Universität Karlsruhe
Fakultät für Informatik
ITI Wagner

3. Musterlösung

Problem 1: Boruvka–MST 3pt

(a) *Beweis durch Widerspruch.* Sei T' MST von G, e die lokal minimale Kante eines Knoten x, y der andere Endpunkt von e. Angenommen T' enthalte e nicht. Dann gibt es in T' eine andere zu x adjazente Kante e', die auf dem kürzesten Pfad in T' von x nach y liegt (T' ist aufspannend) und deren anderer Endpunkt y' sei. Betrachtet man nun den aufspannenden Baum T, der entsteht, indem man aus T' e' entfernt und e hinzunimmt, so fällt auf, dass T geringeres Gewicht als T' besitzt - im Widerspruch zur Vorraussetzung. Genauer:

- $w(T) < w(T')$, denn $w(e) < w(e')$ und $w(T) = w(T') - w(e') + w(e)$ (Hinweis: n.V. e lokal minimal und Kantengewichte paarweise verschieden).
- T spannt G auf. Da T' G aufspannt, bleibt nur zu zeigen, dass jeder Pfad in T' zwischen zwei bel. Knoten v_1, v_2 über e' eine Entsprechung in T hat. Dazu ersetzt man in solchen Pfaden die Kante e' durch den Pfad über e und von y nach y'.
- T ist ein Baum. Da T' Baum ist, müsste ein Zyklus in T die Kante e enthalten. Also gibt es in $T - e$ einen (weiteren) Pfad von x nach y und damit auch in $T' - e$. Dieser Pfad ist verschieden vom kürzesten Pfad in T' von x nach y, denn jener beinhaltet nach Voraussetzung die Kante e' (s.o.), existiert somit jedoch nicht in $T - e$. Also gäbe es in T' zwei verschiedenen Pfade von x nach y und damit einen Zyklus - im Widerspruch dazu, dass T' Baum ist.

(b) Zu zeigen: Zyklenfreiheit. Angenommen es gäbe in der Menge der lokal minimalen Kanten E_{lm} einen Zyklus. Betrachte darauf die Kante größten Gewichts e_{max}. Diese ist eindeutig, da die Gewichtsfunktion injektiv ist. e_{max} kann nun aber keine lokal minimale Kante sein, denn für beide inzidenten Knoten gibt es bereits auf dem Zyklus je eine Kante niedrigeren Gewichtes. Ein Widerspruch. Also ist E_{lm} zyklenfrei und ungerichtet und damit ein Wald.
Bemerkung: E_{lm} ist nicht zwingend zusammenhängend.

(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 $O(m)$ mal ausgeführt. Die amortisierte Laufzeit aller solcher Breitensuchen ist ebenfalls durch $O(m)$ beschränkt. Zeile 8 wird amortisiert genau n mal durchgeführt. Die innere Schleife in Zeile 9 wird *insgesamt* höchstens $2m$-mal durchlaufen, da jede Kante höchstens zwei Endpunkte in B(G) hat. Also hat die gesamte Schleife ab Zeile 7 die Worst-case-Laufzeit $O(n + m)$. Zeile 11 hat nach Voraussetzung die Laufzeit $O(m)$. Ferner ist $n \in O(m)$ da G zusammenhängend. 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. Also werden im *best-case* alle Knoten zu einem verschmolzen. Im *worst-case* dagegen besteht der Wald aus lauter 2er-Komponenten. 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. Also gilt:

$$T(n, m) \leq T\left(\frac{n}{2}, m - \frac{n}{2}\right) + O(m) \leq T\left(\frac{n}{2}, m\right) + O(m)$$

Nach Master-Theorem ergibt sich $T(n, m) \in O(m \log n)$.

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 in dem von L(G) induzierten Subgraphen von G
8  |  Kontrahiere Knoten v_1, ..., v_k ∈ C zu v_C
9  |  Für {v_i, w} ∈ E_G, w ≠ v_1, ..., v_k
10 |  |  E_B ← E_B − {v_i, w} + {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: Maybe-MST 2pt

Zunächst einige Vorüberlegungen:

- Die Korrektheit eines MST-Algorithmus kann über die Färbungsinvariante gezeigt werden: Falls der Algorithmus erschöpfend rote und grüne Regeln ausführt und am Ende die grünen Kanten ausgegeben werden, so arbeitet der Algorithmus korrekt.
- Mittels Breitensuche kann man testen, ob ein Graph zusammenhängend ist. Ferner ist das Auffinden von Zyklen in einem Graphen ebenfalls mit Breitensuche machbar. Die Worst-case Laufzeit von Beitensuche beträgt $O(|E| + |V|)$.
- In zusammenhängenden Graphen gilt $O(|E| + |V|) = O(|E|)$, da $|V| \in O(|E|)$

Im folgenden sei $m := |E|$ und $n := |V|$.

Algorithmus 3 : MAYBE-MST-A(G, w)

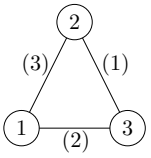
```
1  sortiere die Kanten in nichtsteigender Reihenfolge der Gewichte w
2  T ← E
3  Für jede Kante e in nichtsteigender Reihenfolge der Gewichte w
4  |  Wenn T − {e} ist ein zusammenhängender Graph
5  |  |  T ← T − {e}
6  return T
```

(a) Betrachten wir zunächst Algorithmus 3. Der Algorithmus liefert einen korrekten MST zurück. Zeile 4,5 testen, ob T einen Zyklus enthält und dieser wird gegebenenfalls aufgebrochen, daraus folgt zunächst ein Spannbaum. Da die Schleife in Zeile 3 in nichtsteigender Reihenfolge über die Kanten-

gewichte ausgeführt wird, gilt, dass falls Zeile 5 ausgeführt wird, die Kante e die schwerste dieses Zyklus ist (eine schwerere wäre bereits vorher entnommen worden, da sie auch auf zumindest diesem Kreis liegt). Das Entfernen einer Kante entspricht also dem rot färben. Nach Ablauf des Algorithmus ist T zyklenfrei, da wir alle Kanten betrachten und somit jeden Zyklus aufbrechen. Da heißt Algorithmus 3 führt die rote Regel erschöpfend aus. Die Korrektheit folgt nun, da man die nicht gefärbten Kanten nun noch mit der grünen Regel grün färben könnte.

Das Sortieren benötigt $O(m \log m)$. Für jede Kante muss eine Breitensuche in einem zusammenhängenden Graphen ausgeführt werden. Somit ergibt sich eine Laufzeit von $O(m^2)$.

(b) Algorithmus 4 arbeitet nicht korrekt. Bild b zeigt ein Gegenbeispiel. Eventuell werden die Kanten $\{1, 2\}$ und $\{1, 3\}$ zurückgegeben, was eindeutig nicht der MST des Graphen ist. Der MST beinhaltet die Kanten $\{1, 2\}$ und $\{2, 3\}$.



Algorithmus 4 : MAYBE-MST-B(G, w)

```
1 T ← ∅
2 Für jede Kante e in beliebiger Reihenfolge
3 |   Wenn T ∪ {e} hat keine Zyklen
4 |   |   T ← T ∪ {e}
5 return T
```

Mit Hilfe von UNION-FIND kann der Algorithmus mit Laufzeit $O(mG(m))$ implementiert werden. Hierzu wird zunächst für jeden Knoten ein MAKESET aufgerufen. Zeile 3 wird durch FIND$(u) \neq$ FIND(v) (mit $e = \{u, v\}$) ersetzt. Ferner wird in Zeile 4 noch UNION(u, v) ausgeführt. Dies ergibt dann eine Worst-case Laufzeit von $O(mG(m))$.

(c) Algorithmus 5 berechnet einen MST. In diesem Falle entspricht das Entfernen einer Kante dem rot färben. Analog zu Algorithmus 3 erfolgt hier eine erschöpfende Ausführung der roten Regel, und am Ende entsprechen ungefärbte Kanten grünen Kanten.

Die Laufzeit ist in $O(m^2)$, da für jede Kante eine Breitensuche ausgeführt werden muss.

Algorithmus 5 : MAYBE-MST-C(G, w)

```
1 T ← ∅
2 Für jede Kante e in beliebiger Reihenfolge
3 |   T ← T ∪ {e}
4 |   Wenn T hat einen Zyklus c
5 |   |   e' ← Kante mit maximalem Gewicht auf c
6 |   |   T ← T − {e'}
7 return T
```

Problem 3: MST 2pt

Sei $G = (V, E)$ ein Graph mit reeller Kantengewichtsfunktion $c : V \to \mathbb{R}$. Für jeden Schnitt in G sei außerdem die Kante minimalen Gewichts eindeutig bestimmt.

Seien nun T und T' zwei verschiedene minimal spannende Bäume auf G. Es gibt also mindestens eine

Kante $e \in T$, die nicht in T' enthalten ist.

Betrachte eine solche Kante $e \in T$, $e \notin T'$. Löschen wir e aus T, so zerfällt T in zwei Zusammenhangskomponenten, die einen Schnitt $(V \setminus S, S)$ in G induzieren. Die Kante e ist eine leichte Kante bezüglich S, denn gäbe es eine Kante $\tilde{e} \in G$ mit $c(\tilde{e}) < c(e)$, die den Schnitt kreuzt, so wäre $(T \setminus \{e\}) \cup \{\tilde{e}\}$ ein spannender Baum mit echt kleinerem Gewicht als T, was nicht sein kann.

Da $e \notin T'$ induziert die Kante e in T' einen Kreis. Betrachte nun die Kante $e' \in T'$, die auf diesem Kreis liegt, und ebenfalls den Schnitt S kreuzt. Da die leichte Kante bezüglich S eindeutig ist, folgt $c(e) < c(e')$, und damit ist $(T' \setminus \{e'\}) \cup \{e\}$ ein aufspannender Baum echt kleineren Gewichts als T', was ein Widerspruch zu unserer Annahme ist. Es kann also keine zwei verschiedenen minimalen Spannbäume in G geben. □

Die Umkehrung der Aussage gilt nicht. Der Graph aus Abbildung 1 hat einen eindeutigen minimalen Spannbaum. Der Schnitt $(\{x\}, \{y, z\})$ enthält aber die beiden leichten Kanten $\{x, y\}$ und $\{x, z\}$.

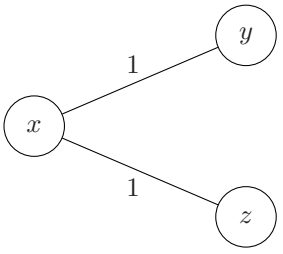
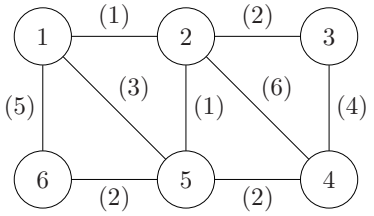


Abbildung 1: Ein Graph dessen MST eindeutig bestimmt ist, aber bei dem nicht jeder Schnitt eine eindeutige leichte Kante hat.

Problem 4: Stoer–Wagner 1pt

Es sei jeweils $G_i = (V_i, E_i)$ der Graph in Phase i. Anwendung des Algorithmus von Stoer & Wagner:

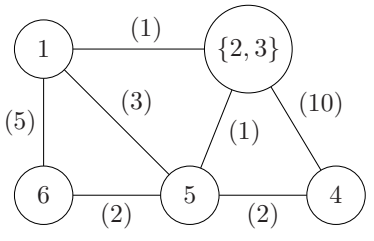
Phase 1: Anwenden von MinSchnittPhase mit Startknoten 1 auf den rechts abgebildeten Graphen liefert



$$S_1 = \{1\}$$
$$S_1 = \{1, 6\}$$
$$S_1 = \{1, 6, 5\}$$
$$S_1 = \{1, 6, 5, 4\}$$
$$S_1 = \{1, 6, 5, 4, 2\}$$
$$S_1 = V_1$$

Somit ist $s = 2$ und $t = 3$, der Schnitt der Phase ist $(V_1 \setminus \{3\}, \{3\})$ mit Gewicht 6. Verschmelze Knoten 3 und 2.

Phase 2: Anwenden von MinSchnittPhase mit Startknoten 2, in unserem Fall also $\{2, 3\}$ auf den rechts abgebildeten Graphen

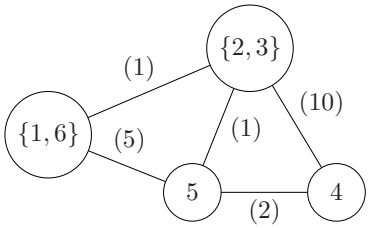


liefert

$$S_2 = \{\{2,3\}\}$$
$$S_2 = \{\{2,3\},4\}$$
$$S_2 = \{\{2,3\},4,5\}$$
$$S_2 = \{\{2,3\},4,5,1\}$$
$$S_2 = V_2$$

Somit ist $s = 1$ und $t = 6$, der Schnitt der Phase ist $(V_2 \setminus \{6\}, \{6\})$ mit Gewicht 7. Verschmelze Knoten 1 und 6.

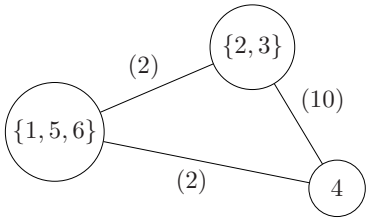
Phase 3: Anwenden von MinSchnittPhase mit Startknoten 3, in unserem Fall also wieder $\{2,3\}$ auf den rechts abgebildeten Graphen liefert



$$S_3 = \{\{2,3\}\}$$
$$S_3 = \{\{2,3\},4\}$$
$$S_3 = \{\{2,3\},4,5\}$$
$$S_3 = V_3$$

Somit ist $s = 5$ und $t = \{1,6\}$, der Schnitt der Phase ist $(V_3 \setminus \{\{1,6\}\}, \{\{1,6\}\})$ mit Gewicht 6. Verschmelze Knoten 5 und $\{1,6\}$.

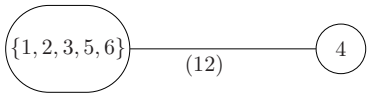
Phase 4: Anwenden von MinSchnittPhase mit Startknoten 4 auf den rechts abgebildeten Graphen liefert



$$S_4 = \{4\}$$
$$S_4 = \{4,\{2,3\}\}$$
$$S_4 = V_4$$

Somit ist $s = \{2,3\}$ und $t = \{1,5,6\}$, der Schnitt der Phase ist $(V_4 \setminus \{\{1,5,6\}\}, \{\{1,5,6\}\})$ mit Gewicht 4. Verschmelze Knoten $\{2,3\}$ und $\{1,5,6\}$.

Phase 5: Anwenden von MinSchnittPhase mit Startknoten 5, also $\{1,2,3,5,6\}$ auf den rechts abgebildeten Graphen liefert



$$S_5 = \{\{1,2,3,5,6\}\}$$
$$S_5 = V_5$$

Somit ist $s = \{1,2,3,5,6\}$ und $t = \{4\}$, der Schnitt der Phase ist $(V_5 \setminus \{4\}, \{4\})$ mit Gewicht 13.

Unter allen Schnitten ist der minimale Schnitt der Schnitt aus Phase 4 mit Gewicht 4. Dieser Schnitt $(V_4 \setminus \{\{1,5,6\}\}, \{\{1,5,6\}\})$ induziert auf G den Min–Schnitt $(\{\{2,3,4\},\{1,5,6\})$, siehe auch Abbildung 2.

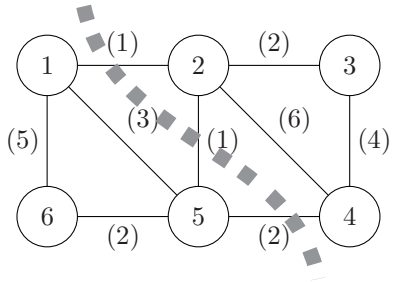


Abbildung 2: Graph G mit minimalem Schnitt