Matroide und Greedy-Algorithmen

Definition

Ein Unabhängigkeitssystem $M = (E, S)$ heißt Matroid, falls für alle $A, B \in S$ mit $|B| = |A| + 1$ ein $x \in B \setminus A$ existiert mit $A \cup \{x\} \in S$.

Definition

Sei $M = (E, S)$ ein Matroid. $A \in S$ heißt Basis von M, wenn A maximal in S ist, d.h. alle B mit $A \subsetneq B$ sind abhängig.

Satz

Alle Basen eines Matroids sind gleichmächtig.

Definition

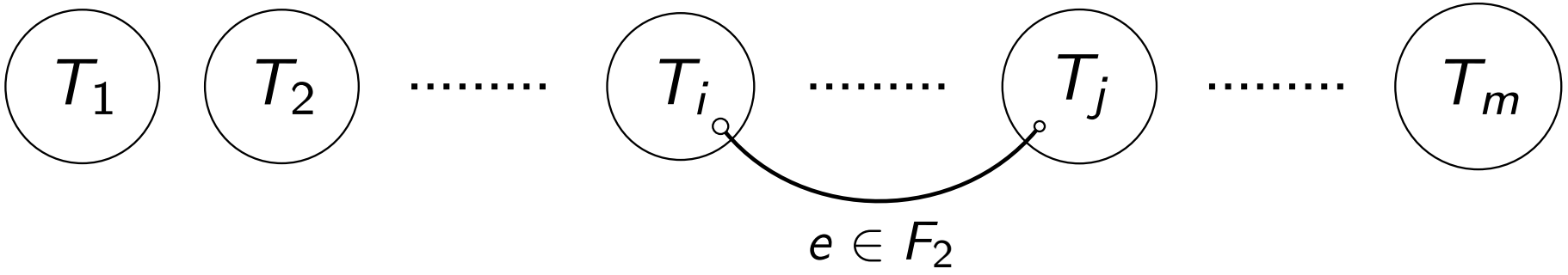
Die Mächtigkeit einer Basis A eines Matroids M heißt Rang von M: $r(M) := |A|$.

Satz

Sei $G = (V, E)$ ein Graph und $S = \{F \subseteq E \mid (V, F)$ ist Wald$\}$.
Dann ist (E, S) ein Matroid (graphisches Matroid).

Beweis: $F_1, F_2 \subseteq E$, F_2 Wald, $F_1 \subseteq F_2 \implies F_1$ Wald.

Seien nun F_1 und F_2 Wälder und $|F_2| = |F_1| + 1$.
Zu zeigen: Es gibt $e \in F_2 \setminus F_1$, sodass $(V, F_1 \cup \{e\})$ ein Wald ist.
F_1 habe m Komponenten $T_i = (V_i, A_i)$, $i = 1, \ldots, m$



F_2 Wald, daher höchstens $|V_i| - 1 = |A_i|$ Kanten vw mit $v, w \in V_i$.
Daher ist $F_1 \cup \{e\}$ ein Wald. □

Ad Beispiel graphisches Matroid

Falls $G = (V, E)$ m Zusammenhangskomponenten hat, dann gilt für das graphische Matroid $M = (E, S)$ mit $S = \{F \subseteq E \mid (V, F) \text{ ist Wald }\}$: $r(M) = |V| - m$.

Beispiel Vektormatroid

Sei $E \subseteq \mathbb{R}^n$ und $S := \{F \subseteq E \mid F \text{ ist lin. unabh.}\}$. Dann ist das Unabhängigkeitssystem $M = (E, S)$ sogar ein Matroid.

- Wenn $A, B \in S$, $|B| = |A| + 1$, dann gibt es $x \in B \setminus A$, sodass $A \cup \{x\} \in S$.
  Denn $\dim[B] = \dim[A] + 1$. Daher gibt es ein $x \in B \setminus A$ mit $\dim[A \cup \{x\}] = \dim[B]$.
- A ist Basis genau dann wenn $[A] = [E]$.
- $r(M) = \dim[E]$.

Satz

Sei $M = (E, S)$ ein Matroid, $w : E \to \mathbb{R}$. Dann löst GREEDY *das Optimierungsproblem „Suche maximales $A \in S$ mit $w(A)$ min!“ korrekt, d.h.* GREEDY *berechnet eine Basis mit minimalem Gewicht.*

Beweis: Sei A die von GREEDY bearbeitete Menge; nach Ausführung gelte $A = C = \{a_1, \ldots, a_r\}$, $w(a_1) \leq \cdots \leq w(a_r)$.

M ist Basis:
Nach Konstruktion gilt $C \in S$. Annahme: C nicht maximal.

Dann gibt es $e \notin C$, sodass $C \cup \{e\} \in S$.
Dann gilt aber auch $A \cup \{e\} \in S$ ↯

$w(A)$ minimal:
Annahme: Es gibt eine Basis $B = \{b_1, \ldots, b_r\}$ mit $w(B) < w(C)$.
Dabei sei $w(b_1) \leq \cdots \leq w(b_r)$.

Sei $i := \min\{j \in \mathbb{N} |\, w(b_j) < w(a_j)\}$.
Weiters sei
$A_{i-1} := \{a_1, a_2, \ldots, a_{i-1}\}$ (das ist A nach $j_i \geq i-1$ Iterationen)
und $B_i := \{b_1, b_2, \ldots, b_i\}$.

Wegen der Austauscheigenschaft existiert $b_j \in B_i \setminus A_{i-1}$ mit
$A_{i-1} \cup \{b_j\} \in S$.
Aber $w(b_j) \leq w(b_i) < w(a_i)$. $\lightning$ $\square$

Satz

Sei $M = (E, S)$ ein Unabhängigkeitssystem. Weiters gelte für jede Gewichtsfunktion $w : E \to \mathbb{R}$: GREEDY *löst das Problem „Suche maximales $A \in S$ mit $w(A)$ min!“ korrekt. Dann ist M ein Matroid.*

Beweis: Annahme: M sei kein Matroid. Dann gibt es $A, B \in S$ mit $|B| = |A| + 1$ derart, dass für jedes $x \in B \setminus A$ die Menge $A \cup \{x\}$ abhängig ist.

Wir definieren die Gewichtsfunktion $w : E \to \mathbb{R}$ wie folgt:

$$w(e) := \begin{cases} -1 - \varepsilon & \text{falls } e \in A; \\ -1 & \text{falls } e \in B \setminus A; \\ 0 & \text{sonst.} \end{cases}$$

GREEDY konstruiert eine Menge $A' \supseteq A$ mit $A' \cap B = \varnothing$.
Es gibt eine Basis $B' \supseteq B$.

Es gilt

$$w(A') = w(A) = -(1+\varepsilon)|A|$$

und

$$\begin{aligned} w(B') \leq w(B) &= -(1+\varepsilon)|A \cap B| - |B \setminus A| \\ &< -|B| = -|A| - 1 \\ &< -(1+\varepsilon)|A| = w(A) \end{aligned}$$

Mit B' gibt es also eine Basis mit geringerem Gewicht als A' ↯ □

14) Kürzeste Pfade

Kürzeste Pfade

Grundlagen

Gegeben: $G = (V, E)$ gerichteter oder ungerichteter Graph,
$w : E \to \mathbb{R}$, $v_0 \in V$
Gesucht: Der gewichtete Abstand $\delta(v_0, v)$, für alle $v \in V$.

Definition

$$\delta(u,v) = \begin{cases} \min_{P(u \to v)} \sum_{e \in P(u \to v)} w(e) & \textit{falls } \exists \textit{ Pfad } P(u \to v), \\ \infty & \textit{sonst.} \end{cases}$$

Bemerkungen:

- Falls G ungerichtet ist, so ist δ eine Metrik auf V.
- Falls G gerichtet, so erfüllt δ die Dreiecksungleichung.
- In der Graphentheorie definiert ein Pfad immer einen **Weg.** In der nichtmathematischen Literatur wird unter Pfad manchmal eine Kantenfolge verstanden; ein Weg wäre dann ein *einfacher Pfad*.

Kürzeste Pfade

Varianten:

- Sei G ungewichtet, also $\delta(u, v) =$ Anzahl der Kanten eines kürzesten Pfades $P(u \to v)$, d.h., $w \equiv 1$: gelöst durch BFS.
- kürzeste Pfade zu einem Zielknoten, also gesucht ist $\delta(v, v_0)$: Orientierung umdrehen.
- Finde $\delta(u, v)$ Knotenpaar (u, v): gelöst durch die Wahl $v_0 = u$.
- Finde kürzeste Pfade für alle Knotenpaare: Setze $v_0 = u$ und durchlaufe alle $u \in V$.

Lemma

$G = (V, E)$, $w : E \to \mathbb{R}$ $p = (v_0, v_1, \ldots, v_k)$ kürzester Pfad von v_0 nach v_k. Dann ist für alle $0 \leq i \leq j \leq k$ der Pfad $(v_i, \ldots, v_j)$ ein kürzester Pfad von v_i nach v_j.

Kürzeste Pfade

Bemerkung: Falls unter einem Pfad eine Kantenfolge verstanden werden soll und von v_0 aus ein Zyklus C mit $w(C) < 0$ erreichbar ist, dann ist $\delta(v_0, v)$ für alle $v \in V$, die vom Zyklus aus erreichbar sind, nicht wohldefiniert. In dem Fall setzen wir $\delta(v_0, v) := -\infty$.

Wir betrachten die Worte **Pfad** und **Weg** als Synonyme.

Die Anzahl der Kanten eines Weges ist durch $|V| - 1$ beschränkt.

Darstellung kürzester Wege mittels Entfernungsbaum:

- $\pi : V \to V \cup \{\text{NIL}\}$
- $G_\pi = (V_\pi, E_\pi)$ mit $V_\pi = \{v \in V \mid \pi(v) \neq \text{NIL}\} \cup \{v_0\}$
- $E_\pi = \{(\pi(v), v) \in E \mid v \in V_\pi \setminus \{v_0\}\}$

Kürzeste Pfade

Relaxation: $D : V \to \mathbb{R}$: vorläufige Distanz, $D(v) \geq \delta(v_0, v)$

Algorithm INIT(G, v_0)

```
1: for v ∈ V do
2:     D(v) := ∞
3:     π(v) :=NIL
4: end for
5: D(v_0) := 0
```

Algorithm RELAX(u, v, w)

```
1: if D(v) > D(u) + w(u, v) then
2:     D(v) := D(u) + w(u, v)
3:     π(v) := u
4: end if
```

Algorithmus: INIT und Folge von RELAX-Schritten
(Dijkstra: eine Relaxation pro Kante,
Bellman-Ford: $|V| - 1$ pro Kante)

Lemma

Es gilt $D(v) \geq \delta(v_0, v) \, \forall v \in V$.

Beweis: Induktion nach der Anzahl der RELAX-Schritte.

I.A.: $D(v_0) = 0 \geq \delta(v_0, v_0) = 0$; $D(v) = \infty \geq \delta(v_0, v)$ für $v \neq v_0$.

I.S.: $D(v) := D(u) + w(u, v) \geq \delta(v_0, u) + w(u, v) \geq \delta(v_0, v)$. □

Lemma

Sei $v_0 \rightsquigarrow u \rightarrow v$ *ein kürzester Weg in* G. *Falls zum Zeitpunkt* t_0 *die Gleichung* $D(u) = \delta(v_0, u)$ *gilt, so gilt nach Aufruf von* $\mathrm{RELAX}(u, v, w)$ *die Gleichung* $D(v) = \delta(v_0, v)$.

Folgerung

Sei $p = (v_0, \ldots, v_k)$ *ein kürzester Weg in* G. *Nach Relaxationen der Kanten* $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ *in dieser Reihenfolge gilt* $D(v_k) = \delta(v_0, v_k)$.

Lemma

Sei $G = (V, E)$ ein Graph, $w : E \to \mathbb{R}$ und für alle Zyklen C gelte $w(C) \geq 0$. Weiters sei $v_0 \in V$.
Angenommen, nach INIT *und mehreren Relaxationen auf G erhalten wir eine Knotenmenge V_π auf der $D(v) = \delta(v_0, v)$ gilt.*
Dann ist G_π ein Baum kürzester Wege mit Wurzel v_0.

Beweis: G_π ist ein Baum: Übung.
Noch zu zeigen: Wenn $p = (v_0, v_1, \ldots, v_k)$ ein Weg in G_π ist, dann ist p ein kürzester Weg von v_0 nach v_k in G.

Laut Voraussetzung gilt $D(v_i) = \delta(v_0, v_i)$ für alle $i = 0, \ldots, k$.
Nach der Relaxation von (v_{i-1}, v_i) keine Relaxation von Kanten der Form (x, v_i), daher gilt $D(v_i) \geq D(v_{i-1}) + w(v_{i-1}, v_i)$.
Daraus folgt $w(v_{i-1}, v_i) \leq \delta(v_0, v_i) - \delta(v_0, v_{i-1})$.
Auswerten einer Teleskopsumme liefert dann $w(p) \leq \delta(v_0, v_k)$, also $w(p) = \delta(v_0, v_k)$. □

Der Algorithmus von Dijkstra

Algorithm DIJKSTRA(G, w, v_0)

```
 1: INIT(G, v_0)
 2: S := ∅
 3: Q := V
 4: while Q ≠ ∅ do
 5:     u :=EXTRACT-MIN(Q)
 6:     S := S ∪ {u}
 7:     for v ∈ Adj[u] do
 8:         RELAX(u, v, w)
 9:     end for
10: end while
```

Bemerkung: Vor jeder Iteration der **while**-Schleife ist $Q = V \setminus S$.

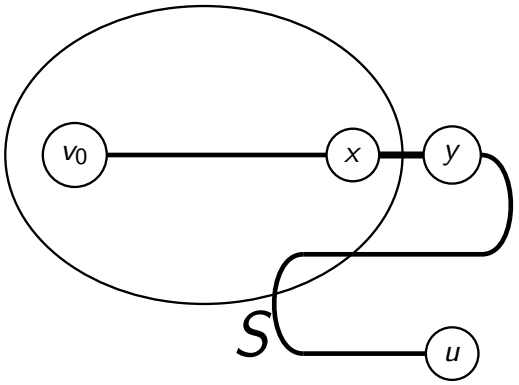
Satz

Sei $G = (V, E)$ ein Graph, $w : E \to \mathbb{R}_0^+$ und $v_0 \in V$.
Dann terminiert $\mathrm{DIJKSTRA}(G, w, v_0)$ mit $D(u) = \delta(v_0, u)$ für alle $u \in V$.

Beweis: Wir verwenden die Schleifeninvariante
„Vor jeder Iteration der **while**-Schleife gilt $D(v) = \delta(v_0, v)$ für alle $v \in S$.“

Aufrechterhaltung:
Betrachte Zeitpunkt, wo $u :=$EXTRACT-MIN(Q) erfolgt.
Annahme: $D(u) > \delta(v_0, u)$.

Der minimale Pfad $P(v_0 \to u)$ verläuft über ein $y \notin S$; sei (x, y) die erste Kante, die S verlässt. Daher $\delta(v_0, y) \leq \delta(v_0, u)$.



Wegen $x \in S$ wurde (x, y) bereits relaxiert.
Daher gilt

$$D(u) > \delta(v_0, u) \geq \delta(v_0, y) = D(y). \lightning \quad \square$$

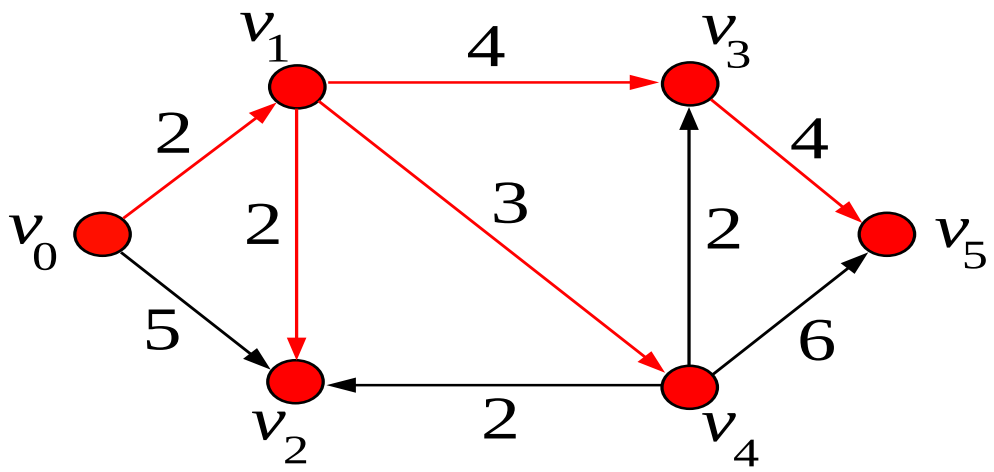
Folgerung

Der von DIJKSTRA *generierte Graph* $G_\pi = (V_\pi, E_\pi)$ *ist ein Entfernungsbaum mit Wurzel* v_0.

Laufzeit: abhängig von der Implementierung der Prioritätswarteschlange Q

- Liste: $\mathcal{O}\left(|V|^2 + |E|\right)$ (EXTRACT-MIN benötigt $\mathcal{O}\left(|V|\right)$);
- Heap: $\mathcal{O}\left((|V| + |E|) \log |V|\right)$ (EXTRACT-MIN benötigt $\mathcal{O}\left(\log |V|\right)$);
- Fibonacci-Heap: $\mathcal{O}\left(|V| \log |V| + |E|\right)$ (EXTRACT-MIN benötigt $\mathcal{O}\left(\log |V|\right)$).

Kürzeste Pfade



	v_0	v_1	v_2	v_3	v_4	v_5	u
0	**0**	∞	∞	∞	∞	∞	v_0
1		**$2/v_0$**	$5/v_0$	∞	∞	∞	v_1
2			**$4/v_1$**	$6/v_1$	$5/v_1$	∞	v_2
3				$6/v_1$	**$5/v_1$**	∞	v_4
4				**$6/v_1$**		$11/v_4$	v_3
5						**$10/v_3$**	v_5

Der Algorithmus von Bellman-Ford

Vorteile:

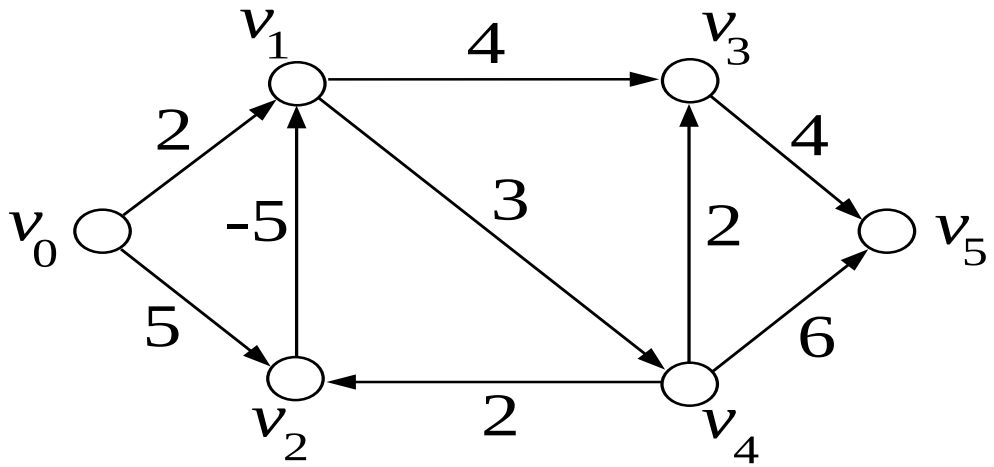
- Kann mit $w : E \to \mathbb{R}$ umgehen;
- erkennt Zyklen negativen Gewichts.

Nachteil: hoher Aufwand ($\mathcal{O}(|V||E|)$).

Algorithm BF(G, w, v_0)

```
 1: INIT(G, v_0)
 2: for i = 1 to |V| − 1 do
 3:     for (u, v) ∈ E do
 4:         RELAX(u, v, w)
 5:     end for
 6: end for
 7: for (u, v) ∈ E do
 8:     if D(v) > D(u) + w(u, v) then
 9:         return FALSE    % Zyklen mit negativen Gewicht!
10:     end if
11: end for
12: return TRUE
```

Kürzeste Pfade



Adj[v] / D/π	v_0	v_1	v_2	v_3	v_4	v_5	i
	$0/-$	$\infty/-$	$\infty/-$	$\infty/-$	$\infty/-$	$\infty/-$	
Adj[v_0]	$0/-$	$2/v_0$	$5/v_0$	$\infty/-$	$\infty/-$	$\infty/-$	1
Adj[v_1]	$0/-$	$2/v_0$	$5/v_0$	$6/v_1$	$5/v_1$	$\infty/-$	1
Adj[v_2]	$0/-$	$0/v_2$	$5/v_0$	$6/v_1$	$5/v_1$	$\infty/-$	1
Adj[v_3]	$0/-$	$0/v_2$	$5/v_0$	$6/v_1$	$5/v_1$	$10/v_3$	1
Adj[v_4]	$0/-$	$0/v_2$	$5/v_0$	$6/v_1$	$5/v_1$	$10/v_3$	1
Adj[v_1]	$0/-$	$0/v_2$	$5/v_0$	$4/v_1$	$3/v_1$	$10/v_3$	2
Adj[v_3]	$0/-$	$0/v_2$	$5/v_0$	$4/v_1$	$3/v_1$	$8/v_3$	2

Korrektheit

Lemma

Sei $G = (V, E)$ ein gerichteter Graph, $w : E \to \mathbb{R}$, $v_0 \in V$. Von v_0 aus seien keine Zyklen C mit $w(C) < 0$ erreichbar. Dann gilt nach $|V| - 1$ Iterationen der **for***-Schleife $D(v) = \delta(v_0, v)$ für alle $v \in V$.*

Beweis: Sei $v \in V$ von v_0 aus erreichbar und $p = (v_0, v_1, \ldots, v_k = v)$ ein kürzester Weg (daher $k \leq |V| - 1$).

BF relaxiert in jeder Iteration alle Kanten. Insbesondere wird (v_{i-1}, v_i) in der i-ten Iteration relaxiert.
Daher werden die Kanten von p in der gegebenen Reihenfolge relaxiert und es folgt $\delta(v_0, v) = D(v)$. □

Folgerung

Es gibt genau dann einen Weg $W(v_0 \to v)$, wenn BF *mit $D(v) < \infty$ terminiert.*

Beweis: Wenn es keinen Pfad gibt, bleibt $D(v) = \infty$.
Andernfalls gibt es einen Weg mit höchstens $|V| - 1$ Kanten, womit nach dem Lemma $D(v) = \delta(v_0, v) < \infty$. □

Satz

Falls in G keine von v_0 aus erreichbaren Zyklen negativen Gewichts existieren, so gibt BF *TRUE aus, für alle Knoten ist $D(v) = \delta(v_0, v)$ und $G_\pi = (V_\pi, E_\pi)$ ist ein Entfernungsbaum. Andernfalls gibt* BF *FALSE aus.*

Beweis:

1. Fall: Es gibt keine solchen Zyklen:

Falls es einen Weg $W(v_0 \to v)$ gibt, dann ist $D(v) = \delta(v_0, v)$, andernfalls $D(v) = \infty$.
Weiters gilt für alle $(u, v) \in E$

$$D(v) = \delta(v_0, v) \leq \delta(v_0, u) + w(u, v) = D(u) + w(u, v).$$

2. Fall: Es gibt solche Zyklen:

Sei $C = (v_0, v_1, \ldots, v_k = v_0)$ ein Zyklus mit $w(C) < 0$.

Daraus folgt $\sum_{i=1}^{k} w(v_{i-1}, v_i) < 0$.

Annahme: BF gibt TRUE aus.
Dann muss für $1 \leq i \leq k$

$$D(v_i) \leq D(v_{i-1}) + w(v_{i-1}, v_i)$$

gelten.
Daraus folgt

$$\sum_{i=1}^{k} D(v_i) \leq \sum_{i=1}^{k} D(v_{i-1}) + w(C) = \sum_{i=1}^{k} D(v_i) + w(C). \; \lightning \qquad \square$$

Der Algorithmus von Floyd und Warshall

Algorithmus für das Kürzeste-Pfade-Problem für alle Knotenpaare.

$V = \{v_1, \ldots, v_n\}$, $W = (w_{i,j})_{1 \leq i,j \leq n}$ mit

$$w_{i,j} = \begin{cases} w(v_i, v_j), & \text{falls } v_i \sim v_j; \\ \infty, & \text{sonst.} \end{cases}$$

Algorithm FLOYD-WARSHALL(W)

```
 1: for i = 1 to n do
 2:     for j = 1 to n do
 3:         d_{i,j} := w_{i,j}
 4:     end for
 5: end for
 6: for i = 1 to n do
 7:     for j = 1 to n do
 8:         for k = 1 to n do
 9:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
10:         end for
11:         if d_{j,j} < 0 then
12:             STOP    % Zyklen mit negativen Gewicht!
13:         end if
14:     end for
15: end for
16: return (d_{i,j})_{1≤i,j≤n}
```

Laufzeit: $\Theta(|V|^3)$.

Kürzeste Pfade

$$D_0 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 8 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 2$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ \mathbf{2} & 0 & 8 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 2$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ \mathbf{2} & 0 & 8 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 2$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} \mathbf{0} & 2 & 4 & \infty & 3 \\ \underline{\mathbf{2}} & 0 & 8 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 2$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & \mathbf{2} & 4 & \infty & 3 \\ \mathbf{2} & \underline{0} & 8 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 2$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & 2 & \mathbf{4} & \infty & 3 \\ \mathbf{2} & 0 & \underline{8} & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 2$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ \mathbf{2} & 0 & 6 & \underline{\infty} & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 2$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & 2 & 4 & \infty & \mathbf{3} \\ \mathbf{2} & 0 & 6 & \infty & \underline{1} \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 2$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ \mathbf{2} & 0 & 6 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 4$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} \mathbf{0} & 2 & 4 & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ \underline{\mathbf{1}} & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 4$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & \mathbf{2} & 4 & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ \mathbf{1} & \underline{\infty} & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 4$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & 2 & \mathbf{4} & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ \mathbf{1} & 3 & \underline{\infty} & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 4$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ \mathbf{1} & 3 & 5 & \underline{0} & 5 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 4$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & 2 & 4 & \infty & \mathbf{3} \\ 2 & 0 & 6 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ \mathbf{1} & 3 & 5 & 0 & \underline{5} \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 1, j = 4$

$$D_0 \longrightarrow D_1 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ \mathbf{1} & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$$D_1 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$i = 2, j = 3$

$$D_1 \longrightarrow D_2 = \begin{pmatrix} 0 & 2 & 4 & \infty & 3 \\ \mathbf{2} & 0 & 6 & \infty & 1 \\ \underline{6} & \mathbf{2} & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

Kürzeste Pfade

$$D_5 = \begin{pmatrix} 0 & 2 & 4 & 4 & 3 \\ 2 & 0 & 6 & 2 & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ 2 & 4 & 6 & 1 & 0 \end{pmatrix}$$

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             d_{j,k} := min(d_{j,k}, d_{j,i} + d_{i,k})
5:         end for
6:     end for
7: end for
```

15) Flüsse

Flüsse

Definition

Ein Flussnetzwerk $G = (V, E, w, s, t)$ *ist ein gerichteter Graph* $G = (V, E)$ *ohne Zyklen der Länge 2 mit einer Gewichtsfunktion* $w : E \to \mathbb{R}_0^+$ *und einem Knoten* $s \in V$ *mit* $d^-(s) = 0$ *und einem Knoten* $t \in V$ *mit* $d^+(t) = 0$.
s heißt Quelle und t Senke des Netzwerks.

Definition

Sei G ein Flussnetzwerk. Eine Funktion $\phi : E \to \mathbb{R}_0^+$ *heißt Fluss auf G, wenn folgende Bedingungen erfüllt sind:*

1. *Kapazitätsbeschränkung:* $\forall e \in E : 0 \leq \phi(e) \leq w(e)$;
2. *Flusserhaltung (Zufluss = Abfluss)*

$$\forall v \in V \setminus \{s, t\} : \sum_{u \in \Gamma^-(v)} \phi(u, v) = \sum_{w \in \Gamma^+(v)} \phi(v, w).$$

Flüsse

Definition

Sei G ein Flussnetzwerk und ϕ ein Fluss auf G. Unter der Größe (oder dem Wert) von ϕ versteht man

$$|\phi| := \sum_{v \in \Gamma^+(s)} \phi(s, v).$$

Lemma

Sei G ein Flussnetzwerk und ϕ ein Fluss auf G. Dann gilt

$$|\phi| = \sum_{v \in \Gamma^+(s)} \phi(s, v) = \sum_{v \in \Gamma^-(t)} \phi(v, t).$$

Maximales Flussproblem:
Gegeben: Flussnetzwerk G
Gesucht: maximaler Fluss, d.h. ein Fluss ϕ, sodass $|\phi|$ maximal ist.

Flüsse

Definition

Sei G ein Flussnetzwerk. Ein Schnitt (S, T) von G ist eine Partition von V, (also $T = V \setminus S$), sodass $s \in S$ und $t \in T$. Die Kapazität von (S, T) ist definiert durch

$$c(S, T) := \sum_{(x,y) \in E : x \in S, y \in T} w(x, y).$$

Lemma

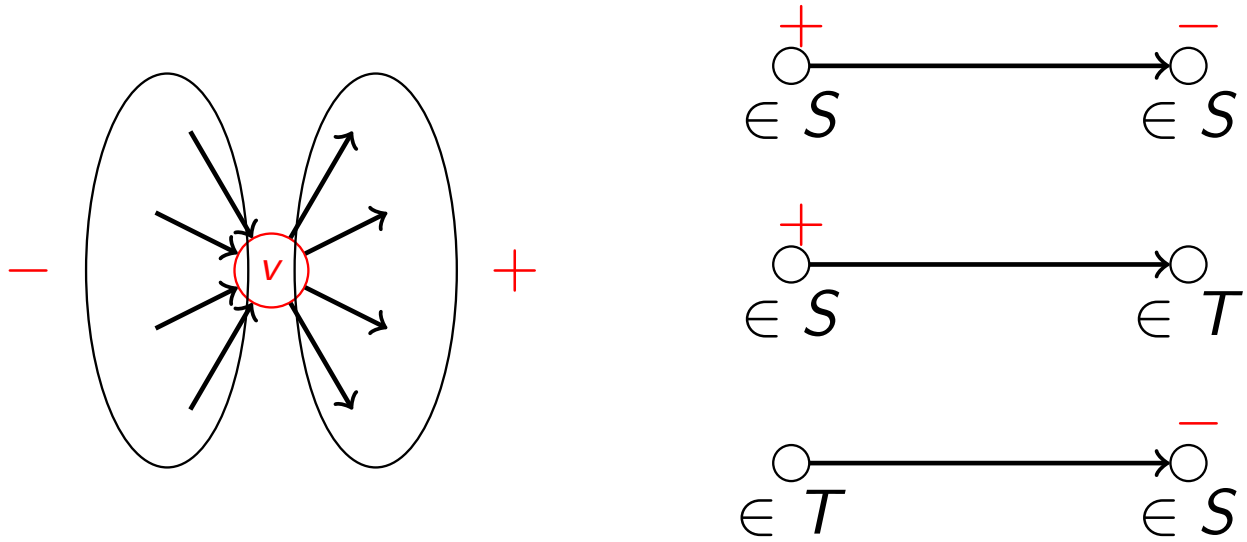
Sei G ein Flussnetzwerk, ϕ ein Fluss auf G und (S, T) ein Schnitt von G. Dann gilt

$$|\phi| = \sum_{(x,y) \in E : x \in S, y \in T} \phi(x, y) - \sum_{(x,y) \in E : x \in T, y \in S} \phi(x, y) \leq c(S, T).$$

Beweis:

$$|\phi| = \sum_{v \in S} \left(\sum_{x \in \Gamma^+(v)} \phi(v,x) - \sum_{y \in \Gamma^-(v)} \phi(y,v) \right),$$

denn Differenz der inneren Summen ist 0, außer für $v = s$.



$$|\phi| = \sum_{(x,y) \in E:\, x \in S, y \in T} \phi(x,y) - \sum_{(x,y) \in E:\, x \in T, y \in S} \phi(x,y) \qquad \square$$

Folgerung

Sei ϕ ein maximaler Fluss auf dem Flussnetzwerk G und (S, T) ein minimaler Schnitt von G ($c(S, T)$ ist minimal). Dann gilt

$$|\phi| \leq c(S, T).$$

Definition

Gegeben sei ein Flussnetzwerk G und ein Fluss ϕ auf G. Ein Pfad $P = (s, v_1, \ldots, v_{k-1}, t)$ des Schattens von G heißt augmentierender Pfad von G bezüglich ϕ, wenn folgende Bedingungen erfüllt sind:

1. *auf allen Vorwärtskanten von P gilt $\phi(e) < w(e)$;*
2. *auf allen Rückwärtskanten von P gilt $\phi(e) > 0$.*

Satz

Sei G ein Flussnetzwerk und ϕ ein Fluss auf G. Dann gilt: ϕ ist genau dann maximal, wenn es keine augmentierenden Pfade bezüglich ϕ gibt.

Beweis: „$\Longrightarrow$“: Annahme: Es existiere ein augmentierender Pfad p.

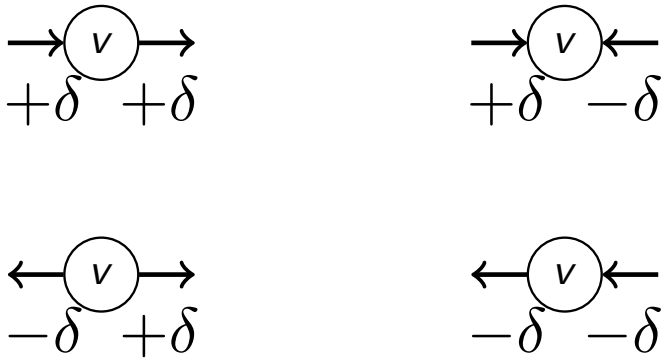
$$\tilde{\phi}(e) := \begin{cases} \phi(e) + \delta, & \text{falls } e \text{ Vorwärtskante von } p, \\ \phi(e) - \delta, & \text{falls } e \text{ Rückwärtskante von } p, \\ \phi(e) & \text{sonst,} \end{cases}$$

wobei $\delta = \min(\delta', \delta'')$ mit $\delta' = \min_{e \in E(p) \text{ vorwärts}} (w(e) - \phi(e))$ und $\delta'' = \min_{e \in E(p) \text{ rückwärts}} \phi(e)$.

Noch zu zeigen: $\tilde{\phi}$ ist ein Fluss.

Flüsse

Flusserhaltung:

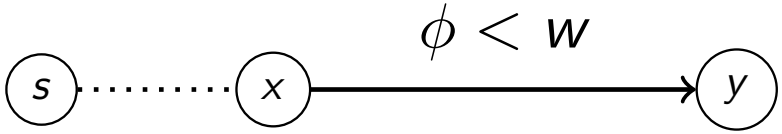


$$|\tilde{\phi}| = |\phi| + \delta > |\phi|$$

„$\Longleftarrow$“:
Wenn es keine augmentierenden Pfade gibt, dann ist t nicht in

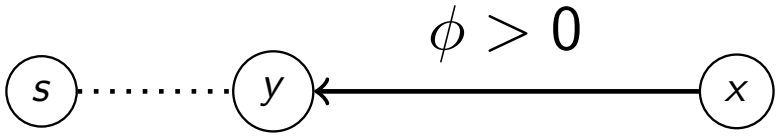
$$S := \{v \in V \mid \exists \text{ augmentierender Pfad } P(s \to v)\}.$$

Somit ist $(S, T) := (S, V \setminus S)$ ein Schnitt.
Kanten (x, y) mit $x \in S$, $y \in T$ sind gesättigt ($\phi(x, y) = w(x, y)$),



$\Longrightarrow y \in S$

Kanten (x, y) mit $x \in T$, $y \in S$ sind leer, d.h. $\phi(x, y) = 0$



$\Longrightarrow x \in S$

$$|\phi| = \sum_{(x,y):\, x \in S, y \in T} \phi(x, y) - \sum_{(x,y):\, x \in T, y \in S} \phi(x, y) = c(S, T),$$

also ist ϕ maximal. □