Transport von Waren:

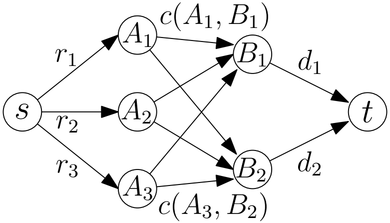
Seehäfen $A_1, \ldots, A_p$ mit Waren, **r_i Einheiten an A_i**

Zielhäfen $B_1, \ldots, B_q$, **d_j Einheiten an B_j angefordert**

Zwischen A_i und B_j gibt es eine Schifffahrtslinie mit **Kapazität $c(A_i, B_j)$**.

Fragen:

1. Ist es möglich, alle Anforderungen zu erfüllen?
2. Wie viele Einheiten können maximal zu den Zielhäfen transportiert werden?
3. Wie sollen die Waren verschifft werden?



Notationen:

- Sei $G = (V, E)$ gerichteter Graph mit **Quelle** $s \in V$ und **Senke** $t \in V$.
- Sei $c : V \times V \rightarrow \mathbb{N}_0$ die **Kapazitätsfunktion** mit $c(u, v) = 0$ für alle $(u, v) \notin E$.
- (G, s, t, c) heißt **Flussnetzwerk**.
- Sei $n = |V|$ und $m = |E|$.
- Sei jeder Knoten von s aus erreichbar. Dies impliziert $m \geq n - 1$.

Definition 5.24

Ein **Fluss** in einem Flussnetzwerk ist eine Funktion $f : V \times V \to \mathbb{R}$ wie folgt:

Definition 5.24

Ein **Fluss** in einem Flussnetzwerk ist eine Funktion $f : V \times V \to \mathbb{R}$ wie folgt:

1. **Flusserhaltung**: Für jeden Knoten $u \in V \setminus \{s, t\}$ gilt

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v).$$

Definition 5.24

Ein **Fluss** in einem Flussnetzwerk ist eine Funktion $f : V \times V \to \mathbb{R}$ wie folgt:

1. **Flusserhaltung**: Für jeden Knoten $u \in V \setminus \{s, t\}$ gilt
$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v).$$
2. **Kapazitätsbeschränkung**: Für alle Knoten $u, v \in V$ gilt $0 \leq f(u, v) \leq c(u, v)$.

Definition 5.24

Ein **Fluss** in einem Flussnetzwerk ist eine Funktion $f : V \times V \to \mathbb{R}$ wie folgt:

1. **Flusserhaltung**: Für jeden Knoten $u \in V \setminus \{s, t\}$ gilt
$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v).$$
2. **Kapazitätsbeschränkung**: Für alle Knoten $u, v \in V$ gilt $0 \leq f(u, v) \leq c(u, v)$.

Wir definieren den **Wert** eines Flusses $f : V \times V \to \mathbb{R}$ als
$$|f| := \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s).$$

Definition 5.24

Ein **Fluss** in einem Flussnetzwerk ist eine Funktion $f : V \times V \to \mathbb{R}$ wie folgt:

1. **Flusserhaltung**: Für jeden Knoten $u \in V \setminus \{s, t\}$ gilt
$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v).$$
2. **Kapazitätsbeschränkung**: Für alle Knoten $u, v \in V$ gilt $0 \leq f(u, v) \leq c(u, v)$.

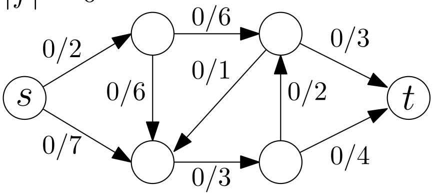
Wir definieren den **Wert** eines Flusses $f : V \times V \to \mathbb{R}$ als
$$|f| := \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s).$$

Maximaler Fluss:

Gegeben sei ein Flussnetzwerk G. Berechne einen **maximalen Fluss** in G,
d. h. einen Fluss f mit **größtmöglichem Wert $|f|$**.

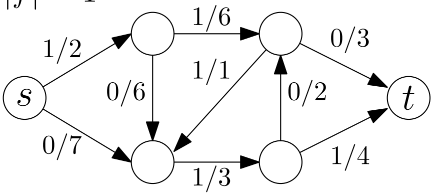
Beispiel:

$|f| = 0$



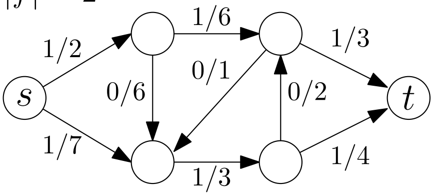
Beispiel:

$|f| = 1$



Beispiel:

$|f| = 2$



Lemma 5.25

Sei f ein Fluss in einem Flussnetzwerk G. Dann gilt

$$|f| := \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) = \sum_{v \in V} f(v, t) - \sum_{v \in V} f(t, v).$$

Lemma 5.25

Sei f ein Fluss in einem Flussnetzwerk G. Dann gilt

$$|f| := \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) = \sum_{v \in V} f(v, t) - \sum_{v \in V} f(t, v).$$

Beweis: Es gilt

$$\sum_{u \in V} \sum_{v \in V} f(v, u) = \sum_{u \in V} \sum_{v \in V} f(u, v)$$

Lemma 5.25

Sei f ein Fluss in einem Flussnetzwerk G. Dann gilt

$$|f| := \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) = \sum_{v \in V} f(v,t) - \sum_{v \in V} f(t,v).$$

Beweis: Es gilt

$$\sum_{u \in V} \sum_{v \in V} f(v,u) = \sum_{u \in V} \sum_{v \in V} f(u,v)$$

$$\iff \sum_{u \in \{s,t\}} \sum_{v \in V} f(v,u) + \sum_{u \in V \setminus \{s,t\}} \sum_{v \in V} f(v,u) = \sum_{u \in \{s,t\}} \sum_{v \in V} f(u,v) + \sum_{u \in V \setminus \{s,t\}} \sum_{v \in V} f(u,v).$$

Lemma 5.25

Sei f ein Fluss in einem Flussnetzwerk G. Dann gilt

$$|f| := \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) = \sum_{v \in V} f(v,t) - \sum_{v \in V} f(t,v).$$

Beweis: Es gilt

$$\sum_{u \in V} \sum_{v \in V} f(v,u) = \sum_{u \in V} \sum_{v \in V} f(u,v)$$

$$\iff \sum_{u \in \{s,t\}} \sum_{v \in V} f(v,u) + \sum_{u \in V \setminus \{s,t\}} \sum_{v \in V} f(v,u) = \sum_{u \in \{s,t\}} \sum_{v \in V} f(u,v) + \sum_{u \in V \setminus \{s,t\}} \sum_{v \in V} f(u,v).$$

Nun nutzen wir die Flusserhaltung für alle Knoten $u \in V \setminus \{s,t\}$ und erhalten:

$$\sum_{v \in V} f(v,s) + \sum_{v \in V} f(v,t) = \sum_{v \in V} f(s,v) + \sum_{v \in V} f(t,v)$$

Lemma 5.25

Sei f ein Fluss in einem Flussnetzwerk G. Dann gilt

$$|f| := \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) = \sum_{v \in V} f(v,t) - \sum_{v \in V} f(t,v).$$

Beweis: Es gilt

$$\sum_{u \in V} \sum_{v \in V} f(v,u) = \sum_{u \in V} \sum_{v \in V} f(u,v)$$

$$\iff \sum_{u \in \{s,t\}} \sum_{v \in V} f(v,u) + \sum_{u \in V \setminus \{s,t\}} \sum_{v \in V} f(v,u) = \sum_{u \in \{s,t\}} \sum_{v \in V} f(u,v) + \sum_{u \in V \setminus \{s,t\}} \sum_{v \in V} f(u,v).$$

Nun nutzen wir die Flusserhaltung für alle Knoten $u \in V \setminus \{s,t\}$ und erhalten:

$$\sum_{v \in V} f(v,s) + \sum_{v \in V} f(v,t) = \sum_{v \in V} f(s,v) + \sum_{v \in V} f(t,v)$$

$$\iff \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) = \sum_{v \in V} f(v,t) - \sum_{v \in V} f(t,v). \quad \square$$

5.4.1 Anwendungsbeispiel

Transport von Waren:

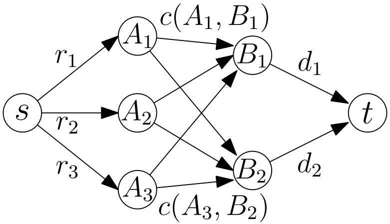
Seehäfen $A_1, \ldots, A_p$ mit Waren, **r_i Einheiten an A_i**

Zielhäfen $B_1, \ldots, B_q$, **d_j Einheiten an B_j angefordert**

Zwischen A_i und B_j gibt es eine Schifffahrtslinie mit **Kapazität $c(A_i, B_j)$**.

Fragen:

1. Ist es möglich, alle Anforderungen zu erfüllen?
2. Wie viele Einheiten können maximal zu den Zielhäfen transportiert werden?
3. Wie sollen die Waren verschifft werden?



5.4.2 Algorithmus von Ford und Fulkerson

Annahme: G enthält für kein Paar $u, v \in V$ die Kanten (u, v) und (v, u).

```
FORD-FULKERSON(G, c, s ∈ V, t ∈ V)
    Setze f(e) = 0 für alle e ∈ E.   // f ist gültiger Fluss mit Wert 0
    while (∃ flussvergrößernder Weg P) {
        Erhöhe den Fluss f entlang P.
    }
    return f;
```

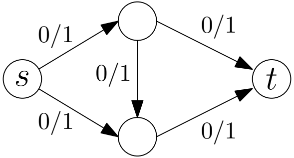
5.4.2 Algorithmus von Ford und Fulkerson

Annahme: G enthält für kein Paar $u, v \in V$ die Kanten (u, v) und (v, u).

```
FORD-FULKERSON(G, c, s ∈ V, t ∈ V)
    Setze f(e) = 0 für alle e ∈ E.   // f ist gültiger Fluss mit Wert 0
    while (∃ flussvergrößernder Weg P) {
        Erhöhe den Fluss f entlang P.
    }
    return f;
```

Was ist ein **flussvergrößernder Weg**?

1. Versuch: Nicht ausgelasteter s-t-Weg.



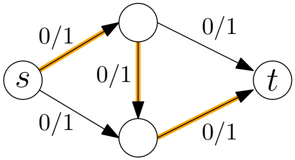
5.4.2 Algorithmus von Ford und Fulkerson

Annahme: G enthält für kein Paar $u, v \in V$ die Kanten (u, v) und (v, u).

```
FORD-FULKERSON(G, c, s ∈ V, t ∈ V)
    Setze f(e) = 0 für alle e ∈ E.   // f ist gültiger Fluss mit Wert 0
    while (∃ flussvergrößernder Weg P) {
        Erhöhe den Fluss f entlang P.
    }
    return f;
```

Was ist ein **flussvergrößernder Weg**?

1. Versuch: Nicht ausgelasteter s-t-Weg.



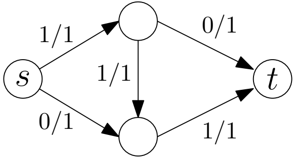
5.4.2 Algorithmus von Ford und Fulkerson

Annahme: G enthält für kein Paar $u, v \in V$ die Kanten (u, v) und (v, u).

```
FORD-FULKERSON(G, c, s ∈ V, t ∈ V)
    Setze f(e) = 0 für alle e ∈ E.   // f ist gültiger Fluss mit Wert 0
    while (∃ flussvergrößernder Weg P) {
        Erhöhe den Fluss f entlang P.
    }
    return f;
```

Was ist ein **flussvergrößernder Weg**?

1. Versuch: Nicht ausgelasteter s-t-Weg.



Definition 5.26

Sei $G = (V, E)$ ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}_0$ und sei f ein Fluss in G. Das dazugehörige **Restnetzwerk** $G_f = (V, E_f)$ ist auf der gleichen Menge von Knoten V definiert wie das Netzwerk G.

Definition 5.26

Sei $G = (V, E)$ ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}_0$ und sei f ein Fluss in G. Das dazugehörige **Restnetzwerk** $G_f = (V, E_f)$ ist auf der gleichen Menge von Knoten V definiert wie das Netzwerk G. Wir definieren eine Funktion $\mathrm{rest}_f : V \times V \to \mathbb{R}$ mit

$$\mathrm{rest}_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{falls } (u, v) \in E, \\ f(v, u) & \text{falls } (v, u) \in E, \\ 0 & \text{sonst.} \end{cases}$$

5.4.2 Algorithmus von Ford und Fulkerson

Definition 5.26

Sei $G = (V, E)$ ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}_0$ und sei f ein Fluss in G. Das dazugehörige **Restnetzwerk** $G_f = (V, E_f)$ ist auf der gleichen Menge von Knoten V definiert wie das Netzwerk G. Wir definieren eine Funktion $\mathrm{rest}_f : V \times V \to \mathbb{R}$ mit

$$\mathrm{rest}_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{falls } (u, v) \in E, \\ f(v, u) & \text{falls } (v, u) \in E, \\ 0 & \text{sonst.} \end{cases}$$

Die Kantenmenge E_f ist definiert als

$$E_f = \{(u, v) \in V \times V \mid \mathrm{rest}_f(u, v) > 0\}.$$

5.4.2 Algorithmus von Ford und Fulkerson

Definition 5.26

Sei $G = (V, E)$ ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}_0$ und sei f ein Fluss in G. Das dazugehörige **Restnetzwerk** $G_f = (V, E_f)$ ist auf der gleichen Menge von Knoten V definiert wie das Netzwerk G. Wir definieren eine Funktion $\mathrm{rest}_f : V \times V \to \mathbb{R}$ mit

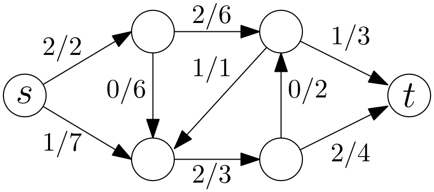
$$\mathrm{rest}_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{falls } (u, v) \in E, \\ f(v, u) & \text{falls } (v, u) \in E, \\ 0 & \text{sonst.} \end{cases}$$

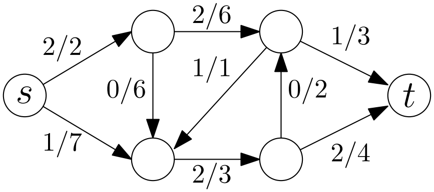
Die Kantenmenge E_f ist definiert als

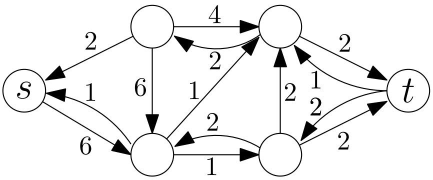
$$E_f = \{(u, v) \in V \times V \mid \mathrm{rest}_f(u, v) > 0\}.$$

Ein **flussvergrößernder Weg** ist ein einfacher Weg von s nach t im Restnetzwerk G_f.

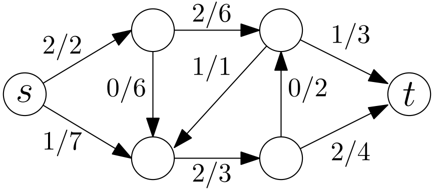
5.4.2 Algorithmus von Ford und Fulkerson

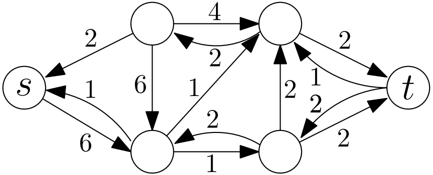


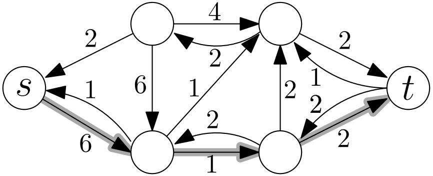

2/2
2/6
1/3
0/6
1/1
0/2
1/7
2/3
2/4
s
t

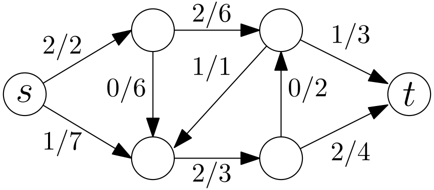

4
2
2
2
1
2
6
1
2
2
1
6
2
1
2
s
t

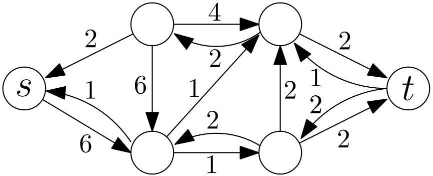
5.4.2 Algorithmus von Ford und Fulkerson

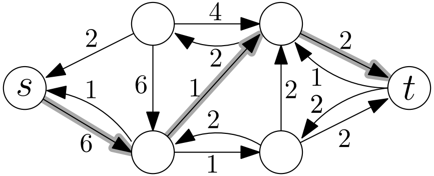


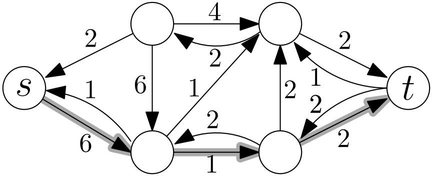





s
t
2/2
2/6
1/3
0/6
1/1
0/2
1/7
2/3
2/4


s
t
4
2
2
2
1
6
1
2
1
2
2
2
6
1


s
t
4
2
2
2
1
6
1
2
1
2
2
2
6
1


s
t
4
2
2
2
1
6
1
2
1
2
2
2
6
1

Definition 5.27

Sei G ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}$, sei $f : V \times V \to \mathbb{R}$ ein Fluss und sei P ein einfacher Weg im Restnetzwerk G_f von s nach t.

Definition 5.27

Sei G ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}$, sei $f : V \times V \to \mathbb{R}$ ein Fluss und sei P ein einfacher Weg im Restnetzwerk G_f von s nach t. **Wir bezeichnen mit $f \uparrow P : V \times V \to \mathbb{R}$ den Fluss, der entsteht, wenn wir f entlang P erhöhen.**

Definition 5.27

Sei G ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}$, sei $f : V \times V \to \mathbb{R}$ ein Fluss und sei P ein einfacher Weg im Restnetzwerk G_f von s nach t. **Wir bezeichnen mit $f \uparrow P : V \times V \to \mathbb{R}$ den Fluss, der entsteht, wenn wir f entlang P erhöhen.** Dieser Fluss ist definiert durch

$$(f \uparrow P)(u, v) = \begin{cases} f(u, v) + \delta & \text{falls } (u, v) \in E \text{ und } (u, v) \in P, \\ f(u, v) - \delta & \text{falls } (u, v) \in E \text{ und } (v, u) \in P, \\ f(u, v) & \text{sonst,} \end{cases}$$

wobei

$$\delta = \min_{e \in P}(\mathrm{rest}_f(e)).$$

5.4.2 Algorithmus von Ford und Fulkerson

Definition 5.27

Sei G ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}$, sei $f : V \times V \to \mathbb{R}$ ein Fluss und sei P ein einfacher Weg im Restnetzwerk G_f von s nach t. **Wir bezeichnen mit $f \uparrow P : V \times V \to \mathbb{R}$ den Fluss, der entsteht, wenn wir f entlang P erhöhen.** Dieser Fluss ist definiert durch

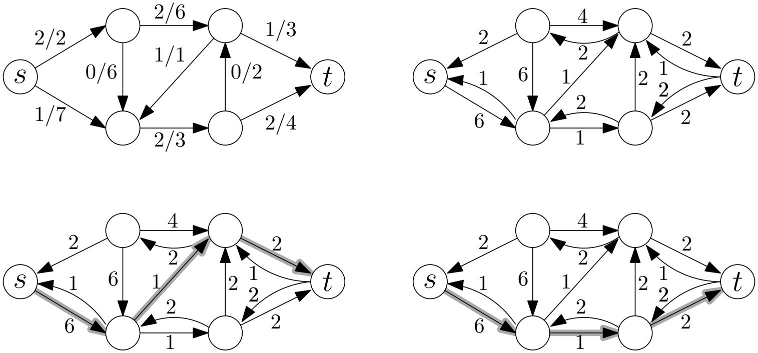
$$(f \uparrow P)(u, v) = \begin{cases} f(u, v) + \delta & \text{falls } (u, v) \in E \text{ und } (u, v) \in P, \\ f(u, v) - \delta & \text{falls } (u, v) \in E \text{ und } (v, u) \in P, \\ f(u, v) & \text{sonst,} \end{cases}$$

wobei

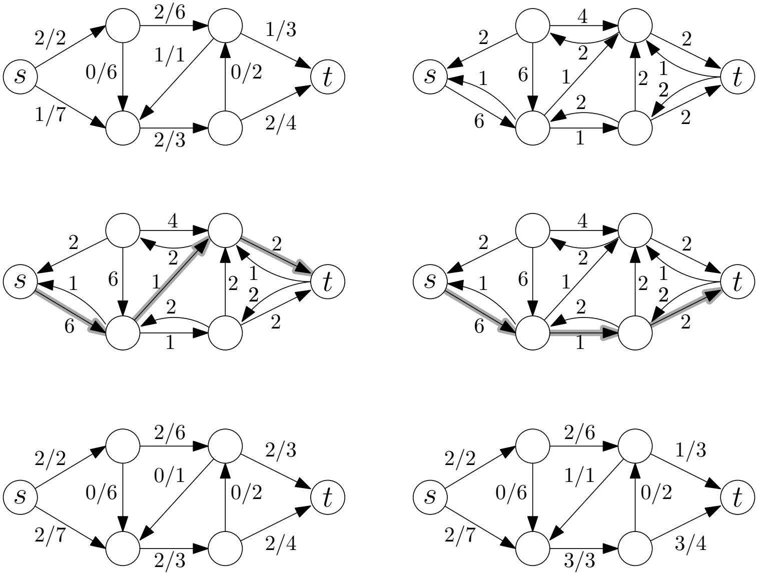
$$\delta = \min_{e \in P}(\mathrm{rest}_f(e)).$$

Aus der Definition von E_f folgt, dass $\delta > 0$ gilt.

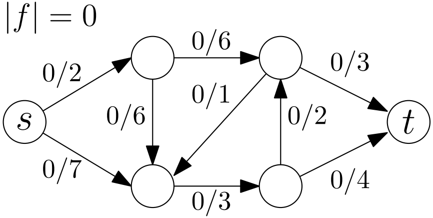
5.4.2 Algorithmus von Ford und Fulkerson



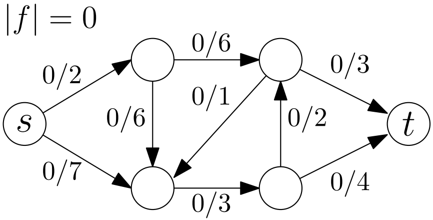
5.4.2 Algorithmus von Ford und Fulkerson

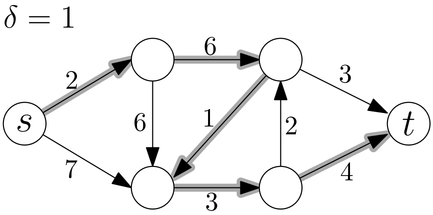


Beispiel:

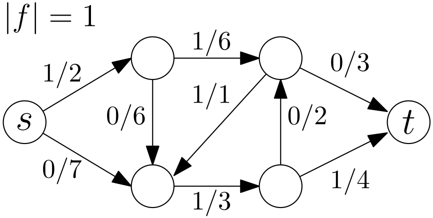


Beispiel:

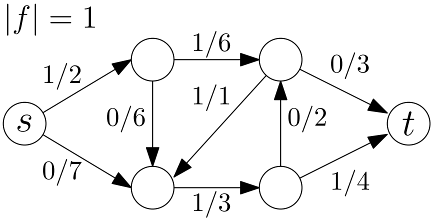


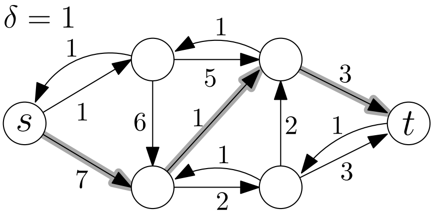


Beispiel:



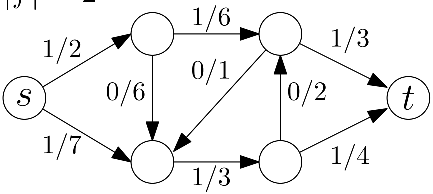
Beispiel:



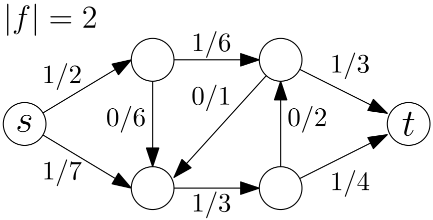


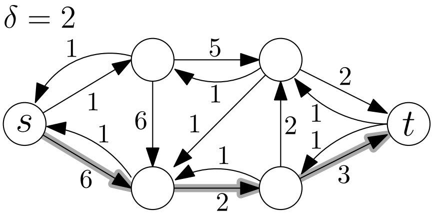
Beispiel:

$|f| = 2$

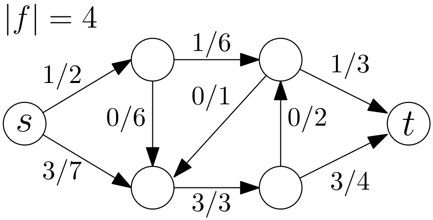


Beispiel:

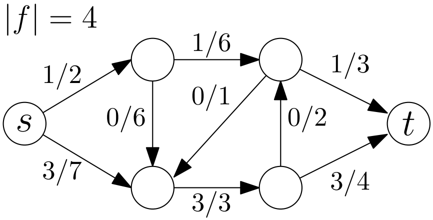


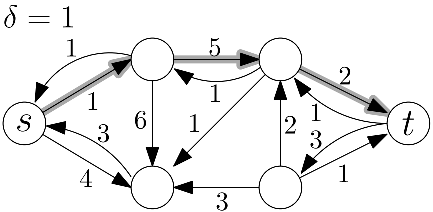


Beispiel:



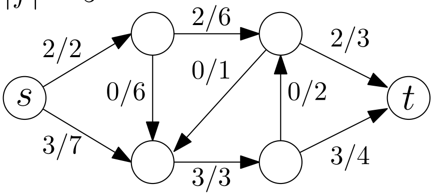
Beispiel:





Beispiel:

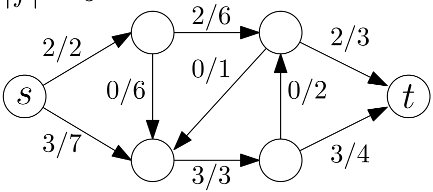
$|f| = 5$

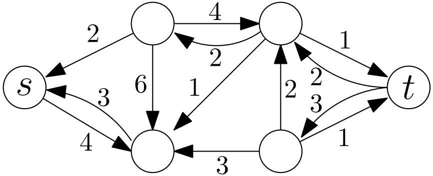


5.4.2 Algorithmus von Ford und Fulkerson

Beispiel:

$|f| = 5$





5.4.2 Algorithmus von Ford und Fulkerson

Lemma 5.28

Sei f ein Fluss in einem Netzwerk G und sei P ein Weg von s nach t im Restnetzwerk G_f.

Die Funktion $f \uparrow P : V \times V \to \mathbb{R}$ ist wieder ein Fluss.

Lemma 5.28

Sei f ein Fluss in einem Netzwerk G und sei P ein Weg von s nach t im Restnetzwerk G_f.

Die Funktion $f \uparrow P : V \times V \to \mathbb{R}$ ist wieder ein Fluss. Für diesen Fluss gilt

$$|f \uparrow P| = |f| + \delta.$$

Lemma 5.28

Sei f ein Fluss in einem Netzwerk G und sei P ein Weg von s nach t im Restnetzwerk G_f.

Die Funktion $f \uparrow P : V \times V \to \mathbb{R}$ ist wieder ein Fluss. Für diesen Fluss gilt

$$|f \uparrow P| = |f| + \delta.$$

Beweis: **Flusserhaltung**: Sei $u \in V \setminus \{s, t\}$. Falls **u nicht auf P**, so gilt $f(u, v) = (f \uparrow P)(u, v)$ und $f(v, u) = (f \uparrow P)(v, u)$ für alle Knoten $v \in V$.

5.4.2 Algorithmus von Ford und Fulkerson

Lemma 5.28

Sei f ein Fluss in einem Netzwerk G und sei P ein Weg von s nach t im Restnetzwerk G_f.

Die Funktion $f \uparrow P : V \times V \to \mathbb{R}$ ist wieder ein Fluss. Für diesen Fluss gilt

$$|f \uparrow P| = |f| + \delta.$$

Beweis: **Flusserhaltung**: Sei $u \in V \setminus \{s, t\}$. Falls **u nicht auf P**, so gilt $f(u, v) = (f \uparrow P)(u, v)$ und $f(v, u) = (f \uparrow P)(v, u)$ für alle Knoten $v \in V$.

Falls **u auf P**, dann sei $(v, u) \in P$ und $(u, v') \in P$:

$$G_f \quad \begin{tikzcd} v \arrow[r, "\in E_f"'] & u \arrow[r, "\in E_f"'] & v' \end{tikzcd}$$

Lemma 5.28

Sei f ein Fluss in einem Netzwerk G und sei P ein Weg von s nach t im Restnetzwerk G_f. **Die Funktion $f \uparrow P : V \times V \to \mathbb{R}$ ist wieder ein Fluss.** Für diesen Fluss gilt

$$|f \uparrow P| = |f| + \delta.$$

Beweis: Flusserhaltung: Sei $u \in V \setminus \{s, t\}$. Falls **u nicht auf P**, so gilt $f(u, v) = (f \uparrow P)(u, v)$ und $f(v, u) = (f \uparrow P)(v, u)$ für alle Knoten $v \in V$.

Falls **u auf P**, dann sei $(v, u) \in P$ und $(u, v') \in P$:

$$G_f \qquad v \xrightarrow{\in E_f} u \xrightarrow{\in E_f} v'$$

Fallunterscheidung der Kanten in G:

$$v \xrightarrow{\in E} u \xrightarrow{\in E} v' \qquad\qquad v \xrightarrow{\in E} u \xleftarrow{\in E} v'$$

$$v \xleftarrow{\in E} u \xrightarrow{\in E} v' \qquad\qquad v \xleftarrow{\in E} u \xleftarrow{\in E} v'$$

5.4.2 Algorithmus von Ford und Fulkerson

Lemma 5.28

Sei f ein Fluss in einem Netzwerk G und sei P ein Weg von s nach t im Restnetzwerk G_f.

Die Funktion $f \uparrow P : V \times V \to \mathbb{R}$ ist wieder ein Fluss. Für diesen Fluss gilt

$$|f \uparrow P| = |f| + \delta.$$

Beweis: **Flusserhaltung**: Sei $u \in V \setminus \{s, t\}$. Falls **u nicht auf P**, so gilt $f(u, v) = (f \uparrow P)(u, v)$ und $f(v, u) = (f \uparrow P)(v, u)$ für alle Knoten $v \in V$.

Falls **u auf P**, dann sei $(v, u) \in P$ und $(u, v') \in P$:

$$G_f \quad v \xrightarrow[\in E_f]{} u \xrightarrow[\in E_f]{} v'$$

Fallunterscheidung der Kanten in G:

$$v \xrightarrow[\in E]{+\delta} u \xrightarrow[\in E]{+\delta} v' \qquad\qquad v \xrightarrow[\in E]{} u \xleftarrow[\in E]{} v'$$

$$v \xleftarrow[\in E]{} u \xrightarrow[\in E]{} v' \qquad\qquad v \xleftarrow[\in E]{} u \xleftarrow[\in E]{} v'$$

5.4.2 Algorithmus von Ford und Fulkerson

Lemma 5.28

Sei f ein Fluss in einem Netzwerk G und sei P ein Weg von s nach t im Restnetzwerk G_f.

Die Funktion $f \uparrow P : V \times V \to \mathbb{R}$ ist wieder ein Fluss. Für diesen Fluss gilt

$$|f \uparrow P| = |f| + \delta.$$

Beweis: **Flusserhaltung**: Sei $u \in V \setminus \{s, t\}$. Falls **u nicht auf P**, so gilt $f(u, v) = (f \uparrow P)(u, v)$ und $f(v, u) = (f \uparrow P)(v, u)$ für alle Knoten $v \in V$.

Falls **u auf P**, dann sei $(v, u) \in P$ und $(u, v') \in P$:

$$G_f \quad \begin{tikzcd} v \arrow[r, "\in E_f"'] & u \arrow[r, "\in E_f"'] & v' \end{tikzcd}$$

Fallunterscheidung der Kanten in G:

$$\begin{tikzcd} v \arrow[r, "+\delta", "\in E"'] & u \arrow[r, "+\delta", "\in E"'] & v' \end{tikzcd} \qquad \begin{tikzcd} v \arrow[r, "+\delta", "\in E"'] & u & v' \arrow[l, "-\delta"', "\in E"] \end{tikzcd}$$

$$\begin{tikzcd} v & u \arrow[l, "\in E"] \arrow[r, "\in E"'] & v' \end{tikzcd} \qquad \begin{tikzcd} v & u \arrow[l, "\in E"] & v' \arrow[l, "\in E"] \end{tikzcd}$$

Lemma 5.28

Sei f ein Fluss in einem Netzwerk G und sei P ein Weg von s nach t im Restnetzwerk G_f. **Die Funktion $f \uparrow P : V \times V \to \mathbb{R}$ ist wieder ein Fluss.** Für diesen Fluss gilt

$$|f \uparrow P| = |f| + \delta.$$

Beweis: **Flusserhaltung**: Sei $u \in V \setminus \{s, t\}$. Falls **u nicht auf P**, so gilt $f(u, v) = (f \uparrow P)(u, v)$ und $f(v, u) = (f \uparrow P)(v, u)$ für alle Knoten $v \in V$.

Falls **u auf P**, dann sei $(v, u) \in P$ und $(u, v') \in P$:

$$G_f \quad \begin{tikzcd} v \arrow[r, "\in E_f"'] & u \arrow[r, "\in E_f"'] & v' \end{tikzcd}$$

Fallunterscheidung der Kanten in G:

$$\begin{tikzcd} v \arrow[r, "+\delta", "\in E"'] & u \arrow[r, "+\delta", "\in E"'] & v' \end{tikzcd} \qquad \begin{tikzcd} v \arrow[r, "+\delta", "\in E"'] & u & v' \arrow[l, "-\delta"', "\in E"] \end{tikzcd}$$

$$\begin{tikzcd} v & u \arrow[l, "-\delta"', "\in E"] \arrow[r, "+\delta", "\in E"'] & v' \end{tikzcd} \qquad \begin{tikzcd} v & u \arrow[l, "\in E"] & v' \arrow[l, "\in E"] \end{tikzcd}$$

5.4.2 Algorithmus von Ford und Fulkerson

Lemma 5.28

Sei f ein Fluss in einem Netzwerk G und sei P ein Weg von s nach t im Restnetzwerk G_f. **Die Funktion $f \uparrow P : V \times V \to \mathbb{R}$ ist wieder ein Fluss.** Für diesen Fluss gilt

$$|f \uparrow P| = |f| + \delta.$$

Beweis: **Flusserhaltung**: Sei $u \in V \setminus \{s, t\}$. Falls **u nicht auf P**, so gilt $f(u, v) = (f \uparrow P)(u, v)$ und $f(v, u) = (f \uparrow P)(v, u)$ für alle Knoten $v \in V$.

Falls **u auf P**, dann sei $(v, u) \in P$ und $(u, v') \in P$:

$$G_f \quad \begin{tikzcd} v \arrow[r, "\in E_f"'] & u \arrow[r, "\in E_f"'] & v' \end{tikzcd}$$

Fallunterscheidung der Kanten in G:

$$\begin{tikzcd} v \arrow[r, "+\delta", "\in E"'] & u \arrow[r, "+\delta", "\in E"'] & v' \end{tikzcd} \qquad \begin{tikzcd} v \arrow[r, "+\delta", "\in E"'] & u & v' \arrow[l, "-\delta"', "\in E"] \end{tikzcd}$$

$$\begin{tikzcd} v & u \arrow[l, "-\delta"', "\in E"] \arrow[r, "+\delta", "\in E"'] & v' \end{tikzcd} \qquad \begin{tikzcd} v & u \arrow[l, "-\delta"', "\in E"] & v' \arrow[l, "-\delta"', "\in E"] \end{tikzcd}$$

Kapazitätsbeschränkung: Sei $e = (u, v) \in E_f$ eine Kante auf dem Weg P.

Kapazitätsbeschränkung: Sei $e = (u, v) \in E_f$ eine Kante auf dem Weg P.

- Ist $\mathbf{e} = (\mathbf{u}, \mathbf{v}) \in \mathbf{E}$, so erhöhen wir den Fluss auf e um δ:

$$0 \leq f(e) + \delta \leq f(e) + \mathrm{rest}_f(e) = f(e) + (c(e) - f(e)) = c(e).$$

5.4.2 Algorithmus von Ford und Fulkerson

Kapazitätsbeschränkung: Sei $e = (u, v) \in E_f$ eine Kante auf dem Weg P.

- Ist $\mathbf{e} = (\mathbf{u}, \mathbf{v}) \in \mathbf{E}$, so erhöhen wir den Fluss auf e um δ:

$$0 \leq f(e) + \delta \leq f(e) + \mathrm{rest}_f(e) = f(e) + (c(e) - f(e)) = c(e).$$

- Ist $\mathbf{e}' = (\mathbf{v}, \mathbf{u}) \in \mathbf{E}$, so verringern wir den Fluss auf e' um δ:

$$c(e') \geq f(e') - \delta \geq f(e') - \mathrm{rest}_f(e) \geq f(e') - f(e') = 0.$$

Wir zeigen, dass für den **Wert des Flusses** gilt $|\mathbf{f} \uparrow \mathbf{P}| = |\mathbf{f}| + \delta$:

P enthält genau eine zu s inzidente Kante $(s, v) \in E_f$.

Wir zeigen, dass für den **Wert des Flusses** gilt $|\mathbf{f} \uparrow \mathbf{P}| = |\mathbf{f}| + \delta$:

P enthält genau eine zu s inzidente Kante $(s, v) \in E_f$.

- Gilt $(\mathbf{s}, \mathbf{v}) \in \mathbf{E}$, so gilt $(f \uparrow P)(s, v) = f(s, v) + \delta$.

Wir zeigen, dass für den **Wert des Flusses** gilt $|\mathbf{f} \uparrow \mathbf{P}| = |\mathbf{f}| + \delta$:
P enthält genau eine zu s inzidente Kante $(s, v) \in E_f$.

- Gilt $(\mathbf{s}, \mathbf{v}) \in \mathbf{E}$, so gilt $(f \uparrow P)(s, v) = f(s, v) + \delta$.
- Gilt $(\mathbf{v}, \mathbf{s}) \in \mathbf{E}$, so gilt $(f \uparrow P)(v, s) = f(v, s) - \delta$. Somit erhöht sich auch in diesem Fall der Wert des Flusses um δ. $\square$

Definition 5.29

Sei $G = (V, E)$ ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}$. Sei $s \in V$ die Quelle und $t \in V$ die Senke. Ein **Schnitt** von G ist eine Partition der Knotenmenge V in zwei Teile $S \subseteq V$ und $T \subseteq V$ mit $s \in S$, $t \in T$ und $T = V \setminus S$.

Definition 5.29

Sei $G = (V, E)$ ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}$. Sei $s \in V$ die Quelle und $t \in V$ die Senke. Ein **Schnitt** von G ist eine Partition der Knotenmenge V in zwei Teile $S \subseteq V$ und $T \subseteq V$ mit $s \in S$, $t \in T$ und $T = V \setminus S$. Wir nennen

$$c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$$

die **Kapazität** des Schnittes (S, T). Ein Schnitt (S, T) heißt **minimal**, wenn es keinen Schnitt (S', T') mit $c(S', T') < c(S, T)$ gibt.

5.4.2 Algorithmus von Ford und Fulkerson

Definition 5.29

Sei $G = (V, E)$ ein Flussnetzwerk mit Kapazitäten $c : V \times V \to \mathbb{N}$. Sei $s \in V$ die Quelle und $t \in V$ die Senke. Ein **Schnitt** von G ist eine Partition der Knotenmenge V in zwei Teile $S \subseteq V$ und $T \subseteq V$ mit $s \in S$, $t \in T$ und $T = V \setminus S$. Wir nennen

$$c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$$

die **Kapazität** des Schnittes (S, T). Ein Schnitt (S, T) heißt **minimal**, wenn es keinen Schnitt (S', T') mit $c(S', T') < c(S, T)$ gibt.

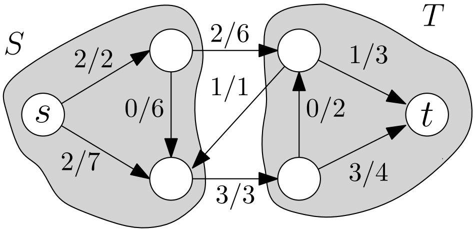
Für einen Fluss f und einen Schnitt (S, T) sei

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u)$$

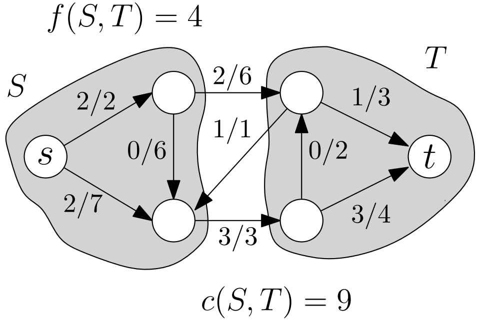
der **Fluss über den Schnitt**.

5.4.2 Algorithmus von Ford und Fulkerson

Beispiel:



Beispiel:



Lemma 5.30

Sei (S, T) ein Schnitt eines Flussnetzwerkes G und sei f ein Fluss in G. Dann gilt

$$|f| = f(S, T) \leq c(S, T).$$

Lemma 5.30

Sei (S, T) ein Schnitt eines Flussnetzwerkes G und sei f ein Fluss in G. Dann gilt

$$|f| = f(S, T) \leq c(S, T).$$

Beweis: Summe über alle Kanten innerhalb von S:

$$\sum_{u \in S} \sum_{v \in S} f(u, v) = \sum_{u \in S} \sum_{v \in S} f(v, u).$$

Lemma 5.30

Sei (S, T) ein Schnitt eines Flussnetzwerkes G und sei f ein Fluss in G. Dann gilt

$$|f| = f(S, T) \leq c(S, T).$$

Beweis: Summe über alle Kanten innerhalb von S:

$$\sum_{u \in S} \sum_{v \in S} f(u, v) = \sum_{u \in S} \sum_{v \in S} f(v, u).$$

Durch Abtrennen der Summanden für $u = s$ erhalten wir

$$\sum_{v \in S} f(s, v) + \sum_{u \in S \setminus \{s\}} \sum_{v \in S} f(u, v) = \sum_{v \in S} f(v, s) + \sum_{u \in S \setminus \{s\}} \sum_{v \in S} f(v, u)$$

$$\iff \sum_{v \in S} f(s, v) - \sum_{v \in S} f(v, s) = \sum_{u \in S \setminus \{s\}} \sum_{v \in S} f(v, u) - \sum_{u \in S \setminus \{s\}} \sum_{v \in S} f(u, v).$$

Es gilt also:

$$\sum_{v \in S} f(s, v) - \sum_{v \in S} f(v, s) = \sum_{u \in S \setminus \{s\}} \left(\sum_{v \in S} f(v, u) - \sum_{v \in S} f(u, v) \right) \tag{1}$$

Für jeden Knoten $u \in S \setminus \{s\}$ gilt die Flusserhaltung:

$$\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u)$$

5.4.2 Algorithmus von Ford und Fulkerson

Es gilt also:

$$\sum_{v \in S} f(s, v) - \sum_{v \in S} f(v, s) = \sum_{u \in S \setminus \{s\}} \left(\sum_{v \in S} f(v, u) - \sum_{v \in S} f(u, v) \right) \tag{1}$$

Für jeden Knoten $u \in S \setminus \{s\}$ gilt die Flusserhaltung:

$$\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u)$$

$$\iff \sum_{v \in T} f(u, v) - \sum_{v \in T} f(v, u) = \sum_{v \in S} f(v, u) - \sum_{v \in S} f(u, v)$$

Es gilt also:

$$\sum_{v \in S} f(s,v) - \sum_{v \in S} f(v,s) = \sum_{u \in S \setminus \{s\}} \left(\sum_{v \in S} f(v,u) - \sum_{v \in S} f(u,v) \right) \tag{1}$$

Für jeden Knoten $u \in S \setminus \{s\}$ gilt die Flusserhaltung:

$$\begin{aligned} \sum_{v \in V} f(u,v) &= \sum_{v \in V} f(v,u) \\ \iff \sum_{v \in T} f(u,v) - \sum_{v \in T} f(v,u) &= \sum_{v \in S} f(v,u) - \sum_{v \in S} f(u,v) \end{aligned}$$

Damit können wir (1) schreiben als

$$\sum_{v \in S} f(s,v) - \sum_{v \in S} f(v,s) = \sum_{u \in S \setminus \{s\}} \left(\sum_{v \in T} f(u,v) - \sum_{v \in T} f(v,u) \right) . \tag{2}$$

Für die Quelle s gilt nach Definition des Wertes

$$|f| = \left(\sum_{v \in S} f(s, v) + \sum_{v \in T} f(s, v) \right) - \left(\sum_{v \in S} f(v, s) + \sum_{v \in T} f(v, s) \right)$$

Für die Quelle s gilt nach Definition des Wertes

$$|f| = \left(\sum_{v \in S} f(s,v) + \sum_{v \in T} f(s,v)\right) - \left(\sum_{v \in S} f(v,s) + \sum_{v \in T} f(v,s)\right)$$

$$\iff \sum_{v \in S} f(s,v) - \sum_{v \in S} f(v,s) = |f| + \sum_{v \in T} f(v,s) - \sum_{v \in T} f(s,v). \tag{3}$$

Für die Quelle s gilt nach Definition des Wertes

$$
\begin{aligned}
|f| &= \left(\sum_{v \in S} f(s,v) + \sum_{v \in T} f(s,v)\right) - \left(\sum_{v \in S} f(v,s) + \sum_{v \in T} f(v,s)\right) \\
\iff \sum_{v \in S} f(s,v) - \sum_{v \in S} f(v,s) &= |f| + \sum_{v \in T} f(v,s) - \sum_{v \in T} f(s,v). \qquad (3)
\end{aligned}
$$

Wir setzen (3) in (2) ein und erhalten

$$
|f| + \sum_{v \in T} f(v,s) - \sum_{v \in T} f(s,v) = \sum_{u \in S \setminus \{s\}} \sum_{v \in T} f(u,v) - \sum_{u \in S \setminus \{s\}} \sum_{v \in T} f(v,u)
$$

$$
\iff |f| = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) = f(S,T).
$$

Für die Quelle s gilt nach Definition des Wertes

$$|f| = \left(\sum_{v\in S} f(s,v) + \sum_{v\in T} f(s,v)\right) - \left(\sum_{v\in S} f(v,s) + \sum_{v\in T} f(v,s)\right)$$
$$\iff \sum_{v\in S} f(s,v) - \sum_{v\in S} f(v,s) = |f| + \sum_{v\in T} f(v,s) - \sum_{v\in T} f(s,v). \tag{3}$$

Wir setzen (3) in (2) ein und erhalten

$$|f| + \sum_{v\in T} f(v,s) - \sum_{v\in T} f(s,v) = \sum_{u\in S\setminus\{s\}} \sum_{v\in T} f(u,v) - \sum_{u\in S\setminus\{s\}} \sum_{v\in T} f(v,u)$$
$$\iff |f| = \sum_{u\in S} \sum_{v\in T} f(u,v) - \sum_{u\in S} \sum_{v\in T} f(v,u) = f(S,T).$$

Die Ungleichung folgt, da $f(u,v) \le c(u,v)$ und $f(v,u) \ge 0$ für alle $u, v \in V$:

$$f(S,T) \le \sum_{u\in S} \sum_{v\in T} f(u,v) \le \sum_{u\in S} \sum_{v\in T} c(u,v) = c(S,T). \quad \square$$

Theorem 5.31

Sei f ein Fluss in einem Netzwerk G. Dann sind die folgenden drei Aussagen äquivalent.

a) f ist ein maximaler Fluss.

b) Das Restnetzwerk G_f enthält keinen flussvergrößernden Weg.

c) Es gibt einen Schnitt (S, T) mit $|f| = c(S, T)$.

5.4.2 Algorithmus von Ford und Fulkerson

Theorem 5.31

Sei f ein Fluss in einem Netzwerk G. Dann sind die folgenden drei Aussagen äquivalent.

a) f ist ein maximaler Fluss.

b) Das Restnetzwerk G_f enthält keinen flussvergrößernden Weg.

c) Es gibt einen Schnitt (S, T) mit $|f| = c(S, T)$.

Beweis: **a) $\Rightarrow$ b)** Widerspruchsbeweis:

Gibt es flussvergrößernden Weg P in G_f, so gilt $|f \uparrow P| = |f| + \delta > |f|$.

Theorem 5.31

Sei f ein Fluss in einem Netzwerk G. Dann sind die folgenden drei Aussagen äquivalent.

a) f ist ein maximaler Fluss.

b) Das Restnetzwerk G_f enthält keinen flussvergrößernden Weg.

c) Es gibt einen Schnitt (S, T) mit $|f| = c(S, T)$.

Beweis: **a) $\Rightarrow$ b)** Widerspruchsbeweis:
Gibt es flussvergrößernden Weg P in G_f, so gilt $|f \uparrow P| = |f| + \delta > |f|$.

c) $\Rightarrow$ a) Sei f ein Fluss mit $|f| = c(S, T)$ für einen Schnitt (S, T) und sei f' ein maximaler Fluss. Dann gilt $|f| \leq |f'|$. Mit Lemma 5.30 folgt $|f'| \leq c(S, T)$ und damit

$$c(S, T) = |f| \leq |f'| \leq c(S, T),$$

woraus $c(S, T) = |f| = |f'|$ folgt. Damit ist auch f ein maximaler Fluss.

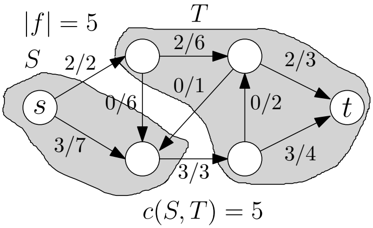
b) $\Rightarrow$ **c)** Laut Voraussetzung gibt es keinen s-t-Weg in G_f. Setze

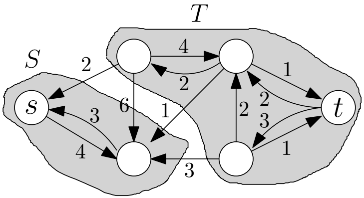
$$S = \{v \in V \mid \text{es gibt Weg in } G_f \text{ von } s \text{ nach } v\} \quad \text{und} \quad T = V \setminus S.$$

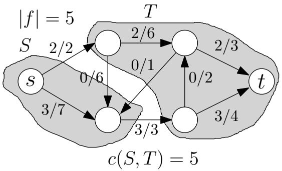
b) $\Rightarrow$ c) Laut Voraussetzung gibt es keinen s-t-Weg in G_f. Setze

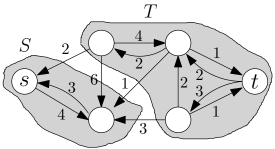
$$S = \{v \in V \mid \text{es gibt Weg in } G_f \text{ von } s \text{ nach } v\} \quad \text{und} \quad T = V \setminus S.$$

Es gilt $s \in S$ und $t \in T$. Damit ist (S, T) ein Schnitt.



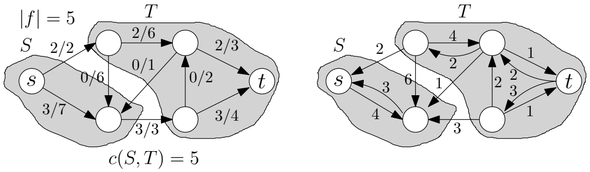






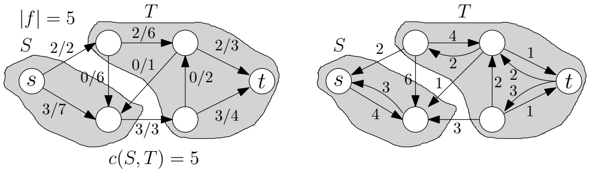
Für diesen Schnitt (S, T) gilt $|f| = c(S, T)$:

5.4.2 Algorithmus von Ford und Fulkerson



Für diesen Schnitt (S, T) gilt $|f| = c(S, T)$:

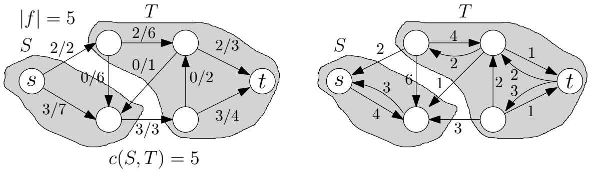
- Sei $(u, v) \in E$ mit $u \in S$ und $v \in T$. Da u in G_f von s aus erreichbar ist, v aber nicht, gilt $(u, v) \notin E_f$. Daraus folgt $\mathrm{rest}_f(u, v) = 0$, also $f(u, v) = c(u, v)$.



Für diesen Schnitt (S, T) gilt $|f| = c(S, T)$:

- Sei $(u, v) \in E$ mit $u \in S$ und $v \in T$. Da u in G_f von s aus erreichbar ist, v aber nicht, gilt $(u, v) \notin E_f$. Daraus folgt $\mathrm{rest}_f(u, v) = 0$, also $f(u, v) = c(u, v)$.
- Sei $(v, u) \in E$ mit $u \in S$ und $v \in T$. Da u in G_f von s aus erreichbar ist, v aber nicht, gilt $(u, v) \notin E_f$. Daraus folgt $\mathrm{rest}_f(u, v) = 0$, also $f(v, u) = 0$.

5.4.2 Algorithmus von Ford und Fulkerson



Für diesen Schnitt (S, T) gilt $|f| = c(S, T)$:

- Sei $(u, v) \in E$ mit $u \in S$ und $v \in T$. Da u in G_f von s aus erreichbar ist, v aber nicht, gilt $(u, v) \notin E_f$. Daraus folgt $\mathrm{rest}_f(u, v) = 0$, also $f(u, v) = c(u, v)$.
- Sei $(v, u) \in E$ mit $u \in S$ und $v \in T$. Da u in G_f von s aus erreichbar ist, v aber nicht, gilt $(u, v) \notin E_f$. Daraus folgt $\mathrm{rest}_f(u, v) = 0$, also $f(v, u) = 0$.

Mit Lemma 5.30 gilt somit

$$c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u) = f(S, T) = |f| \quad \square$$

5.4.2 Algorithmus von Ford und Fulkerson

Theorem 5.32

Für ganzzahlige Kapazitäten $c : V \times V \to \mathbb{N}_0$ ist die Anzahl an Iterationen der while-Schleife im Algorithmus von Ford und Fulkerson durch $C = \sum_{e \in E} c(e)$ nach oben beschränkt. Die Laufzeit des Algorithmus beträgt $O(mC)$.

Theorem 5.32

Für ganzzahlige Kapazitäten $c : V \times V \to \mathbb{N}_0$ ist die Anzahl an Iterationen der while-Schleife im Algorithmus von Ford und Fulkerson durch $C = \sum_{e \in E} c(e)$ nach oben beschränkt. Die Laufzeit des Algorithmus beträgt $O(mC)$.

Beweis:

Es gilt stets $f : V \times V \to \mathbb{N}_0$ gilt. Damit ist auch stets $\delta \in \mathbb{N}$. Der Wert des Flusses steigt mit jeder Iteration der while-Schleife um mindestens eins. Er ist durch C nach oben beschränkt.

5.4.2 Algorithmus von Ford und Fulkerson

Theorem 5.32

Für ganzzahlige Kapazitäten $c : V \times V \to \mathbb{N}_0$ ist die Anzahl an Iterationen der while-Schleife im Algorithmus von Ford und Fulkerson durch $C = \sum_{e \in E} c(e)$ nach oben beschränkt. Die Laufzeit des Algorithmus beträgt $O(mC)$.

Beweis:

Es gilt stets $f : V \times V \to \mathbb{N}_0$ gilt. Damit ist auch stets $\delta \in \mathbb{N}$. Der Wert des Flusses steigt mit jeder Iteration der while-Schleife um mindestens eins. Er ist durch C nach oben beschränkt.

Restnetzwerk G_f kann in Zeit $O(m)$ berechnet werden. Ein s-t-Weg P in G_f kann mittels Tiefensuche in Zeit $O(m)$ gefunden werden. Fluss $f \uparrow P$ kann in Zeit $O(m)$ berechnet werden. Insgesamt dauert also jede Iteration Zeit $O(m)$. $\square$

5.4.2 Algorithmus von Ford und Fulkerson

Theorem 5.32

Für ganzzahlige Kapazitäten $c : V \times V \to \mathbb{N}_0$ ist die Anzahl an Iterationen der while-Schleife im Algorithmus von Ford und Fulkerson durch $C = \sum_{e \in E} c(e)$ nach oben beschränkt. Die Laufzeit des Algorithmus beträgt $O(mC)$.

Beweis:

Es gilt stets $f : V \times V \to \mathbb{N}_0$ gilt. Damit ist auch stets $\delta \in \mathbb{N}$. Der Wert des Flusses steigt mit jeder Iteration der while-Schleife um mindestens eins. Er ist durch C nach oben beschränkt.

Restnetzwerk G_f kann in Zeit $O(m)$ berechnet werden. Ein s-t-Weg P in G_f kann mittels Tiefensuche in Zeit $O(m)$ gefunden werden. Fluss $f \uparrow P$ kann in Zeit $O(m)$ berechnet werden. Insgesamt dauert also jede Iteration Zeit $O(m)$. $\square$

Korollar 5.33

Sind alle Kapazitäten ganzzahlig, so gibt es stets einen ganzzahligen maximalen Fluss.

5.4.3 Algorithmus von Edmonds und Karp

EDMONDS-KARP$(G, c, s \in V, t \in V)$

Setze $f(e) = 0$ für alle $e \in E$. // f ist gültiger Fluss mit Wert 0
while ($\exists$ flussvergrößernder Weg P) $\{$
    Wähle einen flussvergrößernden Weg P **mit so wenig Kanten wie möglich**.
    Erhöhe den Fluss f entlang P.
$\}$
return f;

EDMONDS-KARP$(G, c, s \in V, t \in V)$

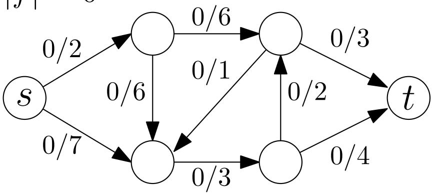
Setze $f(e) = 0$ für alle $e \in E$. // f ist gültiger Fluss mit Wert 0
while ($\exists$ flussvergrößernder Weg P) {
    Wähle einen flussvergrößernden Weg P **mit so wenig Kanten wie möglich**.
    Erhöhe den Fluss f entlang P.
}
return f;

Theorem 5.34

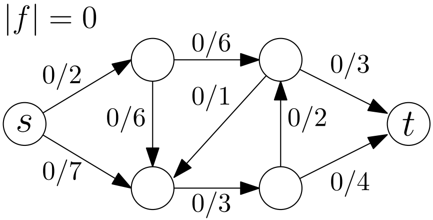
Der Algorithmus von Edmonds und Karp besitzt (auch für Graphen mit nicht ganzzahligen Kapazitäten) eine Laufzeit von $O(m^2 n) = O(n^5)$.

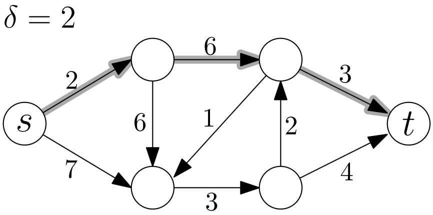
Beispiel:

$|f| = 0$

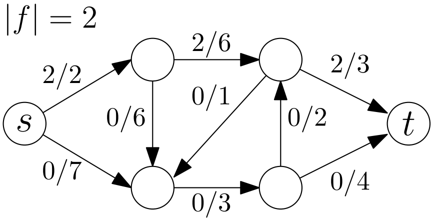


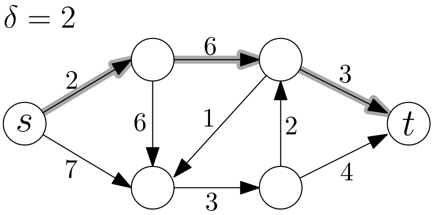
Beispiel:



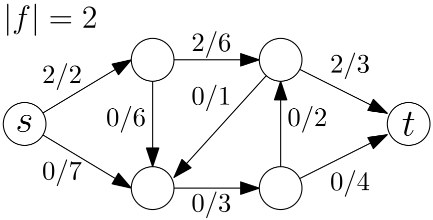


Beispiel:

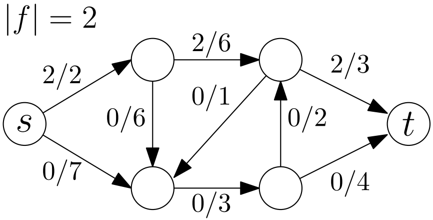


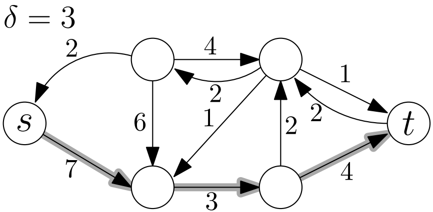


Beispiel:

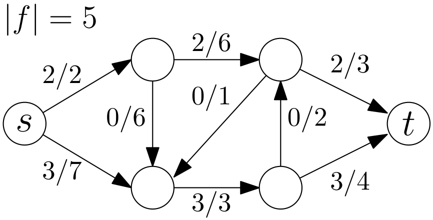


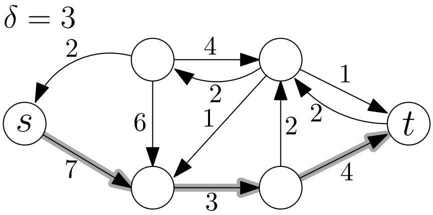
Beispiel:



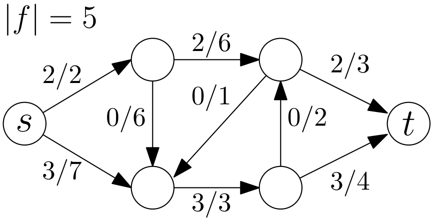


Beispiel:



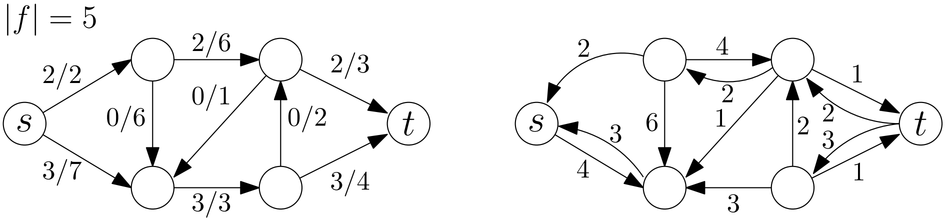


Beispiel:



5.4.3 Algorithmus von Edmonds und Karp

Beispiel:



Lemma 5.35

Die Distanz von s zu jedem $x \in V$ in G_f wird im Laufe des Algorithmus nicht kleiner.

5.4.3 Algorithmus von Edmonds und Karp

Lemma 5.35

Die Distanz von s zu jedem $x \in V$ in G_f wird im Laufe des Algorithmus nicht kleiner.

Beweis: Betrachte Iteration von f zu $f \uparrow P$.

Die Kantenmenge $E_{f \uparrow P}$ unterscheidet sich von der Kantenmenge E_f wie folgt.

Lemma 5.35

Die Distanz von s zu jedem $x \in V$ in G_f wird im Laufe des Algorithmus nicht kleiner.

Beweis: Betrachte Iteration von f zu $f \uparrow P$.

Die Kantenmenge $E_{f \uparrow P}$ unterscheidet sich von der Kantenmenge E_f wie folgt.

- Für jede Kante $(u, v) \in P \subseteq E_f$ verringert sich $\mathrm{rest}_f(u, v)$ um δ, das heißt $\mathrm{rest}_{f \uparrow P}(u, v) = \mathrm{rest}_f(u, v) - \delta$. Eine Kante mit $\mathrm{rest}_f(u, v) = \delta$ heißt **Flaschenhalskante** und sie ist in $E_{f \uparrow P}$ nicht mehr enthalten.

Lemma 5.35

Die Distanz von s zu jedem $x \in V$ in G_f wird im Laufe des Algorithmus nicht kleiner.

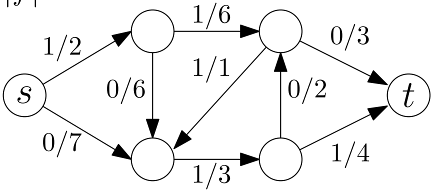
Beweis: Betrachte Iteration von f zu $f \uparrow P$.

Die Kantenmenge $E_{f \uparrow P}$ unterscheidet sich von der Kantenmenge E_f wie folgt.

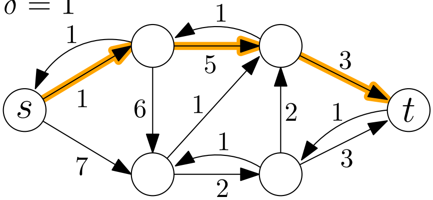
- Für jede Kante $(u, v) \in P \subseteq E_f$ verringert sich $\mathrm{rest}_f(u, v)$ um δ, das heißt $\mathrm{rest}_{f \uparrow P}(u, v) = \mathrm{rest}_f(u, v) - \delta$. Eine Kante mit $\mathrm{rest}_f(u, v) = \delta$ heißt **Flaschenhalskante** und sie ist in $E_{f \uparrow P}$ nicht mehr enthalten.
- Für jede Kante $(u, v) \in P \subseteq E_f$ erhöht sich die Restkapazität $\mathrm{rest}_f(v, u)$ der entgegengesetzten Kante um δ, das heißt $\mathrm{rest}_{f \uparrow P}(v, u) = \mathrm{rest}_f(v, u) + \delta$. War $\mathrm{rest}_f(v, u) = 0$, so war $(v, u) \notin E_f$, nun gilt aber $(v, u) \in E_{f \uparrow P}$.

Beispiel:

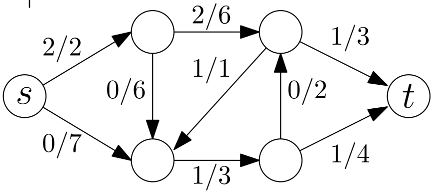
$|f| = 1$

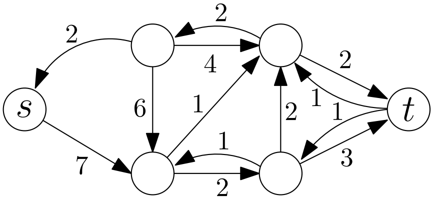


$\delta = 1$



$|f \uparrow P| = 2$





5.4.3 Algorithmus von Edmonds und Karp

Übergang von E_f zu $E_{f\uparrow P}$ in mehreren Schritten:

Übergang von E_f zu $E_{f\uparrow P}$ in mehreren Schritten:

1. **Füge neue Kanten ins Restnetzwerk ein.**

   Einfügen einer Kante (v, u) mit $(u, v) \in P$.

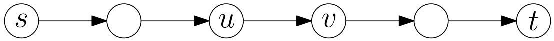
5.4.3 Algorithmus von Edmonds und Karp

Übergang von E_f zu $E_{f\uparrow P}$ in mehreren Schritten:

1. **Füge neue Kanten ins Restnetzwerk ein.**

   Einfügen einer Kante (v, u) mit $(u, v) \in P$.

   Diese Kante kann den Abstand von s zu x nicht reduzieren.



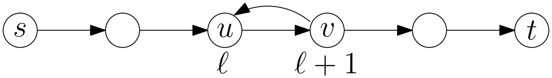
5.4.3 Algorithmus von Edmonds und Karp

Übergang von E_f zu $E_{f\uparrow P}$ in mehreren Schritten:

1. **Füge neue Kanten ins Restnetzwerk ein.**

   Einfügen einer Kante (v, u) mit $(u, v) \in P$.

   Diese Kante kann den Abstand von s zu x nicht reduzieren.



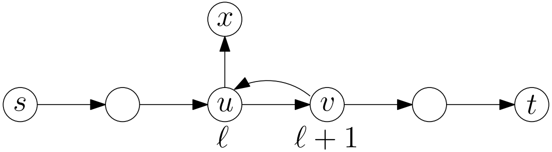
5.4.3 Algorithmus von Edmonds und Karp

Übergang von E_f zu $E_{f\uparrow P}$ in mehreren Schritten:

1. **Füge neue Kanten ins Restnetzwerk ein.**

   Einfügen einer Kante (v, u) mit $(u, v) \in P$.

   Diese Kante kann den Abstand von s zu x nicht reduzieren.

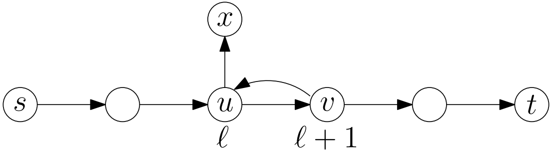


Übergang von E_f zu $E_{f \uparrow P}$ in mehreren Schritten:

1. **Füge neue Kanten ins Restnetzwerk ein.**

   Einfügen einer Kante (v, u) mit $(u, v) \in P$.

   Diese Kante kann den Abstand von s zu x nicht reduzieren.



2. **Entferne alle Flaschenhalskanten.**

   Löschen von Kanten kann Distanzen nicht verringern. $\square$

5.4.3 Algorithmus von Edmonds und Karp

Theorem 5.34

Der Algorithmus von Edmonds und Karp besitzt (auch für Graphen mit nicht ganzzahligen Kapazitäten) eine Laufzeit von $O(m^2 n) = O(n^5)$.

5.4.3 Algorithmus von Edmonds und Karp

Theorem 5.34

Der Algorithmus von Edmonds und Karp besitzt (auch für Graphen mit nicht ganzzahligen Kapazitäten) eine Laufzeit von $O(m^2 n) = O(n^5)$.

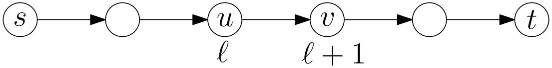
Beweis: Bei jeder Iteration gibt es mindestens eine Flaschenhalskante (u, v). Diese wird aus Restnetzwerk gelöscht. Wird sie später wieder eingefügt, so muss Distanz von s zu u um mindestens 2 gestiegen sein.

5.4.3 Algorithmus von Edmonds und Karp

Theorem 5.34

Der Algorithmus von Edmonds und Karp besitzt (auch für Graphen mit nicht ganzzahligen Kapazitäten) eine Laufzeit von $O(m^2 n) = O(n^5)$.

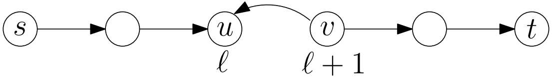
Beweis: Bei jeder Iteration gibt es mindestens eine Flaschenhalskante (u, v). Diese wird aus Restnetzwerk gelöscht. Wird sie später wieder eingefügt, so muss Distanz von s zu u um mindestens 2 gestiegen sein.



Theorem 5.34

Der Algorithmus von Edmonds und Karp besitzt (auch für Graphen mit nicht ganzzahligen Kapazitäten) eine Laufzeit von $O(m^2 n) = O(n^5)$.

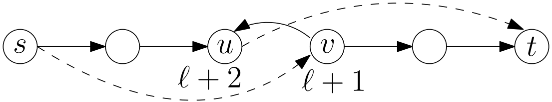
Beweis: Bei jeder Iteration gibt es mindestens eine Flaschenhalskante (u, v). Diese wird aus Restnetzwerk gelöscht. Wird sie später wieder eingefügt, so muss Distanz von s zu u um mindestens 2 gestiegen sein.



Theorem 5.34

Der Algorithmus von Edmonds und Karp besitzt (auch für Graphen mit nicht ganzzahligen Kapazitäten) eine Laufzeit von $O(m^2 n) = O(n^5)$.

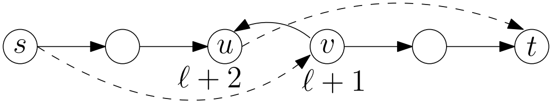
Beweis: Bei jeder Iteration gibt es mindestens eine Flaschenhalskante (u, v). Diese wird aus Restnetzwerk gelöscht. Wird sie später wieder eingefügt, so muss Distanz von s zu u um mindestens 2 gestiegen sein.



Theorem 5.34

Der Algorithmus von Edmonds und Karp besitzt (auch für Graphen mit nicht ganzzahligen Kapazitäten) eine Laufzeit von $O(m^2 n) = O(n^5)$.

Beweis: Bei jeder Iteration gibt es mindestens eine Flaschenhalskante (u, v). Diese wird aus Restnetzwerk gelöscht. Wird sie später wieder eingefügt, so muss Distanz von s zu u um mindestens 2 gestiegen sein.



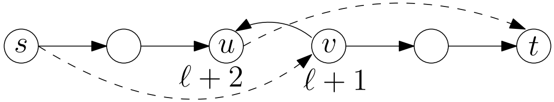
Maximal mögliche Distanz von s zu jedem Knoten ist $n - 1$.

5.4.3 Algorithmus von Edmonds und Karp

Theorem 5.34

Der Algorithmus von Edmonds und Karp besitzt (auch für Graphen mit nicht ganzzahligen Kapazitäten) eine Laufzeit von $O(m^2 n) = O(n^5)$.

Beweis: Bei jeder Iteration gibt es mindestens eine Flaschenhalskante (u, v). Diese wird aus Restnetzwerk gelöscht. Wird sie später wieder eingefügt, so muss Distanz von s zu u um mindestens 2 gestiegen sein.



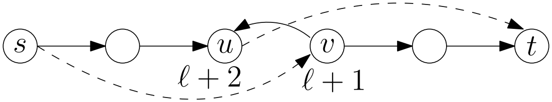
Maximal mögliche Distanz von s zu jedem Knoten ist $n - 1$.

Eine Kante kann nicht öfter als $n/2$ mal entfernt werden.

Theorem 5.34

Der Algorithmus von Edmonds und Karp besitzt (auch für Graphen mit nicht ganzzahligen Kapazitäten) eine Laufzeit von $O(m^2 n) = O(n^5)$.

Beweis: Bei jeder Iteration gibt es mindestens eine Flaschenhalskante (u, v). Diese wird aus Restnetzwerk gelöscht. Wird sie später wieder eingefügt, so muss Distanz von s zu u um mindestens 2 gestiegen sein.



Maximal mögliche Distanz von s zu jedem Knoten ist $n - 1$.

Eine Kante kann nicht öfter als $n/2$ mal entfernt werden.

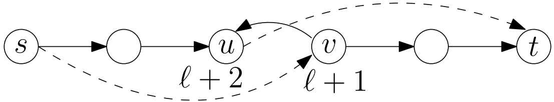
Anzahl Iterationen maximal $\frac{n}{2} \cdot 2m = nm$.

5.4.3 Algorithmus von Edmonds und Karp

Theorem 5.34

Der Algorithmus von Edmonds und Karp besitzt (auch für Graphen mit nicht ganzzahligen Kapazitäten) eine Laufzeit von $O(m^2n) = O(n^5)$.

Beweis: Bei jeder Iteration gibt es mindestens eine Flaschenhalskante (u, v). Diese wird aus Restnetzwerk gelöscht. Wird sie später wieder eingefügt, so muss Distanz von s zu u um mindestens 2 gestiegen sein.



Maximal mögliche Distanz von s zu jedem Knoten ist $n - 1$.

Eine Kante kann nicht öfter als $n/2$ mal entfernt werden.

Anzahl Iterationen maximal $\frac{n}{2} \cdot 2m = nm$.

Eine Iteration besitzt Laufzeit $O(m)$ (BFS). $\square$