Algorithmen II
Übung am 12.11.2013

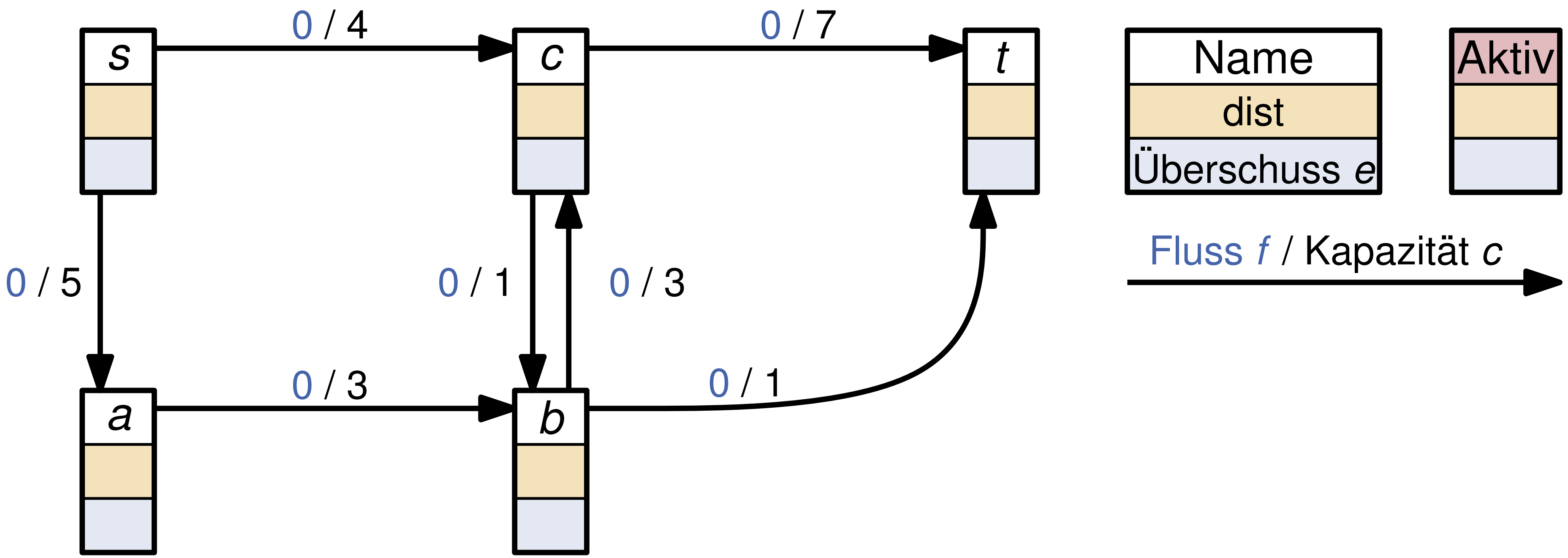
Flüsse und Schnitte

Institut für Theoretische Informatik · Prof. Dr. Dorothea Wagner

Algorithmus von Goldberg & Tarjan

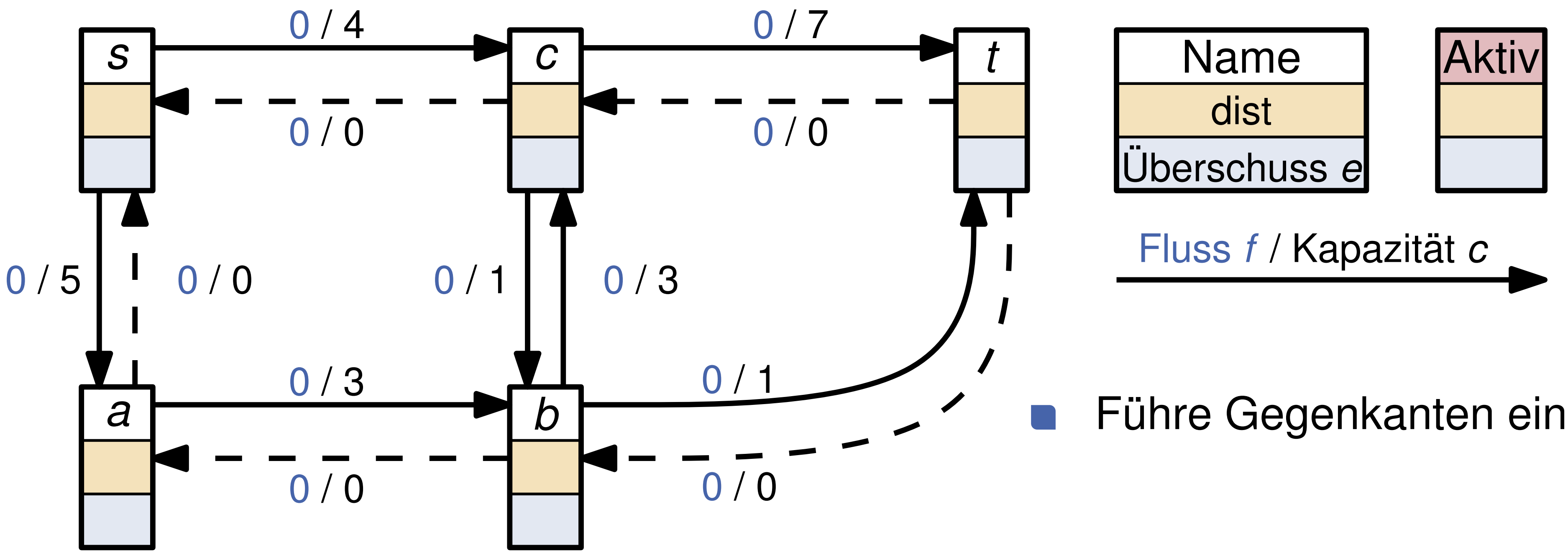
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ein Beispiel



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

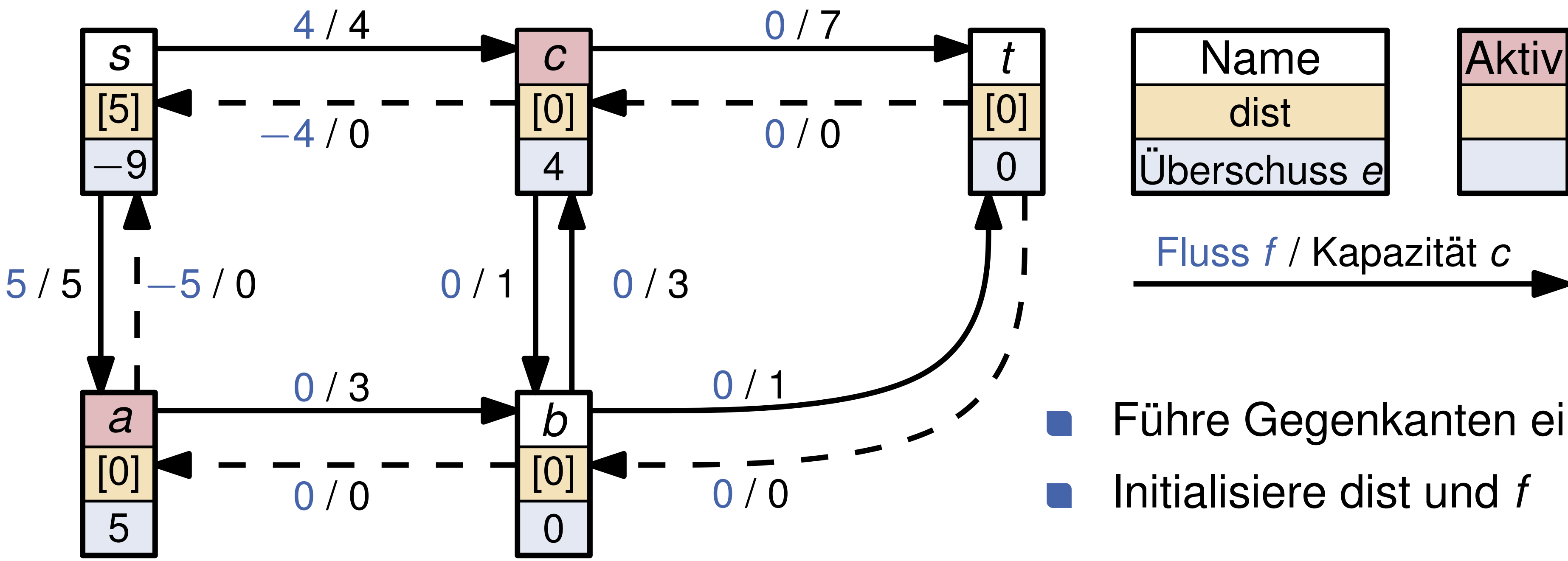
Ein Beispiel



- Führe Gegenkanten ein

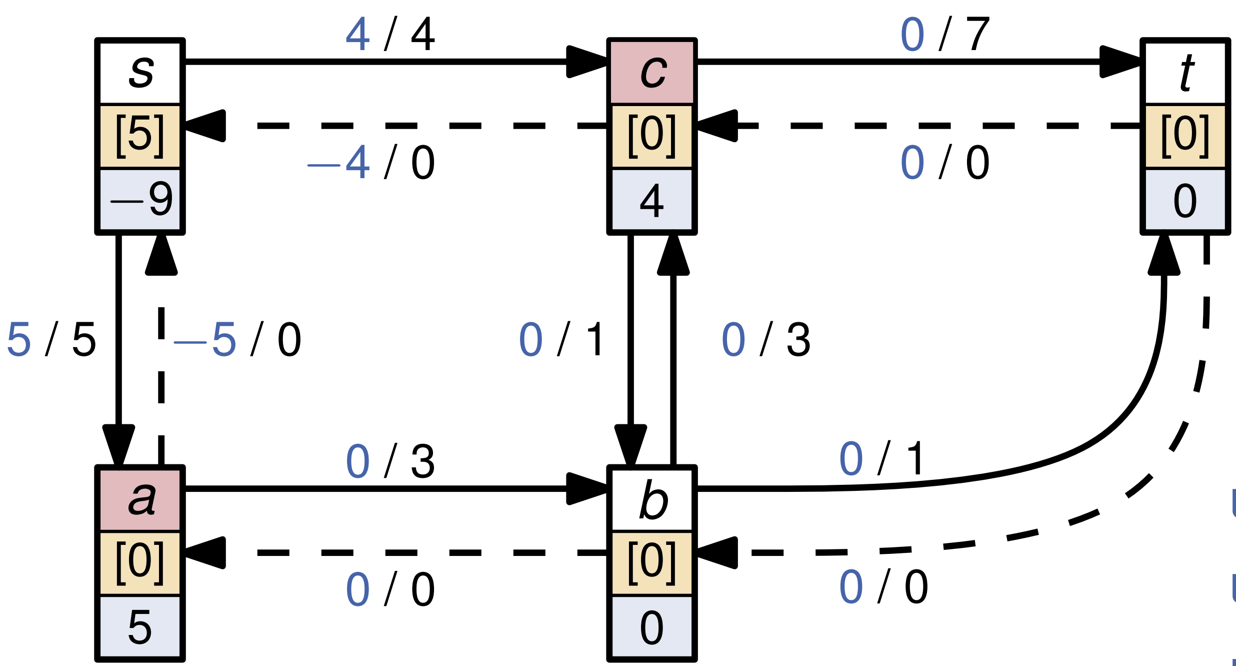
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

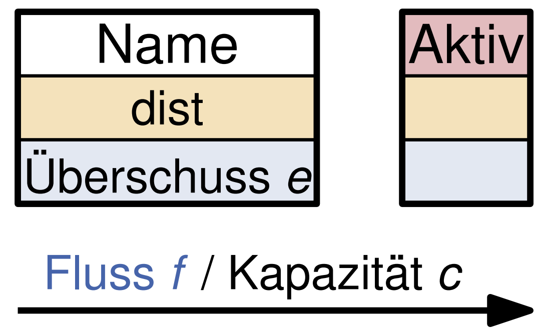
Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f

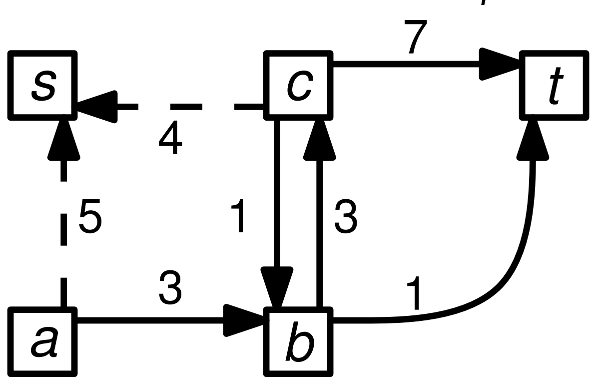
Ein Beispiel





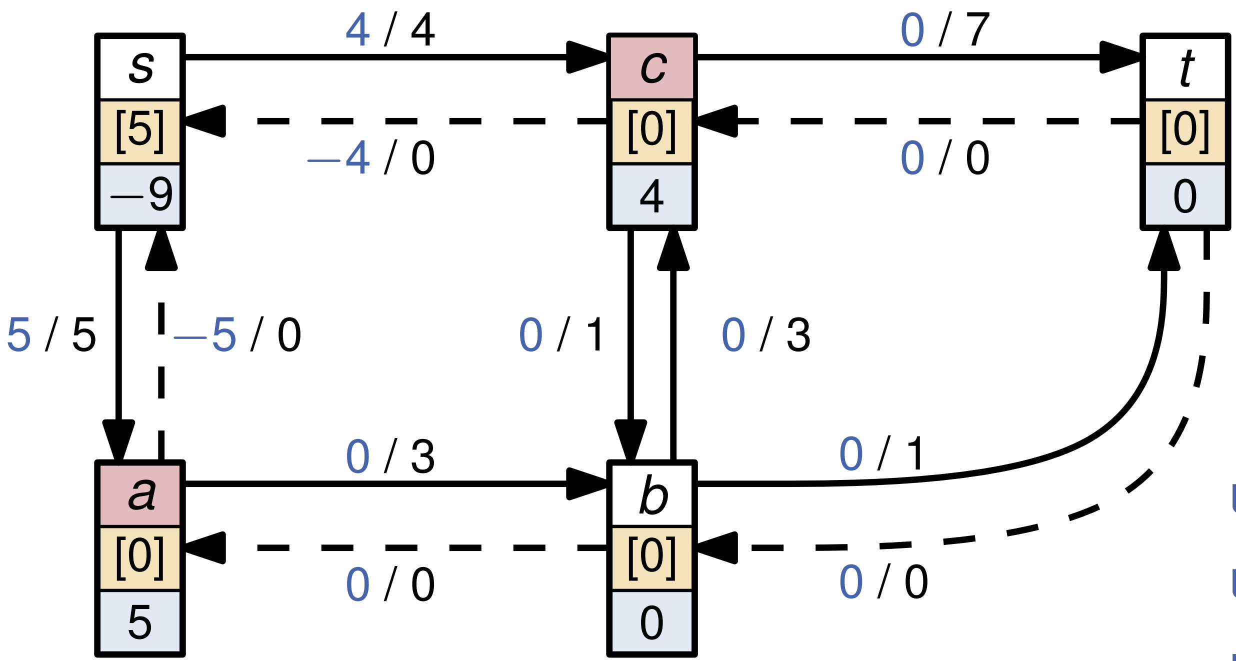
- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

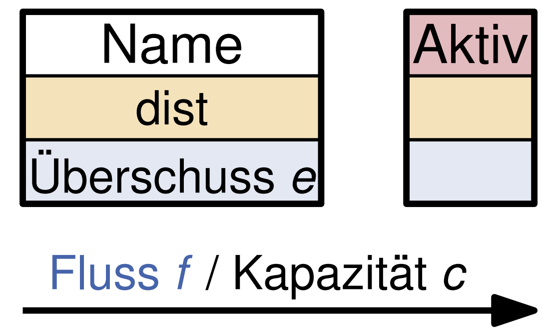
Residualnetzwerk D_f:



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ein Beispiel

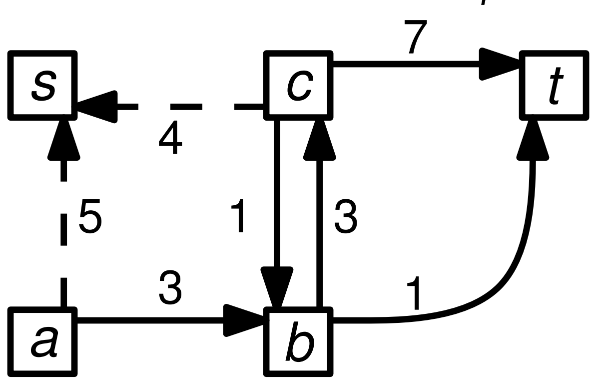




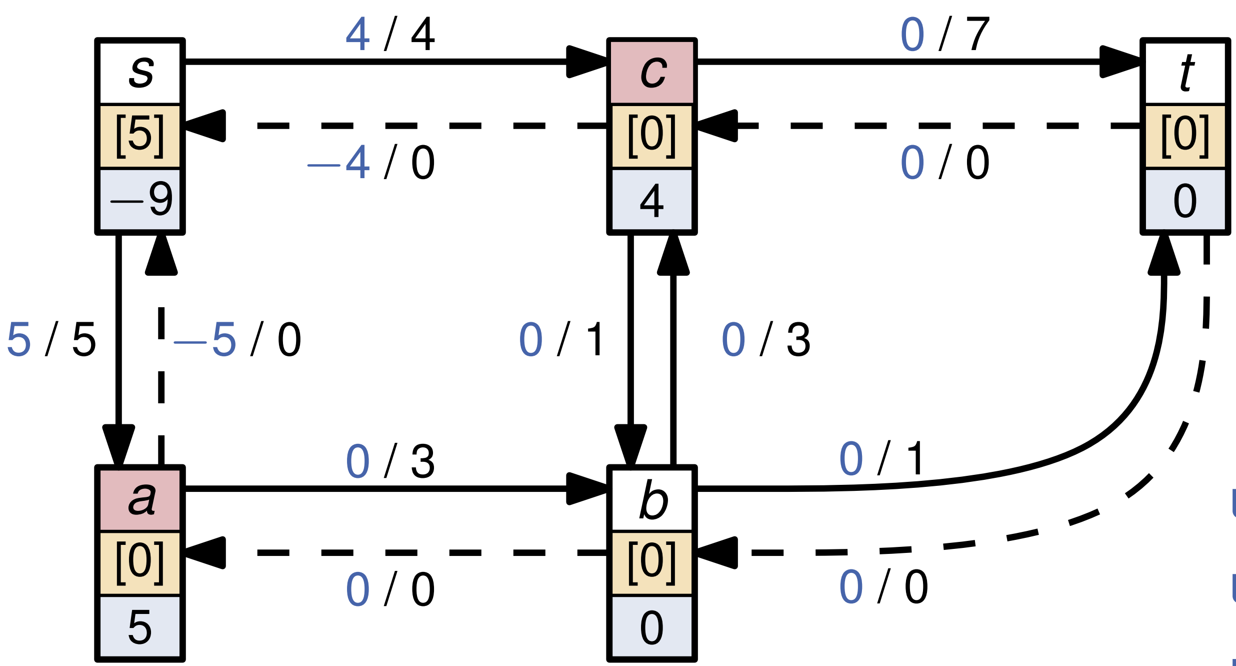
- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

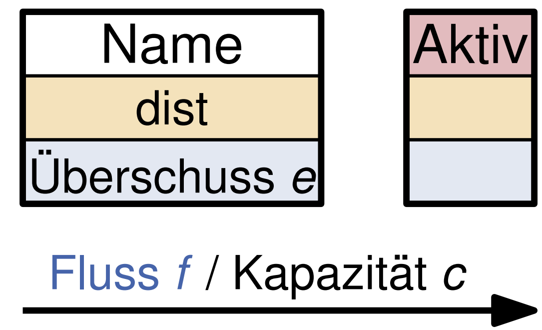
Wähle aktiven Knoten und führe PUSH / RELABEL aus:

Residualnetzwerk D_f:



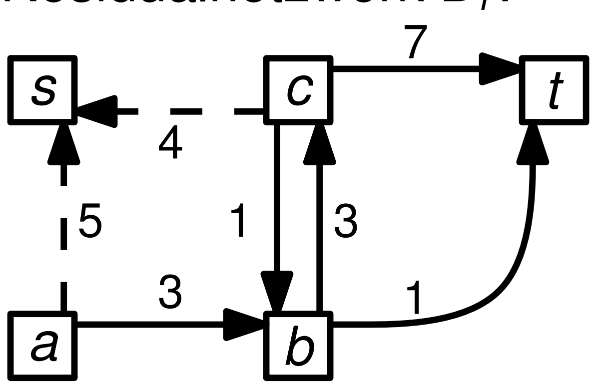
Ein Beispiel





- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Residualnetzwerk D_f:



Wähle aktiven Knoten und führe PUSH / RELABEL aus:

Bedingungen zum Ausführen von RELABEL für a:

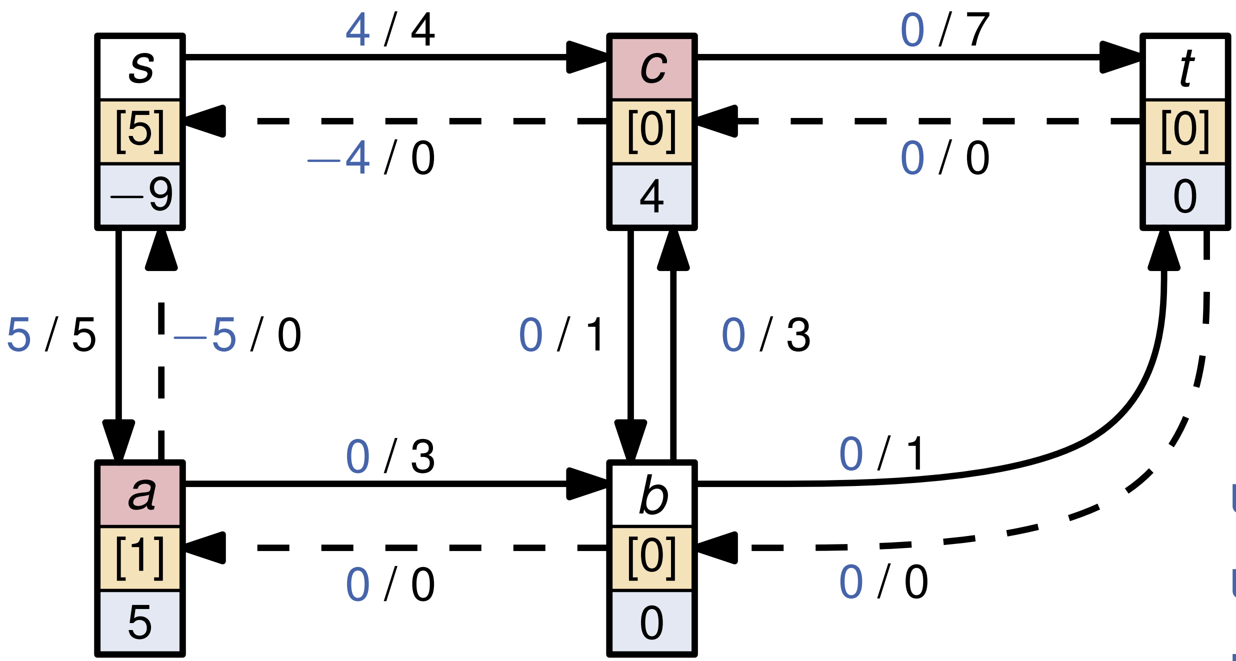
- Für alle Kanten (a, v) in D_f gilt: $\mathrm{dist}(a) \leq \mathrm{dist}(v)$

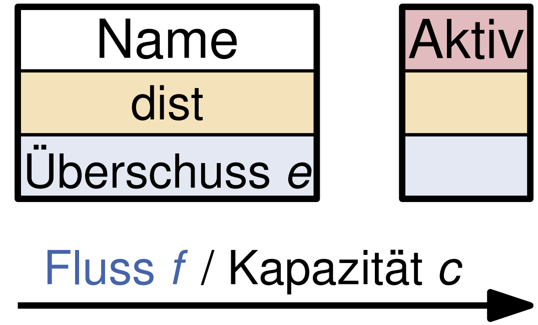
Setze Label $\mathrm{dist}(a)$ auf...

- ∞, falls es in D_f keine Kante (a, v) gibt.
- Minimum von $\mathrm{dist}(v) + 1$ über alle Kanten (a, v) in D_f.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

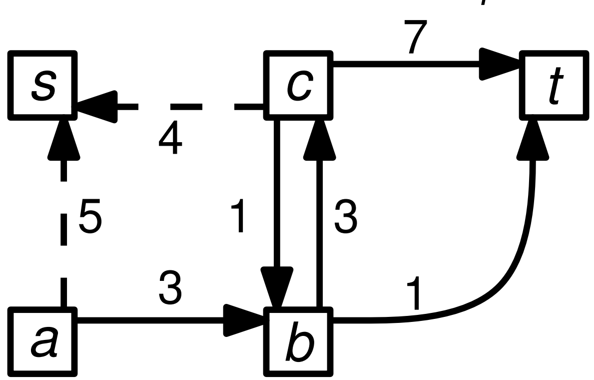
Ein Beispiel





- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Residualnetzwerk D_f:



Wähle aktiven Knoten und führe PUSH / RELABEL aus:

Bedingungen zum Ausführen von RELABEL für a:

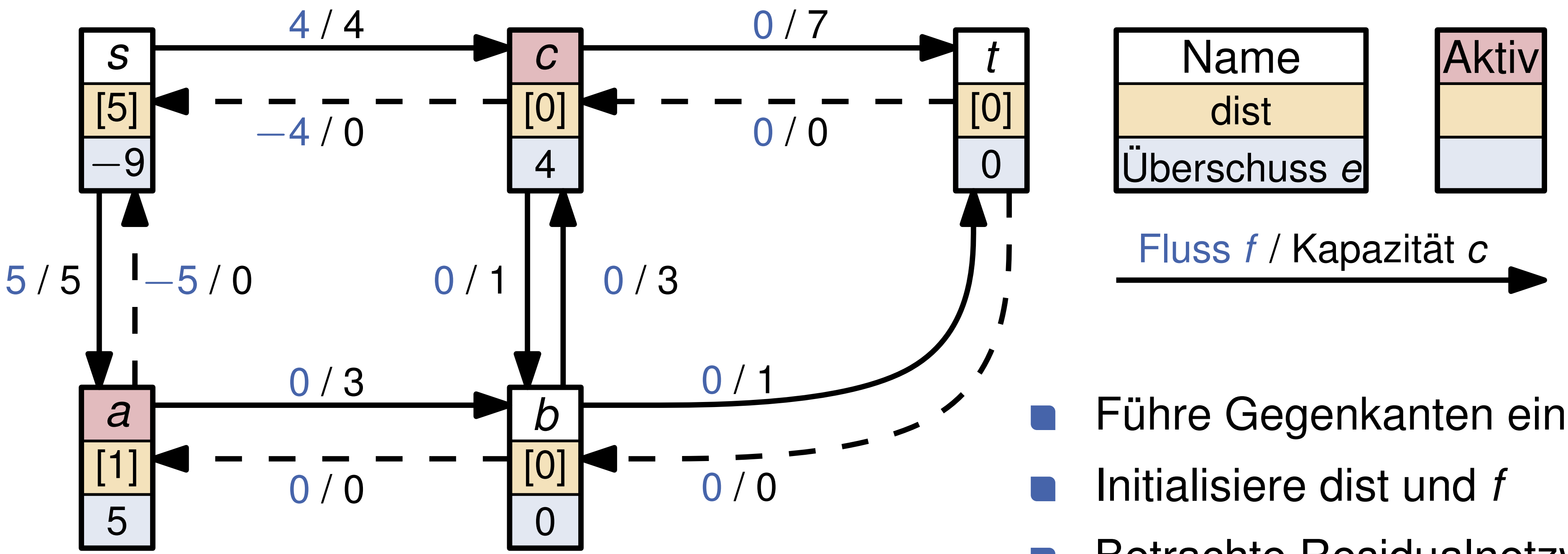
- Für alle Kanten (a, v) in D_f gilt: $\text{dist}(a) \leq \text{dist}(v)$

Setze Label $\text{dist}(a)$ auf...

- ∞, falls es in D_f keine Kante (a, v) gibt.
- Minimum von $\text{dist}(v) + 1$ über alle Kanten (a, v) in D_f.

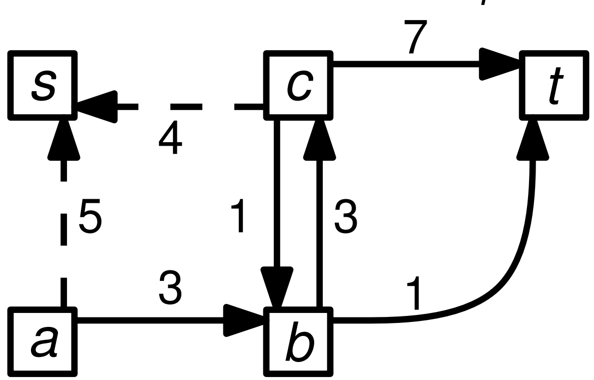
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Residualnetzwerk D_f:



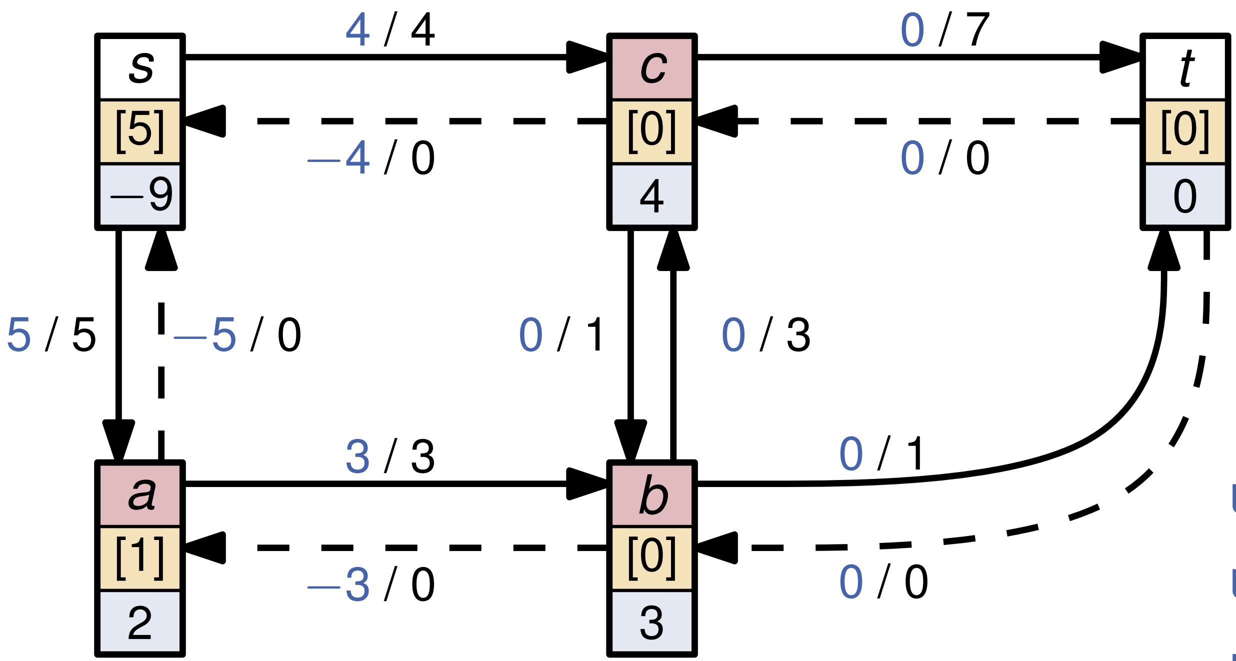
Wähle aktiven Knoten und führe PUSH / RELABEL aus:

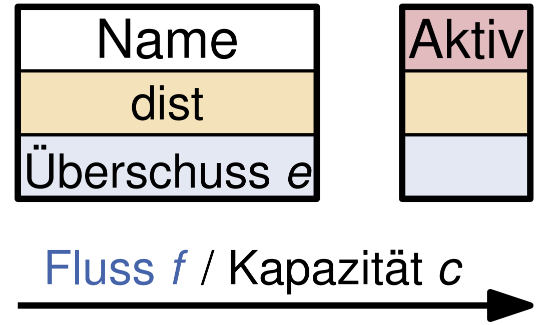
Bedingungen zum Ausführen von PUSH für a:

- D_f enthält Kante (a, v) mit $\text{dist}(a) = \text{dist}(v) + 1$

Erhöhe Fluss auf dieser Kante (a, v) möglichst stark.

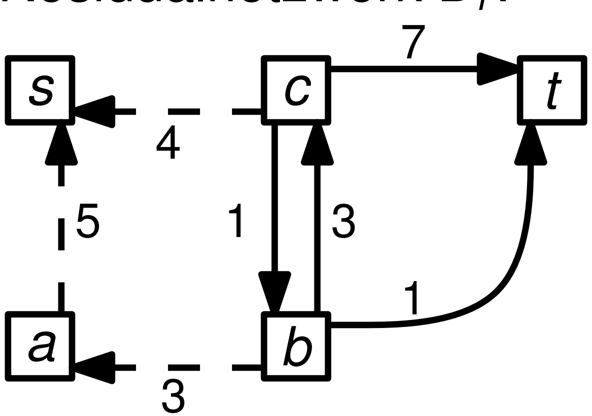
Ein Beispiel





- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Residualnetzwerk D_f:



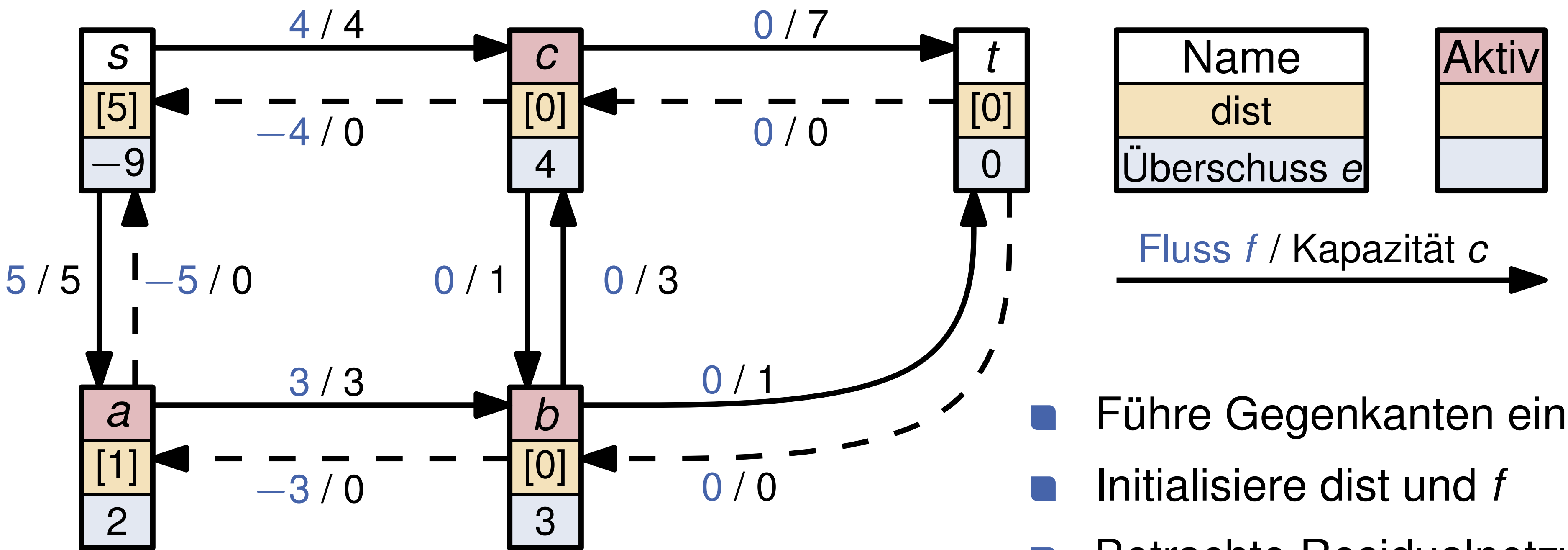
Wähle aktiven Knoten und führe PUSH / RELABEL aus:

Bedingungen zum Ausführen von PUSH für a:

- D_f enthält Kante (a, v) mit $\text{dist}(a) = \text{dist}(v) + 1$

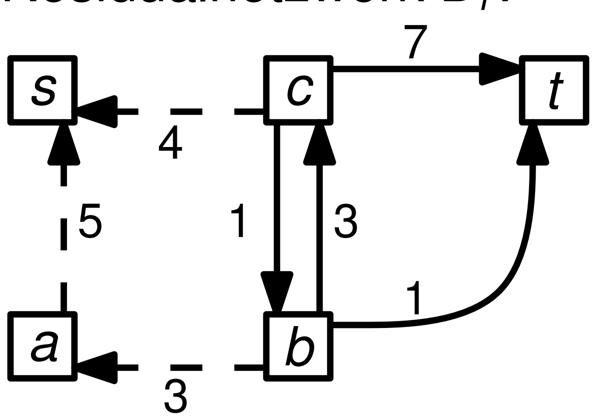
Erhöhe Fluss auf dieser Kante (a, v) möglichst stark.

Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Residualnetzwerk D_f:

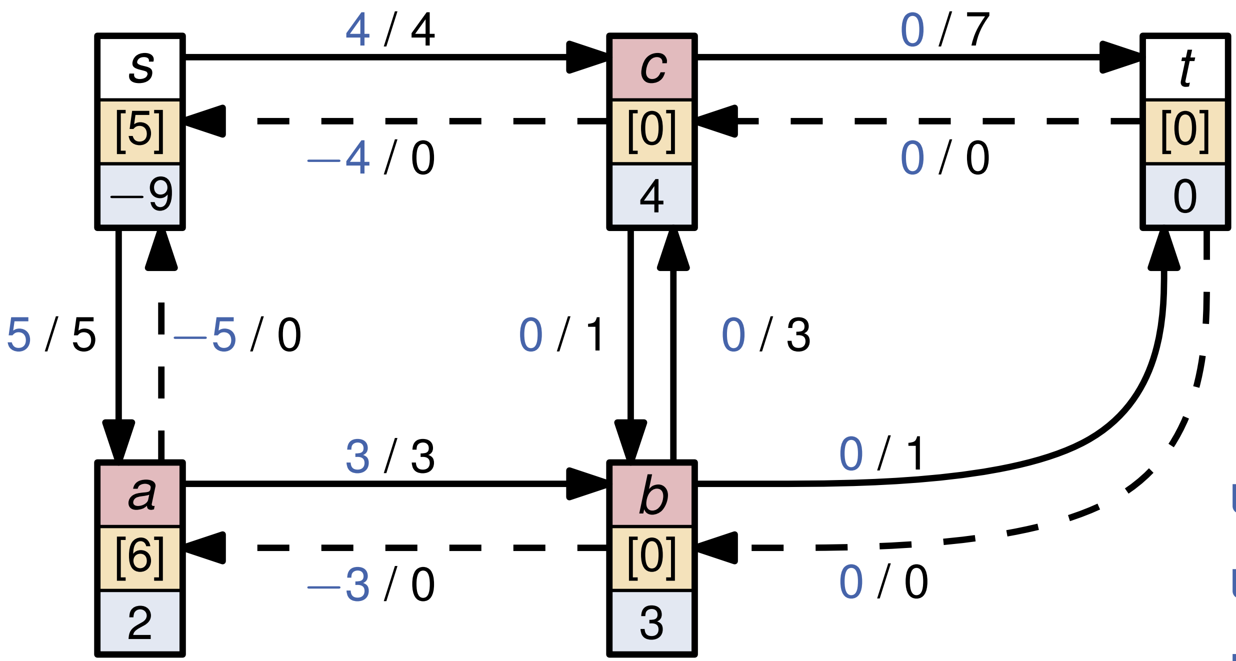


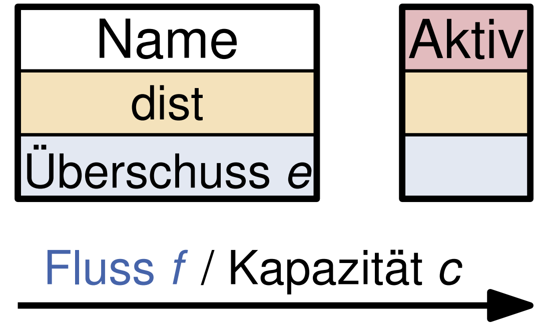
Wähle aktiven Knoten und führe PUSH / RELABEL aus:

RELABEL(a)

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ein Beispiel

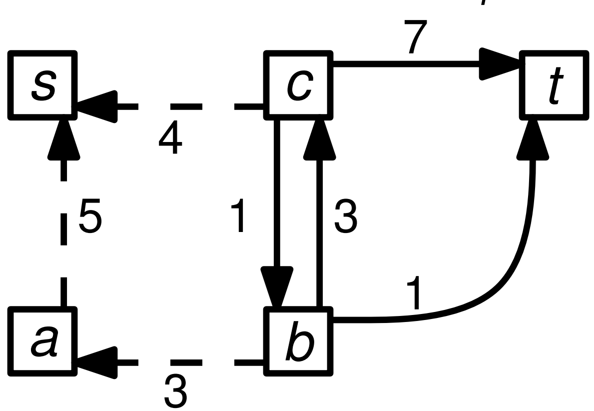




- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Wähle aktiven Knoten und führe Push / Relabel aus:

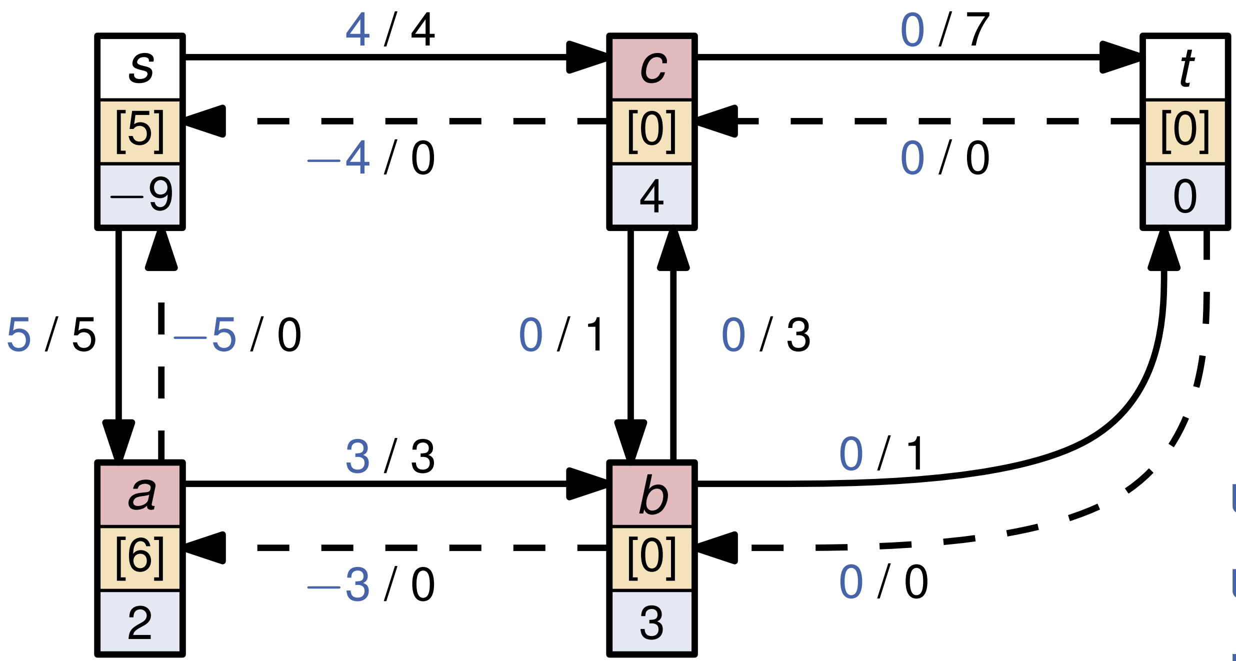
Residualnetzwerk D_f:



RELABEL(a)

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

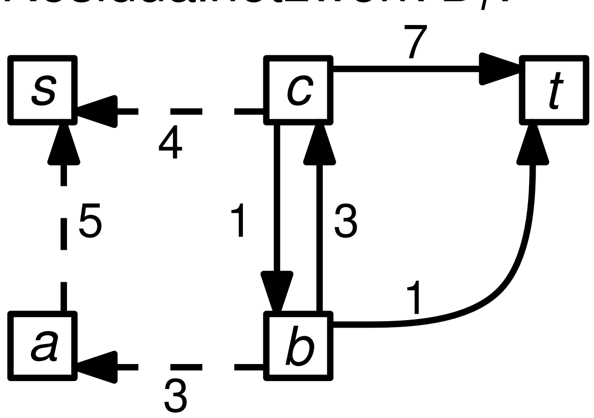
Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

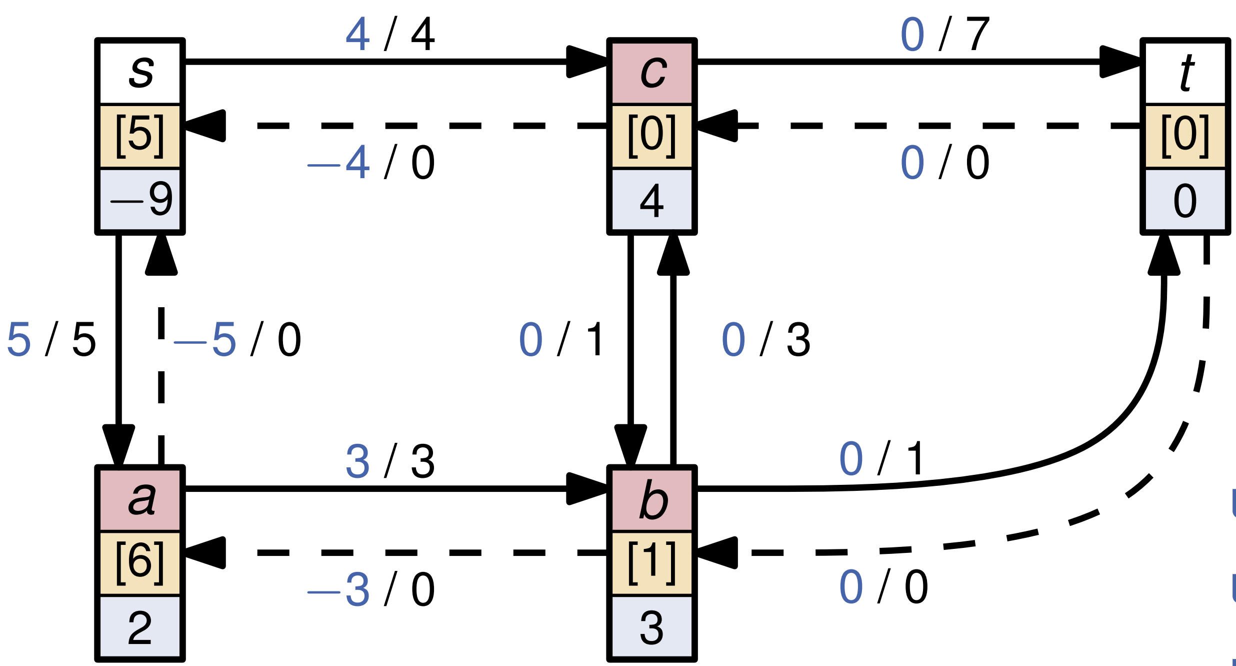
Wähle aktiven Knoten und führe PUSH / RELABEL aus:

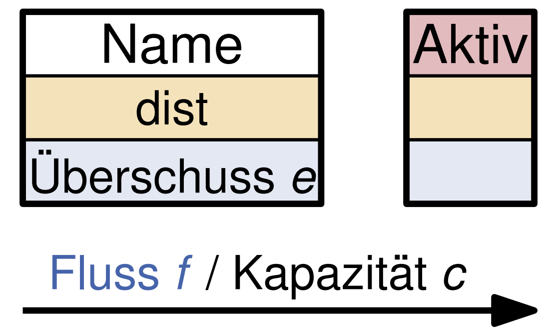
Residualnetzwerk D_f:



RELABEL(b)

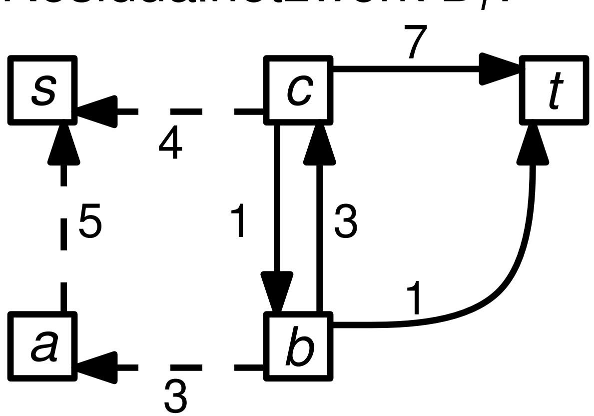
Ein Beispiel





- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

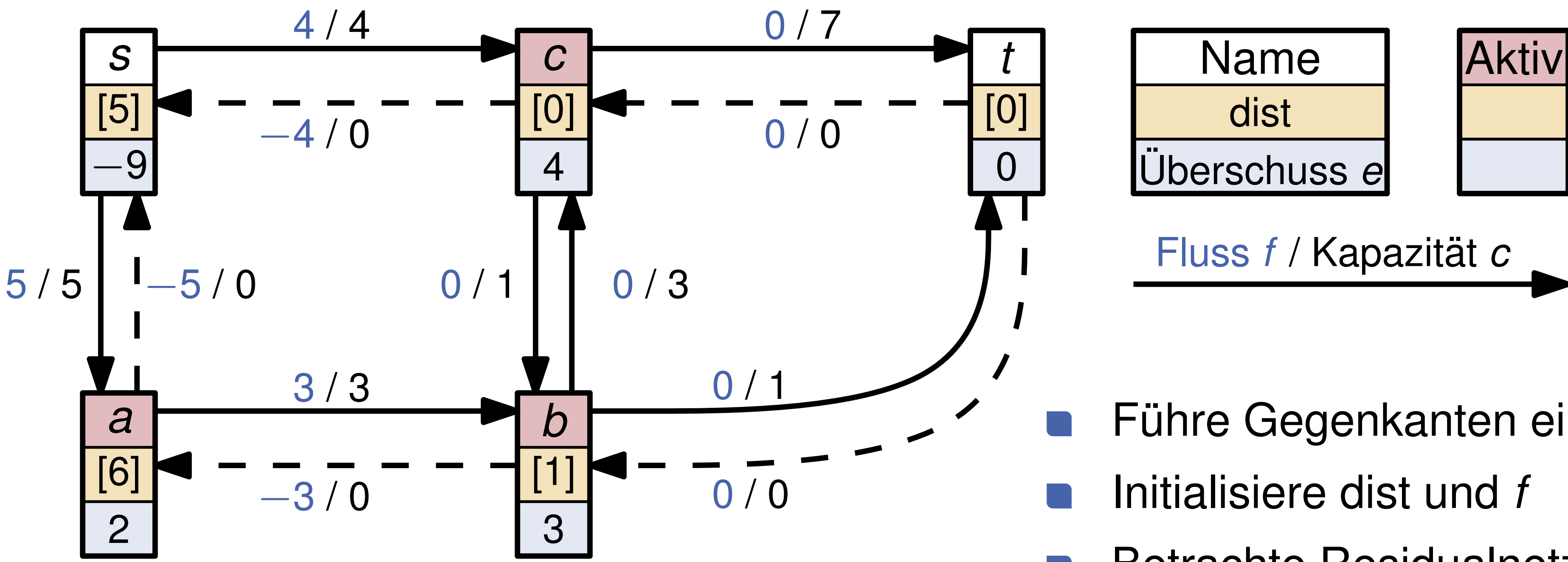
Residualnetzwerk D_f:



Wähle aktiven Knoten und führe PUSH / RELABEL aus:

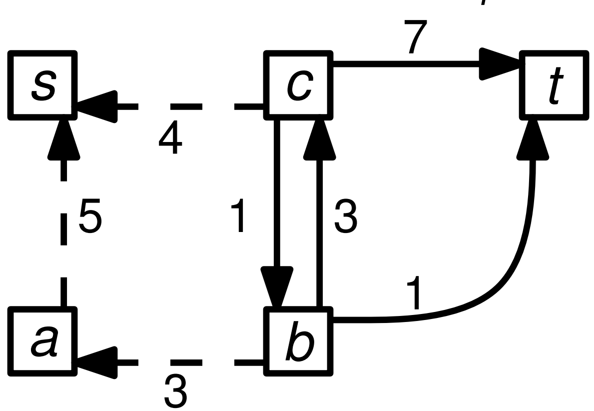
RELABEL(b)

Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

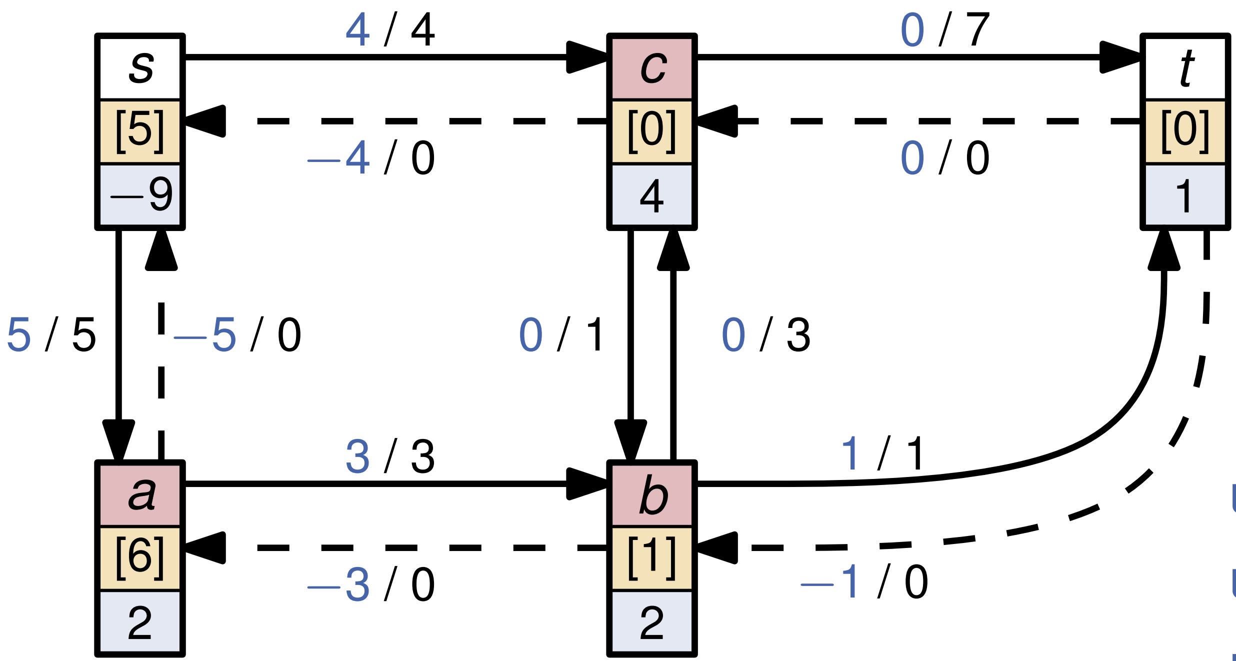
Residualnetzwerk D_f:

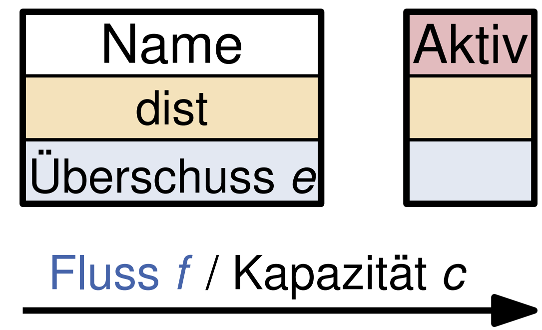


Wähle aktiven Knoten und führe PUSH / RELABEL aus:

PUSH(b, t)

Ein Beispiel

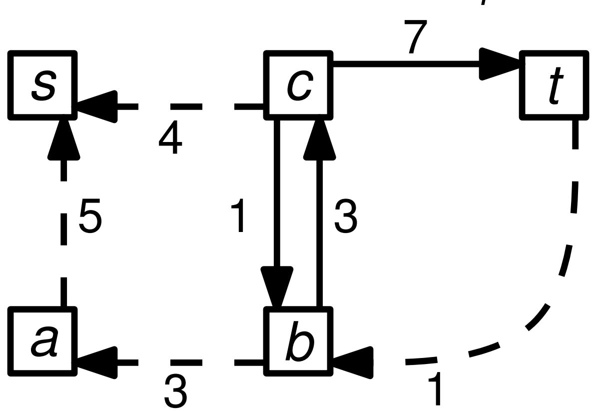




- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Wähle aktiven Knoten und führe PUSH / RELABEL aus:

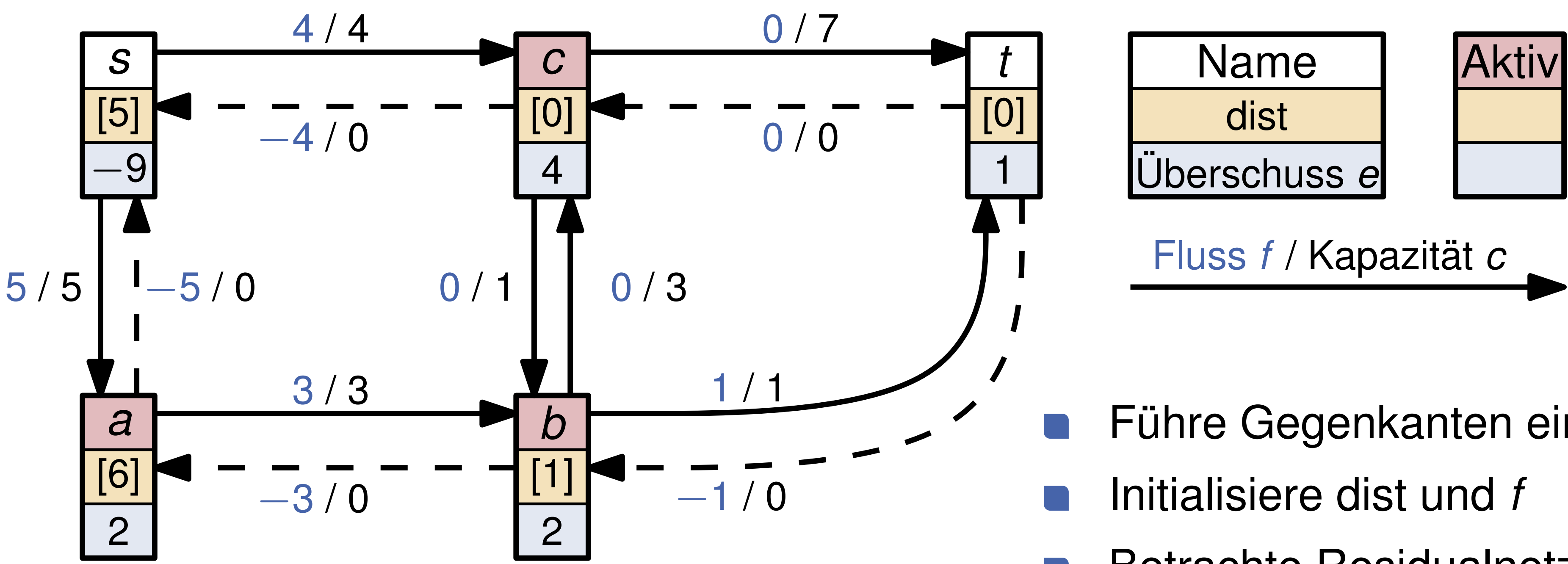
Residualnetzwerk D_f:



PUSH(b, t)

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

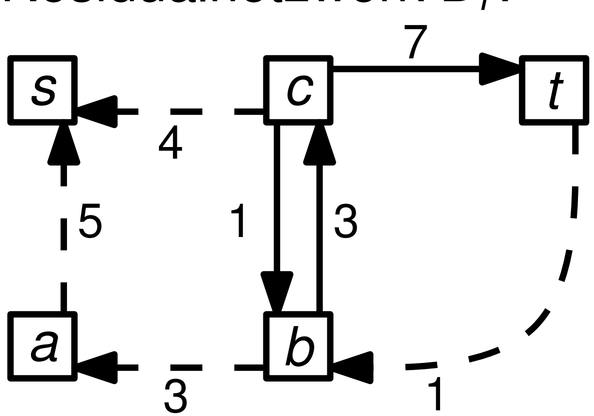
Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Wähle aktiven Knoten und führe PUSH / RELABEL aus:

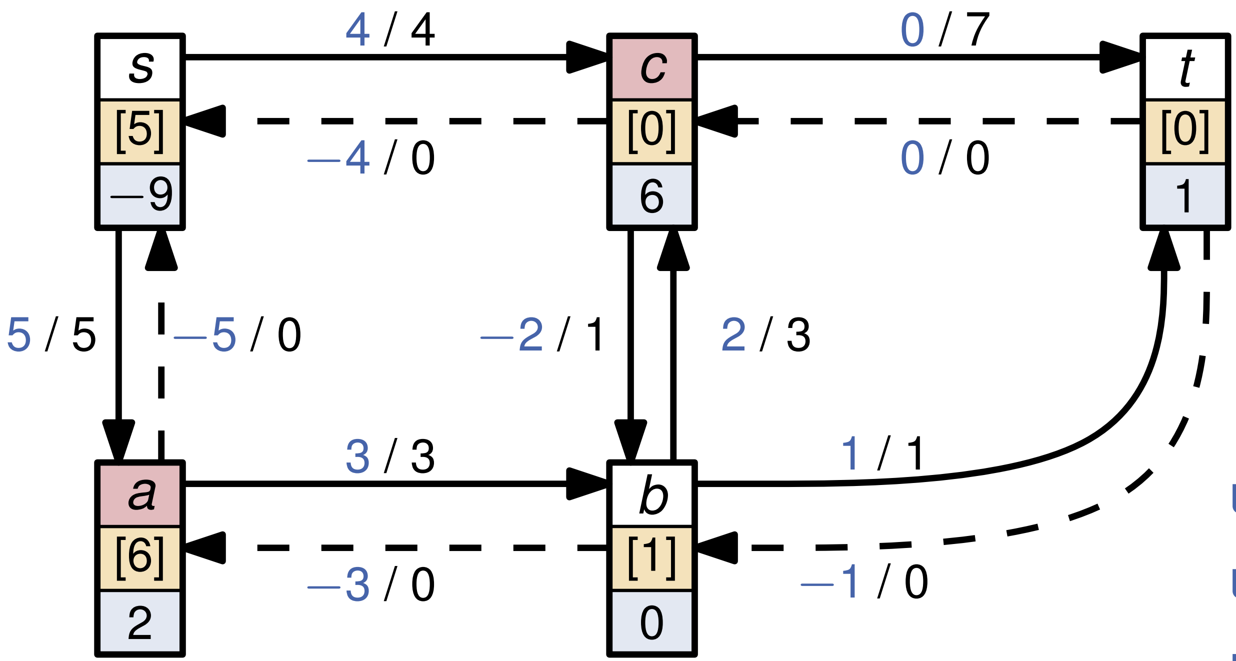
Residualnetzwerk D_f:

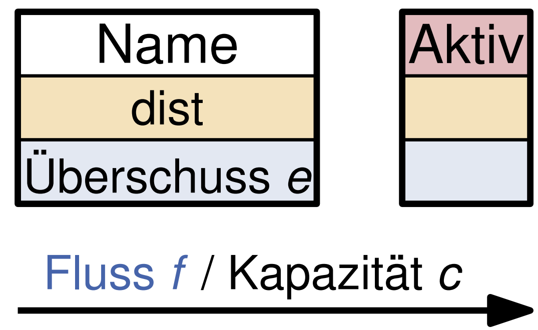


$$\textsc{Push}(b, c)$$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ein Beispiel

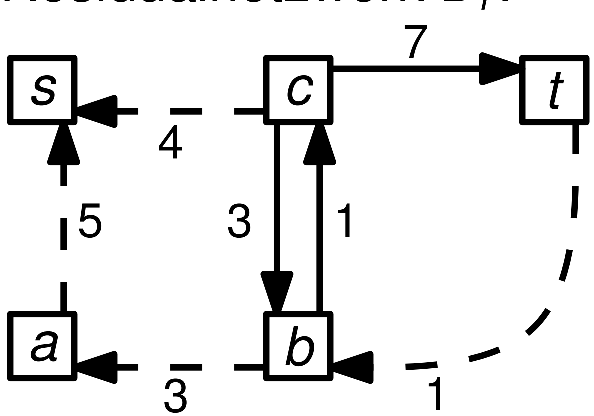




- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Wähle aktiven Knoten und führe PUSH / RELABEL aus:

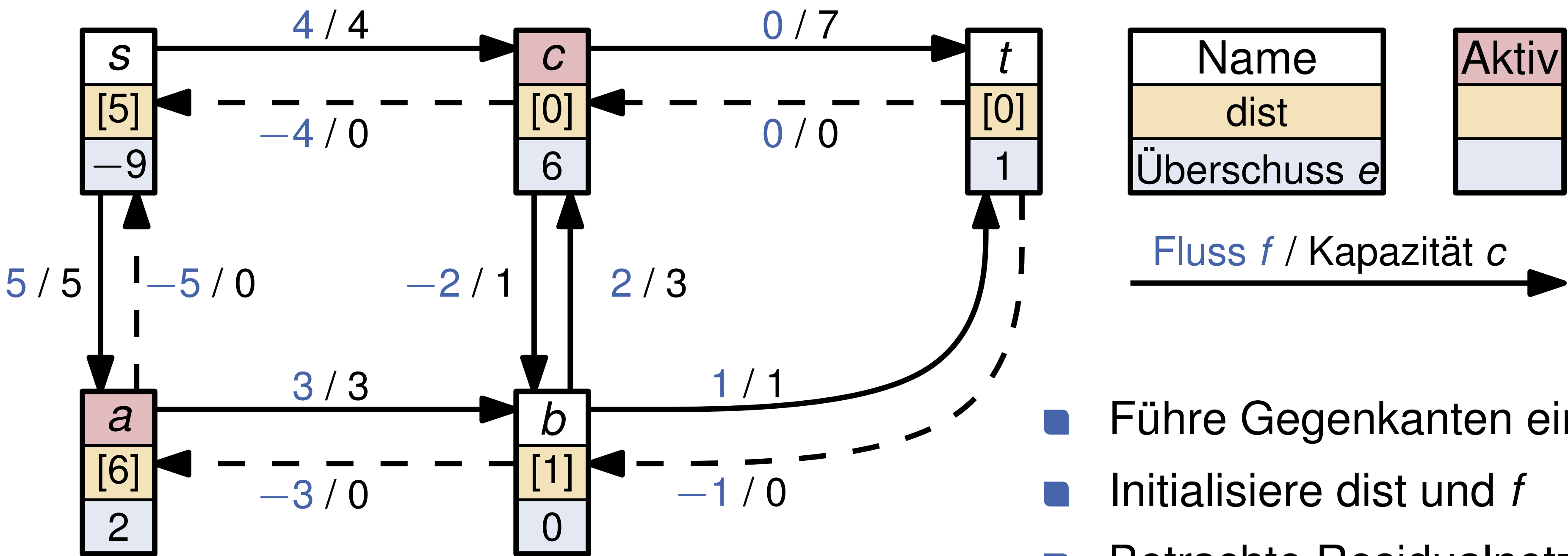
Residualnetzwerk D_f:



$$\textsc{Push}(b, c)$$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

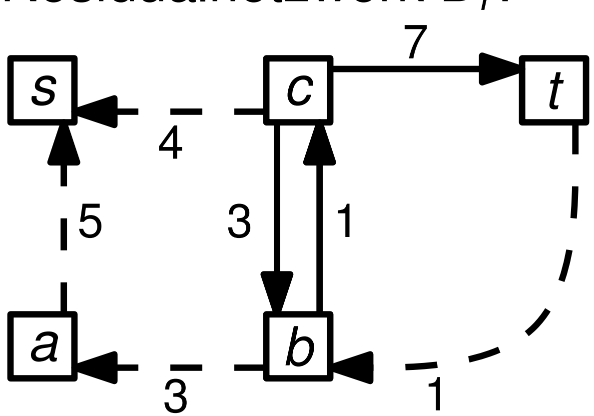
Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Wähle aktiven Knoten und führe PUSH / RELABEL aus:

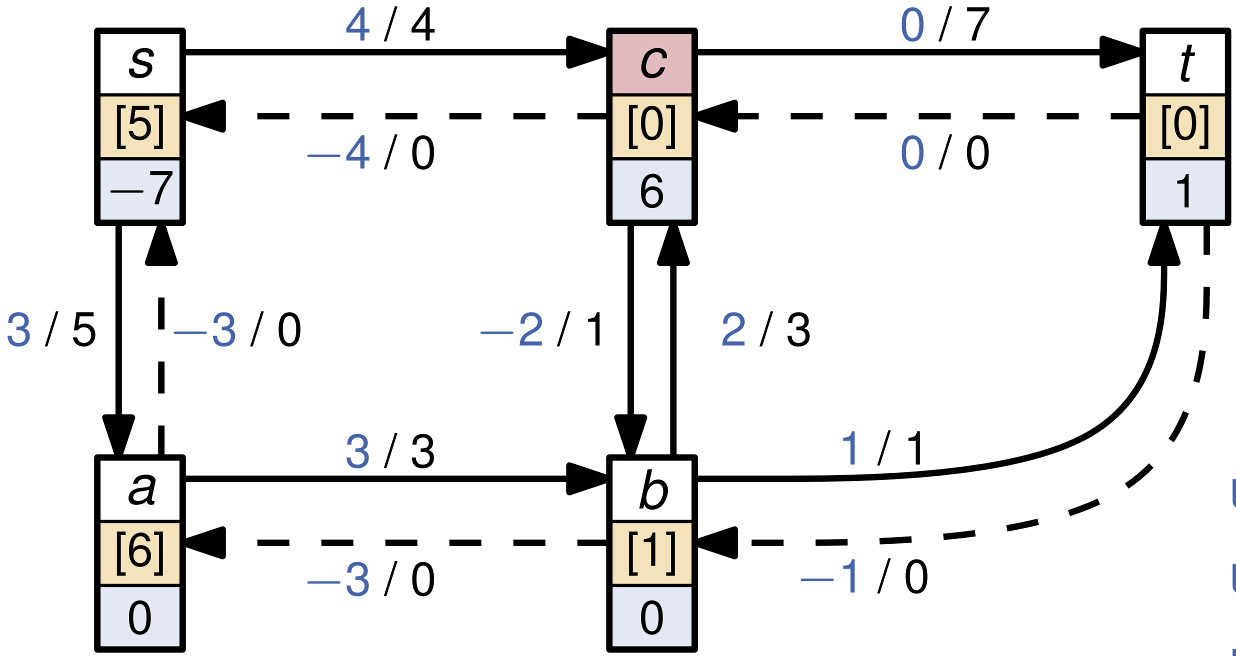
Residualnetzwerk D_f:



$$\textsc{Push}(a, s)$$

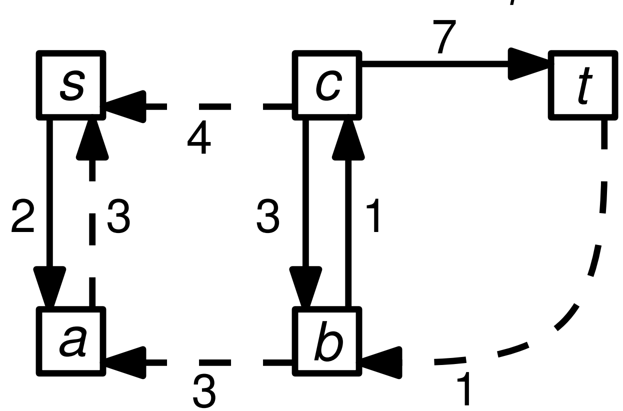
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

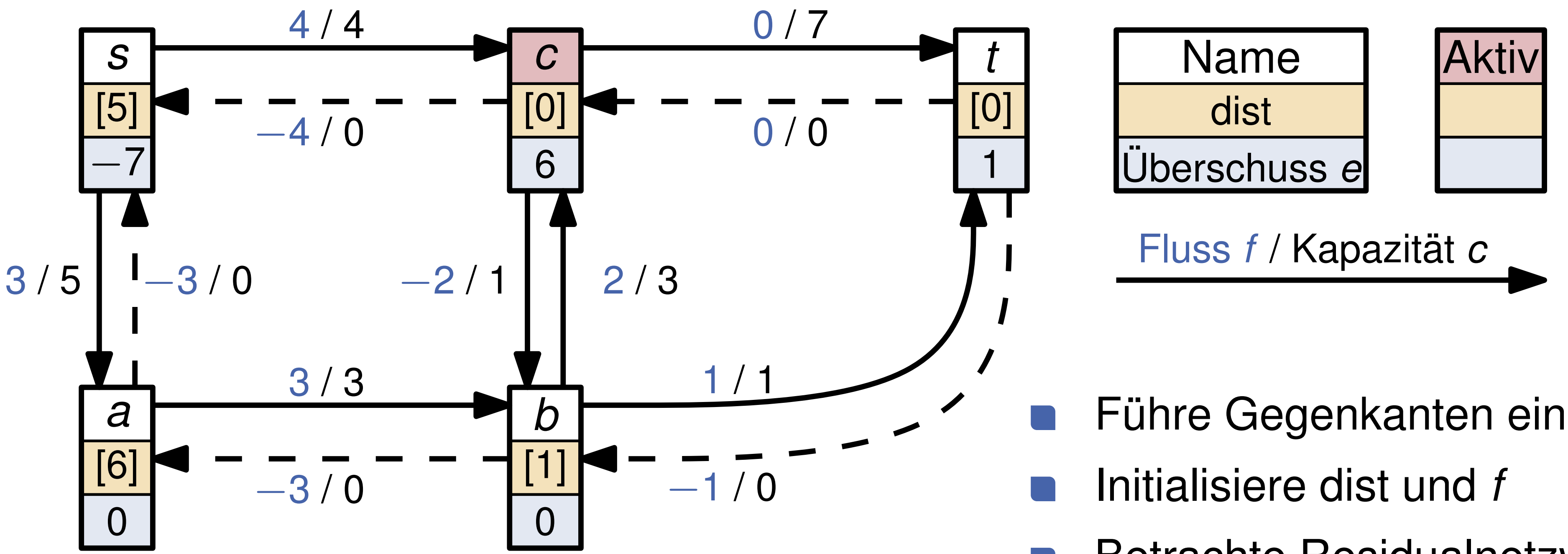
Residualnetzwerk D_f:



Wähle aktiven Knoten und führe PUSH / RELABEL aus:

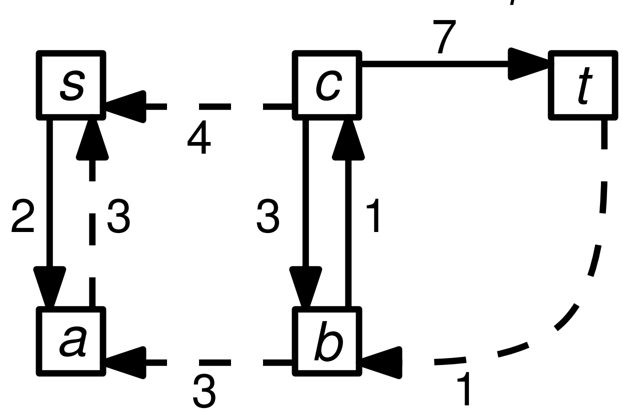
$$\textsc{Push}(a, s)$$

Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Residualnetzwerk D_f:

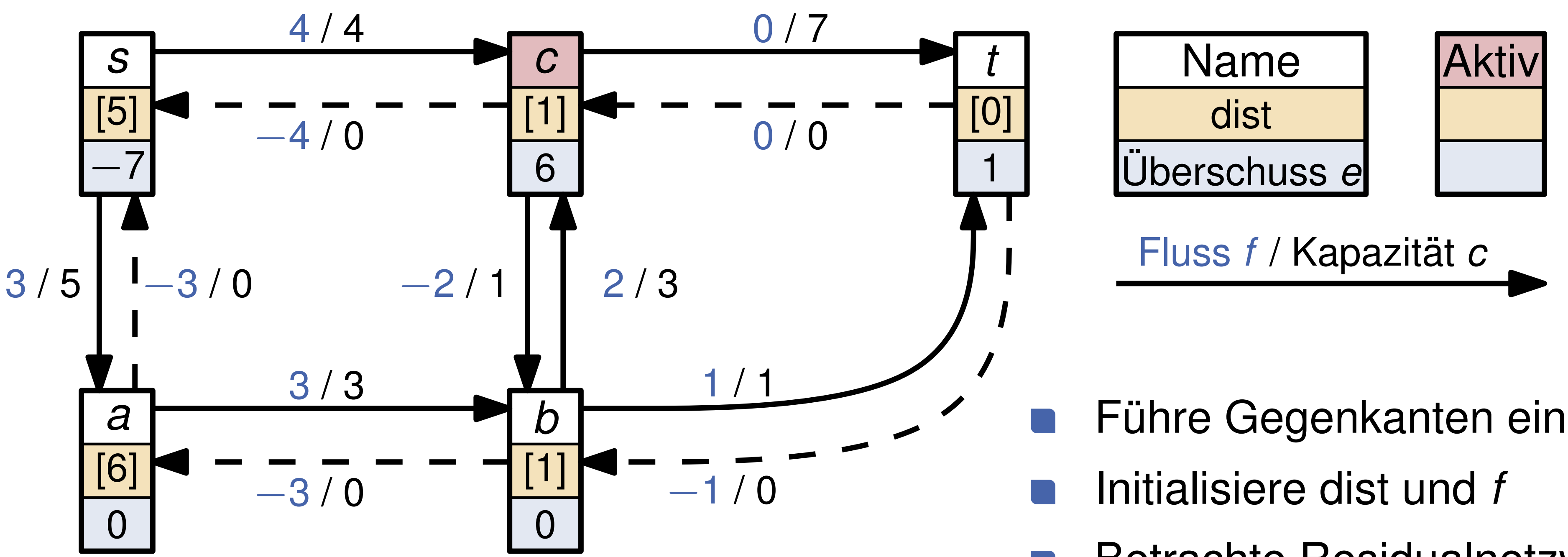


Wähle aktiven Knoten und führe PUSH / RELABEL aus:

RELABEL(c)

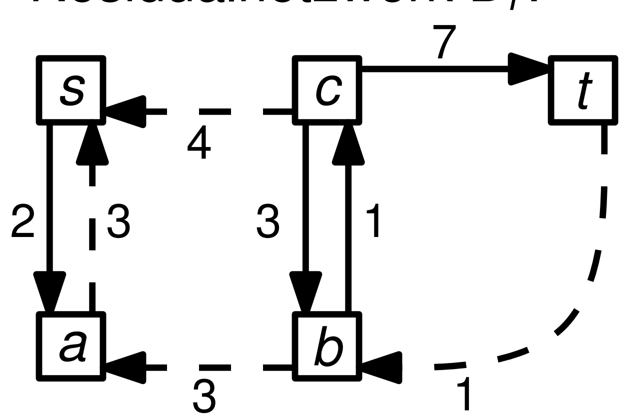
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Residualnetzwerk D_f:

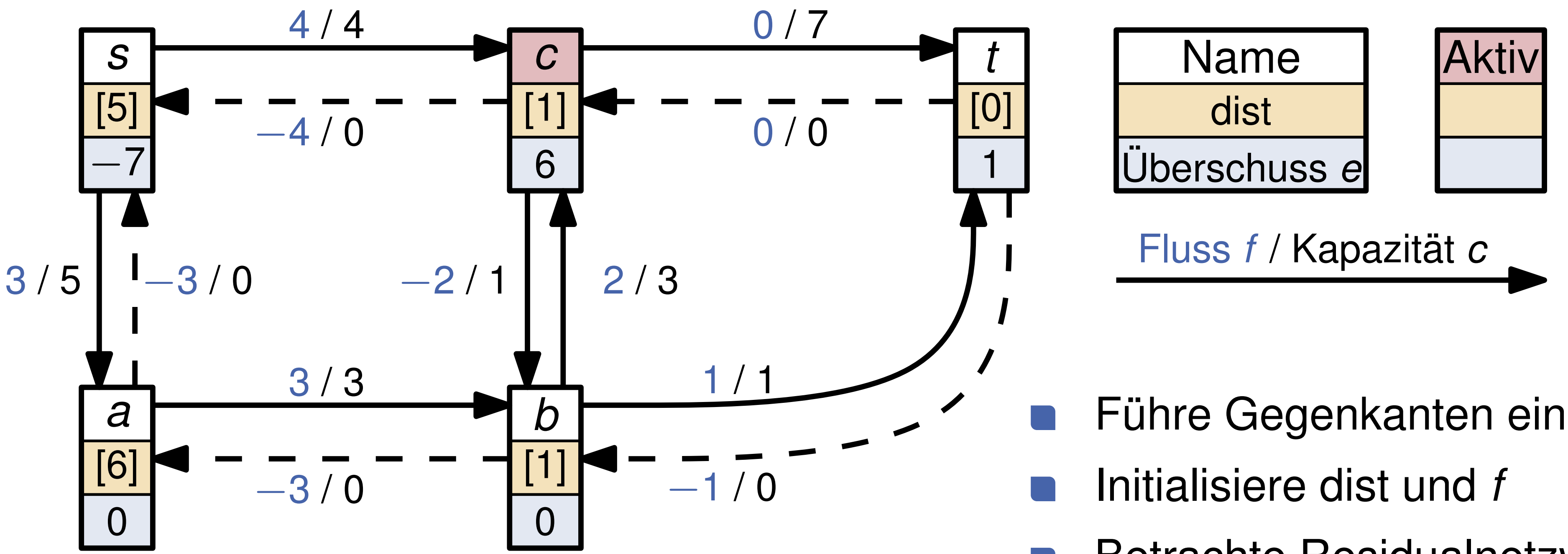


Wähle aktiven Knoten und führe PUSH / RELABEL aus:

RELABEL(c)

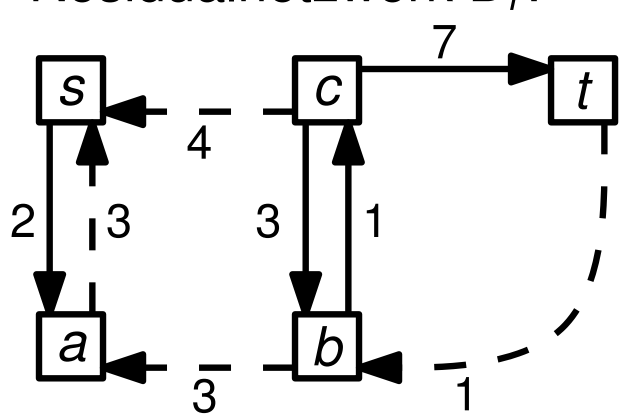
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ein Beispiel



- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Residualnetzwerk D_f:

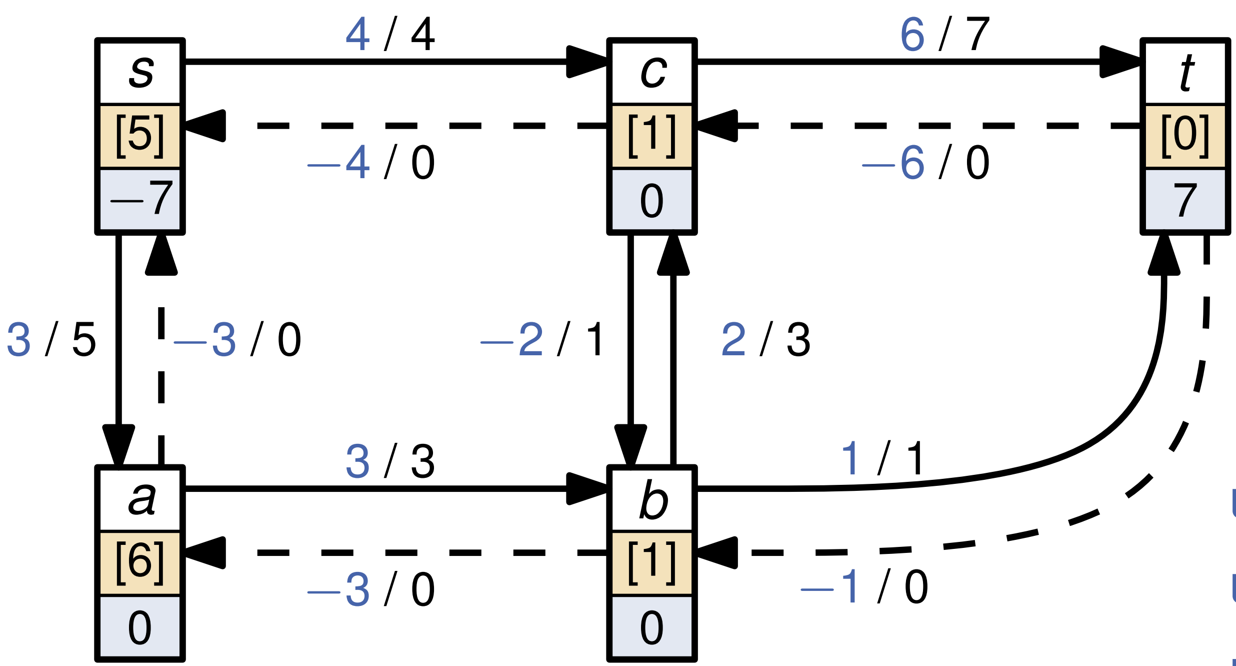


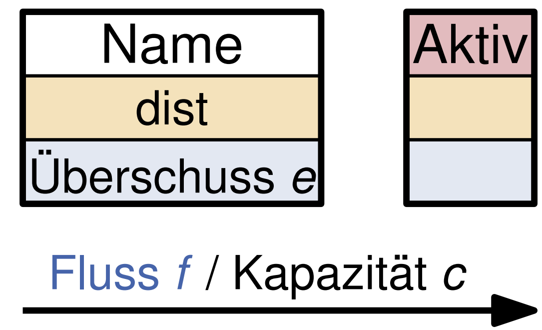
Wähle aktiven Knoten und führe PUSH / RELABEL aus:

$$\textsc{Push}(c, t)$$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ein Beispiel

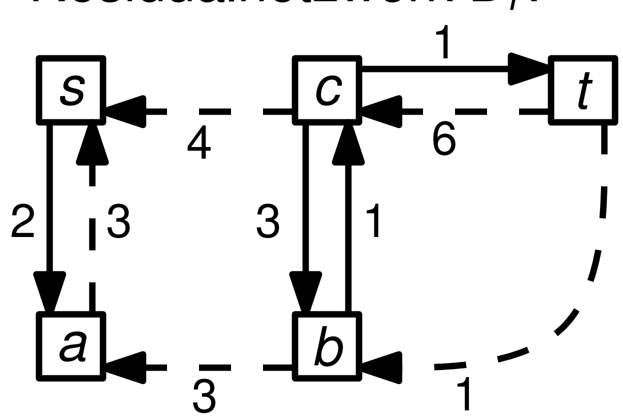




- Führe Gegenkanten ein
- Initialisiere dist und f
- Betrachte Residualnetzwerk

Wähle aktiven Knoten und führe PUSH / RELABEL aus:

Residualnetzwerk D_f:

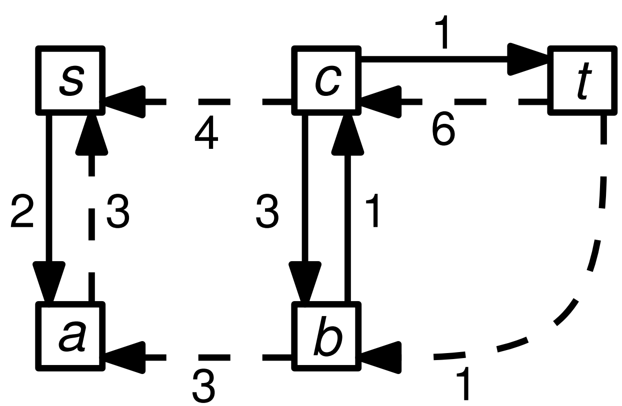


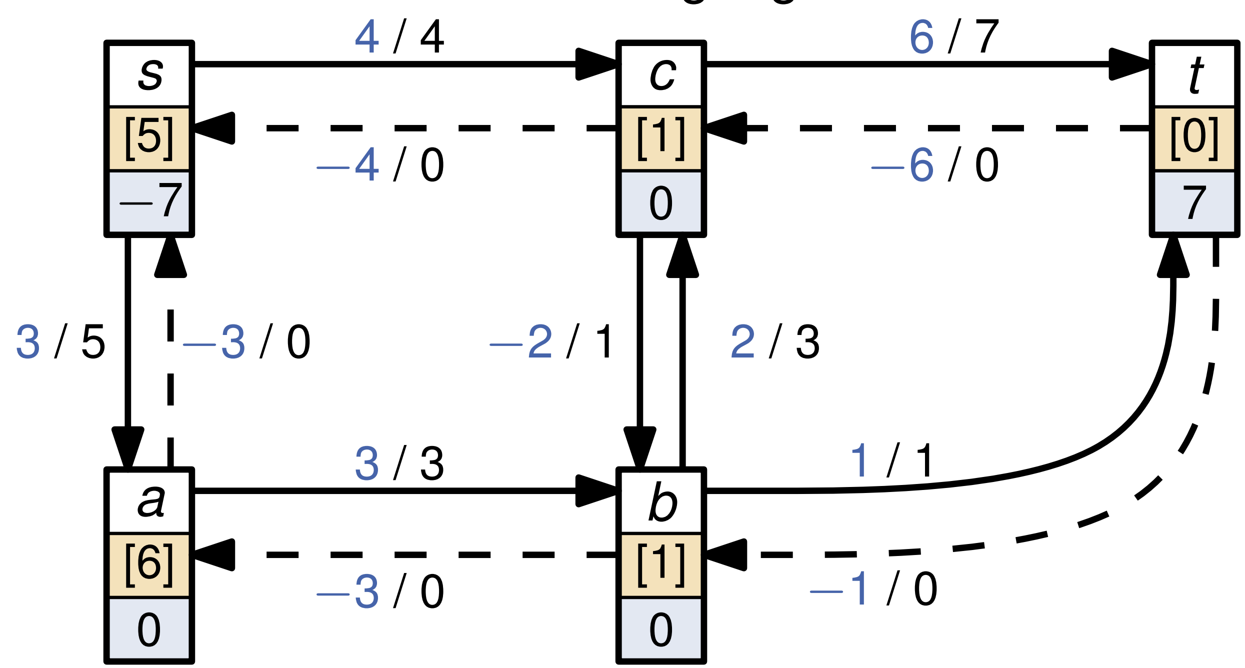
$$\textsc{Push}(c, t)$$

Problem 1

Gegeben sei ein Flussnetzwerk zusammen mit dem Ergebnis des *Push-Relabel*-Algorithmus. D.h. es stehen folgende Informationen zur Verfügung:

- Ein maximaler Fluss f.
- Residualnetzwerk zu f.
- Überschüsse aller Knoten.
- Label dist(v) für alle Knoten.



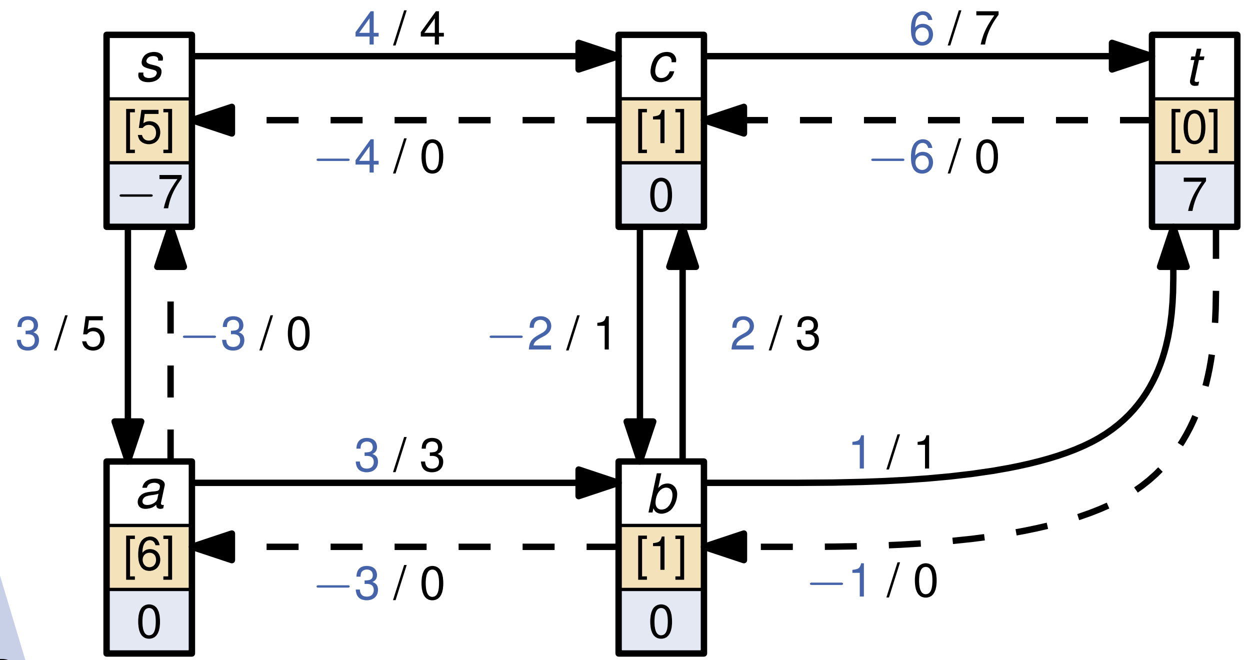


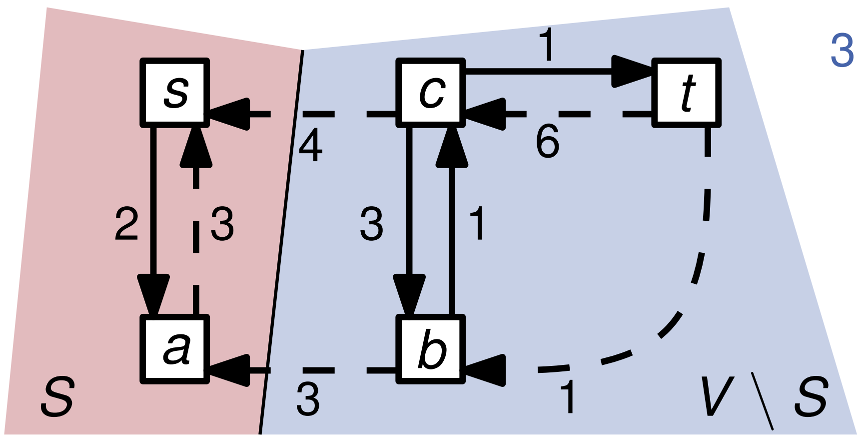
Gesucht: Algorithmus der einen minimalen s-t-Schnitt $(S, V \setminus S)$ berechnet.

Problem 1

Gegeben sei ein Flussnetzwerk zusammen mit dem Ergebnis des *Push-Relabel*-Algorithmus. D.h. es stehen folgende Informationen zur Verfügung:

- Ein maximaler Fluss f.
- Residualnetzwerk zu f.
- Überschüsse aller Knoten.
- Label dist(v) für alle Knoten.





Gesucht: Algorithmus der einen minimalen s-t-Schnitt $(S, V \setminus S)$ berechnet.

Eine Partition $(S, V \setminus S)$ tut das richtige genau dann wenn

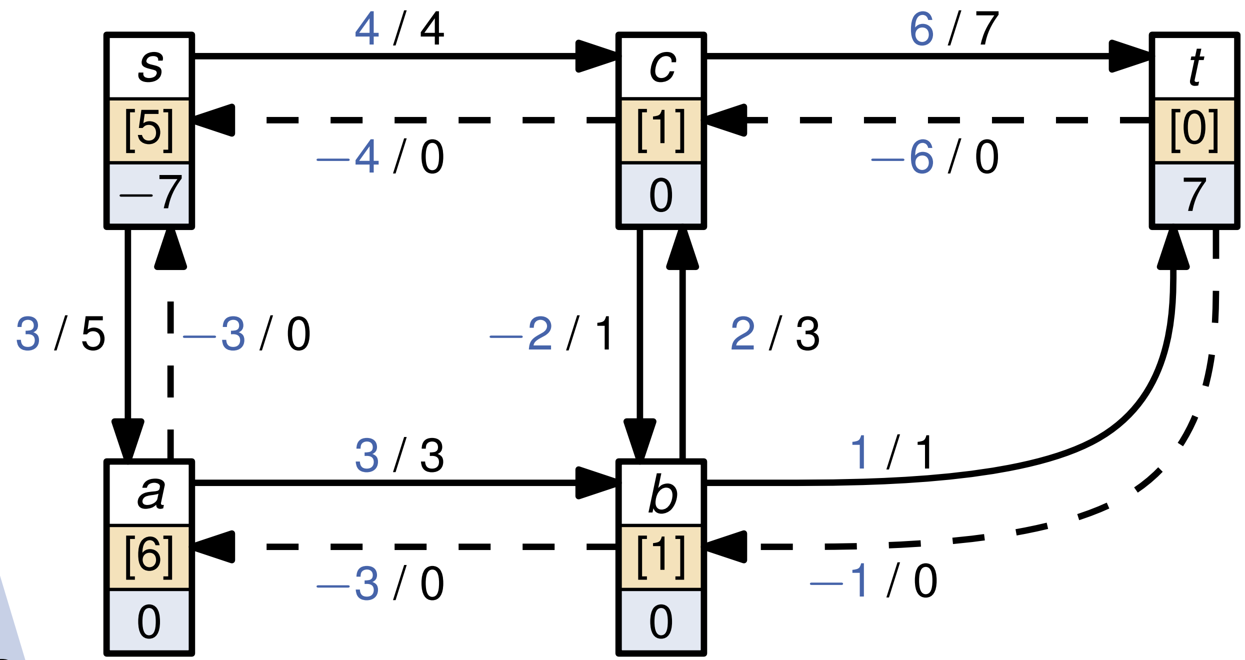
- $s \in S$ und $t \in V \setminus S$
- Das Residualnetzwerk D_f enthält keine Kante von S nach $V \setminus S$.

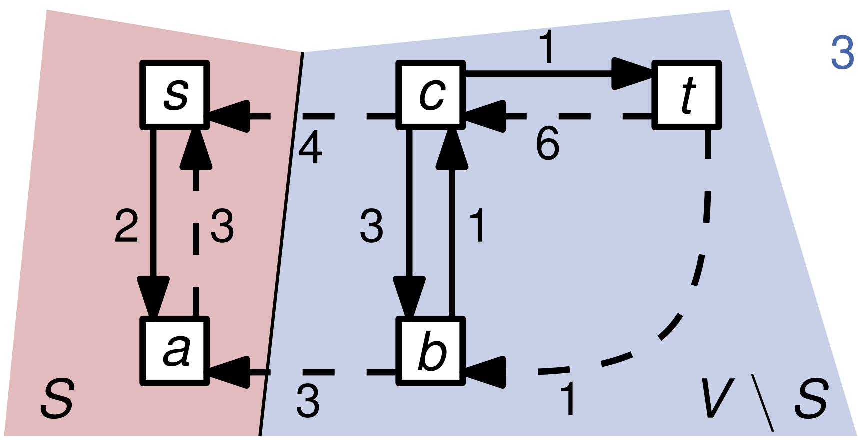
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 1

Gegeben sei ein Flussnetzwerk zusammen mit dem Ergebnis des *Push-Relabel*-Algorithmus. D.h. es stehen folgende Informationen zur Verfügung:

- Ein maximaler Fluss f.
- Residualnetzwerk zu f.
- Überschüsse aller Knoten.
- Label dist(v) für alle Knoten.





Gesucht: Algorithmus der einen minimalen s-t-Schnitt $(S, V \setminus S)$ berechnet.

Eine Partition $(S, V \setminus S)$ tut das richtige genau dann wenn

- $s \in S$ und $t \in V \setminus S$
- Das Residualnetzwerk D_f enthält keine Kante von S nach $V \setminus S$.

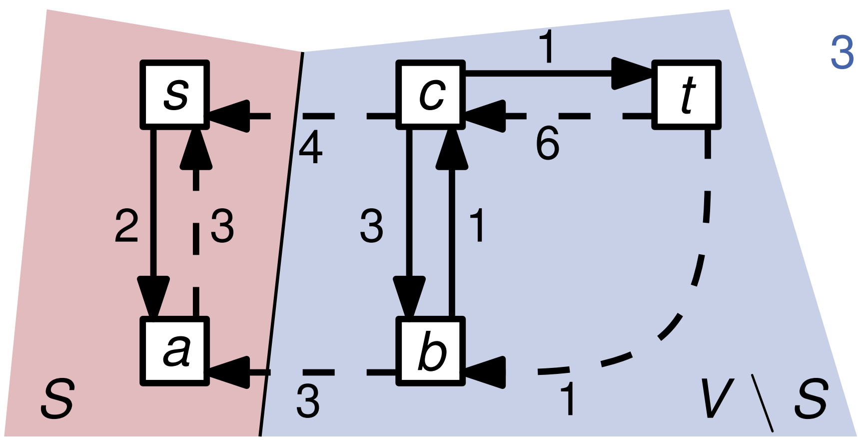
Ermittle mittels Breitensuche in D_f alle von s aus erreichbaren Knoten. $\Rightarrow O(|E|)$ Zeit.

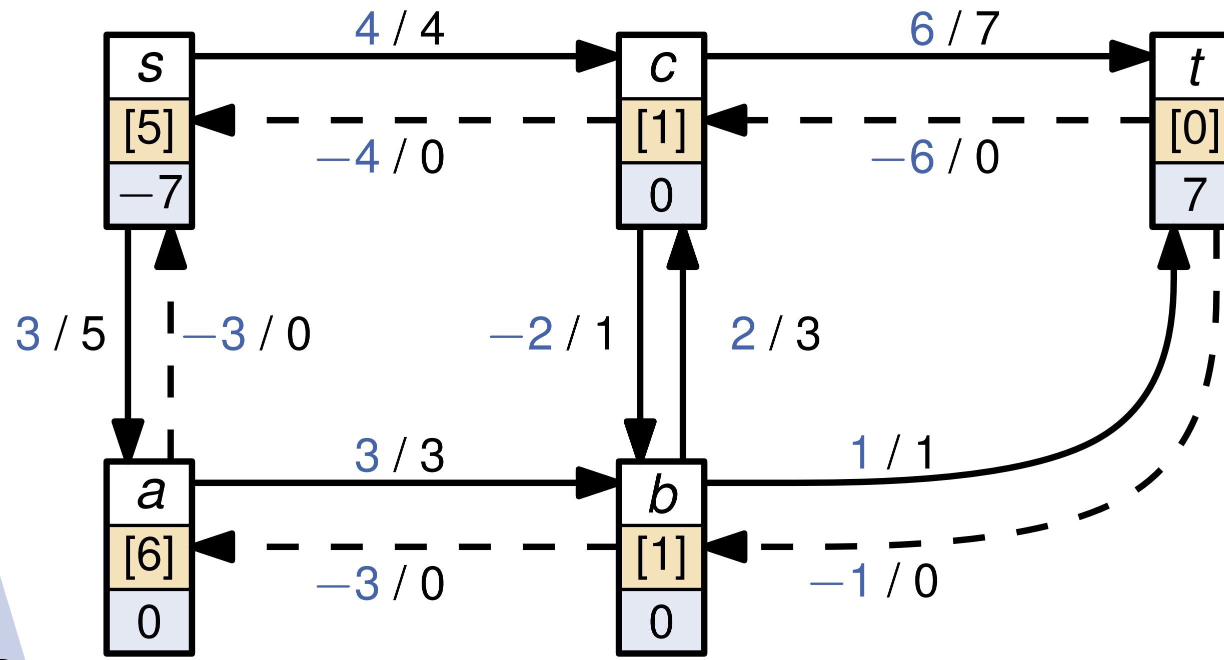
Problem 1

Karlsruher Institut für Technologie

Gegeben sei ein Flussnetzwerk zusammen mit dem Ergebnis des *Push-Relabel*-Algorithmus. D.h. es stehen folgende Informationen zur Verfügung:

- Ein maximaler Fluss f.
- Residualnetzwerk zu f.
- Überschüsse aller Knoten.
- Label dist(v) für alle Knoten.





Gesucht: Algorithmus der einen minimalen s-t-Schnitt $(S, V \setminus S)$ berechnet.

Eine Partition $(S, V \setminus S)$ tut da

Hinweis: Es gibt einen Algorithmus Laufzeit $O(|V|)$.

- $s \in S$ und $t \in V \setminus S$
- Das Residualnetzwerk D_f enthält keine Kante von S nach $V \setminus S$.

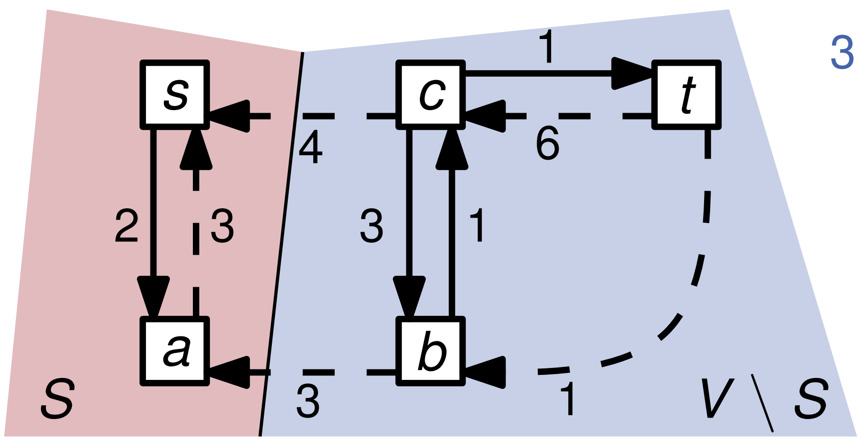
Ermittle mittels Breitensuche in D_f alle von s aus erreichbaren Knoten. $\Rightarrow O(|E|)$ Zeit.

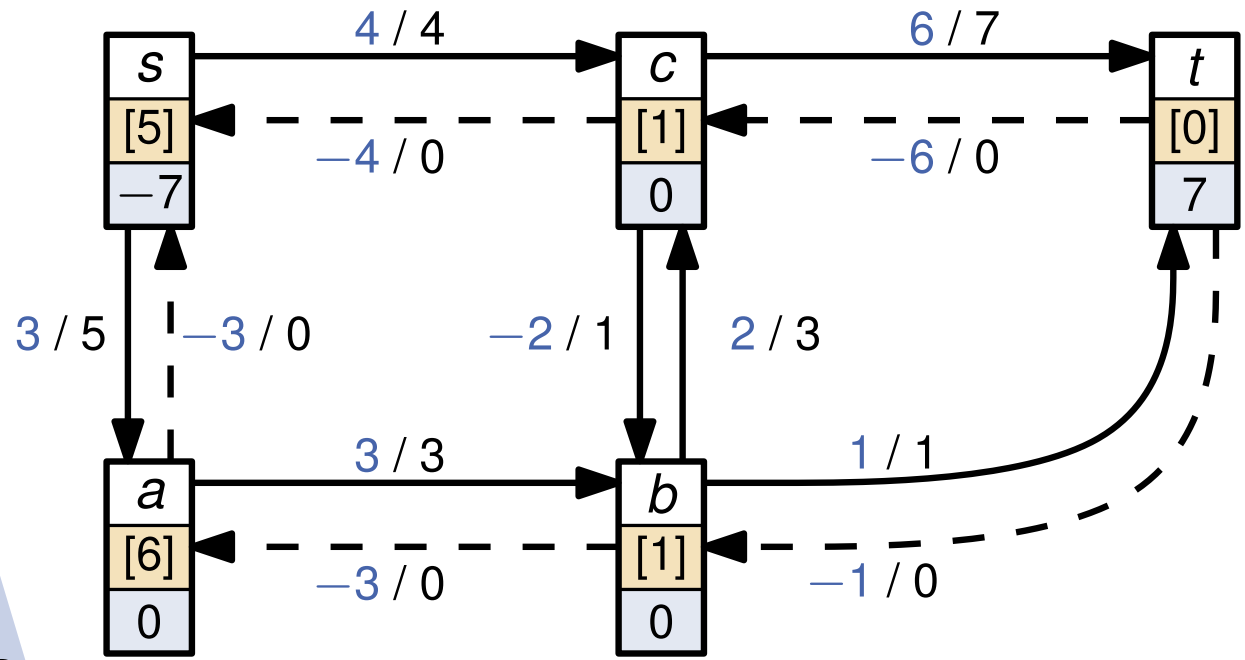
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 1

Gegeben sei ein Flussnetzwerk zusammen mit dem Ergebnis des *Push-Relabel*-Algorithmus. D.h. es stehen folgende Informationen zur Verfügung:

- Ein maximaler Fluss f.
- Residualnetzwerk zu f.
- Überschüsse aller Knoten.
- Label dist(v) für alle Knoten.





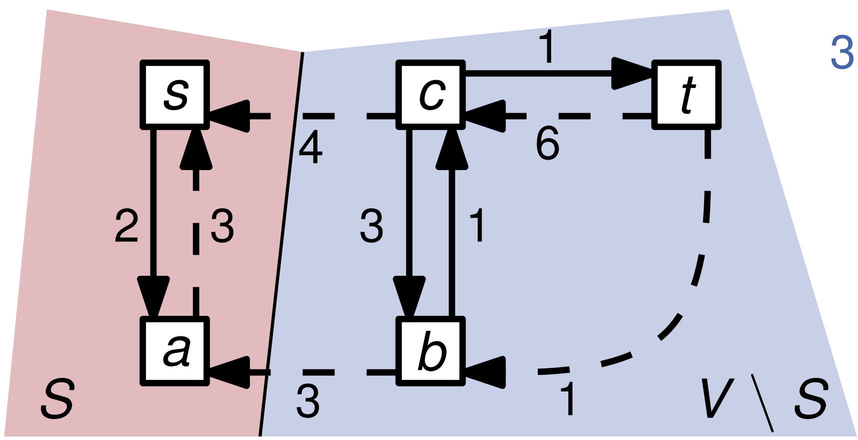
Gesucht: Algorithmus der einen minimalen s-t-Schnitt $(S, V \setminus S)$ berechnet.

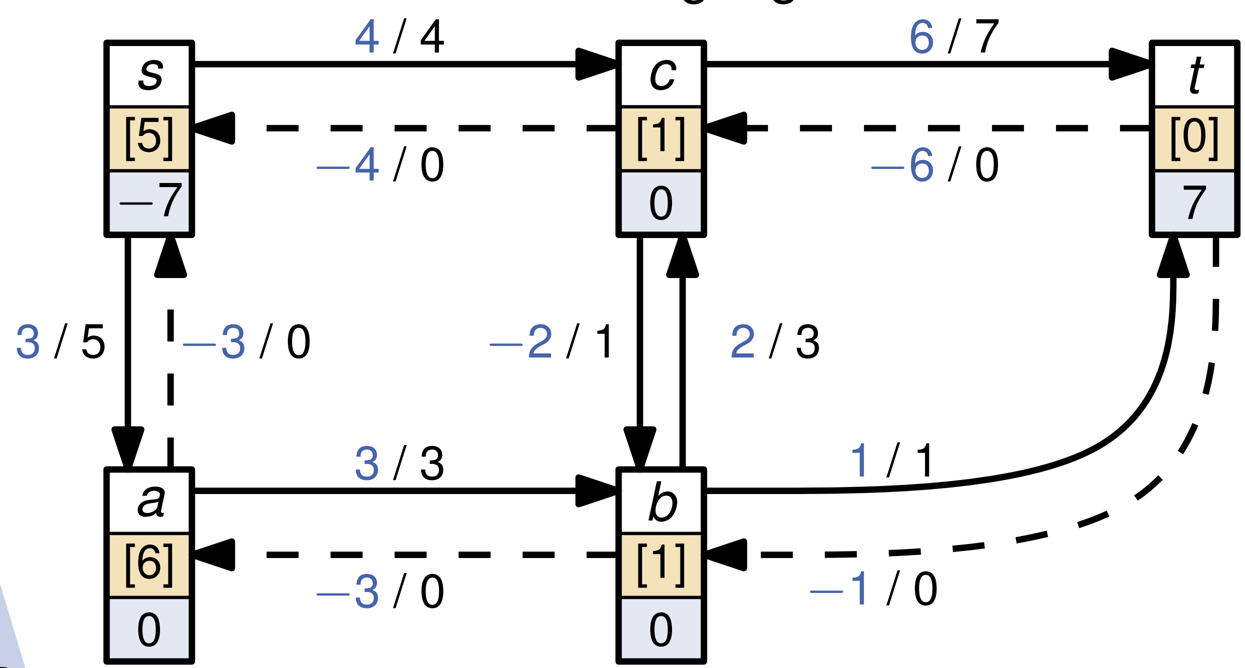
Behauptung 1: Es gibt eine Zahl $0 < x < |V|$, sodass kein Knoten x als Label hat.

Problem 1

Gegeben sei ein Flussnetzwerk zusammen mit dem Ergebnis des *Push-Relabel*-Algorithmus. D.h. es stehen folgende Informationen zur Verfügung:

- Ein maximaler Fluss f.
- Residualnetzwerk zu f.
- Überschüsse aller Knoten.
- Label dist(v) für alle Knoten.





Gesucht: Algorithmus der einen minimalen s-t-Schnitt $(S, V \setminus S)$ berechnet.

Behauptung 1: Es gibt eine Zahl $0 < x < |V|$, sodass kein Knoten x als Label hat.

Behauptung 2: Die Partitionierung $S = \{v \in V \mid \text{dist}(v) > x\}$ und $V \setminus S = \{v \in V \mid \text{dist}(v) < x\}$ ist ein minimaler s-t-Schnitt.

Problem 1

Behauptung 1: Es gibt eine Zahl $0 < x < |V|$, sodass kein Knoten x als Label hat.

- $\text{dist}(s) = |V|$ und $\text{dist}(t) = 0$
- Die restlichen $|V| - 2$ Knoten können nicht alle $|V| - 1$ möglichen x mit $0 < x < |V|$ abdecken.

Behauptung 2: Die Partitionierung $S = \{v \in V \mid \text{dist}(v) > x\}$ und $V \setminus S = \{v \in V \mid \text{dist}(v) < x\}$ ist ein minimaler s-t-Schnitt.

Problem 1

Behauptung 1: Es gibt eine Zahl $0 < x < |V|$, sodass kein Knoten x als Label hat.

- $\text{dist}(s) = |V|$ und $\text{dist}(t) = 0$
- Die restlichen $|V| - 2$ Knoten können nicht alle $|V| - 1$ möglichen x mit $0 < x < |V|$ abdecken.

Behauptung 2: Die Partitionierung $S = \{v \in V \mid \text{dist}(v) > x\}$ und $V \setminus S = \{v \in V \mid \text{dist}(v) < x\}$ ist ein minimaler s-t-Schnitt.

Zeige: Es gibt keine Kante (u, v) im Residualnetzwerk mit $u \in S$ und $v \in V \setminus S$.

Problem 1

Behauptung 1: Es gibt eine Zahl $0 < x < |V|$, sodass kein Knoten x als Label hat.

- $\text{dist}(s) = |V|$ und $\text{dist}(t) = 0$
- Die restlichen $|V| - 2$ Knoten können nicht alle $|V| - 1$ möglichen x mit $0 < x < |V|$ abdecken.

Behauptung 2: Die Partitionierung $S = \{v \in V \mid \text{dist}(v) > x\}$ und $V \setminus S = \{v \in V \mid \text{dist}(v) < x\}$ ist ein minimaler s-t-Schnitt.

Zeige: Es gibt keine Kante (u, v) im Residualnetzwerk mit $u \in S$ und $v \in V \setminus S$.

- Angenommen (u, v) ist eine solche Kante, dann gilt $\text{dist}(u) \geq \text{dist}(v) + 2$

Problem 1

Behauptung 1: Es gibt eine Zahl $0 < x < |V|$, sodass kein Knoten x als Label hat.

- $\text{dist}(s) = |V|$ und $\text{dist}(t) = 0$
- Die restlichen $|V| - 2$ Knoten können nicht alle $|V| - 1$ möglichen x mit $0 < x < |V|$ abdecken.

Behauptung 2: Die Partitionierung $S = \{v \in V \mid \text{dist}(v) > x\}$ und $V \setminus S = \{v \in V \mid \text{dist}(v) < x\}$ ist ein minimaler s-t-Schnitt.

Zeige: Es gibt keine Kante (u, v) im Residualnetzwerk mit $u \in S$ und $v \in V \setminus S$.

- Angenommen (u, v) ist eine solche Kante, dann gilt $\text{dist}(u) \geq \text{dist}(v) + 2$
- Eine zulässige Markierung erfüllt immer die folgende Eigenschaft: Wenn das Residualnetzwerk die Kante (u, v) enthält, dann gilt $\text{dist}(u) \leq \text{dist}(v) + 1$.

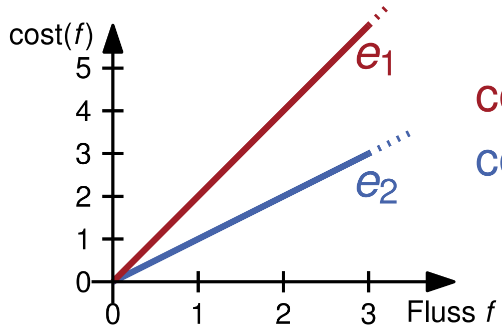
$\lightning$

Flüsse mit komplizierten Kosten

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

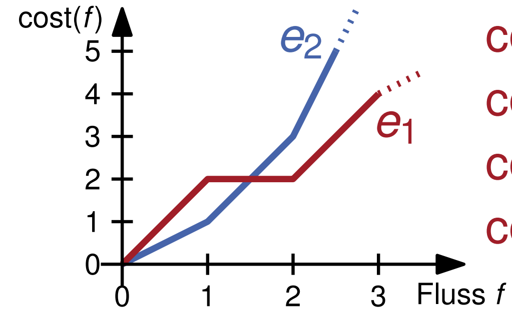
Problem 2 (a)

In der Vorlesung:
konstante Kosten pro Flusseinheit



$\text{cost}(e_1) = 2$
$\text{cost}(e_2) = 1$

Jetzt:
bel. Kostenfunktion für jede Kante

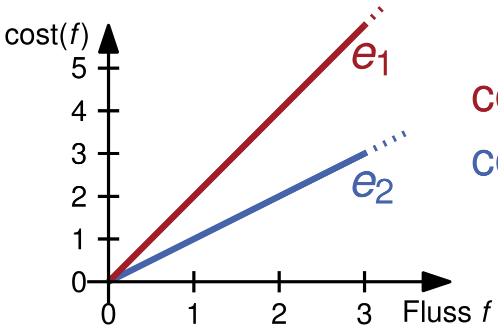


$\text{cost}_{e_1}(0) = 0$ $\text{cost}_{e_2}(0) = 0$
$\text{cost}_{e_1}(1) = 2$ $\text{cost}_{e_2}(1) = 1$
$\text{cost}_{e_1}(2) = 2$ $\text{cost}_{e_2}(2) = 3$
$\text{cost}_{e_1}(3) = 4$ $\text{cost}_{e_2}(3) = 5$

(a) Wenn cost_e für alle $e \in E$ konvex ist, dann kann MINCOSTFLOW effizient gelöst werden. (unter der Voraussetzung, dass die Kapazitäten polynomiell beschränkt sind).
Hinweis: Modellieren sie ein äquivalentes Flussnetzwerk mit herkömmlichen Kosten. Sie dürfen Mehrfachkanten benutzen.

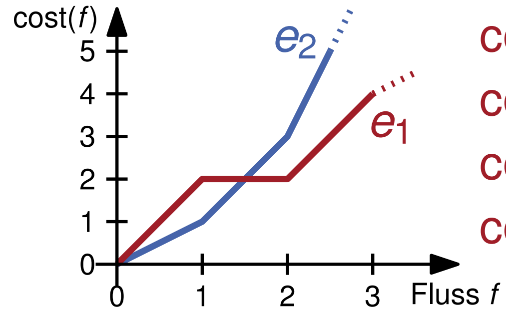
Problem 2 (a)

In der Vorlesung:
konstante Kosten pro Flusseinheit



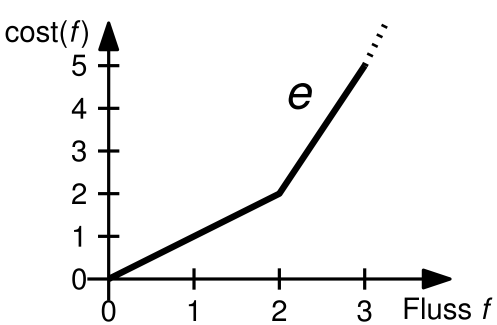
$\text{cost}(e_1) = 2$
$\text{cost}(e_2) = 1$

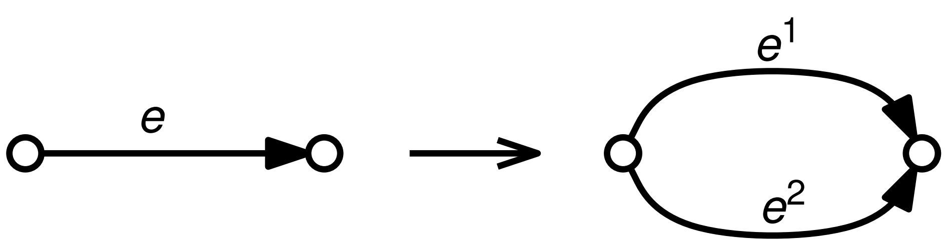
Jetzt:
bel. Kostenfunktion für jede Kante



$\text{cost}_{e_1}(0) = 0$	$\text{cost}_{e_2}(0) = 0$
$\text{cost}_{e_1}(1) = 2$	$\text{cost}_{e_2}(1) = 1$
$\text{cost}_{e_1}(2) = 2$	$\text{cost}_{e_2}(2) = 3$
$\text{cost}_{e_1}(3) = 4$	$\text{cost}_{e_2}(3) = 5$

(a) Wenn cost_e für alle $e \in E$ konvex ist, dann kann MinCostFlow effizient gelöst werden. (unter der Voraussetzung, dass die Kapazitäten polynomiell beschränkt sind).
Hinweis: Modellieren sie ein äquivalentes Flussnetzwerk mit herkömmlichen Kosten. Sie dürfen Mehrfachkanten benutzen.





$\text{cost}(e^1) = 1,\ c(e^1) = 2$

$\text{cost}(e^2) = 2,\ c(e^2) = 1$

Problem 2 (b)

(b) Für potentiell nicht konvexe Kostenfunktionen ist das Entscheidungsproblem von MinCostFlow $\mathcal{NP}$-vollständig.

(offensichtlich ist MinCostFlow in $\mathcal{NP}$)

Hinweis: 3-Dimensional-Matching ist $\mathcal{NP}$-vollständig.

Problem 2 (b)

(b) Für potentiell nicht konvexe Kostenfunktionen ist das Entscheidungsproblem von MinCostFlow $\mathcal{NP}$-vollständig.
(offensichtlich ist MinCostFlow in $\mathcal{NP}$)

Hinweis: 3-Dimensional-Matching ist $\mathcal{NP}$-vollständig.

3-Dimensional-Matching

Geg.: Endliche, disjunkte Mengen X, Y und Z mit einer Menge von Tripeln $T \subseteq X \times Y \times Z$, sowie Zahl $k \in \mathbb{N}$.

Ges.: Teilmenge $M \subseteq T$ mit $|M| \geq k$, sodass jedes Element aus X, Y bzw. Z in maximal einem Tripel in M auftaucht.

Problem 2 (b)

(b) Für potentiell nicht konvexe Kostenfunktionen ist das Entscheidungsproblem von MINCOSTFLOW $\mathcal{NP}$-vollständig. (offensichtlich ist MINCOSTFLOW in $\mathcal{NP}$)

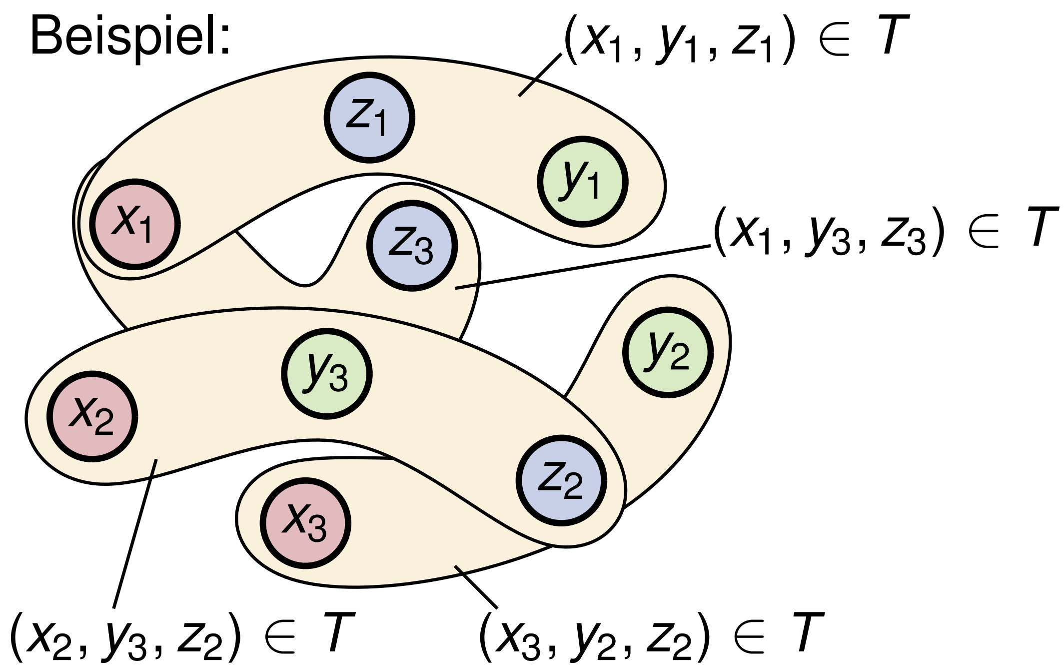
Hinweis: 3-DIMENSIONAL-MATCHING ist $\mathcal{NP}$-vollständig.

3-DIMENSIONAL-MATCHING

Geg.: Endliche, disjunkte Mengen X, Y und Z mit einer Menge von Tripeln $T \subseteq X \times Y \times Z$, sowie Zahl $k \in \mathbb{N}$.

Ges.: Teilmenge $M \subseteq T$ mit $|M| \geq k$, sodass jedes Element aus X, Y bzw. Z in maximal einem Tripel in M auftaucht.

Beispiel:



Für M ausgewählte Tripel dürfen sich nicht überlappen!

Problem 2 (b)

(b) Für potentiell nicht konvexe Kostenfunktionen ist das Entscheidungsproblem von MinCostFlow $\mathcal{NP}$-vollständig. (offensichtlich ist MinCostFlow in $\mathcal{NP}$)

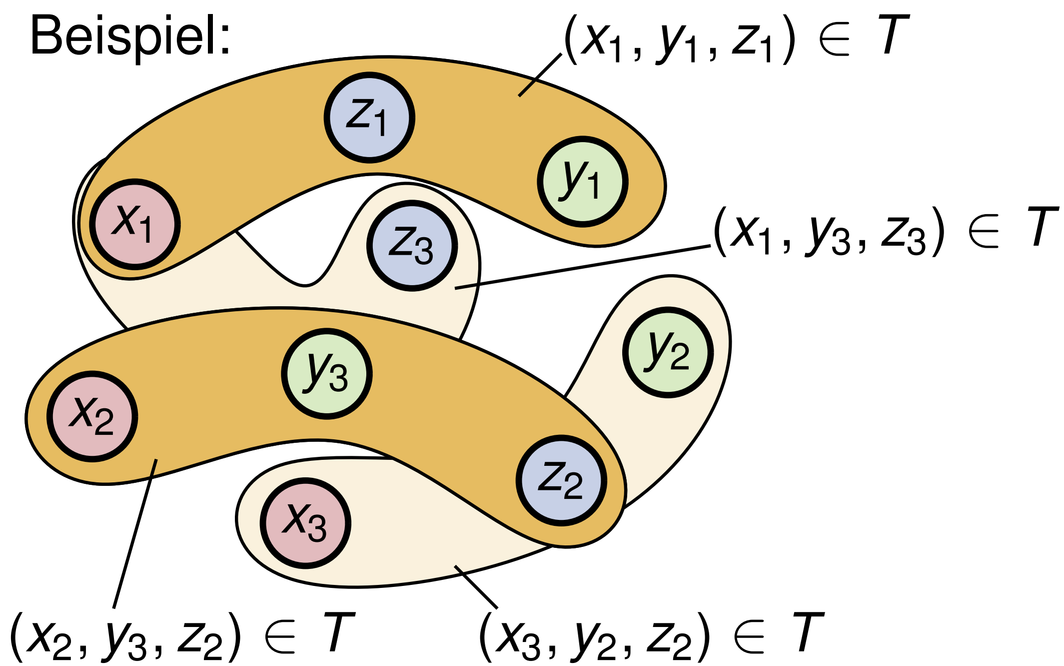
Hinweis: 3-Dimensional-Matching ist $\mathcal{NP}$-vollständig.

3-Dimensional-Matching

Geg.: Endliche, disjunkte Mengen X, Y und Z mit einer Menge von Tripeln $T \subseteq X \times Y \times Z$, sowie Zahl $k \in \mathbb{N}$.

Ges.: Teilmenge $M \subseteq T$ mit $|M| \geq k$, sodass jedes Element aus X, Y bzw. Z in maximal einem Tripel in M auftaucht.

Beispiel:



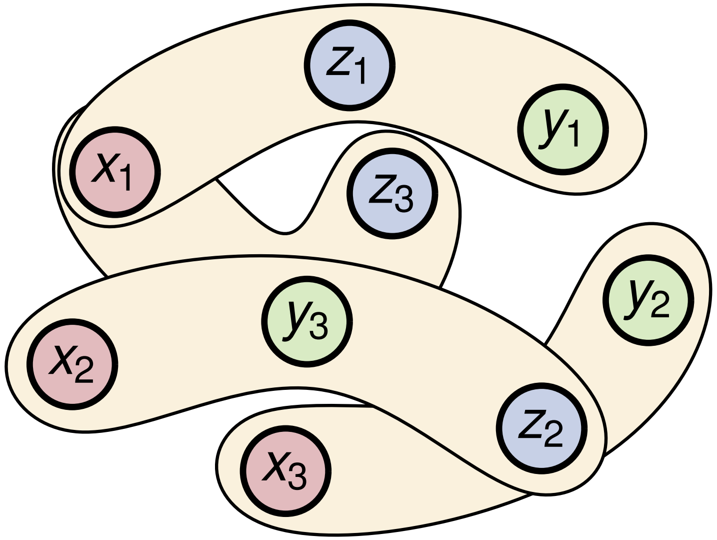
Für M ausgewählte Tripel dürfen sich nicht überlappen!

- $|M| \geq 2$ ist möglich.
- $|M \geq 3$ nicht.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

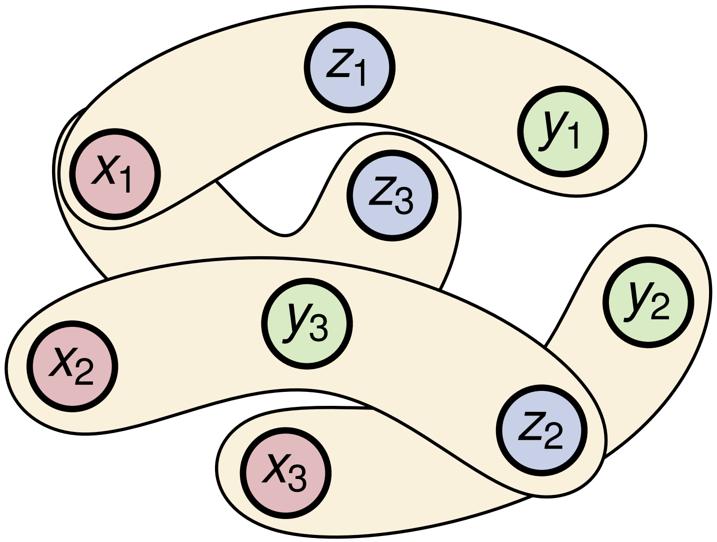
Problem 2 (b)

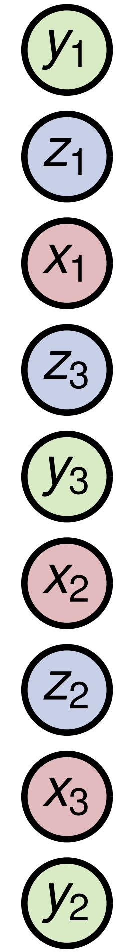
Ziel: Konstruiere eine Flussnetzwerk D, sodass ein Fluss f mit cost$(f) \leq k$ existiert, genau dann wenn es ein 3DM M mit $|M| \geq k$ gibt.



Problem 2 (b)

Ziel: Konstruiere eine Flussnetzwerk D, sodass ein Fluss f mit $\text{cost}(f) \leq k$ existiert, genau dann wenn es ein 3DM M mit $|M| \geq k$ gibt.

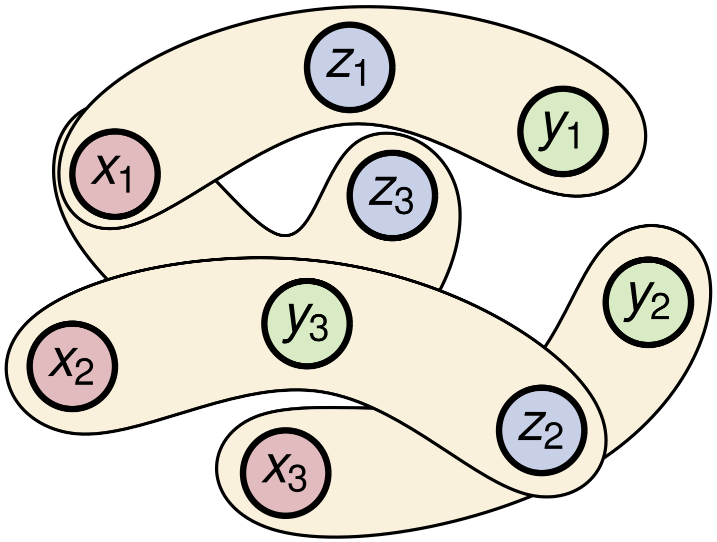


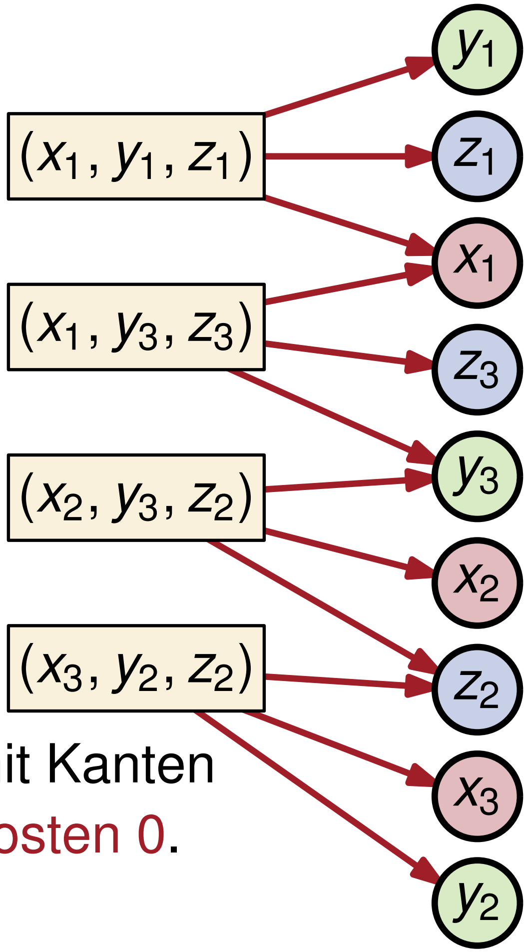


- Erstelle für jedes Element einen Knoten.

Problem 2 (b)

Ziel: Konstruiere eine Flussnetzwerk D, sodass ein Fluss f mit $\mathrm{cost}(f) \leq k$ existiert, genau dann wenn es ein 3DM M mit $|M| \geq k$ gibt.



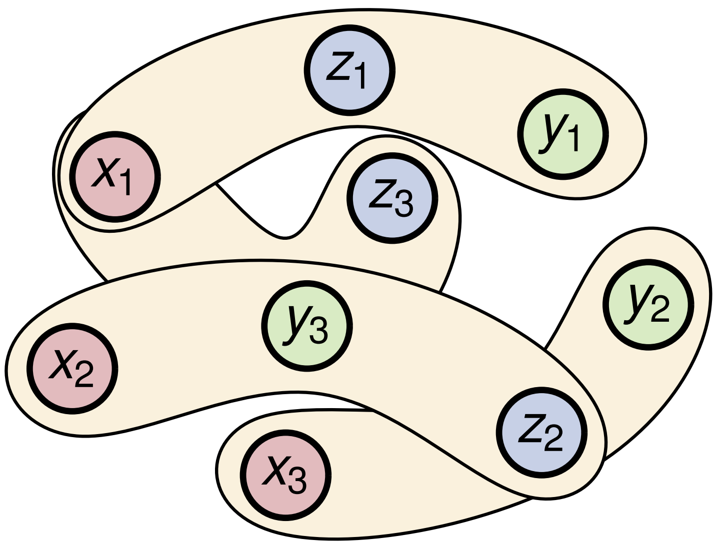


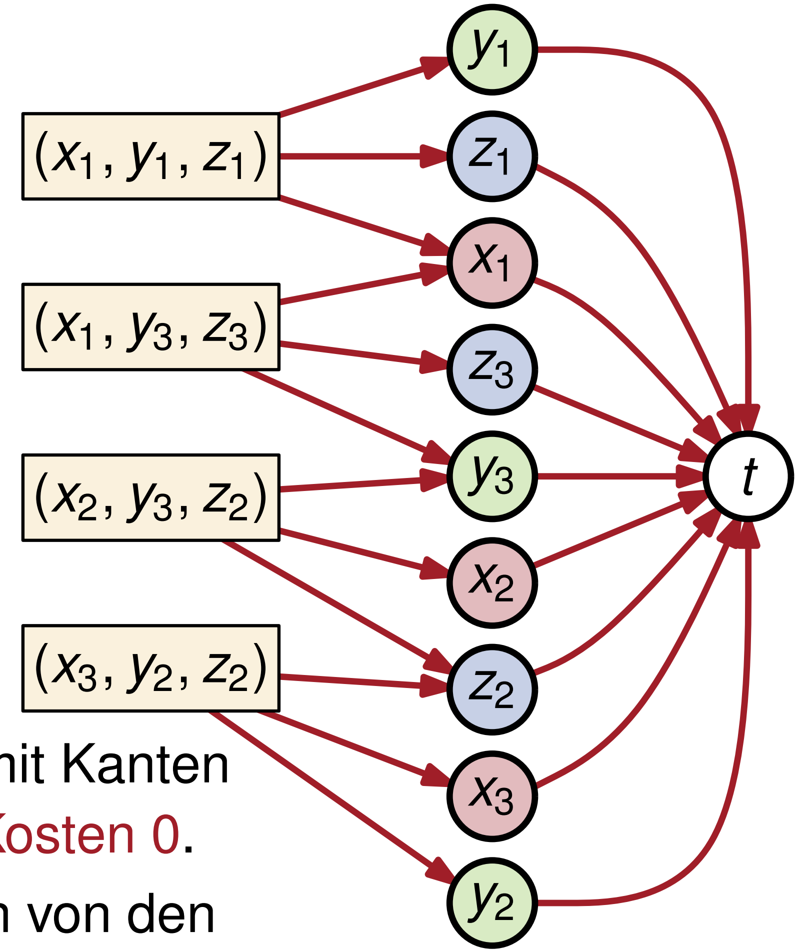
- Erstelle für jedes Element einen Knoten.
- Erstelle für jedes Tripel (x, y, z) einen Knoten mit Kanten zu x, y und z mit Kapazität 1 und konstanten Kosten 0.

Problem 2 (b)

Karlsruher Institut für Technologie

Ziel: Konstruiere eine Flussnetzwerk D, sodass ein Fluss f mit $\text{cost}(f) \leq k$ existiert, genau dann wenn es ein 3DM M mit $|M| \geq k$ gibt.

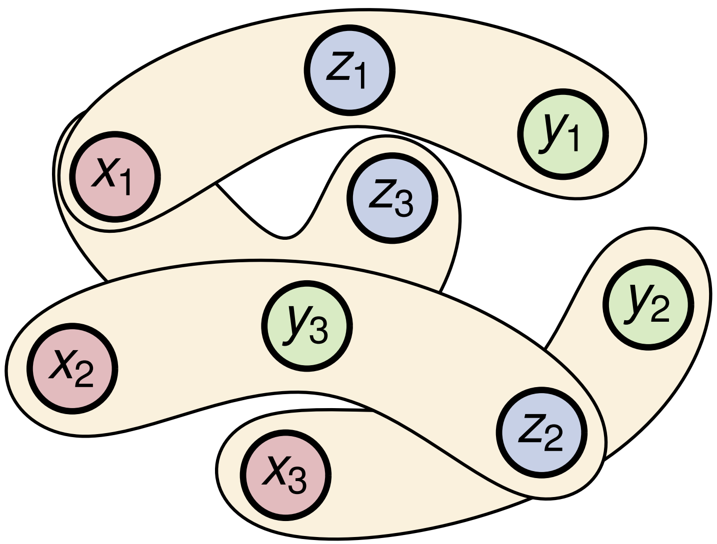


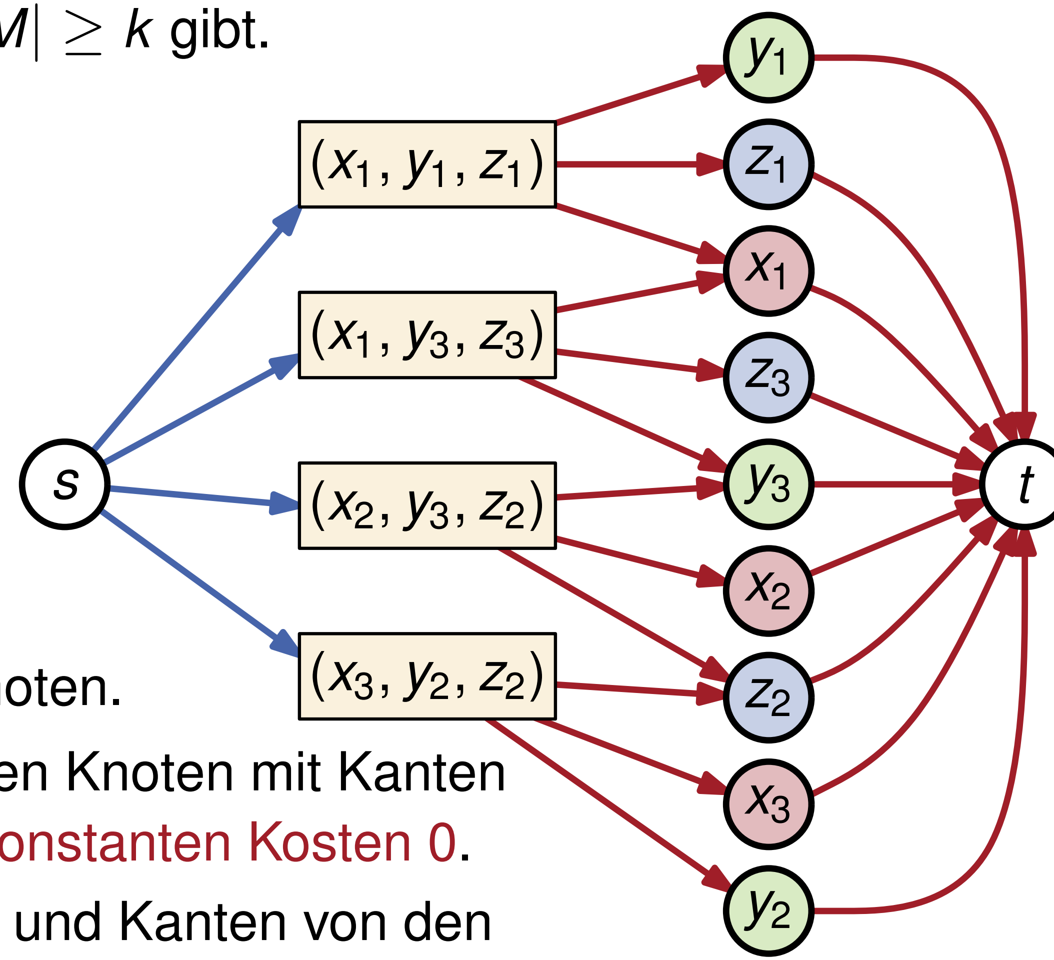


- Erstelle für jedes Element einen Knoten.
- Erstelle für jedes Tripel (x, y, z) einen Knoten mit Kanten zu x, y und z mit Kapazität 1 und konstanten Kosten 0.
- Erstelle eine Senke t mit Bedarf $3k$ und Kanten von den „Elementknoten“ mit Kapazität 1 und konstanten Kosten 0.

Problem 2 (b)

Ziel: Konstruiere eine Flussnetzwerk D, sodass ein Fluss f mit $\text{cost}(f) \leq k$ existiert, genau dann wenn es ein 3DM M mit $|M| \geq k$ gibt.

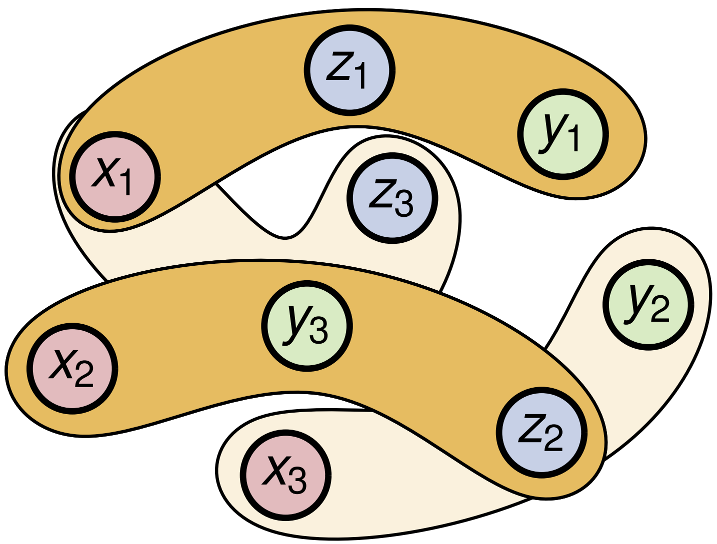


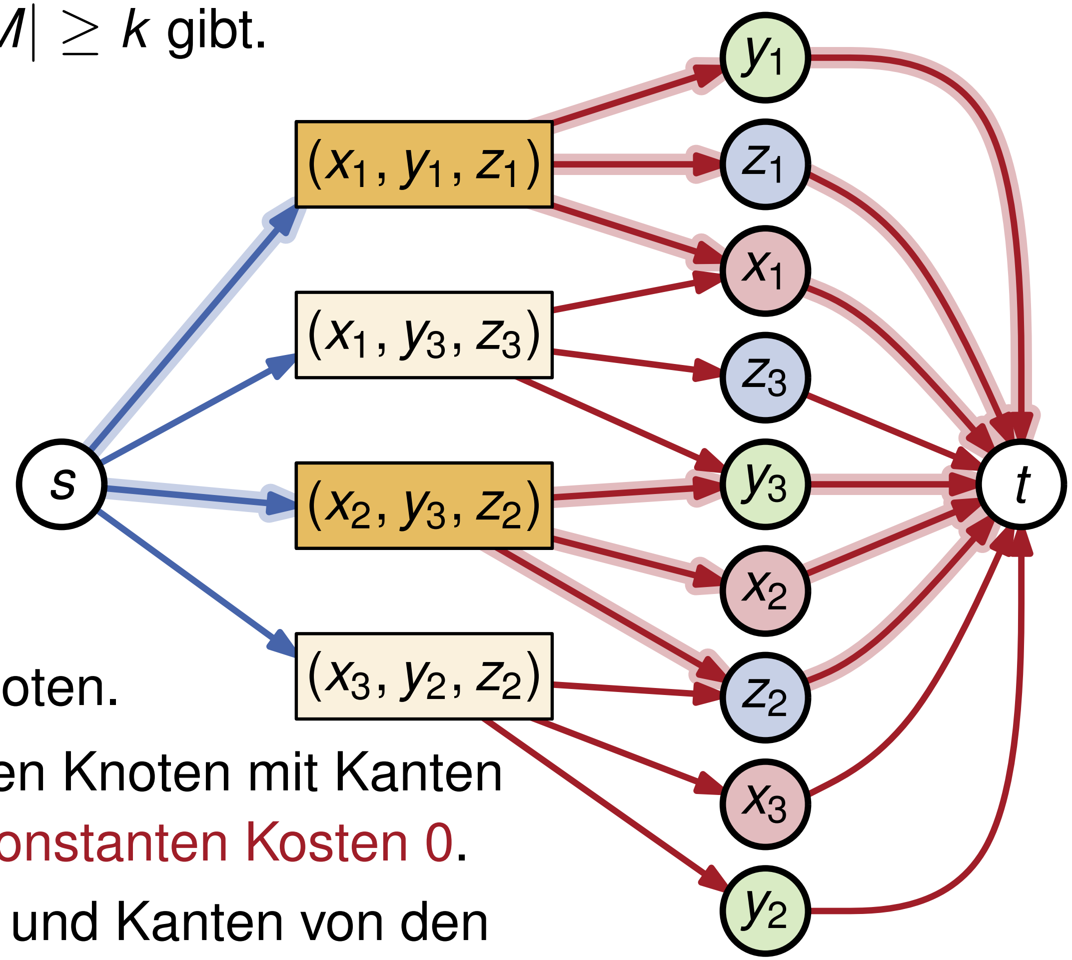


- Erstelle für jedes Element einen Knoten.
- Erstelle für jedes Tripel (x, y, z) einen Knoten mit Kanten zu x, y und z mit Kapazität 1 und konstanten Kosten 0.
- Erstelle eine Senke t mit Bedarf $3k$ und Kanten von den „Elementknoten“ mit Kapazität 1 und konstanten Kosten 0.
- Erstelle Quelle s mit Überschuss $3k$ und Kanten zu den „Tripelknoten“ mit Kapazitäten 3 und Kosten $\text{cost}_e(0) = 0$, $\text{cost}_e(1) = \text{cost}_e(2) = \text{cost}_e(3) = 1$.

Problem 2 (b)

Ziel: Konstruiere eine Flussnetzwerk D, sodass ein Fluss f mit $\text{cost}(f) \leq k$ existiert, genau dann wenn es ein 3DM M mit $|M| \geq k$ gibt.

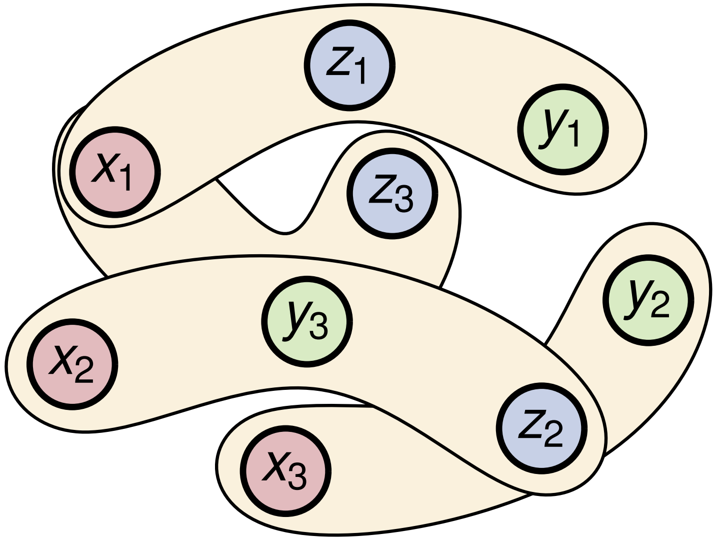




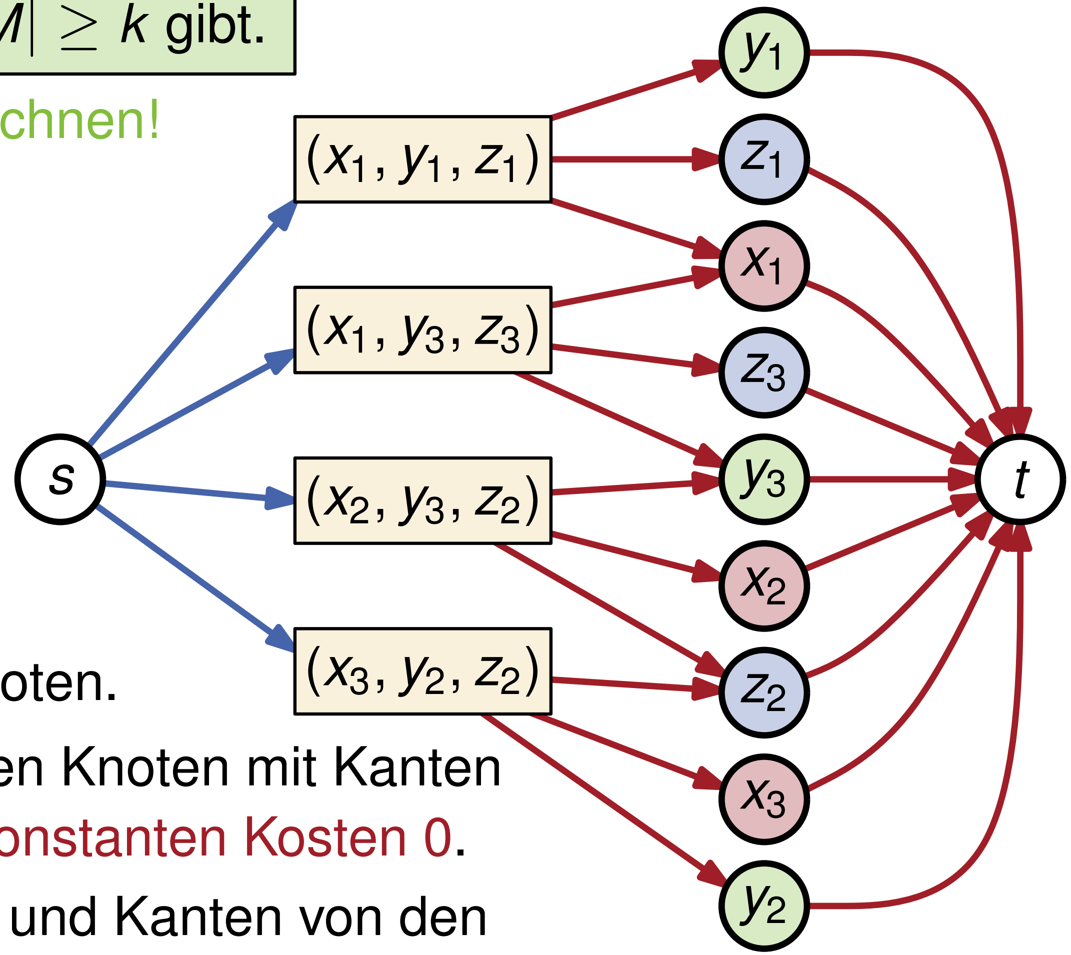
- Erstelle für jedes Element einen Knoten.
- Erstelle für jedes Tripel (x, y, z) einen Knoten mit Kanten zu x, y und z mit Kapazität 1 und konstanten Kosten 0.
- Erstelle eine Senke t mit Bedarf $3k$ und Kanten von den „Elementknoten“ mit Kapazität 1 und konstanten Kosten 0.
- Erstelle Quelle s mit Überschuss $3k$ und Kanten zu den „Tripelknoten“ mit Kapazitäten 3 und Kosten $\text{cost}_e(0) = 0$, $\text{cost}_e(1) = \text{cost}_e(2) = \text{cost}_e(3) = 1$.

Problem 2 (b)

Ziel: Konstruiere eine Flussnetzwerk D, sodass ein Fluss f mit $\text{cost}(f) \leq k$ existiert, genau dann wenn es ein 3DM M mit $|M| \geq k$ gibt.



Nachrechnen!



- Erstelle für jedes Element einen Knoten.
- Erstelle für jedes Tripel (x, y, z) einen Knoten mit Kanten zu x, y und z mit Kapazität 1 und konstanten Kosten 0.
- Erstelle eine Senke t mit Bedarf $3k$ und Kanten von den „Elementknoten“ mit Kapazität 1 und konstanten Kosten 0.
- Erstelle Quelle s mit Überschuss $3k$ und Kanten zu den „Tripelknoten“ mit Kapazitäten 3 und Kosten $\text{cost}_e(0) = 0$, $\text{cost}_e(1) = \text{cost}_e(2) = \text{cost}_e(3) = 1$.

Flüsse mit Mindestfluss

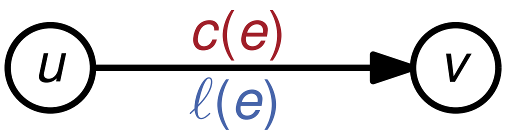
Problem 3

Sei $D = (V, E)$ ein Flussnetzwerk wobei zusätzlich auf jeder Kante ein *Mindestfluss* gefordert ist. Das heißt, gegeben ist eine Abbildung $\ell\colon E \to \mathbb{R}_0^+$ und ein Fluss f muss zusätzlich zur Kapazitäts- und Flusserhaltungsbedingung die Ungleichung $\ell(e) \leq f(e)$ (für alle $e \in E$) erfüllen.
Zeigen Sie, dass ein Flussproblem mit gefordertem Mindestfluss in ein Flussproblem ohne diese zusätzliche Bedingung transformiert werden kann.

Problem 3

Sei $D = (V, E)$ ein Flussnetzwerk wobei zusätzlich auf jeder Kante ein *Mindestfluss* gefordert ist. Das heißt, gegeben ist eine Abbildung $\ell\colon E \to \mathbb{R}_0^+$ und ein Fluss f muss zusätzlich zur Kapazitäts- und Flusserhaltungsbedingung die Ungleichung $\ell(e) \leq f(e)$ (für alle $e \in E$) erfüllen.
Zeigen Sie, dass ein Flussproblem mit gefordertem Mindestfluss in ein Flussproblem ohne diese zusätzliche Bedingung transformiert werden kann.

Betrachte eine einzelne Kanten $e(u, v)$ mit Mindestfluss $\ell(e)$ und Kapazität $c(e)$.

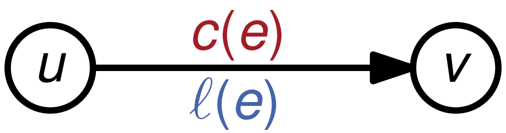


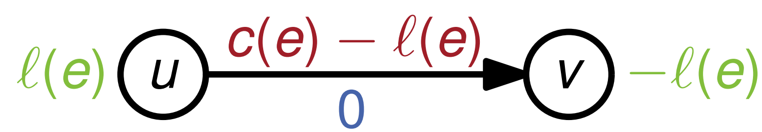
Problem 3

Karlsruher Institut für Technologie

Sei $D = (V, E)$ ein Flussnetzwerk wobei zusätzlich auf jeder Kante ein *Mindestfluss* gefordert ist. Das heißt, gegeben ist eine Abbildung $\ell \colon E \to \mathbb{R}_0^+$ und ein Fluss f muss zusätzlich zur Kapazitäts- und Flusserhaltungsbedingung die Ungleichung $\ell(e) \leq f(e)$ (für alle $e \in E$) erfüllen.
Zeigen Sie, dass ein Flussproblem mit gefordertem Mindestfluss in ein Flussproblem ohne diese zusätzliche Bedingung transformiert werden kann.

Betrachte eine einzelne Kanten $e(u, v)$ mit Mindestfluss $\ell(e)$ und Kapazität $c(e)$.





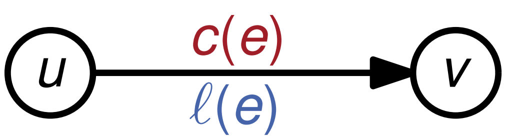
Konstruktion des Flussnetzwerks D':

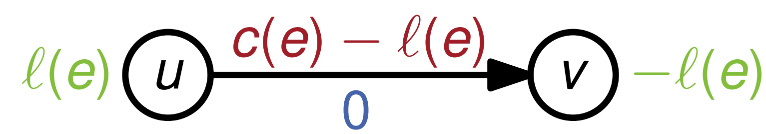
- Setze den Mindestfluss von e auf 0 und die Kapazität von e auf $c(e) - \ell(e)$.
- die Bedarfe von u und v auf $b(u) = \ell(e)$ bzw. $b(v) = -\ell(e)$

Problem 3

Sei $D = (V, E)$ ein Flussnetzwerk wobei zusätzlich auf jeder Kante ein *Mindestfluss* gefordert ist. Das heißt, gegeben ist eine Abbildung $\ell\colon E \to \mathbb{R}_0^+$ und ein Fluss f muss zusätzlich zur Kapazitäts- und Flusserhaltungsbedingung die Ungleichung $\ell(e) \leq f(e)$ (für alle $e \in E$) erfüllen.
Zeigen Sie, dass ein Flussproblem mit gefordertem Mindestfluss in ein Flussproblem ohne diese zusätzliche Bedingung transformiert werden kann.

Betrachte eine einzelne Kanten $e(u, v)$ mit Mindestfluss $\ell(e)$ und Kapazität $c(e)$.





Konstruktion des Flussnetzwerks D':

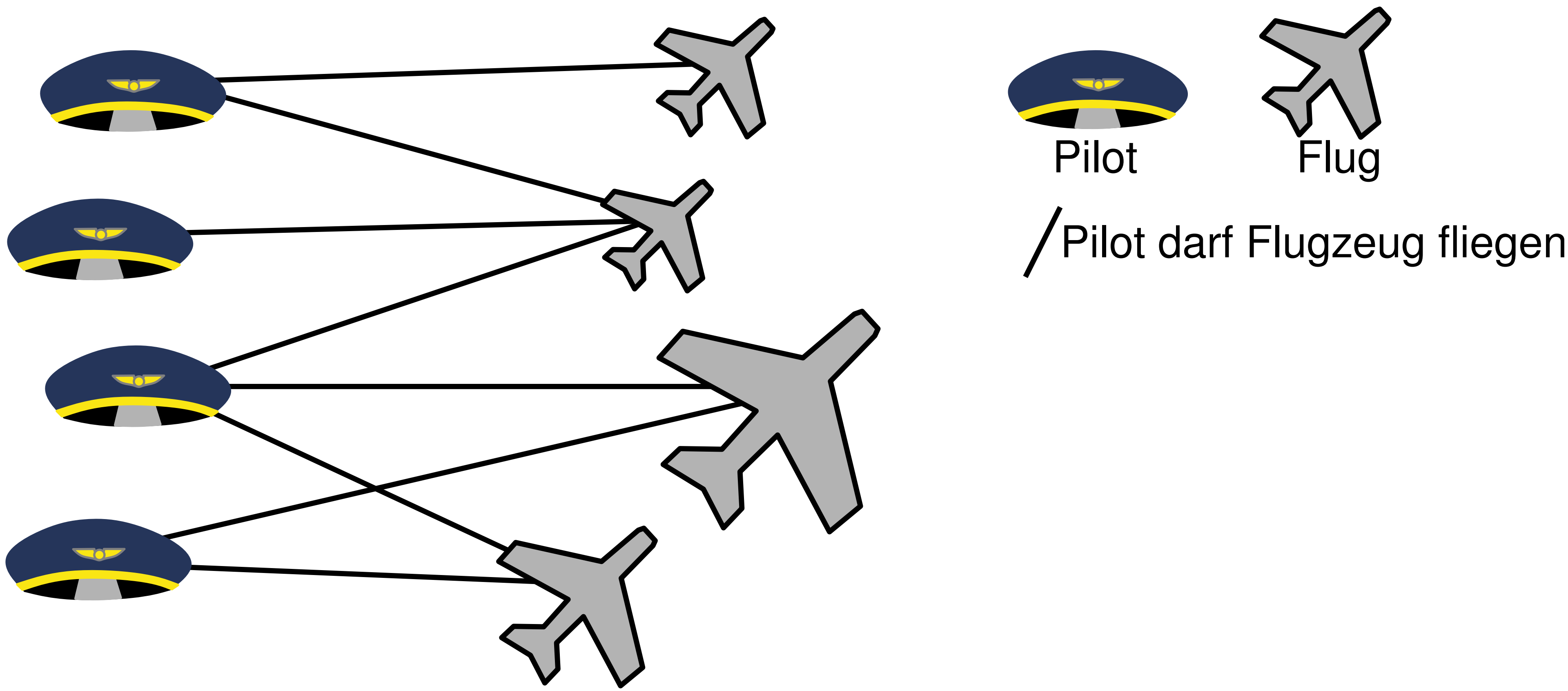
- Setze den Mindestfluss von e auf 0 und die Kapazität von e auf $c(e) - \ell(e)$.
- die Bedarfe von u und v auf $b(u) = \ell(e)$ bzw. $b(v) = -\ell(e)$

Zeige: Gültiger Fluss f in D liefert gültigen Fluss f' in D' und umgekehrt, wobei $f(e) = f'(e) + \ell(e)$.

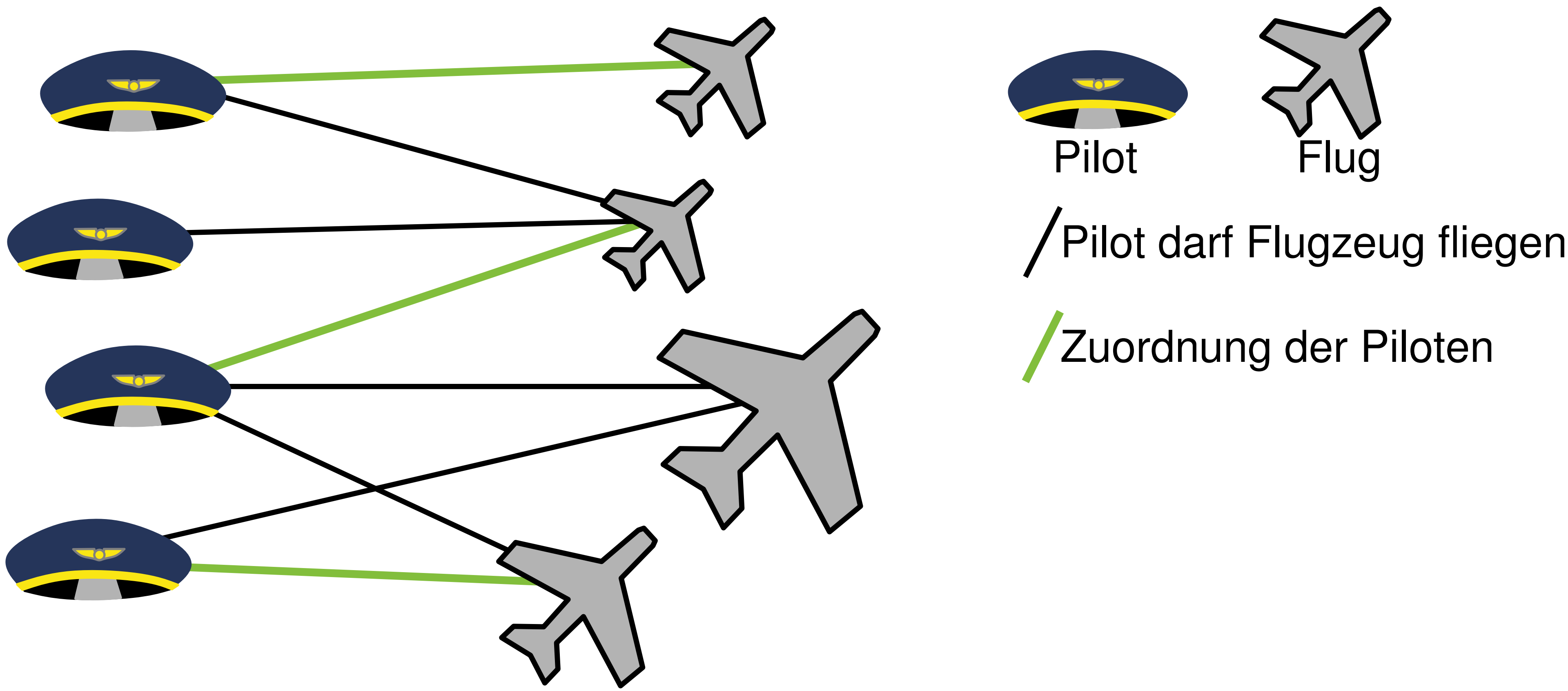
Matchings – Erhöhende Wege

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

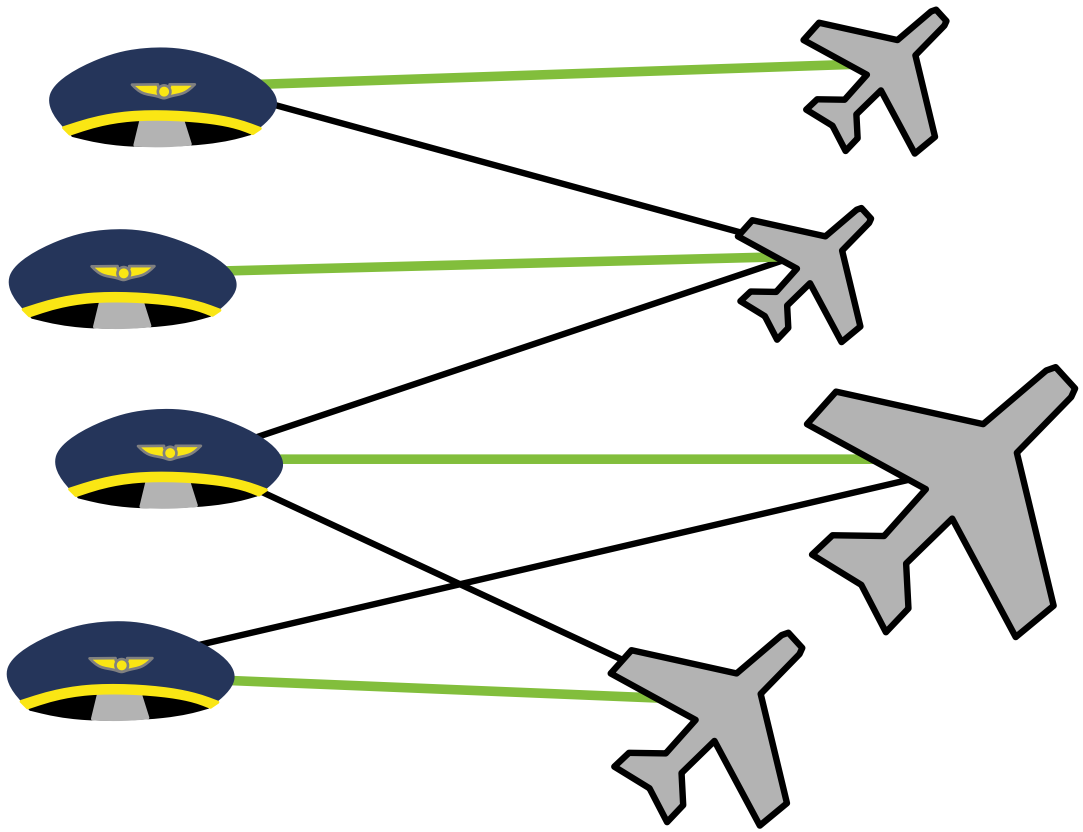
Wiederholung: Motivation



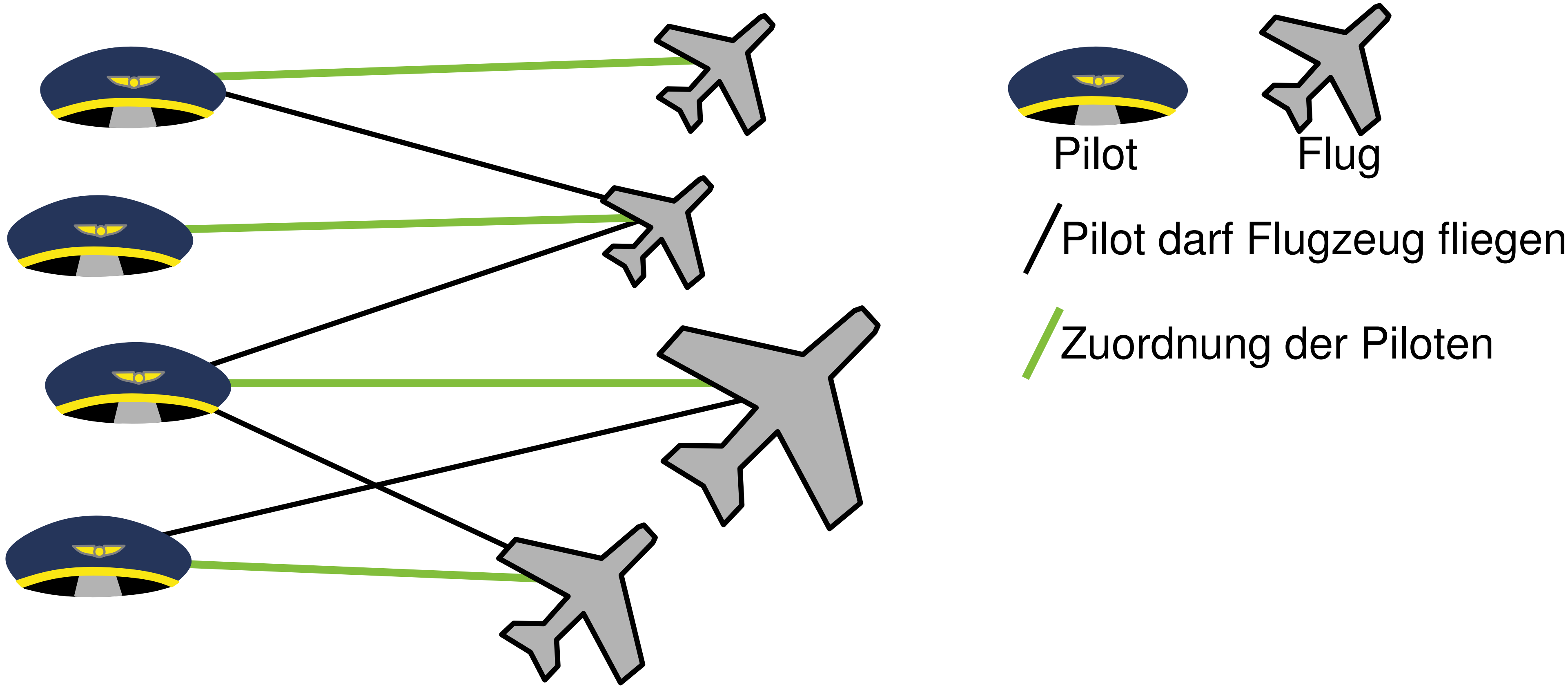
Wiederholung: Motivation



Wiederholung: Motivation



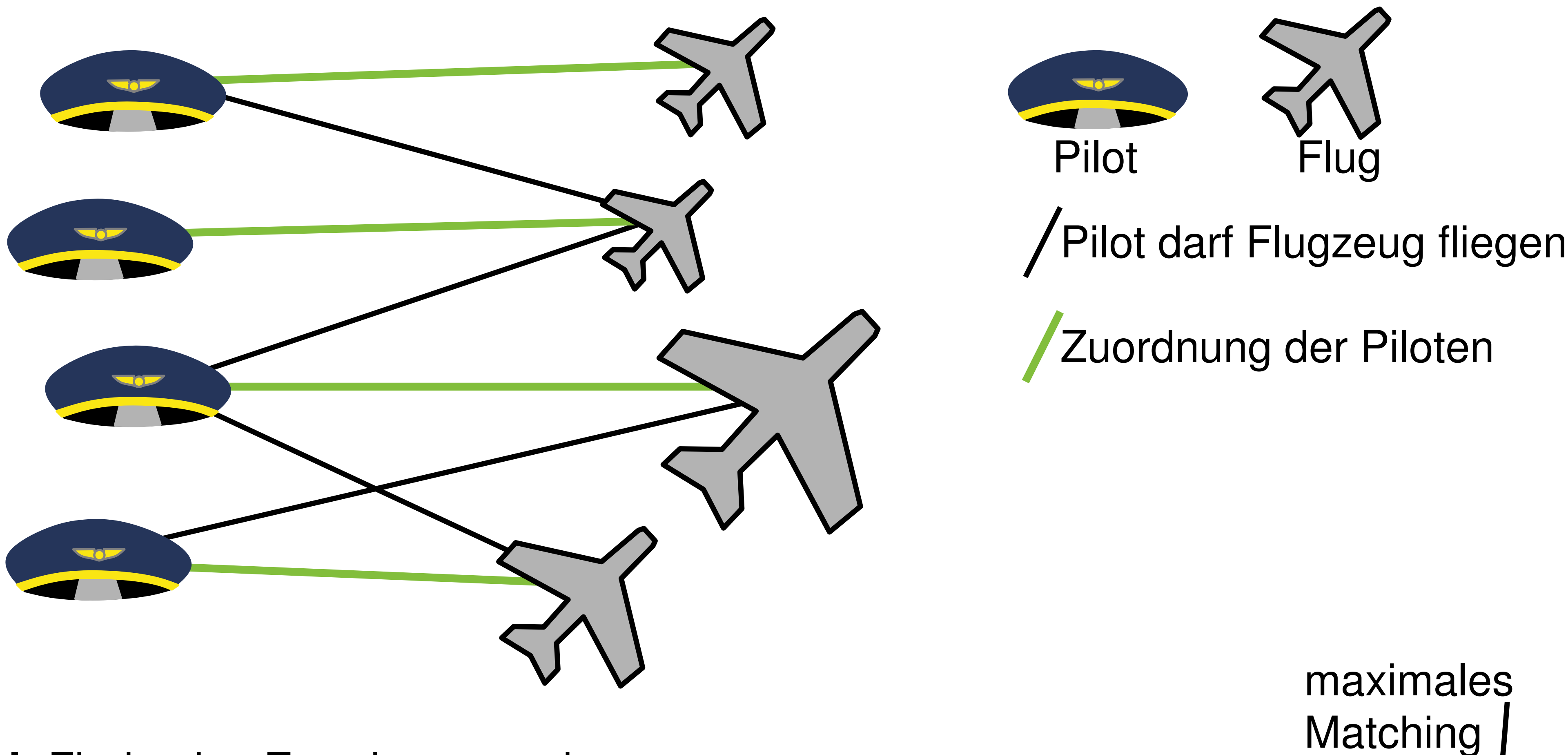
Wiederholung: Motivation



Ziel: Finde eine Zuordnung, sodass:

- Jeder Pilot maximal ein Flugzeug fliegt, jedes Flugzeug von maximal einem Pilot geflogen wird.
- Möglichst viele Flugzeuge besetzt (bzw. Piloten beschäftigt) sind.

Wiederholung: Motivation

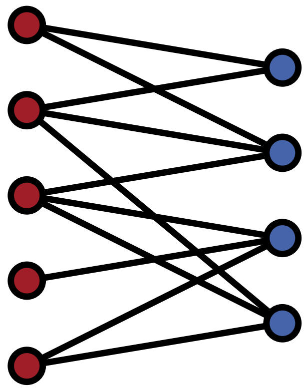


Ziel: Finde eine Zuordnung, sodass:

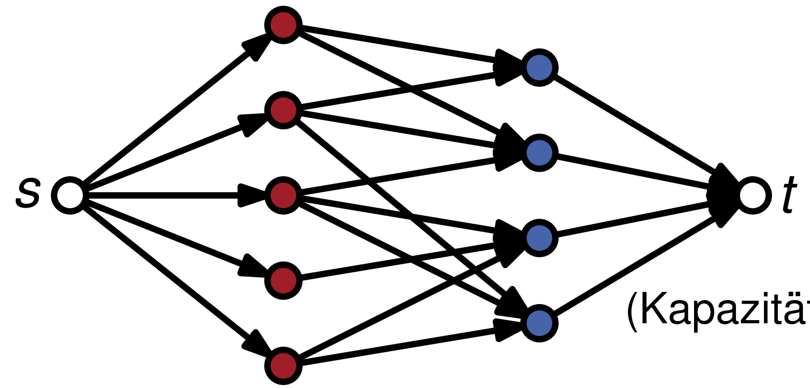
- Jeder Pilot maximal ein Flugzeug fliegt, jedes Flugzeug von maximal einem Pilot geflogen wird.
- Möglichst viele Flugzeuge besetzt (bzw. Piloten beschäftigt) sind.

Berechnung mithilfe eines Flusses

bipartiter Graph G



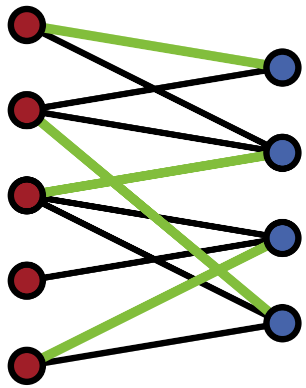
zugehöriges Flussnetzwerk G'



(Kapazitäten sind 1)

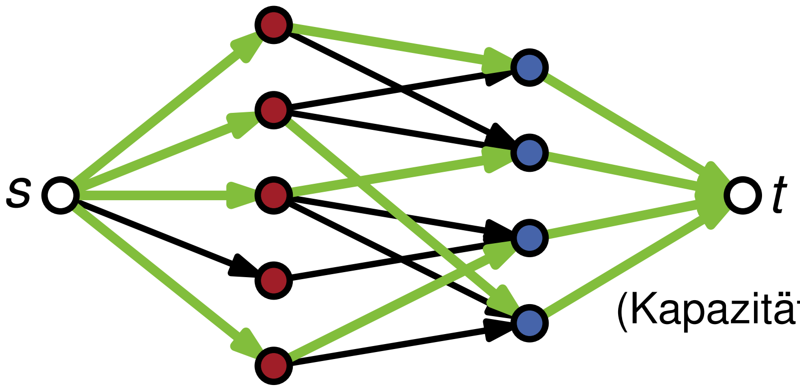
Berechnung mithilfe eines Flusses

bipartiter Graph G



mit maximalem Matching

zugehöriges Flussnetzwerk G'

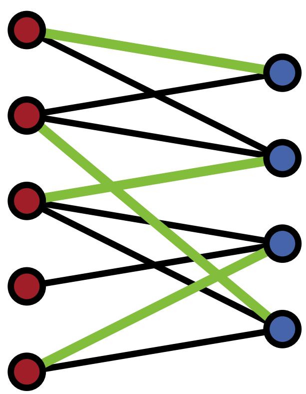


(Kapazitäten sind 1)

mit maximalem Fluss

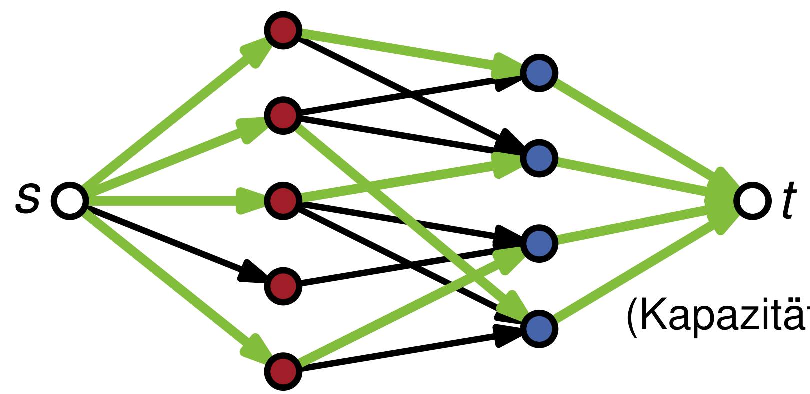
Berechnung mithilfe eines Flusses

bipartiter Graph G



mit maximalem Matching

zugehöriges Flussnetzwerk G'



(Kapazitäten sind 1)

mit maximalem Fluss

Problem 4

Der maximale Fluss kann schrittweise mithilfe von erhöhenden Wegen berechnet werden (Ford-Fulkerson-Algorithmus). In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching.

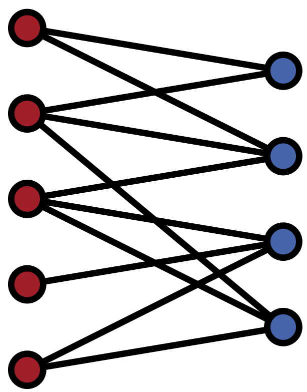
Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

Problem 4

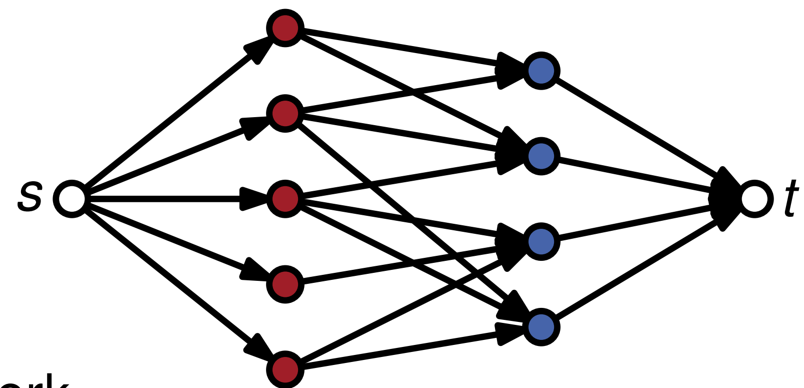
Der maximale Fluss kann schrittweise mithilfe von erhöhenden Wegen berechnet werden (Ford-Fulkerson-Algorithmus). In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching.

Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

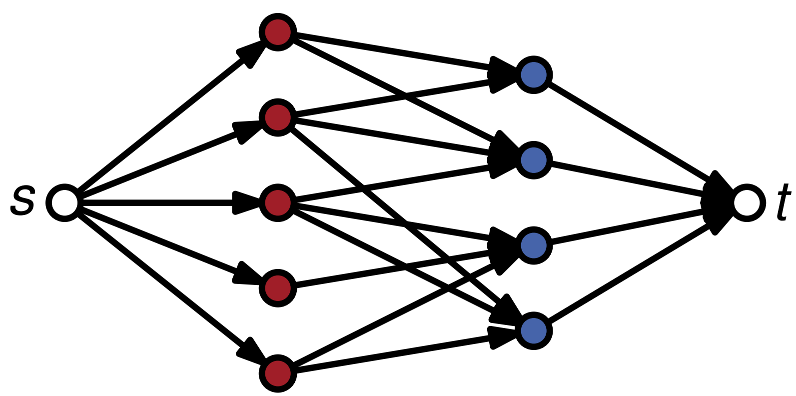
bipartiter Graph G



zugehöriges Flussnetzwerk G'



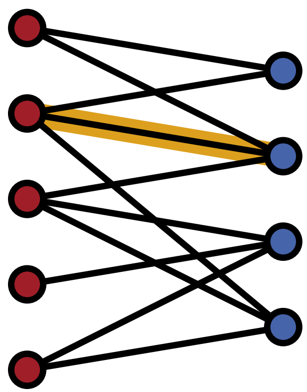
Residualnetzwerk



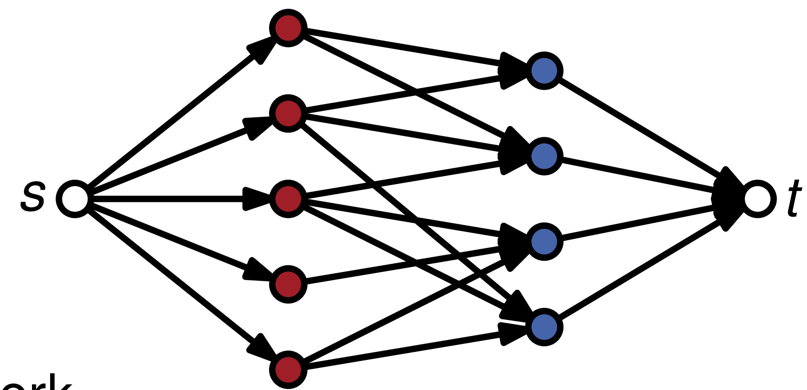
Problem 4

Der maximale Fluss kann schrittweise mithilfe von erhöhenden Wegen berechnet werden (Ford-Fulkerson-Algorithmus). In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching.
Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

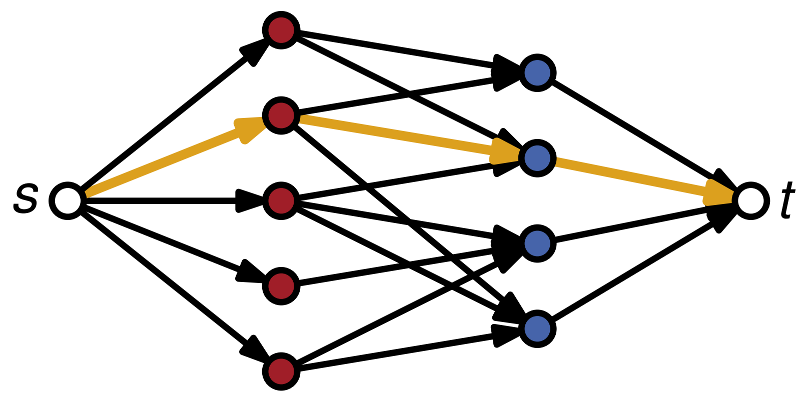
bipartiter Graph G



zugehöriges Flussnetzwerk G'



Residualnetzwerk

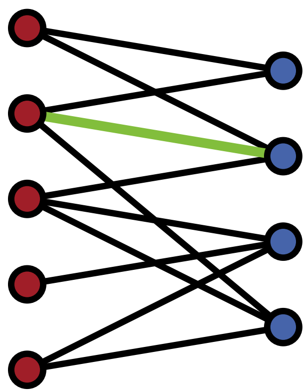


Problem 4

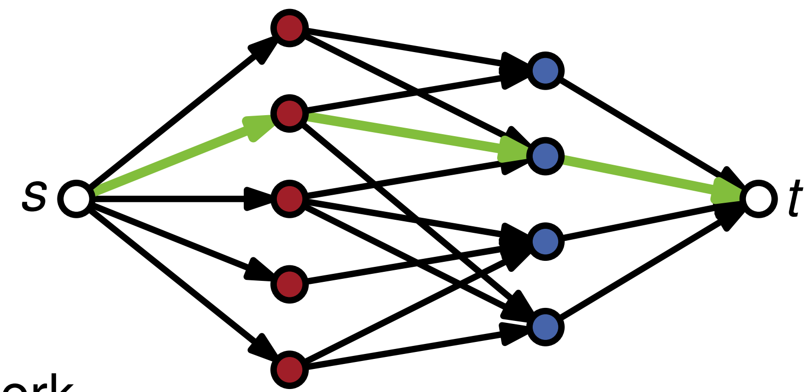
Der maximale Fluss kann schrittweise mithilfe von erhöhenden Wegen berechnet werden (Ford-Fulkerson-Algorithmus). In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching.

Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

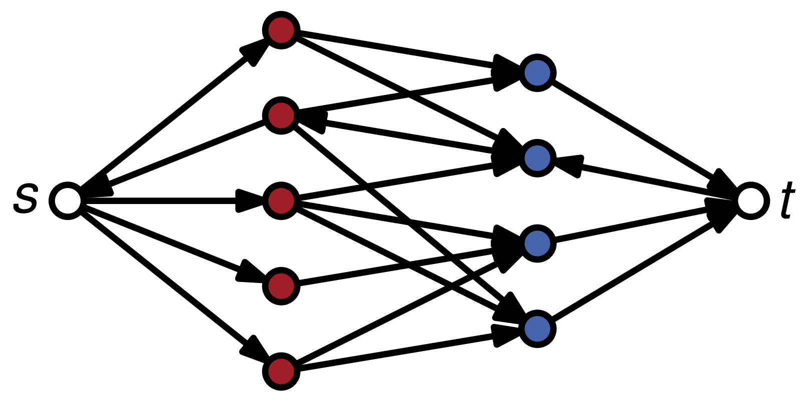
bipartiter Graph G



zugehöriges Flussnetzwerk G'



Residualnetzwerk

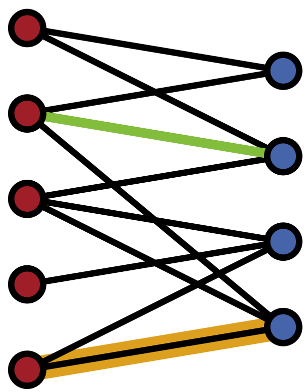


Problem 4

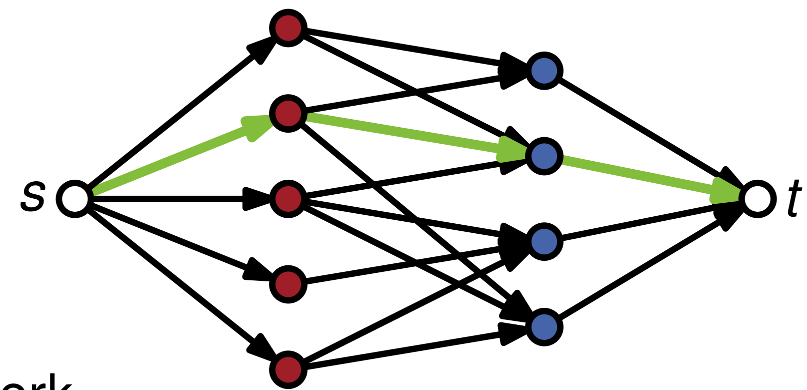
Der maximale Fluss kann schrittweise mithilfe von erhöhenden Wegen berechnet werden (Ford-Fulkerson-Algorithmus). In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching.

Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

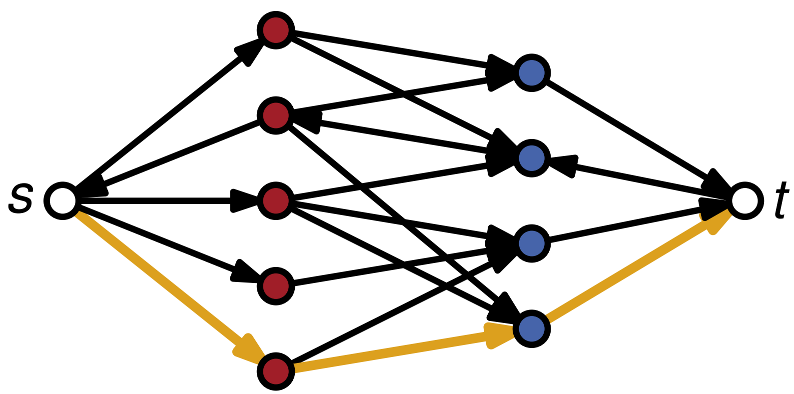
bipartiter Graph G



zugehöriges Flussnetzwerk G'



Residualnetzwerk

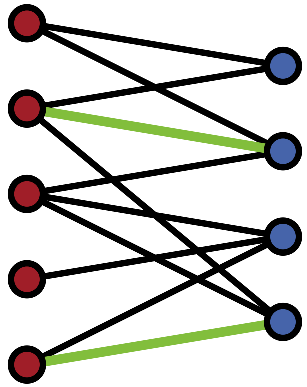


Problem 4

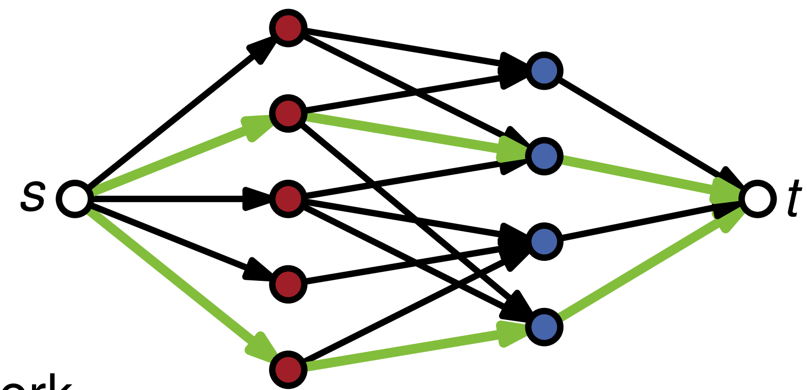
Der maximale Fluss kann schrittweise mithilfe von erhöhenden Wegen berechnet werden (Ford-Fulkerson-Algorithmus). In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching.

Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

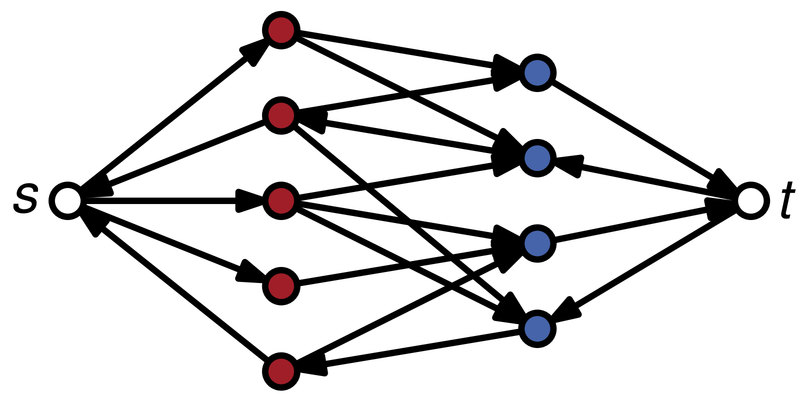
bipartiter Graph G



zugehöriges Flussnetzwerk G'



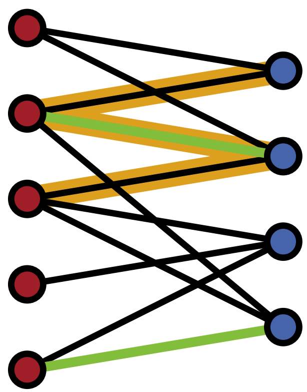
Residualnetzwerk



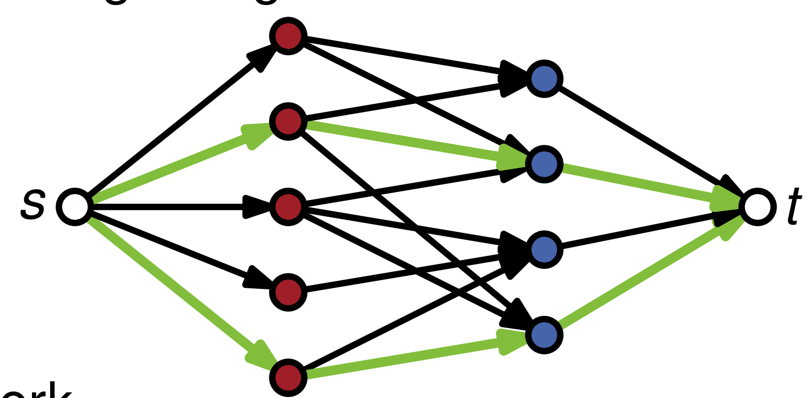
Problem 4

Der maximale Fluss kann schrittweise mithilfe von erhöhenden Wegen berechnet werden (Ford-Fulkerson-Algorithmus). In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching.
Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

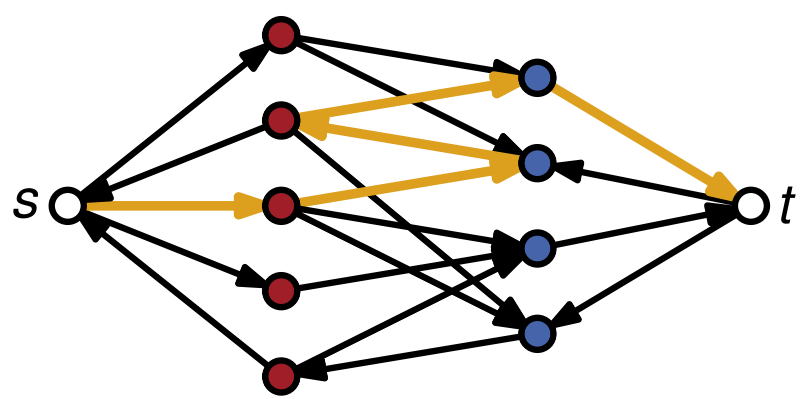
bipartiter Graph G



zugehöriges Flussnetzwerk G'



Residualnetzwerk

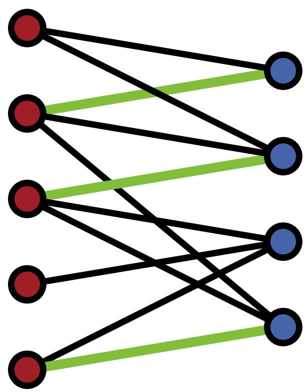


Problem 4

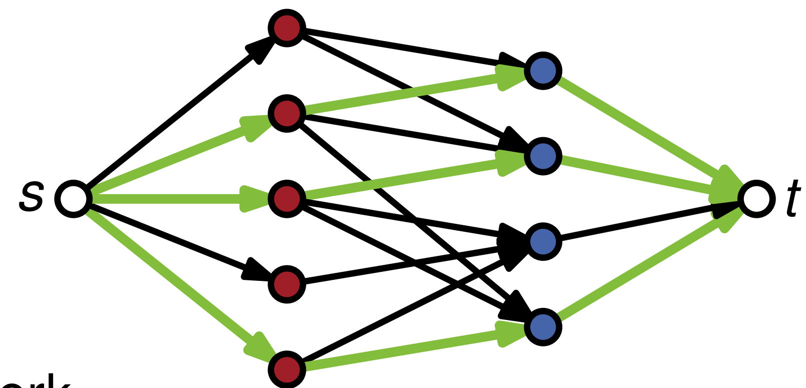
Der maximale Fluss kann schrittweise mithilfe von erhöhenden Wegen berechnet werden (Ford-Fulkerson-Algorithmus). In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching.

Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

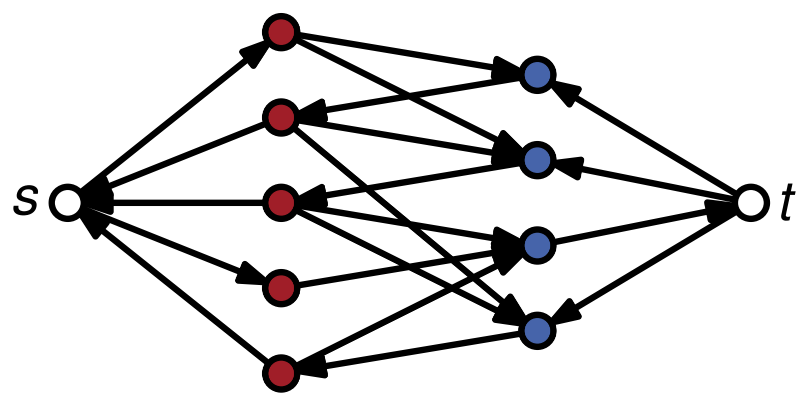
bipartiter Graph G



zugehöriges Flussnetzwerk G'



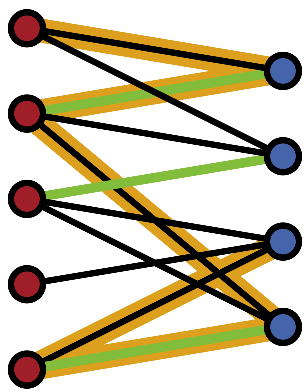
Residualnetzwerk



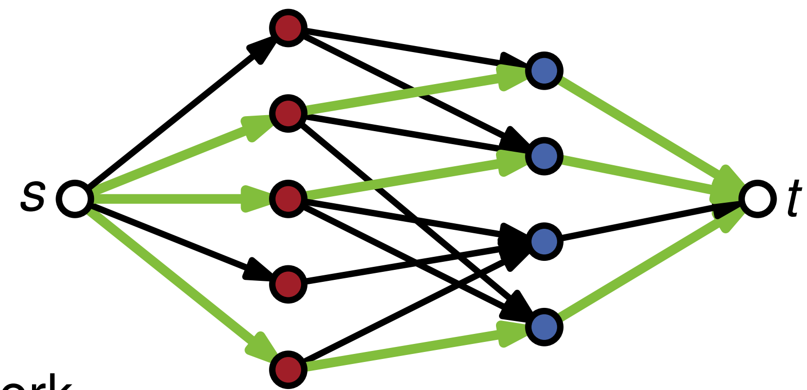
Problem 4

Der maximale Fluss kann schrittweise mithilfe von erhöhenden Wegen berechnet werden (Ford-Fulkerson-Algorithmus). In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching.
Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

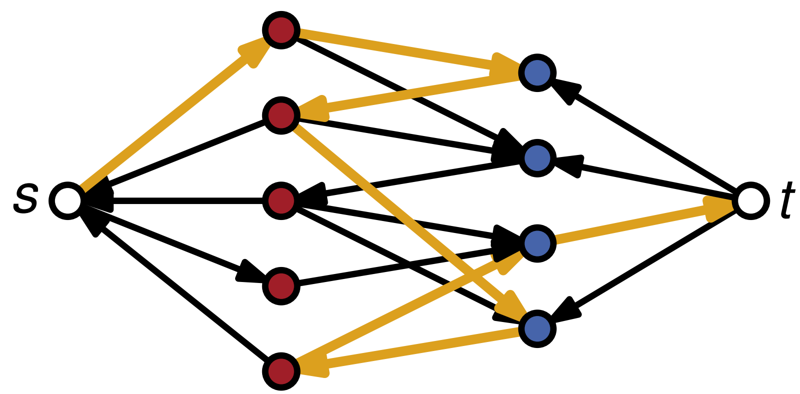
bipartiter Graph G



zugehöriges Flussnetzwerk G'



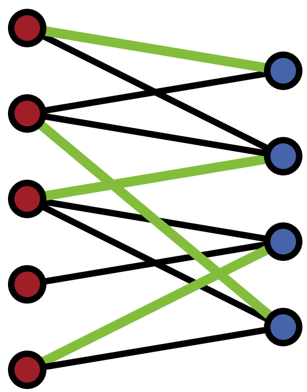
Residualnetzwerk



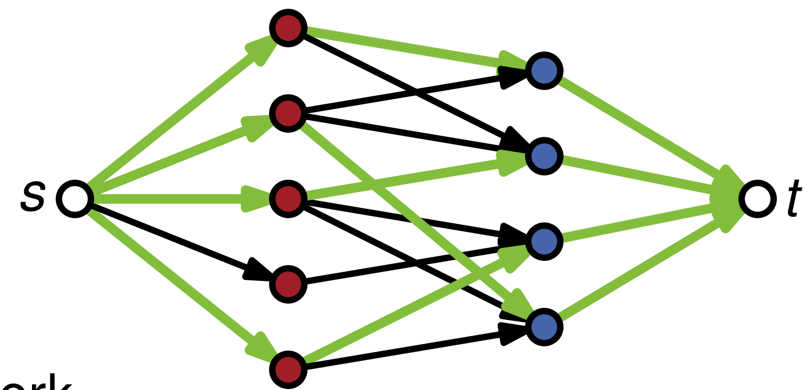
Problem 4

Der maximale Fluss kann schrittweise mithilfe von erhöhenden Wegen berechnet werden (Ford-Fulkerson-Algorithmus). In jedem dieser Schritte induziert der aktuelle Fluss f ein (nicht notwendigerweise maximales) Matching.
Erklären Sie inwiefern dieses Matching sich im Verlauf der Berechnung von f ändert. Gehen Sie hierzu insbesondere auf die Rolle der erhöhenden Wege ein.

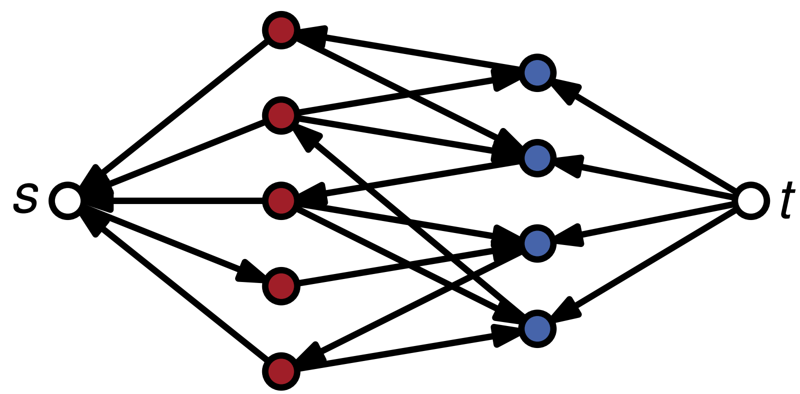
bipartiter Graph G



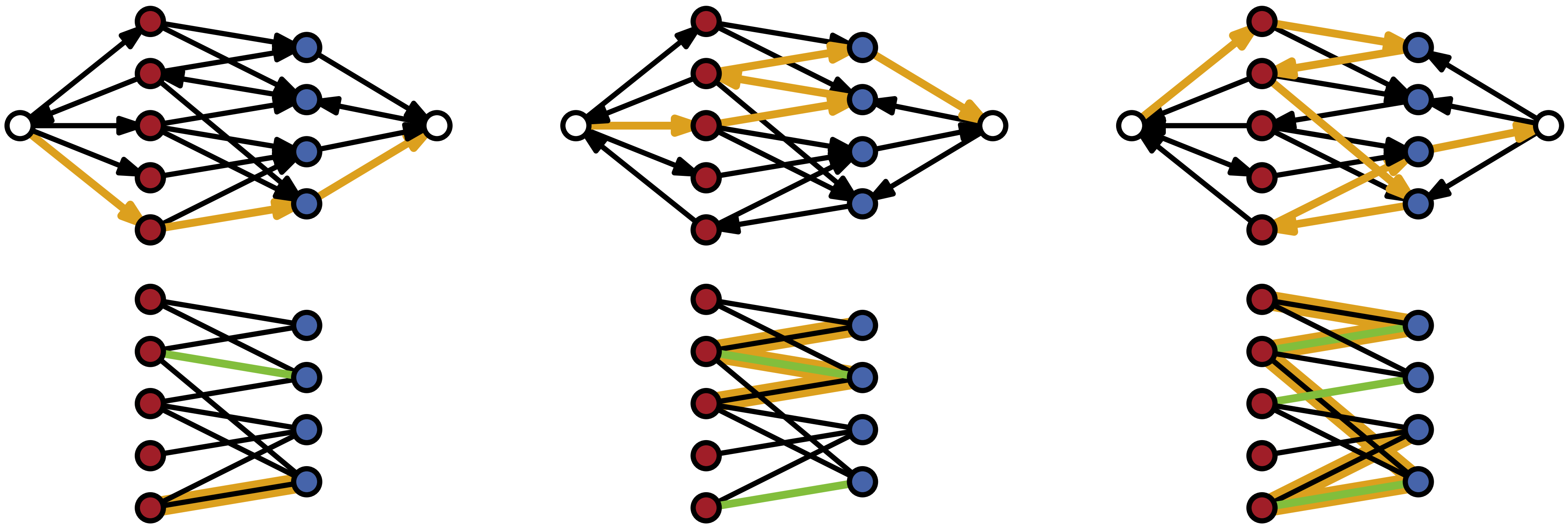
zugehöriges Flussnetzwerk G'



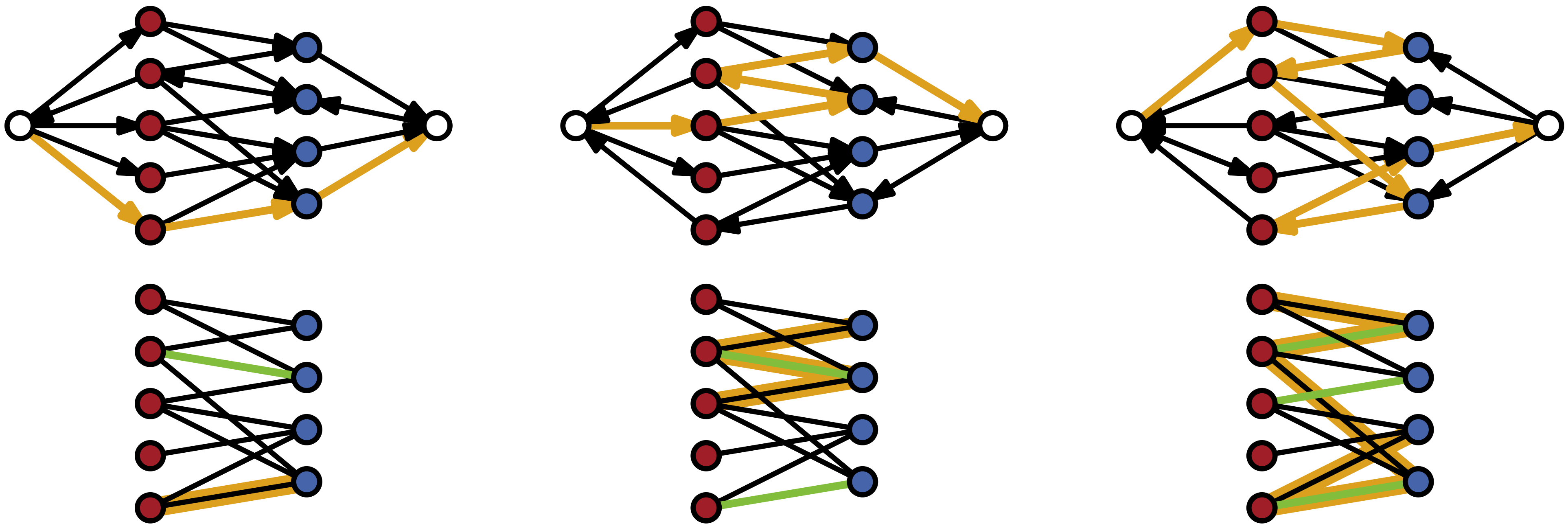
Residualnetzwerk



Problem 4

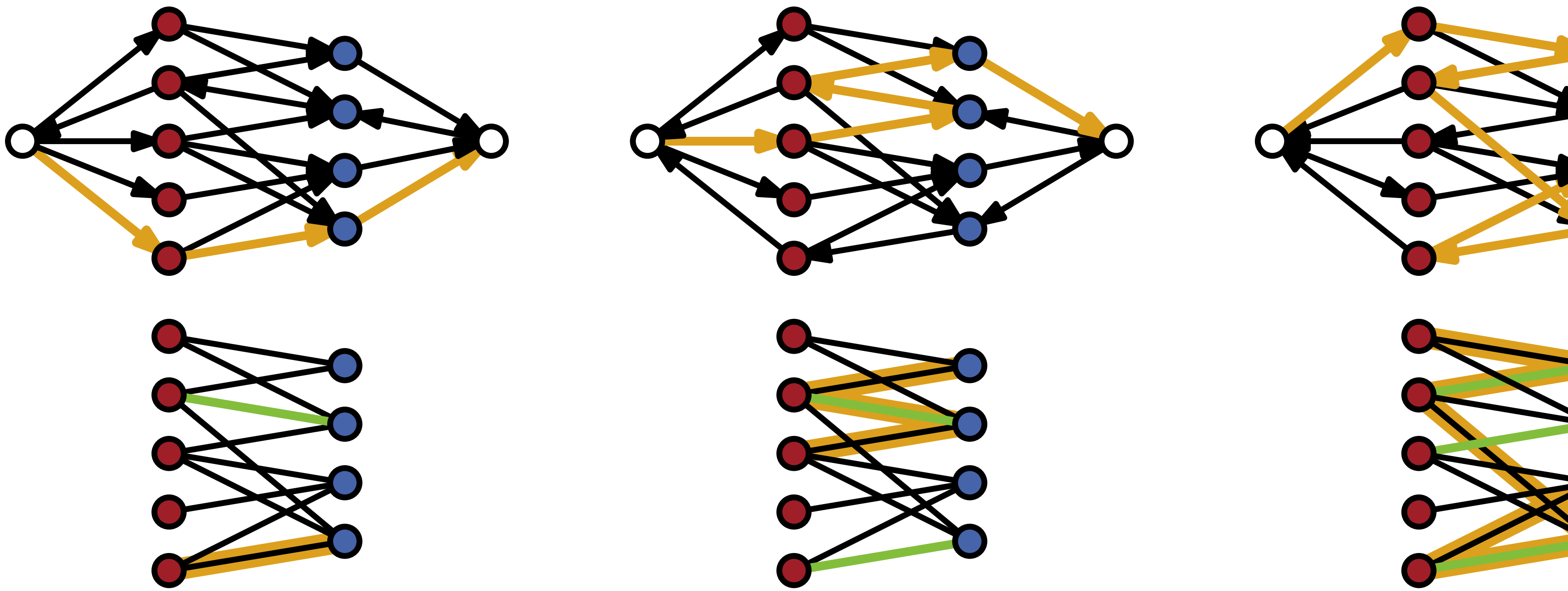


Problem 4



Ein erhöhender Pfad im Residualnetzwerk entspricht einem Pfad Π im Graph, sodass …

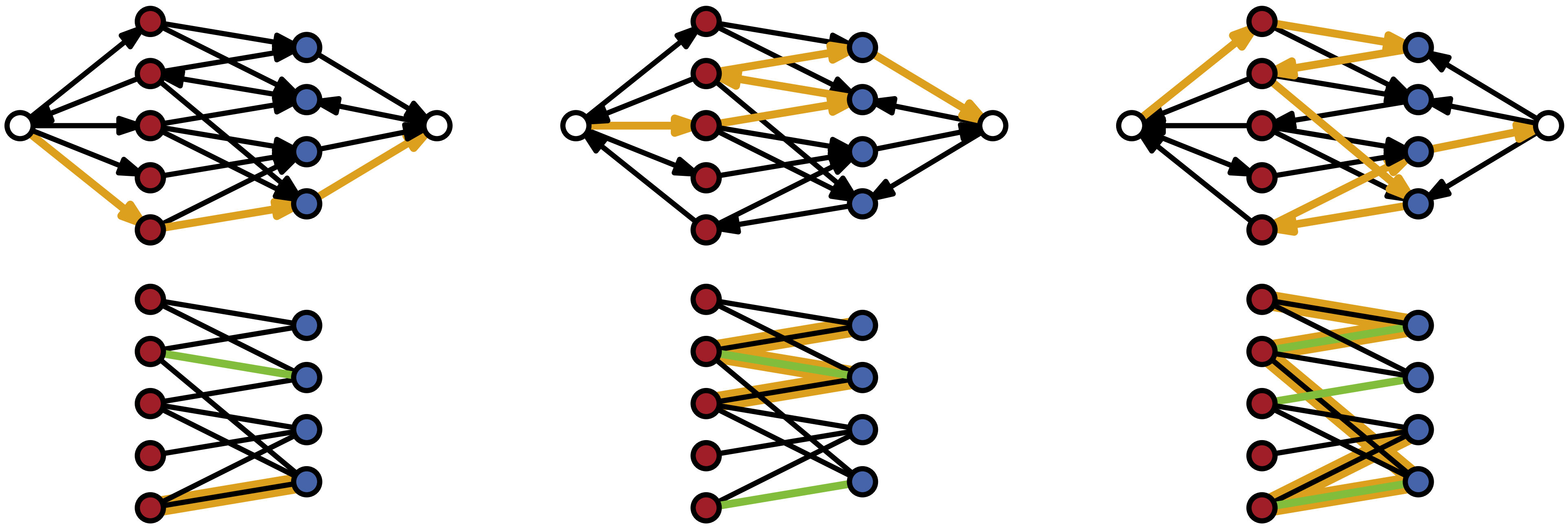
Problem 4



Ein erhöhender Pfad im Residualnetzwerk entspricht einem Pfad Π im Graph, sodass ...

- der erste und letzte Knoten in Π nicht gematched sind (d.h. sie sind zu keiner Matchingkante inzident)
- und Π aus abwechselnd ungematchten und gematchten Kanten besteht.

Problem 4



Ein erhöhender Pfad im Residualnetzwerk entspricht einem Pfad Π im Graph, sodass ...

- der erste und letzte Knoten in Π nicht gematched sind (d.h. sie sind zu keiner Matchingkante inzident)
- und Π aus abwechselnd ungematchten und gematchten Kanten besteht.

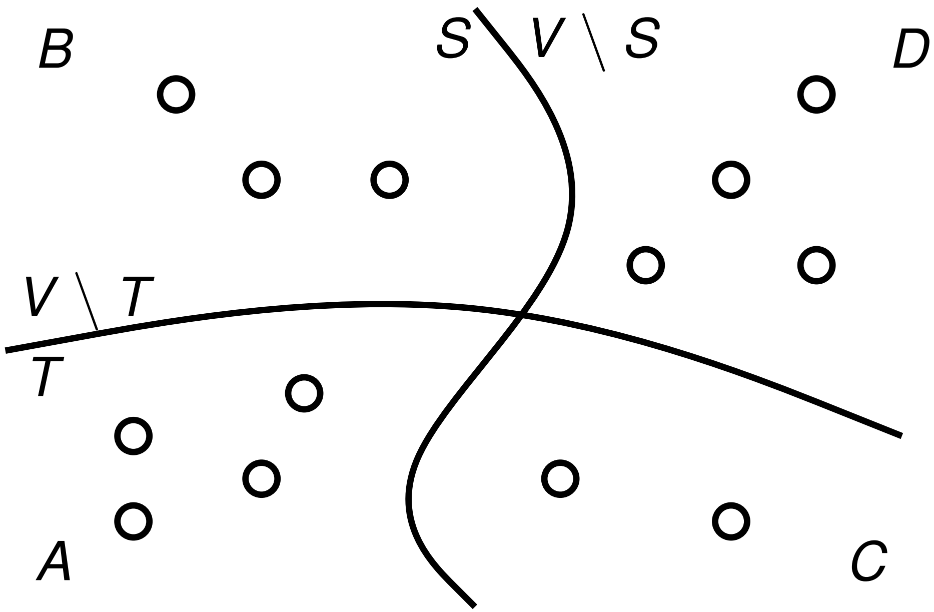
Man könnte auch iterativ nach solchen Pfaden suchen, statt nach erhöhenden Pfaden im Residualnetzwerk.

Schnitte in Graphen

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

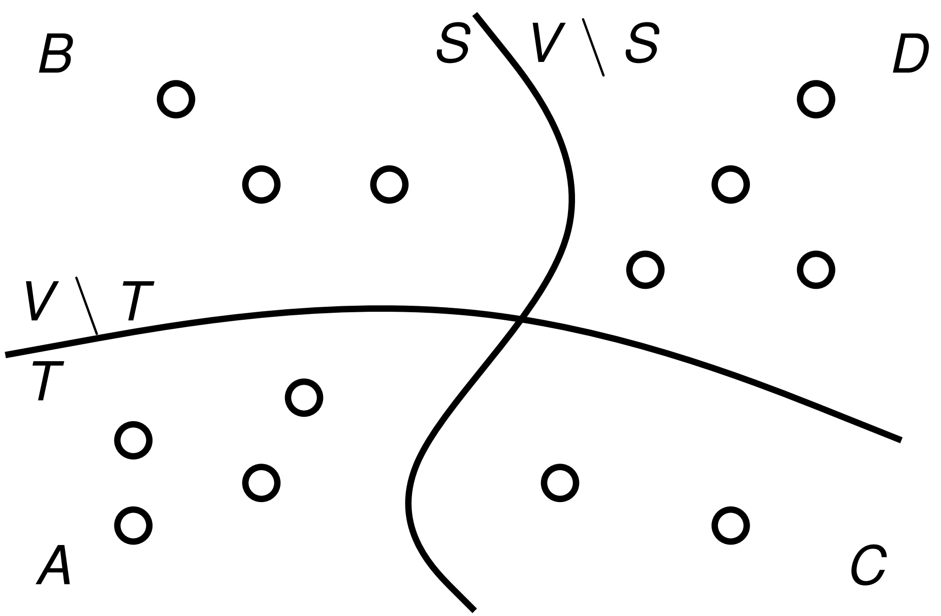
(a) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: Es gilt $s \in B$ und $t \in C$.



Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(a) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: Es gilt $s \in B$ und $t \in C$.



s liegt in $S = A \cup B$ und t in $T = A \cup C$

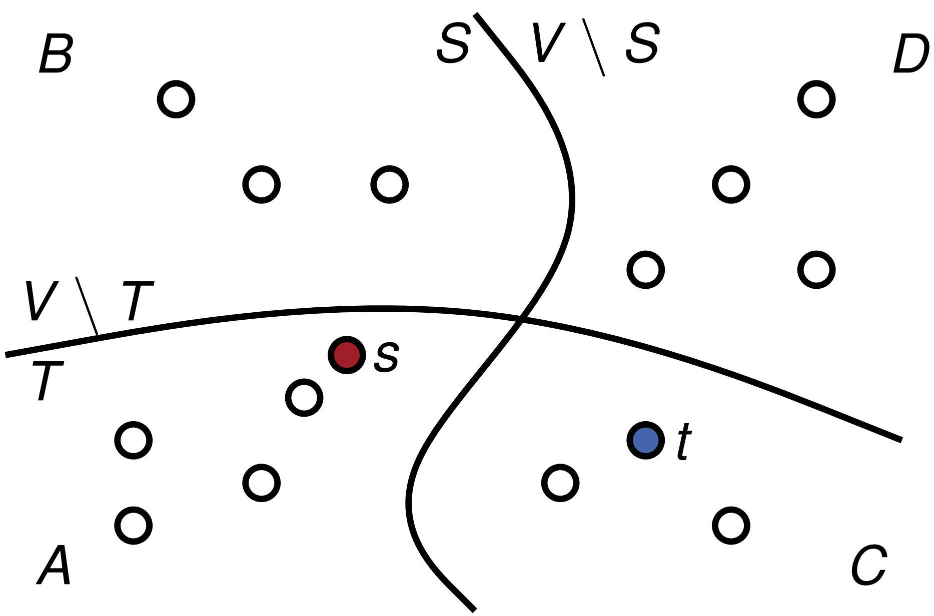
$\Rightarrow$ Es gibt vier mögliche Kombinationen:

- $s \in A$, $t \in C$
- $s \in A$, $t \in A$
- $s \in B$, $t \in C$
- $s \in B$, $t \in A$

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(a) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: Es gilt $s \in B$ und $t \in C$.



s liegt in $S = A \cup B$ und t in $T = A \cup C$

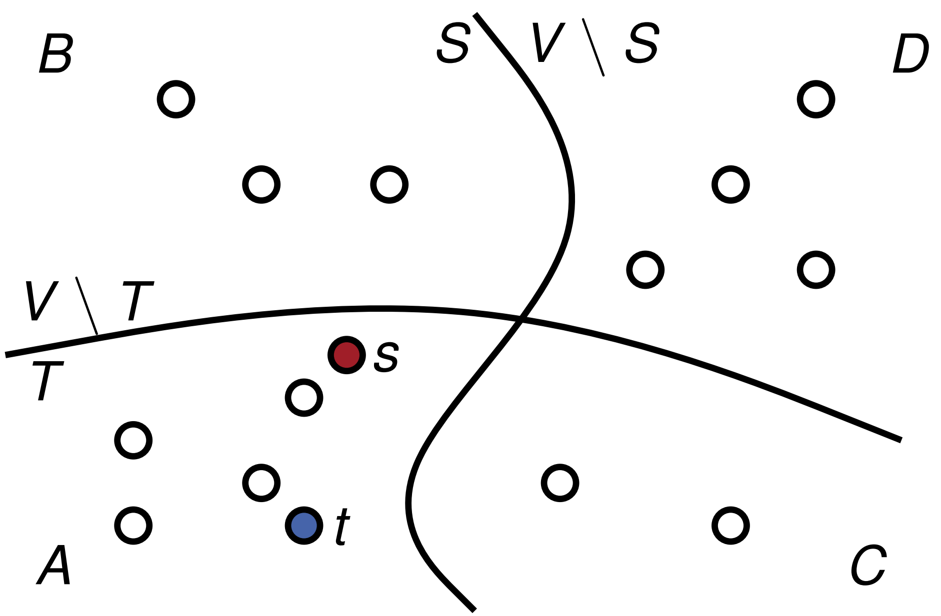
$\Rightarrow$ Es gibt vier mögliche Kombinationen:

- $\boxed{s \in A, t \in C}$ T ist kein s-t-Schnitt
- $s \in A, t \in A$
- $s \in B, t \in C$
- $s \in B, t \in A$

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(a) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: Es gilt $s \in B$ und $t \in C$.



s liegt in $S = A \cup B$ und t in $T = A \cup C$

$\Rightarrow$ Es gibt vier mögliche Kombinationen:

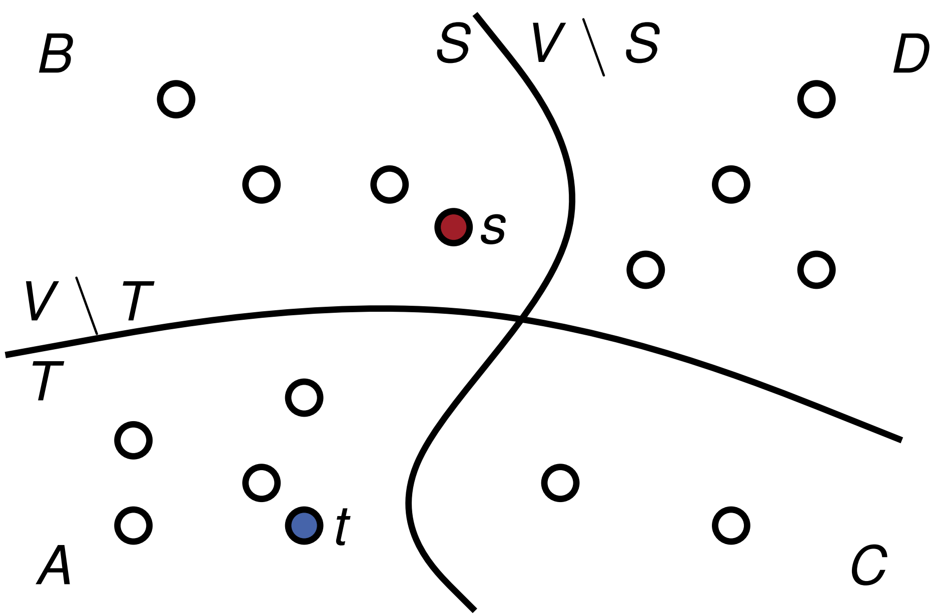
- $\boxed{s \in A, t \in C}$ — T ist kein s-t-Schnitt
- $\boxed{s \in A, t \in A}$ — T & S keine s-t-Schnitte
- $s \in B, t \in C$
- $s \in B, t \in A$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(a) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: Es gilt $s \in B$ und $t \in C$.



s liegt in $S = A \cup B$ und t in $T = A \cup C$

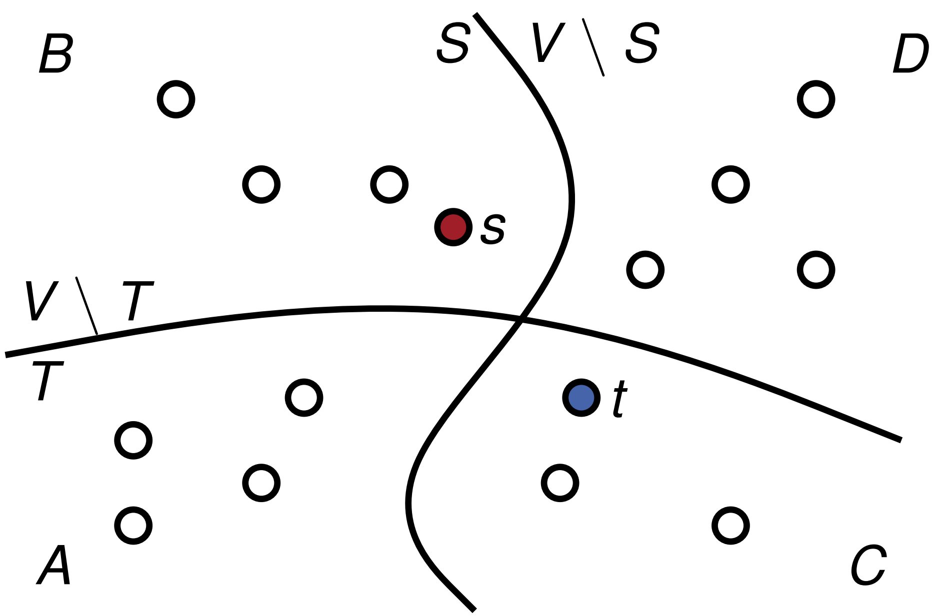
$\Rightarrow$ Es gibt vier mögliche Kombinationen:

- $\boxed{s \in A, t \in C}$ T ist kein s-t-Schnitt
- $\boxed{s \in A, t \in A}$ T & S keine s-t-Schnitte
- $s \in B, t \in C$
- $\boxed{s \in B, t \in A}$ S ist kein s-t-Schnitt

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(a) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: Es gilt $s \in B$ und $t \in C$.



s liegt in $S = A \cup B$ und t in $T = A \cup C$

$\Rightarrow$ Es gibt vier mögliche Kombinationen:

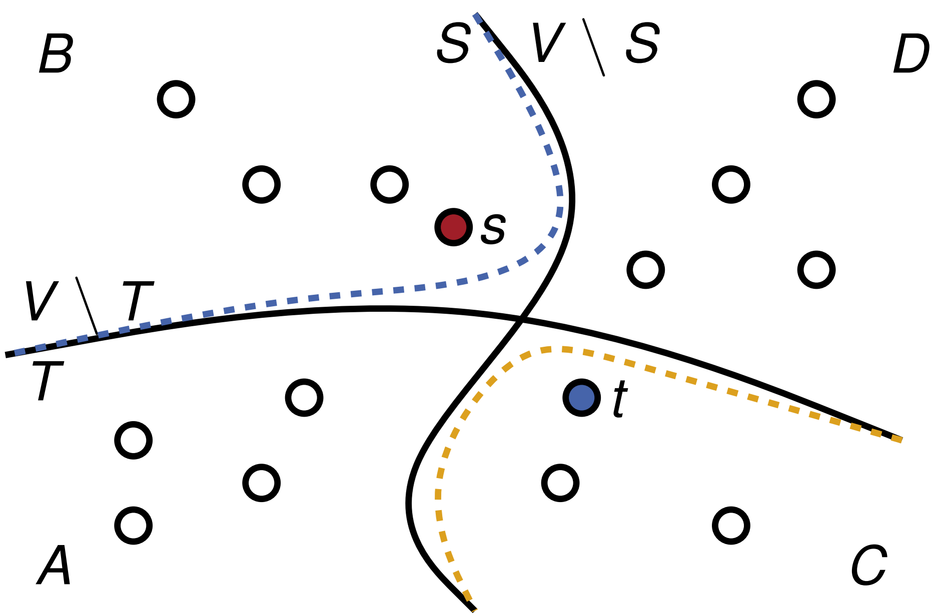
- $\boxed{s \in A, t \in C}$ T ist kein s-t-Schnitt
- $\boxed{s \in A, t \in A}$ T & S keine s-t-Schnitte
- $\boxed{s \in B, t \in C}$ ⟵
- $\boxed{s \in B, t \in A}$ S ist kein s-t-Schnitt

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(b) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende minimale s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind zwei kreuzungsfreie minimale s-t-Schnitte.

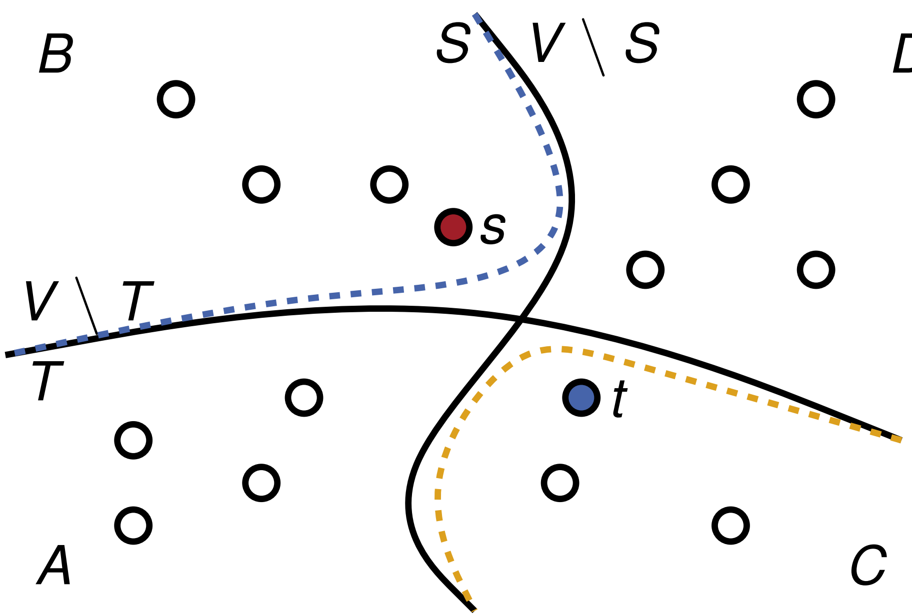


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(b) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende minimale s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind zwei kreuzungsfreie minimale s-t-Schnitte.



Offensichtlich: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind kreuzungsfreie s-t-Schnitte.

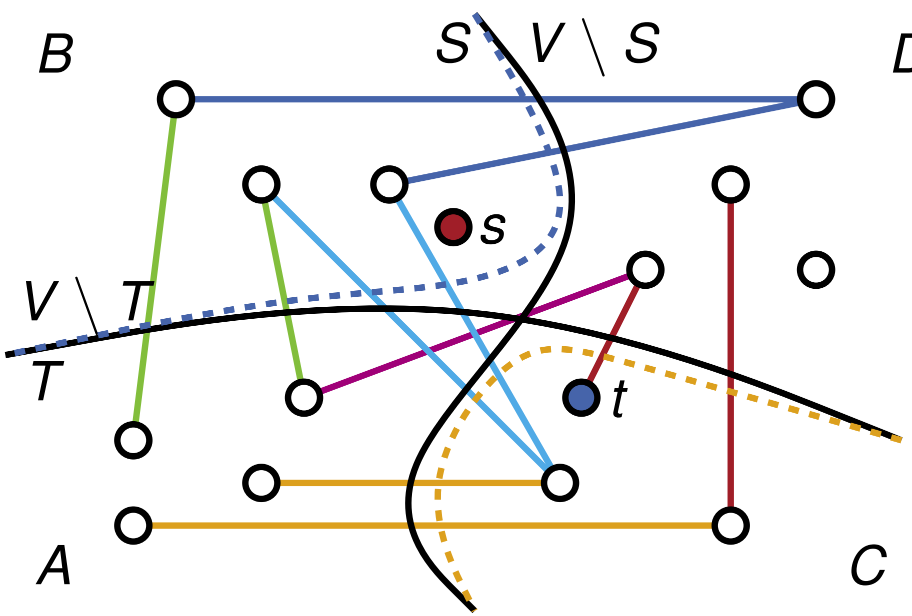
Zeige:

$c(B, V \setminus B) = c(C, V \setminus C) = c(S, V \setminus S) = c(T, V \setminus T)$

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \rightarrow \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(b) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende minimale s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind zwei kreuzungsfreie minimale s-t-Schnitte.



Offensichtlich: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind kreuzungsfreie s-t-Schnitte.

Zeige:

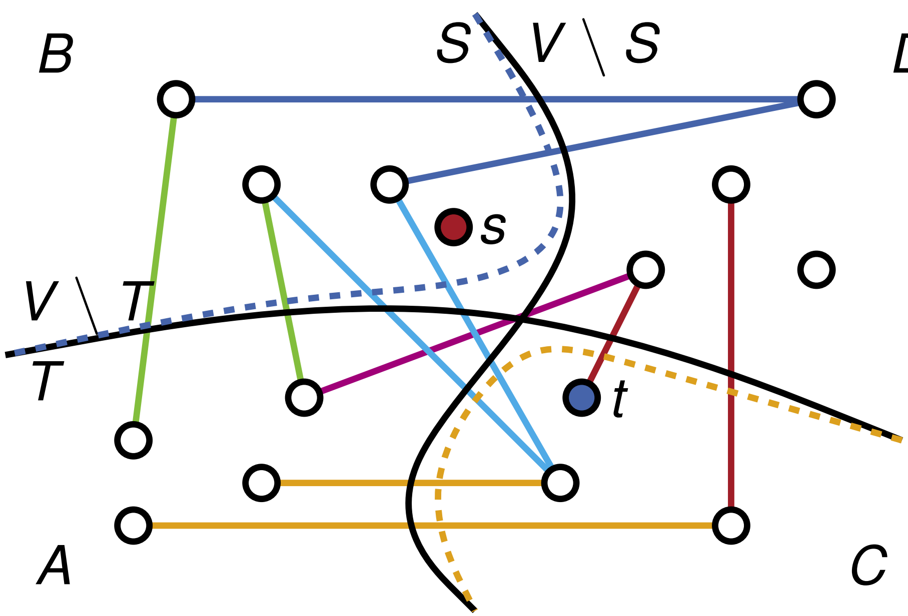
$c(B, V \setminus B) = c(C, V \setminus C) = c(S, V \setminus S) = c(T, V \setminus T)$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(b) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende minimale s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind zwei kreuzungsfreie minimale s-t-Schnitte.



Offensichtlich: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind kreuzungsfreie s-t-Schnitte.

Zeige:

$c(B, V \setminus B) = c(C, V \setminus C) = c(S, V \setminus S) = c(T, V \setminus T)$

$c(B, V \setminus B) = \boxed{c(B, A)} + \boxed{c(B, C)} + \boxed{c(B, D)}$

$c(C, V \setminus C) = \boxed{c(C, A)} + \boxed{c(C, B)} + \boxed{c(C, D)}$

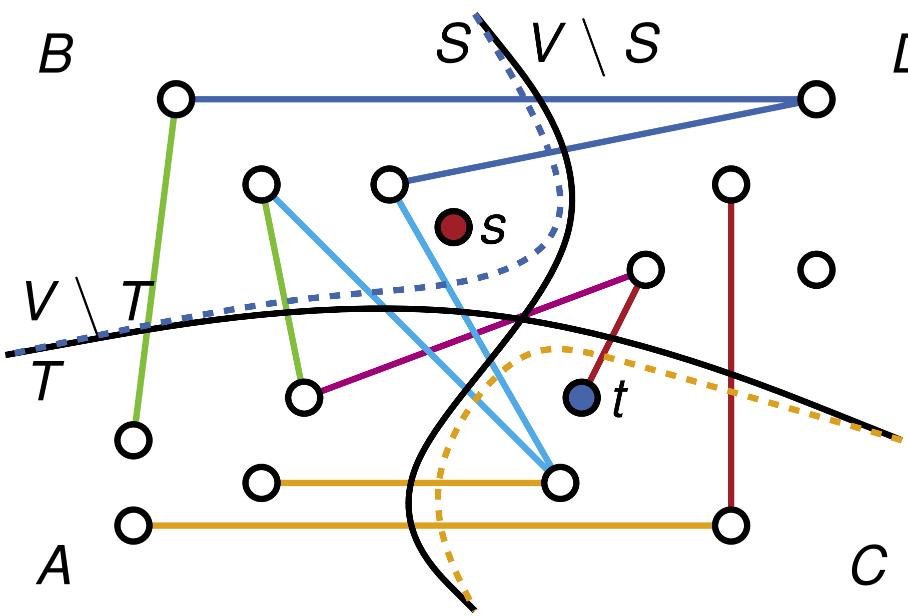
$c(S, V \setminus S) = \boxed{c(A, C)} + \boxed{c(A, D)} + \boxed{c(B, C)} + \boxed{c(B, D)}$

$c(T, V \setminus T) = \boxed{c(A, B)} + \boxed{c(A, D)} + \boxed{c(C, B)} + \boxed{c(C, D)}$

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(b) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende minimale s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind zwei kreuzungsfreie minimale s-t-Schnitte.



Offensichtlich: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind kreuzungsfreie s-t-Schnitte.

Zeige:

$c(B, V \setminus B) = c(C, V \setminus C) = c(S, V \setminus S) = c(T, V \setminus T)$

$c(B, V \setminus B) = \boxed{c(B, A)} + \boxed{c(B, C)} + \boxed{c(B, D)}$

$c(C, V \setminus C) = \boxed{c(C, A)} + \boxed{c(C, B)} + \boxed{c(C, D)}$

$c(S, V \setminus S) = \boxed{c(A, C)} + \boxed{c(A, D)} + \boxed{c(B, C)} + \boxed{c(B, D)}$

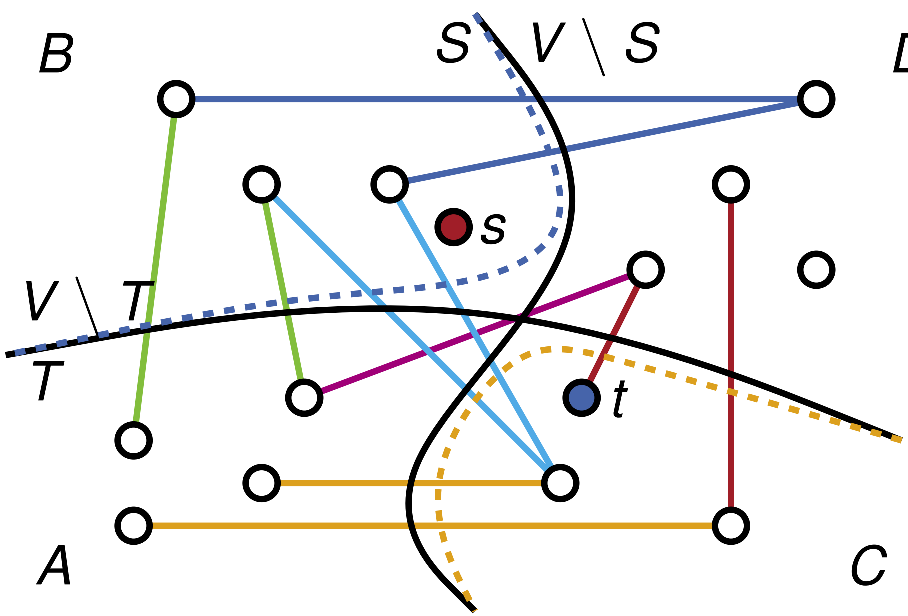
$c(T, V \setminus T) = \boxed{c(A, B)} + \boxed{c(A, D)} + \boxed{c(C, B)} + \boxed{c(C, D)}$

$\Rightarrow c(B, V \setminus B) + c(C, V \setminus C) \leq c(S, V \setminus S) + c(T, V \setminus S)$

Problem 5

Zwei Schnitte $(S, V \setminus S)$ und $(T, V \setminus T)$ in einem (ungerichteten) Graphen $G = (V, E)$ *kreuzen sich*, wenn keine der Mengen $A = S \cap T$, $B = S \setminus T$, $C = T \setminus S$ und $D = V \setminus (S \cup T)$ leer ist. Sei $c : E \to \mathbb{R}_0^+$ eine Kantengewichtsfunktion auf G.

(b) Seien $(S, V \setminus S)$ und $(T, V \setminus T)$ zwei sich kreuzende minimale s-t-Schnitte mit $s \in S$ und $t \in T$. Zeigen Sie: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind zwei kreuzungsfreie minimale s-t-Schnitte.



Offensichtlich: $(B, V \setminus B)$ und $(C, V \setminus C)$ sind kreuzungsfreie s-t-Schnitte.

Zeige:

$c(B, V \setminus B) = c(C, V \setminus C) = c(S, V \setminus S) = c(T, V \setminus T)$

$c(B, V \setminus B) = \boxed{c(B, A)} + \boxed{c(B, C)} + \boxed{c(B, D)}$

$c(C, V \setminus C) = \boxed{c(C, A)} + \boxed{c(C, B)} + \boxed{c(C, D)}$

$c(S, V \setminus S) = \boxed{c(A, C)} + \boxed{c(A, D)} + \boxed{c(B, C)} + \boxed{c(B, D)}$

$c(T, V \setminus T) = \boxed{c(A, B)} + \boxed{c(A, D)} + \boxed{c(C, B)} + \boxed{c(C, D)}$

$\Rightarrow c(B, V \setminus B) + c(C, V \setminus C) \leq c(S, V \setminus S) + c(T, V \setminus S)$

$\Rightarrow c(B, V \setminus B) = c(C, V \setminus C) = c(S, V \setminus S) = c(T, V \setminus S)$ (da $c(S, V \setminus S)$ und $c(T, V \setminus T)$ minimal)

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

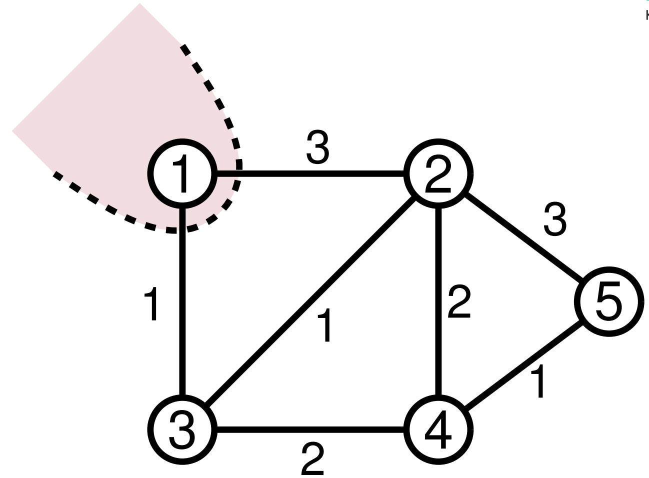
Problem 6 – Algorithmus von Stoer & Wagner

Phase 1

$G_1 = G$

$S_1 = \{1\}$

(Startknoten 1)



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

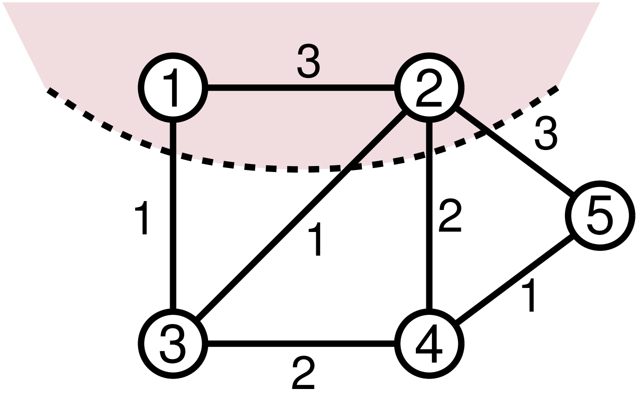
Problem 6 – Algorithmus von Stoer & Wagner

Phase 1

$G_1 = G$

$S_1 = \{1\}$ (Startknoten 1)

$S_1 = \{1, 2\}$ (2 am stärksten zu $\{1\}$ verbunden)



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 6 – Algorithmus von Stoer & Wagner

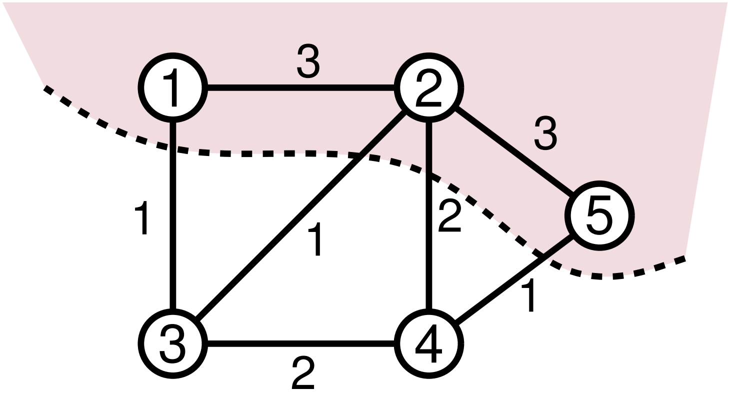
Phase 1

$G_1 = G$

$S_1 = \{1\}$ (Startknoten 1)

$S_1 = \{1, 2\}$ (2 am stärksten zu $\{1\}$ verbunden)

$S_1 = \{1, 2, 5\}$



Problem 6 – Algorithmus von Stoer & Wagner

Phase 1

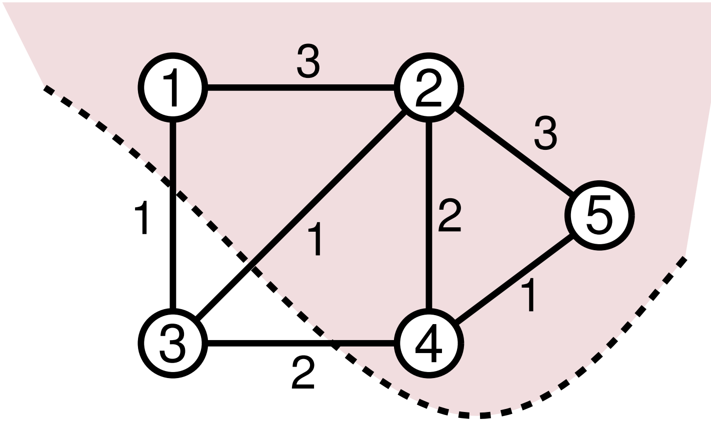
$G_1 = G$

$S_1 = \{1\}$ (Startknoten 1)

$S_1 = \{1, 2\}$ (2 am stärksten zu $\{1\}$ verbunden)

$S_1 = \{1, 2, 5\}$

$S_1 = \{1, 2, 5, 4\}$



Problem 6 – Algorithmus von Stoer & Wagner

Phase 1

$G_1 = G$

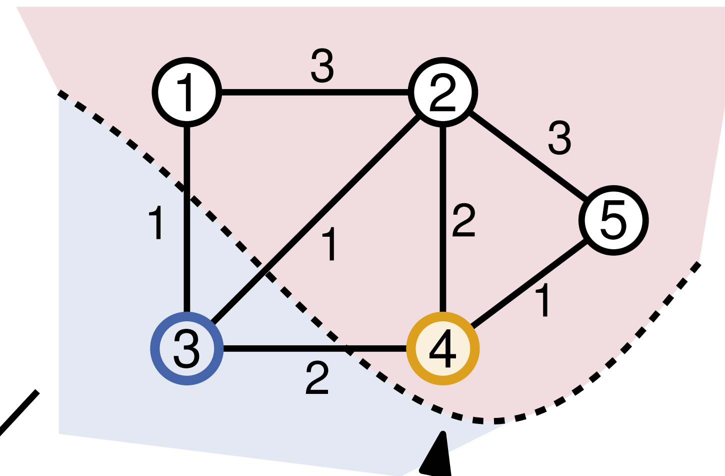
$S_1 = \{1\}$ (Startknoten 1)

$S_1 = \{1, 2\}$ (2 am stärksten zu $\{1\}$ verbunden)

$S_1 = \{1, 2, 5\}$

$S_1 = \{1, 2, 5, 4\}$

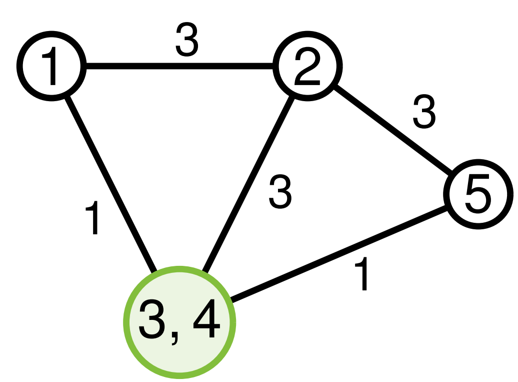
$S_1 = \{1, 2, 5, \underbrace{4}_{s}, \underbrace{3}_{t}\}$



Schnitt der Phase: $\{V_1 \setminus \{3\}, \{3\}\}$

$\rightarrow$ Gewicht 4

Verschmelzen von s und t ergibt G_2

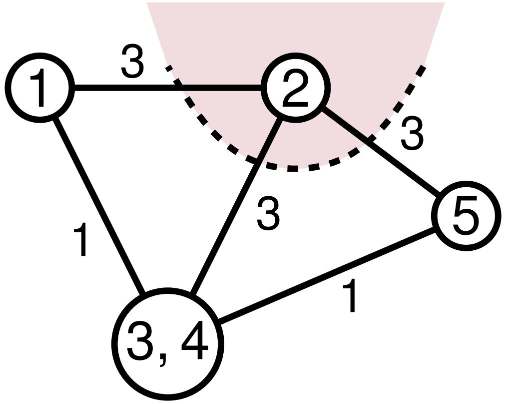


Problem 6 – Algorithmus von Stoer & Wagner

Phase 2

$G_2 = G_1$ mit 3 und 4 verschmolzen

$S_2 = \{2\}$ (Startknoten 2)



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

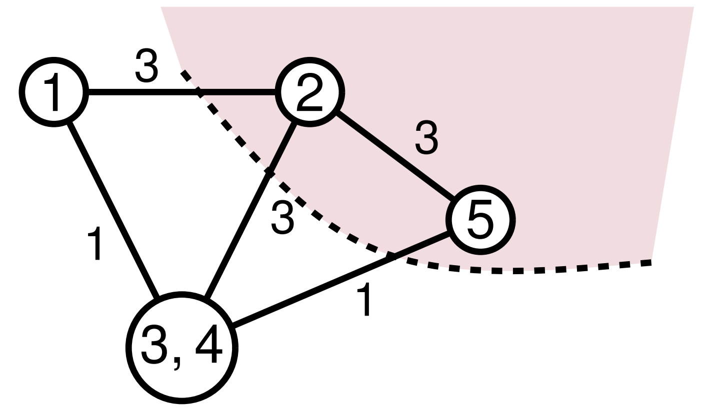
Problem 6 – Algorithmus von Stoer & Wagner

Phase 2

$G_2 = G_1$ mit 3 und 4 verschmolzen

$S_2 = \{2\}$ (Startknoten 2)

$S_2 = \{2, 5\}$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 6 – Algorithmus von Stoer & Wagner

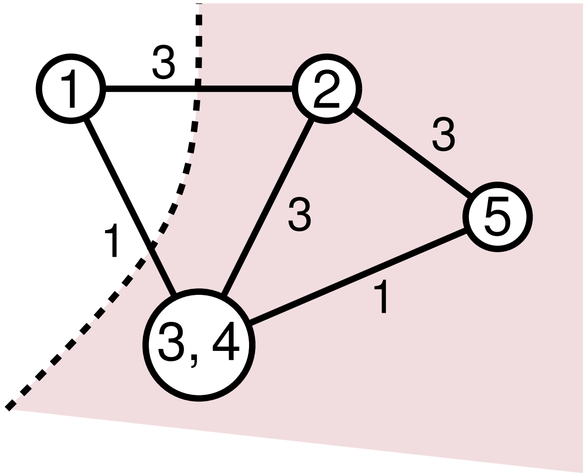
Phase 2

$G_2 = G_1$ mit 3 und 4 verschmolzen

$S_2 = \{2\}$ (Startknoten 2)

$S_2 = \{2, 5\}$

$S_2 = \{2, 5, \{3, 4\}\}$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 6 – Algorithmus von Stoer & Wagner

Phase 2

$G_2 = G_1$ mit 3 und 4 verschmolzen

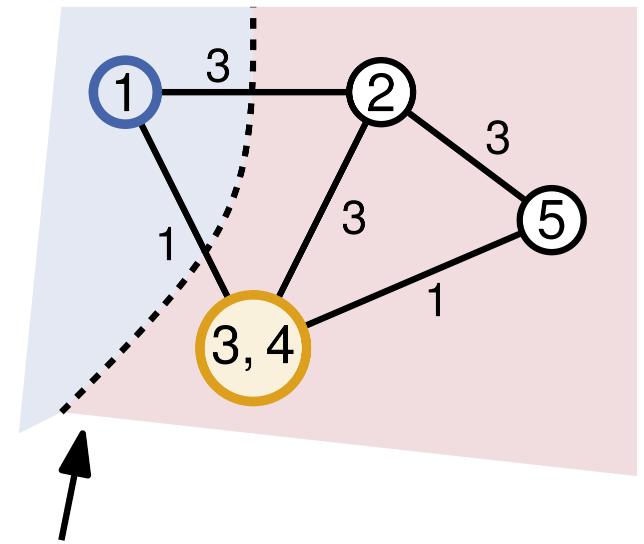
$S_2 = \{2\}$ (Startknoten 2)

$S_2 = \{2, 5\}$

$S_2 = \{2, 5, \{3, 4\}\}$

$S_2 = \{2, 5, \{3, 4\}, 1\}$

s t



Schnitt der Phase: $\{V_2 \setminus \{1\}, \{1\}\}$

$\rightarrow$ Gewicht 4

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 6 – Algorithmus von Stoer & Wagner

Phase 2

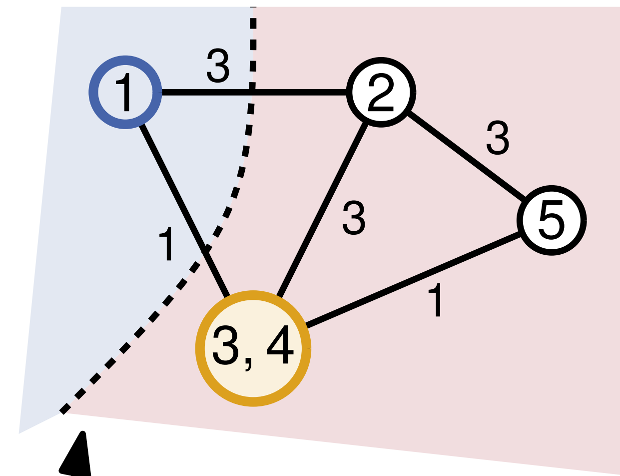
$G_2 = G_1$ mit 3 und 4 verschmolzen

$S_2 = \{2\}$ (Startknoten 2)

$S_2 = \{2, 5\}$

$S_2 = \{2, 5, \{3, 4\}\}$

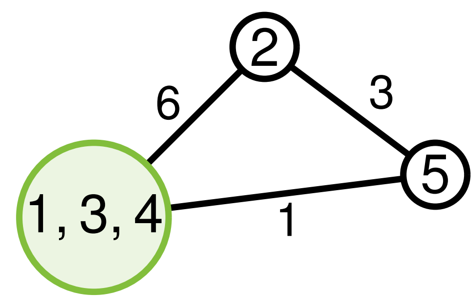
$S_2 = \{2, 5, \underbrace{\{3, 4\}}_{s}, \underbrace{1}_{t}\}$



Schnitt der Phase: $\{V_2 \setminus \{1\}, \{1\}\}$

$\rightarrow$ Gewicht 4

Verschmelzen von s und t ergibt G_3

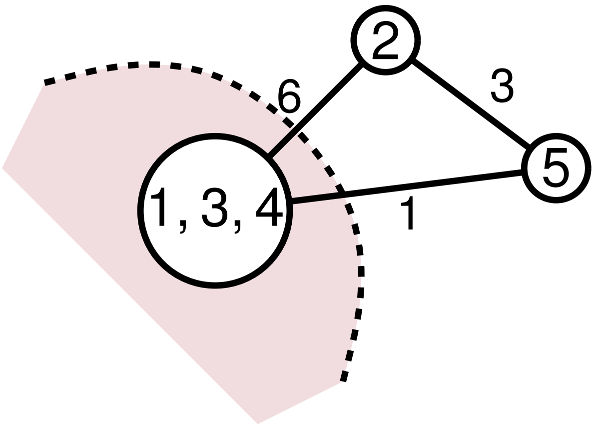


Problem 6 – Algorithmus von Stoer & Wagner

Phase 3

$G_3 = G_2$ mit 1 und $\{3, 4\}$ verschmolzen

$S_3 = \{\{1, 3, 4\}\}$ (Startknoten $\{1, 3, 4\}$)



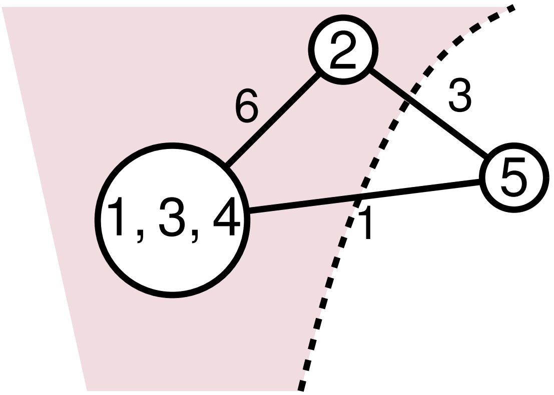
Problem 6 – Algorithmus von Stoer & Wagner

Phase 3

$G_3 = G_2$ mit 1 und $\{3, 4\}$ verschmolzen

$S_3 = \{\{1, 3, 4\}\}$ (Startknoten $\{1, 3, 4\}$)

$S_3 = \{\{1, 3, 4\}, 2\}$



Problem 6 – Algorithmus von Stoer & Wagner

Phase 3

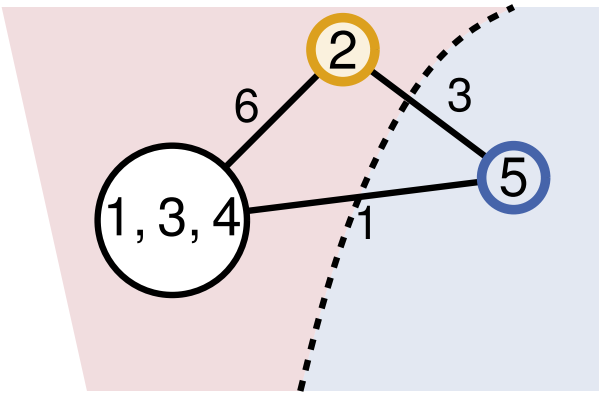
$G_3 = G_2$ mit 1 und $\{3, 4\}$ verschmolzen

$S_3 = \{\{1, 3, 4\}\}$ (Startknoten $\{1, 3, 4\}$)

$S_3 = \{\{1, 3, 4\}, 2\}$

$S_3 = \{\{1, 3, 4\}, 2, 5\}$

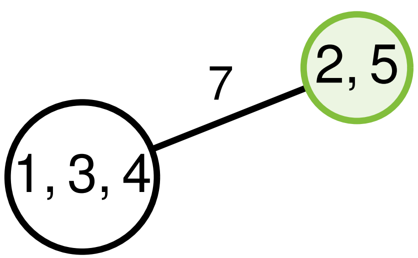
s t



Schnitt der Phase: $\{V_3 \setminus \{5\}, \{5\}\}$

$\rightarrow$ Gewicht 4

Verschmelzen von s und t ergibt G_4

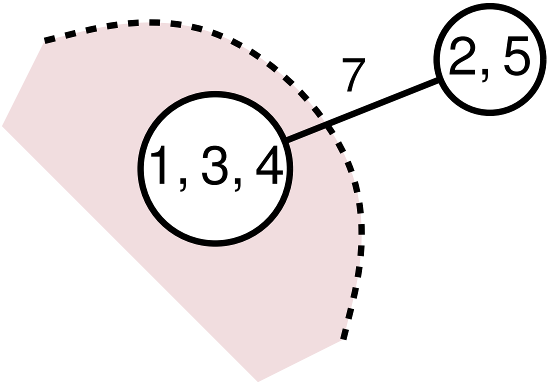


Problem 6 – Algorithmus von Stoer & Wagner

Phase 4

$G_4 = G_3$ mit 2 und 5 verschmolzen

$S_4 = \{\{1, 3, 4\}\}$ (Startknoten $\{1, 3, 4\}$)



Problem 6 – Algorithmus von Stoer & Wagner

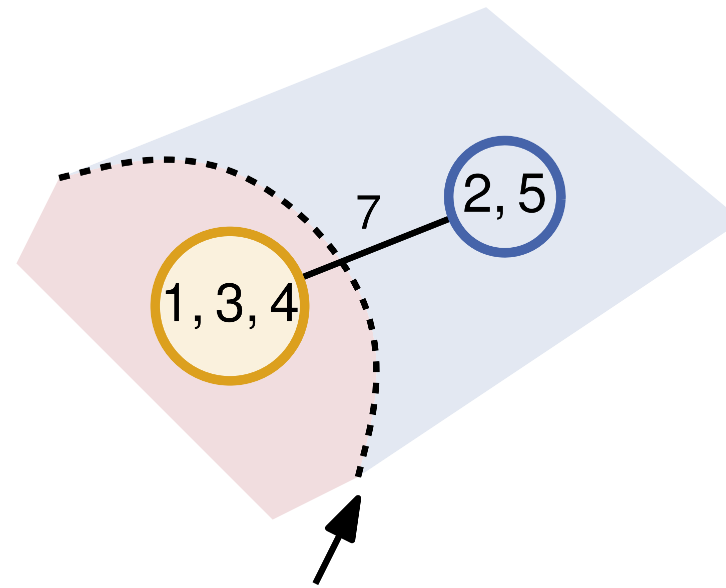
Phase 4

$G_4 = G_3$ mit 2 und 5 verschmolzen

$S_4 = \{\{1,3,4\}\}$ (Startknoten $\{1,3,4\}$)

$S_3 = \{\{1,3,4\},\{2,5\}\}$

s t



Schnitt der Phase: $\{V_3 \setminus \{2,5\}, \{2,5\}\}$

$\rightarrow$ Gewicht 7

Problem 6 – Algorithmus von Stoer & Wagner

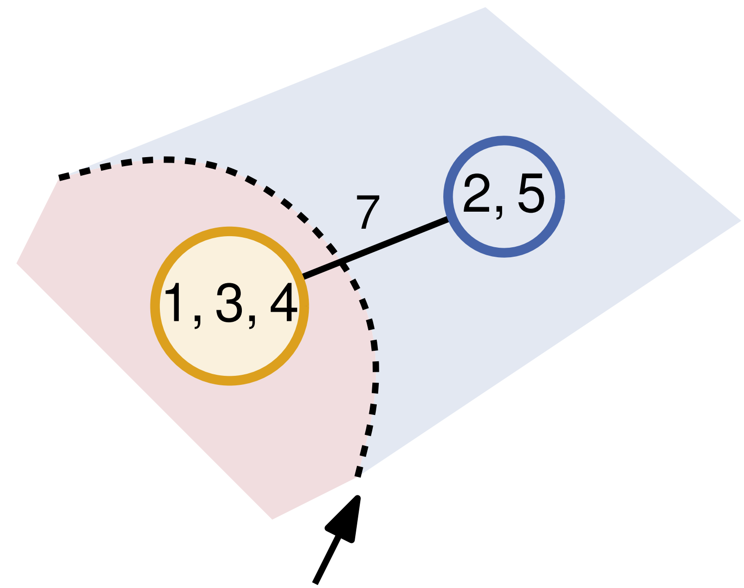
Phase 4

$G_4 = G_3$ mit 2 und 5 verschmolzen

$S_4 = \{\{1,3,4\}\}$ (Startknoten $\{1,3,4\}$)

$S_3 = \{\{1,3,4\},\{2,5\}\}$

s t



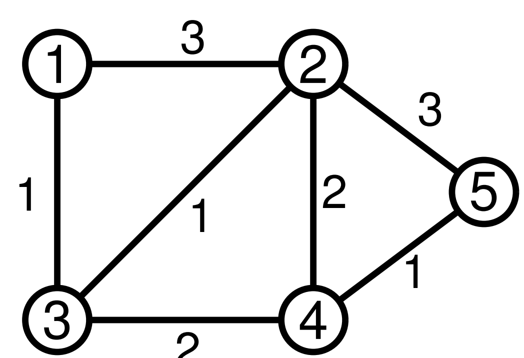
Schnitt der Phase: $\{V_3 \setminus \{2,5\},\{2,5\}\}$

$\rightarrow$ Gewicht 7

Zusammenfassung:

Drei minimale Schnitte gefunden.

Phase 1 Schnitt der Phase: $\{V \setminus \{3\},\{3\}\} \rightarrow$ Gewicht 4
Phase 2 Schnitt der Phase: $\{V \setminus \{1\},\{1\}\} \rightarrow$ Gewicht 4
Phase 3 Schnitt der Phase: $\{V \setminus \{5\},\{5\}\} \rightarrow$ Gewicht 4
Phase 4 Schnitt der Phase: $\{V \setminus \{\{2,5\}\},\{\{2,5\}\}\} \rightarrow$ Gewicht 7



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

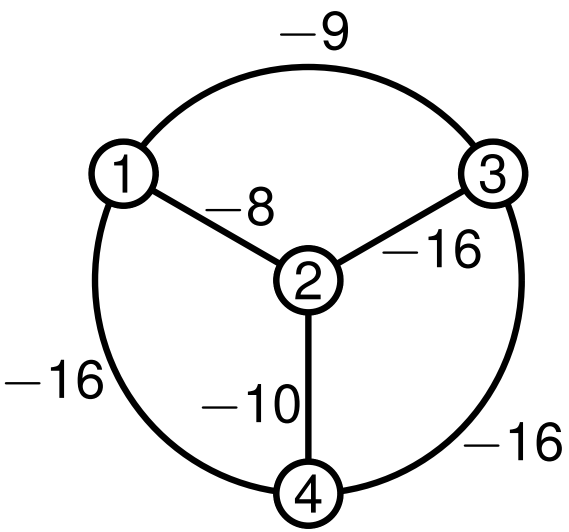
Problem 6

(b) Liefert der Algorithmus von Stoer & Wagner auch für negative Kantengewichte einen global minimalen, nichttrivialen Schnitt? Begründen Sie Ihre Antwort.

Problem 6

(b) Liefert der Algorithmus von Stoer & Wagner auch für negative Kantengewichte einen global minimalen, nichttrivialen Schnitt? Begründen Sie Ihre Antwort.

Nein, denn:



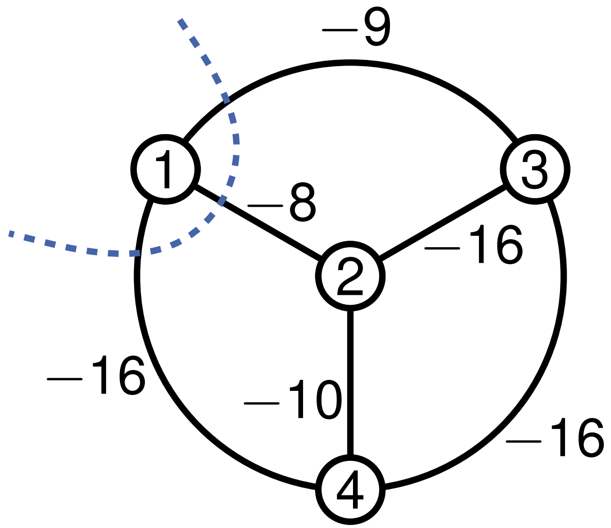
Startknoten 1

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 6

(b) Liefert der Algorithmus von Stoer & Wagner auch für negative Kantengewichte einen global minimalen, nichttrivialen Schnitt? Begründen Sie Ihre Antwort.

Nein, denn:

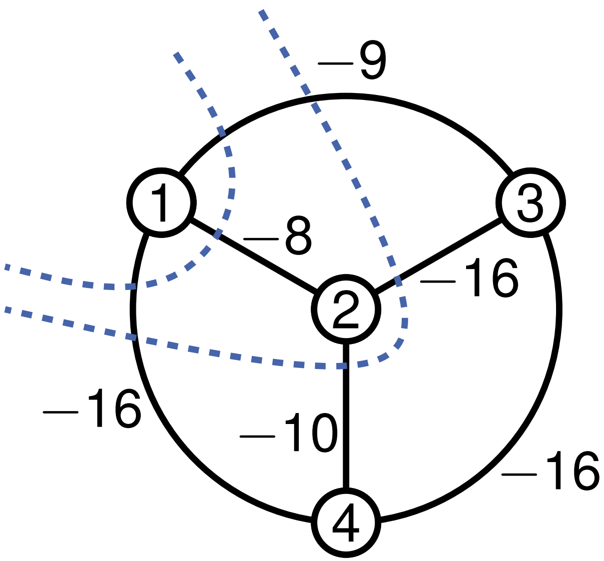


Startknoten 1

Problem 6

(b) Liefert der Algorithmus von Stoer & Wagner auch für negative Kantengewichte einen global minimalen, nichttrivialen Schnitt? Begründen Sie Ihre Antwort.

Nein, denn:

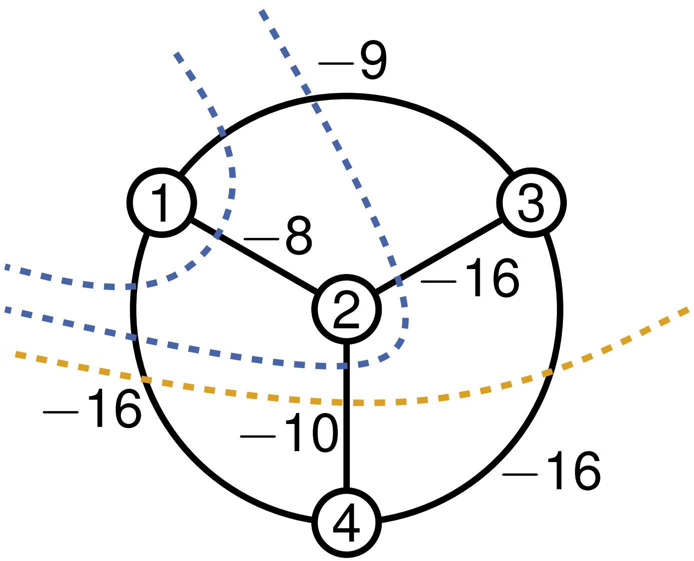


Startknoten 1

Problem 6

(b) Liefert der Algorithmus von Stoer & Wagner auch für negative Kantengewichte einen global minimalen, nichttrivialen Schnitt? Begründen Sie Ihre Antwort.

Nein, denn:



Startknoten 1

$\Rightarrow$ Schnitt der Phase 1:

$(V \setminus \{4\}, \{4\})$

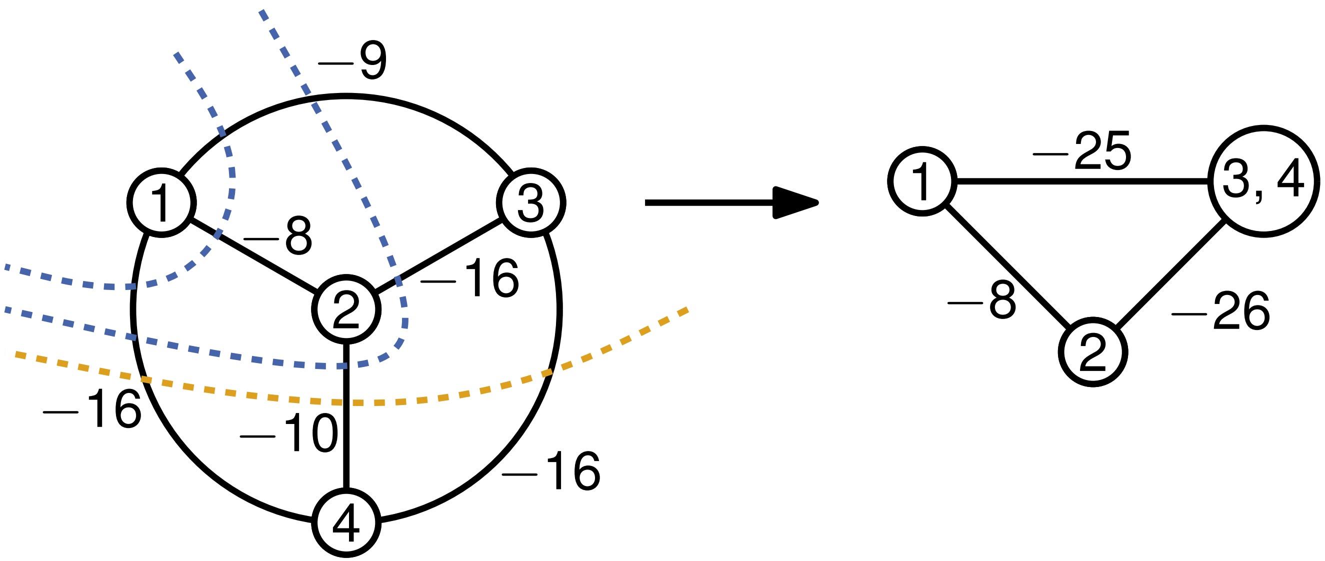
↳ Gewicht -42

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 6

(b) Liefert der Algorithmus von Stoer & Wagner auch für negative Kantengewichte einen global minimalen, nichttrivialen Schnitt? Begründen Sie Ihre Antwort.

Nein, denn:



Startknoten 1

$\Rightarrow$ Schnitt der Phase 1:

$(V \setminus \{4\}, \{4\})$

↳ Gewicht -42

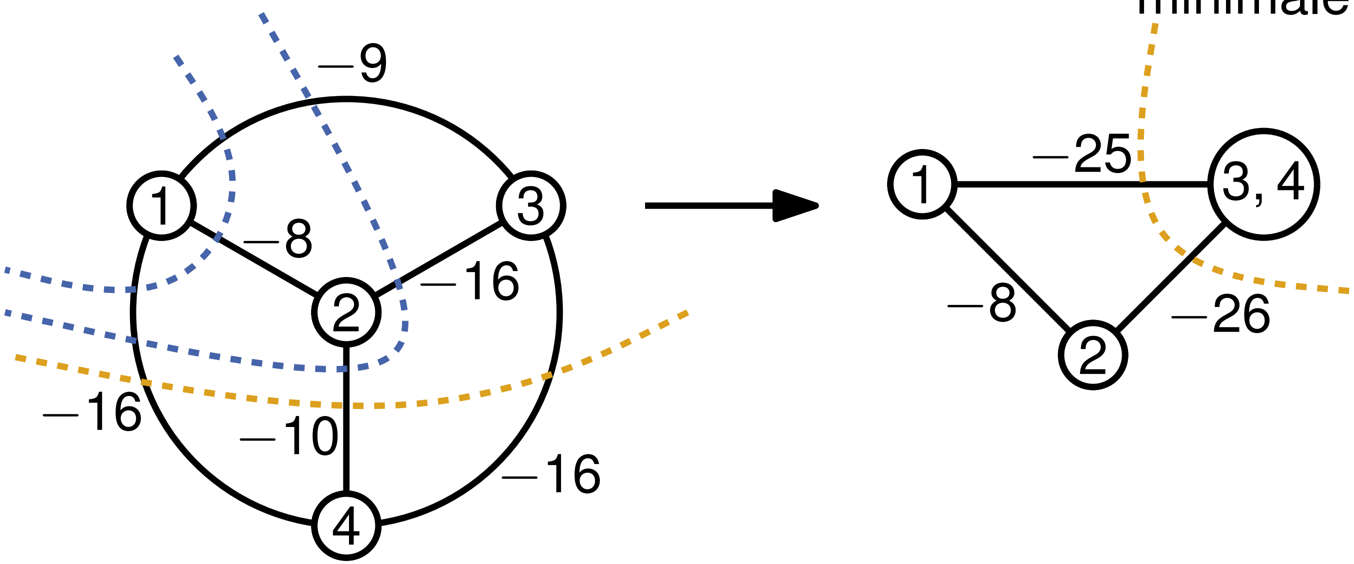
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 6

(b) Liefert der Algorithmus von Stoer & Wagner auch für negative Kantengewichte einen global minimalen, nichttrivialen Schnitt? Begründen Sie Ihre Antwort.

Nein, denn:

minimaler Schnitt hat Gewicht -51



Startknoten 1

$\Rightarrow$ Schnitt der Phase 1:

$(V \setminus \{4\}, \{4\})$

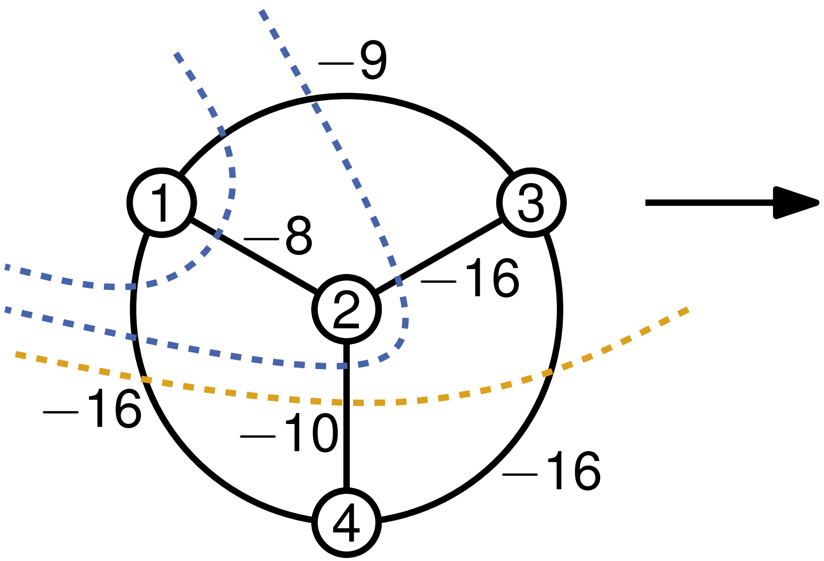
$\Rightarrow$ Gewicht -42

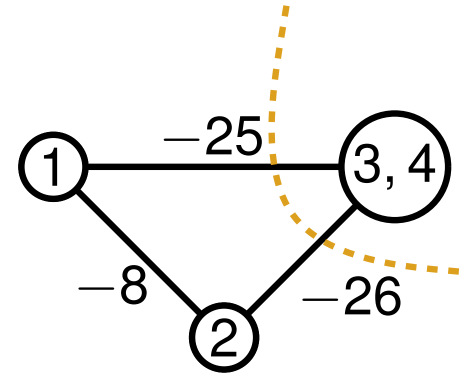
Problem 6

(b) Liefert der Algorithmus von Stoer & Wagner auch für negative Kantengewichte einen global minimalen, nichttrivialen Schnitt? Begründen Sie Ihre Antwort.

Nein, denn:

minimaler Schnitt hat Gewicht -51





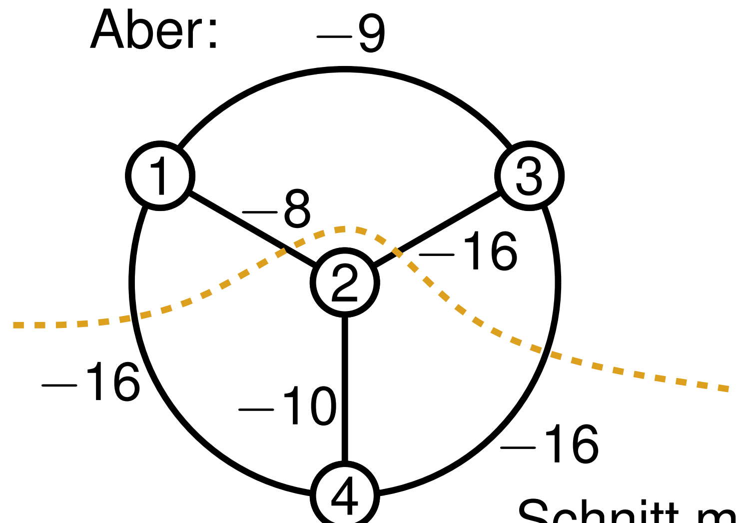
Startknoten 1

$\Rightarrow$ Schnitt der Phase 1:

$(V \setminus \{4\}, \{4\})$

$\Rightarrow$ Gewicht -42

Aber:



Schnitt mit Gewicht -56