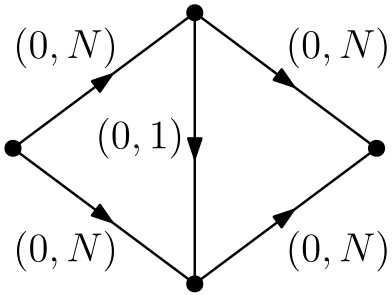
Technische Universität Braunschweig

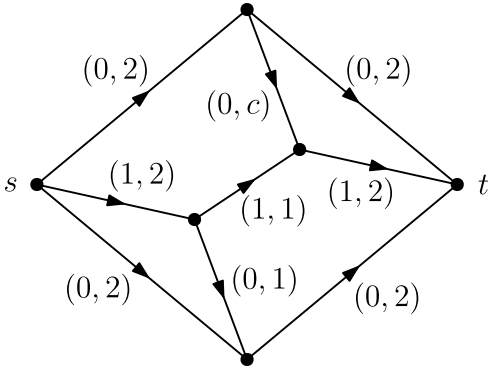
Netzwerkalgorithmen – Vorlesung #9

Arne Schmidt

Wiederholung

Wiederholung





Satz 4.12
Sei $N = (D, u, s, t)$ ein Netzwerk und f ein st-Fluss in N.
Dann gibt es eine Menge
- $\mathcal{P}$ von st-Pfaden
- $\mathcal{C}$ von Kreisen

sodass folgendes gilt:
- $f(e) = \sum_{\substack{p \in \mathcal{P} \cup \mathcal{C} \\ e \in p}} w(p)$
- $Wert(f) = \sum_{p \in \mathcal{P}} w(p)$
- $|\mathcal{P}| + |\mathcal{C}| \leq |E|$

Dabei ist $w(p)$ der Flusswert des Pfades bzw. Kreises.

Korollar 4.14
Für ein Netzwerk $N = (D, u, s, t)$ mit $u: A \to \mathbb{Q}^+$ maximalem Flusswert f^* benötigt der Algorithmus von Ford-Fulkerson maximal $m \log f^*$ Iterationen, wenn immer ein Pfad mit maximaler Verbesserung gewählt wird.

4.2 Berechnung von maximalen Flüssen

Algorithmus von Edmonds-Karp

Algorithmus 4.15
Eingabe:
Netzwerk $N = (D, u, s, t)$
Ausgabe:
st-Fluss f mit maximalem Wert

1. Function EDMONDSKARP(N)
2. $f(e) \coloneqq 0, \forall e \in A$
3. Bestimme Residualgraph D_f und Residualkapazitäten u_f.
4. Bestimme **kürzesten** st-Pfad P in D_f; falls keiner existiert **return** f.
5. Berechne $\gamma \coloneqq \min_{e \in P} \left(u_f(e) \right)$.
6. Augmentiere f entlang P.
7. Gehe zu Zeile 3.

Analyse Edmonds-Karp

Lemma 4.16
Sei $N = (D, u, s, t)$ ein Netzwerk und $f_1, \dots, f_k$ eine Folge von Flüssen, wobei f_{i+1} aus f_i durch Augmentieren entlang eines kürzesten f_i-augmentierenden Pfades P_i entsteht. Dann gilt:

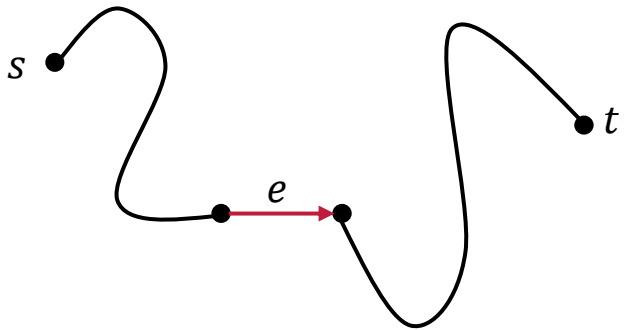
a) Für alle $1 \leq i < k$ gilt $|E(P_i)| \leq |E(P_{i+1})|$

b) Falls P_i und P_j mit $i < j$ ein Paar entgegengesetzter Kanten enthält, dann gilt $|E(P_i)| \leq |E(P_j)| - 2$

Lemma 4.17
Für ein Netzwerk $N = (D, u, s, t)$ mit $u: A \to \mathbb{R}^+$ terminiert der Algorithmus von Edmonds-Karp nach maximal $\frac{mn}{2}$ Iterationen.

Analyse Edmonds-Karp

Lemma 4.17
Für ein Netzwerk $N = (D, u, s, t)$ mit $u: A \to \mathbb{R}^+$ terminiert der Algorithmus von Edmonds-Karp nach maximal $\frac{mn}{2}$ Iterationen.



Jeder augm. Pfad besitzt Bottleneckkante e.
$\Rightarrow$ Nach Augmentieren existiert $\tilde{e}$ in D_f.

Fragen:
- Wann wird e wieder benutzt?
- Wie oft kann e als Bottleneckkante genutzt werden?

Analyse Edmonds-Karp

Lemma 4.17
Für ein Netzwerk $N = (D, u, s, t)$ mit $u: A \to \mathbb{R}^+$ terminiert der Algorithmus von Edmonds-Karp nach maximal $\frac{mn}{2}$ Iterationen.

Lemma 4.16
Sei $N = (D, u, s, t)$ ein Netzwerk und $f_1, \dots, f_k$ eine Folge von Flüssen, wobei f_{i+1} aus f_i durch Augmentieren entlang eines kürzesten f_i-augmentierenden Pfades P_i entsteht. Dann gilt:

a) Für alle $1 \leq i < k$ gilt $|E(P_i)| \leq |E(P_{i+1})|$

b) Falls P_i und P_j mit $i < j$ ein Paar entgegengesetzter Kanten enthält, dann gilt $|E(P_i)| \leq |E(P_j)| - 2$

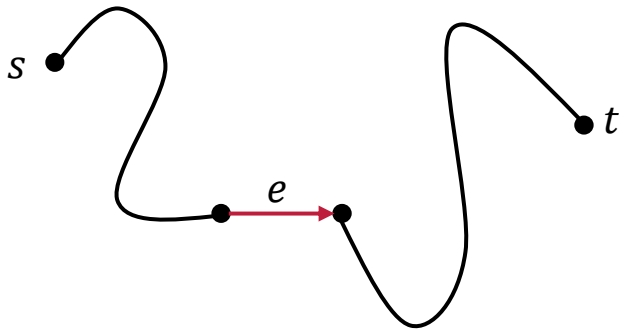
Betrachte augm. Pfade P_i, P_j, die e als Bottleneckkante benutzen.
$\Rightarrow$ Es existiert ein Pfad P_ℓ, welcher $\overleftarrow{e}$ in D_f benutzt.

Nach Lemma 4.16:

$$|E(P_i)| \leq |E(P_\ell)| - 2 \leq |E(P_j)| - 4$$

Analyse Edmonds-Karp

Lemma 4.17
Für ein Netzwerk $N = (D, u, s, t)$ mit $u: A \to \mathbb{R}^+$ terminiert der Algorithmus von Edmonds-Karp nach maximal $\frac{mn}{2}$ Iterationen.



Nach Lemma 4.16:

$$|E(P_i)| \leq |E(P_\ell)| - 2 \leq |E(P_j)| - 4$$

Jeder augm. Pfad besitzt maximal n Kanten.
$\Rightarrow e$ ist maximal $\frac{n}{4}$ mal eine Bottleneckkante.

Für $2|E| = 2m$ Vor- und Rückwärtskanten benötigen wir also maximal $\frac{mn}{2}$ Iterationen.

Analyse Edmonds-Karp

Lemma 4.17
Für ein Netzwerk $N = (D, u, s, t)$ mit $u: A \to \mathbb{R}^+$ terminiert der Algorithmus von Edmonds-Karp nach maximal $\frac{mn}{2}$ Iterationen.

Satz 4.18
Der Algorithmus von Edmonds-Karp löst Problem 4.2 (MaxFlow) in Zeit $O(m^2 n)$.

In jeder Iteration wird der kürzeste augm. Pfad in Zeit $O(m)$ mit BFS gefunden.
$\Rightarrow$ mit $O(mn)$ Iterationen: Gesamtlaufzeit $O(m^2 n)$


Geht das schneller?

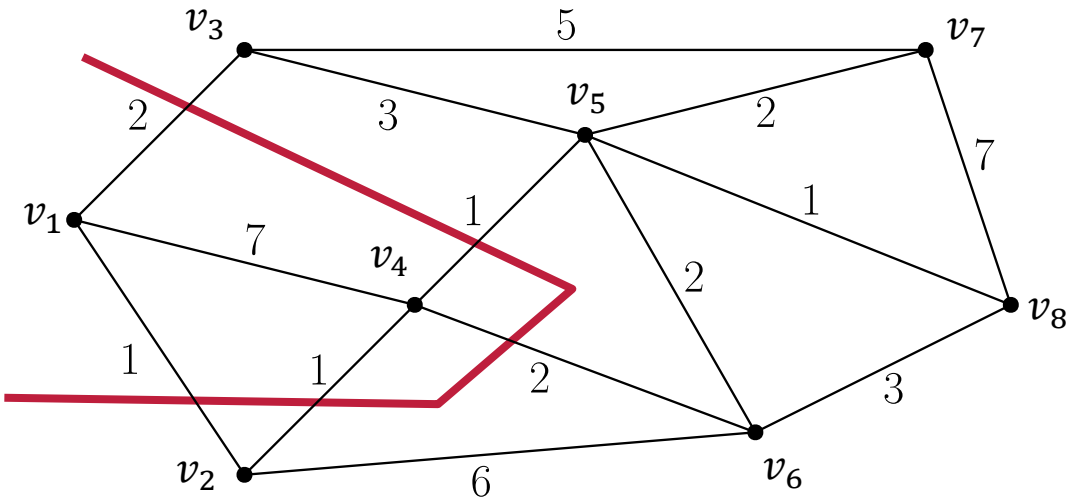
Algorithmen für MaxFlow (https://en.wikipedia.org/wiki/Maximum_flow_problem#Algorithms)

Algorithmus	Laufzeit	Bemerkung
Ford-Fulkerson	$O(mf_{\max})$	Terminiert nicht immer für reelle Kapazitäten.
Edmonds-Karp	$O(m^2 n)$	
Dinic	$O(mn^2)$	
Malhotra, Kumar, Maheshwari	$O(n^3)$	Modifikation von Dinic
Dinic+	$O(mn \log n)$	Dinic mit „Link/Cut Trees“
Push-Relabel	$O\left(mn \log \frac{n^2}{m}\right)$	
Orlin + KRT	$O(mn)$	Mischung zweier Algorithmen

4.3 Bottlenecks in Netzwerken

Globaler Minimaler Schnitt

Problem 4.19: Globaler Minimaler Schnitt (Global MinCut)
Gegeben:
Ungerichteter Graph $G = (V, E)$ und Kostenfunktion $c: E \to \mathbb{R}^+$
Gesucht:
Nicht-leere Menge $X \subsetneq V$ mit $\sum_{e \in E(X, V \setminus X)} c(e)$ minimal.

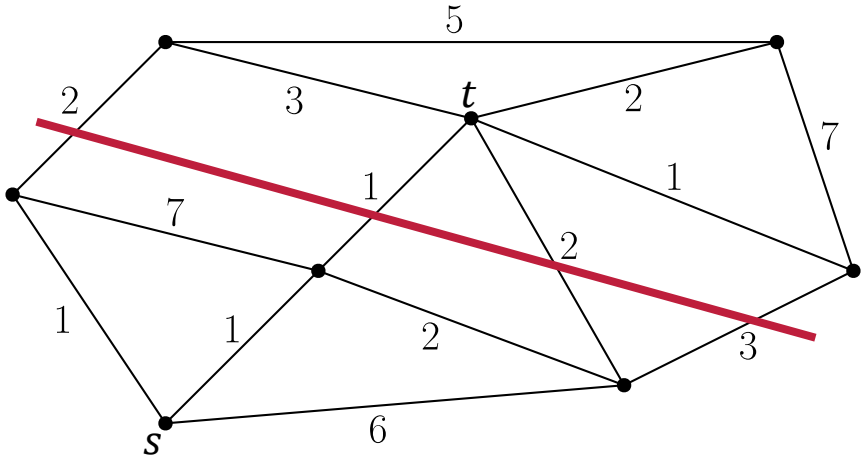


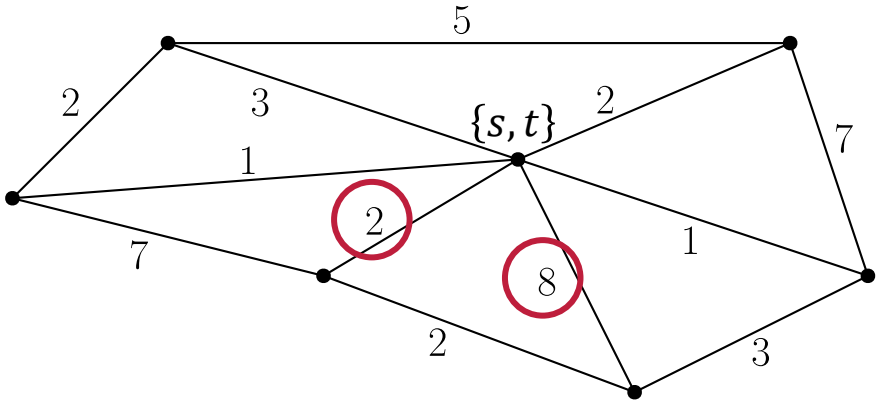
Idee

Suche zwischen zwei Knoten s, t einen MinCut.
Zwei Fälle:

- Das ist der Globale MinCut.
- Oder s und t liegen im Cut auf der gleichen Seite.

$\Rightarrow$ Betrachte Graphen mit s und t verschmolzen.





Algorithmus von Stoer-Wagner

Algorithmus 4.20

Eingabe:

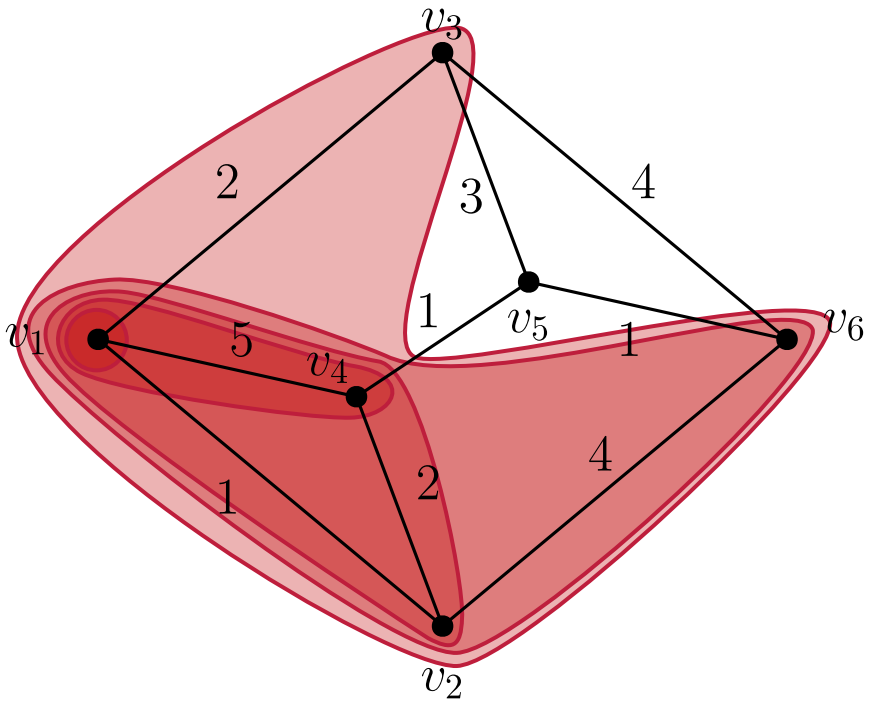
Ungerichteter Graph $G = (V, E)$ und Kostenfunktion $c: E \to \mathbb{R}^+$

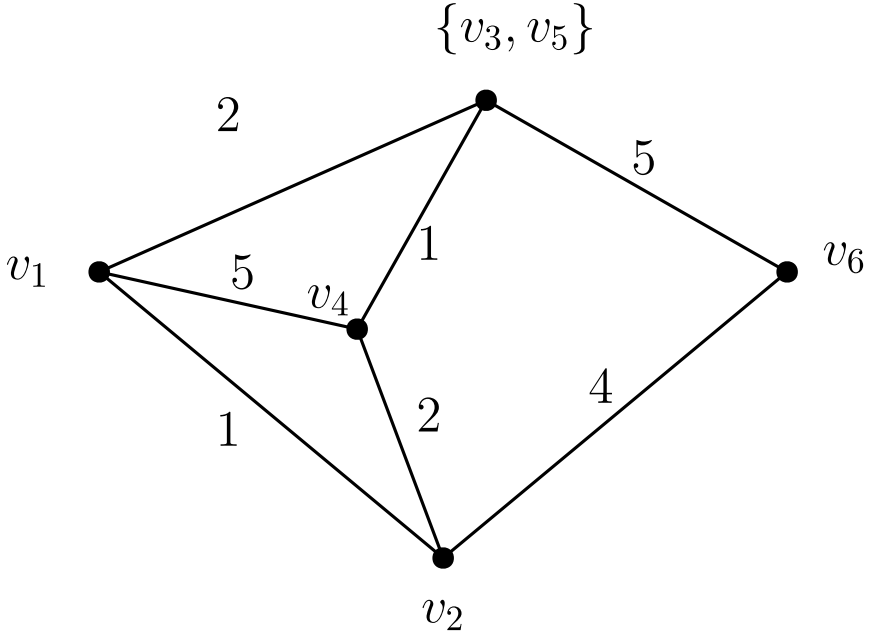
Ausgabe:

Nicht-leere Menge $X \subsetneq V$ mit $\sum_{e \in E(X, V \setminus X)} c(e)$ minimal.

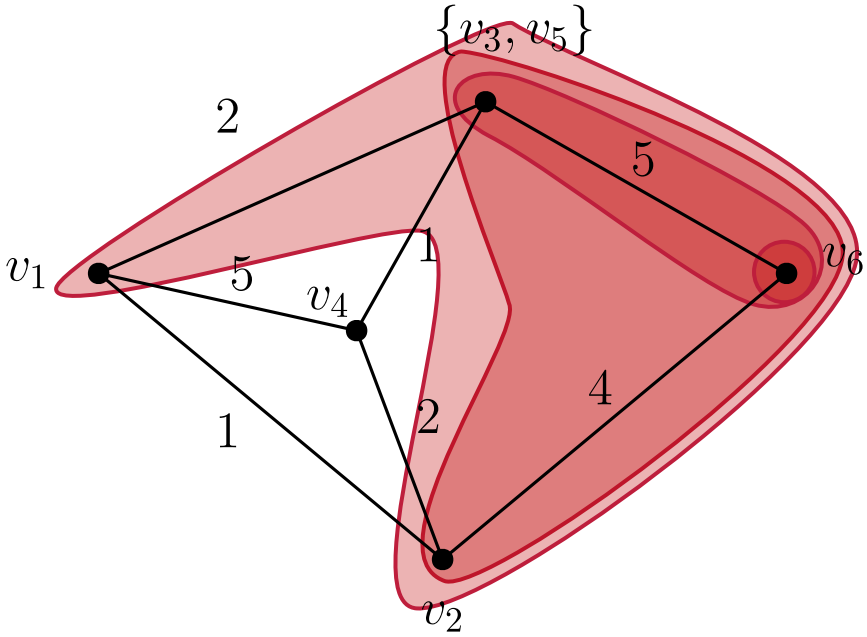
1. Function StoerWagner(G, c)
2. Wähle $v \in V$ und setze $X := \{v\}$
3. While $(X \neq V)$
4. Wähle $w \in V \setminus X$ mit $\sum_{\substack{v \in X \\ e=\{v,w\}}} c(e)$ maximal
5. Sei s (bzw. t) der Knoten der als vorletztes (bzw. letztes) zu X hinzugefügt wurde.
6. Speichere Wert des Schnittes $(V \setminus \{t\}, \{t\})$.
7. Verschmelze s und t und gehe zu Zeile 2.; Falls $|V| = 1$ **return** kleinsten Schnitt.

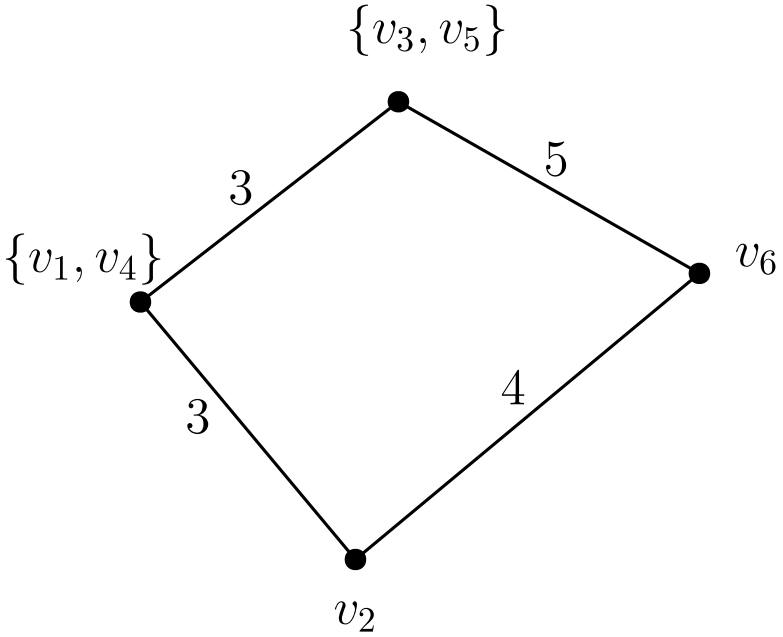
Beispiel



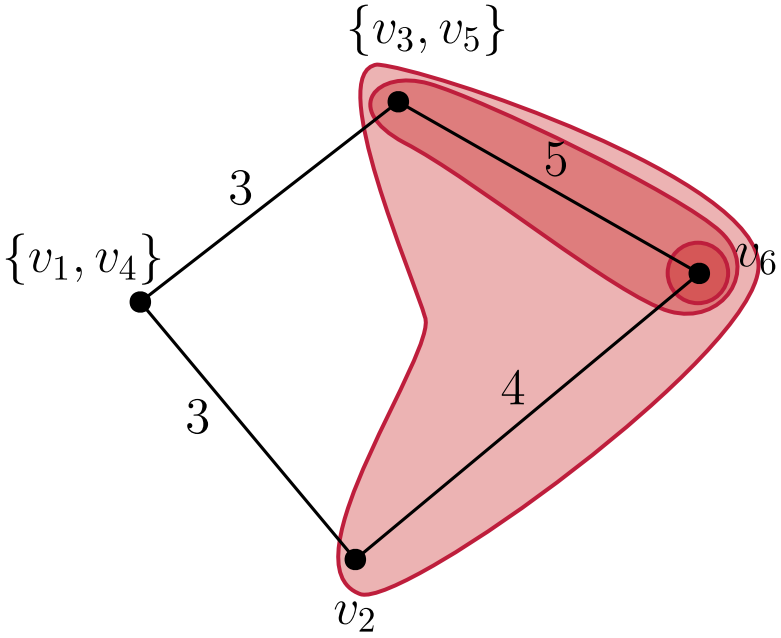


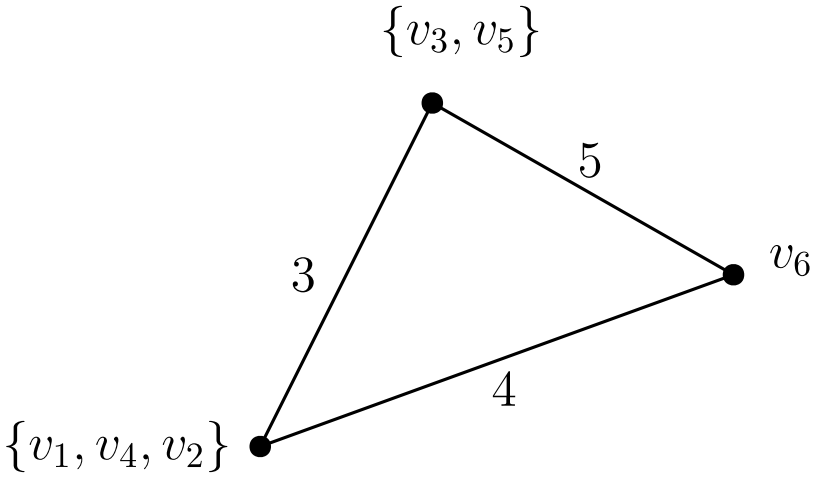
Beispiel



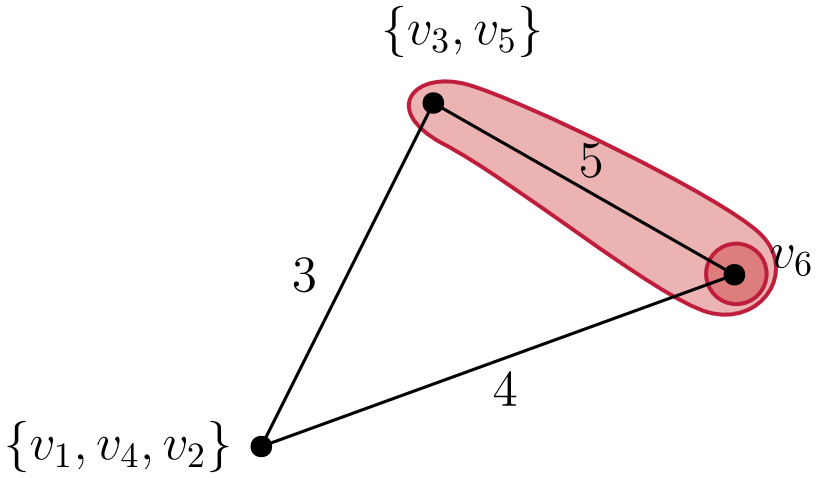


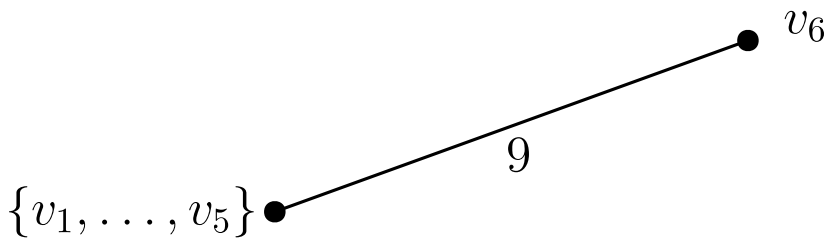
Beispiel



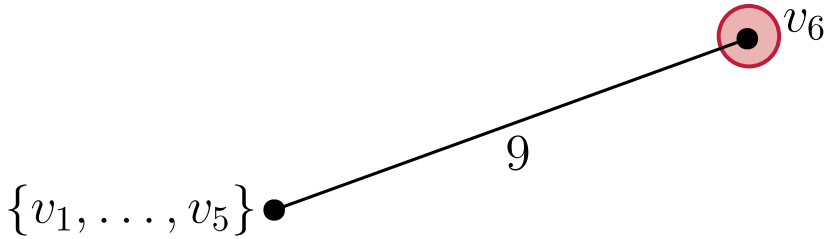


Beispiel





Beispiel



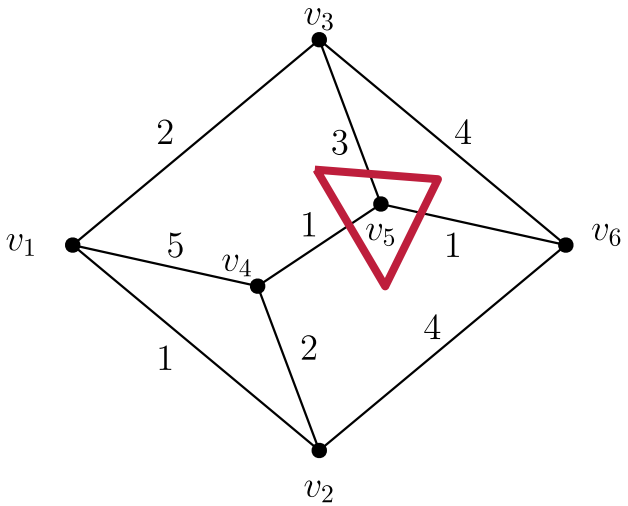
Gefundene Schnitte:

$(V \setminus \{v_5\}, \{v_5\})$: Wert 5

$(V \setminus \{v_4\}, \{v_4\})$: Wert 8

$\left(V \setminus \{\{v_1, v_4\}\}, \{\{v_1, v_4\}\}\right)$: Wert 6

$(V \setminus \{v_6\}, \{v_6\})$: Wert 9



Analyse

Lemma 4.21
Der in Zeilen 3 bis 6 berechnete Schnitt $(V \setminus \{t\}, \{t\})$ ist ein minimaler st-Schnitt.

Algorithmus 4.20
Eingabe:
Ungerichteter Graph $G = (V, E)$ und Kostenfunktion $c: E \to \mathbb{R}^+$
Ausgabe:
Nicht-leere Menge $X \subsetneq V$ mit $\sum_{e \in E(X, V \setminus X)} c(e)$ minimal.

1. Function EdmondsKarp(G, c)
2. Wähle $v \in V$ und setze $X := \{v\}$
3. While $(X \neq V)$
4. Wähle $w \in V \setminus X$ mit $\sum_{\substack{v \in X \\ e=\{v,w\}}} c(e)$ maximal
5. Sei s (bzw. t) der Knoten der als vorletztes (bzw. letztes) zu X hinzugefügt wurde.
6. Speichere Wert des Schnittes $(V \setminus \{t\}, \{t\})$.
7. Verschmelze s und t und gehe zu Zeile 2.; Falls $|V| = 1$ **return** kleinsten Schnitt.

Analyse

Satz 4.22
Der Algorithmus von Stoer-Wagner löst Problem 4.19 korrekt in $O(nm + n^2 \log n)$ Zeit.

Algorithmus 4.20
Eingabe:
Ungerichteter Graph $G = (V, E)$ und Kostenfunktion $c: E \to \mathbb{R}^+$
Ausgabe:
Nicht-leere Menge $X \subsetneq V$ mit $\sum_{e \in E(X, V \setminus X)} c(e)$ minimal.

1. Function EdmondsKarp(G, c)
2. Wähle $v \in V$ und setze $X := \{v\}$
3. While $(X \neq V)$
4. Wähle $w \in V \setminus X$ mit $\sum_{\substack{v \in X \\ e=\{v,w\}}} c(e)$ maximal
5. Sei s (bzw. t) der Knoten der als vorletztes (bzw. letztes) zu X hinzugefügt wurde.
6. Speichere Wert des Schnittes $(V \setminus \{t\}, \{t\})$.
7. Verschmelze s und t und gehe zu Zeile 2.; Falls $|V| = 1$ **return** kleinsten Schnitt.

Fibonacci-Heaps!

Pro Iteration von Zeile 3:
w finden: $O(\log n)$
DS updaten: $O(\delta(w))$

Insgesamt: $O(m + n \log n)$

Quiz!

Mentimeter