Karlsruher Institut für Technologie

Algorithmentechnik - Übung 3

4. Sitzung

Tanja Hartmann | 03. Dezember 2009

INSTITUT FÜR THEORETISCHE INFORMATIK, PROF. DR. DOROTHEA WAGNER

Kreuzende Schnitte - Problem 1 [Kap. 3]

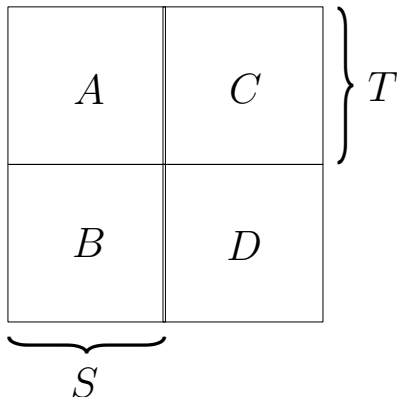
(Schnitte in Graphen)

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ **kreuzen sich**, wenn keine der Mengen
$A := S \cap T$,
$B := S \setminus T$,
$C := T \setminus S$,
$D := V \setminus (S \cup T)$
leer ist.
Sei $c : E \to R_0^+$ eine Kantengewichtsfunktion auf G.

Kreuzende Schnitte - Problem 1 [Kap. 3]

(Schnitte in Graphen)

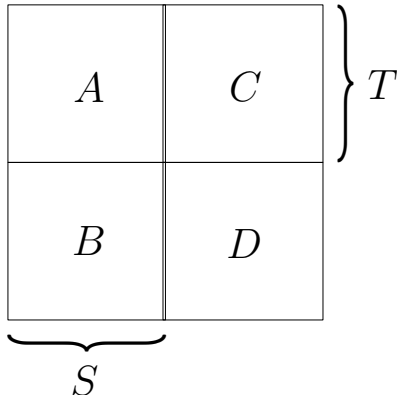
Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ **kreuzen sich**, wenn keine der Mengen
$A := S \cap T$,
$B := S \setminus T$,
$C := T \setminus S$,
$D := V \setminus (S \cup T)$
leer ist.
Sei $c : E \to R_0^+$ eine Kantengewichtsfunktion auf G.



Kreuzende Schnitte - Problem 1 [Kap. 3]

(Schnitte in Graphen)

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ **kreuzen sich**, wenn keine der Mengen
$A := S \cap T$,
$B := S \setminus T$,
$C := T \setminus S$,
$D := V \setminus (S \cup T)$
leer ist.
Sei $c : E \to R_0^+$ eine Kantengewichtsfunktion auf G.



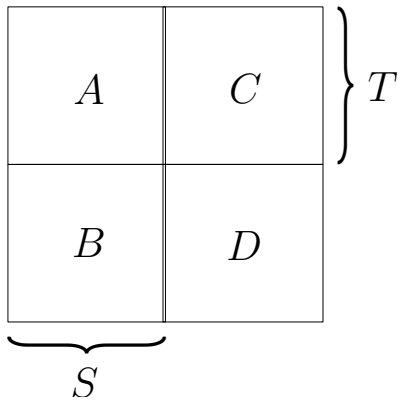
(a) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende s-t-Schnitte mit $s \in S$ und $t \in T$.
Zeigen Sie: Es gilt $s \in B$ und $t \in C$.

Beweis an der Tafel...

Kreuzende Schnitte - Problem 1 [Kap. 3]

(Schnitte in Graphen)

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ **kreuzen sich**, wenn keine der Mengen
$A := S \cap T$,
$B := S \setminus T$,
$C := T \setminus S$,
$D := V \setminus (S \cup T)$
leer ist.
Sei $c : E \to R_0^+$ 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 $t \in C$.

Beweis an der Tafel...

(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.

Beweis an der Tafel...

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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.

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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).

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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 $O(|V|)$).

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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 $O(|V|)$).

Grundlage: Dualität von maximalen Flüssen und minimalen Schnitten (in Flussnetzwerken)

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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 $O(|V|)$).

Grundlage: Dualität von maximalen Flüssen und minimalen Schnitten (in Flussnetzwerken)

Definition (gerichteter minimaler s-t-Schnitt)

Ein (gerichteter) minimaler s-t-Schnitt in einem Flussnetzwerk ist eine Partition (S, T) der Knotenmenge mit $s \in S$ und $t \in T$, so dass das summierte (gerichtete) Kantengewicht $c(S, T) = \sum_{\substack{u \in S \\ v \in T}} c(u, v)$ minimal ist unter all solchen Partitionen.

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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 $O(|V|)$).

Grundlage: Dualität von maximalen Flüssen und minimalen Schnitten (in Flussnetzwerken)

Definition (gerichteter minimaler s-t-Schnitt)

Ein (gerichteter) minimaler s-t-Schnitt in einem Flussnetzwerk ist eine Partition (S, T) der Knotenmenge mit $s \in S$ und $t \in T$, so dass das summierte (gerichtete) Kantengewicht $c(S, T) = \sum_{\substack{u \in S \\ v \in T}} c(u, v)$ minimal ist unter all solchen Partitionen.

$\Rightarrow (S, T)$ ist eine Art richtungsbezogene Engstelle/Engpass

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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 $O(|V|)$).

Was tut Preflow-Push-Algorithmus (extrem vereinfacht)?

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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 $O(|V|)$).

Was tut Preflow-Push-Algorithmus (extrem vereinfacht)?

- speist maximal mögliche Menge an “Fluss” bei s ein

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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 $O(|V|)$).

Was tut Preflow-Push-Algorithmus (extrem vereinfacht)?

- speist maximal mögliche Menge an “Fluss” bei s ein
- Fluss fließt Richtung t (PUSH), was nicht durch engste Stelle passt fließt zurück zu s (RELABEL)

$\Rightarrow$ Kanten an engster Stelle sind **saturiert!**

$\Rightarrow$ Jeder Schnitt durch saturierte Kanten ist minimal.

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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 $O(|V|)$).

Algo-Idee: Nutze Eigenschaft 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 an der Tafel...)
- **Behauptung:** Die Partition (S, T) mit $S := \{u \in V \mid dist(u) > \hat{d}\}$ und $T := \{v \in V \mid dist(v) < \hat{d}\}$ induziert minimalen s-t-Schnitt.
  (Beweis an der Tafel...)

Goldberg u. Tarjan - Problem 2 [Kap. 4.2.3]

(Maximale Flüsse)

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)
```

Gomory-Hu-Bäume - Problem 3 [Kap. 3]

(Schnitte in Graphen)

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:

Gomory-Hu-Bäume - Problem 3 [Kap. 3]

(Schnitte in Graphen)

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:

- 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.

Gomory-Hu-Bäume - Problem 3 [Kap. 3]

(Schnitte in Graphen)

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:

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

Gomory-Hu-Bäume - Problem 3 [Kap. 3]

(Schnitte in Graphen)

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:

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

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.

Beweis an der Tafel...

Stoer & Wagner - Problem 4 [Kap. 3.2]

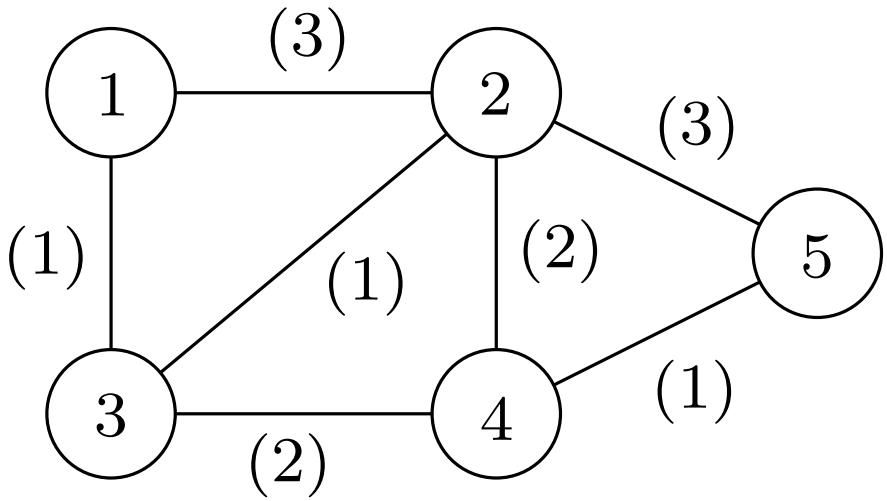
(Schnitte in Graphen)

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.

Stoer & Wagner - Problem 4 [Kap. 3.2]

(Schnitte in Graphen)

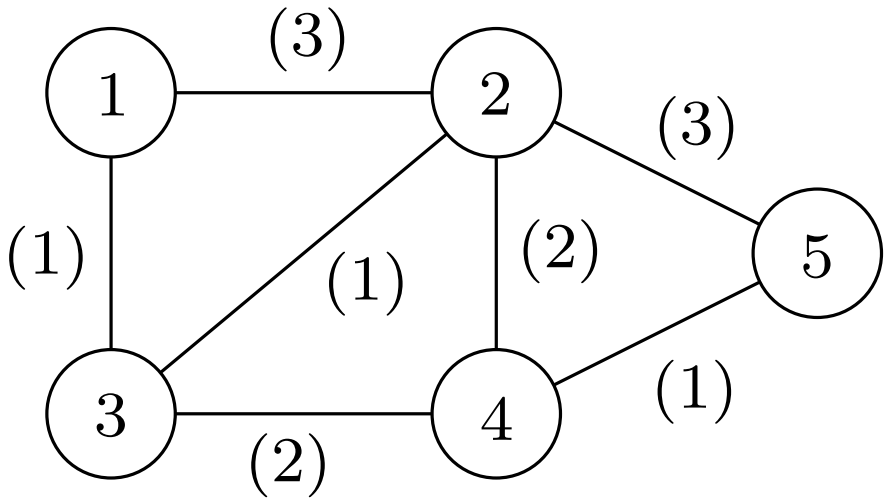
Wenden Sie auf den unten abgebildeten Graphen (Kantengewichte in Klammern) den Algorithmus von Stoer & Wagner an. Geben Sie nach jeder Phase die Knoten *s* und *t*, den Schnitt der Phase und dessen Gewicht an und zeichnen Sie den nach dem Verschmelzen resultierenden Graphen (mit Kantengewichten). **Verwenden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



Stoer & Wagner - Problem 4 [Kap. 3.2]

(Schnitte in Graphen)

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.



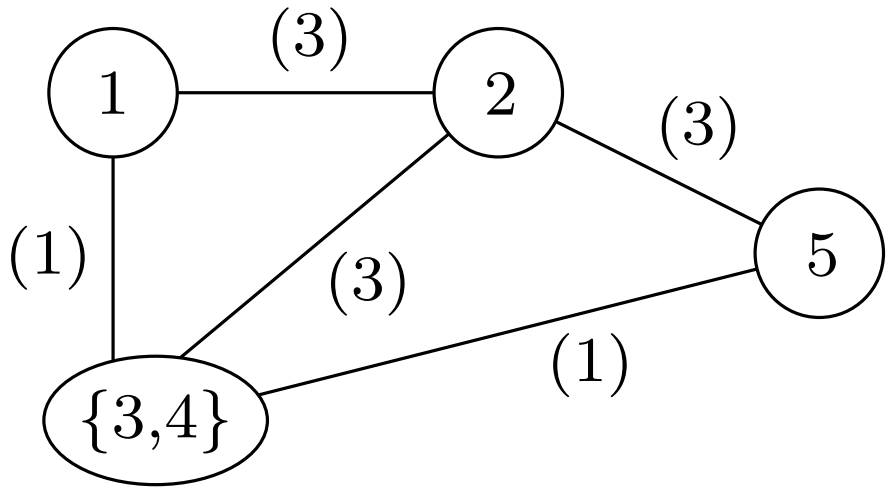
$S_1 = 1, 2, 5, 4,\ s = 4,\ t = 3$

Schnitt der Phase 1: $c(\{1, 2, 4, 5\}, 3) = 4.$

Stoer & Wagner - Problem 4 [Kap. 3.2]

(Schnitte in Graphen)

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.



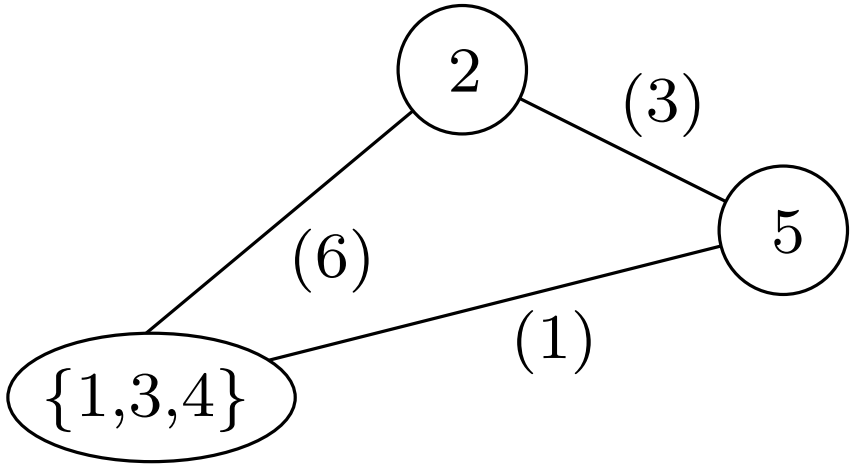
$S_2 = 2, 5, \{3, 4\}$, $s = \{3, 4\}$, $t = 1$

Schnitt der Phase 2: $c(\{2, \{3, 4\}, 5\}, 1) = 4.$

Stoer & Wagner - Problem 4 [Kap. 3.2]

(Schnitte in Graphen)

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.



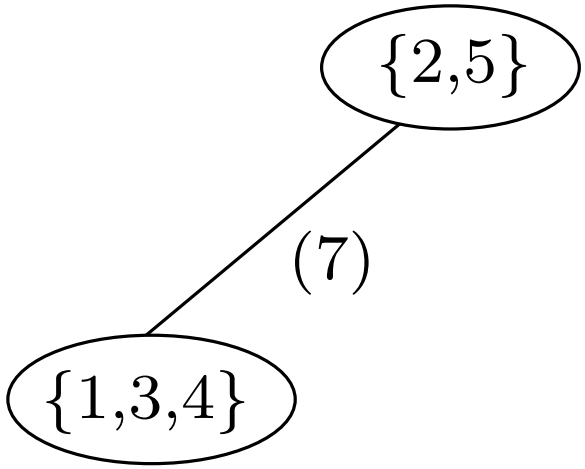
$S_3 = \{1, 3, 4\}, 2,\ s = 2,\ t = 5$

Schnitt der Phase 3: $c(\{\{1, 3, 4\}, 2\}, 5) = 4.$

Stoer & Wagner - Problem 4 [Kap. 3.2]

(Schnitte in Graphen)

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.

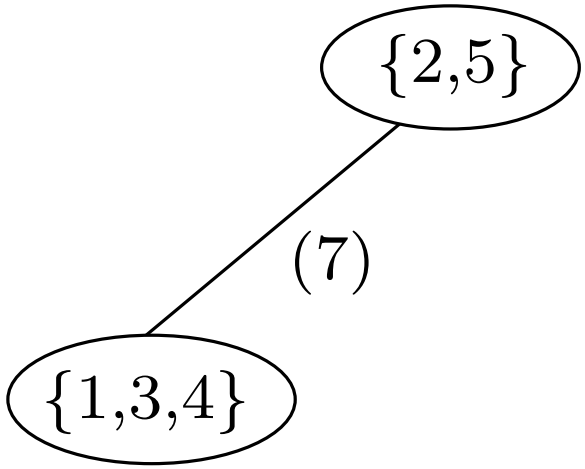


$S_4 = \{1, 3, 4\}$, $s = \{1, 3, 4\}$, $t = \{2, 5\}$
Schnitt der Phase 4: $c(\{\{1, 3, 4\}, \{2, 5\}) = 7.$

Stoer & Wagner - Problem 4 [Kap. 3.2]

(Schnitte in Graphen)

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.



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

Stoer & Wagner - Problem 4 [Kap. 3.2]

(Schnitte in Graphen)

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.

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

Gegenbeispiel an der Tafel...