29. Flüsse in Netzen

Flussnetzwerk, Fluss, Maximaler Fluss
Restkapazität, Restnetzwerk, Erweiterungsweg

Ford-Fulkerson Algorithmus, Edmonds-Karp Algorithmus

Schnitte, Max-Flow Min-Cut Theorem

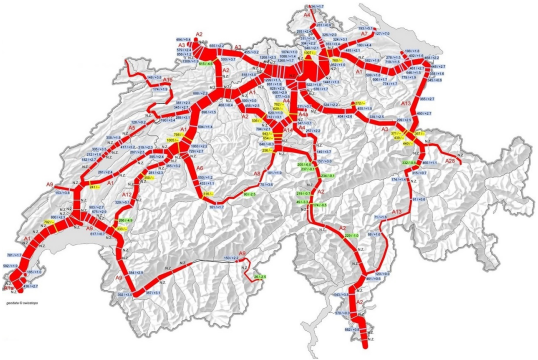
[Ottman/Widmayer, Kap. 9.7, 9.8.1], [Cormen et al, Kap. 26.1-26.3]

Folienredesign: Manuela Fischer – vielen Dank!

Maximaler Verkehrsfluss

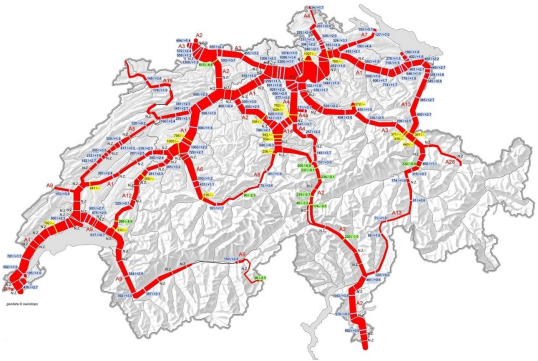
Maximaler Verkehrsfluss

Gegeben: Strassennetzwerk mit Kapazitäten



Maximaler Verkehrsfluss

Gegeben: Strassennetzwerk mit Kapazitäten

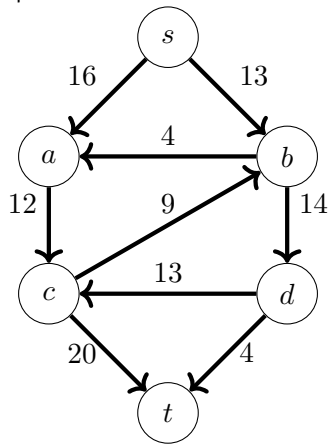


Gesucht: Maximaler Verkehrsfluss zwischen Zürich und Genf

Flussnetzwerk

Flussnetzwerk

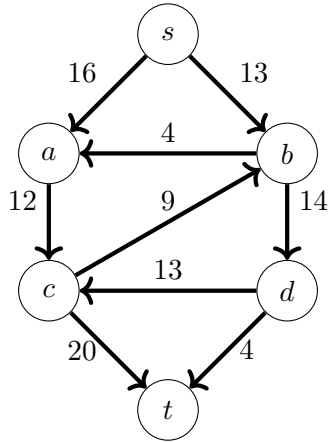
gerichteter, gewichteter Graph $G = (V, E, c)$ mit Kapazitäten $c\colon E \to \mathbb{R}^{>0}$



Flussnetzwerk

gerichteter, gewichteter Graph $G = (V, E, c)$ mit Kapazitäten $c\colon E \to \mathbb{R}^{>0}$

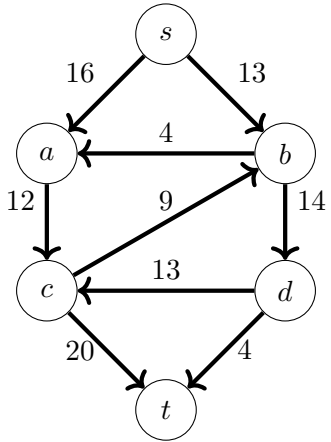
- ohne antiparallele Kanten:



Flussnetzwerk

gerichteter, gewichteter Graph $G = (V, E, c)$ mit Kapazitäten $c\colon E \to \mathbb{R}^{>0}$

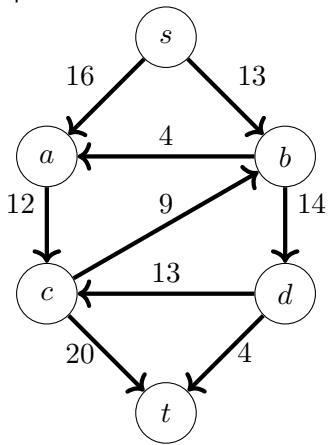
- ohne antiparallele Kanten:
  $(u, v) \in E$



Flussnetzwerk

gerichteter, gewichteter Graph $G = (V, E, c)$ mit Kapazitäten $c\colon E \to \mathbb{R}^{>0}$

- ohne antiparallele Kanten:
  $(u, v) \in E \;\Rightarrow\; (v, u) \notin E$

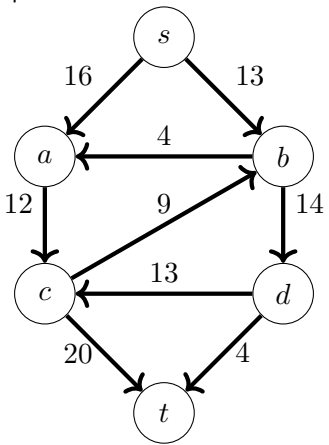


Flussnetzwerk

gerichteter, gewichteter Graph $G = (V, E, c)$ mit Kapazitäten $c\colon E \to \mathbb{R}^{>0}$

- ohne antiparallele Kanten:
  $(u, v) \in E \;\Rightarrow\; (v, u) \notin E$

- Quelle $s \in V$ ohne eingehende Kanten:

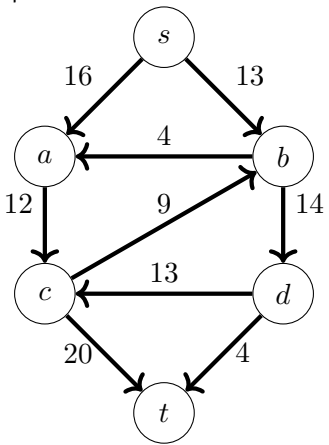


Flussnetzwerk

gerichteter, gewichteter Graph $G = (V, E, c)$ mit Kapazitäten $c\colon E \to \mathbb{R}^{>0}$

- ohne antiparallele Kanten:
  $(u, v) \in E \;\Rightarrow\; (v, u) \notin E$

- Quelle $s \in V$ ohne eingehende Kanten:
  $\forall v \in V\colon (v, s) \notin E$



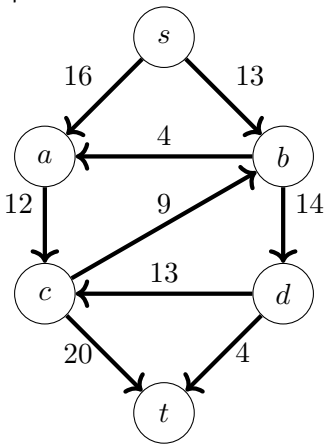
Flussnetzwerk

gerichteter, gewichteter Graph $G = (V, E, c)$ mit Kapazitäten $c\colon E \to \mathbb{R}^{>0}$

- ohne antiparallele Kanten:
  $(u, v) \in E \;\Rightarrow\; (v, u) \notin E$

- Quelle $s \in V$ ohne eingehende Kanten:
  $\forall v \in V\colon (v, s) \notin E$

- Senke $t \in V$ ohne ausgehende Kanten:



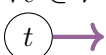
Flussnetzwerk

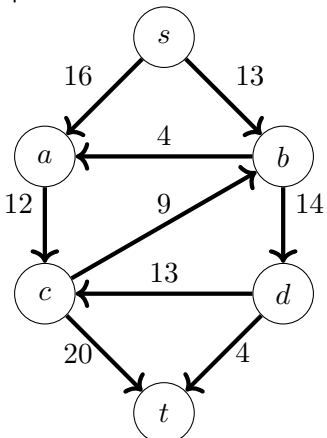
gerichteter, gewichteter Graph $G = (V, E, c)$ mit Kapazitäten $c\colon E \to \mathbb{R}^{>0}$

- ohne antiparallele Kanten:
  $(u, v) \in E \;\Rightarrow\; (v, u) \notin E$

- Quelle $s \in V$ ohne eingehende Kanten:
  $\forall v \in V\colon (v, s) \notin E$

- Senke $t \in V$ ohne ausgehende Kanten:
  $\forall v \in V\colon (t, v) \notin E$





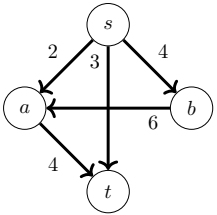
Quiz Flussnetzwerk

Quiz Flussnetzwerk

Welche der folgenden Graphen sind Flussnetzwerke?

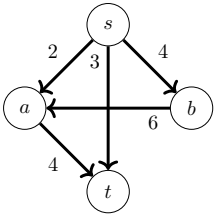
Quiz Flussnetzwerk

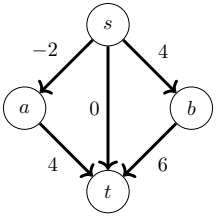
Welche der folgenden Graphen sind Flussnetzwerke?



Quiz Flussnetzwerk

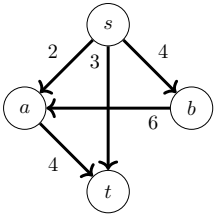
Welche der folgenden Graphen sind Flussnetzwerke?

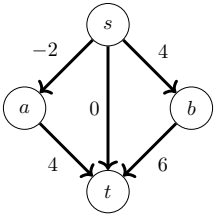


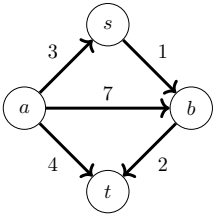


Quiz Flussnetzwerk

Welche der folgenden Graphen sind Flussnetzwerke?

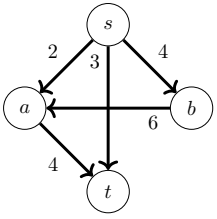


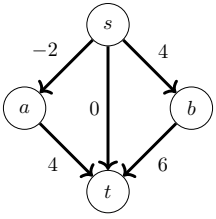


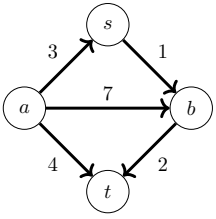


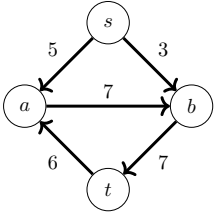
Quiz Flussnetzwerk

Welche der folgenden Graphen sind Flussnetzwerke?



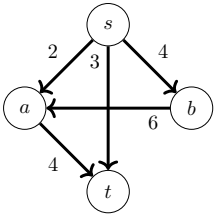


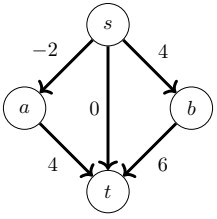


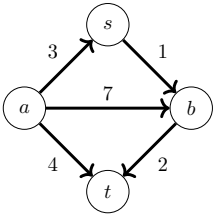


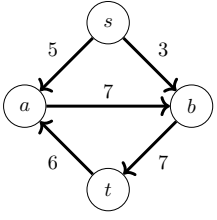
Quiz Flussnetzwerk

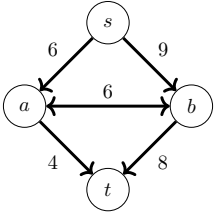
Welche der folgenden Graphen sind Flussnetzwerke?





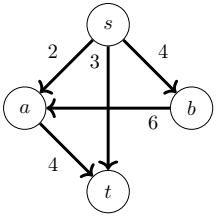


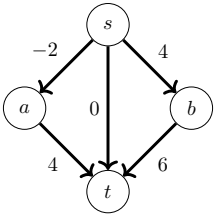


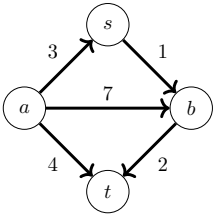


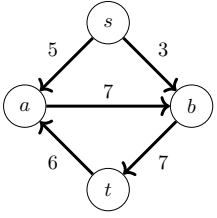
Quiz Flussnetzwerk

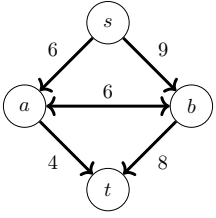
Welche der folgenden Graphen sind Flussnetzwerke?

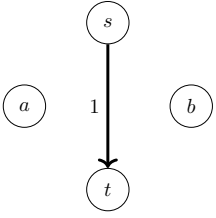






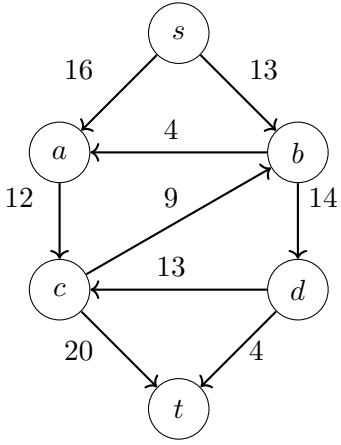






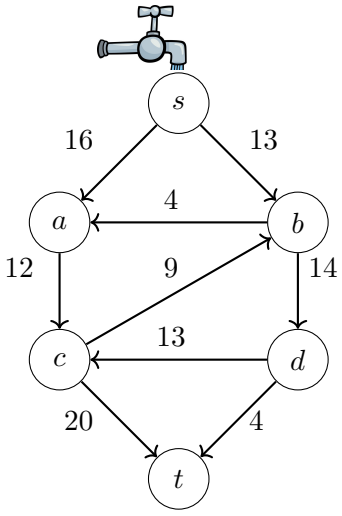
Fluss in Flussnetzwerk

Fluss in Flussnetzwerk



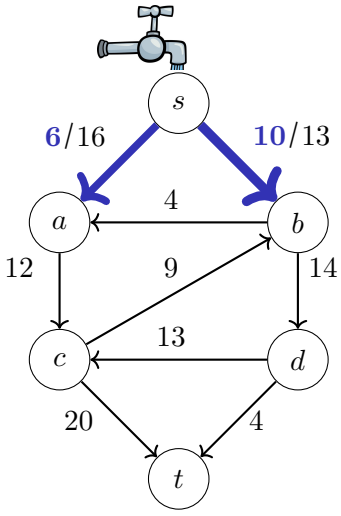
Fluss in Flussnetzwerk

Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass



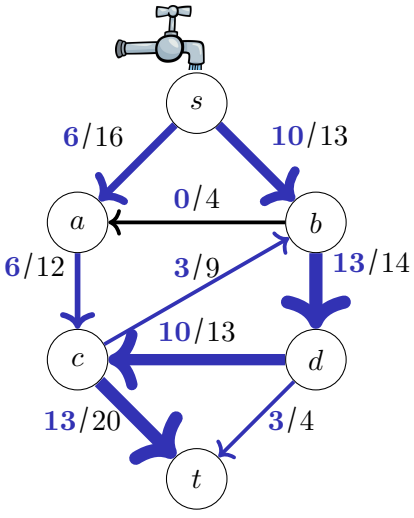
Fluss in Flussnetzwerk

Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass



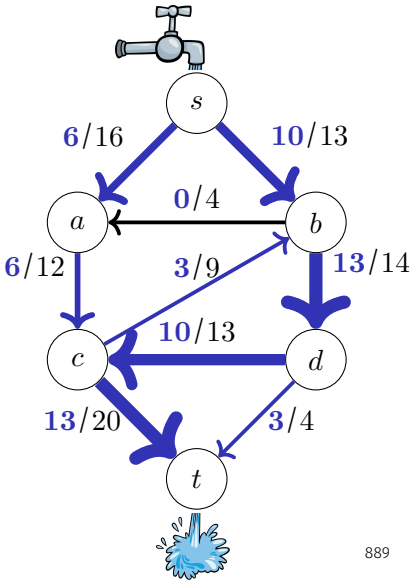
Fluss in Flussnetzwerk

Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass



Fluss in Flussnetzwerk

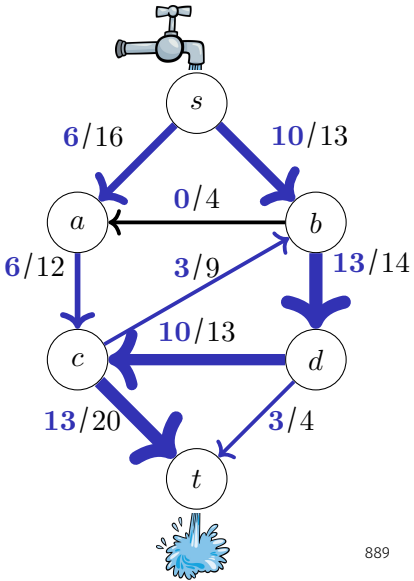
Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass



Fluss in Flussnetzwerk

Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass

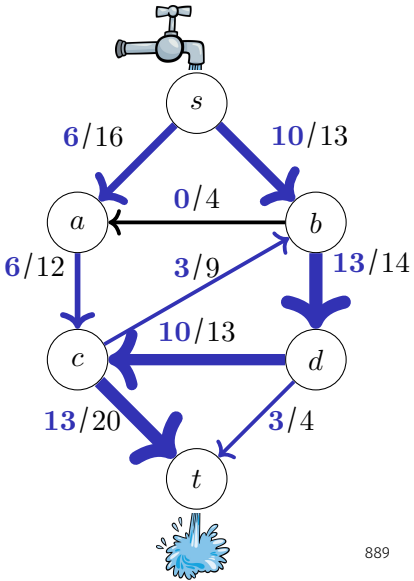
- **Kapazitätsbeschränkung**:
  $\forall e \in E\colon f(e) \leq c(e)$



Fluss in Flussnetzwerk

Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass

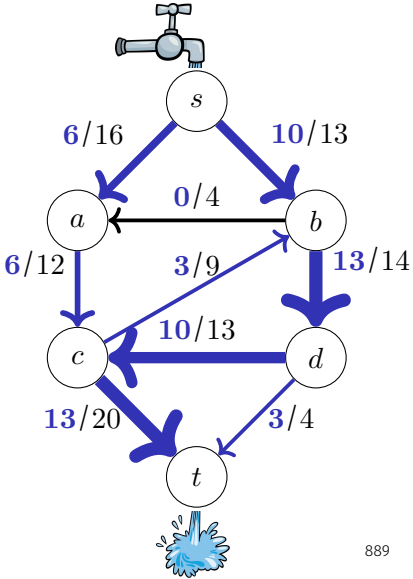
- **Kapazitätsbeschränkung**:
  $\forall e \in E\colon f(e) \leq c(e)$
- **Flusserhaltung**:



Fluss in Flussnetzwerk

Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass

- **Kapazitätsbeschränkung**:
  $\forall e \in E\colon f(e) \leq c(e)$
- **Flusserhaltung**: $\forall v \in V \setminus \{s,t\}\colon$

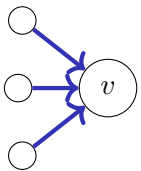


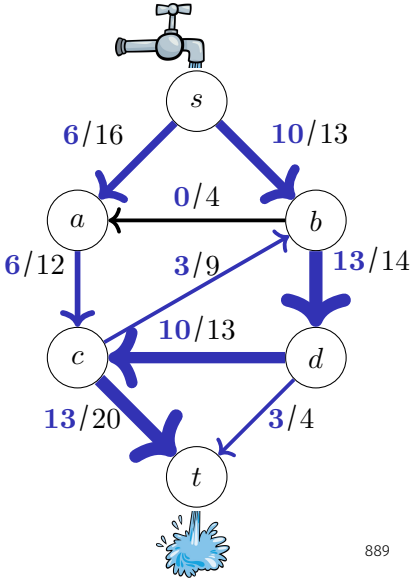
Fluss in Flussnetzwerk

Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass

- **Kapazitätsbeschränkung**:
  $\forall e \in E\colon f(e) \leq c(e)$
- **Flusserhaltung**: $\forall v \in V \setminus \{s,t\}\colon$

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



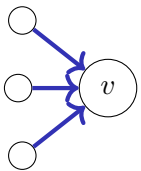


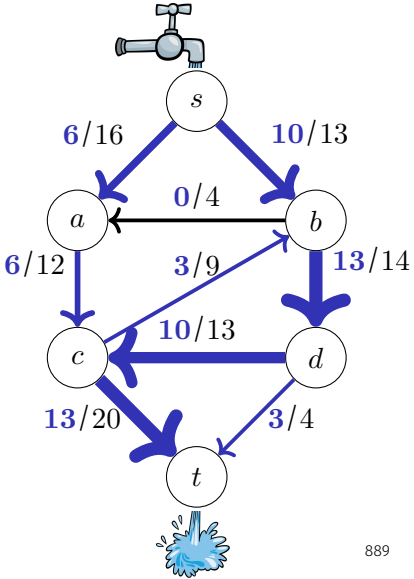
Fluss in Flussnetzwerk

Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass

- **Kapazitätsbeschränkung**:
  $\forall e \in E\colon f(e) \leq c(e)$
- **Flusserhaltung**: $\forall v \in V \setminus \{s,t\}\colon$

$$\underbrace{\sum_{e \in E^-(v)} f(e)}_{=:f^-(v)}$$



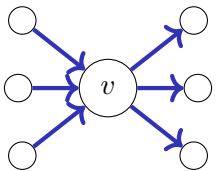


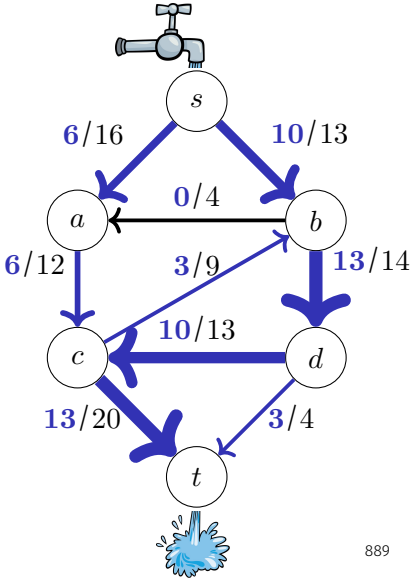
Fluss in Flussnetzwerk

Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass

- **Kapazitätsbeschränkung**:
  $\forall e \in E\colon f(e) \leq c(e)$
- **Flusserhaltung**: $\forall v \in V \setminus \{s,t\}\colon$

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



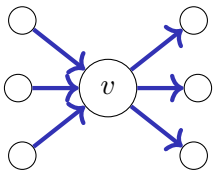


Fluss in Flussnetzwerk

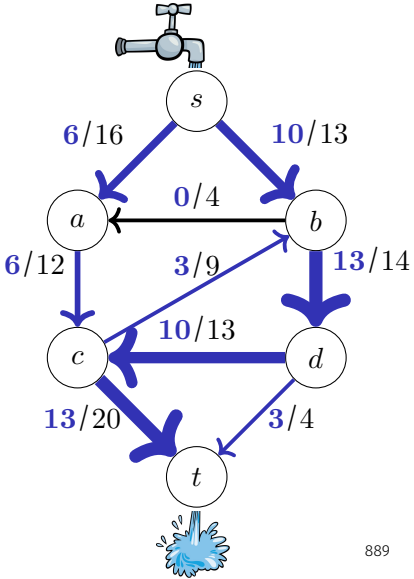
Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass

- **Kapazitätsbeschränkung**:
  $\forall e \in E\colon f(e) \leq c(e)$
- **Flusserhaltung**: $\forall v \in V \setminus \{s,t\}\colon$

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



Grösse des Flusses: $|f| := f^+(s) = f^-(t)$

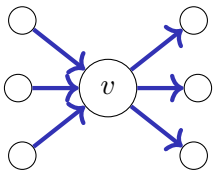


Fluss in Flussnetzwerk

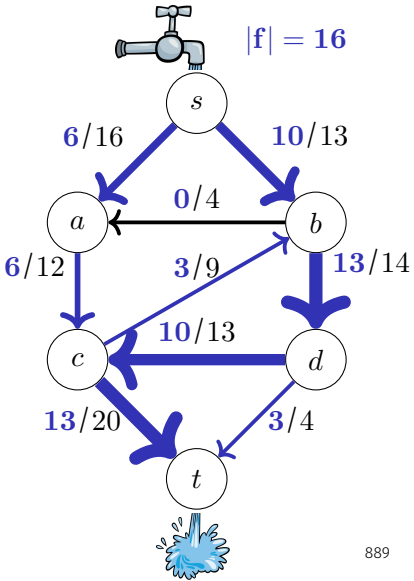
Fluss ist Funktion $f\colon E \to \mathbb{R}^{\geq 0}$ so dass

- **Kapazitätsbeschränkung**:
  $\forall e \in E\colon f(e) \leq c(e)$
- **Flusserhaltung**: $\forall v \in V \setminus \{s,t\}\colon$

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

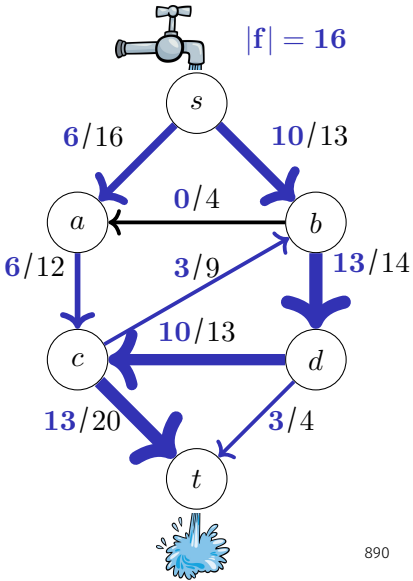


Grösse des Flusses: $|f| := f^+(s) = f^-(t)$

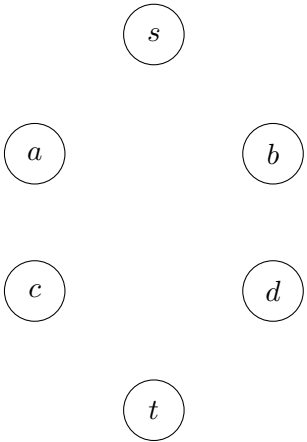


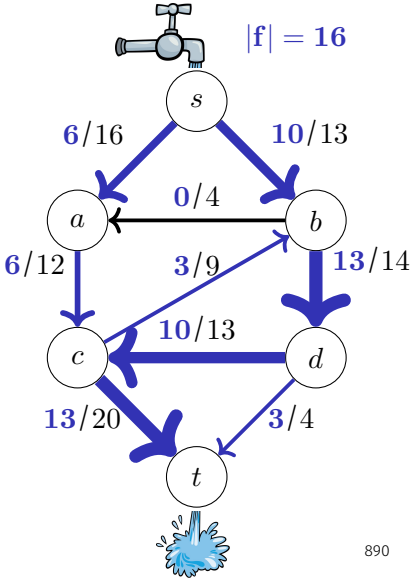
Intuition: Fluss als Menge von Wegen $s \rightsquigarrow t$

Intuition: Fluss als Menge von Wegen $s \rightsquigarrow t$

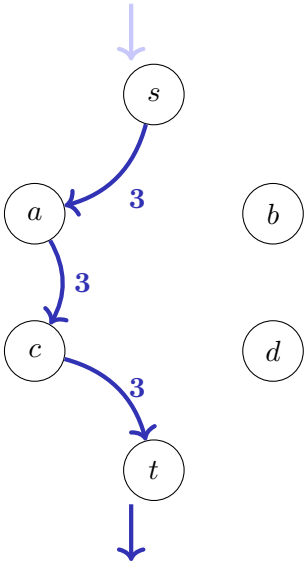


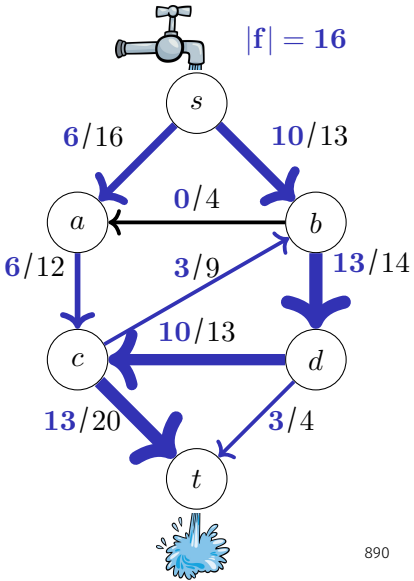
Intuition: Fluss als Menge von Wegen $s \rightsquigarrow t$



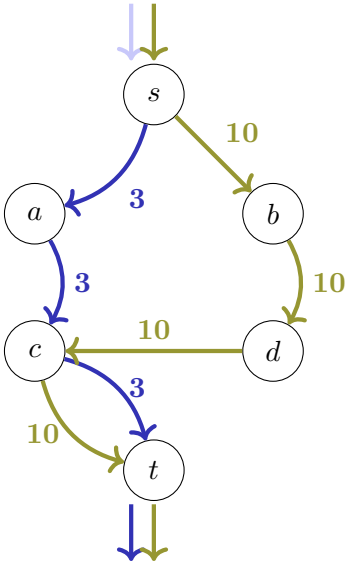


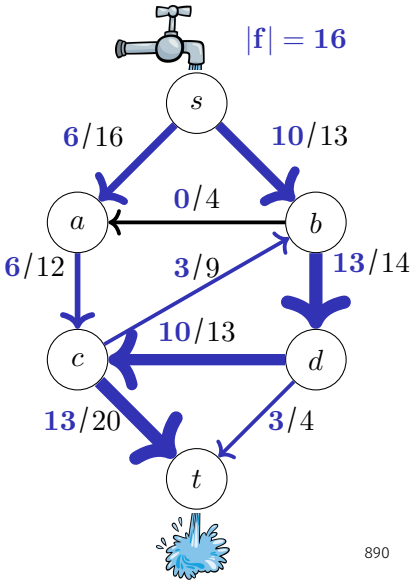
Intuition: Fluss als Menge von Wegen $s \rightsquigarrow t$



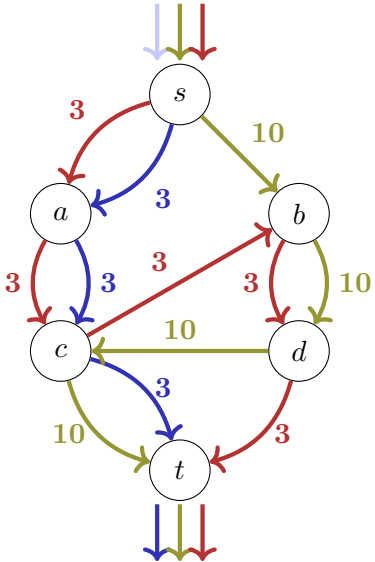


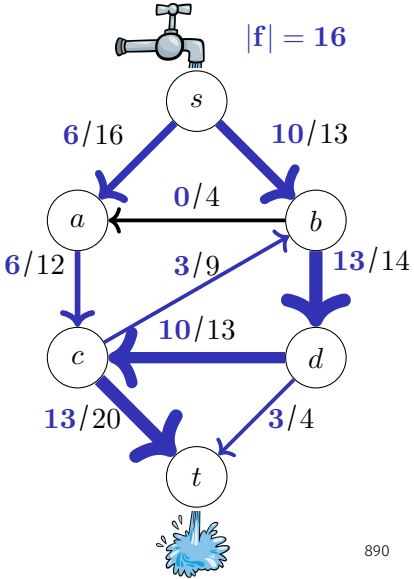
Intuition: Fluss als Menge von Wegen $s \rightsquigarrow t$





Intuition: Fluss als Menge von Wegen $s \rightsquigarrow t$





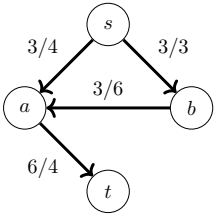
Quiz Fluss

Quiz Fluss

Welches sind Flüsse?

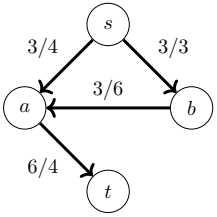
Quiz Fluss

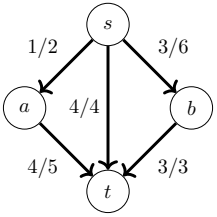
Welches sind Flüsse?



Quiz Fluss

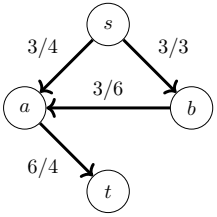
Welches sind Flüsse?

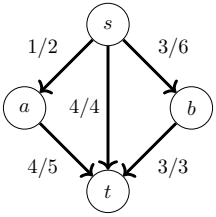


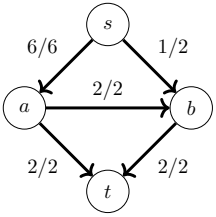


Quiz Fluss

Welches sind Flüsse?

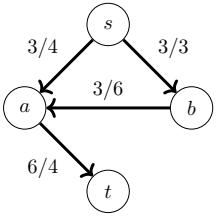


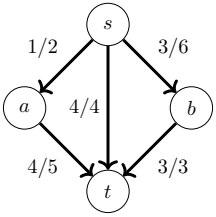


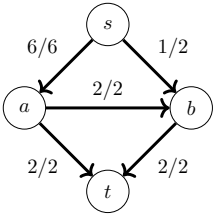


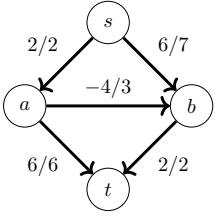
Quiz Fluss

Welches sind Flüsse?



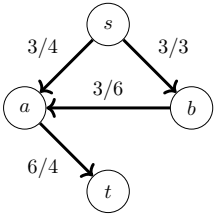


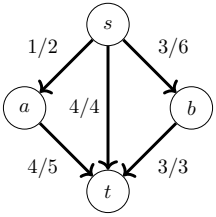


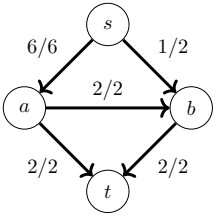


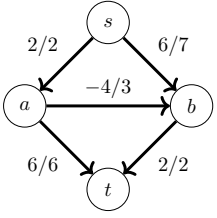
Quiz Fluss

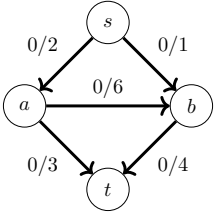
Welches sind Flüsse?





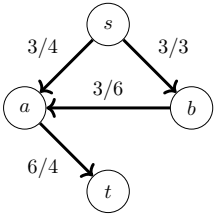


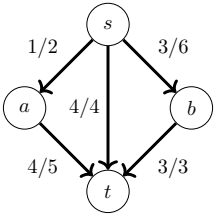


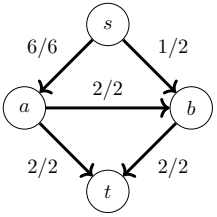


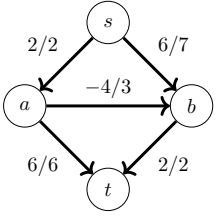
Quiz Fluss

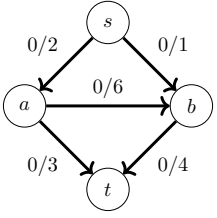
Welches sind Flüsse?

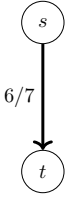








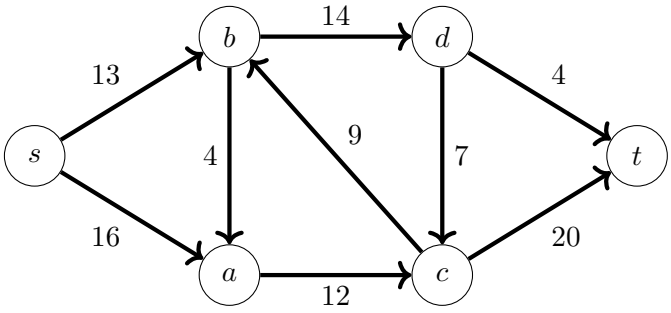




Maximaler Fluss

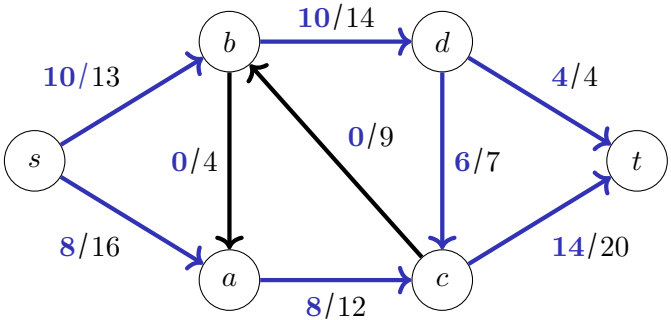
Maximaler Fluss

Gegeben: Flussnetzwerk: $G = (V, E, c)$, gerichtet, positiv gewichtet, ohne antiparallele Kanten, mit Quelle s und Senke t



Maximaler Fluss

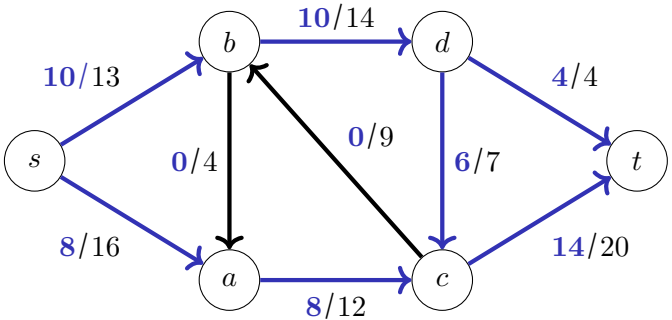
Gegeben: Flussnetzwerk: $G = (V, E, c)$, gerichtet, positiv gewichtet, ohne antiparallele Kanten, mit Quelle s und Senke t



Gesucht: Grösse $|f_{\max}|$ des maximalen Flusses in G

Maximaler Fluss

Gegeben: Flussnetzwerk: $G = (V, E, c)$, gerichtet, positiv gewichtet, ohne antiparallele Kanten, mit Quelle s und Senke t



$$\mathbf{18} = |\mathbf{f}| \leq |\mathbf{f}_{\max}| = \mathbf{23}$$

Gesucht: Grösse $|f_{\max}|$ des maximalen Flusses in G

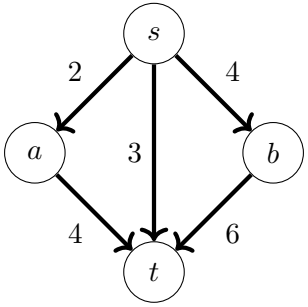
Quiz Maximaler Fluss

Quiz Maximaler Fluss

Was ist der maximale Fluss im folgenden Flussnetzwerk?

Quiz Maximaler Fluss

Was ist der maximale Fluss im folgenden Flussnetzwerk?



Greedy Algorithmus?

Greedy Algorithmus?

Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy Algorithmus?

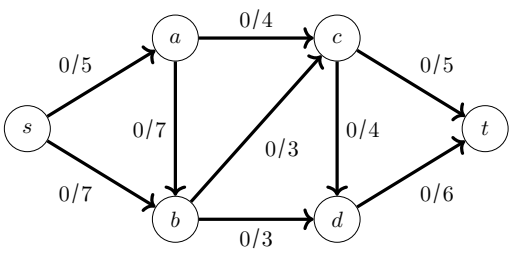
Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.

Greedy Algorithmus?

Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.

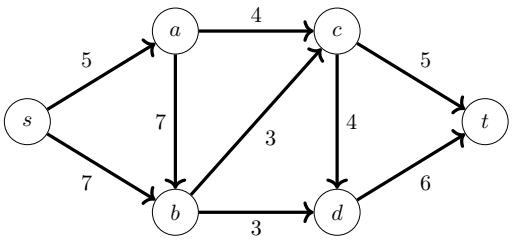


$$|f| = 0$$

Greedy Algorithmus?

Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



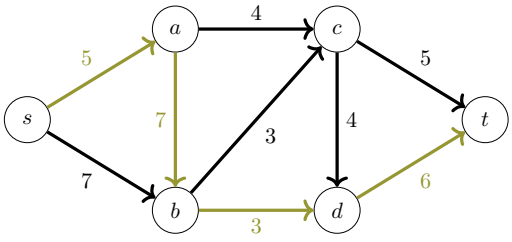
$|f| = 0$

$G_f^+ := (V, E, r := c - f)$

Greedy Algorithmus?

Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



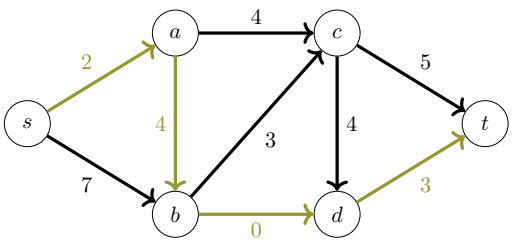
$|f| = 0$

$G_f^+ := (V, E, r := c - f)$

Greedy Algorithmus?

Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



$G_f^+ := (V, E, r := c - f)$

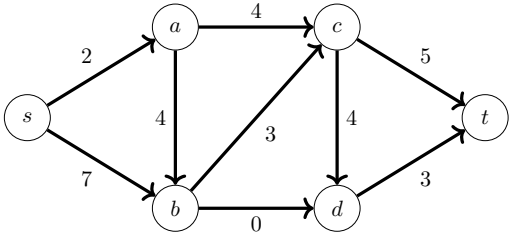
$|f| = 3$

$s \to a \to b \to d \to t$: 3

Greedy Algorithmus?

Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



$G_f^+ := (V, E, r := c - f)$

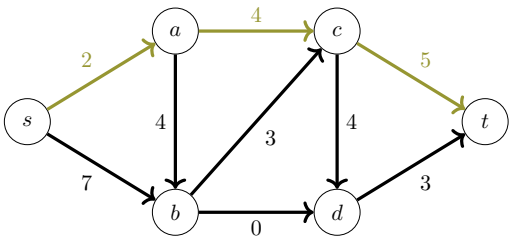
$|f| = 3$

$s \to a \to b \to d \to t$: 3

Greedy Algorithmus?

Restkapazität einer Kante e**:** $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P**:** $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



$G_f^+ := (V, E, r := c - f)$

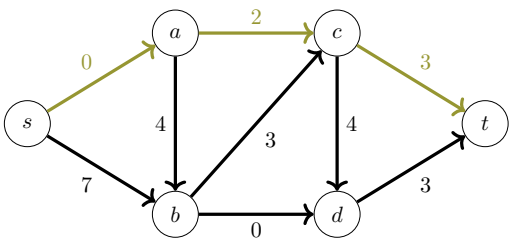
$|f| = 3$

$s \to a \to b \to d \to t$: 3

Greedy Algorithmus?

Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



$G_f^+ := (V, E, r := c - f)$

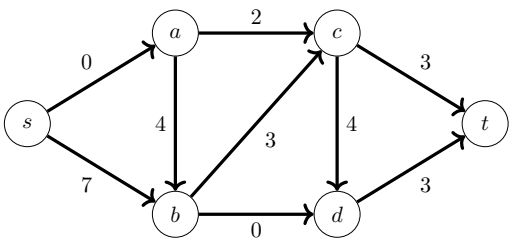
$|f| = 5$

$s \to a \to b \to d \to t$: 3
$s \to a \to c \to t$: 2

Greedy Algorithmus?

Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



$G_f^+ := (V, E, r := c - f)$

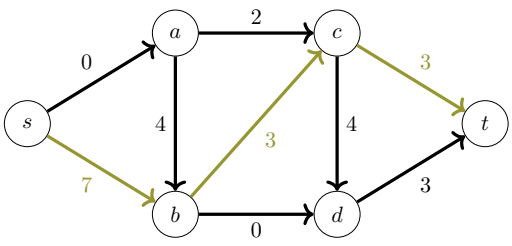
$|f| = 5$

$s \to a \to b \to d \to t$: 3
$s \to a \to c \to t$: 2

Greedy Algorithmus?

Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



$G_f^+ := (V, E, r := c - f)$

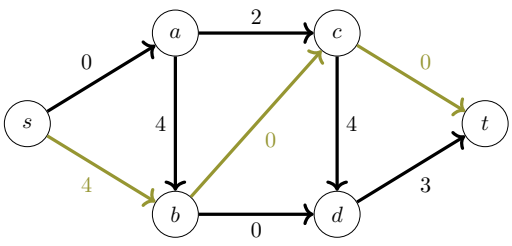
$|f| = 5$

$s \to a \to b \to d \to t$: 3
$s \to a \to c \to t$: 2

Greedy Algorithmus?

Restkapazität einer Kante e**:** $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P**:** $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



$G_f^+ := (V, E, r := c - f)$

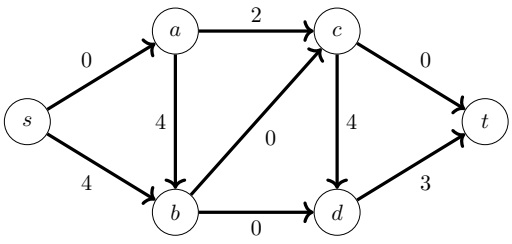
$|f| = 8$

$s \to a \to b \to d \to t$: 3
$s \to a \to c \to t$: 2
$s \to b \to c \to t$: 3

Greedy Algorithmus?

Restkapazität einer Kante e: $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P: $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



$G_f^+ := (V, E, r := c - f)$

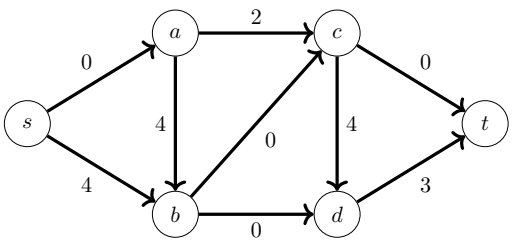
$|f| = 8$

$s \to a \to b \to d \to t$: 3
$s \to a \to c \to t$: 2
$s \to b \to c \to t$: 3

Greedy Algorithmus?

Restkapazität einer Kante e**:** $r(e) := c(e) - f(e)$
Restkapazität eines Wegs P**:** $\min_{e \in P} r(e)$

Greedy: Startend mit $f(e) = 0$ für alle $e \in E$, solange es Weg $s \rightsquigarrow t$ mit Restkapazität $d > 0$ gibt, Fluss entlang dieses Wegs um d erhöhen.



$G_f^+ := (V, E, r := c - f)$

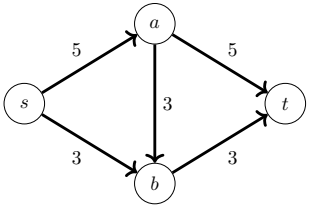
$|f| = 8$

$s \to a \to b \to d \to t$: 3
$s \to a \to c \to t$: 2
$s \to b \to c \to t$: 3

Aber $|f_{\max}| = 10$

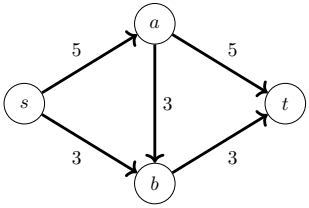
Problem mit Greedy

Problem mit Greedy

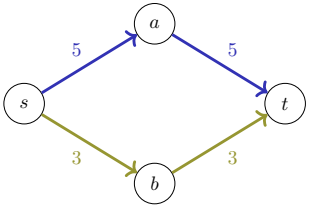


$G = (V, E, c)$

Problem mit Greedy

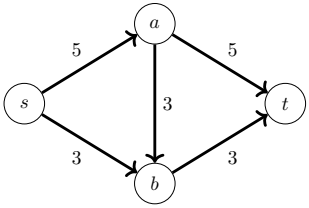


$G = (V, E, c)$

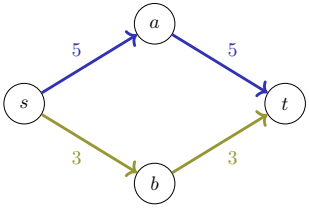


$|f_{\max}| = 8$

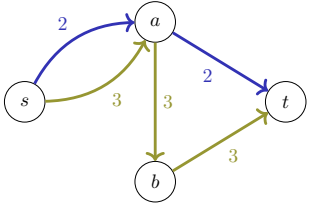
Problem mit Greedy



$G = (V, E, c)$

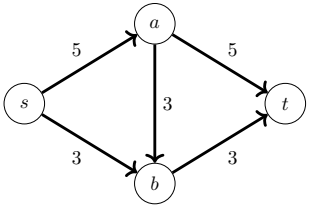


$|f_{\max}| = 8$

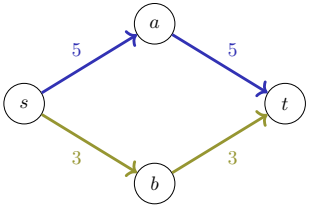


Greedy: $|f| = 5$

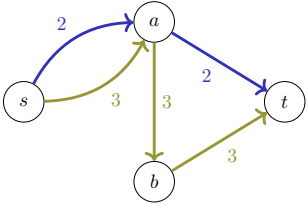
Problem mit Greedy



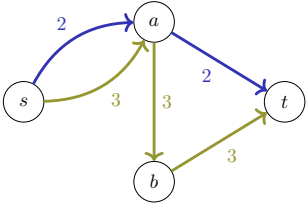
$G = (V, E, c)$



$|f_{\max}| = 8$

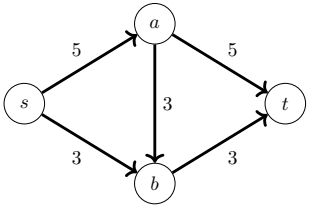


Greedy: $|f| = 5$

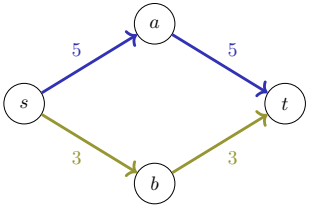


Umleitung

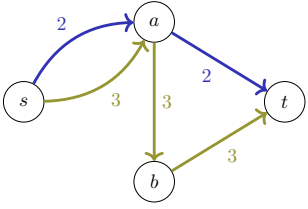
Problem mit Greedy



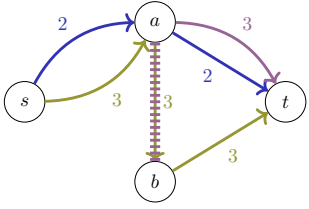
$G = (V, E, c)$



$|f_{\max}| = 8$

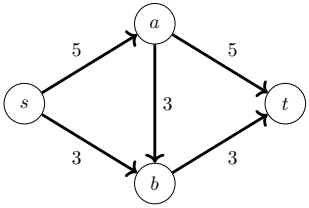


Greedy: $|f| = 5$

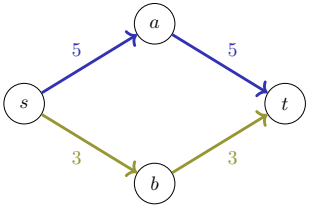


Umleitung

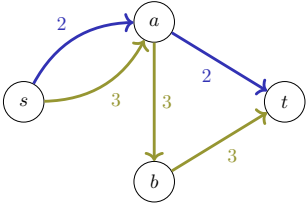
Problem mit Greedy



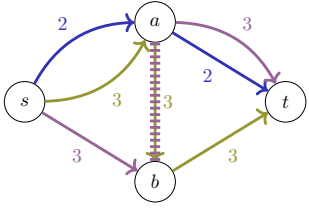
$G = (V, E, c)$



$|f_{\max}| = 8$



Greedy: $|f| = 5$



Umleitung

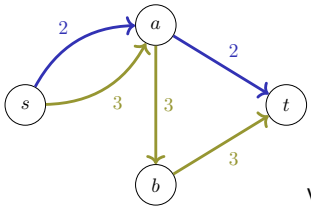
29.1 Fluss-Algorithmen

Ford-Fulkerson Algorithmus

Edmonds-Karp Algorithmus

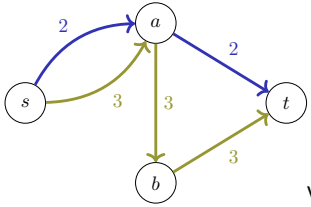
Idee: Umleitung durch Flussverringerung

Idee: Umleitung durch Flussverringerung

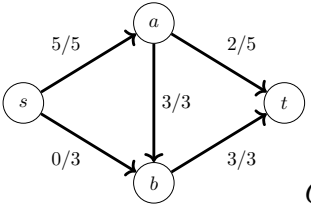


vor Umleitung

Idee: Umleitung durch Flussverringerung

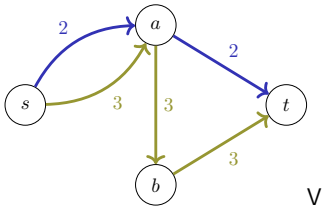


vor Umleitung

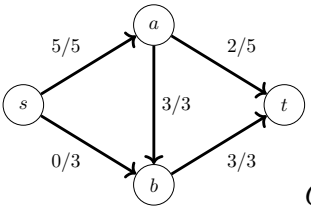


$G = (V, E, f/c)$

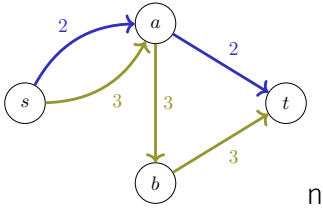
Idee: Umleitung durch Flussverringerung



vor Umleitung

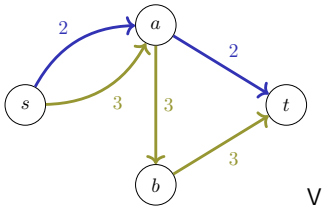


$G = (V, E, f/c)$

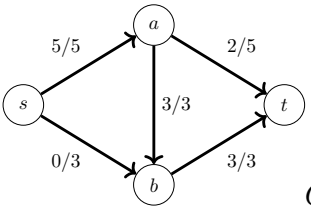


nach Umleitung

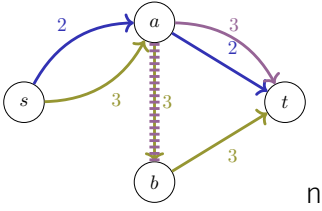
Idee: Umleitung durch Flussverringerung



vor Umleitung

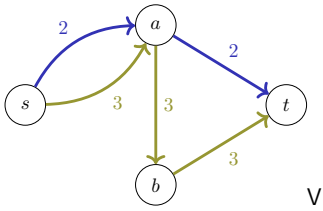


$G = (V, E, f/c)$

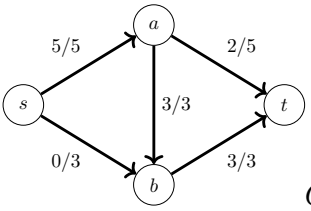


nach Umleitung

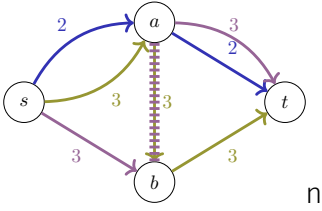
Idee: Umleitung durch Flussverringerung



vor Umleitung

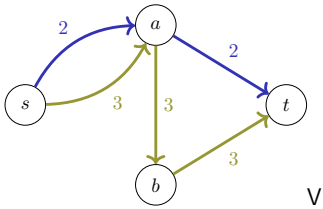


$G = (V, E, f/c)$

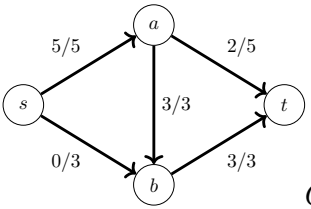


nach Umleitung

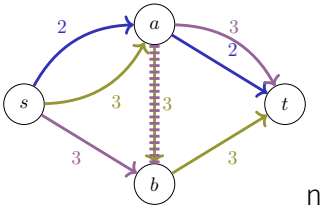
Idee: Umleitung durch Flussverringerung



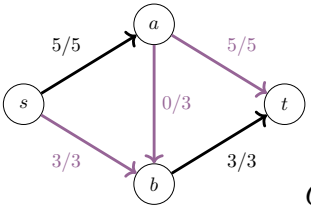
vor Umleitung



$G = (V, E, f/c)$

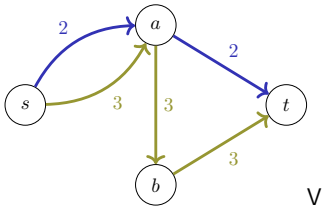


nach Umleitung

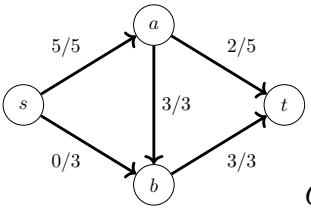


$G = (V, E, f'/c)$

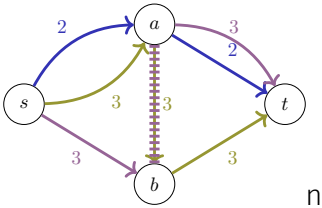
Idee: Umleitung durch Flussverringerung



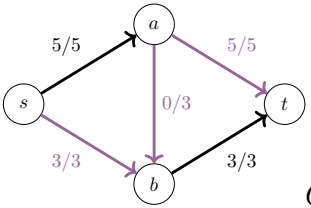
vor Umleitung



$G = (V, E, f/c)$



nach Umleitung

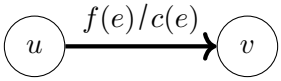


$G = (V, E, f'/c)$

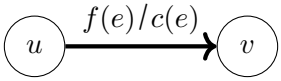
$\Rightarrow$ Umleitung entspricht Verringerung des Flusses durch Kante

Idee: Erhöhung und Verringerung

Idee: Erhöhung und Verringerung

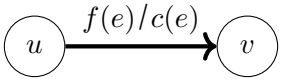


Idee: Erhöhung und Verringerung



- **Erhöhung:**
  Fluss durch e kann um höchstens $r(e) := c(e) - f(e)$ erhöht werden

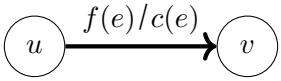
Idee: Erhöhung und Verringerung



- **Erhöhung:**
  Fluss durch e kann um höchstens $r(e) := c(e) - f(e)$ erhöht werden

$G_f^+ := (V, E, r := c - f)$

Idee: Erhöhung und Verringerung

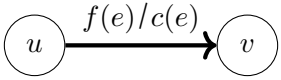


- **Erhöhung:**
  Fluss durch e kann um höchstens $r(e) := c(e) - f(e)$ erhöht werden

- **Verringerung:**
  Fluss durch e kann um höchstens $f(e)$ verringert werden

$G_f^+ := (V, E, r := c - f)$

Idee: Erhöhung und Verringerung



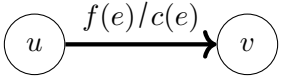
- **Erhöhung:**
  Fluss durch e kann um höchstens $r(e) := c(e) - f(e)$ erhöht werden

- **Verringerung:**
  Fluss durch e kann um höchstens $f(e)$ verringert werden

  $\Rightarrow$ Fluss durch $\overleftarrow{e}$ kann um höchstens $f(e)$ erhöht werden

$G_f^+ := (V, E, r := c - f)$

Idee: Erhöhung und Verringerung

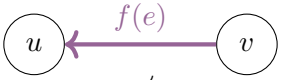


- **Erhöhung:**
  Fluss durch e kann um höchstens $r(e) := c(e) - f(e)$ erhöht werden

$G_f^+ := (V, E, r := c - f)$

- **Verringerung:**
  Fluss durch e kann um höchstens $f(e)$ verringert werden

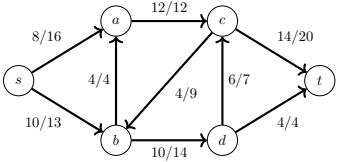
  $\Rightarrow$ Fluss durch $\overleftarrow{e}$ kann um höchstens $f(e)$ erhöht werden



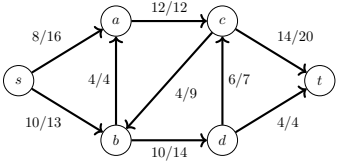
$G_f^- := (V, \overleftarrow{E}, f)$

Restnetzwerk

Restnetzwerk

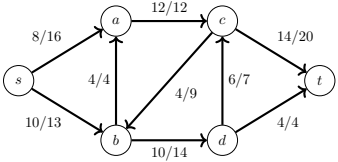


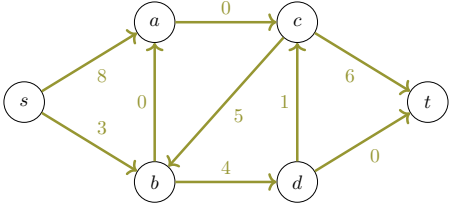
Restnetzwerk



Restnetzwerk: $G_f := \mathbf{G_f^+} \cup \mathbf{G_f^-} = (V, E_f, c_f)$

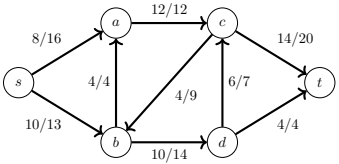
Restnetzwerk

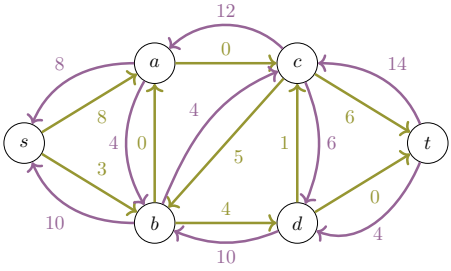




Restnetzwerk: $G_f := \mathbf{G_f^+} \cup \mathbf{G_f^-} = (V, E_f, c_f)$

Restnetzwerk

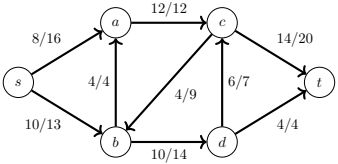


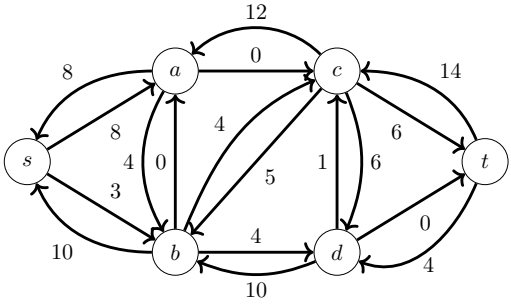


Restnetzwerk: $G_f := \mathbf{G_f^+} \cup \mathbf{G_f^-} = (V, E_f, c_f)$

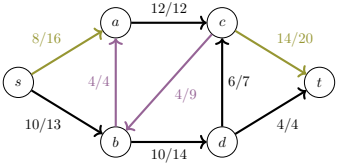
Ford-Fulkerson: Flusserweiterung im Restnetzwerk

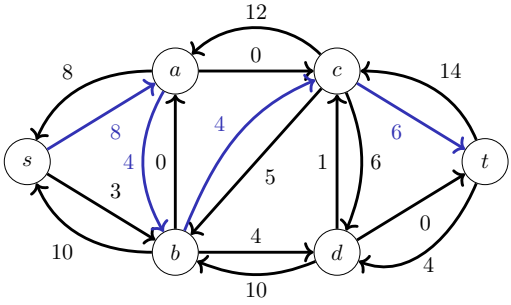
Ford-Fulkerson: Flusserweiterung im Restnetzwerk





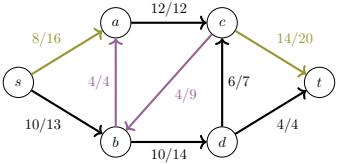
Ford-Fulkerson: Flusserweiterung im Restnetzwerk

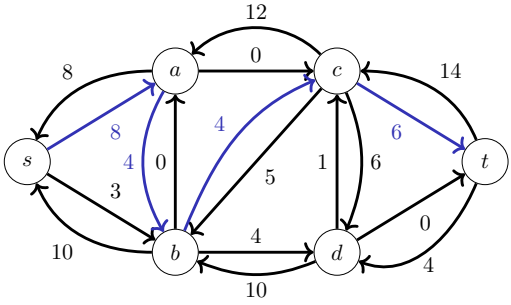




- **Erweiterungsweg:** $\mathbf{P}: \mathbf{s} \rightsquigarrow \mathbf{t}$ in G_f mit Restkapazität $d > 0$ finden

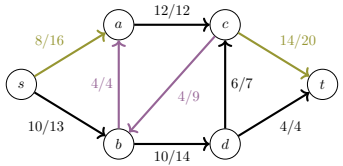
Ford-Fulkerson: Flusserweiterung im Restnetzwerk

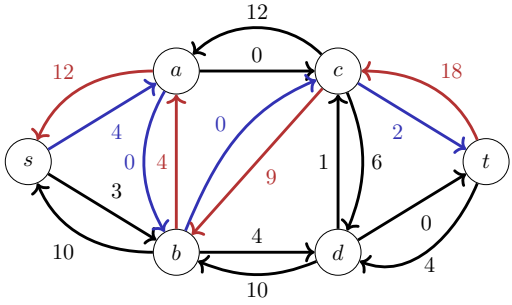




- **Erweiterungsweg:** $\mathbf{P}: \mathbf{s} \rightsquigarrow \mathbf{t}$ in G_f mit Restkapazität $d > 0$ finden
- Fluss entlang dieses Wegs für alle $e \in P$ um d erweitern:

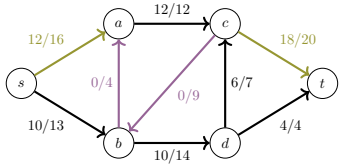
Ford-Fulkerson: Flusserweiterung im Restnetzwerk

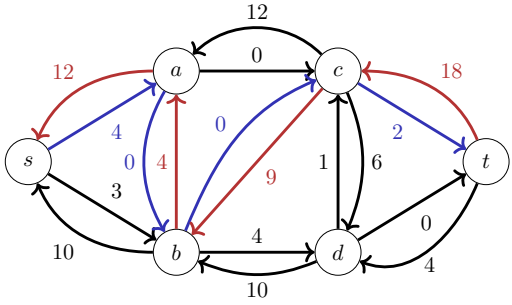




- **Erweiterungsweg:** $\mathbf{P}: \mathbf{s} \rightsquigarrow \mathbf{t}$ in G_f mit Restkapazität $d > 0$ finden
- Fluss entlang dieses Wegs für alle $e \in P$ um d erweitern:
  - Restkapazität $\mathbf{c_f(e)}$ in G_f um d verringern; $\mathbf{c_f}(\overleftarrow{e})$ um d erhöhen

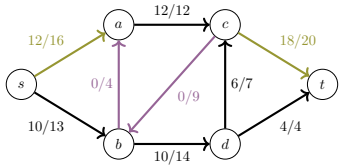
Ford-Fulkerson: Flusserweiterung im Restnetzwerk

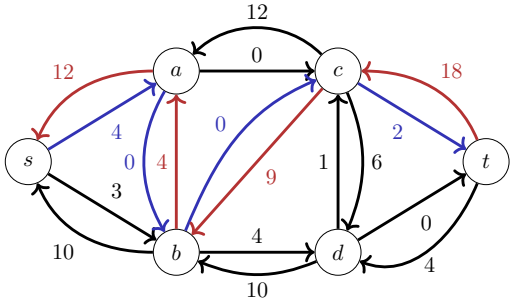




- **Erweiterungsweg:** $\mathbf{P}: \mathbf{s} \rightsquigarrow \mathbf{t}$ in G_f mit Restkapazität $d > 0$ finden
- Fluss entlang dieses Wegs für alle $e \in P$ um d erweitern:
  - Restkapazität $\mathbf{c_f(e)}$ in G_f um d verringern; $\mathbf{c_f}(\overleftarrow{e})$ um d erhöhen
  - Fluss durch $\mathbf{e} \in \mathbf{E}$ um d erhöhen; durch $\overleftarrow{e} \in \mathbf{E}$ verringern

Ford-Fulkerson: Flusserweiterung im Restnetzwerk





- **Erweiterungsweg:** $\mathbf{P}: \mathbf{s} \rightsquigarrow \mathbf{t}$ in G_f mit Restkapazität $d > 0$ finden
- Fluss entlang dieses Wegs für alle $e \in P$ um d erweitern:
  - Restkapazität $\mathbf{c_f(e)}$ in G_f um d verringern; $\mathbf{c_f}(\overleftarrow{e})$ um d erhöhen
  - Fluss durch $\mathbf{e} \in \mathbf{E}$ um d erhöhen; durch $\overleftarrow{e} \in \mathbf{E}$ verringern

$\Rightarrow$ totaler Fluss $|f|$ in G erhöht sich um d, da erste (und letzte) Kante $\in \mathbf{E}$

Algorithmus Ford-Fulkerson(G, s, t)

Input: Flussnetzwerk $G = (V, E, c)$, Quelle s, Senke t
Output: Maximaler Fluss f

Algorithmus Ford-Fulkerson(G, s, t)

Input: Flussnetzwerk $G = (V, E, c)$, Quelle s, Senke t
Output: Maximaler Fluss f

for $e \in E$ **do**
    $f(e) \leftarrow 0$

Algorithmus Ford-Fulkerson(G, s, t)

Input: Flussnetzwerk $G = (V, E, c)$, Quelle s, Senke t
Output: Maximaler Fluss f

for $e \in E$ **do**
    $f(e) \leftarrow 0$

while existiert positiver Weg $P\colon s \rightsquigarrow t$ im Restnetzwerk $G_f = (V, E_f, c_f)$ **do**

Algorithmus Ford-Fulkerson(G, s, t)

Input: Flussnetzwerk $G = (V, E, c)$, Quelle s, Senke t
Output: Maximaler Fluss f

for $e \in E$ **do**
    $f(e) \leftarrow 0$

while existiert positiver Weg $P\colon s \rightsquigarrow t$ im Restnetzwerk $G_f = (V, E_f, c_f)$ **do**
    $d \leftarrow \min_{e \in P} c_f(e)$

Algorithmus Ford-Fulkerson(G, s, t)

Input: Flussnetzwerk $G = (V, E, c)$, Quelle s, Senke t
Output: Maximaler Fluss f

for $e \in E$ **do**
  $f(e) \leftarrow 0$

while existiert positiver Weg $P\colon s \rightsquigarrow t$ im Restnetzwerk $G_f = (V, E_f, c_f)$ **do**
  $d \leftarrow \min_{e \in P} c_f(e)$
  foreach $e \in P$ **do**

Algorithmus Ford-Fulkerson(G, s, t)

Input: Flussnetzwerk $G = (V, E, c)$, Quelle s, Senke t
Output: Maximaler Fluss f

for $e \in E$ **do**
  $f(e) \leftarrow 0$

while existiert positiver Weg $P\colon s \rightsquigarrow t$ im Restnetzwerk $G_f = (V, E_f, c_f)$ **do**
  $d \leftarrow \min_{e \in P} c_f(e)$
  foreach $e \in P$ **do**
    if $e \in E$ **then**
      $f(e) \leftarrow f(e) + d$

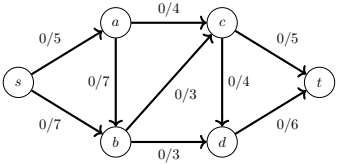
Algorithmus Ford-Fulkerson(G, s, t)

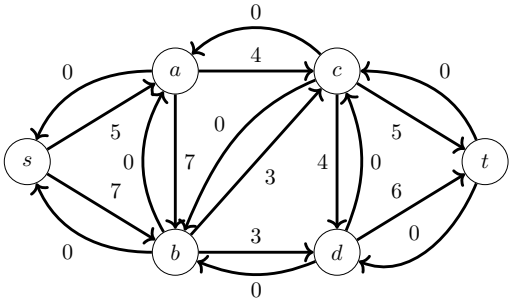
Input: Flussnetzwerk $G = (V, E, c)$, Quelle s, Senke t
Output: Maximaler Fluss f

for $e \in E$ **do**
  $f(e) \leftarrow 0$
while existiert positiver Weg $P\colon s \rightsquigarrow t$ im Restnetzwerk $G_f = (V, E_f, c_f)$ **do**
  $d \leftarrow \min_{e \in P} c_f(e)$
  foreach $e \in P$ **do**
    if $e \in E$ **then**
      $f(e) \leftarrow f(e) + d$
    else
      $f(\overleftarrow{e}) \leftarrow f(\overleftarrow{e}) - d$

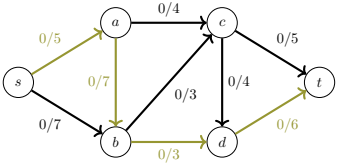
Beispiel Ford-Fulkerson

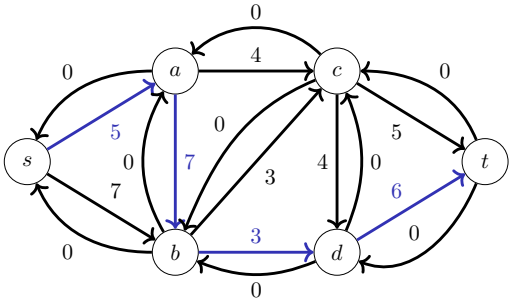
Beispiel Ford-Fulkerson



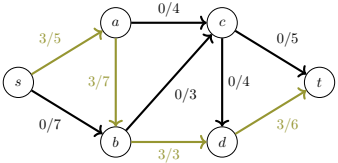


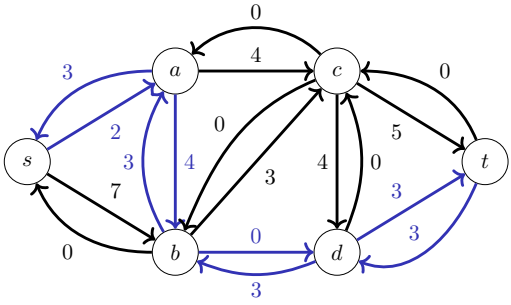
Beispiel Ford-Fulkerson



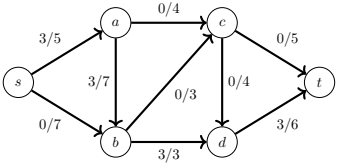


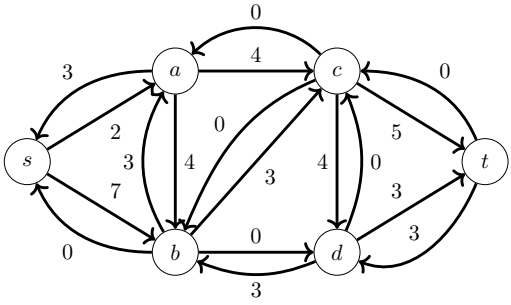
Beispiel Ford-Fulkerson



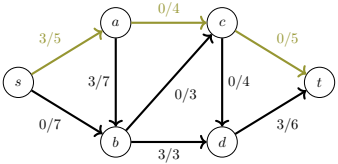


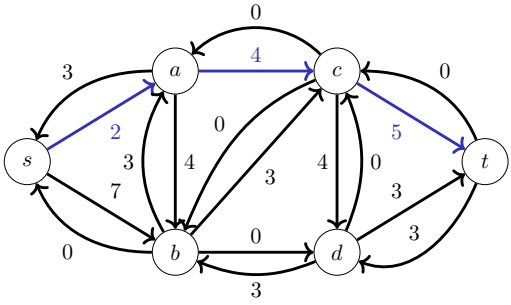
Beispiel Ford-Fulkerson



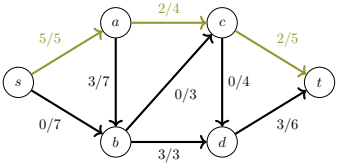


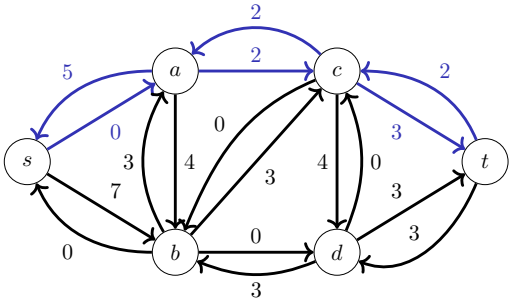
Beispiel Ford-Fulkerson



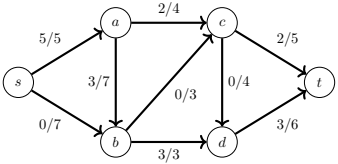


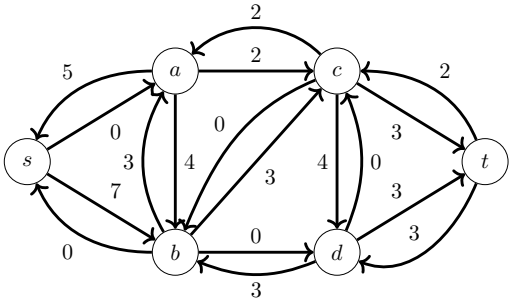
Beispiel Ford-Fulkerson



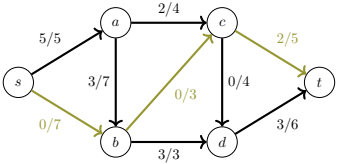


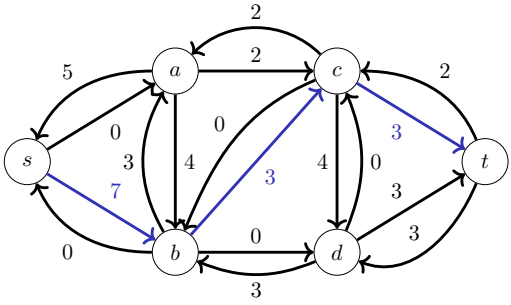
Beispiel Ford-Fulkerson



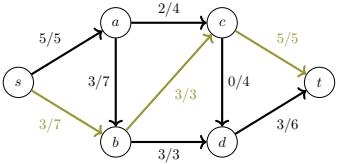


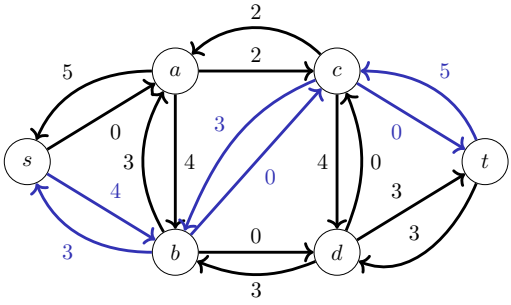
Beispiel Ford-Fulkerson



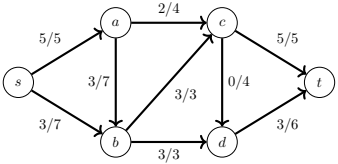


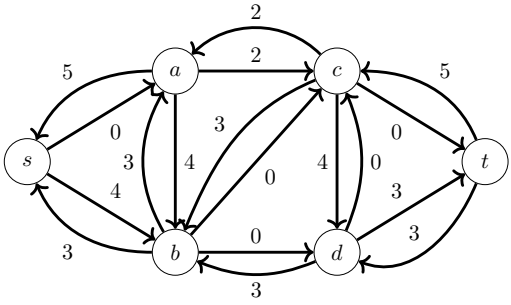
Beispiel Ford-Fulkerson



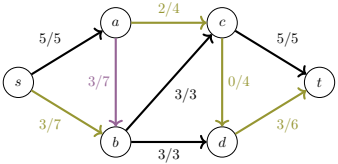


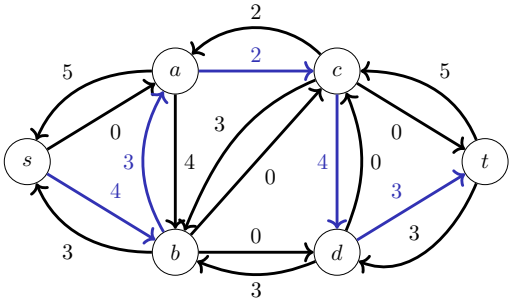
Beispiel Ford-Fulkerson



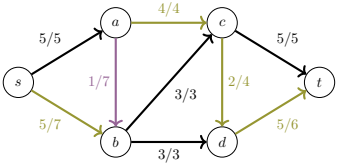


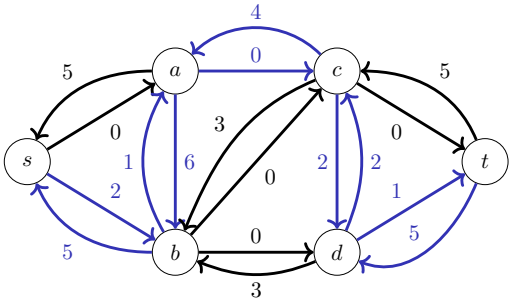
Beispiel Ford-Fulkerson



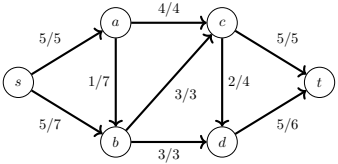


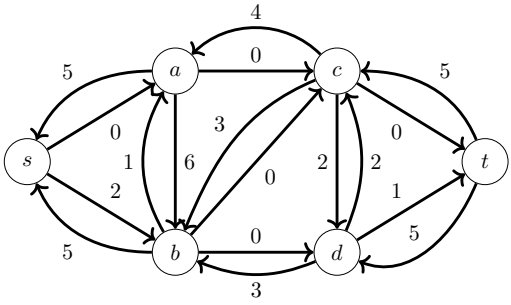
Beispiel Ford-Fulkerson



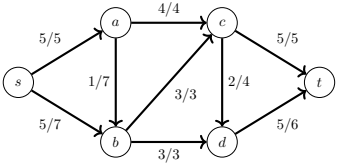


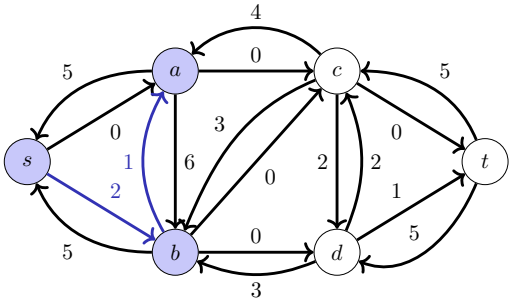
Beispiel Ford-Fulkerson





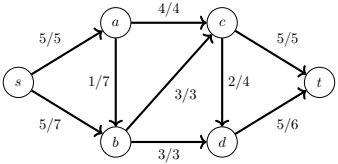
Beispiel Ford-Fulkerson

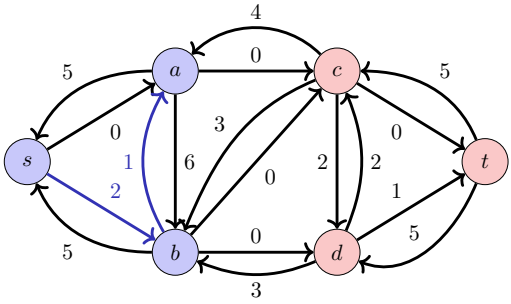




Knoten $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s

Beispiel Ford-Fulkerson

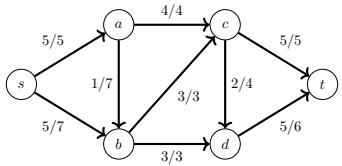


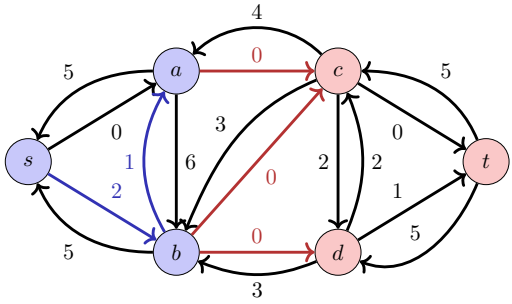


Knoten $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s

Knoten $\mathbf{T} \subseteq \mathbf{V}$ nicht erreichbar von s

Beispiel Ford-Fulkerson



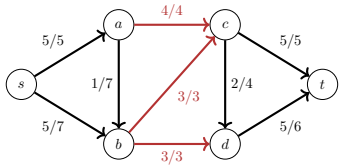


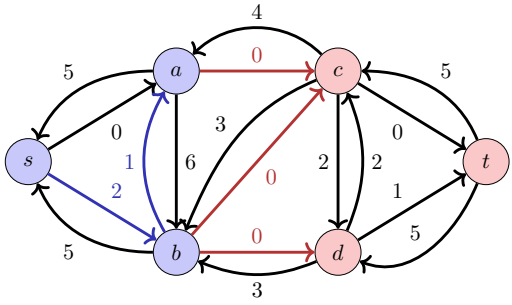
Knoten $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s

Knoten $\mathbf{T} \subseteq \mathbf{V}$ nicht erreichbar von s

alle ausgehenden Kanten haben Restkapazität 0 in G_f

Beispiel Ford-Fulkerson



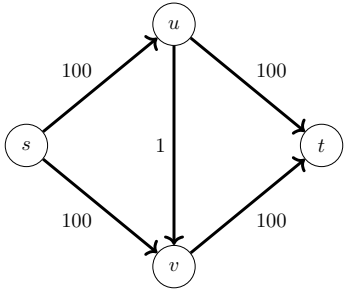


Knoten $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s
Knoten $\mathbf{T} \subseteq \mathbf{V}$ nicht erreichbar von s

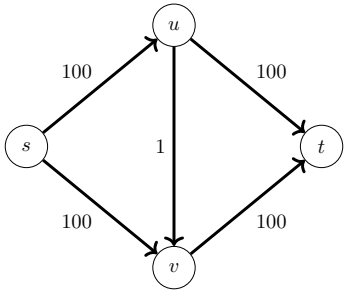
alle ausgehenden Kanten haben Restkapazität 0 in G_f
$\Rightarrow$ Fluss schöpft Kapazität auf diesen Kanten voll aus!

Quiz Ford-Fulkerson

Quiz Ford-Fulkerson



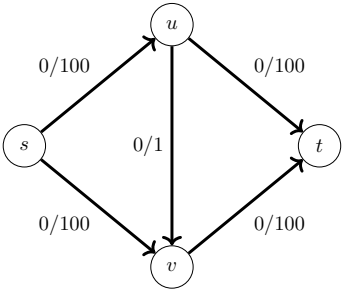
Quiz Ford-Fulkerson



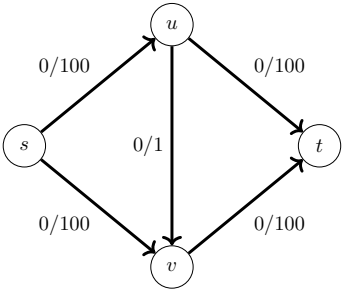
Wie viele Iterationen braucht Ford-Fulkerson im schlimmsten Fall?

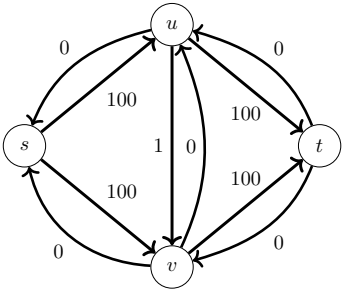
Lösung

Lösung

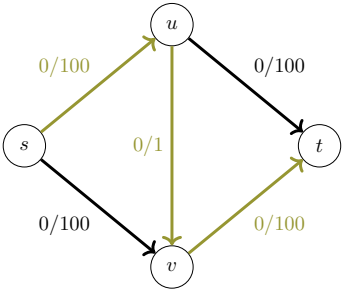


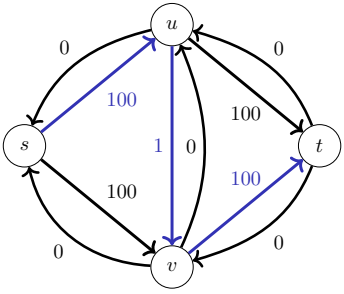
Lösung



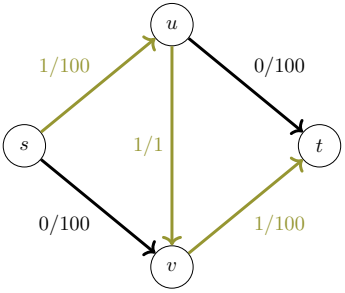


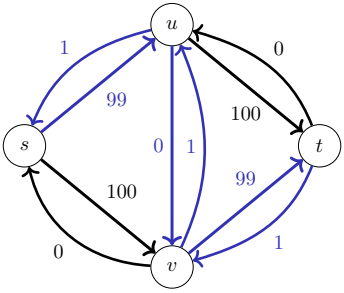
Lösung



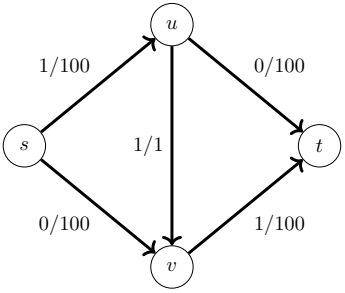


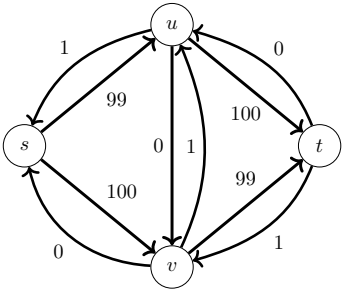
Lösung



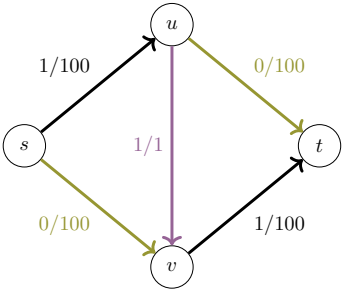


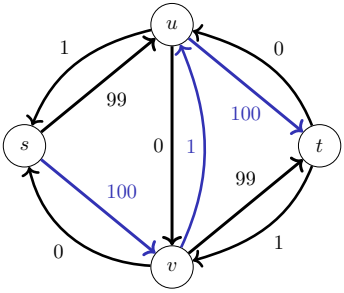
Lösung



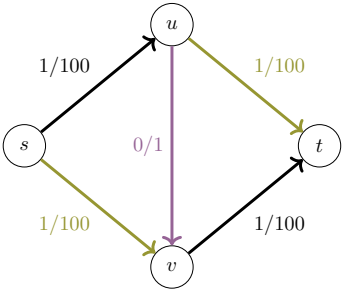


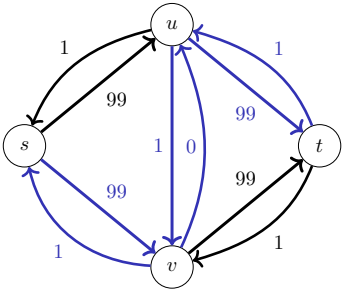
Lösung



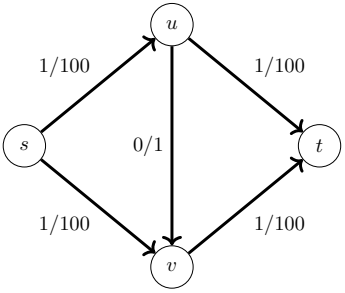


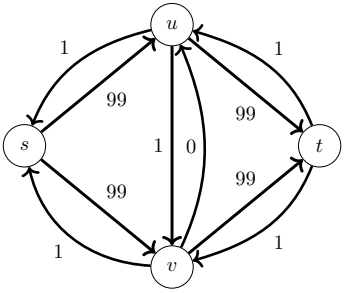
Lösung



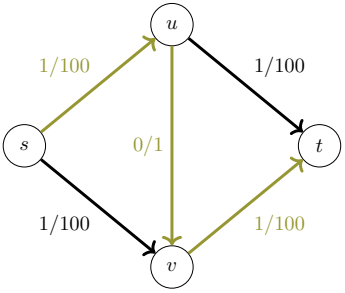


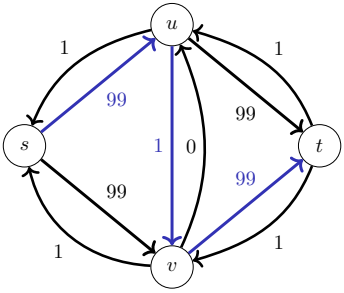
Lösung



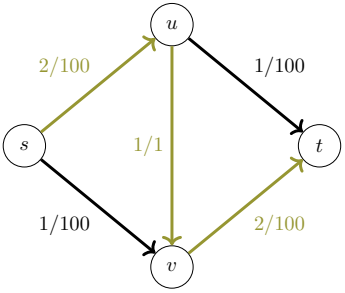


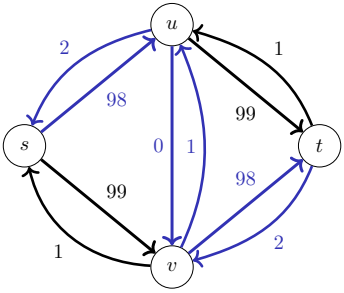
Lösung



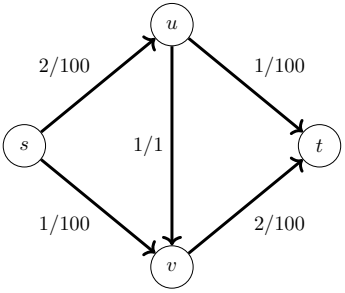


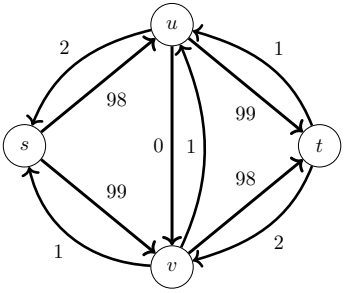
Lösung



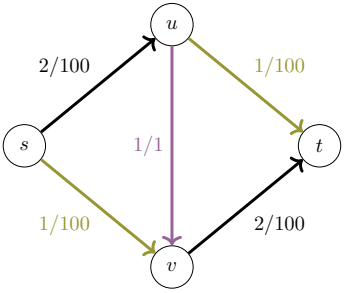


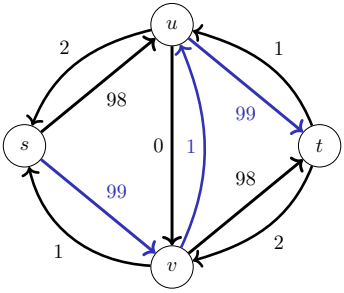
Lösung



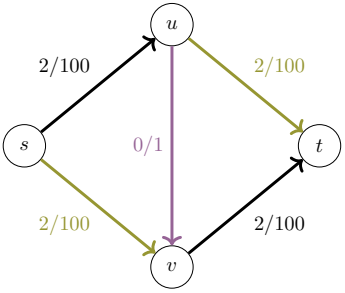


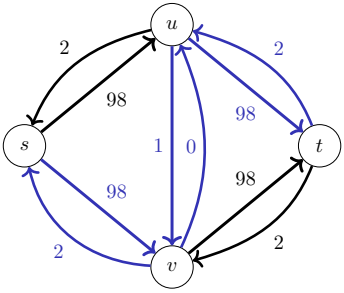
Lösung



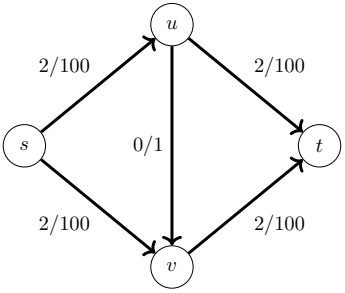


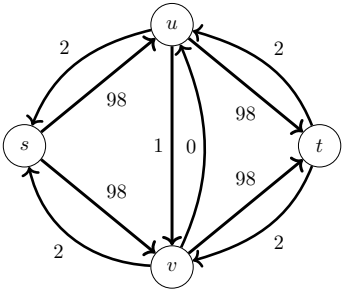
Lösung



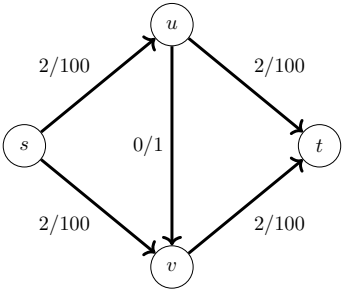


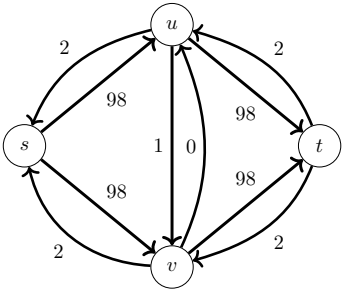
Lösung





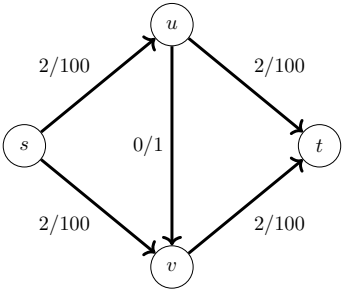
Lösung

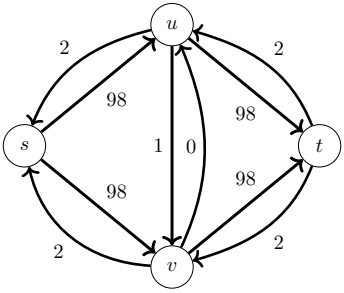




Nach i Iterationen: $|f| = i$

Lösung

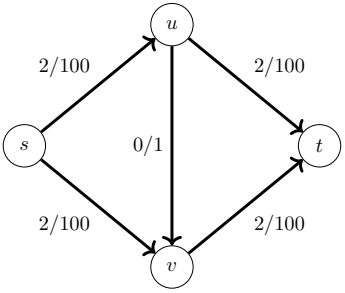


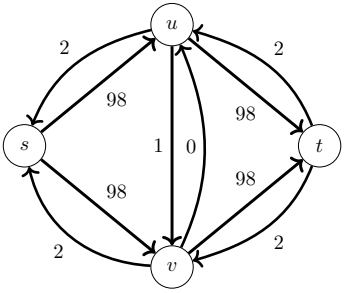


Nach i Iterationen: $|f| = i$

$\Rightarrow$ insgesamt $|f_{\max}|$ Iterationen

Lösung





Nach i Iterationen: $|f| = i$

$\Rightarrow$ insgesamt $|f_{\max}| = 200$ Iterationen

Laufzeit-Analyse von Ford-Fulkerson

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration:

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$

($|V| \leq |E|$, da man alle nicht erreichbaren Knoten ignorieren kann.)

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$

($|V| \leq |E|$, da man alle nicht erreichbaren Knoten ignorieren kann.)

Anzahl Iterationen:

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$

($|V| \leq |E|$, da man alle nicht erreichbaren Knoten ignorieren kann.)

Anzahl Iterationen:

In jedem Schritt erhöht sich der Grösse des Flusses $|f|$ um $d > 0$.

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$

($|V| \leq |E|$, da man alle nicht erreichbaren Knoten ignorieren kann.)

Anzahl Iterationen:

In jedem Schritt erhöht sich der Grösse des Flusses $|f|$ um $d > 0$.

ganzzahlige Kapazitäten

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$

($|V| \leq |E|$, da man alle nicht erreichbaren Knoten ignorieren kann.)

Anzahl Iterationen:

In jedem Schritt erhöht sich der Grösse des Flusses $|f|$ um $d > 0$.

ganzzahlige Kapazitäten $\Rightarrow$ Erhöhung um ≥ 1

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$

($|V| \leq |E|$, da man alle nicht erreichbaren Knoten ignorieren kann.)

Anzahl Iterationen:

In jedem Schritt erhöht sich der Grösse des Flusses $|f|$ um $d > 0$.

ganzzahlige Kapazitäten $\Rightarrow$ Erhöhung um $\geq 1 \Rightarrow$ höchstens $|f_{\max}|$ Iterationen

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$

($|V| \leq |E|$, da man alle nicht erreichbaren Knoten ignorieren kann.)

Anzahl Iterationen:

In jedem Schritt erhöht sich der Grösse des Flusses $|f|$ um $d > 0$.

ganzzahlige Kapazitäten $\Rightarrow$ Erhöhung um $\geq 1 \Rightarrow$ höchstens $|f_{\max}|$ Iterationen

$\Rightarrow \mathcal{O}(|f_{\max}| \cdot |E|)$ für Flussnetzwerke $G = (V, E, c)$ mit $c\colon E \to \mathbb{N}^{\geq 1}$

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$

($|V| \leq |E|$, da man alle nicht erreichbaren Knoten ignorieren kann.)

Anzahl Iterationen:

In jedem Schritt erhöht sich der Grösse des Flusses $|f|$ um $d > 0$.

ganzzahlige Kapazitäten $\Rightarrow$ Erhöhung um $\geq 1 \Rightarrow$ höchstens $|f_{\max}|$ Iterationen

$\Rightarrow \mathcal{O}(|f_{\max}| \cdot |E|)$ für Flussnetzwerke $G = (V, E, c)$ mit $c\colon E \to \mathbb{N}^{\geq 1}$

Edmonds-Karp Algorithmus: (Variante von Ford-Fulkerson)

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$

$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$

($|V| \leq |E|$, da man alle nicht erreichbaren Knoten ignorieren kann.)

Anzahl Iterationen:

In jedem Schritt erhöht sich der Grösse des Flusses $|f|$ um $d > 0$.

ganzzahlige Kapazitäten $\Rightarrow$ Erhöhung um $\geq 1 \Rightarrow$ höchstens $|f_{\max}|$ Iterationen

$\Rightarrow \mathcal{O}(|f_{\max}| \cdot |E|)$ für Flussnetzwerke $G = (V, E, c)$ mit $c\colon E \to \mathbb{N}^{\geq 1}$

Edmonds-Karp Algorithmus: (Variante von Ford-Fulkerson)

kürzester Erweiterungsweg (Anzahl Kanten)

Laufzeit-Analyse von Ford-Fulkerson

Zeit pro Iteration: Suche eines Erweiterungswegs $s \rightsquigarrow t$
$\Rightarrow$ Tiefensuche oder Breitensuche: $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$
($|V| \leq |E|$, da man alle nicht erreichbaren Knoten ignorieren kann.)

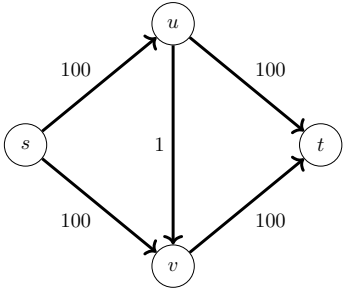
Anzahl Iterationen:
In jedem Schritt erhöht sich der Grösse des Flusses $|f|$ um $d > 0$.
ganzzahlige Kapazitäten $\Rightarrow$ Erhöhung um $\geq 1 \Rightarrow$ höchstens $|f_{\max}|$ Iterationen

$\Rightarrow \mathcal{O}(|f_{\max}| \cdot |E|)$ für Flussnetzwerke $G = (V, E, c)$ mit $c\colon E \to \mathbb{N}^{\geq 1}$

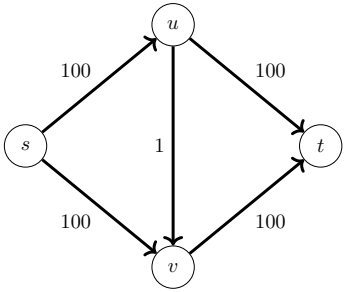
Edmonds-Karp Algorithmus: (Variante von Ford-Fulkerson)
kürzester Erweiterungsweg (Anzahl Kanten) $\Rightarrow \mathcal{O}(|V| \cdot |E|^2)$ (ohne Herleitung)

Quiz Edmonds-Karp

Quiz Edmonds-Karp



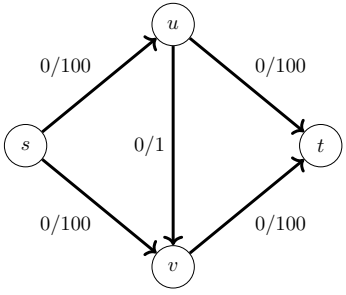
Quiz Edmonds-Karp



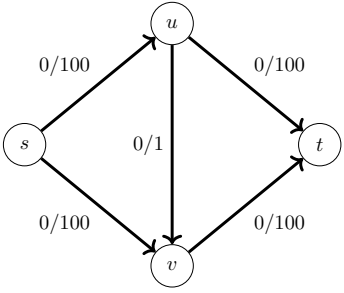
Wie viele Iterationen braucht Edmonds-Karp im schlimmsten Fall?

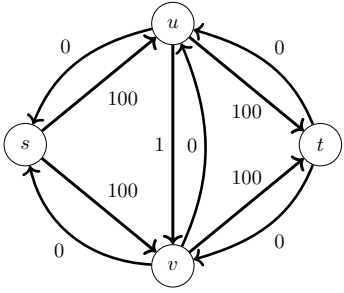
Lösung

Lösung

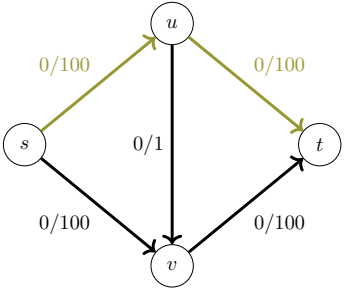


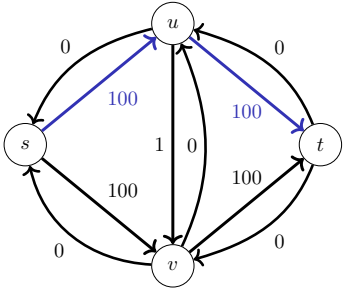
Lösung



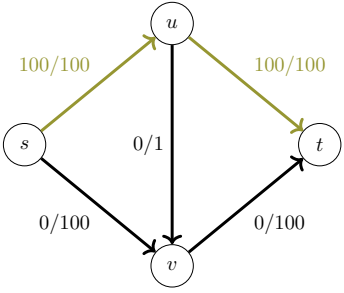


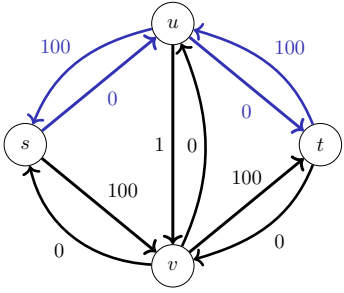
Lösung



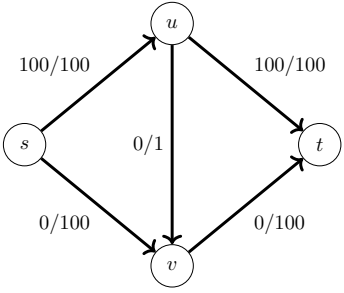


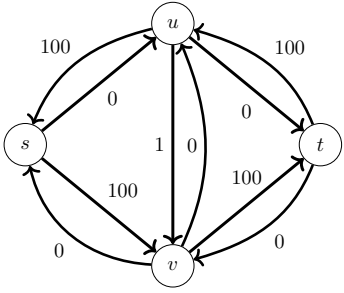
Lösung



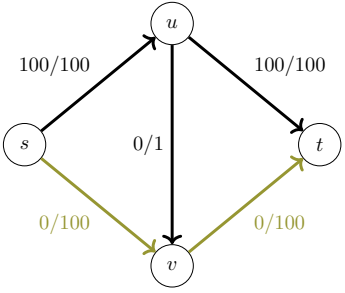


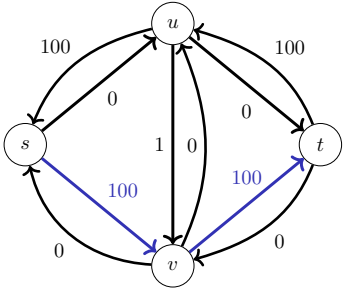
Lösung



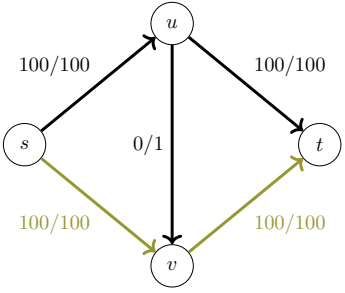


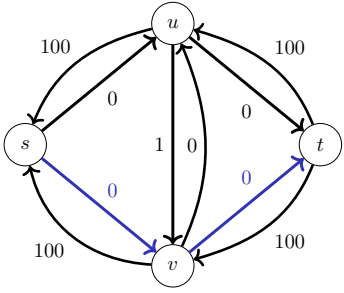
Lösung



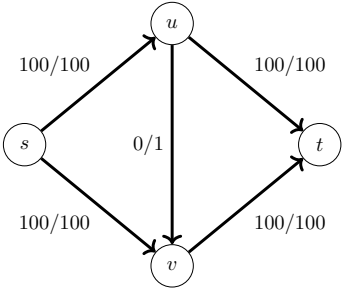


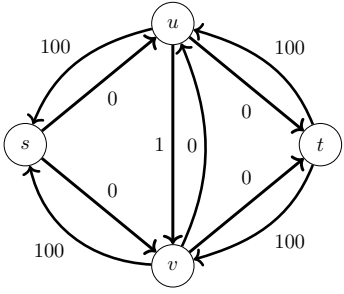
Lösung


u
100/100
100/100
s
0/1
t
100/100
100/100
v


u
100
100
0
0
s
1
0
t
0
0
100
100
v

Lösung



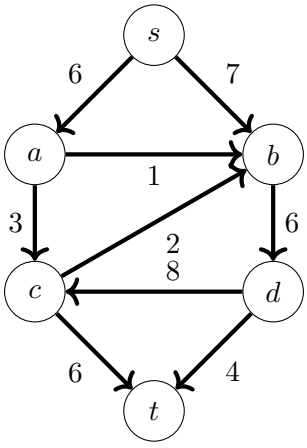


Termination nach 2 Iterationen!

29.2 Max-Flow Min-Cut

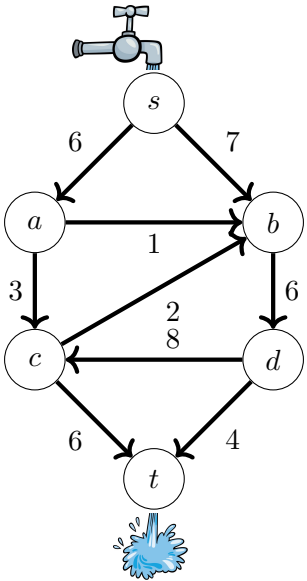
Flüsse und Schnitte: Bottleneck-Intuition

Flüsse und Schnitte: Bottleneck-Intuition



Flüsse und Schnitte: Bottleneck-Intuition

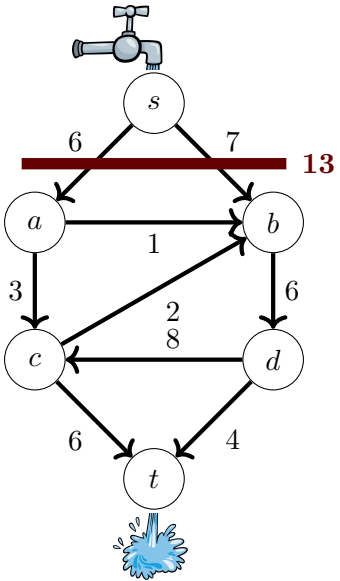
Obere Schranken für Flussgrösse:



Flüsse und Schnitte: Bottleneck-Intuition

Obere Schranken für Flussgrösse:

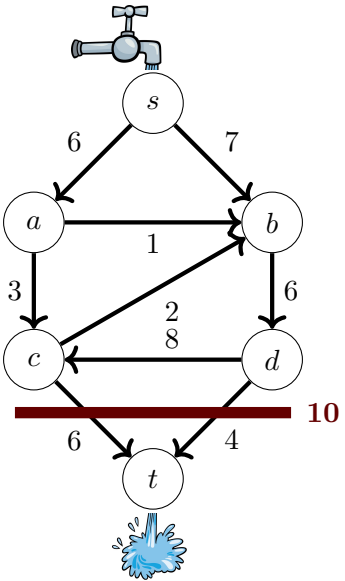
- was aus s fliessen kann: $c^+(s)$



Flüsse und Schnitte: Bottleneck-Intuition

Obere Schranken für Flussgrösse:

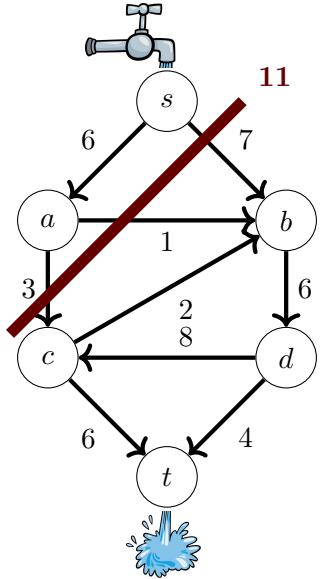
- was aus s fliessen kann: $c^+(s)$
- was in t fliessen kann: $c^-(t)$



Flüsse und Schnitte: Bottleneck-Intuition

Obere Schranken für Flussgrösse:

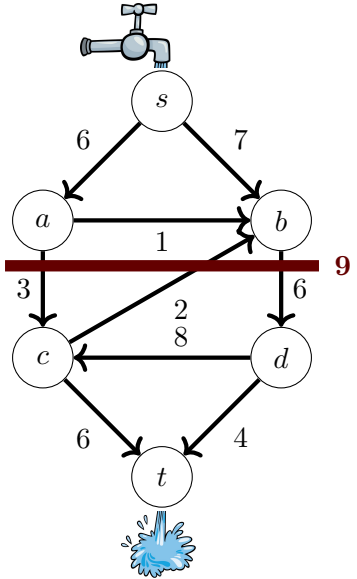
- was aus s fliessen kann: $c^+(s)$
- was in t fliessen kann: $c^-(t)$
- was durch beliebigen Schnitt fliessen kann



Flüsse und Schnitte: Bottleneck-Intuition

Obere Schranken für Flussgrösse:

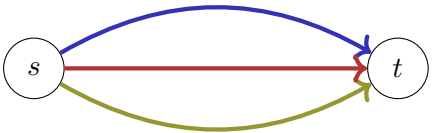
- was aus s fliessen kann: $c^+(s)$
- was in t fliessen kann: $c^-(t)$
- was durch beliebigen Schnitt fliessen kann
- was durch Bottleneck fliessen kann: $c_{\min}$

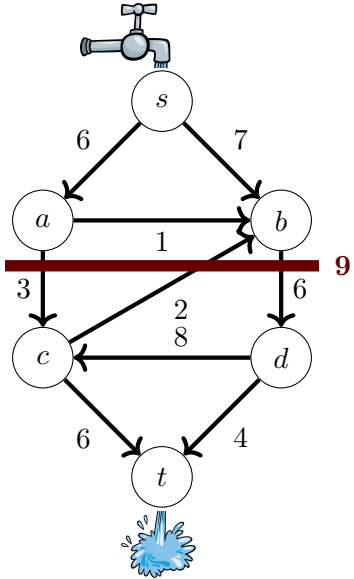


Flüsse und Schnitte: Bottleneck-Intuition

Obere Schranken für Flussgrösse:

- was aus s fliessen kann: $c^+(s)$
- was in t fliessen kann: $c^-(t)$
- was durch beliebigen Schnitt fliessen kann
- was durch Bottleneck fliessen kann: $c_{\min}$

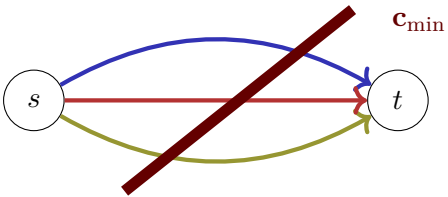


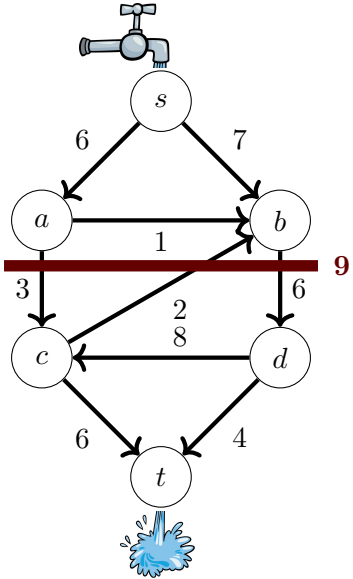


Flüsse und Schnitte: Bottleneck-Intuition

Obere Schranken für Flussgrösse:

- was aus s fliessen kann: $c^+(s)$
- was in t fliessen kann: $c^-(t)$
- was durch beliebigen Schnitt fliessen kann
- was durch Bottleneck fliessen kann: $c_{\min}$

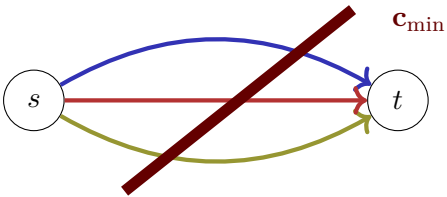




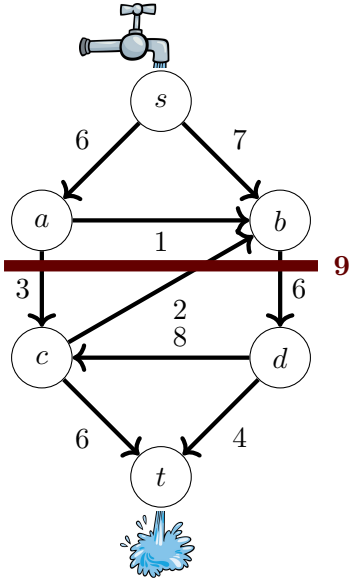
Flüsse und Schnitte: Bottleneck-Intuition

Obere Schranken für Flussgrösse:

- was aus s fliessen kann: $c^+(s)$
- was in t fliessen kann: $c^-(t)$
- was durch beliebigen Schnitt fliessen kann
- was durch Bottleneck fliessen kann: $c_{\min}$



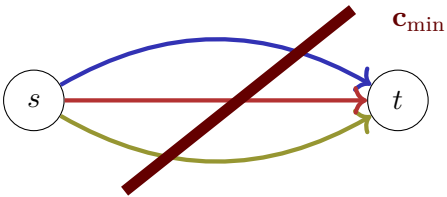
$\Rightarrow$ Fluss $|f| \leq$ Bottleneck



Flüsse und Schnitte: Bottleneck-Intuition

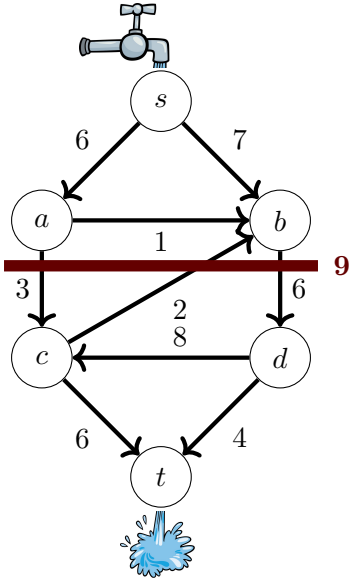
Obere Schranken für Flussgrösse:

- was aus s fliessen kann: $c^+(s)$
- was in t fliessen kann: $c^-(t)$
- was durch beliebigen Schnitt fliessen kann
- was durch Bottleneck fliessen kann: $c_{\min}$



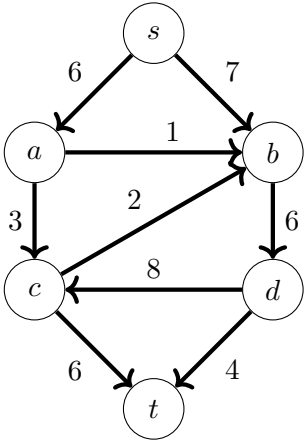
$\Rightarrow$ Fluss $|f| \leq$ Bottleneck

$\Rightarrow$ Maximaler Fluss $\leq$ Bottleneck



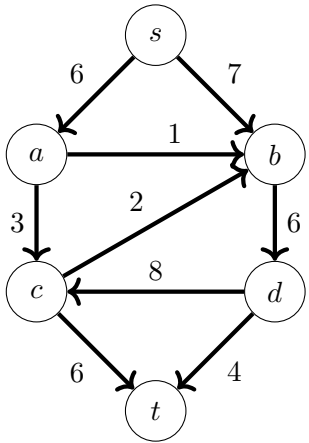
Schnitt

Schnitt



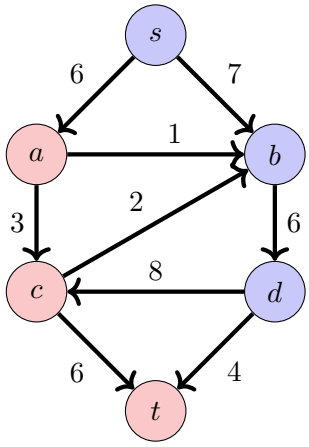
Schnitt

(s,t)**-Schnitt** von Graph $G = (V, E, c)$:



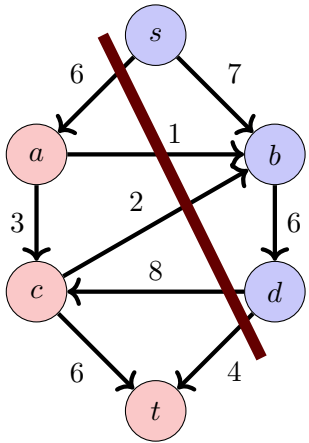
Schnitt

(s,t)**-Schnitt** von Graph $G = (V, E, c)$:
Partition $(\mathbf{S}, \mathbf{T})$ von V so dass $s \in S, t \in T$



Schnitt

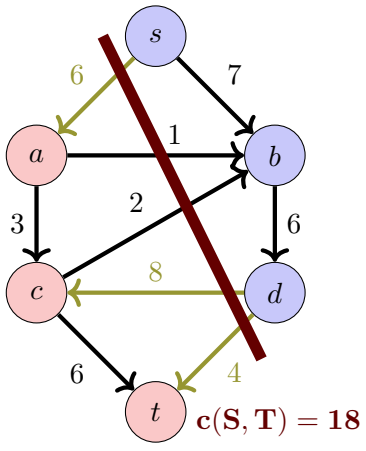
(s,t)**-Schnitt** von Graph $G = (V, E, c)$:
Partition $(\mathbf{S}, \mathbf{T})$ von V so dass $s \in S, t \in T$



Schnitt

(s,t)**-Schnitt** von Graph $G = (V, E, c)$:
Partition $(\mathbf{S}, \mathbf{T})$ von V so dass $s \in S, t \in T$

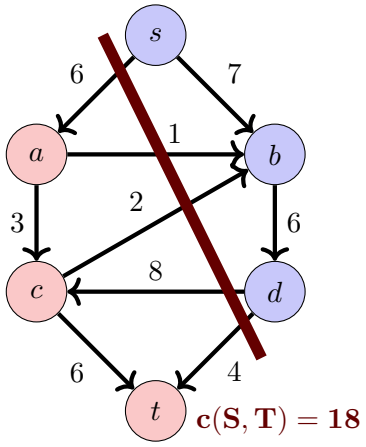
Grösse des Schnitts:
$c(S,T) := \sum_{\mathbf{e}:\ \mathbf{S}\to\mathbf{T}} \mathbf{c(e)}$



Schnitt

(s,t)**-Schnitt** von Graph $G = (V, E, c)$:
Partition $(\mathbf{S}, \mathbf{T})$ von V so dass $s \in S, t \in T$

Grösse des Schnitts:
$c(S,T) := \sum_{\mathbf{e}:\ \mathbf{S}\to\mathbf{T}} \mathbf{c(e)}$

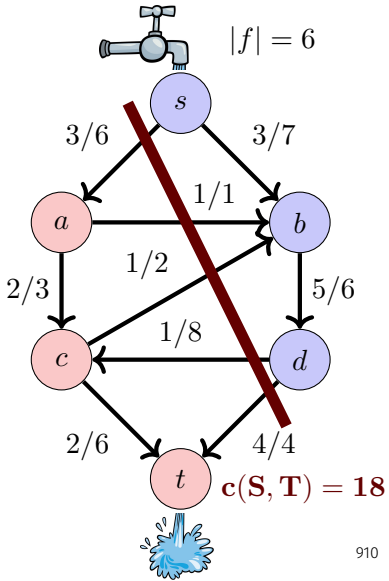


Schnitt

(s,t)**-Schnitt** von Graph $G = (V, E, c)$:
Partition $(\mathbf{S}, \mathbf{T})$ von V so dass $s \in S, t \in T$

Grösse des Schnitts:
$c(S,T) := \sum_{\mathbf{e}:\ \mathbf{S}\to\mathbf{T}} \mathbf{c(e)}$

Fluss durch Schnitt von Flussnetzwerk:

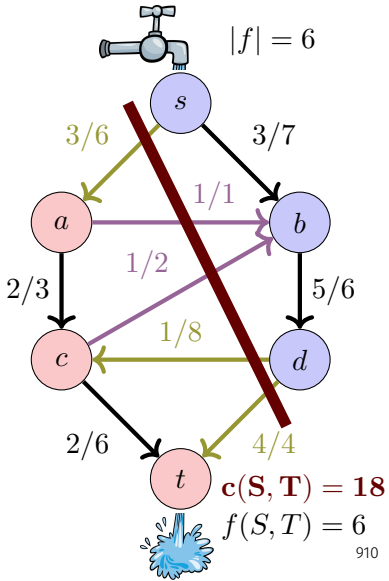


Schnitt

(s,t)**-Schnitt** von Graph $G = (V, E, c)$:
Partition $(\mathbf{S}, \mathbf{T})$ von V so dass $s \in S, t \in T$

Grösse des Schnitts:
$c(S,T) := \sum_{\mathbf{e}:\ \mathbf{S}\to\mathbf{T}} \mathbf{c(e)}$

Fluss durch Schnitt von Flussnetzwerk:
$f(S,T) := \sum_{\mathbf{e}:\ \mathbf{S}\to\mathbf{T}} \mathbf{f(e)} - \sum_{\mathbf{e}:\ \mathbf{T}\to\mathbf{S}} \mathbf{f(e)}$

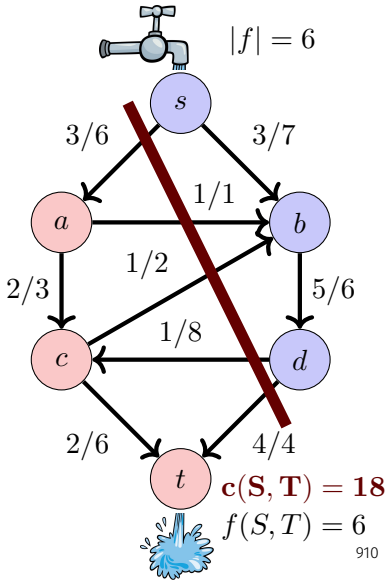


Schnitt

(s,t)**-Schnitt** von Graph $G = (V, E, c)$:
Partition $(\mathbf{S}, \mathbf{T})$ von V so dass $s \in S, t \in T$

Grösse des Schnitts:
$c(S,T) := \sum_{\mathbf{e}:\ \mathbf{S}\to\mathbf{T}} \mathbf{c(e)}$

Fluss durch Schnitt von Flussnetzwerk:
$f(S,T) := \sum_{\mathbf{e}:\ \mathbf{S}\to\mathbf{T}} \mathbf{f(e)} - \sum_{\mathbf{e}:\ \mathbf{T}\to\mathbf{S}} \mathbf{f(e)}$



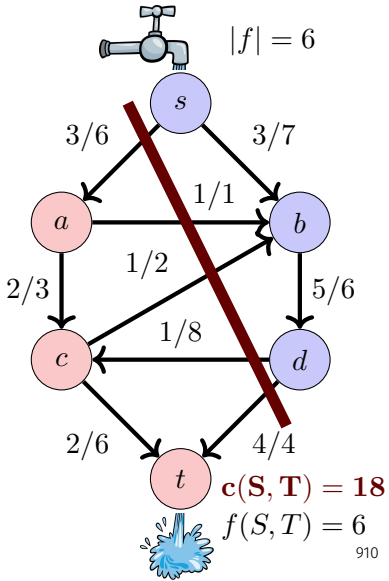
Schnitt

(s,t)**-Schnitt** von Graph $G = (V, E, c)$:
Partition $(\mathbf{S}, \mathbf{T})$ von V so dass $s \in S, t \in T$

Grösse des Schnitts:
$c(S,T) := \sum_{\mathbf{e}:\, \mathbf{S}\to\mathbf{T}} \mathbf{c(e)}$

Fluss durch Schnitt von Flussnetzwerk:
$f(S,T) := \sum_{\mathbf{e}:\, \mathbf{S}\to\mathbf{T}} \mathbf{f(e)} - \sum_{\mathbf{e}:\, \mathbf{T}\to\mathbf{S}} \mathbf{f(e)}$

Beobachtung:
$\forall f, S, T\colon |f| = f(S,T) \leq c(S,T)$



Schnitt

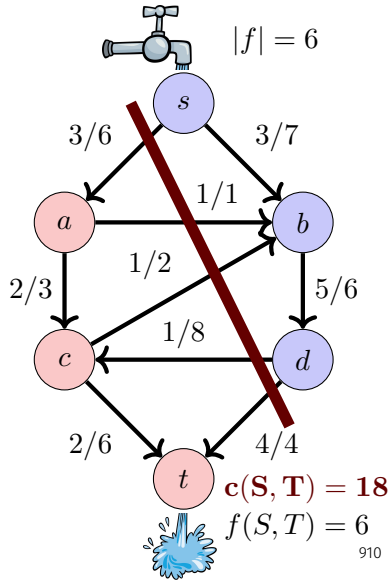
(s,t)**-Schnitt** von Graph $G=(V,E,c)$:
Partition $(\mathbf{S},\mathbf{T})$ von V so dass $s\in S, t\in T$

Grösse des Schnitts:
$c(S,T) := \sum_{\mathbf{e}:\ \mathbf{S}\to\mathbf{T}} \mathbf{c(e)}$

Fluss durch Schnitt von Flussnetzwerk:
$f(S,T) := \sum_{\mathbf{e}:\ \mathbf{S}\to\mathbf{T}} \mathbf{f(e)} - \sum_{\mathbf{e}:\ \mathbf{T}\to\mathbf{S}} \mathbf{f(e)}$

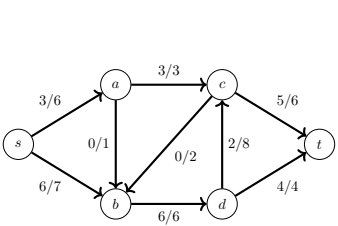
Beobachtung:
$\forall f,S,T\colon |f| = f(S,T) \le c(S,T)$

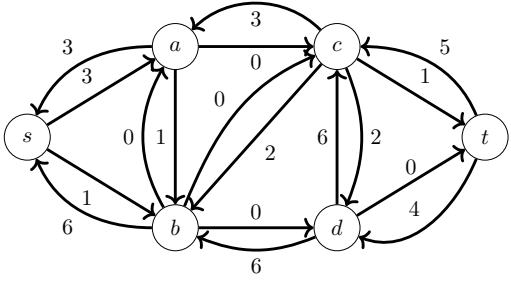
$\Rightarrow |f_{\max}| \le c_{\min}$



Maximaler Fluss und Minimaler Schnitt

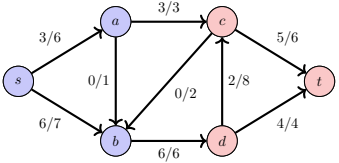
Maximaler Fluss und Minimaler Schnitt

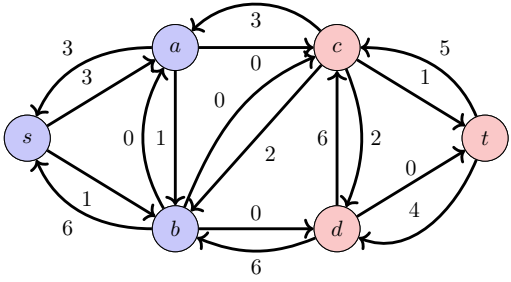




nach Terminierung von Ford-Fulkerson/Edmonds-Karp:

Maximaler Fluss und Minimaler Schnitt

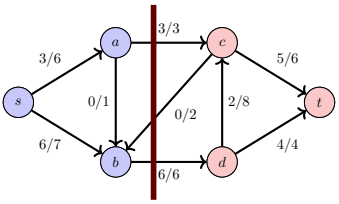


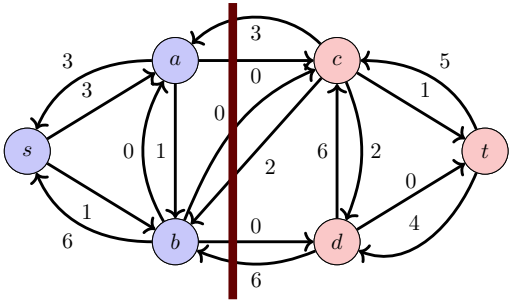


nach Terminierung von Ford-Fulkerson/Edmonds-Karp:

- $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s, $\mathbf{T} \subseteq \mathbf{V}$ nicht erreichbar von s

Maximaler Fluss und Minimaler Schnitt

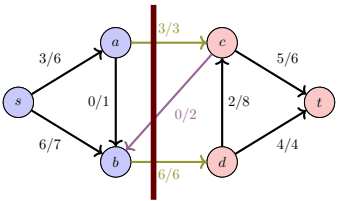


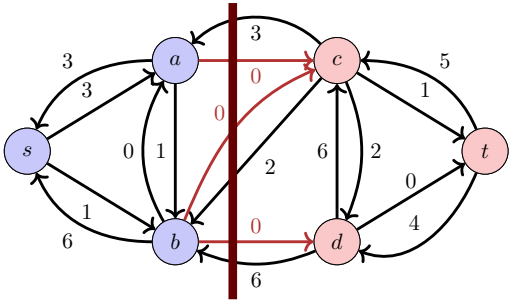


nach Terminierung von Ford-Fulkerson/Edmonds-Karp:

- $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s, $\mathbf{T} \subseteq \mathbf{V}$ nicht erreichbar von $s \Rightarrow$ **Schnitt** $(\mathbf{S}, \mathbf{T})$

Maximaler Fluss und Minimaler Schnitt

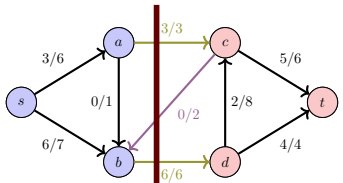


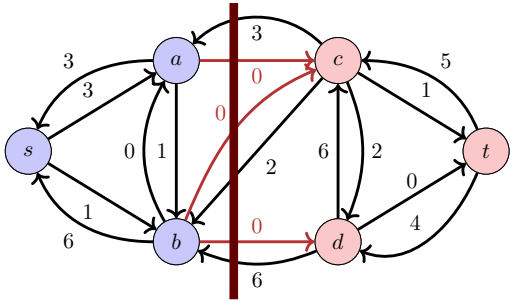


nach Terminierung von Ford-Fulkerson/Edmonds-Karp:

- $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s, $\mathbf{T} \subseteq \mathbf{V}$ nicht erreichbar von $s \Rightarrow$ **Schnitt** $(\mathbf{S}, \mathbf{T})$
- alle ausgehenden Kanten e haben Restkapazität 0 in G_f

Maximaler Fluss und Minimaler Schnitt

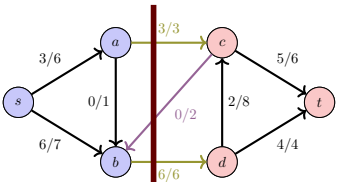


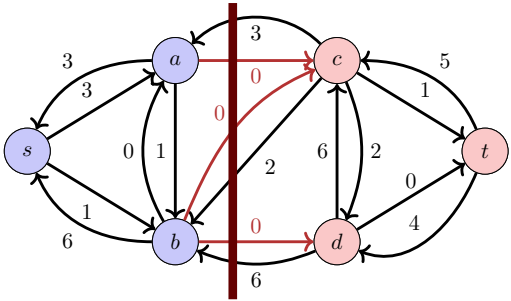


nach Terminierung von Ford-Fulkerson/Edmonds-Karp:

- $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s, $\mathbf{T} \subseteq \mathbf{V}$ nicht erreichbar von $s \Rightarrow$ **Schnitt** $(\mathbf{S}, \mathbf{T})$
- alle ausgehenden Kanten e haben Restkapazität 0 in G_f
- $f(S,T) = \sum_{\mathbf{e}:\, \mathbf{S}\to\mathbf{T}} \mathbf{f(e)} - \sum_{\mathbf{e}:\, \mathbf{T}\to\mathbf{S}} \mathbf{f(e)}$

Maximaler Fluss und Minimaler Schnitt

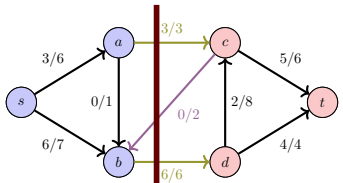


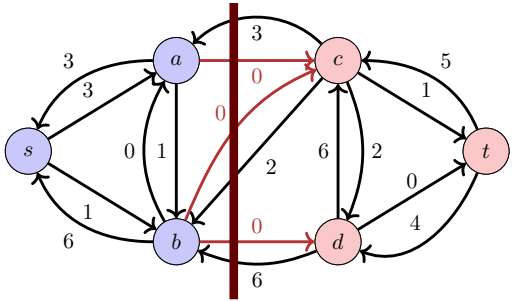


nach Terminierung von Ford-Fulkerson/Edmonds-Karp:

- $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s, $\mathbf{T} \subseteq \mathbf{V}$ nicht erreichbar von $s \Rightarrow$ **Schnitt** $(\mathbf{S}, \mathbf{T})$
- alle ausgehenden Kanten e haben Restkapazität 0 in G_f
- $f(S,T) = \sum_{\mathbf{e}:\, \mathbf{S}\to\mathbf{T}} \mathbf{f(e)} - \sum_{\mathbf{e}:\, \mathbf{T}\to\mathbf{S}} \mathbf{f(e)} = \sum_{\mathbf{e}:\, \mathbf{S}\to\mathbf{T}} \mathbf{c(e)}$

Maximaler Fluss und Minimaler Schnitt

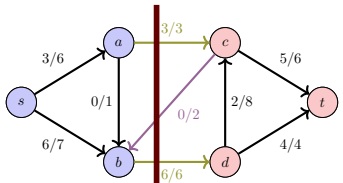


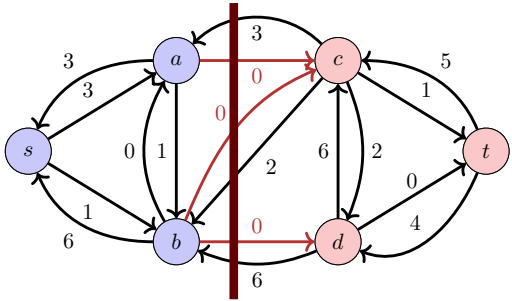


nach Terminierung von Ford-Fulkerson/Edmonds-Karp:

- $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s, $\mathbf{T} \subseteq \mathbf{V}$ nicht erreichbar von $s \Rightarrow$ **Schnitt** $(\mathbf{S}, \mathbf{T})$
- alle ausgehenden Kanten e haben Restkapazität 0 in G_f
- $f(S,T) = \sum_{\mathbf{e}:\, \mathbf{S}\to\mathbf{T}} \mathbf{f(e)} - \sum_{\mathbf{e}:\, \mathbf{T}\to\mathbf{S}} \mathbf{f(e)} = \sum_{\mathbf{e}:\, \mathbf{S}\to\mathbf{T}} \mathbf{c(e)} = \mathbf{c(S,T)}$

Maximaler Fluss und Minimaler Schnitt





nach Terminierung von Ford-Fulkerson/Edmonds-Karp:

- $\mathbf{S} \subseteq \mathbf{V}$ erreichbar von s, $\mathbf{T} \subseteq \mathbf{V}$ nicht erreichbar von $s \Rightarrow$ **Schnitt** $(\mathbf{S}, \mathbf{T})$
- alle ausgehenden Kanten e haben Restkapazität 0 in G_f
- $f(S,T) = \sum_{\mathbf{e}:\, \mathbf{S}\to\mathbf{T}} \mathbf{f(e)} - \sum_{\mathbf{e}:\, \mathbf{T}\to\mathbf{S}} \mathbf{f(e)} = \sum_{\mathbf{e}:\, \mathbf{S}\to\mathbf{T}} \mathbf{c(e)} = \mathbf{c(S,T)}$
  $\Rightarrow |f_{\max}| = c_{\min}$

Max-Flow Min-Cut Theorem

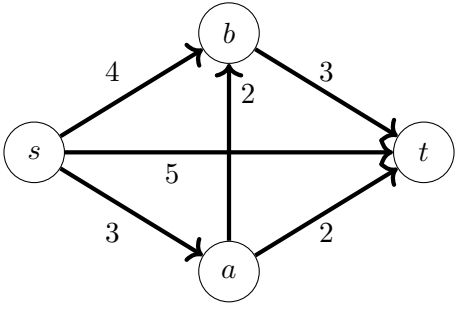
Max-Flow Min-Cut Theorem

Für einen Fluss f in einem Flussnetzwerk $G = (V, E, c)$ mit Quelle s und Senke t sind die folgende Aussagen äquivalent:

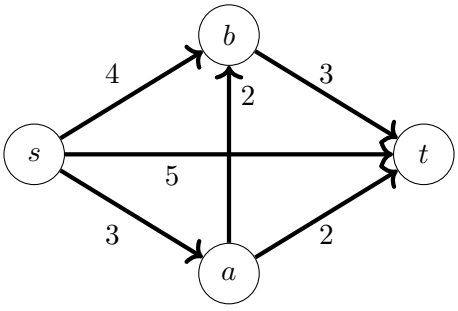
1. f ist ein maximaler Fluss in G
2. Das Restnetzwerk G_f enthält keine Erweiterungswege
3. $|f| = c(S, T)$ für einen Schnitt (S, T) von G.

Quiz

Quiz

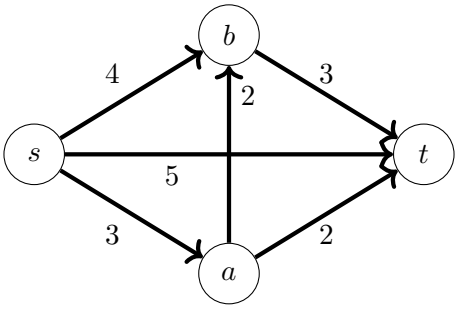


Quiz



Was ist der minimale Schnitt?

Quiz



Was ist der minimale Schnitt?
Was ist der maximale Fluss?

Anwendungsbeispiele

Anwendungsbeispiele

- Maximale Rate:
  - Wasser in Abwassersystem
  - Autos in Verkehr

Anwendungsbeispiele

- Maximale Rate:
  - Wasser in Abwassersystem
  - Autos in Verkehr
  - Strom in elektrischen Netwerken

Anwendungsbeispiele

- Maximale Rate:
  - Wasser in Abwassersystem
  - Autos in Verkehr
  - Strom in elektrischen Netwerken
  - Bauteile auf Fliessbändern

Anwendungsbeispiele

- Maximale Rate:
  - Wasser in Abwassersystem
  - Autos in Verkehr
  - Strom in elektrischen Netwerken
  - Bauteile auf Fliessbändern
  - Information in Kommunikationsnetzwerken

Anwendungsbeispiele

- Maximale Rate:
  - Wasser in Abwassersystem
  - Autos in Verkehr
  - Strom in elektrischen Netwerken
  - Bauteile auf Fliessbändern
  - Information in Kommunikationsnetzwerken
- Scheduling

Anwendungsbeispiele

- Maximale Rate:
  - Wasser in Abwassersystem
  - Autos in Verkehr
  - Strom in elektrischen Netwerken
  - Bauteile auf Fliessbändern
  - Information in Kommunikationsnetzwerken
- Scheduling
- Bipartites Matching

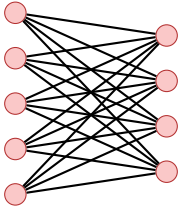
Anwendungsbeispiele

- Maximale Rate:
  - Wasser in Abwassersystem
  - Autos in Verkehr
  - Strom in elektrischen Netwerken
  - Bauteile auf Fliessbändern
  - Information in Kommunikationsnetzwerken
- Scheduling
- Bipartites Matching
- Segmentierung von Bildern

29.4 Maximales Bipartites Matching

Notation

Ein Graph, bei dem V so in disjunkte U und W aufgeteilt werden kann, dass alle $e \in E$ einen Knoten in U und einen in W haben heisst **bipartit**.

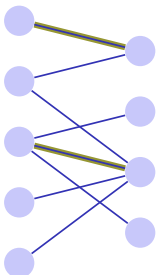


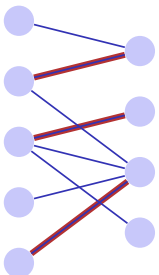
Anwendung: Maximales bipartites Matching

Gegeben: bipartiter ungerichteter Graph $G = (V, E)$.

Matching M: $M \subseteq E$ so dass $|\{m \in M : v \in m\}| \leq 1$ für alle $v \in V$.

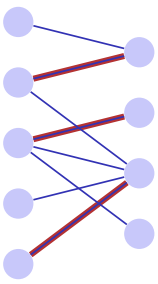
Maximales Matching M: Matching M, so dass $|M| \geq |M'|$ für jedes Matching M'.

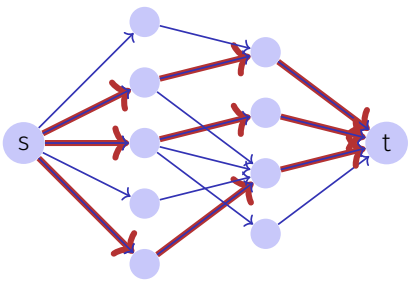




Korrespondierendes Flussnetzwerk

Konstruiere zur einer Partition L, R eines bipartiten Graphen ein korrespondierendes Flussnetzwerk mit Quelle s und Senke t, mit gerichteten Kanten von s nach L, von L nach R und von R nach t. Jede Kante bekommt Kapazität 1.





Zusammenfassung

Zusammenfassung

- Definitionen: Flussnetzwerk, Fluss, Schnitt

Zusammenfassung

- Definitionen: Flussnetzwerk, Fluss, Schnitt
- Konzepte: Umleitung, Restnetzwerk, Erweiterungsweg

Zusammenfassung

- Definitionen: Flussnetzwerk, Fluss, Schnitt
- Konzepte: Umleitung, Restnetzwerk, Erweiterungsweg
- Algorithmen

Zusammenfassung

- Definitionen: Flussnetzwerk, Fluss, Schnitt
- Konzepte: Umleitung, Restnetzwerk, Erweiterungsweg
- Algorithmen
  - Greedy: inkorrekt!

Zusammenfassung

- Definitionen: Flussnetzwerk, Fluss, Schnitt
- Konzepte: Umleitung, Restnetzwerk, Erweiterungsweg
- Algorithmen
  - Greedy: inkorrekt!
  - Ford-Fulkerson: $\mathcal{O}(|f_{\max}| \cdot |E|)$
    Greedy Erweiterungswege im Restnetzwerk

Zusammenfassung

- Definitionen: Flussnetzwerk, Fluss, Schnitt
- Konzepte: Umleitung, Restnetzwerk, Erweiterungsweg
- Algorithmen
  - Greedy: inkorrekt!
  - Ford-Fulkerson: $\mathcal{O}(|f_{\max}| \cdot |E|)$
    Greedy Erweiterungswege im Restnetzwerk
  - Edmonds-Karp: $\mathcal{O}(|V| \cdot |E|^2)$
    Ford-Fulkerson mit kürzesten Erweiterungswegen (Anzahl Kanten)

Zusammenfassung

- Definitionen: Flussnetzwerk, Fluss, Schnitt
- Konzepte: Umleitung, Restnetzwerk, Erweiterungsweg
- Algorithmen
  - Greedy: inkorrekt!
  - Ford-Fulkerson: $\mathcal{O}(|f_{\max}| \cdot |E|)$
    Greedy Erweiterungswege im Restnetzwerk
  - Edmonds-Karp: $\mathcal{O}(|V| \cdot |E|^2)$
    Ford-Fulkerson mit kürzesten Erweiterungswegen (Anzahl Kanten)
- Max Flow = Min Cut

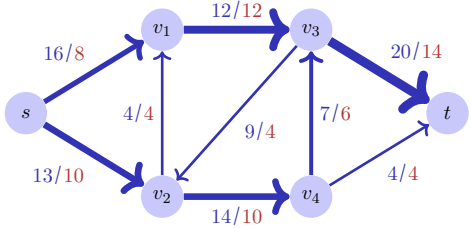
29.5 Anhang: Formales

Fluss: Formulierung mit Schiefsymmetrie

Ein **Fluss** $f : V \times V \to \mathbb{R}$ erfüllt folgende Bedingungen:

- **Kapazitätsbeschränkung**:
  Für alle $u, v \in V$: $f(u, v) \leq c(u, v)$.
- **Schiefsymmetrie**:
  Für alle $u, v \in V$: $f(u, v) = -f(v, u)$.
- **Flusserhaltung**:
  Für alle $u \in V \setminus \{s, t\}$:

$$\sum_{v \in V} f(u, v) = 0.$$



Wert w des Flusses:
$|f| = \sum_{v \in V} f(s, v)$.
Hier $|f| = 18$.

Schnitte

- **Kapazität** eines (s,t)-Schnittes: $c(S,T) = \sum_{v \in S, v' \in T} c(v,v')$
- **Minimaler Schnitt**: Schnitt mit minimaler Kapazität.
- **Fluss über Schnitt**: $f(S,T) = \sum_{v \in S, v' \in T} f(v,v')$

Allgemein Seien $U, U' \subseteq V$

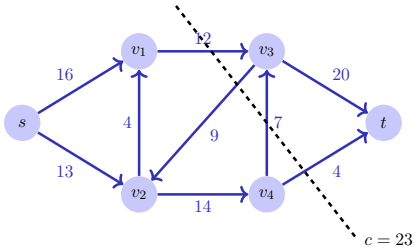
$$f(U,U') := \sum_{\substack{u \in U \\ u' \in U'}} f(u,u'), \qquad f(u,U') := f(\{u\},U')$$

Dann

- $|f| = f(s,V)$
- $f(U,U) = 0$
- $f(U,U') = -f(U',U)$
- $f(X \cup Y, Z) = f(X,Z) + f(Y,Z)$, wenn $X \cap Y = \emptyset$.
- $f(R,V) = 0$ wenn $R \cap \{s,t\} = \emptyset$. [Flusserhaltung!]

Wie gross kann ein Fluss sein?

$$f(S,T) = f(S,V) - \underbrace{f(S,S)}_{0} = f(S,V)$$
$$= f(s,V) + f(\underbrace{S - \{s\}}_{\not\ni t, \not\ni s}, V) = |f|.$$
$$\Rightarrow |f| \leq \sum_{v \in S, v' \in T} c(v,v') = c(S,T)$$



Restnetzwerk

Restnetzwerk G_f gegeben durch alle Kanten mit Restkapazität:

$$
\begin{aligned}
G_f &:= (V, E_f, c_f) \\
c_f(u,v) &:= c(u,v) - f(u,v) \quad \forall u, v \in V \\
E_f &:= \{(u,v) \in V \times V | c_f(u,v) > 0\}
\end{aligned}
$$

- Flusserhöhung in Richtung einer Kante möglich, wenn Fluss entlang der Kante erhöht werden kann, also wenn $f(u,v) < c(u,v)$.
  Restkapazität $c_f(u,v) = c(u,v) - f(u,v) > 0$.
- Flusserhöhung **entgegen** der Kantenrichtung möglich, wenn Fluss entlang der Kante verringert werden kann, also wenn $f(u,v) > 0$.
  Restkapazität $c_f(v,u) = f(u,v) > 0$.

Flusserhöhungen liefern Flüsse

Theorem 32

Sei $G = (V, E, c)$ ein Flussnetzwerk mit Quelle s und Senke t und f ein Fluss in G. Sei G_f das dazugehörige Restnetzwerk und sei f' ein Fluss in G_f. Dann definiert $f \oplus f'$ mit

$$(f \oplus f')(u, v) = f(u, v) + f'(u, v)$$

einen Fluss in G mit Wert $|f| + |f'|$.

Beweis

$f \oplus f'$ ist ein Fluss in G:

- Kapazitätsbeschränkung

$$(f \oplus f')(u,v) = f(u,v) + \underbrace{f'(u,v)}_{\leq c(u,v) - f(u,v)} \leq c(u,v)$$

- Schiefsymmetrie

$$(f \oplus f')(u,v) = -f(v,u) + -f'(v,u) = -(f \oplus f')(v,u)$$

- Flusserhaltung $u \in V - \{s,t\}$:

$$\sum_{v \in V} (f \oplus f')(u,v) = \sum_{v \in V} f(u,v) + \sum_{v \in V} f'(u,v) = 0$$

Beweis

Wert von $f \oplus f'$

$$
\begin{aligned}
|f \oplus f'| &= (f \oplus f')(s, V) \\
&= \sum_{u \in V} f(s,u) + f'(s,u) \\
&= f(s,V) + f'(s,V) \\
&= |f| + |f'|
\end{aligned}
$$

■

Erweiterungspfade

Erweiterungspfad p: einfacher Pfad von s nach t im Restnetzwerk G_f.
Restkapazität $c_f(p) = \min\{c_f(u,v) : (u,v) \text{ Kante in } p\}$

Theorem 33

Die Funktion $f_p : V \times V \to \mathbb{R}$,

$$f_p(u,v) = \begin{cases} c_f(p) & \text{wenn } (u,v) \text{ Kante in } p \\ -c_f(p) & \text{wenn } (v,u) \text{ Kante in } p \\ 0 & \text{sonst} \end{cases}$$

ist ein Fluss in G_f *mit dem Wert* $|f_p| = c_f(p) > 0$.

f_p ist ein Fluss (leicht nachprüfbar). Es gibt genau einen Knoten $u \in V$ mit $(s,u) \in p$. Somit $|f_p| = \sum_{v \in V} f_p(s,v) = f_p(s,u) = c_f(p)$.

Max-Flow Min-Cut Theorem

Theorem 34

Wenn f ein Fluss in einem Flussnetzwerk $G = (V, E, c)$ mit Quelle s und Senke t is, dann sind folgende Aussagen äquivalent:

1. *f ist ein maximaler Fluss in G*
2. *Das Restnetzwerk G_f enthält keine Erweiterungspfade*
3. *Es gilt $|f| = c(S, T)$ für einen Schnitt (S, T) von G.*

Beweis

- $(3) \Rightarrow (1)$:
  Es gilt $|f| \leq c(S,T)$ für alle Schnitte S, T. Aus $|f| = c(S,T)$ folgt also $|f|$ maximal.
- $(1) \Rightarrow (2)$:
  f maximaler Fluss in G. Annahme: G_f habe einen Erweiterungsfad.
  Dann gilt $|f \oplus f_p| = |f| + |f_p| > |f|$. Widerspruch.

Beweis $(2) \Rightarrow (3)$

Annahme: G_f habe keinen Erweiterungsfad
Definiere $S = \{v \in V :$ es existiert Pfad $s \rightsquigarrow v$ in $G_f\}$.
$(S, T) := (S, V \setminus S)$ ist ein Schnitt: $s \in S, t \in T$.
Sei $u \in S$ und $v \in T$. Dann $c_f(u, v) = 0$, also $c_f(u, v) = c(u, v) - f(u, v) = 0$.
Somit $f(u, v) = c(u, v)$.
Somit

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

■

Edmonds-Karp Algorithmus

Theorem 35

Wenn der Edmonds-Karp Algorithmus auf ein ganzzahliges Flussnetzwerk $G = (V, E)$ mit Quelle s und Senke t angewendet wird, dann ist die Gesamtanzahl der durch den Algorithmus angewendete Flusserhöhungen in $\mathcal{O}(|V| \cdot |E|)$.

$\Rightarrow$ *Gesamte asymptotische Laufzeit: $\mathcal{O}(|V| \cdot |E|^2)$*

[Ohne Beweis]

Edmonds-Karp Algorithmus

Theorem 36

Wenn der Edmonds-Karp Algorithmus auf Flussnetzwerk $G = (V, E)$ mit Quelle s und Senke t angewendet wird, dann wächst für jeden Knoten $v \in V \setminus \{s, t\}$ die Distanz $\delta_f(s, v)$ des kürzesten Pfades von s nach v im Restnetzwerk G_f monoton mit jeder Flusserhöhung.

Beweis

Annahme: Distanz $\delta_f(s, v)$ wird bei Flusserhöhung $f \to f'$ kleiner für ein v: $\delta_f(s, v) < \delta_{f'}(s, v)$
Sei $p = s \rightsquigarrow u \to v$ kürzester Pfad von s nach v in $G_{f'}$, so dass $(u, v) \in E_{f'}$ und $\delta_{f'}(s, u) = \delta_{f'}(s, v) - 1$. Es gilt $\delta_{f'}(s, u) \geq \delta_f(s, u)$.
Wenn $(u, v) \in E_f$: $\delta_f s, v \leq \delta_f(s, u) + 1 \leq \delta_{f'}(s, u) + 1 = \delta_{f'}(s, v)$ Widerspruch.
Also $(u, v) \notin E_f$.

Ganzzahligkeitstheorem

Theorem 37

Wenn die Kapazitäten eines Flussnetzwerks nur ganzzahlige Werte anehmen, dann hat der durch Ford-Fulkerson erzeugte maximale Fluss die Eigenschaft, dass der Wert von $f(u, v)$ für alle $u, v \in V$ eine ganze Zahl ist.

[ohne Beweis]

Folgerung: Ford Fulkerson erzeugt beim zum bipartiten Graph gehörenden Flussnetzwerk ein maximales Matching $M = \{(u, v) : f(u, v) = 1\}$.