Technische Universität Braunschweig

Netzwerkalgorithmen – Vorlesung #7

Arne Schmidt

Kapitel 4

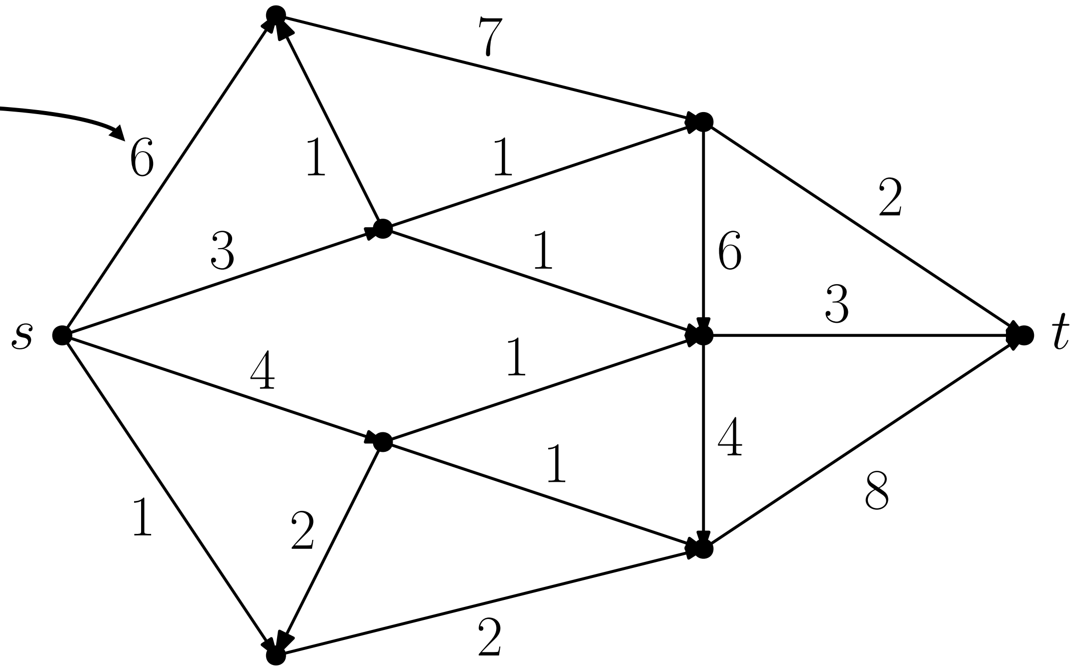
Maximale Flüsse

Gütertransport

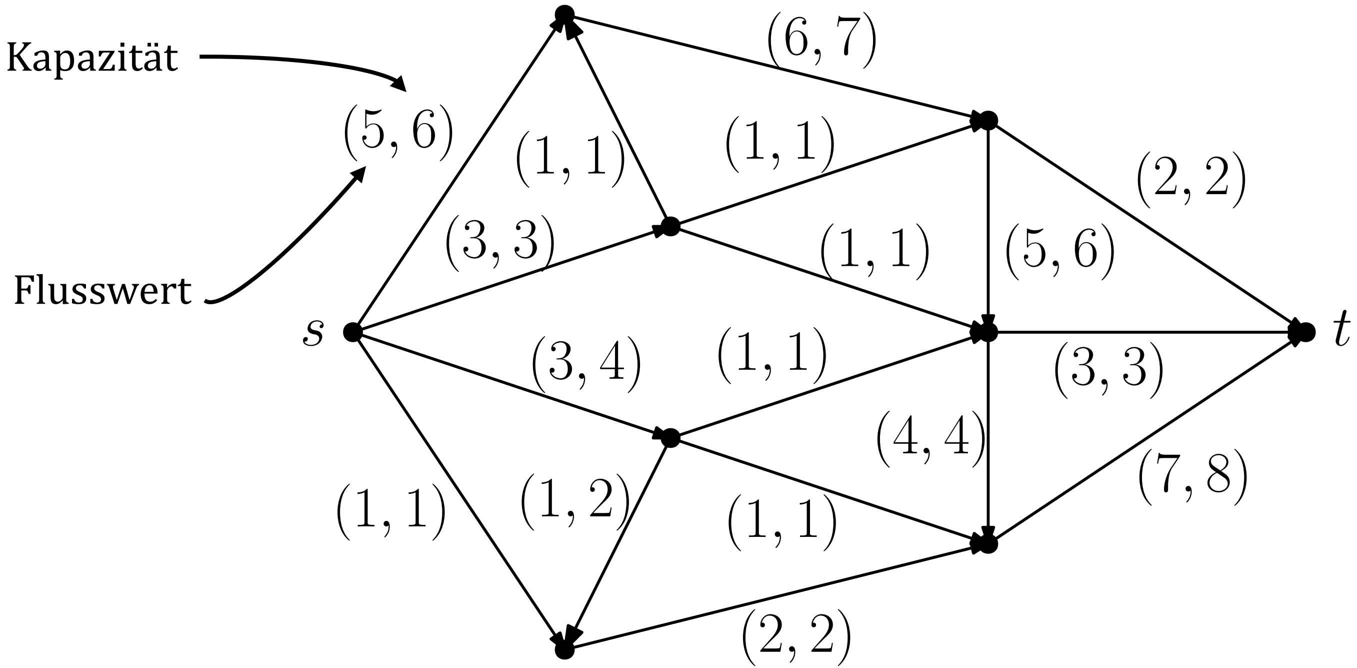
Verkehrsfluss

Netzwerke

Kapazität



Netzwerke


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

Netzwerke

Definition 4.1
Sei $D = (V, A)$ ein Digraph und $u: A \to \mathbb{R}^+$ die Kapazität der Kanten.

(1) Ein **Fluss** ist eine Funktion $f: A \to \mathbb{R}^+$ mit $f(e) \leq u(e)$ für alle $e \in A$.

(2) An einem Knoten $v \in V$ gilt **Flusserhaltung**, wenn folgendes gilt.
$$\sum_{e \in E^+(v)} f(e) = \sum_{e \in E^-(v)} f(e)$$
Dabei ist $E^+(v)$ (bzw $E^-(v)$) die Menge an Kanten, die von v heraus führen (bzw. nach v hinein führen).

(3) Eine **Zirkulation** ist ein Fluss, für den an jedem Knoten Flusserhaltung gilt.

(4) Für ein **Netzwerk** $N = (D, u, s, t)$ ist ein Fluss ein ***st*-Fluss**, wenn Flusserhaltung an allen Knoten außer an s und t gilt.

(5) Für ein st-Fluss ist s die **Quelle** (engl. source) und t die **Senke** (engl. sink).

(6) Der Wert eines st-Flusses f ist definiert als
$$\text{Wert}(f) := \sum_{e \in E^+(s)} f(e) - \sum_{e \in E^-(s)} f(e)$$

Maximum Flow

Problem 4.2: Maximaler Fluss (Maximum Flow)
Gegeben:
Netzwerk $N = (D, u, s, t)$
Gesucht:
Ein st-Fluss f mit Wert(f) maximal.

Beschreibbar als lineares Optimierungsproblem:
Max F
s.t.

$$\sum_{e \in E^+(s)} f_e - \sum_{e \in E^-(s)} f_e = F$$

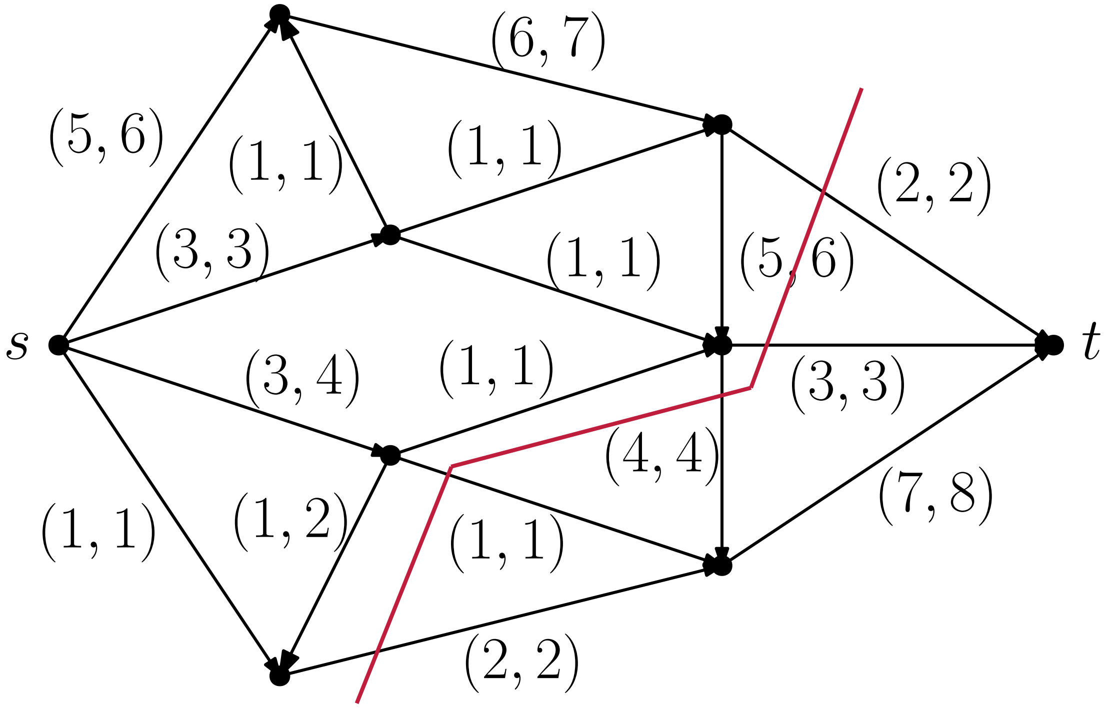
$$\sum_{e \in E^+(v)} f_e - \sum_{e \in E^-(v)} f_e = 0, \qquad \forall v \in V \setminus \{s, t\}$$

$$0 \leq f_e \leq u(e), \qquad \forall e \in A$$

Algorithmen:
Simplex
Ellipsoidmethode
Innere Punkteverfahren
...

4.1 Eigenschaften von Flüssen

Gerichtete Schnitte



Gerichtete Schnitte

Definition 4.3
Sei $D = (V, A)$ ein Digraph und $X \subseteq V$. Dann ist der gerichtete Schnitt

- $E^+(X) := \{(u, v) \in A \mid u \in X, v \notin X\}$
- $E^-(X) := \{(v, u) \in A \mid u \in X, v \notin X\}$

Wir nennen $E^+(X)$ einen ***st*-Schnitt** für zwei Knoten $s, t \in V$, wenn $s \in X$ und $t \notin X$.

Lemma 4.4
Sei $D = (V, A)$ ein Digraph, $X \subsetneq V$ mit $s \in X$ und $t \notin X$. Dann gilt in einem Netzwerk $N = (D, u, s, t)$ für jeden st-Fluss f:

a) $\text{Wert}(f) = \sum_{e \in E^+(X)} f(e) - \sum_{e \in E^-(X)} f(e)$
b) $\text{Wert}(f) \leq \sum_{e \in E^+(X)} u(e)$

Gerichtete Schnitte

Lemma 4.4
Sei $D = (V, A)$ ein Digraph, $X \subsetneq V$ mit $s \in X$ und $t \notin X$. Dann gilt in einem Netzwerk $N = (D, u, s, t)$ für jeden st-Fluss f:

a) $\text{Wert}(f) = \sum_{e \in E^+(X)} f(e) - \sum_{e \in E^-(X)} f(e)$

b) $\text{Wert}(f) \leq \sum_{e \in E^+(X)} u(e)$

Beweis:
Zu a)
Wegen Flusserhaltung ist für alle $v \in V \setminus \{s, t\}$:

$$\sum_{e \in E^+(v)} f(e) - \sum_{e \in E^-(v)} f(e) = 0$$

Also ist:

$$Wert(f) = \sum_{e \in E^+(s)} f(e) - \sum_{e \in E^-(s)} f(e) + 0$$

$$= \sum_{v \in X} \left(\sum_{e \in E^+(v)} f(e) - \sum_{e \in E^-(v)} f(e) \right)$$

$$= \sum_{e \in E^+(X)} f(e) - \sum_{e \in E^-(X)} f(e)$$

Gerichtete Schnitte

Lemma 4.4
Sei $D = (V, A)$ ein Digraph, $X \subsetneq V$ mit $s \in X$ und $t \notin X$. Dann gilt in einem Netzwerk $N = (D, u, s, t)$ für jeden st-Fluss f:

a) $\text{Wert}(f) = \sum_{e \in E^+(X)} f(e) - \sum_{e \in E^-(X)} f(e)$

b) $\text{Wert}(f) \leq \sum_{e \in E^+(X)} u(e)$

Beweis:

Zu b)

Nach a) gilt:

$$Wert(f) = \sum_{e \in E^+(X)} f(e) - \sum_{e \in E^-(X)} f(e)$$

Also ist:

$$Wert(f) \leq \sum_{e \in E^+(X)} f(e)$$

$$\leq \sum_{e \in E^+(X)} u(e)$$

Algorithmus – Grundidee

Finde Pfade, über die der aktuelle Fluss verbessert werden kann.

Wie kann ein solcher Pfad gefunden werden?

Problem:
Flusswert über Kanten müssen ggf. verringert werden, d.h. wir laufen rückwärts über Kanten.

Lösung:
Hilfsgraph D_f für Fluss f, der angibt, um wie viel ein Fluss für eine Kante erhöht bzw. reduziert werden kann.

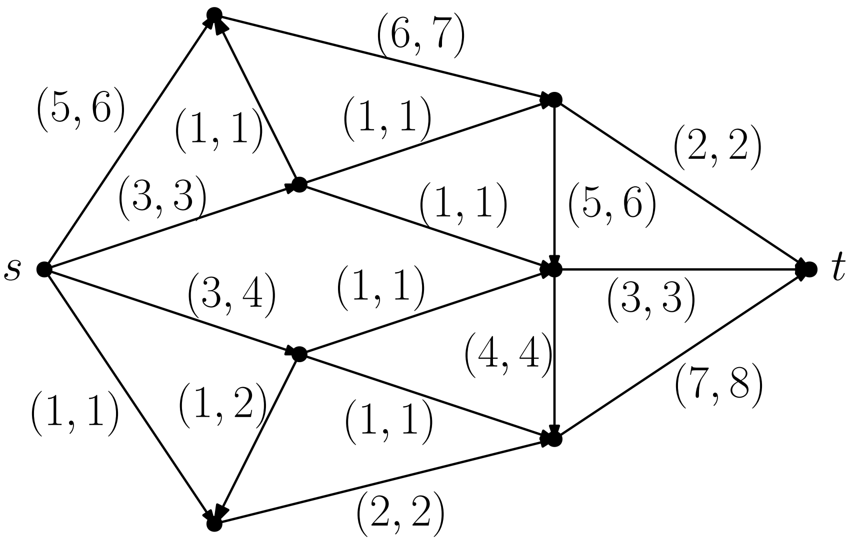
Residualgraph

Definition 4.5
Sei $D = (V, A)$ ein Digraph und $N = (D, u, s, t)$ ein Netzwerk.

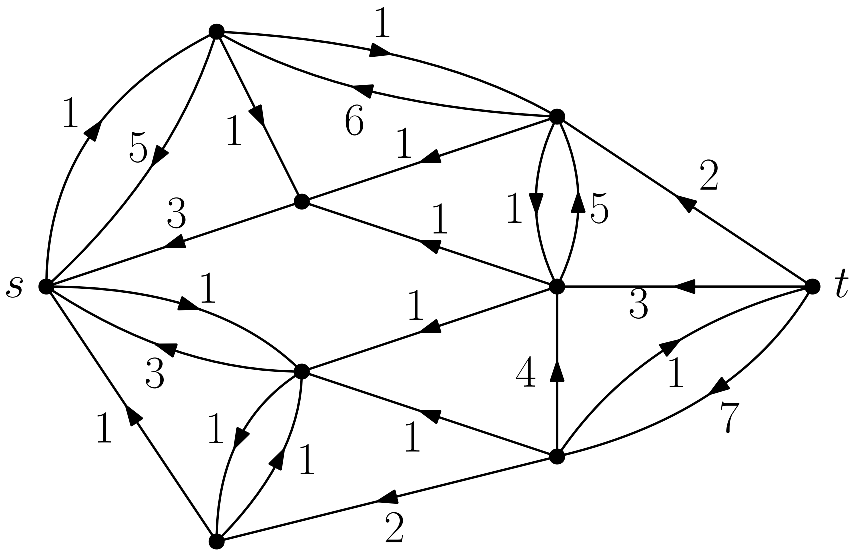
1. $\overleftrightarrow{D} := (V, A')$ mit $A' := A \mathbin{\dot\cup} \{(v, u) \mid (u, v) \in A\}$ der doppelt gerichtete Graph von D.
2. Kanten aus A heißen **Vorwärtskanten**, Kanten aus $\{(v, u) \mid (u, v) \in A\}$ heißen **Rückwärtskanten**
3. Für einen st-Fluss f sind die **Residualkapazitäten**
$$u_f(e) := \begin{cases} u(e) - f(e), & \text{für Vorwärtskanten aus } A' \\ f(e), & \text{für Rückwärtskanten aus } A' \end{cases}$$
4. Damit ist der **Residualgraph** $D_f := \big(V, \{e \in A' \mid u_f(e) > 0\}\big)$.
5. Der Wert $u_f(e)$ gibt an, um wie viel der Fluss über die Kante e maximal erhöht werden kann (bzw. reduziert werden kann, wenn e eine Rückwärtskante ist).
6. Um einen Fluss f entlang eines Pfades (oder Kreises) P um γ zu **augmentieren**, muss man
   - den Fluss für jede Vorwärtskante um γ erhöhen.
   - den Fluss für jede Rückwärtskante um γ reduzieren.
7. Ein ***f*-augmentierender Pfad** in N mit Fluss f ist ein st-Pfad in D_f.

Beispiel

Netzwerk



Residualgraph



Ein Optimalitätskriterium

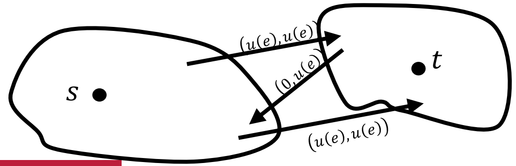
Lemma 4.6
Sei $N = (D, u, s, t)$ ein Netzwerk. Ein st-Fluss f ist genau dann maximal, wenn es keinen st-Pfad in D_f gibt.

„⇒“:
Annahme: es gibt einen st-Pfad P in D_f.
⇒ Augmentiere f über P.
⇒ f war nicht maximal.

Ein Optimalitätskriterium

Lemma 4.6
Sei $N = (D, u, s, t)$ ein Netzwerk. Ein st-Fluss f ist genau dann maximal, wenn es keinen st-Pfad in D_f gibt.

„$\Leftarrow$“:
Sei
$$R := \{v \in V \mid v \text{ von } s \text{ in } D_f \text{ erreichbar}\}$$
Da $t \notin R$ ist $E^+(R) = \emptyset$ in D_f.
Also gilt:
1. Für alle $e \in E^+(R)$ in D: $f(e) = u(e)$
2. Für alle $e \in E^-(R)$ in D: $f(e) = 0$



Nach Lemma 4.4 a:
$$\text{Wert}(f) = \sum_{e \in E^+(R)} f(e) = \sum_{e \in E^+(R)} u(e)$$
Nach Lemma 4.4 b:
$$\text{Wert}(f) \leq \sum_{e \in E^+(R)} u(e)$$
Also ist f maximal.

Ein Optimalitätskriterium

Lemma 4.6
Sei $N = (D, u, s, t)$ ein Netzwerk. Ein st-Fluss f ist genau dann maximal, wenn es keinen st-Pfad in D_f gibt.

Mit dem Beweis von Lemma 4.6 folgt direkt:

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.

Das motiviert folgenden Algorithmus.

4.2 Berechnung von maximalen Flüssen

Algorithmus von Ford-Fulkerson

Algorithmus 4.8

Eingabe:

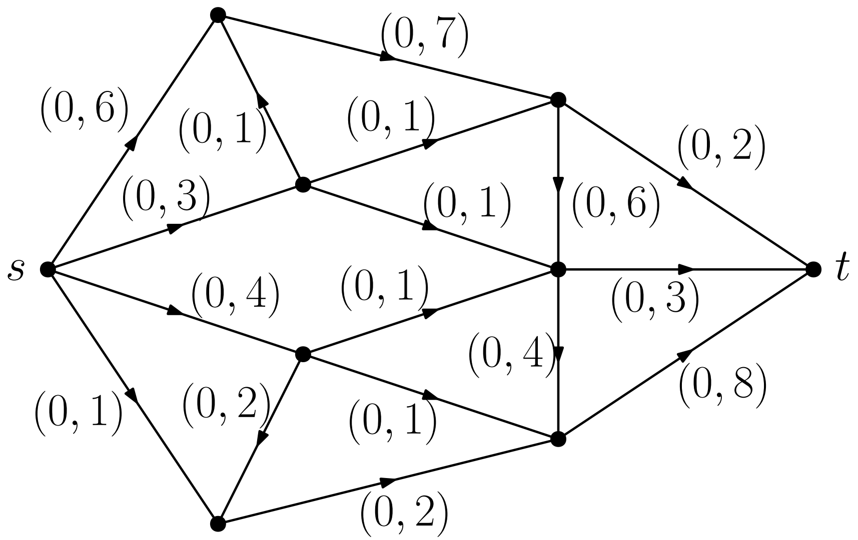
Netzwerk $N = (D, u, s, t)$

Ausgabe:

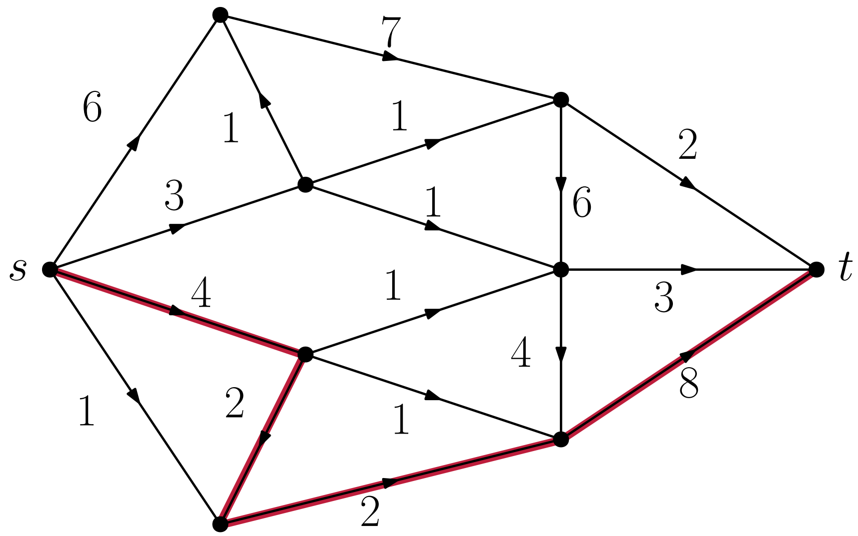
st-Fluss f mit maximalem Wert

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

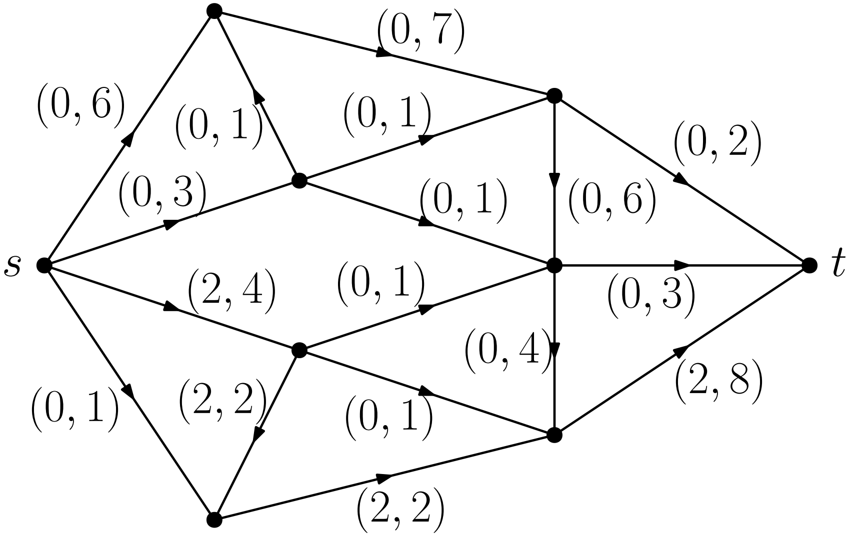
Beispiel

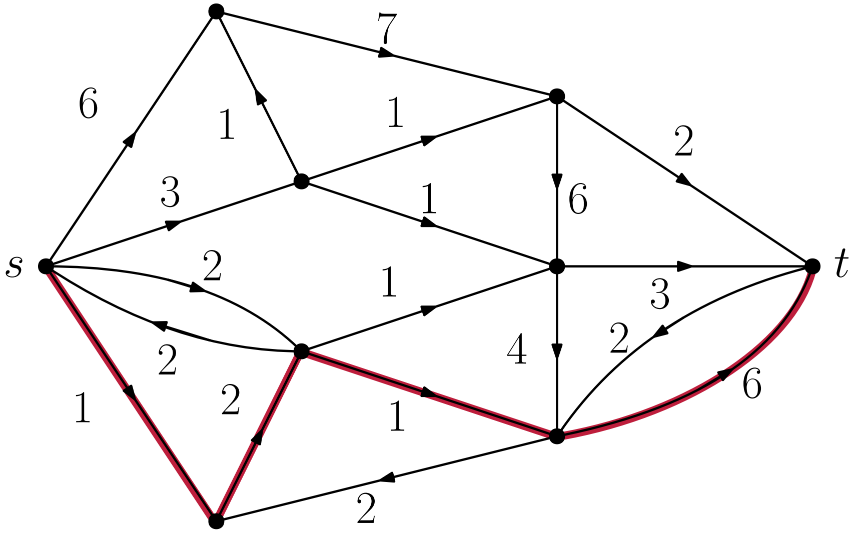


(0, 7)
(0, 6)
(0, 1)
(0, 1)
(0, 2)
(0, 3)
(0, 1)
(0, 6)
s
t
(0, 4)
(0, 1)
(0, 3)
(0, 4)
(0, 8)
(0, 1)
(0, 2)
(0, 1)
(0, 2)

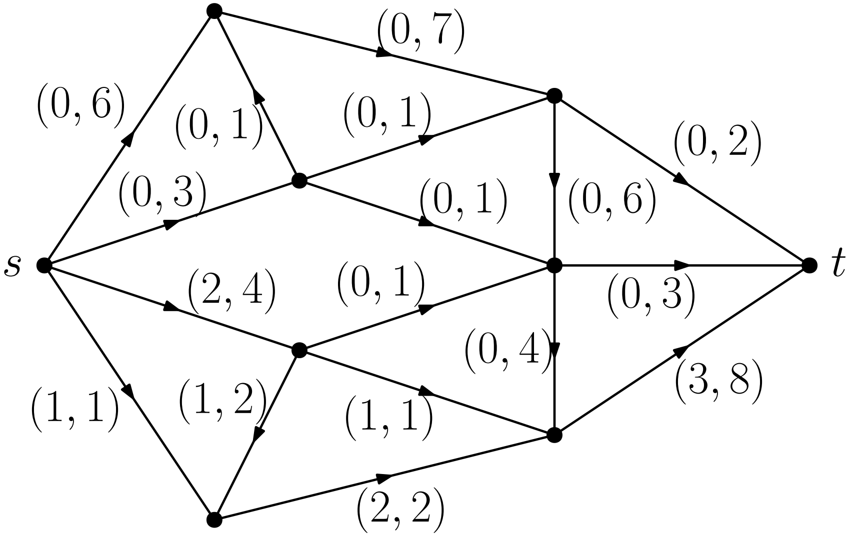


7
6
1
1
2
3
1
6
s
t
4
1
3
4
8
1
2
1
2

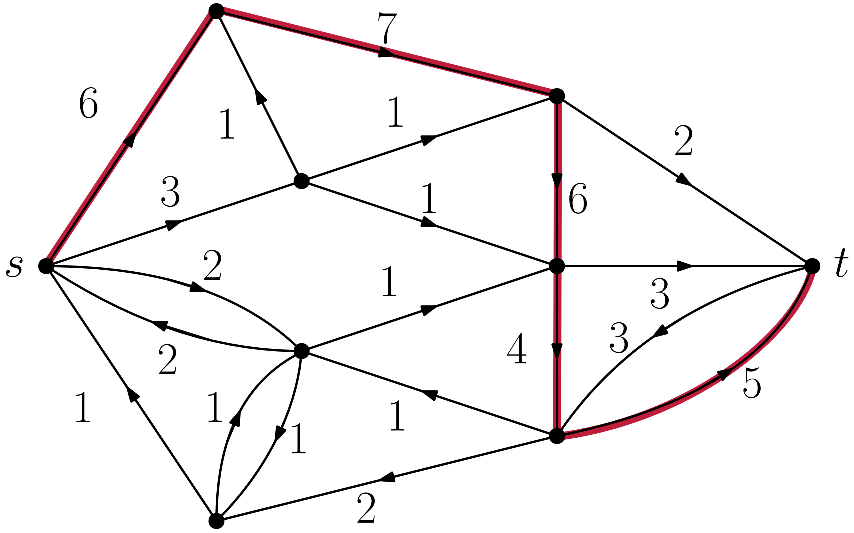

(0, 6)
(0, 7)
(0, 1)
(0, 1)
(0, 2)
(0, 3)
(0, 1)
(0, 6)
s
t
(2, 4)
(0, 1)
(0, 3)
(0, 4)
(0, 1)
(2, 2)
(0, 1)
(2, 8)
(2, 2)


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

Beispiel

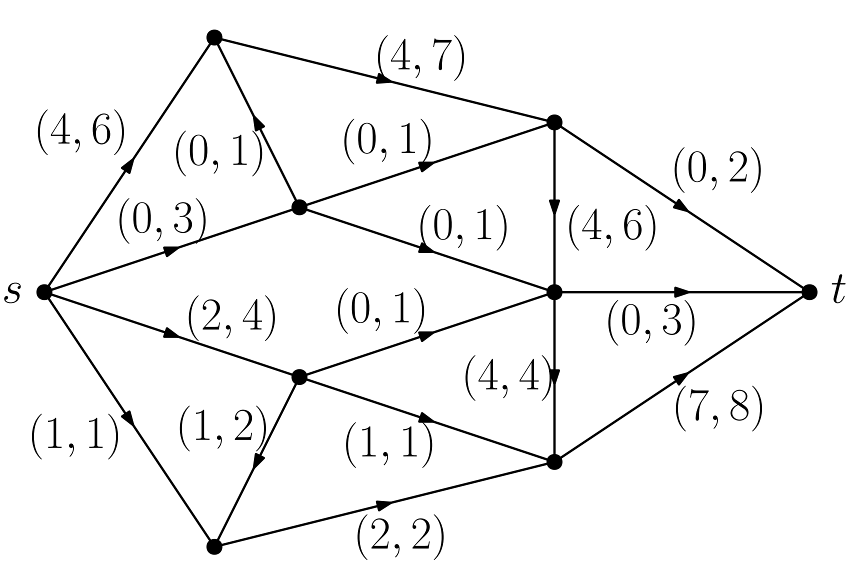


(0, 7)
(0, 6)
(0, 1)
(0, 1)
(0, 2)
(0, 3)
(0, 1)
(0, 6)
s
t
(2, 4)
(0, 1)
(0, 3)
(0, 4)
(1, 1)
(1, 2)
(1, 1)
(3, 8)
(2, 2)

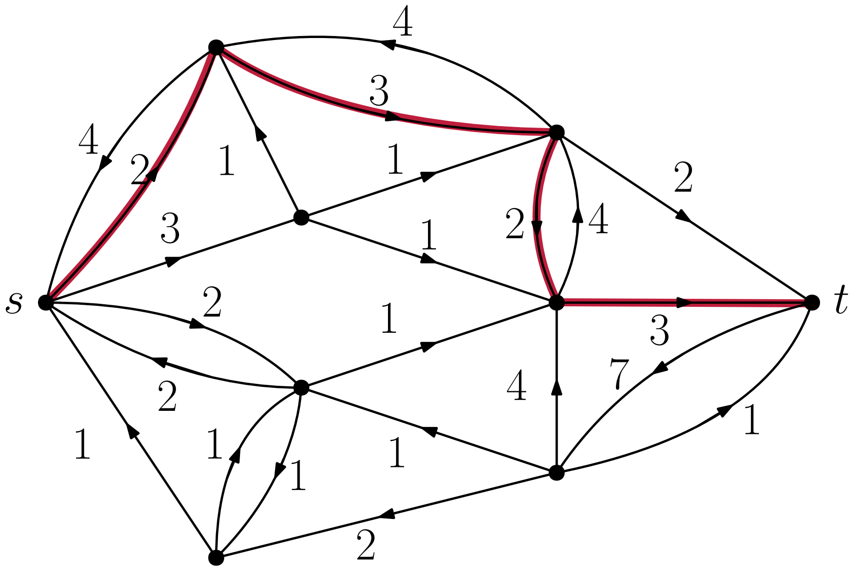


7
6
1
1
2
3
1
6
s
2
1
t
3
2
4
3
2
1
1
1
1
5
1
2

Beispiel

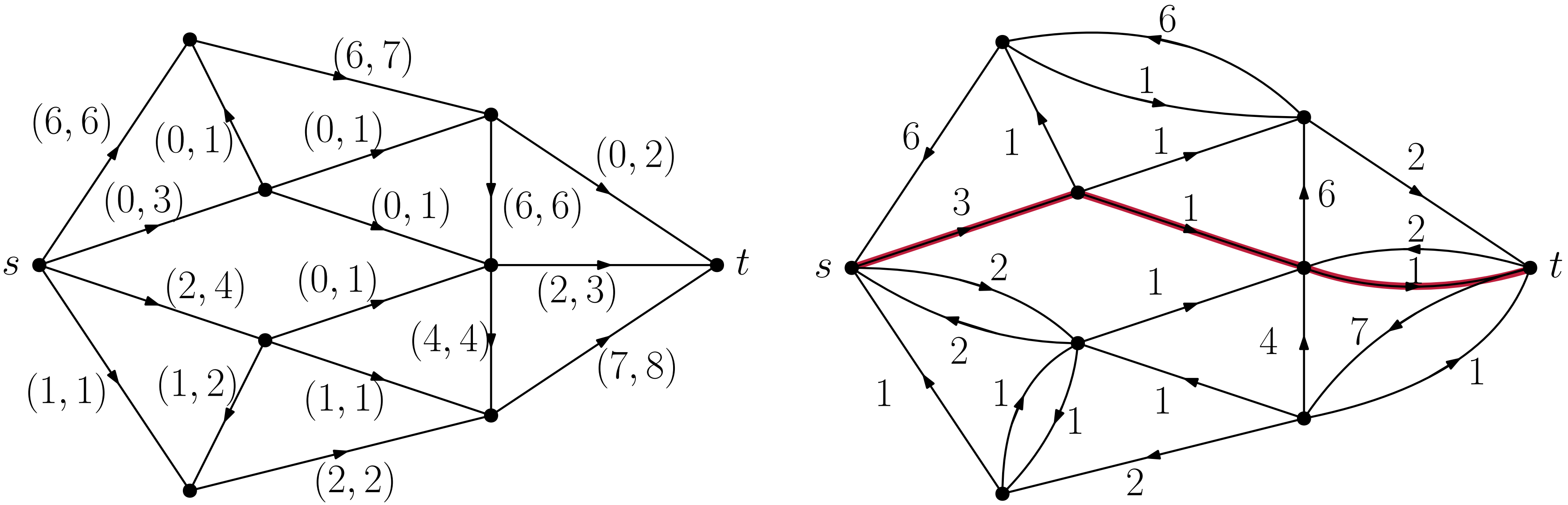


(4, 7)
(4, 6)
(0, 1)
(0, 1)
(0, 2)
(0, 3)
(0, 1)
(4, 6)
s
t
(2, 4)
(0, 1)
(0, 3)
(4, 4)
(1, 1)
(1, 2)
(1, 1)
(7, 8)
(2, 2)



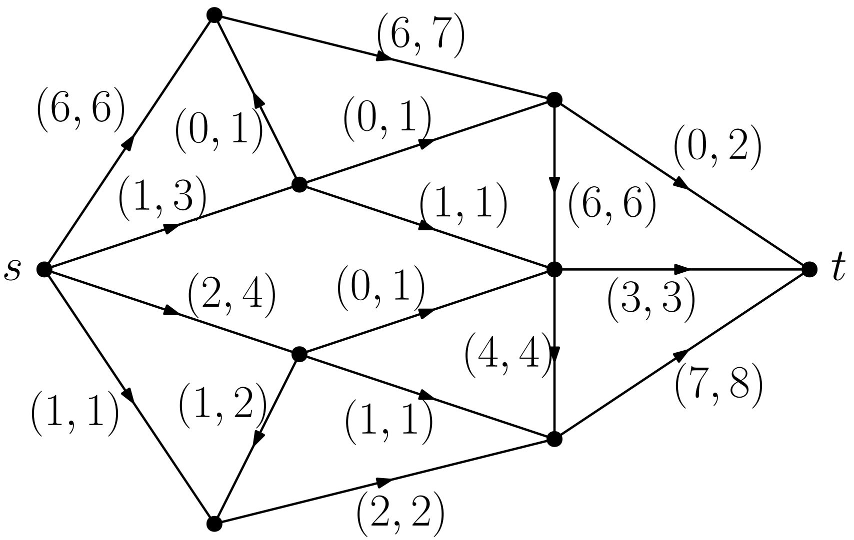
4
3
4
2
1
1
2
3
1
2
4
s
t
2
1
3
2
7
4
1
1
1
1
1
2

Beispiel

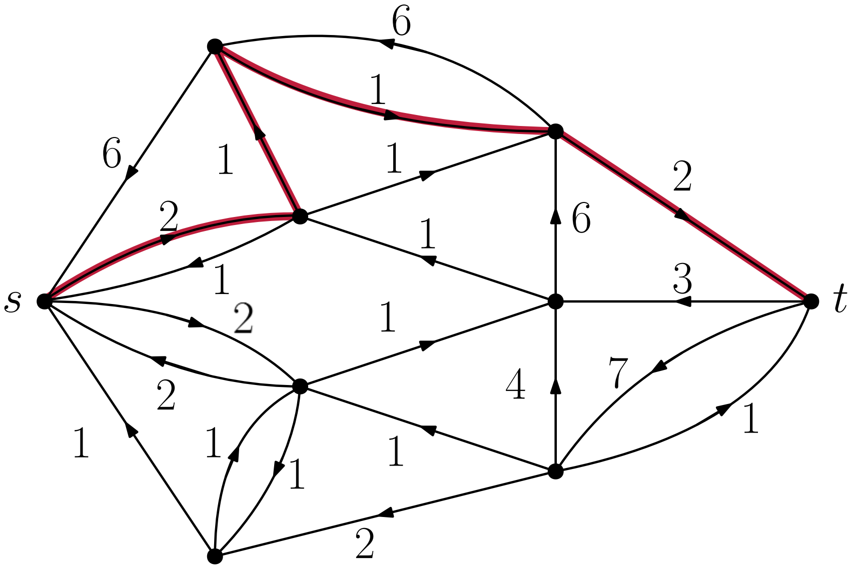


(6, 7)
(6, 6)
(0, 1)
(0, 1)
(0, 2)
(0, 3)
(0, 1)
(6, 6)
s
t
(2, 4)
(0, 1)
(2, 3)
(4, 4)
(1, 1)
(1, 2)
(1, 1)
(7, 8)
(2, 2)
6
1
6
1
1
2
3
1
6
2
s
1
t
2
1
2
7
4
1
1
1
1
1
2

Beispiel

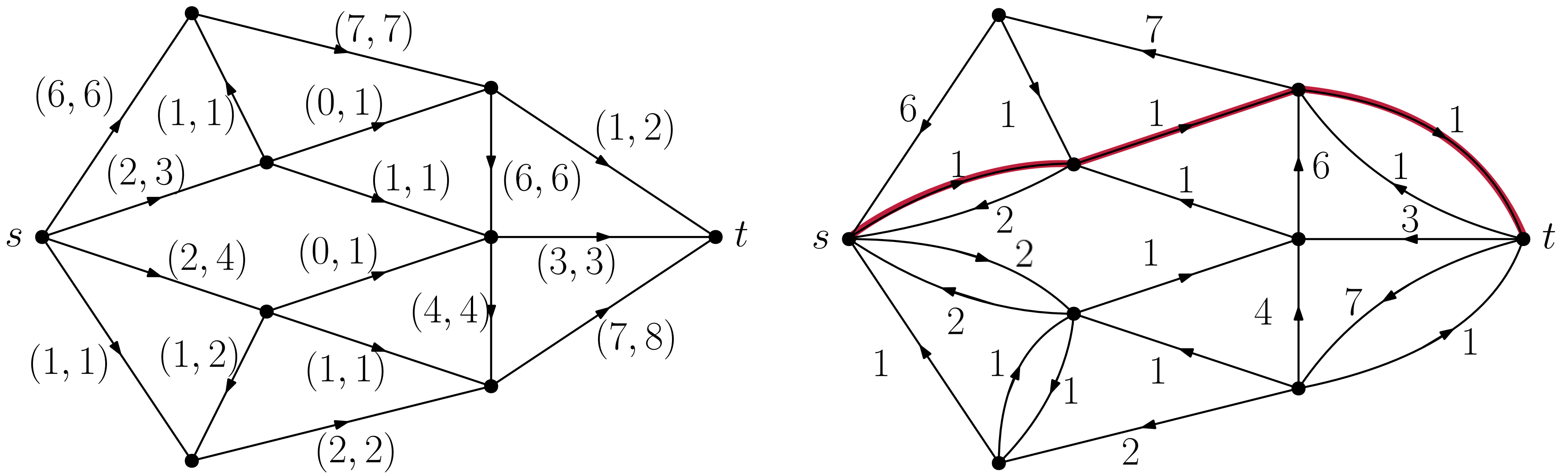


(6, 7)
(6, 6)
(0, 1)
(0, 1)
(0, 2)
(1, 3)
(1, 1)
(6, 6)
s
t
(2, 4)
(0, 1)
(3, 3)
(4, 4)
(1, 1)
(1, 2)
(1, 1)
(7, 8)
(2, 2)

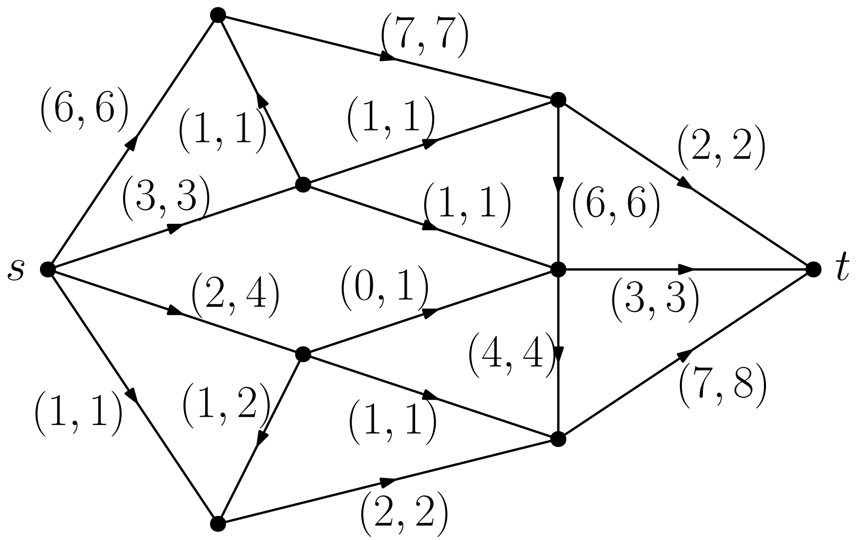


6
1
6
1
1
2
2
6
1
1
3
s
t
2
1
2
4
7
1
1
1
1
1
2

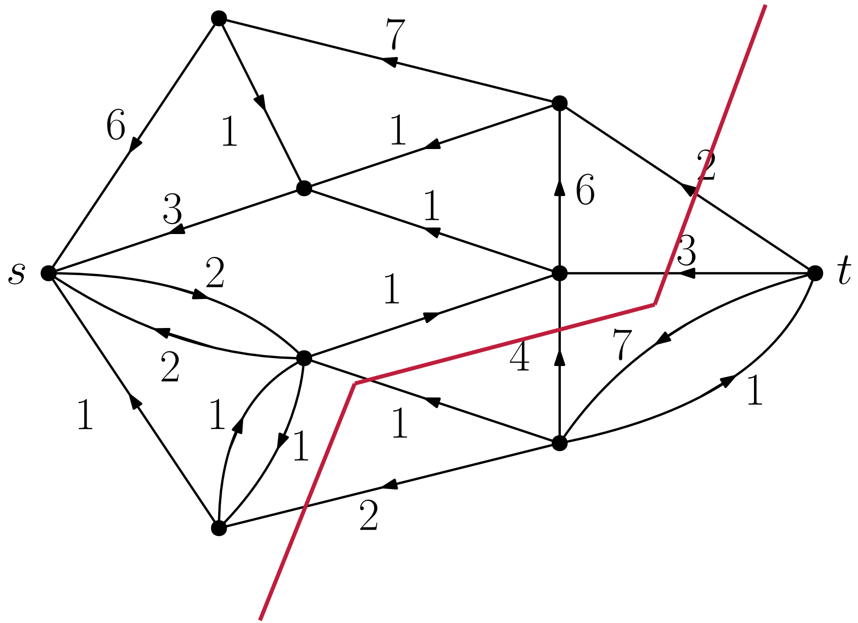
Beispiel


(7, 7)
(6, 6)
(1, 1)
(0, 1)
(1, 2)
(2, 3)
(1, 1)
(6, 6)
s
t
(2, 4)
(0, 1)
(3, 3)
(4, 4)
(7, 8)
(1, 1)
(1, 2)
(1, 1)
(2, 2)
7
6
1
1
1
1
1
6
1
2
3
2
1
2
4
7
1
1
1
1
1
2

Beispiel



(6, 6)
(1, 1)
(7, 7)
(1, 1)
(3, 3)
(1, 1)
(6, 6)
(2, 2)
s
t
(2, 4)
(0, 1)
(3, 3)
(4, 4)
(1, 1)
(1, 2)
(1, 1)
(7, 8)
(2, 2)



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

Nächste Woche – Zum Überlegen!

Ist der Algorithmus korrekt?

Wie lange braucht der Algorithmus im Worst-Case?

Was passiert bei…
- Ganzzahligen Kapazitäten?
- Rationalen Kapazitäten?
- Irrationalen Kapazitäten?