) Implementieren sie in Pseudocode die Boruvka-Phase mit einer Worst-case-Laufzeit von $O(m)$ Vergleichsoperationen. Begründen Sie die Laufzeit. Ihnen steht eine Funktion LöscheTeureMehrfachkanten(G) zur Verfügung, die (im Worst-case) mit $O(m)$ Vergleichsoperationen auskommt.

(d) Zeigen sie, dass durch eine Boruvka-Phase die Anzahl der Knoten um mindestens die Hälfte reduziert wird.

Man kann zeigen, dass für jeden Graphen G gilt: Die Kanten in $L(G)$ und die Kanten in einem MST von $B(G)$ bilden zusammen einen MST von G.

(e) Nutzen sie Teilaufgabe (c) um einen MST mit $O(m \log n)$ Vergleichsoperationen im Worst-case zu bestimmen (Pseudocode). Begründen sie die Worst-case-Laufzeit von $O(m \log n)$.

Lösung.

(a) Sei T' MST von G, (x, y) eine lokal minimale Kante. Angenommen T' enthalte (x, y) nicht. Dann gibt es in T' eine andere Kante (x, y'), die auf dem Pfad in T' von x nach y liegt. Der aufspannende Baum T, der entsteht, indem man (x, y') aus T' entfernt und (x, y) hinzunimmt besitzt geringeres Gewicht als T' - im Widerspruch zur Vorraussetzung. Genauer:

- *$w(T) < w(T')$, denn $w(x, y) < w(x, y')$ und $w(T) = w(T') - w(x, y') + w(x, y)$ denn (x, y) ist lokal minimal und Kantengewichte sind paarweise verschieden.*
- *T spannt G auf. Da T' den Graph G aufspannt, bleibt nur zu zeigen, dass jeder Pfad in T' zwischen zwei beliebigen Knoten v_1, v_2 ber (x, y') eine Entsprechung in T hat. Dazu ersetzt man in solchen Pfaden die Kante (x, y') durch den Pfad über (x, y) nach y'.*
- *T ist ein Baum. Da T' Baum ist, müsste ein Zyklus in T die Kante (x, y) enthalten. Also gibt es in $T \setminus (x, y) = T' \setminus (x, y')$ einen Pfad von x nach y. Der einzige x-y-Pfad in T' beinhaltet allerdings (x, y'). Widerspruch.*

(b) Zu zeigen: Zyklenfreiheit. Angenommen es gäbe in der Menge der lokal minimalen Kanten E_{lm} einen Zyklus. Betrachte darauf die Kante größten Gewichts (u, v). Diese ist eindeutig, da die Gewichtsfunktion injektiv ist. (u, v) kann nun aber keine lokal minimale Kante sein, denn für u und v gibt es bereits auf dem Zyklus je eine anliegende Kante niedrigeren Gewichtes. Widerspruch.

(c) Rufe Boruvka-Phase(A, n) *auf wie in Algorithmus 1. Die Schleife in Zeile 4 hat in jedem Fall Laufzeit $\Theta(n+m)$, denn sie betrachtet jeden Knoten genau einmal und jede Kante genau zweimal.*

Die Zusammenhangskomponenten in dem durch $L(G)$ induzierten Graphen können durch Breitensuche gefunden werden. Merkt man sich so gefundenen Kanten, dann wird die Schleife in Zeile 7 höchstens $O(m)$ mal ausgeführt. Die amortisierte Laufzeit aller solcher Breitensuchen ist ebenfalls durch $O(m)$ beschränkt.

Zeile 8 wird insgesamt höchstens n mal durchgeführt, denn der Graph besteht aus höchstens n Zusammenhangskomponenten und besitzt konstanten Aufwand. Zeilen 9 und 10 haben amortisiert insgesamt einen Aufwand in $O(m)$ denn jede Kante wird genau einmal berücksichtigt. Zeile 11 hat nach Vorraussetzung einen Aufwand in $O(m)$. Insgesamt ergibt sich also die Laufzeit zu: $T_{worst}(n) \in O(m)$.

(d) In jeder Boruvka-Phase *werden alle Knoten einer Zusammenhangskomponente (des Waldes der lokal minimalen Kanten) zu einem Knoten verschmolzen. Jeder Knoten besitzt eine lokal minimale Kante. Da jede Kante nur zwei Endpunkte hat, kann sie für höchstens zwei Knoten lokal minimal sein. Also gibt es mindestens $n/2$ lokal minimale Kanten, die in einer Boruvka-Phase kontrahiert werden. Mit jeder kontrahierten Kante verringert sich jedoch auch die Anzahl der Knoten um Eins.*

(e) Rufe Boruvka-MST(A, n) *auf wie in Algorithmus 2. Laut Teilaufgabe c) benötigt jede Boruvka-Phase $O(m)$ Zeit und halbiert laut d) mindestens die Knotenanzahl. Der Algorithmus terminiert, wenn nur noch ein Knoten vorhanden ist. Die Anzahl der Phasen liegt folglich in $O(\log n)$ und die Laufzeit $T(n, m)$ in $O(m \log n)$.*

Algorithmus 1 : BORUVKA-PHASE

Eingabe : Graph $G(V_G, E_G)$
Ausgabe : Kantenmenge $L(G)$, Graph $B(G)(V_B, E_B)$

```
1  L(G) ← ∅
2  V_B ← V_G
3  E_B ← E_G
4  Für v ∈ V_G
5  |  Bestimme lokal minimale Kante e von v
6  |  L(G) ← L(G) ∪ {e}
7  Für jede Zusammenhangskomponente C = {v_1, ..., v_k} in dem von L(G) induzierten
   Subgraphen von G
8  |  füge Knoten v_C in V_B ein
9  |  lösche alle in C × C enthaltenen Kanten aus E_B
10 |  ersetze in E_B alle Kanten der Form (v_i, w) mit w ∉ C durch (v_C, w)
11 LöscheTeuereMehrfachkanten(B(G))
```

Algorithmus 2 : BORUVKA-MST

Eingabe : Graph $G(V, E)$
Ausgabe : MST von G

```
1  T ← ∅
2  solange |G(V)| > 1 tue
3  |  L(G), G ← Boruvka-Phase(G)
4  |  T ← T ∪ L(G)
5  return L(G)
```

Problem 2: Schnitte und MSTs

(a) Zeigen Sie: Wenn in einem Graph G mit reellen Kantengewichten für jeden Schnitt die den Schnitt kreuzende Kante minimalen Gewichts eindeutig ist, dann hat G einen eindeutigen aufspannenden Baum minimalen Gewichts.

Proof. Wir nehmen an, dass die Voraussetzung erfüllt ist, aber G zwei verschiedene MSTs T_1 und T_2 hat. Sei $e_1 \in E(T_1) \setminus E(T_2)$ eine Kante von T_1, die nicht in T_2 ist. Diese partitioniert T_1 in zwei disjunkte Teilbäume. Wir fassen diese als Schnitt $(S, V \setminus S)$ auf. Unter allen Kanten, die den Schnitt kreuzen, hat e_1 minimales Gewicht (sonst wäre T_1 nicht minimal). Andererseits induziert e_1 in T_2 einen Kreis. Mindestens eine weitere der Kanten dieses Kreises kreuzt den Schnitt. Diese Kante sei e_2. Wegen $c(e_1) < c(e_2)$ ist $T_2 \setminus \{e_2\} \cup \{e_1\}$ ein Baum mit echt kleinerem Gewicht als T_2, ein Widerspruch. □

(b) Gilt die Umkehrung von (a)?

Proof. Nein, als Gegenbeispiel siehe Abbildung 1. □

(c) Zeigen oder widerlegen Sie: Wenn in einem Graph G mit reellen Kantengewichten alle Kanten paarweise verschiedene Gewichte haben, dann hat G einen eindeutigen aufspannenden Baum minimalen Gewichts.

Proof. Folgt aus (a): Wenn alle Gewichte verschieden sind, dann gibt es unter den Kanten, die einen Schnitt kreuzen, stets genau eine mit minimalem Gewicht. □

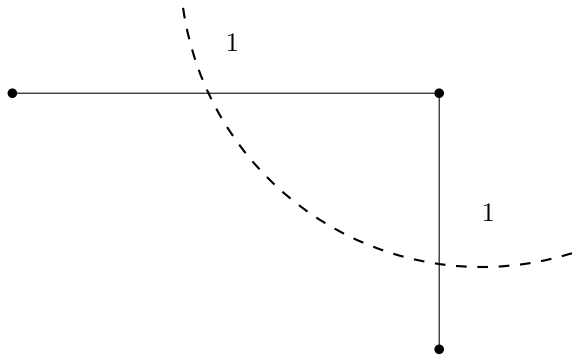


Abbildung 1: Gegenbeispiel zu 5 (b)

(d) Zeigen oder widerlegen Sie die Umkehrung der Aussage in (c).

Proof. Auch diese Umkehrung gilt nicht, siehe als Gegenbeispiel wie in Teilaufgabe (b) Abbildung 1. □

Problem 3: Vier Äquivalente Aussagen

Zur Erinnerung: Ein Zyklus ist eine Knotenfolge, bei der Anfangs- und Endknoten übereinstimmen und zwischen je zwei aufeinanderfolgenden Knoten eine Kante existiert. Ein Kreis ist ein Zyklus, bei dem keine Knoten außer dem Anfangsknoten mehrmals vorkommen. In jedem Zyklus ist ein Kreis enthalten.

Zeigen Sie, dass die folgenden Aussagen für einen ungerichteten Graphen $G = (V, E)$ äquivalent sind.

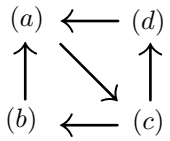
(a) G ist zusammenhängend und enthält keine Kreise.

(b) G ist maximal kreisfrei, d.h. G enthält keinen Kreis und für je zwei Knoten $u, v \in V$ mit $\{u, v\} \notin E$ enthält der Graph $G' = (V, E \cup \{\{u, v\}\})$ einen Kreis.

(c) Zwischen je zwei Knoten in V gibt es genau einen Weg in G.

(d) G ist minimal zusammenhängend, d.h. G ist zusammenhängend, und für jede Kante $e \in E$ ist der Graph $G' = (V, E \setminus \{e\})$ nicht zusammenhängend.

Proof. Falls G nur einen Knoten enthält, dann sind die Aussagen trivialerweise äquivalent. Also bestehe der Graph G im folgenden aus mindestens zwei Knoten. Folgende Implikationen induzieren die Äquivalenz der vier Aussagen:



(a) $(a \Rightarrow c \equiv \neg c \Rightarrow \neg a)$:
Der Graph G enthalte zwei Knoten v und w, zwischen denen 0 oder 2 Pfade existieren. Falls es keinen Weg gibt, dann ist G nicht zusammenhängend. Falls es zwei verschiedene Pfade W_1 und W_2 gibt, dann ergibt die folgende Knotenfolge einen Zyklus: v, W_1, w, W_2, v. Daraus folgt, dass G nicht kreisfrei ist.

(b) $(c \Rightarrow b)$:
Falls G einen Kreis enthält, dann gibt es mindestens zwei Knoten, zwischen denen es mehr als einen Pfad gibt. Betrachte nun zwei beliebige Knoten v und w, so dass $\{w, v\} \notin E$. Zwischen v und w gibt es genau einen Pfad W in G. In $G' := (V, E \cup \{\{w, v\}\})$ gäbe es dann zwei verschiedene Pfade von v nach w. Daraus folgt, dass G einen Zyklus (v, W, w, v) und somit einen Kreis enthalten müsste.

(c) $(c \Rightarrow d)$:
G ist offensichtlich zusammenhängend. Betrachte die Endknoten v und w einer beliebigen Kante $e \in E$. Diese Kante ist der einzige Pfad von v nach w. Durch Entfernen von e würde kein Weg mehr zwischen v und w existieren, d.h. diese Knoten würden zwei verschiedenen Zusammenhangskomponenten angehören. Somit wäre $G' = (V, E \backslash \{e\})$ nicht zusammenhängend. Da e beliebig gewählt wurde folgt die Behauptung.

(d) $(b \Rightarrow a)$:
Wäre G nicht zusammenhängend, so würden mindestens zwei Zusammenhangskomponenten Z_1 und Z_2 existieren. Wähle zwei Knoten $b_1 \in Z_1$ und $b_2 \in Z_2$ und verbinde sie durch eine Kante e. Diese Kante induziert keinen Kreis. Also wäre G nicht maximal kreisfrei. Somit wäre G zusammenhängend und enthält außerdem keine Kreise.

(e) $(d \Rightarrow a)$:
Wir zeigen durch einen Widerspruchsbeweis, dass G keine Kreise enthält. Würde G einen Kreis enthalten, dann könnten wir eine Kante aus diesem Kreis entfernen und der resultierende Graph bliebe immer noch zusammenhängend. Somit ist G zusammenhängend und kreisfrei. □

Problem 4: Einmaschinenscheduling

Eine Reihe von Aufträgen $A_1, \ldots, A_n$ müssen von einer einzigen Maschine abgearbeitet werden, es kann also zu jedem Zeitpunkt höchstens ein Auftrag bearbeitet werden. Alle Aufträge benötigen die gleiche Bearbeitungszeit. Jeder Auftrag A_i hat eine Deadline $D_i \in \mathbb{R}$ bis zu der er fertig sein muss.

(a) Zeigen Sie, dass die Menge aller Teilmengen von Aufträgen, die rechtzeitig bearbeitet werden können, ein Matroid ist.

Proof. Eine Menge von Aufträgen, die rechtzeitig bearbeitet werden können, heiße *zulässig*. Sei o.B.d.A. die Bearbeitungszeit eines Auftrages auf 1 normiert und $D_1 \leq D_2 \leq \cdots \leq D_n$. Da die Bearbeitungszeit für j Aufträge j beträgt, ist eine Teilmenge $\{A_{i_1}, \ldots, A_{i_k}\}$ von k paarweise verschiedenen Aufträgen $(i_1 < \cdots < i_k \in \{1, \ldots, n\})$ genau dann zulässig, wenn gilt:

$$\forall j \leq k \colon D_{i_j} \geq j \qquad (*)$$

- Eine leere Menge von Aufträgen kann stets rechtzeitig bearbeitet werden.
- Eine Teilmenge einer zulässigen Menge von Aufträgen kann ebenfalls rechtzeitig bearbeitet werden.
- Seien $\mathcal{I} := \{A_{i_1}, \ldots, A_{i_k}\}$ und $\mathcal{J} := \{A_{j_1}, \ldots, A_{j_\ell}\}$ zwei zulässige Mengen mit $k < \ell$ und $i_1 < i_2 < \ldots < i_k$ sowie $j_1 < j_2 < \ldots < j_l$. Seien k' bzw. ℓ' die minimalen Indizes mit $i_{k'+1} = j_{\ell'+1}$, $i_{k'+2} = j_{\ell'+2}$, …, $i_k = j_\ell$, d.h. die Aufträge nach k' bzw. ℓ' sind paarweise gleich und $A_{i_{k'}}$ und $A_{j_{\ell'}}$ sind verschieden.
  Behauptung: $\mathcal{I}' := \{A_{i_1}, \ldots, A_{i_{k'}}, A_{j_{\ell'}}, A_{i_{k'+1}}, \ldots, A_{i_k}\}$ ist zulässig.
  Beweis: Für $\nu \leq k'$ gilt $D_{i_\nu} \geq \nu$ da $\mathcal{I}$ zulässig ist. Da $\ell > k$ ist, gilt auch $\ell' > k'$, also $\ell' \geq k' + 1$. Daraus folgt $D_{i_{\ell'}} \geq \ell' \geq k' + 1$, da $\mathcal{J}$ zulässig ist. Außerdem gilt für $1 \leq \mu \leq k - k'$, dass $D_{i_{k'+\mu}} = D_{j_{\ell'+\mu}} \geq \ell' + \mu \geq k' + 1 + \mu$ ist. Also ist $\mathcal{I}'$ zulässig.

Verbal: Falls $A_{i_k} \neq A_{j_\ell}$ dann kann A_{j_ℓ} an $\mathcal{I}$ angehängt werden, da $|\mathcal{J}| > |\mathcal{I}|$ und $\mathcal{J}$ zulässig. Falls $A_{i_k} = A_{j_\ell}$ dann laufe in $\mathcal{I}$ und $\mathcal{J}$ solange parallel rückwärts, bis der erste Unterschied $A_{i_{k'}} \neq A_{j_{\ell'}}$ auftritt. Dann, füge $A_{j_{\ell'}}$ nach $A_{i_{k'}}$ in $\mathcal{I}$ ein. Da die Elemente $A_{j_{\ell'}}$ bis A_{j_ℓ} in $\mathcal{J}$ zulässig sind, sind sie es auch in $\mathcal{I}$. □

(b) Wird ein Auftrag nicht rechtzeitig fertig, so muss eine Strafe P_i bezahlt werden, deren Höhe vom jeweiligen Auftrag abhängt. In welcher Reihenfolge sollten die Aufträge abgearbeitet werden, damit die Gesamtstrafe minimiert wird?

Hinweis/Spoiler: Sei o.B.d.A. $D_1 \leq D_2 \leq \cdots \leq D_n$. Eine Teilmenge $\{A_{i_1}, \ldots, A_{i_k}\}$ von k Aufträgen ($i_1 \neq \cdots \neq i_k \in \{1, \ldots, n\}$) kann rechtzeitig bearbeitet werden, wenn für alle $j \leq k$ gilt : $D_{i_j} \geq j$.

Proof. Die optimale Reihenfolge sieht so aus, dass zuerst eine Menge von Aufträgen abgearbeitet wird, deren Gesamtstrafe maximal ist (aber in aufsteigender Reihenfolge ihrer Deadlines). Damit wird die Gesamtstrafe minimiert, die durch die folgenden, nicht rechtzeitig bearbeiteten Aufträge erzeugt wird.

Also führt folgender Greedy-Ansatz zum Ziel:

1. Sortiere die Aufträge in nicht-aufsteigender Reihenfolge ihrer Strafen.
2. Setze $S := \emptyset$.
3. Für jeden Auftrag A in obiger Reihenfolge
4. Wenn $S \cup \{A\}$ zulässig ist
5. Setze $S := S \cup \{A\}$.

□