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

Übungsblatt 3 – Lösungsvorschläge

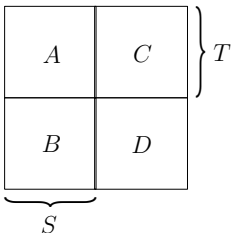
Vorlesung Algorithmentechnik im WS 09/10

Problem 1: Kreuzende Schnitte – Schnitte in Graphen [vgl. Kapitel 3 im Skript]

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A := S \cap T$, $B := S \setminus T$, $C := T \setminus S$ und $D := V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(a) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: Es gilt $s \in B$ und $t \in C$.

Lösung.



Per Definition gilt $s, t \notin D$. Außerdem gilt $t \notin S$, da sonst $(S, V \setminus S)$ kein s-t-Schnitt wäre. Analog gilt $s \notin T$ und insgesamt $s, t \notin A$. Damit bleibt nur noch $s \in B$ und $t \in C$.

(b) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende minimale s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind zwei kreuzungsfreie minimale s-t-Schnitte.

Lösung.

Sei $\lambda := c(S, V \setminus S) = c(T, V \setminus T)$. Mit $s \in B$ und $t \in C$ (vgl. (a)) sind $(B, V \setminus B)$ und $(C, V \setminus C)$ zwei kreuzungsfreie s-t-Schnitte, und es gilt $c(B, V \setminus B) \geq \lambda$ und $c(C, V \setminus C) \geq \lambda$.

$$\begin{aligned} c(B, V \setminus B) &= c(B, D) + c(B, C) + c(B, A) \\ &\geq c(B, D) + c(B, C) + c(A, D) + c(A, C) \\ &= c(S, V \setminus S) = \lambda \\ \Rightarrow c(B, A) &\geq c(A, D) + c(A, C) \end{aligned} \tag{1}$$

$$\begin{aligned} c(C, V \setminus C) &= c(C, D) + c(C, B) + c(C, A) \\ &\geq c(C, D) + c(C, B) + c(A, D) + c(A, B) \\ &= c(T, V \setminus T) = \lambda \\ \Rightarrow c(C, A) &\geq c(A, D) + c(A, B) \end{aligned} \tag{2}$$

Mit (2) eingesetzt in (1) gilt $c(B, A) \geq c(A, D) + c(A, C) \geq 2\, c(A, D) + c(A, B)$. Damit muss $c(A, D) = 0$ gelten. Mit (1) und (2) gilt dann $c(B, A) \geq c(A, C)$ und $c(C, A) \geq c(A, B)$. Damit gilt $c(B, A) = c(A, C)$ und insgesamt ergibt sich $c(B, V \setminus B) = c(C, V \setminus C) = \lambda$.

Problem 2: Algorithmus von Goldberg und Tarjan – Maximale Flüsse [vgl. Kapitel 4.2.3 im Skript]

Die Grundidee des Algorithmus von Goldberg und Tarjan basiert auf Präflüssen und PUSH- und RELABEL-Operationen. Algorithmen, die dieses Grundkonzept nutzen, nennt man daher auch *Preflow-Push-* oder *Push-Relabel*-Algorithmen.

Gegeben sei nun ein Flussnetzwerk $(D; s, t; c)$ bzw. dessen Erweiterung $(D'; s.t; c')$ und ein darin mit einem *Push-Relabel*-Algorithmus berechneter maximaler Fluss (d.h. die Abbildungen f, r_f, e und $dist$ stehen bezüglich des fertig berechneten Flusses zur Verfügung).

Entwickeln Sie einen möglichst schnellen Algorithmus um die Knotenpartition eines zugehörigen minimalen s-t-Schnitts (S, T) zu berechnen. Geben Sie Ihren Algorithmus in Pseudocode an. Begünden Sie die Laufzeit und die Korrektheit Ihres Algorithmus (Hinweis: Es gibt einen Algorithmus in $O(|V|)$).

Lösung.

Der Schlüssel liegt in den Eigenschaften der Abbildung $dist : V \to \mathbb{N}_0 \cup \{\infty\}$.

- **Behauptung:** Es gibt einen Wert $\hat{d} \in \mathbb{N}_0$ mit $0 < \hat{d} < |V|$ und $dist(v) \neq \hat{d} \quad \forall v \in V$.
  Beweis: Mit $dist(s) = |V|$ und $dist(t) = 0$ bleiben noch $|V| - 2$ Knoten zu prüfen. Diese können aber keine $|V| - 1$ verschiedenen Werte für $\hat{d}$ abdecken. Es gibt also mindestens einen Wert $0 < \hat{d} < |V|$ mit $dist(v) \neq \hat{d} \quad \forall v \in V$.

- **Behauptung:** Die Partition (S, T) mit $S := \{u \in V \,|\, dist(u) > \hat{d}\}$ und $T := \{v \in V \,|\, dist(v) < \hat{d}\}$ induziert einen zugehörigen minimalen s-t-Schnitt.
  Beweis: Mit $dist(s) = |V|$ und $dist(t) = 0$ gilt $s \in S$ und $t \in T$. Nach Definition 4.4 ist (S, T) also ein s-t-Schnitt.

  Zeige nun für alle Kanten (u, v) mit $u \in S, v \in T$, dass (u, v) saturiert, also keine Residualkante ist. Dann folgt aus Lemma 4.5 (Schnitt-Lemma), dass (S, T) ein **minimaler** s-t-Schnitt ist. [Denn der Wert $w(f)$ des maximalen Flusses ist unabhängig von der Wahl des s-t-Schnittes (vgl. Definition 4.3). Einen s-t-Schnitt mit kleinerem Gewicht als $w(f)$ kann es also nicht geben. Unter der Voraussetzung saturierter Kanten (u, v) mit $u \in S, v \in T$ gilt jedoch $w(f) = \sum_{\substack{(u,v)\in E \\ u\in S, v\in T}} f(u, v) - \sum_{\substack{(v,u)\in E \\ u\in S, v\in T}} f(v, u) = \sum_{\substack{(u,v)\in E \\ u\in S, v\in T}} f(u, v) - 0 = \sum_{\substack{(u,v)\in E \\ u\in S, v\in T}} c(u, v) = c(S, T)$. Damit ist (S, T) minimal.]

  Angenommen es gäbe eine nicht saturierte Kante (u, v) mit $u \in S, v \in V$. Dann gilt $(u, v) \in E_f$ und $dist(u) \leq dist(v) + 1$ (nach Definition 4.18). Es gilt aber auch $dist(v) < \hat{d} < dist(u)$ und somit $dist(v) \leq dist(u) - 2$. Daraus folgt $dist(u) \leq dist(u) - 1$, was einen Widerspruch darstellt. Damit sind alle Kanten (u, v) mit $u \in S, v \in T$ saturiert.

Obige Ausfürungen gelten als Korrektheitsbeweis. Anhand des folgenden Pseudocodes lässt sich die Laufzeit ablesen. Diese ergibt sich zu 3 $O(|V|) = O(|V|)$.

Algorithmus 1 : getPartition

Eingabe : Knotenmenge V des Netzwerks, Abbildung $dist$
Ausgabe : Partition (S, T) eines minimalen s-t-Schnitts

```
1  A[1, ..., |V| − 1] ← FALSE                          // initialisiere Array der
                                                       // |V| − 1 möglichen Werte für d̂
2  S ← {s}, T ← {t}
3  für alle Knoten v ∈ V \ {s, t} tue                   // O(|V|)
4  |  A[dist(v)] ← TRUE
5  Suche Index d̂ mit A[d̂] = FALSE                      // O(|V|)
6  für alle Knoten v ∈ V \ {s, t} tue                   // O(|V|)
7  |  if dist(v) > d̂ then
8  |  |  S ← S ∪ {v}                                   // O(1)
9  |  else
10 |  |  T ← T ∪ {v}                                   // O(1)
11 Return (S, T)
```

Bemerkung: Die Partition (S, T) mit $S := \{u \in V \mid dist(u) \geq |V|\}$ und $T := \{v \in V \mid dist(v) < |V|\}$ induziert im Allgemeinen **KEINEN** minimalen s-t-Schnitt! (siehe Gegenbeispiel auf der Vorlesungshomepage; verfügbar ab 08.12.09)

Problem 3: Gomory-Hu-Bäume – Schnitte in Graphen [vgl. Kapitel 3 im Skript]

Gegeben sei ein ungerichteter, nicht-negativ gewichteter, zusammenhängender Graph $G = (V, E)$. Ein zu G gehöriger Gomory-Hu-Baum ist ein ungerichteter, nicht-negativ gewichteter, zusammenhängender Baum $T(G)$ über der Knotenmenge V so, dass gilt:

(a) jede Kante $\{u, v\}$ in $T(G)$ induziert einen minimalen u-v-Schnitt in G, indem $T(G)$ beim Entfernen von $\{u, v\}$ in zwei Teilbäume zerfällt, deren Knotenmengen gerade die beiden Seiten des Schnitts darstellen.

(b) das Gewicht einer Kante $\{u, v\}$ in $T(G)$ entspricht dem Gewicht des durch $\{u, v\}$ induzierten minimalen u-v-Schnitts in G.

Bemerkung: Die Kantemenge eines Gomory-Hu-Baumes $T(G)$ muss **keine** Teilmenge der Kantenmenge des zugrundeliegenden Graphen G sein. Zu jedem ungerichteten, nicht-negativ gewichteten, zusammenhängenden Graphen existiert mindestens ein Gomory-Hu-Baum.

Zeigen Sie: Für zwei beliebige Knoten $s, t \in V$ induziert eine auf dem s-t-Pfad in $T(G)$ minimal gewichtete Kante einen minimalen s-t-Schnitt in G.

Lösung.

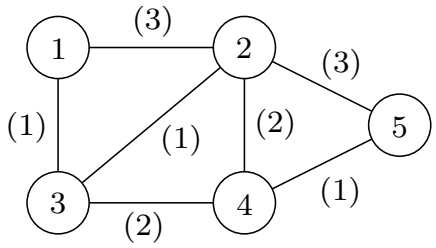
Sei $\{u, v\}$ eine minimal gewichtete Kante auf dem s-t-Pfad p in $T(G)$ mit $c(\{u, v\}) = \lambda$ und $(U, V \backslash U)$ sei der durch $\{u, v\}$ in G induzierte minimale u-v-Schnitt mit $u, s \in U$. Dann ist $(U, V \setminus U)$ auch ein s-t-Schnitt.

Zeige, jeder s-t-Schnitt in G teilt ein in $T(G)$ adjazentes Knotenpaar auf dem Pfad p.
Annahme: Sei $(S, V \setminus S)$ mit $s \in S$ ein s-t-Schnitt, der **kein** in $T(G)$ adjazentes Knotenpaar auf dem Pfad $p = (s, p_1 \ldots, p_k, t)$ teilt. So gilt mit $s \in S$ auch $p_1 \in S$ und somit $p_2 \in S$ usw. Schließlich gilt für den gesamten Pfad $p \subset S$ und damit ist $(S, V \setminus S)$ im Widerspruch zur Annahme kein s-t-Schnitt!

Sein nun $(S, V \setminus S)$ mit $s \in S$ ein s-t-Schnitt in G mit $c(S, V \setminus S) < \lambda$, so teilt, wie eben gezeigt, dieser Schnitt $(S, V \setminus S)$ mindestens ein in $T(G)$ adjazentes Knotenpaar $\{p_i, p_{i+1}\}$ auf p und ist somit auch ein p_i-p_{i+1}-Schnitt in G. Damit gilt $c(S, V \setminus S) \geq c(\{p_i, p_{i+1}\}) \geq c(\{u, v\}) = \lambda$, was einen Widerspruch zur Annahme $c(S, V \setminus S) < \lambda$ darstellt. Der durch die minimal gewichtete Kante $\{u, v\}$ induzierte s-t-Schnitt $(U, V \setminus U)$ mit $c(U, V \setminus U) = \lambda$ ist also minimal.

Problem 4: Algorithmus von Stoer & Wagner – Schnitte in Graphen [vgl. Kapitel 3.2 im Skript]

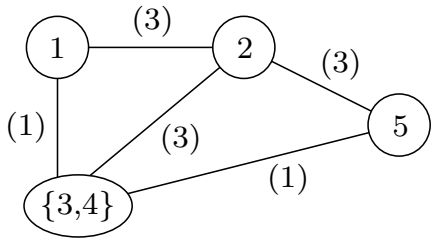
(a) Wenden Sie auf den unten abgebildeten Graphen (Kantengewichte in Klammern) den Algorithmus von Stoer & Wagner an. Geben Sie nach jeder Phase die Knoten s und t, den Schnitt der Phase und dessen Gewicht an und zeichnen Sie den nach dem Verschmelzen resultierenden Graphen (mit Kantengewichten). **Verwenden Sie in Phase i den Knoten als Startknoten, der Knoten i des Originalgraphen enthält.** Geben Sie zum Schluss den minimalen Schnitt S_{min} an.



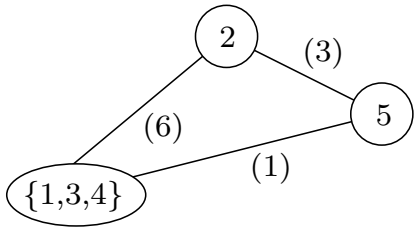
(b) Liefert der Algorithmus von Stoer & Wagner auch für negative Kantengewichte einen global minimalen, nichttrivialen Schnitt? Begründen Sie Ihre Antwort.

Lösung.

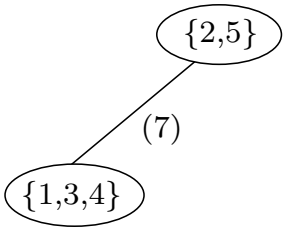
Ein minimaler Schnitt S_{min} ist der Schnitt der Phase 1: $c(S_{min}) = c(\{1, 2, 4, 5\}, 3) = 4$.



(a) Graph nach Phase 1, $S_1 = 1, 2, 5, 4$, $s = 4$, $t = 3$, $c(\{1, 2, 4, 5\}, 3) = 4$.



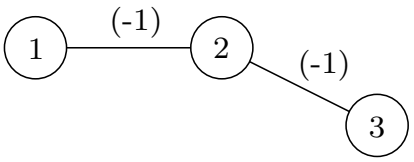
(b) Graph nach Phase 2, $S_2 = 2, 5, \{3, 4\}$, $s = \{3, 4\}$, $t = 1$, $c(\{2, 3, 4, 5\}, 1) = 4$.



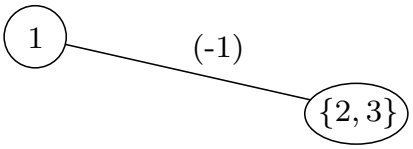
(c) Graph nach Phase 3, $S_3 = \{1, 3, 4\}, 2$, $s = 2$, $t = 5$, $c(\{1, 2, 3, 4\}, 5) = 4$.

(d) Nach Phase 4, $s = \{1, 3, 4\}$, $t = \{2, 5\}$, $c(\{1, 3, 4\}, \{2, 5\}) = 7$.

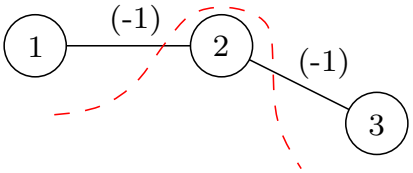
Für negative Kanten liefert der Algorithmus von Stoer & Wagner im Allgemeinen **KEINEN** global minimalen, nichttrivialen Schnitt, wie untenstehendes Gegenbeispiel zeigt: Hier berechnet der Algorithmus von Stoer & Wagner einen minimalen Schnitt mit Gewicht -1, der tatsächliche minimale Schnitt hat aber Gewicht -2.



(a) Graph vor Phase 1.



(b) Graph nach Phase 1, $S_1 = 1, 2$, $s = 2$, $t = 3$, $c(\{1, 2\}, 3) = -1$.



(c) Nach Phase 2, $S_2 = 1$, $s = 1$, $t = \{2, 3\}$, $c(1, \{2, 3\}) = -1$.

(d) ABER: Tatsächlich minimaler Schnitt ist $(\{1, 3\}, 2)$ mit Gewicht -2.