Technische Universität Braunschweig

Netzwerkalgorithmen – Vorlesung #13

Arne Schmidt

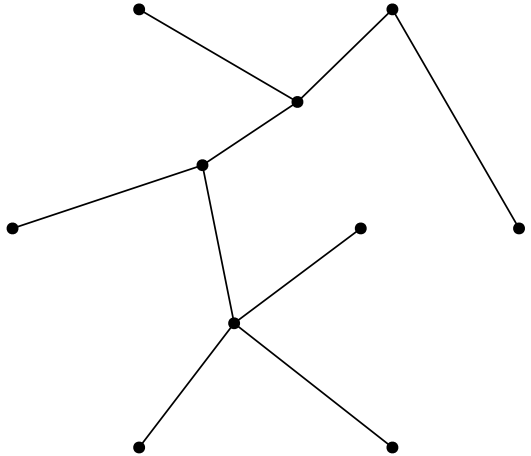
Zusammenfassung

Minimale Spannbäume

Bäume - VL 1

Satz 2.5
Für einen Graphen $T = (V, E)$ sind folgende Aussagen äquivalent:

1. T ist ein Baum (zshgd. + kreisfrei).
2. T besitzt $n - 1$ Kanten und ist zshgd.
3. T besitzt $n - 1$ Kanten und ist kreisfrei.
4. T ist maximal kreisfrei.
5. T ist minimal zusammenhängend.
6. T enthält einen eindeutigen Pfad zwischen jedem Paar von Knoten.

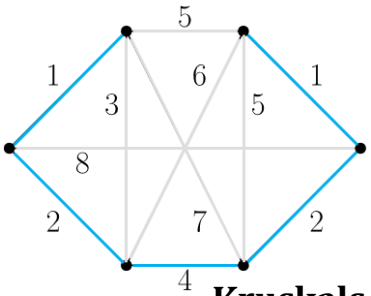


MST Kruskal – VL 2

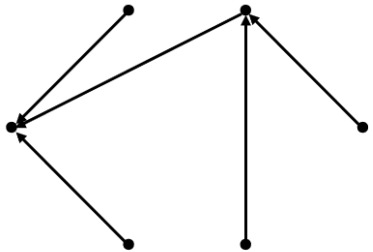
Problem 2.1: Minimal aufspannender Baum (kurz: MST)
Gegeben:
Zshgd. Graph $G = (V, E)$
Kostenfunktion $c\colon E \to \mathbb{R}$
Gesucht:
Kantenmenge $F \subseteq E$ mit $\sum_{e \in F} c(e)$ minimal und $T = (V, F)$ zusammenhängend.

Satz 2.6 (Formel von Cayley)
Sei G ein vollständiger Graph mit n Knoten. Dann gibt es n^{n-2} viele verschiedene Spannbäume von G.

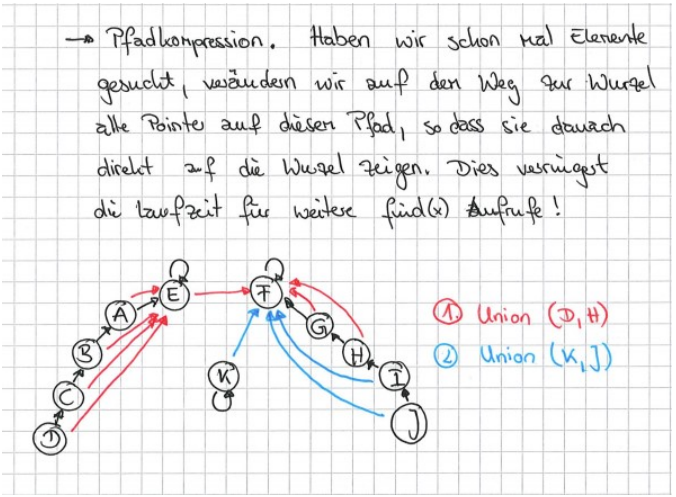
1. Kanten sortieren.
2. In dieser Reihenfolge einfügen



Kruskals Algorithmus



Disjoint Set

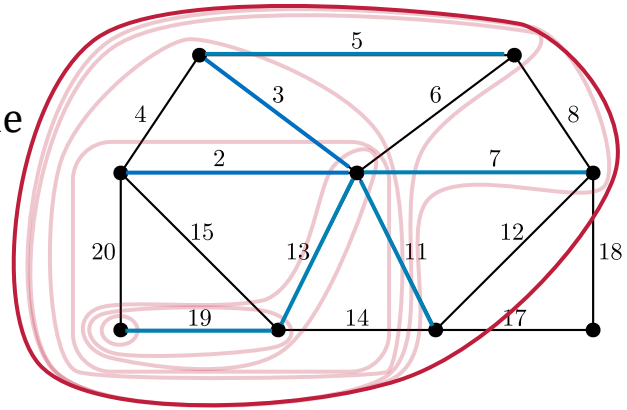


MST Prim – VL 2/3

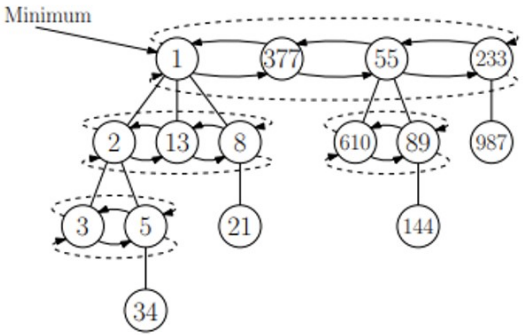
Satz 2.9 / 2.15
Sei $G = (V, E)$ ein Graph, $c\colon E \to \mathbb{R}^+$ eine Kostenfunktion und $T = (V, F)$ ein aufspannender Baum von G. Dann sind folgende Aussagen äquivalent.

(a) T ist ein MST
(b) Für jede Kante $f = \{v, w\} \in E \setminus F$ gilt: Für jede Kante e auf dem vw-Pfad in T ist $c(e) \leq c(f)$
(c) Für alle Mengen $\emptyset \neq X \subsetneq V$ ist die kleinste Kante aus $E(X, V \setminus X)$ in T enthalten.

1. Starte bei einem Knoten
2. Füge kleinste herausführende Kante hinzu.



Prims Algorithmus



Fibonacci Heaps

Matroide – VL 3

Definition 2.19
Ein Mengensystem $(E, \mathcal{I})$ mit $\mathcal{I} \subseteq 2^E$ ist ein *Unabhängigkeitssystem*, wenn
(1) $\emptyset \in \mathcal{I}$
(2) $I \in \mathcal{I}, I' \subset I \Rightarrow I' \in \mathcal{I}$

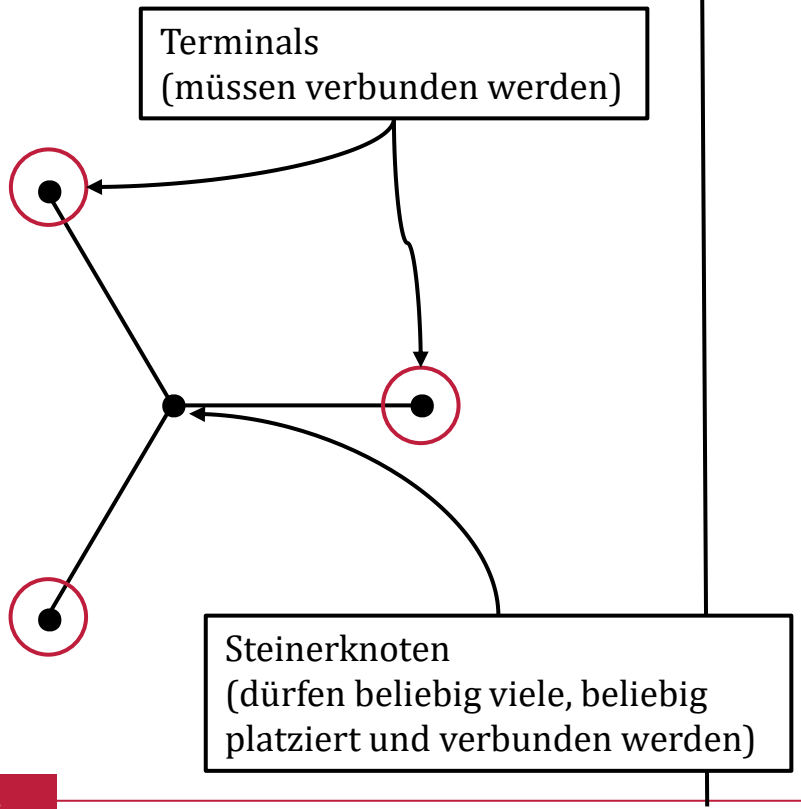
Elemente aus $\mathcal{I}$ heißen *unabhängig*.
Maximal unabhängige Mengen heißen *Basen*.

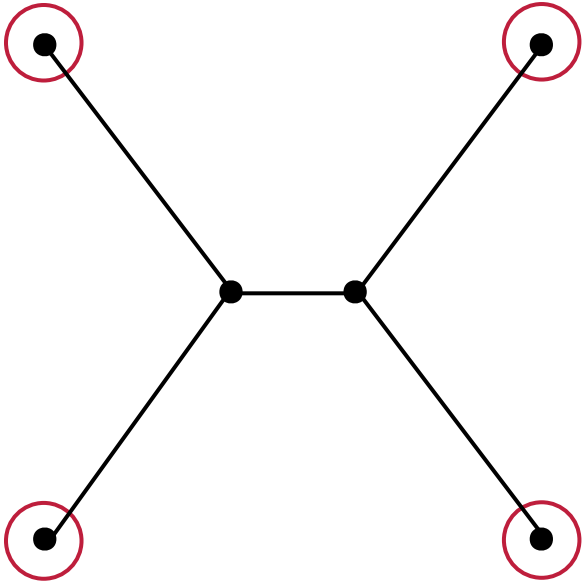
Ein Unabhängigkeitssystem heißt *Matroid*, wenn gilt:
(3) Wenn $I_1, I_2 \in \mathcal{I}$ und $|I_1| < |I_2|$, dann existiert $x \in I_2 \setminus I_1$, sodass $I_1 \cup \{x\} \in \mathcal{I}$

Beispiel 2:
Gegeben sei ein Graph $G = (V, E_2)$ uns sei $\mathcal{I}_2 := \{F \subseteq E_2 \mid F \text{ enthält keinen Kreis}\}$.

Behauptung:
$(E_2, \mathcal{I}_2)$ ist ein Matroid.

Varianten – Steinerbäume - Geometrisch



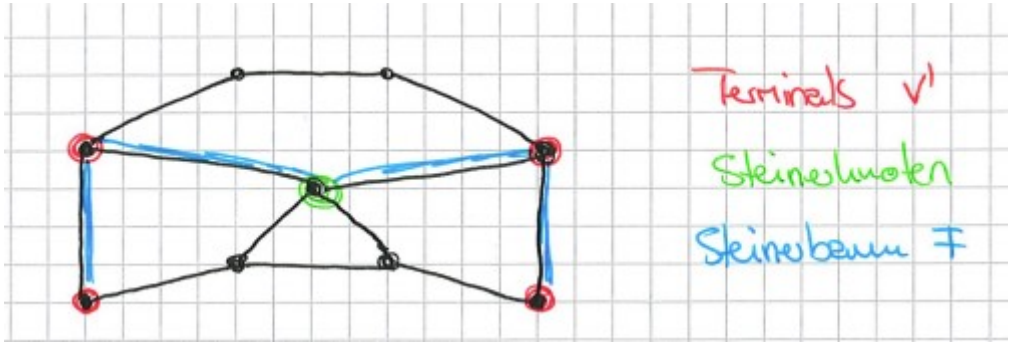


Varianten – Steinerbäume - Graphen

Steinerbaum Problem

Gegeben: Graph $G=(V,E)$ mit Kantenkosten $c: E \to \mathbb{R}^+$
und Knotenteilmenge $V' \subseteq V$ (Terminals)

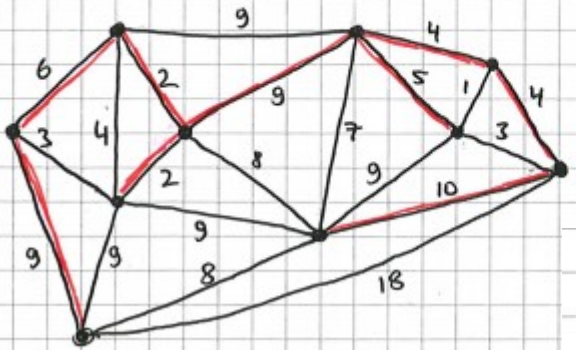
Gesucht: Kantenteilmenge $F \subseteq E$ minimalen Gesamtgewichts,
so dass alle Knoten V' verbunden/zshg. sind



Varianten – Bottleneck MST

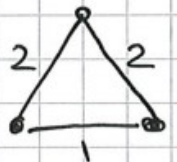
Gegeben: Graph $G=(V,E)$ mit Kantengewichten $c: E \to \mathbb{R}^+$, $b \in \mathbb{R}^+$

Gesucht: Spannbaum, so dass jede Kante höchstens "b" lang ist.

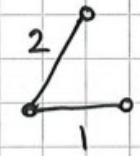


BST mit $b=10$.

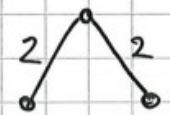
Geht es besser?



G



MST



MBST

Kürzeste Wege

Kürzeste Wege

Problem 3.1: Kürzester Weg Problem (Shortest Path, SP)
Gegeben:

Digraph $D = (V, A)$
Kostenfunktion $c\colon A \to \mathbb{R}$
Knoten $s, t \in V$

Gesucht:

Ein kostenminimaler st-Weg

Schwer

Beliebige Gewichte:
Es können negative gerichtete Kreise auftreten.

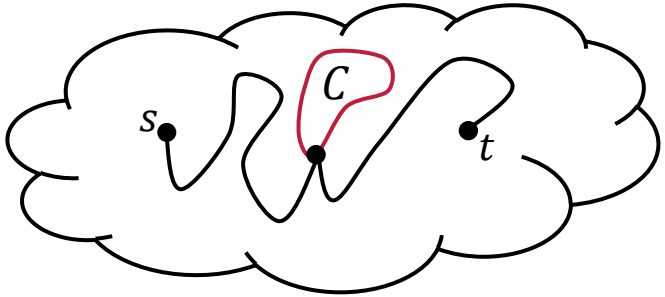
Konservative Gewichte:
Jeder gerichtete Kreis besitzt nicht-negative Gesamtkosten.

Nicht-Negative Kosten:
Jede Kante besitzt ein Gewicht von mindestens Null.

Einfach

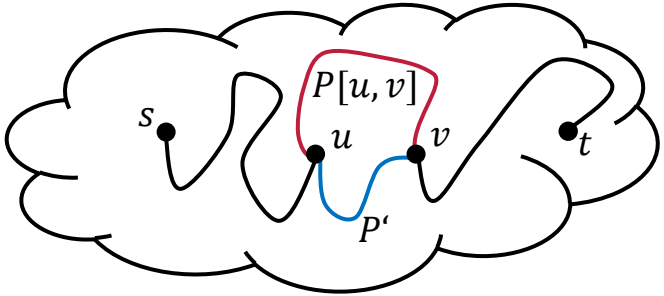
Konstante (positive) Kosten:
Entspricht der ungewichteten Variante. (siehe AuD)

Kürzeste Wege – Eigenschaften



Betrachte nur Pfade.

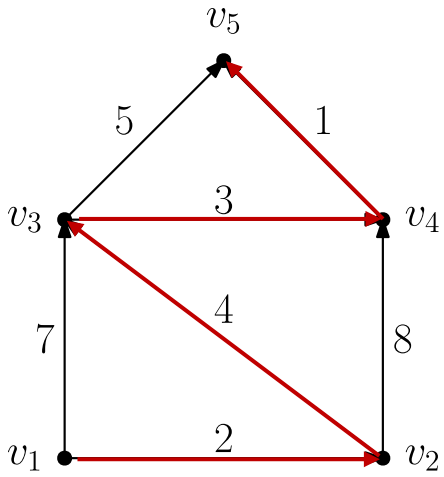
(Gilt das auch für den zweitkürzesten Weg?)



Teilpfade kürzester Pfade sind kürzeste Pfade

(Gilt das auch für längste Pfade?)

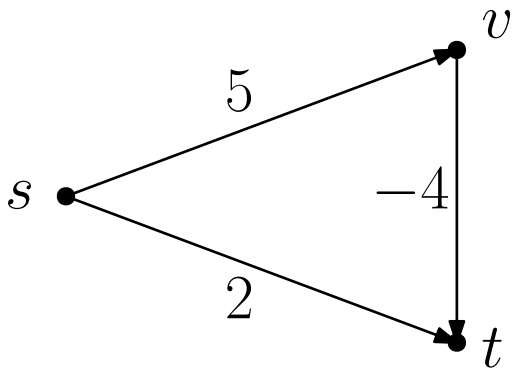
Dijkstras Algorithmus



It.	v_1		v_2		v_3		v_4		v_5		R
	$\ell(v_1)$	$p(v_1)$	$\ell(v_2)$	$p(v_2)$	$\ell(v_3)$	$p(v_3)$	$\ell(v_4)$	$p(v_4)$	$\ell(v_5)$	$p(v_5)$	
0	0	–	∞	–	∞	–	∞	–	∞	–	$\emptyset$
1			2	v_1	7	v_1					v_1
2					6	v_2	10	v_2			v_1, v_2
3							9	v_3	11	v_3	v_1, v_2, v_3
4									10	v_4	v_1, v_2, v_3, v_4
5											v_1, v_2, v_3, v_4, v_5

Satz 3.8
Dijkstra‘s Algorithmus löst Problem 3.4 (Kürzeste Wege) korrekt für nicht-negative Kantenkosten in $O(n \log n + m)$ Zeit.

Konservative Kantenkosten



Für alle $v_k \in V \setminus R$ ist:

$$\ell_k \leq \min_{v_i \in R} \left(\ell_i + c\big((i,k)\big) \right)$$

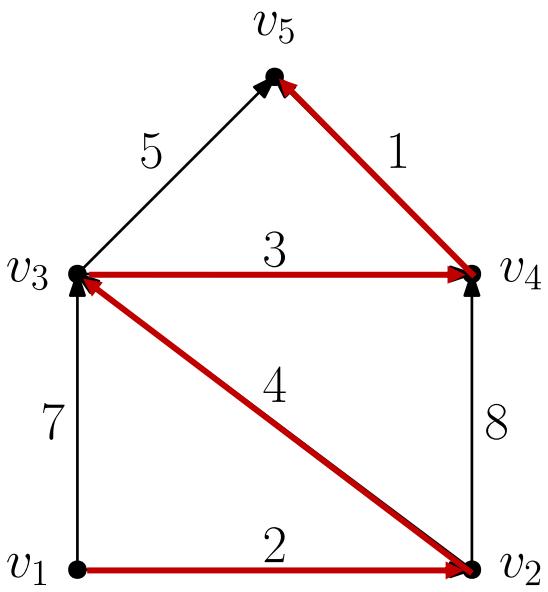
Dijkstra

Für alle $v_k \in V$ ist:

$$\ell_k = \min_{(v_i, v_k) \in A} \left(\ell_i + c\big((i,k)\big) \right)$$

Moore-Bellman-Ford

Algorithmus von MBF



It.	v_1		v_2		v_3		v_4		v_5	
	$\ell(v_1)$	$p(v_1)$	$\ell(v_2)$	$p(v_2)$	$\ell(v_3)$	$p(v_3)$	$\ell(v_4)$	$p(v_4)$	$\ell(v_5)$	$p(v_5)$
0	0	–	∞	–	∞	–	∞	–	∞	–
1			2	v_1	6	v_2	9	v_3	11	v_4
2									10	v_4
3										
4										
5										

Satz 3.10
Der Algorithmus von Moore, Bellman und Ford löst Problem 3.4 (Kürzeste Wege) korrekt für konservative Kantenkosten in $O(nm)$ Zeit.

APSP – Floyd-Warshall

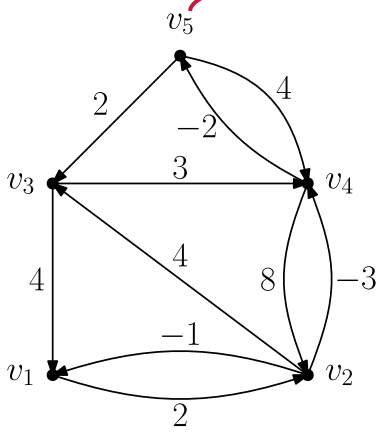
$$\ell(i,j,k) := \begin{cases} c\left((v_i, v_j)\right), & \text{falls } k = 0 \\ \min(\ell(i,j,k-1), \ell(i,k,k-1) + \ell(k,j,k-1)), & \text{falls } k > 0 \end{cases}$$

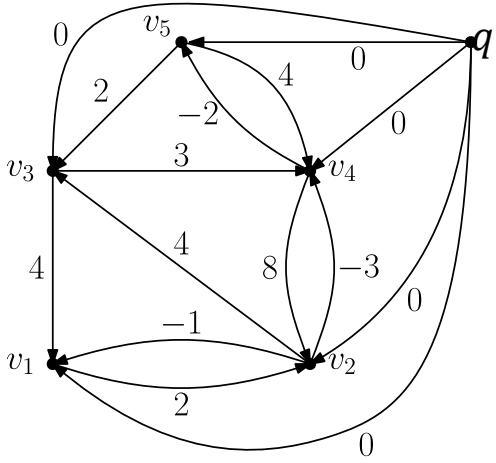
⟶ Algorithmus von Floyd-Warshall

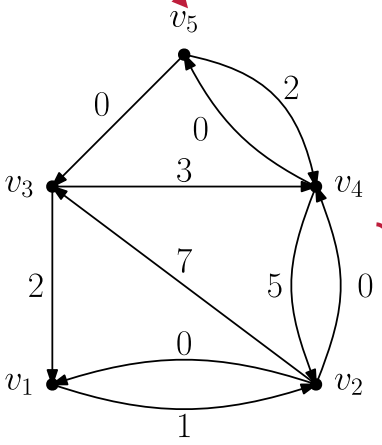
Satz 3.15
Der Algorithmus von Floyd-Warshall löst Problem 3.13 (APSP) korrekt in Zeit $O(n^3)$.

Lemma 3.16
Ein Digraph $D = (V, A)$ mit Kostenfunktion $c: A \to \mathbb{R}$ enthält genau dann einen negativen Kreis, wenn für (mind.) einen Knoten $v_i \in V$ nach Ausführung des Algorithmus von Floyd-Warshall $\ell_i(i) < 0$ gilt.

APSP – Johnsons Algorithmus







$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}
\begin{array}{c}
\begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\
\begin{pmatrix} 0 & 2 & -1 & -1 & -3 \\ -1 & 0 & -3 & -3 & -5 \\ 4 & 6 & 0 & 3 & 1 \\ 4 & 6 & 0 & 0 & -2 \\ 6 & 8 & 2 & 4 & 0 \end{pmatrix}
\end{array}
$$

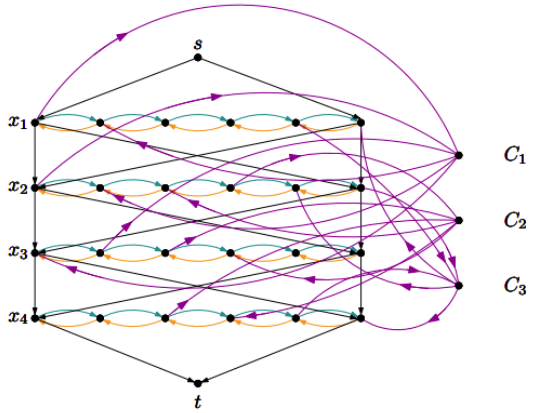
$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}
\begin{array}{c}
\begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\
\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 3 & 3 \\ 2 & 3 & 0 & 0 & 0 \\ 2 & 3 & 0 & 2 & 0 \end{pmatrix}
\end{array}
$$

Algorithmen

Algorithmus	Kostenfunktionen	Problem	Laufzeit	Paradigma
Dijkstra	$\mathbb{R}_{\geq 0}$	SSSP	$O(m + n \log n)$	Greedy, Dyn. Programming
Moore, Bellman, Ford	Konservativ	SSSP	$O(mn)$	Dyn. Programming
Floyd-Warshall	Konservativ	APSP	$O(n^3)$	Dyn. Programming
Johnson	Konservativ	APSP	$O(mn + n^2 \log n)$	Reweighting, Dyn. Programming

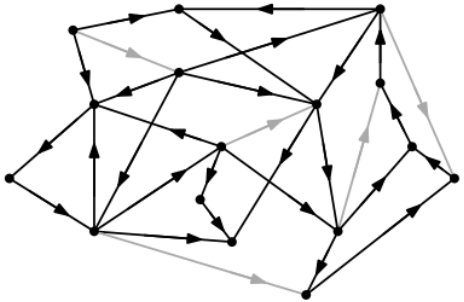
Weitere Probleme (Gr. Übung)

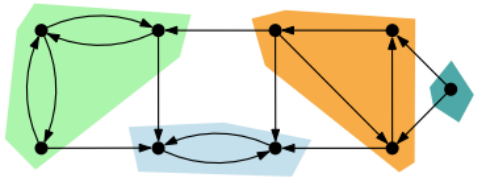
Der längste Pfad ist schwer!



$$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee x_4)$$

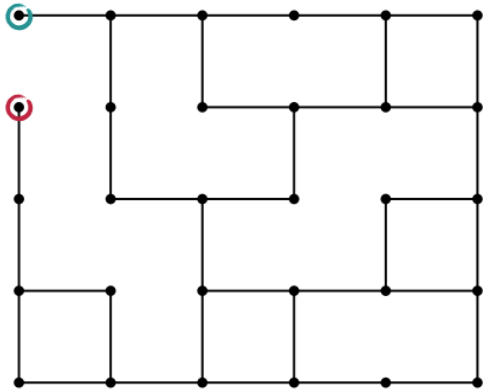
Wurmlöcher!





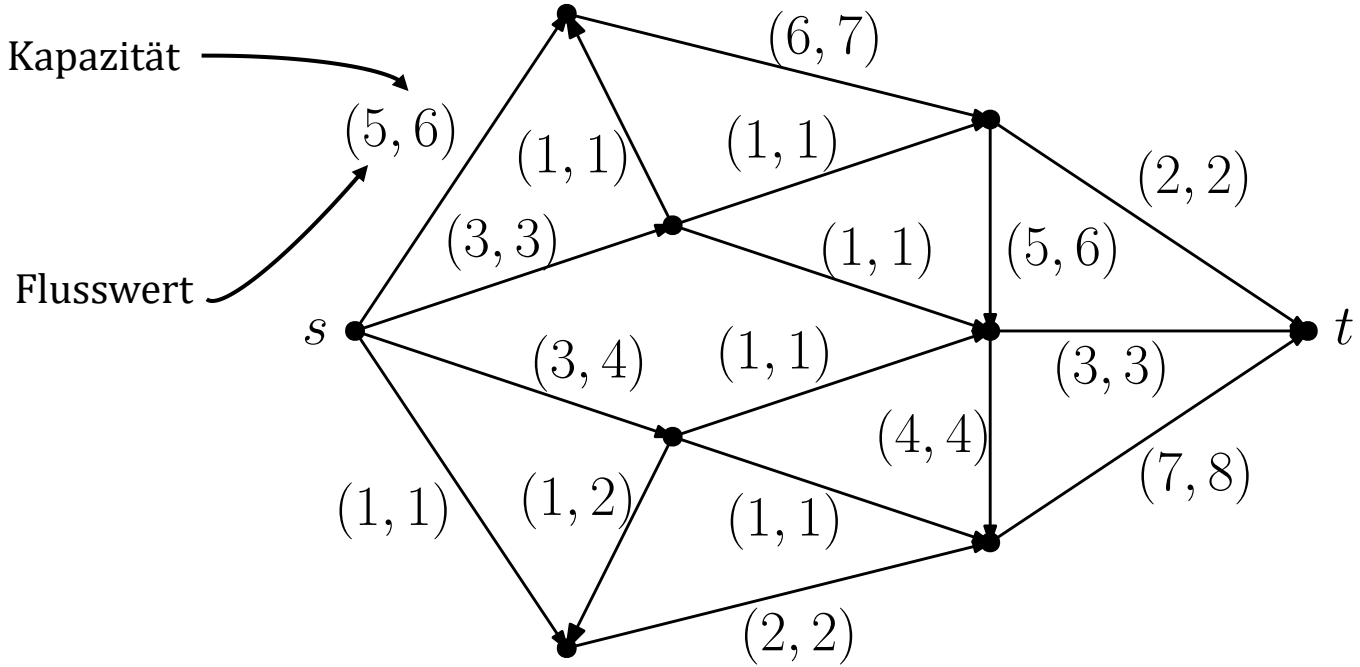
Starke ZHK

Abbiegungen kosten viel Zeit

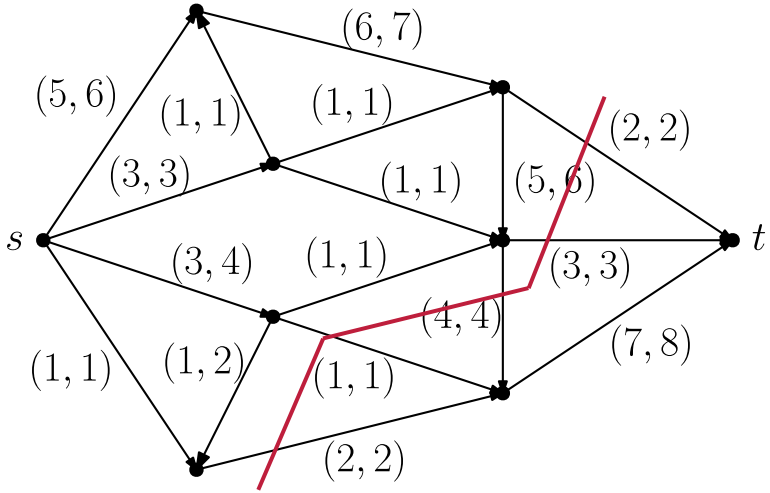


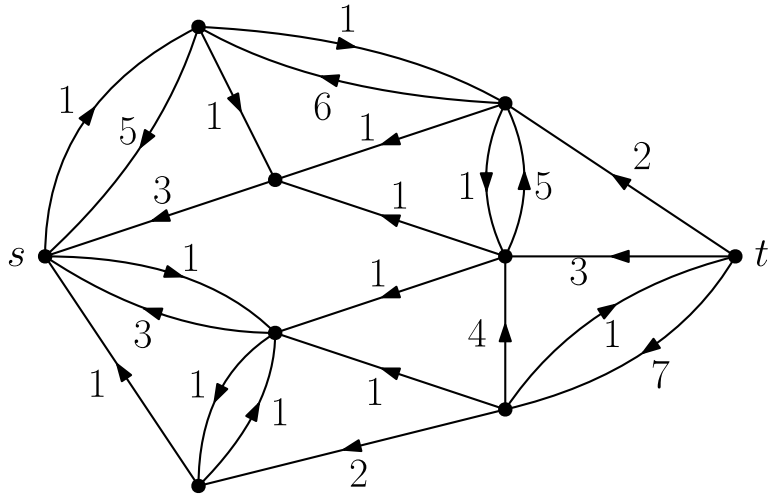
Maximale Flüsse

Netzwerke


Kapazität
Flusswert
(5, 6)
(6, 7)
(1, 1)
(1, 1)
(2, 2)
(3, 3)
(1, 1)
(5, 6)
s
t
(3, 4)
(1, 1)
(3, 3)
(4, 4)
(7, 8)
(1, 1)
(1, 2)
(1, 1)
(2, 2)

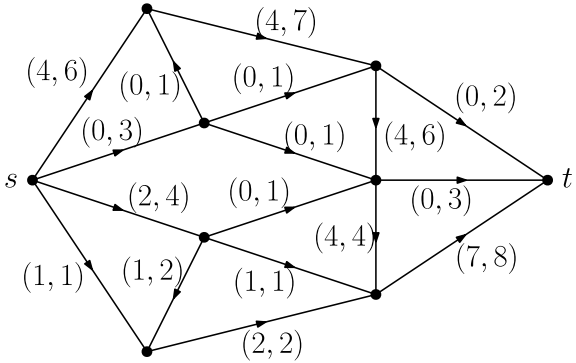
Min-Cut = Max-Flow

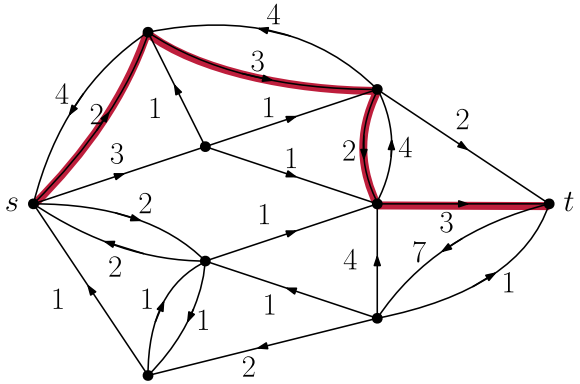


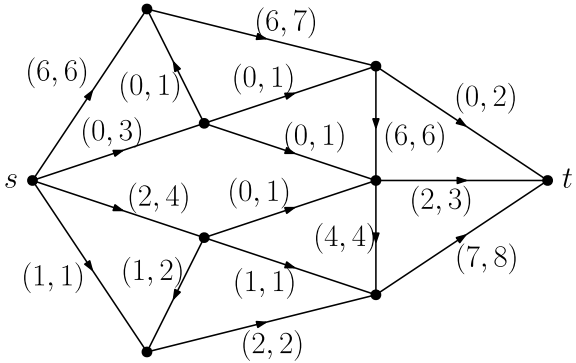


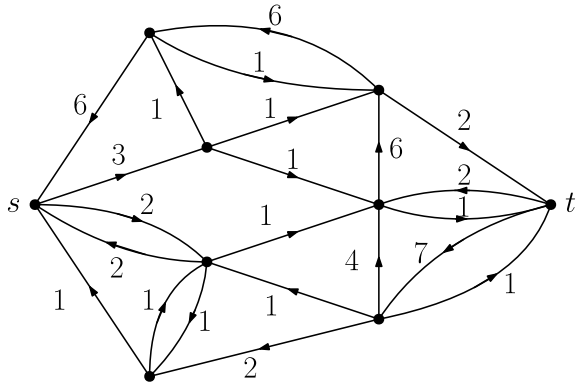
Satz 4.7 (Min-Cut = Max-Flow)
Sei $N = (D, u, s, t)$ ein Netzwerk. Der Wert eines maximalen st-Fluss f entspricht dem Kapazitätswert eines minimalen Schnittes.

Augmentieren von Flüssen – Ford-Fulkerson-Algorithmus

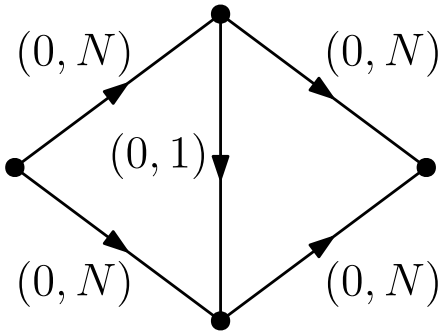




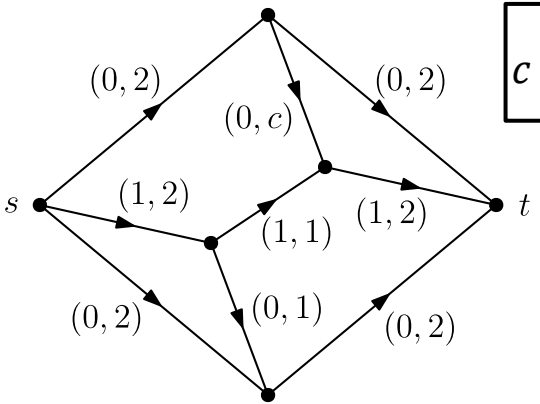




Worst-Case-Beispiele



Ggf MaxFlow viele Iterationen



$$c := \frac{\sqrt{5} - 1}{2}$$

Unter reellen Kapazitäten terminiert Ford-Fulkerson nicht immer.

Flow Decomposition Theorem

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:

a) $f(e) = \sum_{\substack{p \in \mathcal{P} \cup \mathcal{C} \\ e \in p}} w(p)$

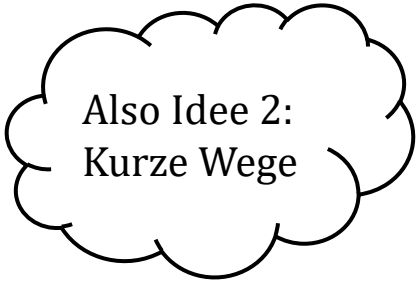
b) $Wert(f) = \sum_{p \in \mathcal{P}} w(p)$

c) $|\mathcal{P}| + |\mathcal{C}| \leq |E|$

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.

Kürzeste augmentierende Pfade

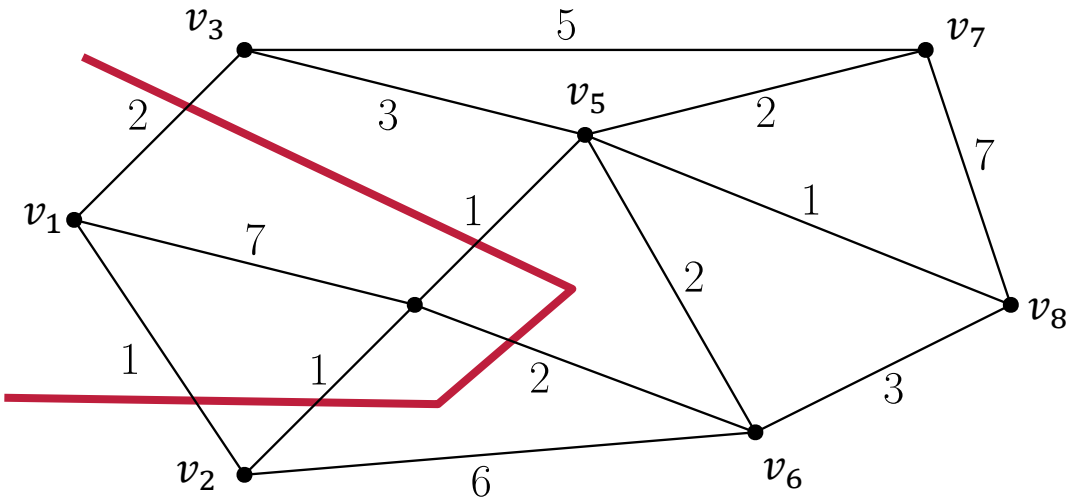


Lemma 4.17
Für ein Netzwerk $N = (D, u, s, t)$ mit $u: A \to \mathbb{R}^+$ terminiert der Algorithmus von Edmonds-Karp nach maximal $\frac{mn}{2}$ Iterationen.

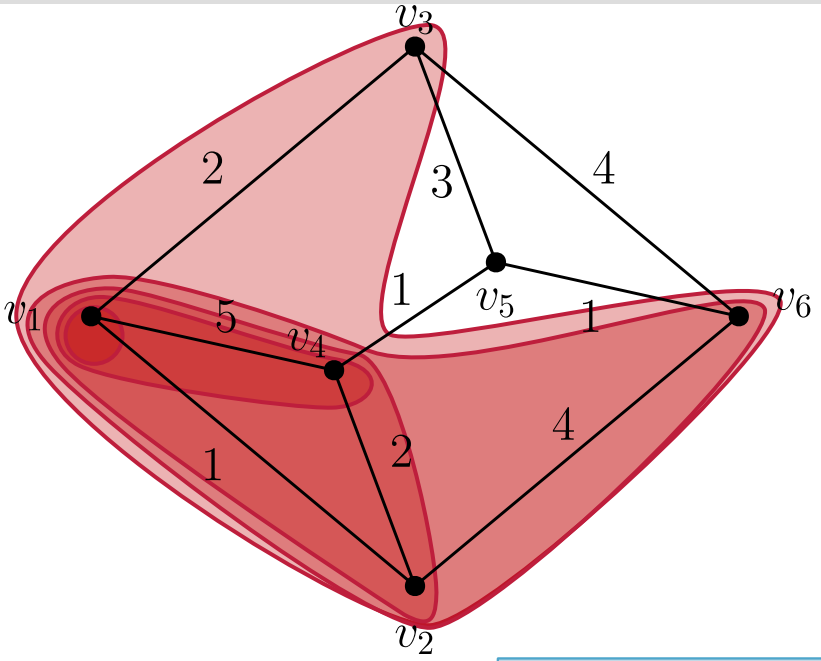
Satz 4.18
Der Algorithmus von Edmonds-Karp löst Problem 4.2 (MaxFlow) in Zeit $O(nm^2)$.

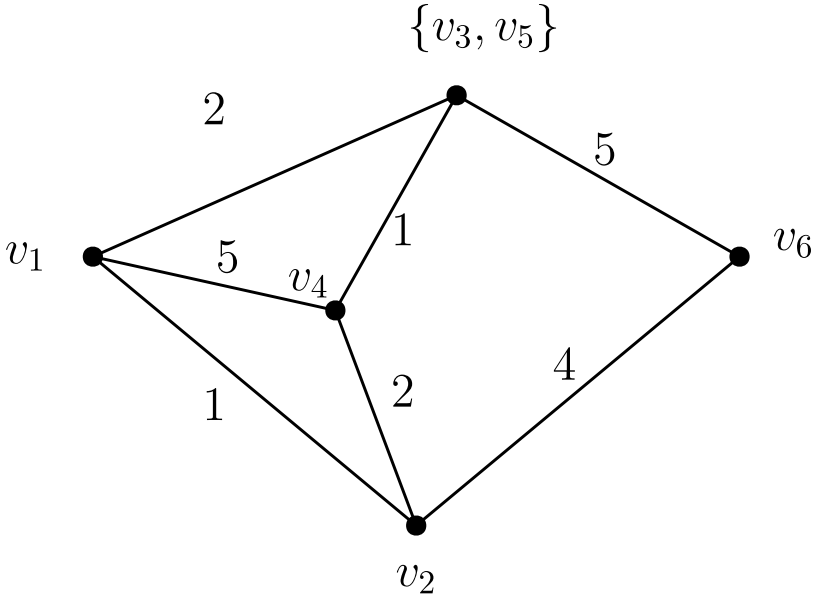
Global MinCut

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.



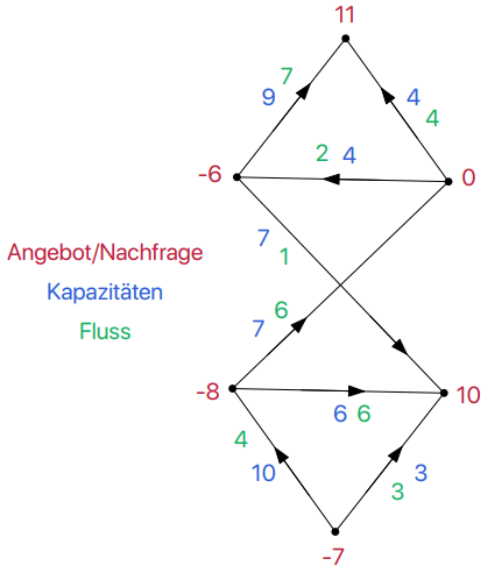
Stoer-Wagner





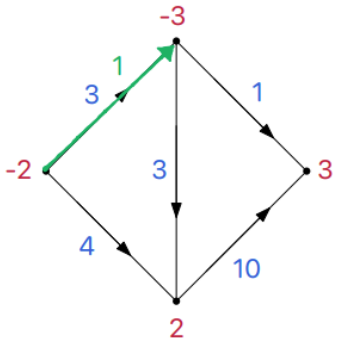
Satz 4.22
Der Algorithmus von Stoer-Wagner löst Problem 4.19 korrekt in $O(nm + n^2 \log n)$ Zeit.

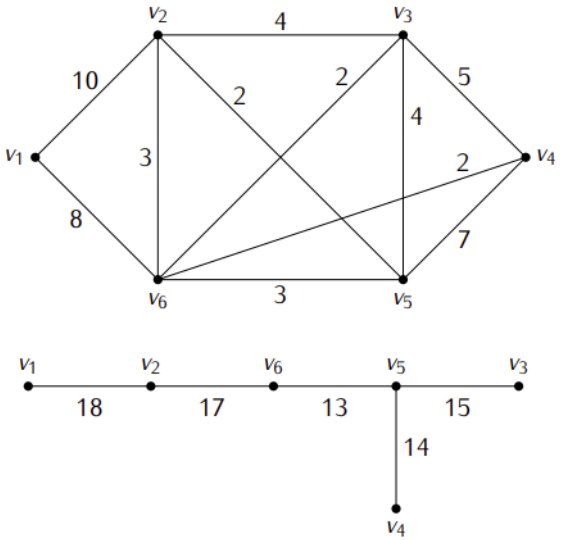
Weitere Probleme (gr. Übung)



Zirkulationen mit Angebot und Nachfrage

Flüsse mit unteren Schranken

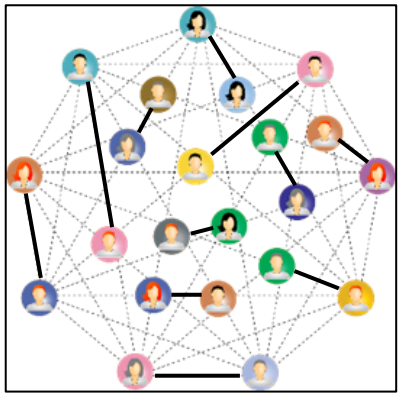




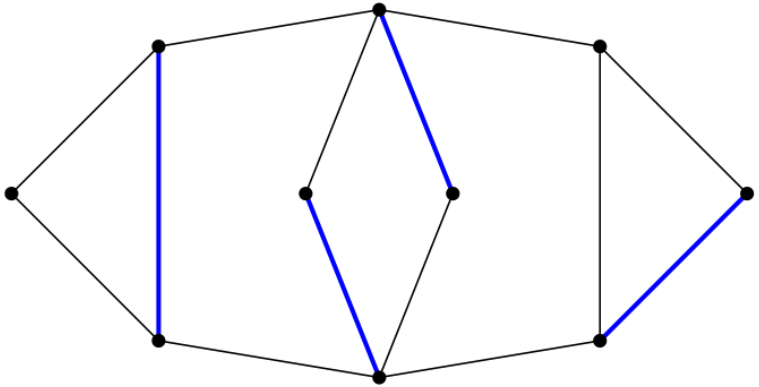
Gomory-Hu Bäume

Maximale Matchings

Matchings

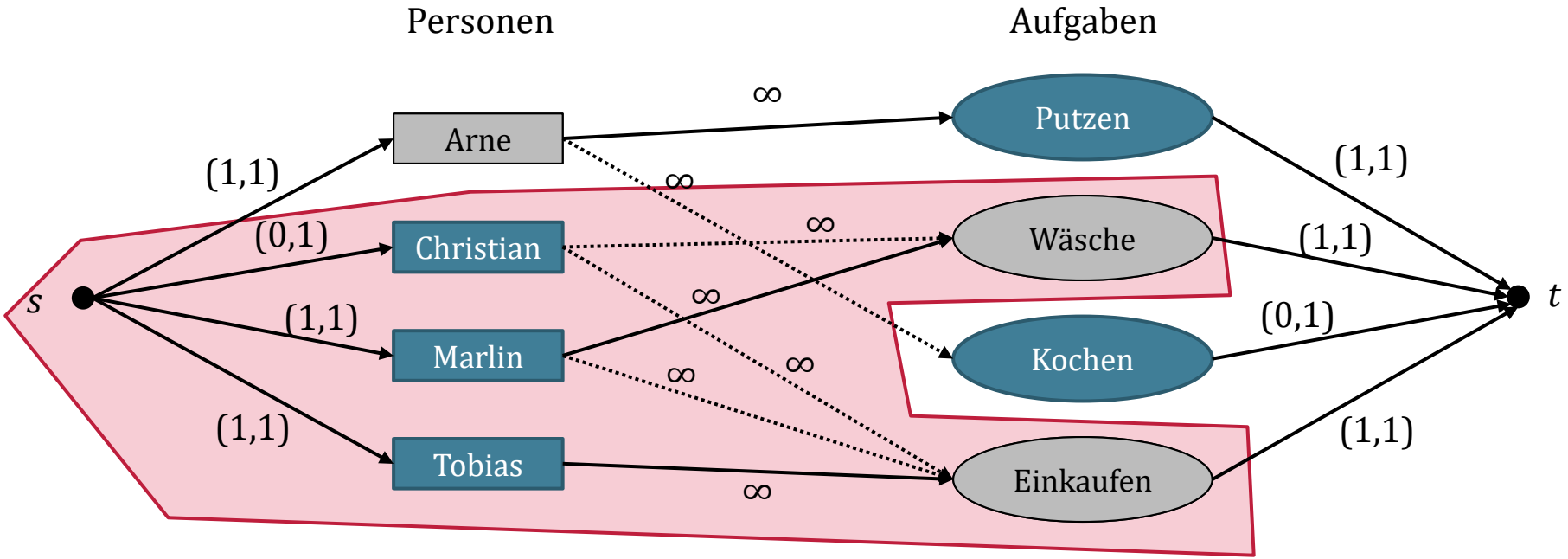


Perfektes Matching



Inklusionsmaximales Matching
(Auch kardinalitätsmaximal?)

Matchings – Bipartite Graphen

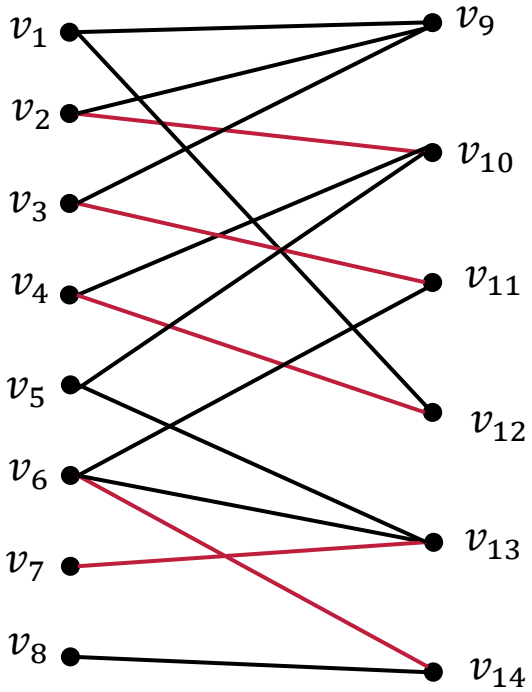


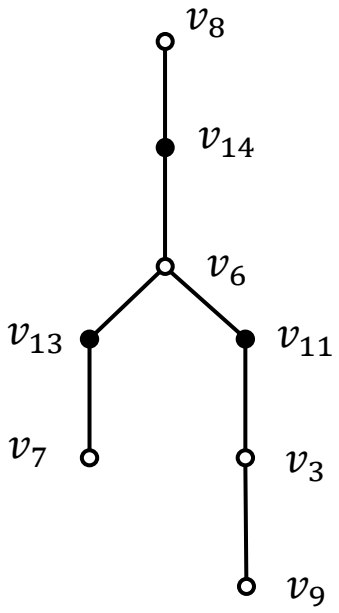
Reduktion auf ein Flussproblem.

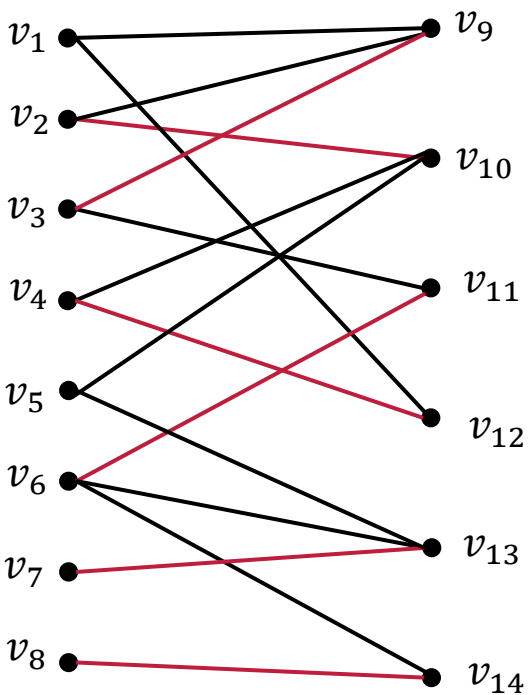
$\Rightarrow$ Suche augmentierende Pfade

$\Rightarrow$ Algorithmus für bipartite Graphen

Beispiel


v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
v_{14}


v_8
v_{14}
v_6
v_{13}
v_{11}
v_7
v_3
v_9

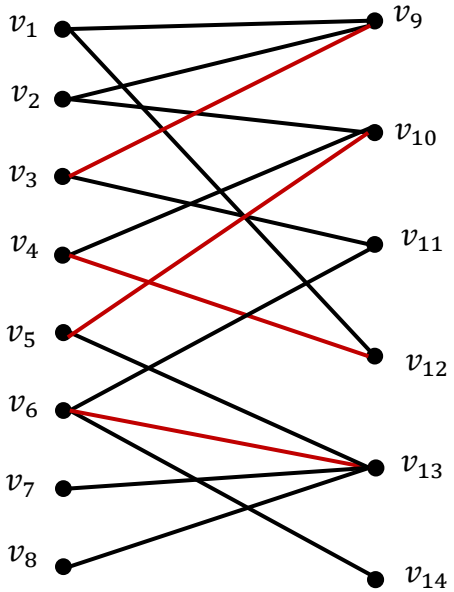

v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
v_{14}

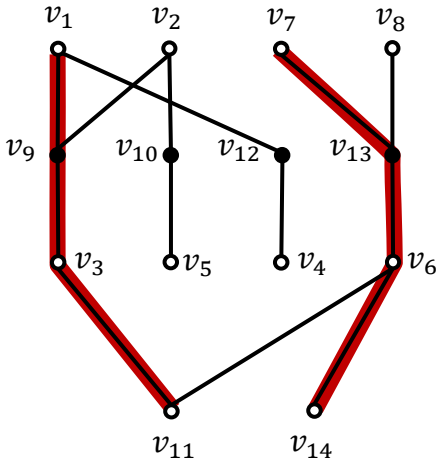
Matching – Bipartite Graphen

Satz 5.9
Algorithmus 5.8 löst Problem 5.2 (Max Matching) korrekt in Zeit $O(mn)$.

Suche mehrere Pfade gleichzeitig!

Satz 5.11
Algorithmus von Hopcroft-Karp löst Problem 5.2 (Max Matching) korrekt in Zeit $O(m\sqrt{n})$.





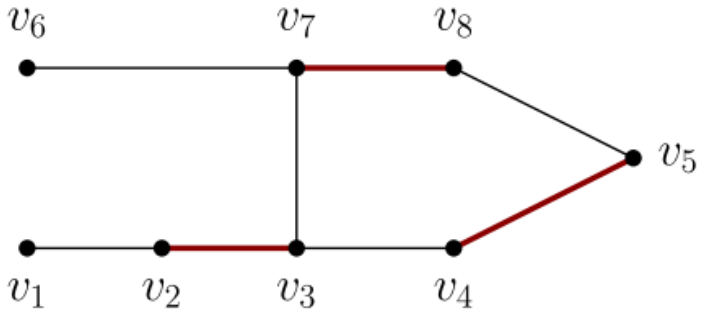
Matchings – Eigenschaften in bip. Graphen

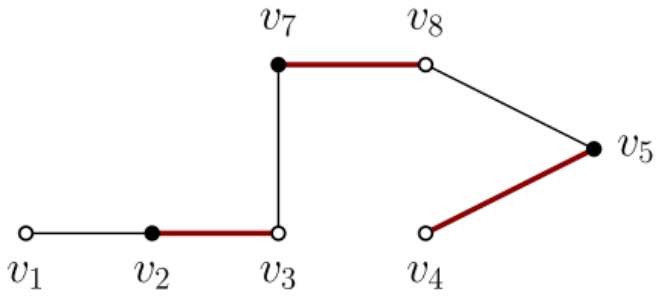
Satz 5.7 (Kőnig-Egerváry)
Sei G ein bipartiter Graph. Die Größe eines minimalen Vertex Covers in G entspricht der Größes eines maximalen Matchings in G.

Satz 5.13
Sei $G = (V, E)$ ein bipartiter Graph mit $V = V_1 \dot{\cup} V_2$. Dann hat G genau dann ein V_1 überdeckendes Matching, wenn $|N(X)| \geq |X|$ für alle $X \subseteq V_1$ gilt.

Korollar 5.14
Sei $G = (V, E)$ ein bipartiter Graph mit $V = V_1 \dot{\cup} V_2$. Dann hat G genau dann ein perfektes Matching, wenn $|V_1| = |V_2|$ und $|N(X)| \geq |X|$ für alle $X \subseteq V_1$ gilt.

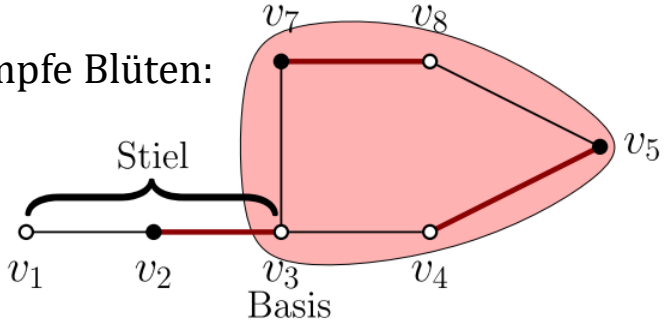
Matchings – Allgemeine Graphen





Nicht augmentierend!

Schrumpfe Blüten:



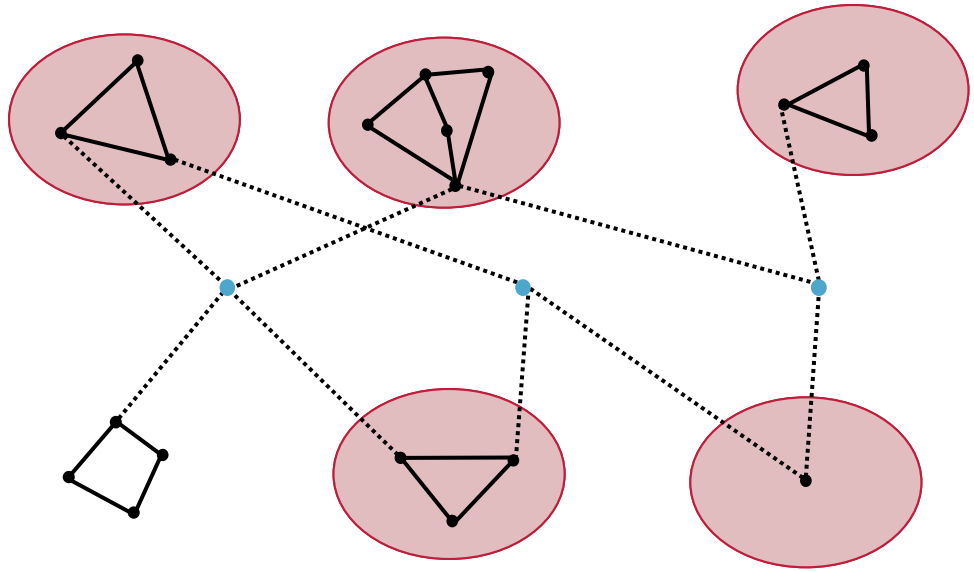
Lemma 5.23
Der Algorithmus von Edmonds (Blossom-Algorithmus) löst Problem 5.2 korrekt in Zeit $O(mn^2)$.

Matchings – Eigenschaften in allg. Graphen

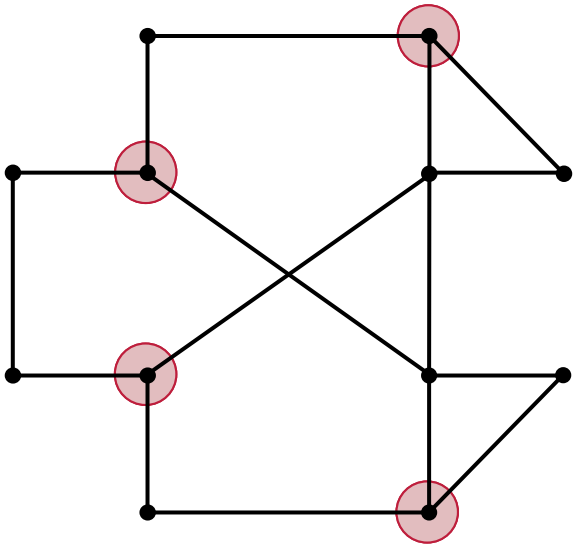
Satz 5.18
Sei $G = (V, E)$ ein Graph. G enthält genau dann ein perfektes Matching, wenn $\mathrm{oc}(G \setminus A) \leq |A|$ für alle $A \subseteq V$ gilt.

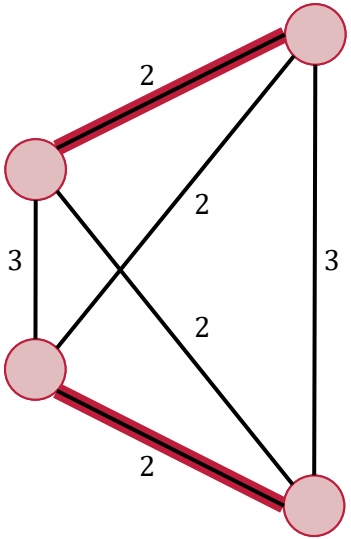
Satz 5.19
Sei $G = (V, E)$ ein Graph. Dann ist

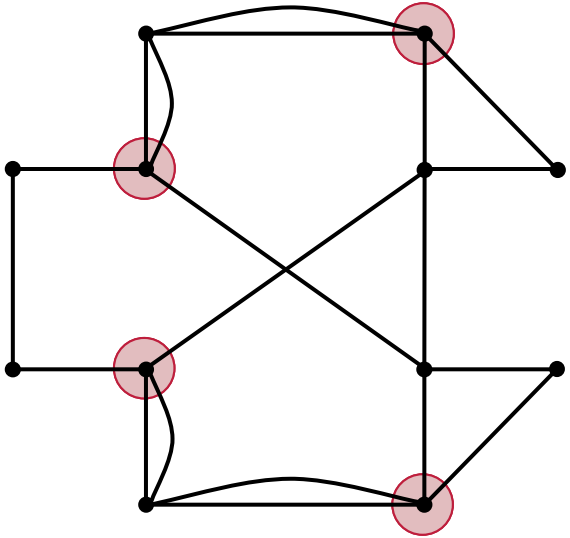
$$\max_{\text{Matching } M} |M| = \min_{A \subseteq V} \frac{1}{2}\left(|\mathrm{V}| - (\mathrm{oc}(G \setminus A) - |A|)\right)$$



(Chinesisches) Briefträgerproblem

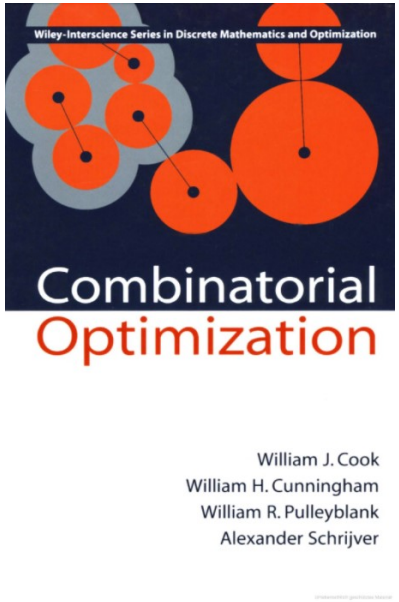


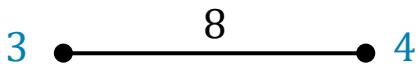




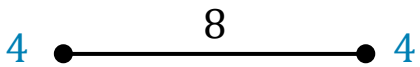
Kanten im Graphen dürfen auch gewichtet sein (s. gr. Übung).

Gewichtete Matchings





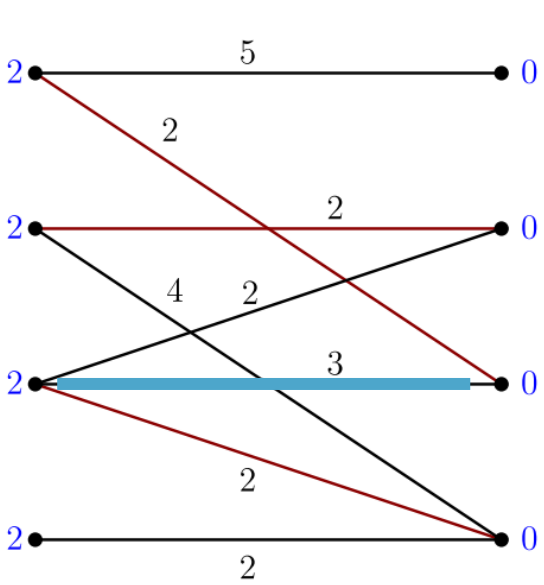
Reduzierte Kosten: 1

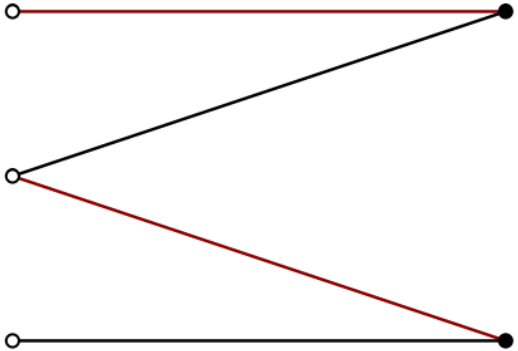


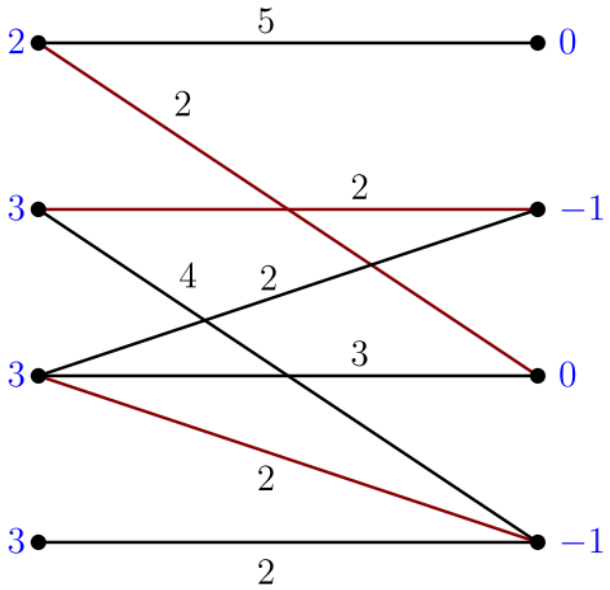
Kante überdeckt

$$e \in M \Rightarrow \bar{c}(e) = 0$$

Gewichtete Matchings – Ungarische Methode

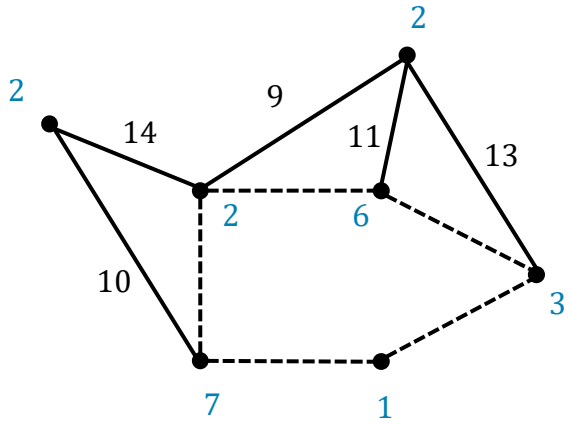


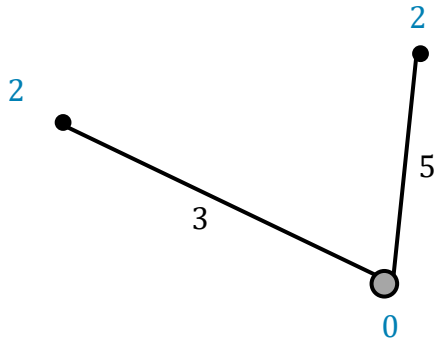




Satz 5.28
Algorithmus 5.27 löst Problem 5.24 (MinCost Perfect Matchings) für bipartite Graphen optimal in Zeit $O(n^3)$

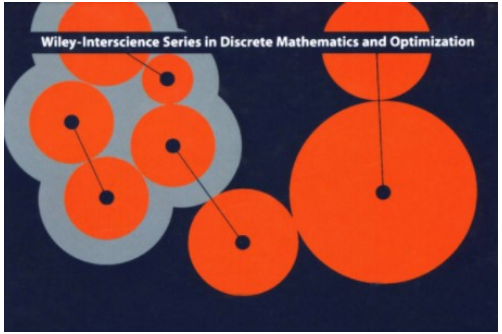
Gewichtete Matchings – Allgemeine Graphen





Updateregeln:

1. $\varepsilon_1 := \min\{\bar{c}(e) \mid e \in E(W(T), V \setminus W(T))\}$
2. $\varepsilon_2 := \frac{1}{2}\min\{\bar{c}(e) \mid e = \{v,w\} \text{ mit } v,w \in W(T)\}$
3. $\varepsilon_3 := \min\{y_B \mid \text{Blüte } B \in S(T)\}$



Gewichtete Matchings – Primal-Dual Algorithmus

Algorithmus 5.29
Eingabe:
Graph $G = (V, E)$, Kantenkosten $c: E \to \mathbb{R}^+$
Ausgabe:
Min Cost Perfect Matching M

1. **Function** PRIMALDUAL(G, c)
2. Setze $G' := G$ und $M' = M = \emptyset$
3. **for** $r \in V$ mit r ungematcht **do**
4. Konstruiere alternierenden Baum T mit r als Start.
5. **while** T enthält keinen augmentierenden Pfad **do**
6. Bestimme $\varepsilon := \min(\varepsilon_1, \varepsilon_2, \varepsilon_3)$
7. Setze $y_v := y_v + \varepsilon$ für alle $v \in W(T)$.
8. Setze $y_w := y_w - \varepsilon$ für alle $w \in S(T)$.
9. ***Erweitere*** T (solange möglich)
10. Augmentiere M'
11. Konstruiere Matching M für G aus M' in G'.
12. **return** M

Satz 5.30
Algorithmus 5.29 löst Problem 5.24 (MinCost Perfect Matchings) optimal in Zeit $O(mn^2)$

Erweitere T:
Case $\exists$ Blüte $B \in S(T)$ mit $y_B = 0$:
Löse Blüte auf.
Passe G', M' und T entsprechend an.
Case Blüte B gefunden:
Schrumpfe B.
Passe G', M' und T entsprechend an.
Case Sonst:
Erweitere T über überdeckte Kanten.

Gewichtete Matchings – weitere Anwendung

→ Metrisches Traveling Salesman Problem

Geg: vollständiger Graph mit Kantengewichten $c: E \to \mathbb{R}^+$ sodass für je drei Knoten a, b, k gilt:

$$c(\{a,k\}) \leq c(\{a,b\}) + c(\{b,k\}).$$

Ges: Hamiltonkreis mit minimalem Gesamtgewicht.

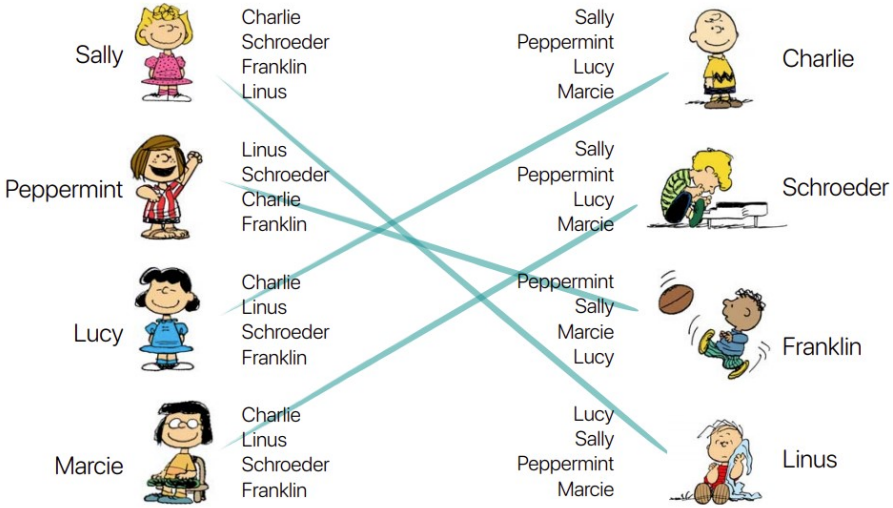
Satz. Der Algorithmus von Christofides ist eine $3/2$-approximation für metrisches TSP.

Algorithmus von Christofides (1976)

1. Vollständiger Graph mit Δ-Ungleichung
2. Berechne MST T mit Gewicht $c(T)$
3. Berechne kostenminimales perfektes Matching M auf ungeraden Knoten von T mit Gewicht $c(M)$
4. Bestimme Eulertour R in $G' = (V(G), E(T) \cup E(M))$ mit Gewicht $c(R)$
5. Bestimme TSP-Tour π durch Abkürzen in R mit Gewicht $c(\pi) \leq c(R)$

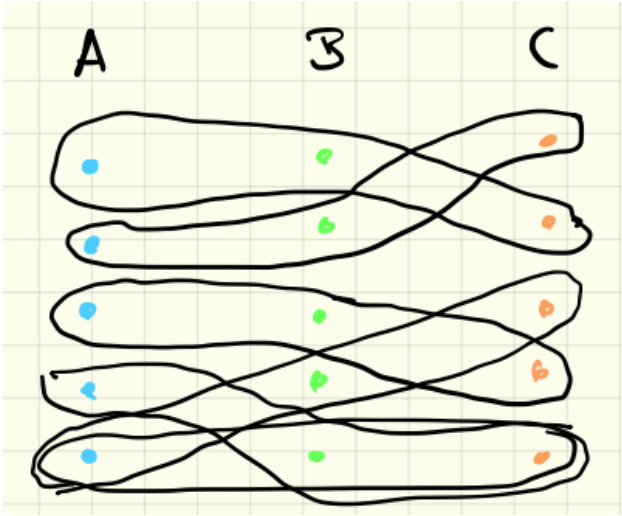
Varianten (gr. Übung)

Stabile Matchings



Hier: nicht stabil.

3D-Matching



NP-schwer!