

Algorithmische Graphentheorie

Sommersemester 2022

4. Vorlesung

Flussalgorithmen

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert.

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

- den Fluss erhält

- zulässig ist

- maximal ist

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

– den Fluss erhält, d.h. für jeden Knoten $v \notin \{s, t\}$ sicherstellt:

$$\text{Nettozufluss}_f(v) = \sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) = 0,$$

– zulässig ist

– maximal ist

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

– den Fluss erhält, d.h. für jeden Knoten $v \notin \{s, t\}$ sicherstellt:

$$\text{Nettozufluss}_f(v) = \sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) = 0,$$

– zulässig ist, d.h. für jede Kante e garantiert:

$$0 \leq f(e) \leq c(e),$$

– maximal ist

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

– den Fluss erhält, d.h. für jeden Knoten $v \notin \{s, t\}$ sicherstellt:

$$\text{Nettozufluss}_f(v) = \sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) = 0,$$

– zulässig ist, d.h. für jede Kante e garantiert:

$$0 \leq f(e) \leq c(e),$$

– maximal ist, d.h. unter allen zulässigen s - t -Flüssen

$$|f| = \text{Nettozufluss}_f(t) \text{ maximiert.}$$

Aufgabe


Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

– den Fluss erhält, d.h. für jeden Knoten $v \notin \{s, t\}$ sicherstellt:

$$\text{Nettozufluss}_f(v) = \sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) = 0,$$

Variable



– zulässig ist, d.h. für jede Kante e garantiert:

$$0 \leq f(e) \leq c(e),$$

– maximal ist, d.h. unter allen zulässigen s - t -Flüssen

$$|f| = \text{Nettozufluss}_f(t) \text{ maximiert.}$$

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

– den Fluss erhält, d.h. für jeden Knoten $v \notin \{s, t\}$ sicherstellt:

$$\text{Nettozufluss}_f(v) = \sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) = 0,$$

Variable →

– zulässig ist, d.h. für jede Kante e garantiert:

$$\text{Konstante} \rightarrow 0 \leq f(e) \leq c(e),$$

– maximal ist, d.h. unter allen zulässigen s - t -Flüssen

$$|f| = \text{Nettozufluss}_f(t) \text{ maximiert.}$$

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

– den Fluss erhält, d.h. für jeden Knoten $v \notin \{s, t\}$ sicherstellt:

$$\text{Nettozufluss}_f(v) = \sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) = 0,$$

Variable →

– zulässig ist, d.h. für jede Kante e garantiert:

$$0 \leq f(e) \leq c(e),$$

Konstante →

– maximal ist, d.h. unter allen zulässigen s - t -Flüssen

$$|f| = \text{Nettozufluss}_f(t) \text{ maximiert.}$$

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

- den Fluss erhält, d.h. für jeden Knoten $v \notin \{s, t\}$ sicherstellt:

$$\text{Nettozufluss}_f(v) = \sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) = 0,$$

- zulässig ist, d.h. für jede Kante e garantiert:

$$0 \leq f(e) \leq c(e),$$

$|V| - 2 + |E|$ lineare Beschränkungen!

- maximal ist, d.h. unter allen zulässigen s - t -Flüssen

$$|f| = \text{Nettozufluss}_f(t) \text{ maximiert.}$$

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

– den Fluss erhält, d.h. für jeden Knoten $v \notin \{s, t\}$ sicherstellt:

$$\text{Nettozufluss}_f(v) = \sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) = 0,$$

– zulässig ist, d.h. für jede Kante e garantiert:

$$0 \leq f(e) \leq c(e),$$

$|V| - 2 + |E|$ lineare Beschränkungen!

– maximal ist, d.h. unter allen zulässigen s - t -Flüssen

$$|f| = \text{Nettozufluss}_f(t) \text{ maximiert.}$$

Aufgabe

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

– den Fluss erhält, d.h. für jeden Knoten $v \notin \{s, t\}$ sicherstellt:

$$\text{Nettozufluss}_f(v) = \sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) = 0,$$

– zulässig ist, d.h. für jede Kante e garantiert:

Konstante

$$0 \leq f(e) \leq c(e),$$

$|V| - 2 + |E|$ lineare Beschränkungen!

– maximal ist, d.h. unter allen zulässigen s - t -Flüssen

$$|f| = \text{Nettozufluss}_f(t) \text{ maximiert.}$$

lineare Zielfunktion!

Variable

Aufgabe

Indeed, it's an LP!

Gegeben ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$ und Kantenkapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Geben Sie eine Methode an, die einen **maximalen s - t -Fluss f** konstruiert, also eine Funktion $f: E \rightarrow \mathbb{R}_{\geq 0}$, die

– den Fluss erhält, d.h. für jeden Knoten $v \notin \{s, t\}$ sicherstellt:

$$\text{Nettozufluss}_f(v) = \sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) = 0,$$

– zulässig ist, d.h. für jede Kante e garantiert:

$$0 \leq f(e) \leq c(e),$$

$|V| - 2 + |E|$ lineare Beschränkungen!

– maximal ist, d.h. unter allen zulässigen s - t -Flüssen

$$|f| = \text{Nettozufluss}_f(t) \text{ maximiert.}$$

lineare Zielfunktion!

Variable

Konstante

Flussalgorithmen

Kann man maximale Flüsse (= Spezialfall eines LPs) auch mit maßgeschneiderten kombinatorischen Algorithmen berechnen?

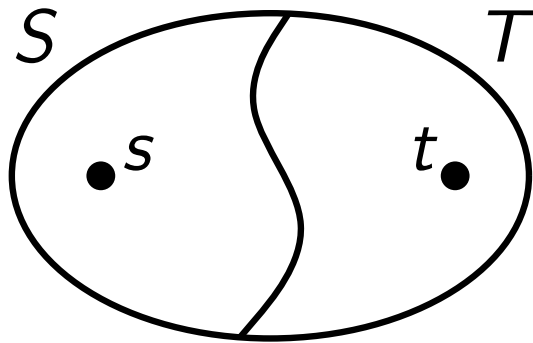
Flussalgorithmen

Kann man maximale Flüsse (= Spezialfall eines LPs) auch mit maßgeschneiderten kombinatorischen Algorithmen berechnen?

Hoffnung: Das könnte schneller gehen –
und strukturelle Einsichten liefern.

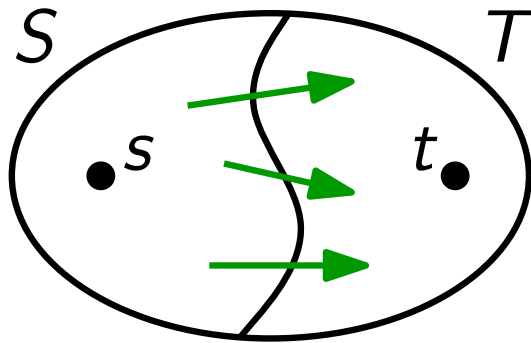
Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.
Eine Zerlegung (S, T) von V ist ein s - t -Schnitt,
falls $s \in S, t \in T$.



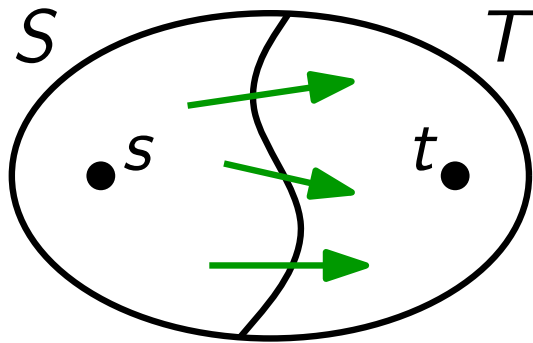
Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.
Eine Zerlegung (S, T) von V ist ein s - t -Schnitt,
falls $s \in S, t \in T$.



Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.
Eine Zerlegung (S, T) von V ist ein s - t -Schnitt,
falls $s \in S, t \in T$.

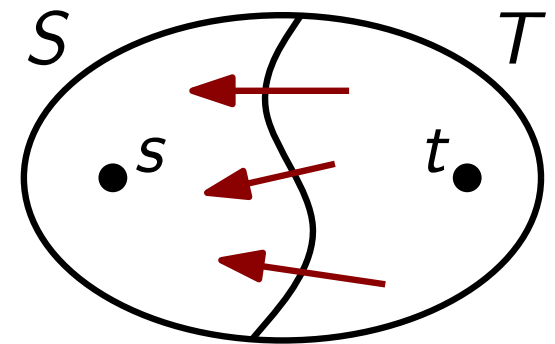
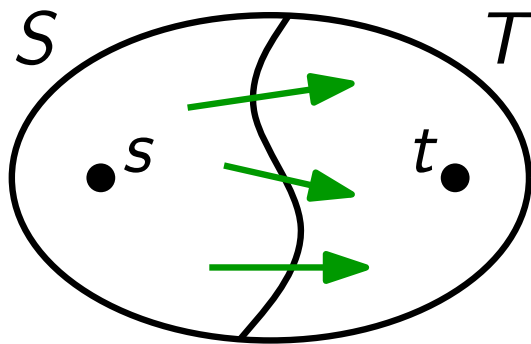


$$\text{Raus}(S) = \{uv \in E \mid u \in S, v \in T\}$$

Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.

Eine Zerlegung (S, T) von V ist ein s - t -Schnitt, falls $s \in S, t \in T$.

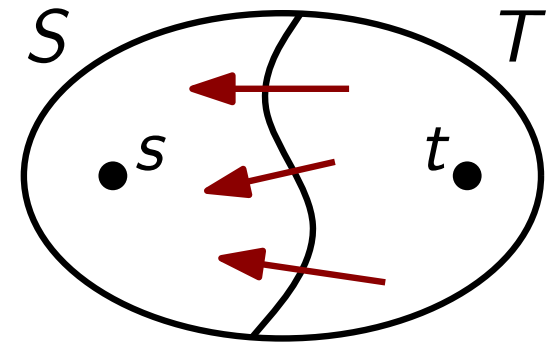
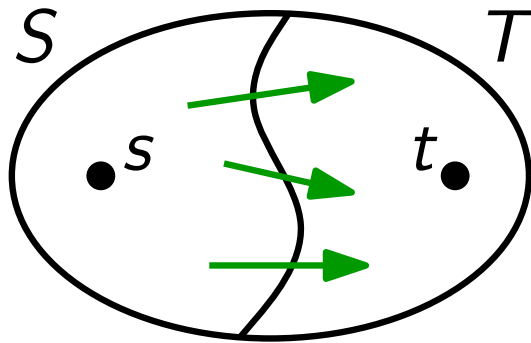


$$\text{Raus}(S) = \{uv \in E \mid u \in S, v \in T\}$$

Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.

Eine Zerlegung (S, T) von V ist ein s - t -Schnitt, falls $s \in S, t \in T$.

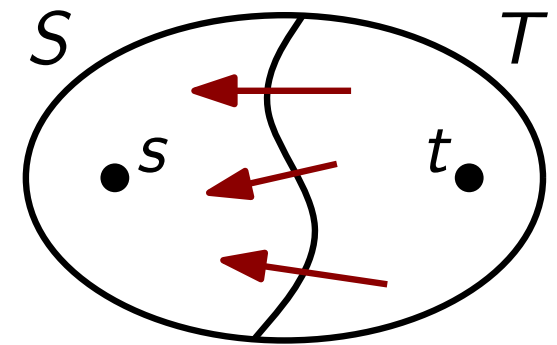
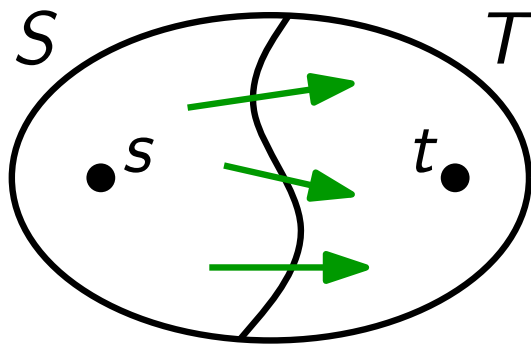


$$\text{Raus}(S) = \{uv \in E \mid u \in S, v \in T\}$$

$$\{uv \in E \mid u \in T, v \in S\} = \text{Rein}(S)$$

Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.
 Eine Zerlegung (S, T) von V ist ein s - t -Schnitt,
 falls $s \in S, t \in T$.



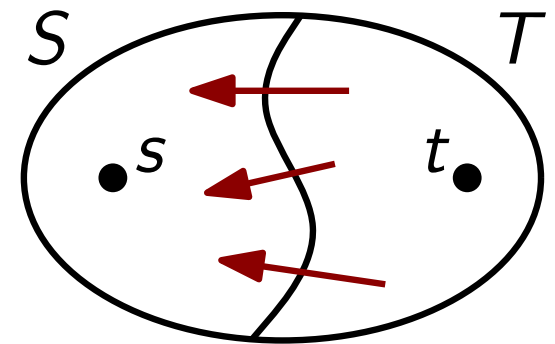
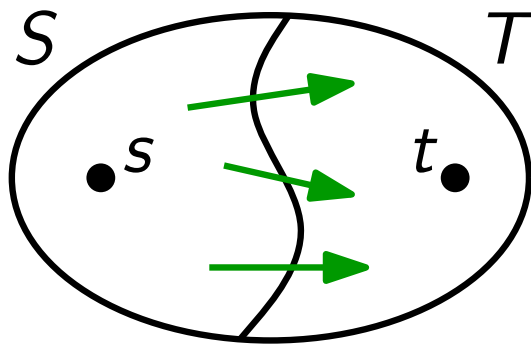
$$\begin{aligned} \text{Raus}(S) &= \{uv \in E \mid u \in S, v \in T\} \\ \{uv \in E \mid u \in T, v \in S\} &= \text{Rein}(S) \end{aligned}$$

$$\text{Zufluss}_f(S) = f(\text{Rein}(S))$$

Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.

Eine Zerlegung (S, T) von V ist ein s - t -Schnitt, falls $s \in S, t \in T$.



$$\text{Raus}(S) = \{uv \in E \mid u \in S, v \in T\}$$

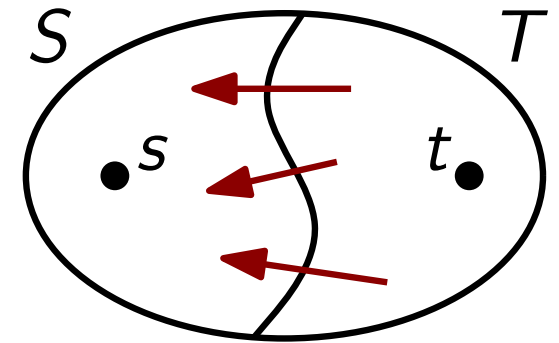
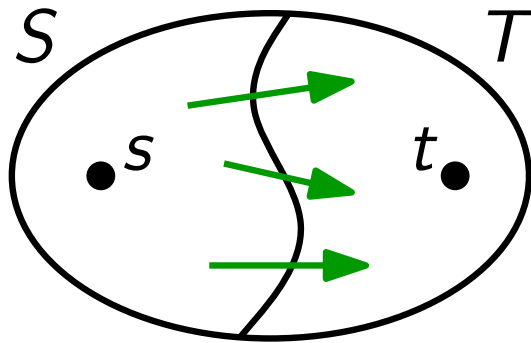
$$\{uv \in E \mid u \in T, v \in S\} = \text{Rein}(S)$$

$$\text{Zufluss}_f(S) = f(\text{Rein}(S)) = \sum_{e \in \text{Rein}(S)} f(e)$$

Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.

Eine Zerlegung (S, T) von V ist ein s - t -Schnitt, falls $s \in S, t \in T$.



$$\begin{aligned} \text{Raus}(S) &= \{uv \in E \mid u \in S, v \in T\} \\ \{uv \in E \mid u \in T, v \in S\} &= \text{Rein}(S) \end{aligned}$$

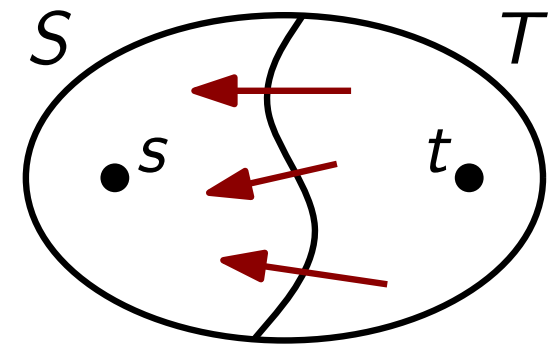
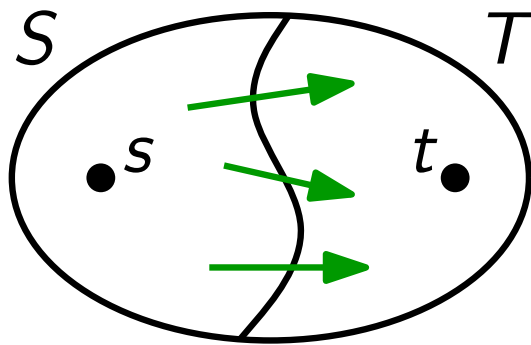
$$\text{Zufluss}_f(S) = f(\text{Rein}(S)) = \sum_{e \in \text{Rein}(S)} f(e)$$

$$\text{Abfluss}_f(S) = f(\text{Raus}(S))$$

Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.

Eine Zerlegung (S, T) von V ist ein s - t -Schnitt, falls $s \in S, t \in T$.



$$\begin{aligned} \text{Raus}(S) &= \{uv \in E \mid u \in S, v \in T\} \\ \{uv \in E \mid u \in T, v \in S\} &= \text{Rein}(S) \end{aligned}$$

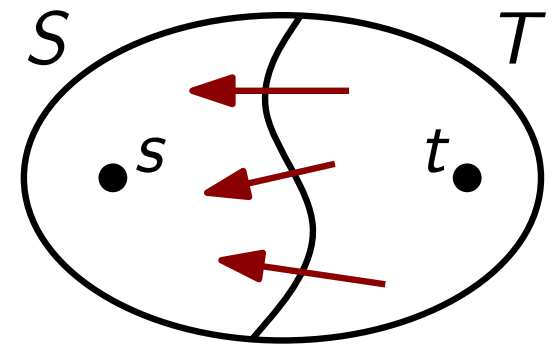
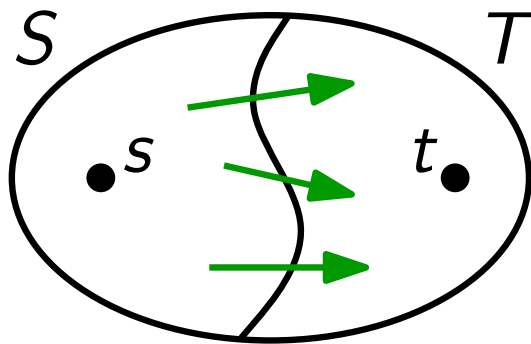
$$\text{Zufluss}_f(S) = f(\text{Rein}(S)) = \sum_{e \in \text{Rein}(S)} f(e)$$

$$\text{Abfluss}_f(S) = f(\text{Raus}(S)) = \sum_{e \in \text{Raus}(S)} f(e)$$

Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.

Eine Zerlegung (S, T) von V ist ein s - t -Schnitt, falls $s \in S, t \in T$.



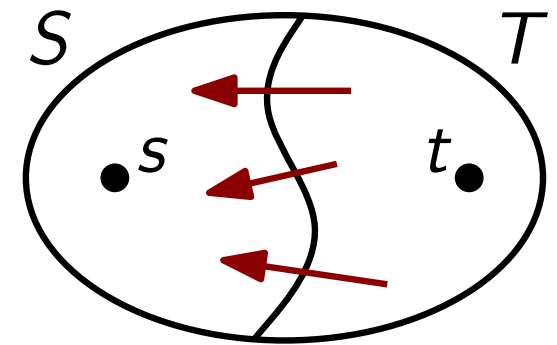
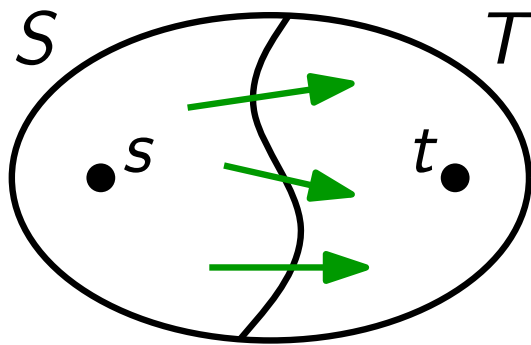
$$\begin{aligned} \text{Raus}(S) &= \{uv \in E \mid u \in S, v \in T\} \\ \{uv \in E \mid u \in T, v \in S\} &= \text{Rein}(S) \end{aligned}$$

$$\text{Zufluss}_f(S) = f(\text{Rein}(S))$$

$$\text{Abfluss}_f(S) = f(\text{Raus}(S))$$

Nichts ist praktischer als eine gute Theorie

Def. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$.
 Eine Zerlegung (S, T) von V ist ein s - t -Schnitt,
 falls $s \in S, t \in T$.

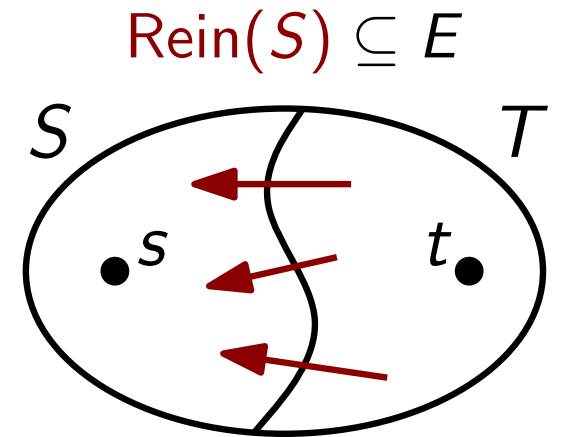
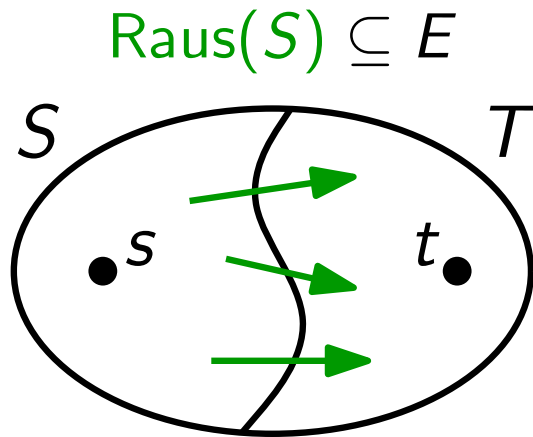


$$\begin{aligned} \text{Raus}(S) &= \{uv \in E \mid u \in S, v \in T\} \\ \{uv \in E \mid u \in T, v \in S\} &= \text{Rein}(S) \end{aligned}$$

$$\left. \begin{aligned} \text{Zufluss}_f(S) &= f(\text{Rein}(S)) \\ \text{Abfluss}_f(S) &= f(\text{Raus}(S)) \end{aligned} \right\} \text{minus} \quad =: \text{Nettozufluss}_f(S)$$

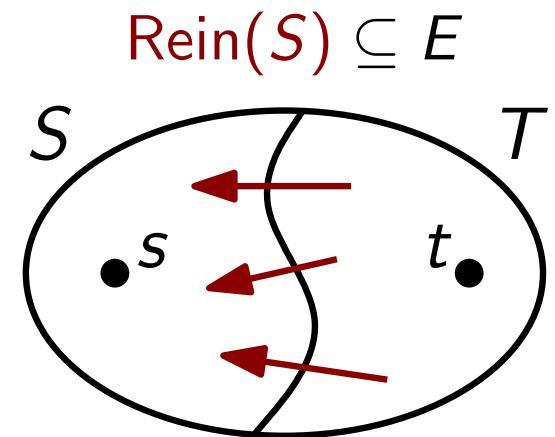
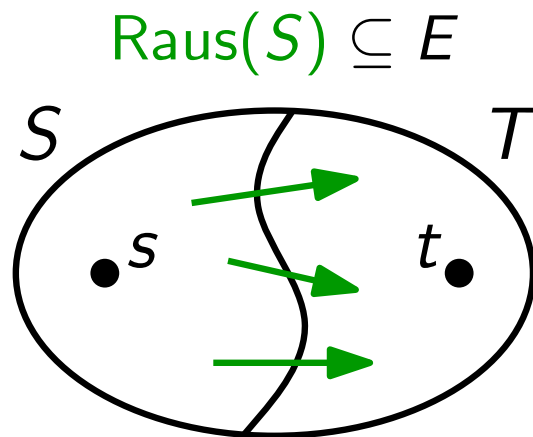
Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Nettozuflüsse von Schnitten und Knoten

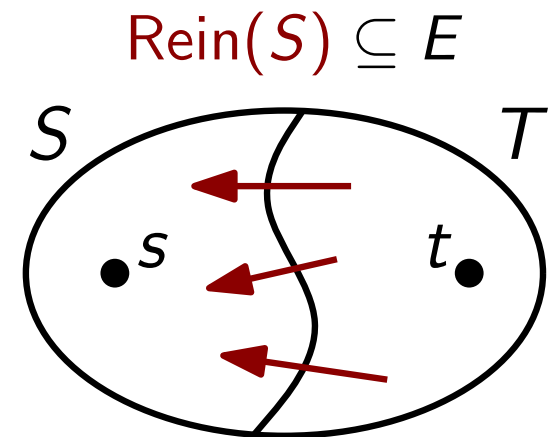
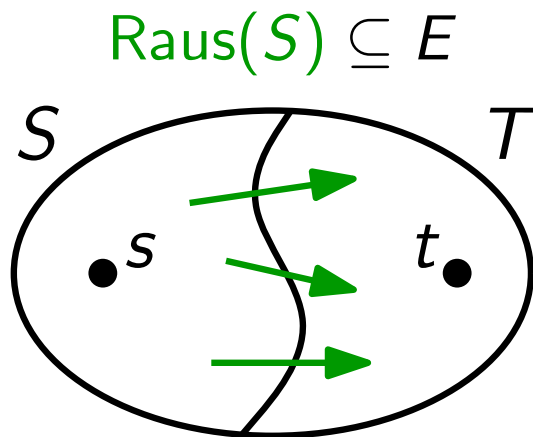
Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
 $\text{Nettozufluss}_f(S) =$

Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$

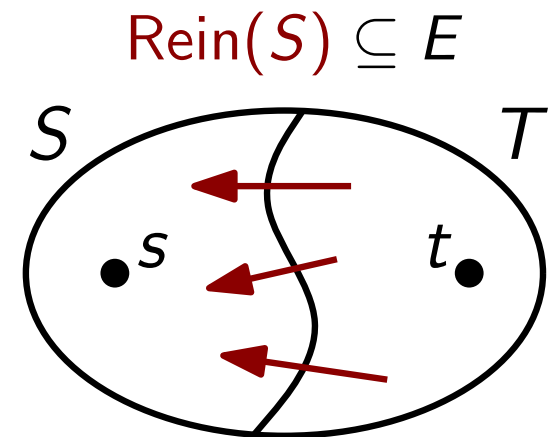
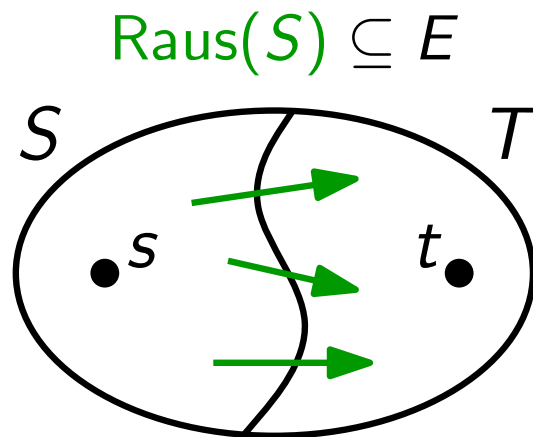


Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S}$$

Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$

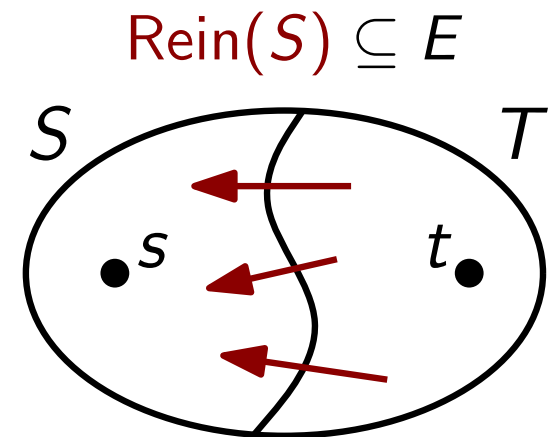
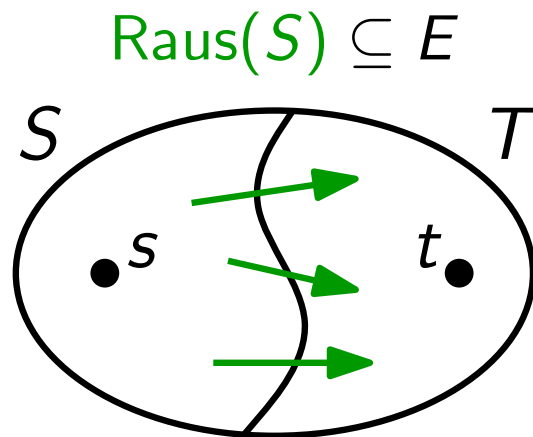


Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



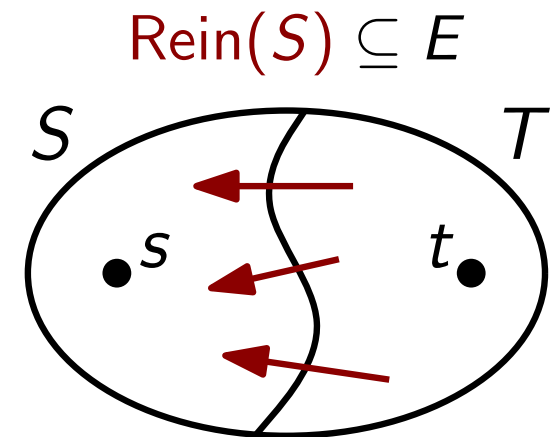
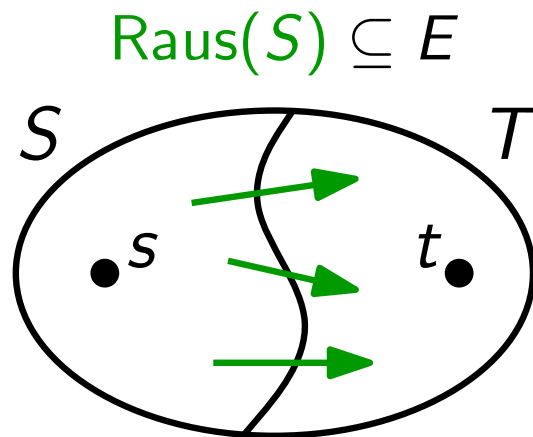
Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Beweis. $\sum_{v \in S} \text{Nettozufl}_f(v) =$

Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



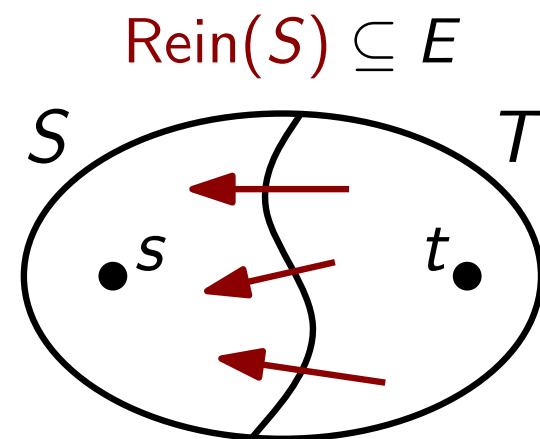
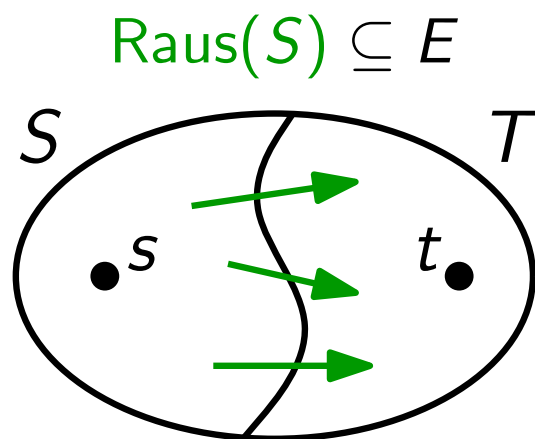
Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Beweis.
$$\sum_{v \in S} \text{Nettozufluss}_f(v) = \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v))$$

Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



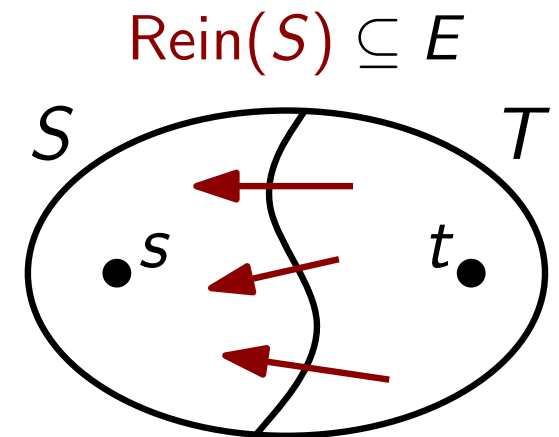
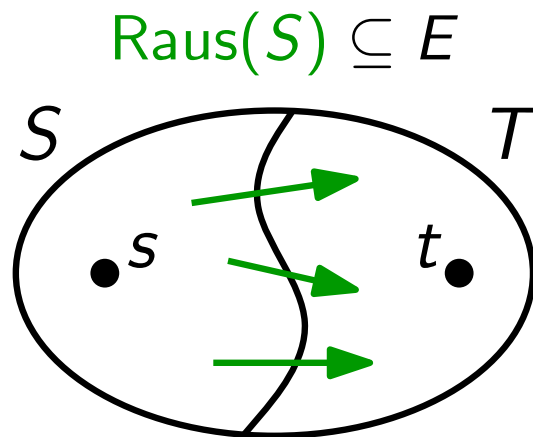
Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Beweis.
$$\begin{aligned} \sum_{v \in S} \text{Nettozufluss}_f(v) &= \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v)) \\ &= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right) \end{aligned}$$

Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



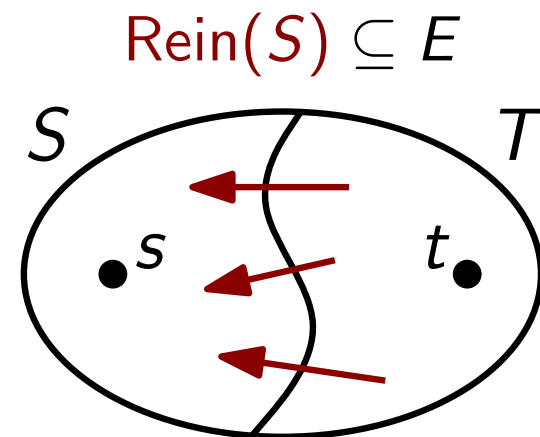
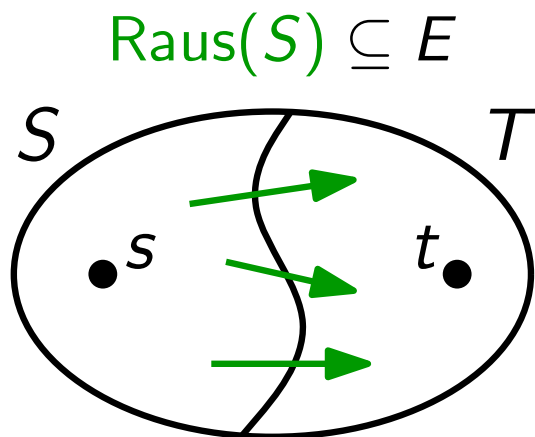
Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Beweis.
$$\begin{aligned} \sum_{v \in S} \text{Nettozufluss}_f(v) &= \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v)) \\ &= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right) \\ &= \end{aligned}$$

Nettozuflüsse von Schnitten und Knoten

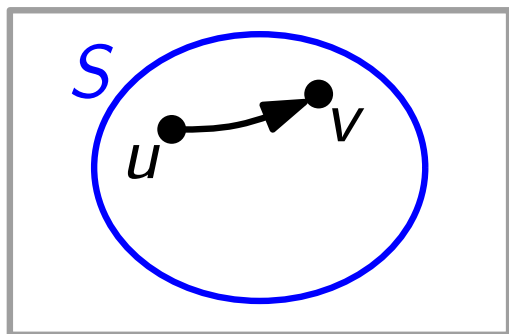
Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Beweis. $\sum_{v \in S} \text{Nettozufluss}_f(v) = \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v))$

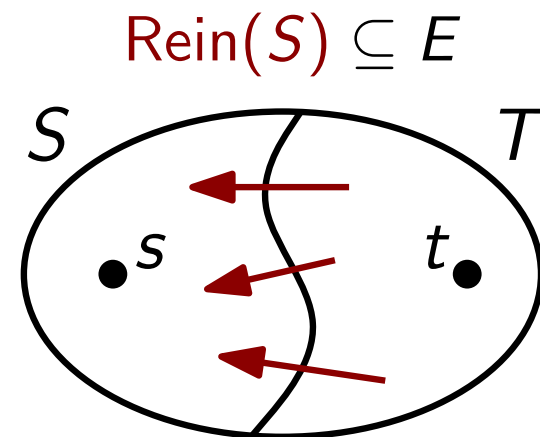
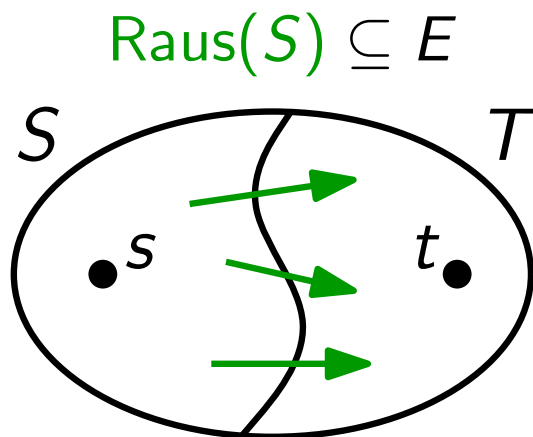


$$= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right)$$

=

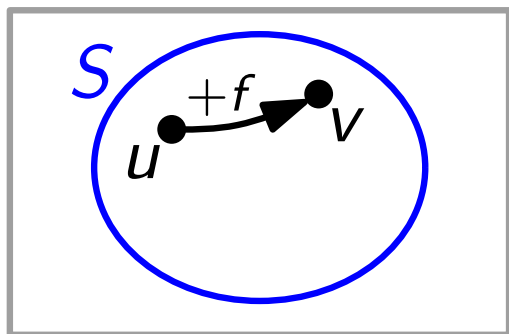
Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
 $\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v)$.

Beweis. $\sum_{v \in S} \text{Nettozufluss}_f(v) = \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v))$

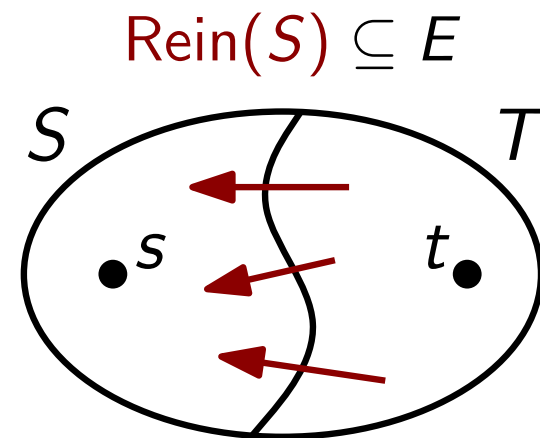
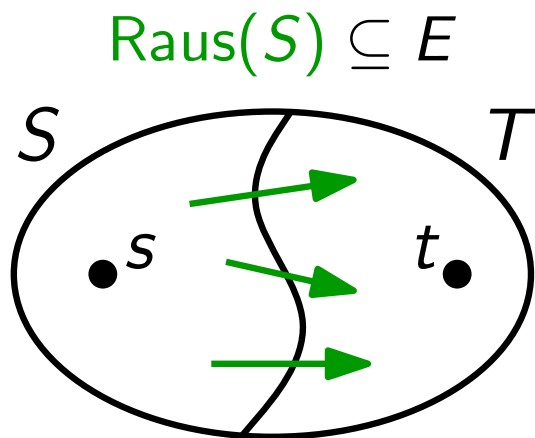


$$= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right)$$

=

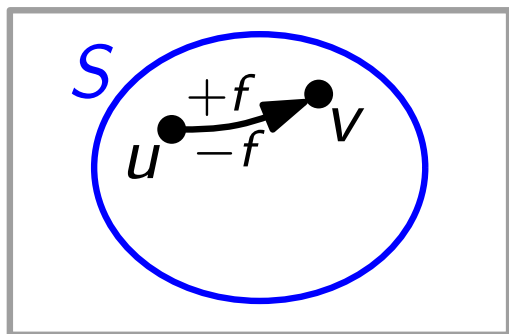
Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
 $\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v)$.

Beweis. $\sum_{v \in S} \text{Nettozufluss}_f(v) = \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v))$

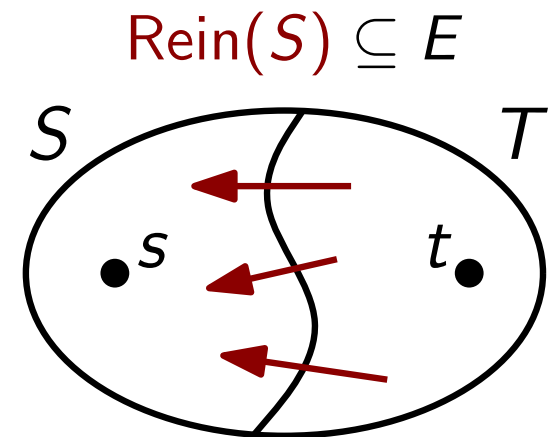
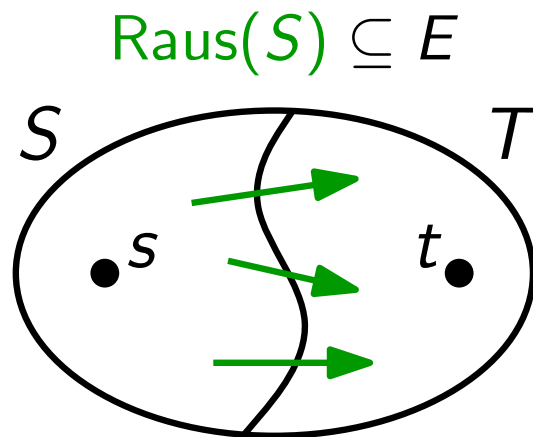


$$= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right)$$

=

Nettozuflüsse von Schnitten und Knoten

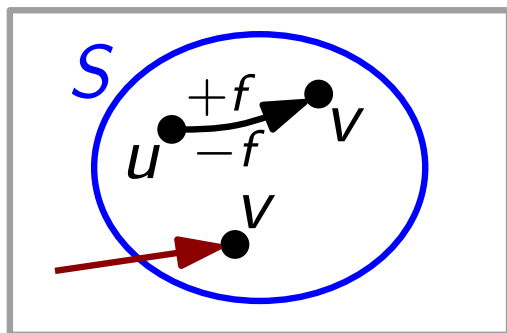
Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

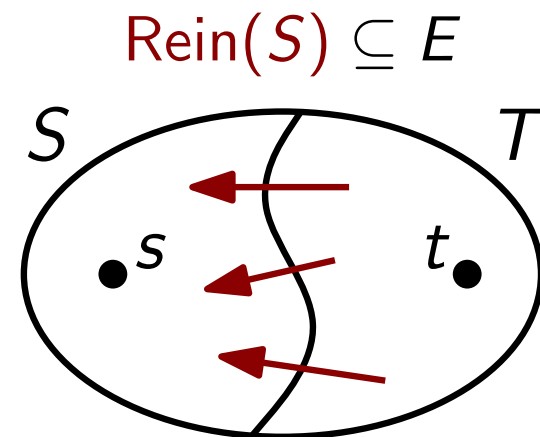
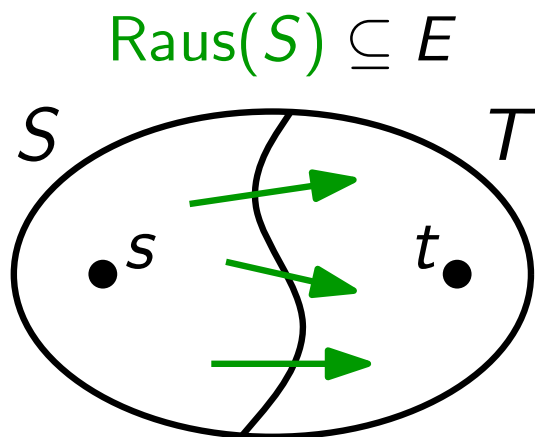
Beweis. $\sum_{v \in S} \text{Nettozufluss}_f(v) = \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v))$



$$= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right)$$

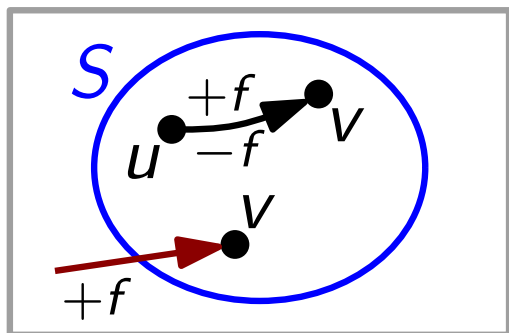
Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
 $\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v)$.

Beweis. $\sum_{v \in S} \text{Nettozufluss}_f(v) = \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v))$

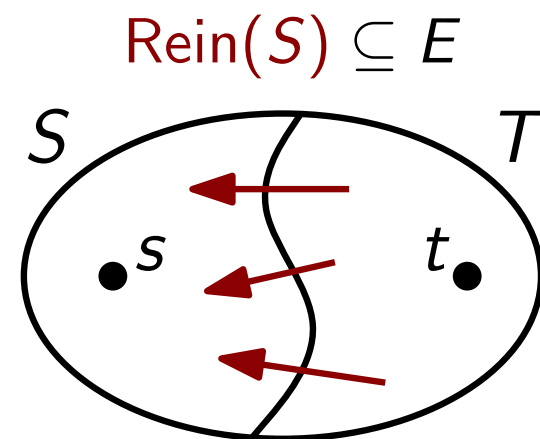
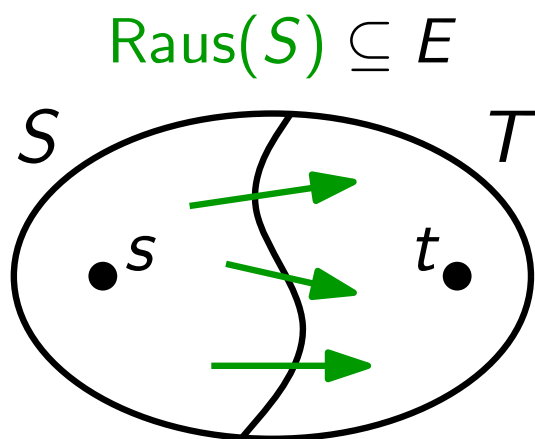


$$= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right)$$

=

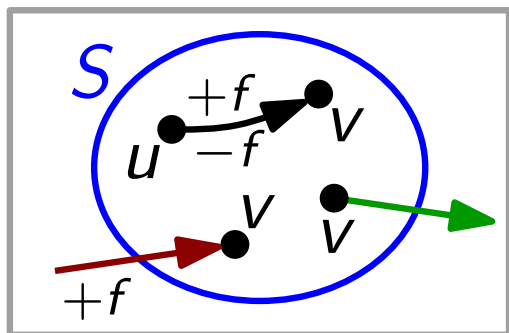
Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
 $\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v)$.

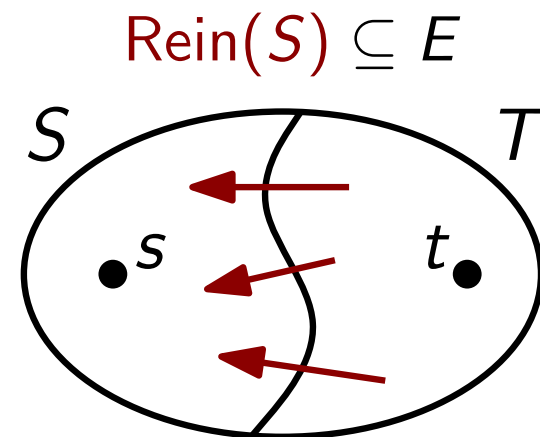
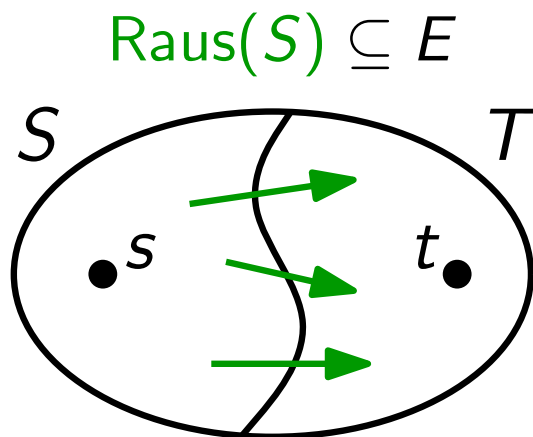
Beweis. $\sum_{v \in S} \text{Nettozufluss}_f(v) = \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v))$



$$= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right)$$

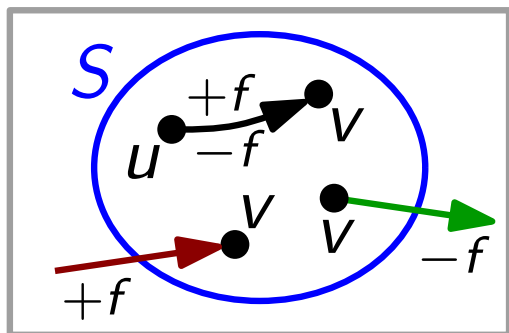
Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
 $\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v)$.

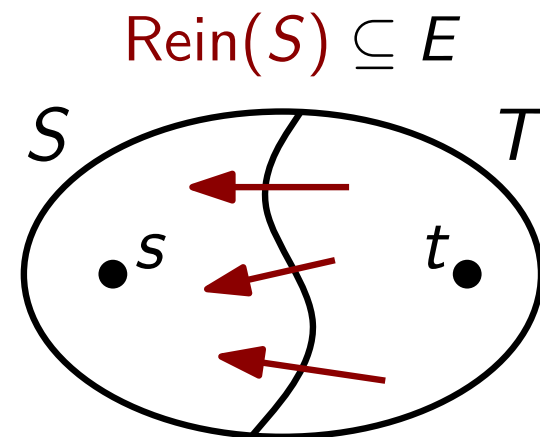
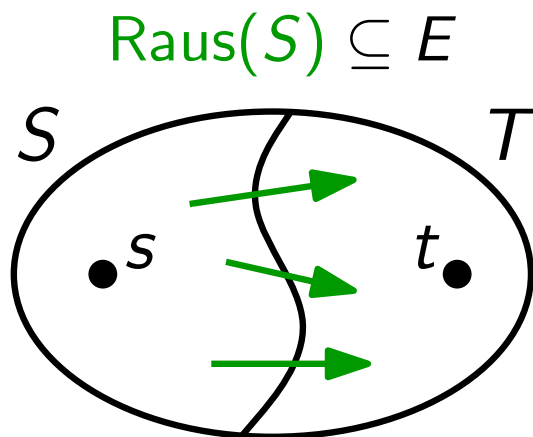
Beweis. $\sum_{v \in S} \text{Nettozufluss}_f(v) = \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v))$



$$= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right)$$

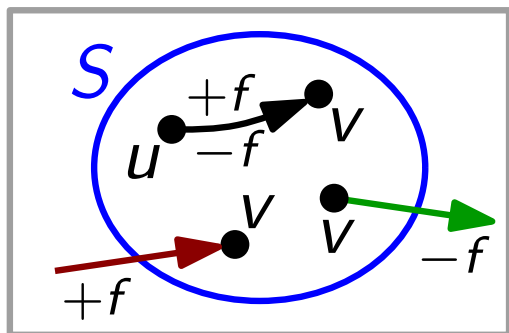
Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
 $\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v)$.

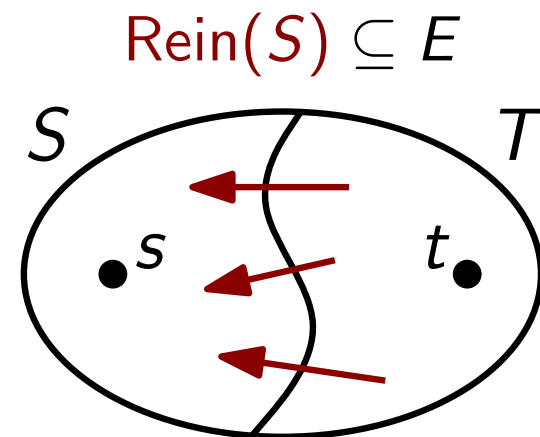
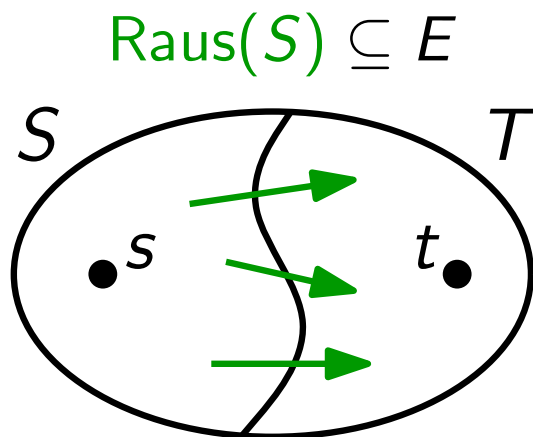
Beweis. $\sum_{v \in S} \text{Nettozufluss}_f(v) = \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v))$



$$\begin{aligned}
 &= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right) \\
 &= \sum_{e \in \text{Rein}(S)} f(e) - \sum_{e \in \text{Raus}(S)} f(e)
 \end{aligned}$$

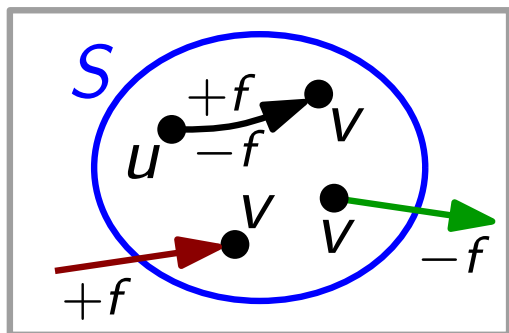
Nettozuflüsse von Schnitten und Knoten

Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



Lem.¹ Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
 $\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v)$.

Beweis. $\sum_{v \in S} \text{Nettozufluss}_f(v) = \sum_{v \in S} (\text{Zufluss}_f(v) - \text{Abfluss}_f(v))$



$$\begin{aligned}
 &= \sum_{v \in S} \left(\sum_{u: v \in \text{Adj}[u]} f(uv) - \sum_{w \in \text{Adj}[v]} f(vw) \right) \\
 &= \sum_{e \in \text{Rein}(S)} f(e) - \sum_{e \in \text{Raus}(S)} f(e) = \text{Nettozufluss}_f(S) \quad \square
 \end{aligned}$$

Noch mehr Schnitte

Lemma¹. Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Noch mehr Schnitte

Lemma¹. Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| =$

Noch mehr Schnitte

Lemma¹. Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T)$.

Noch mehr Schnitte

Lemma¹. Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T)$.

Beweis. $|f| \stackrel{\text{Def.}}{=} \sum_{v \in V} \text{Nettozufluss}_f(v)$

Noch mehr Schnitte

Lemma¹. Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
 Dann gilt $|f| = \text{Nettozufluss}_f(T)$.

Beweis. $|f| \stackrel{\text{Def.}}{=} \text{Nettozufluss}_f(t)$
 $=$

Noch mehr Schnitte

Lemma¹. Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T)$.

Beweis. $|f| \stackrel{\text{Def.}}{=} \text{Nettozufluss}_f(t)$
 $= \sum_{v \in T} \text{Nettozufluss}_f(v)$

Noch mehr Schnitte

Lemma¹. Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:
$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T)$.

Beweis. $|f| \stackrel{\text{Def.}}{=} \text{Nettozufluss}_f(t)$
 $= \sum_{v \in T} \text{Nettozufluss}_f(v)$
da $\text{Nettozufluss}_f(v) = 0$ für alle $v \neq s, t$

Noch mehr Schnitte

Lemma¹. Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
 Dann gilt $|f| = \text{Nettozufluss}_f(T)$.

Beweis. $|f| \stackrel{\text{Def.}}{=} \text{Nettozufluss}_f(t)$
 $= \sum_{v \in T} \text{Nettozufluss}_f(v)$
 da $\text{Nettozufluss}_f(v) = 0$ für alle $v \neq s, t$
 $=$

Noch mehr Schnitte

Lemma¹. Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
 Dann gilt $|f| = \text{Nettozufluss}_f(T)$.

Beweis. $|f| \stackrel{\text{Def.}}{=} \text{Nettozufluss}_f(t)$
 $= \sum_{v \in T} \text{Nettozufluss}_f(v)$
 da $\text{Nettozufluss}_f(v) = 0$ für alle $v \neq s, t$

$\underline{=}$

Noch mehr Schnitte

Lemma¹. Sei $G = (V, E)$ Graph, $S \subseteq V$ und $f: E \rightarrow \mathbb{R}$. Dann:

$$\text{Nettozufluss}_f(S) = \sum_{v \in S} \text{Nettozufluss}_f(v).$$

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
 Dann gilt $|f| = \text{Nettozufluss}_f(T)$.

Beweis. $|f| \stackrel{\text{Def.}}{=} \text{Nettozufluss}_f(t)$
 $= \sum_{v \in T} \text{Nettozufluss}_f(v)$
 da $\text{Nettozufluss}_f(v) = 0$ für alle $v \neq s, t$
 $\Rightarrow \text{Nettozufluss}_f(T)$ □

Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T)$

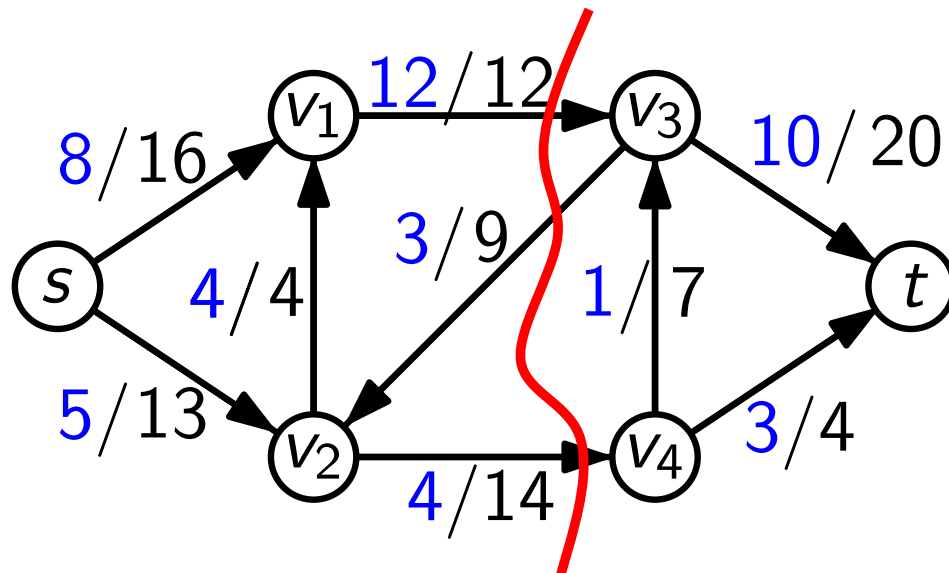
Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

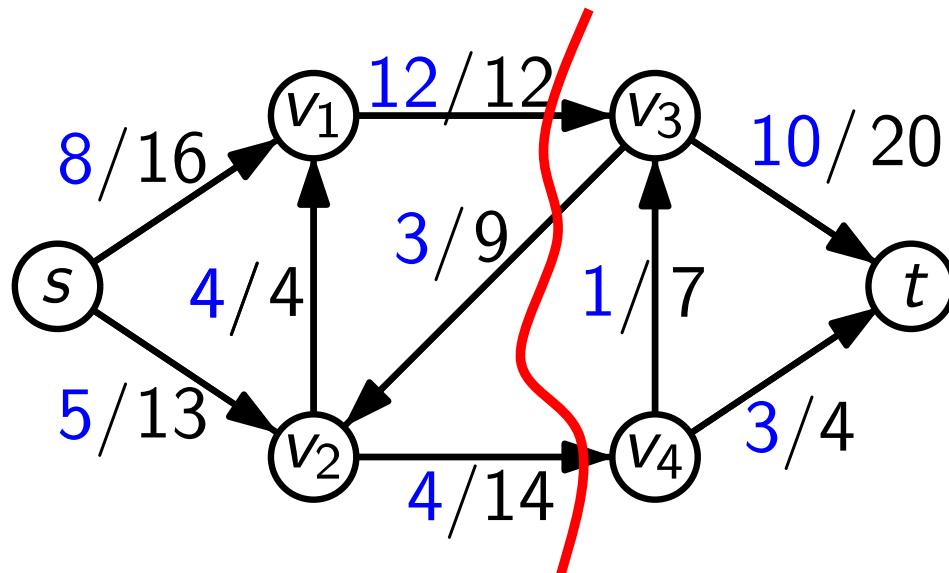
Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .



Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

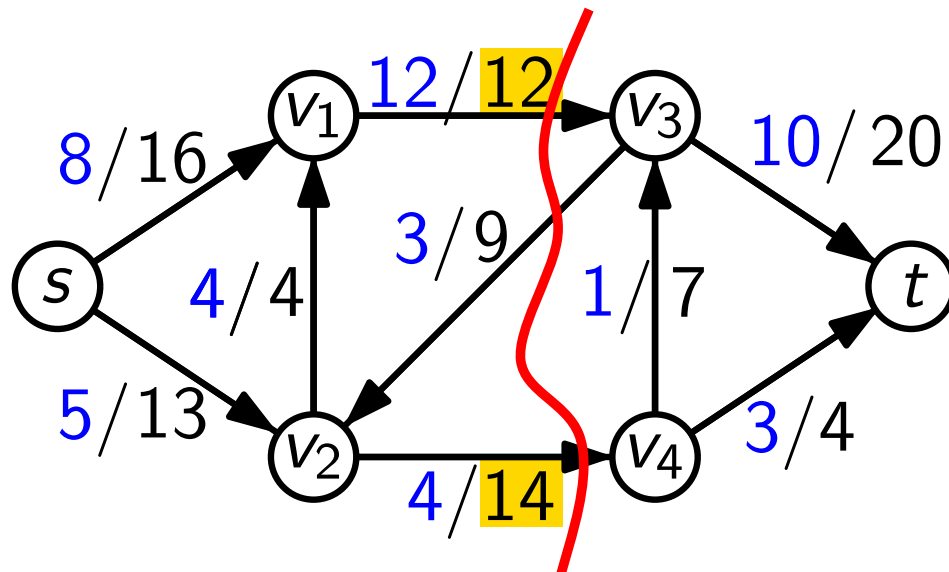
Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .



Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

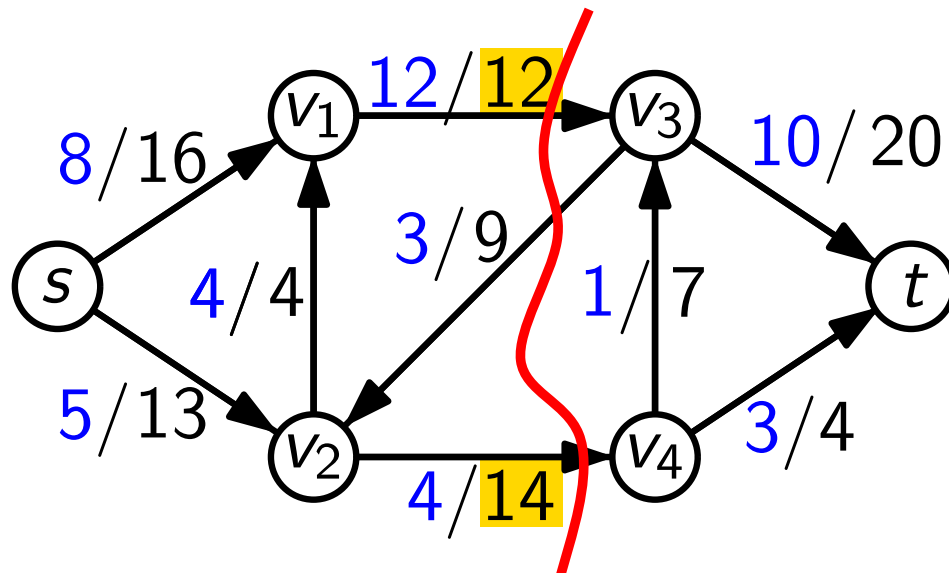


Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.



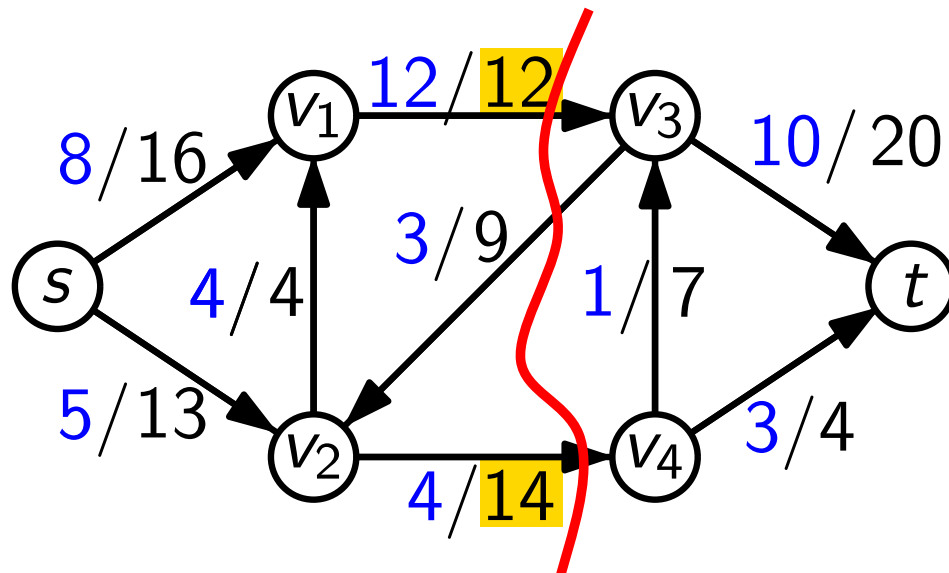
Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| =$



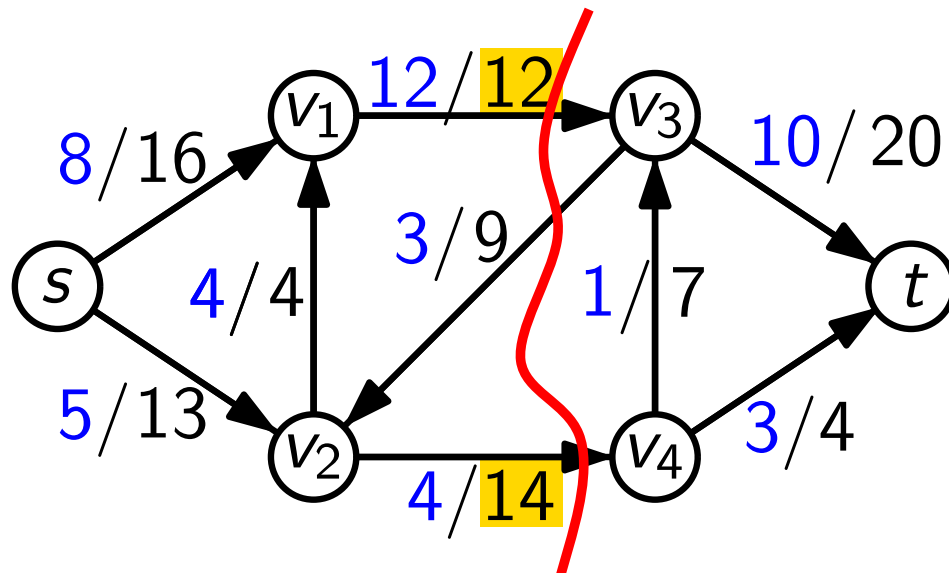
Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| =$



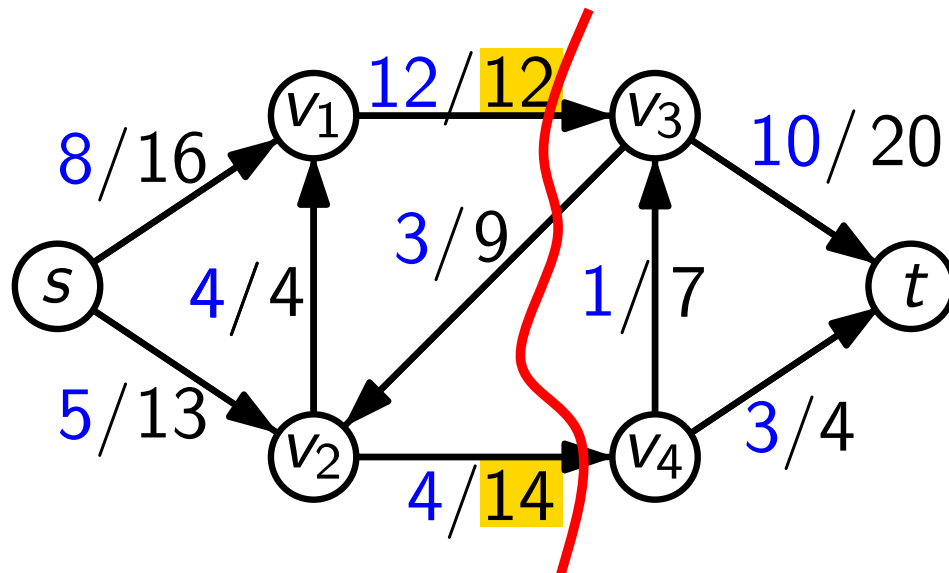
Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) =$



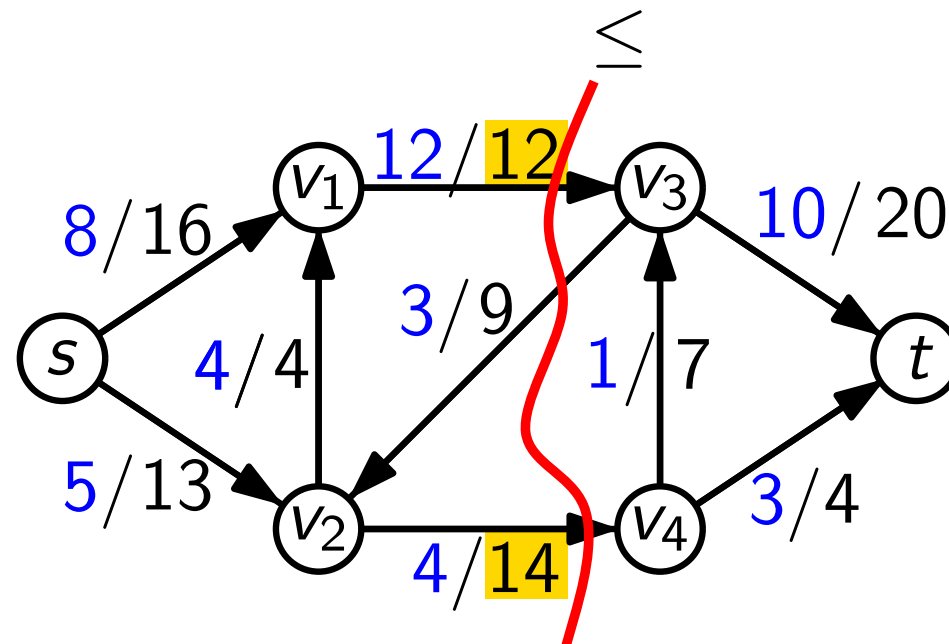
Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$



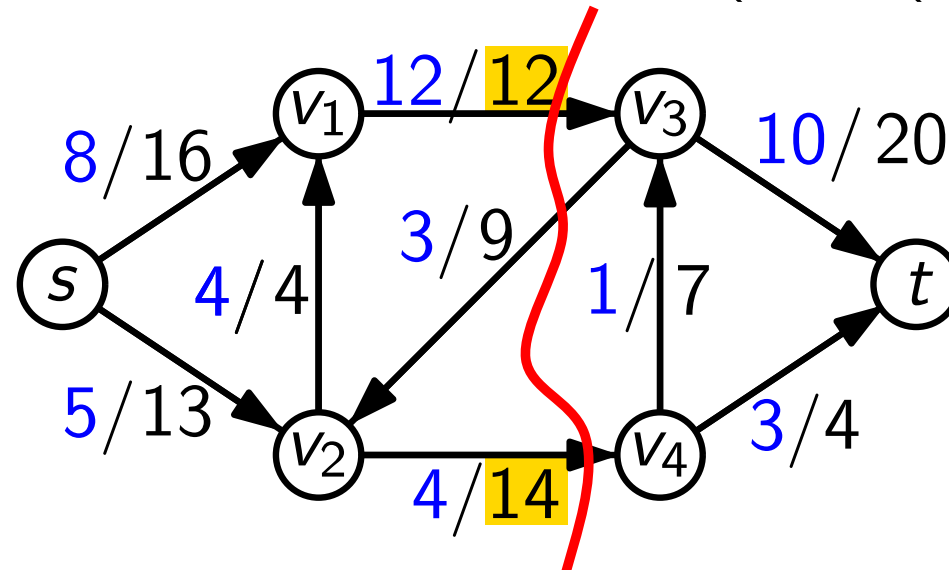
Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $\leq f(\text{Raus}(S)) \leq$



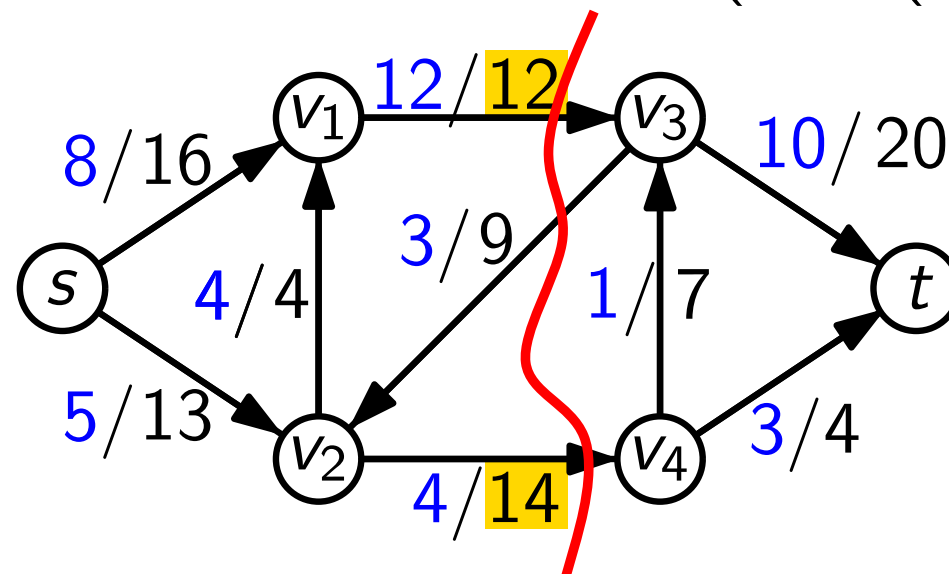
Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $\leq f(\text{Raus}(S)) \leq c(\text{Raus}(S))$
 $=$



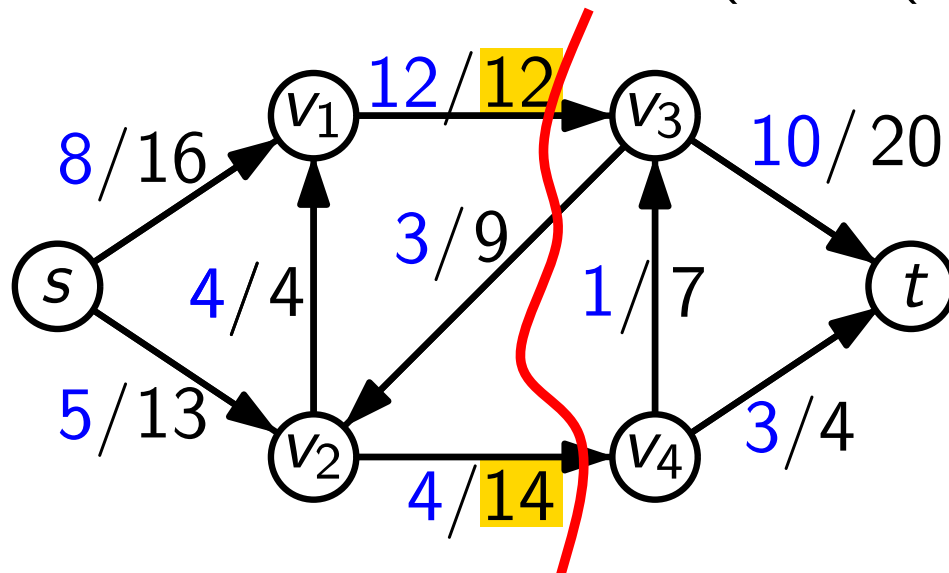
Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $\leq f(\text{Raus}(S)) \leq c(\text{Raus}(S)) = c(S) \quad \square$



Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

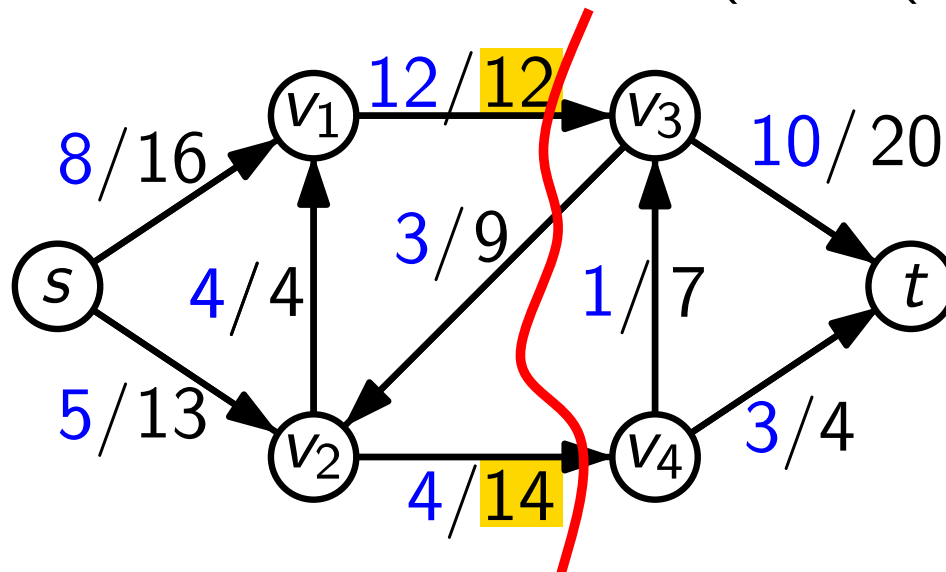
Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $\leq f(\text{Raus}(S)) \leq c(\text{Raus}(S)) = c(S) \quad \square$

Speziell:

$$\min_S c(S) \geq \max_f |f|$$



Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

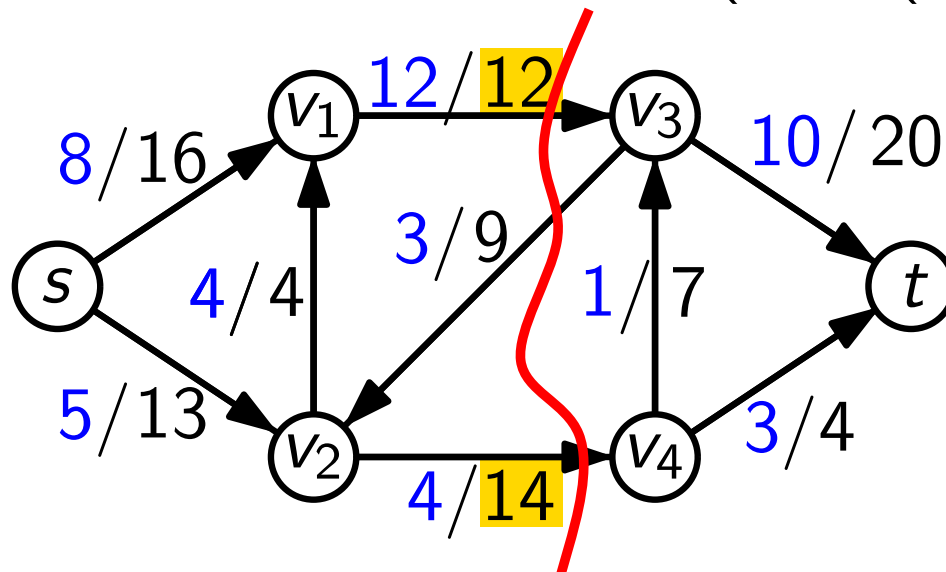
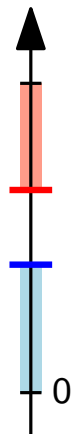
Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $\leq f(\text{Raus}(S)) \leq c(\text{Raus}(S)) = c(S) \quad \square$

Speziell:

$$\min_S c(S)$$

$$\geq \max_f |f|$$



Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

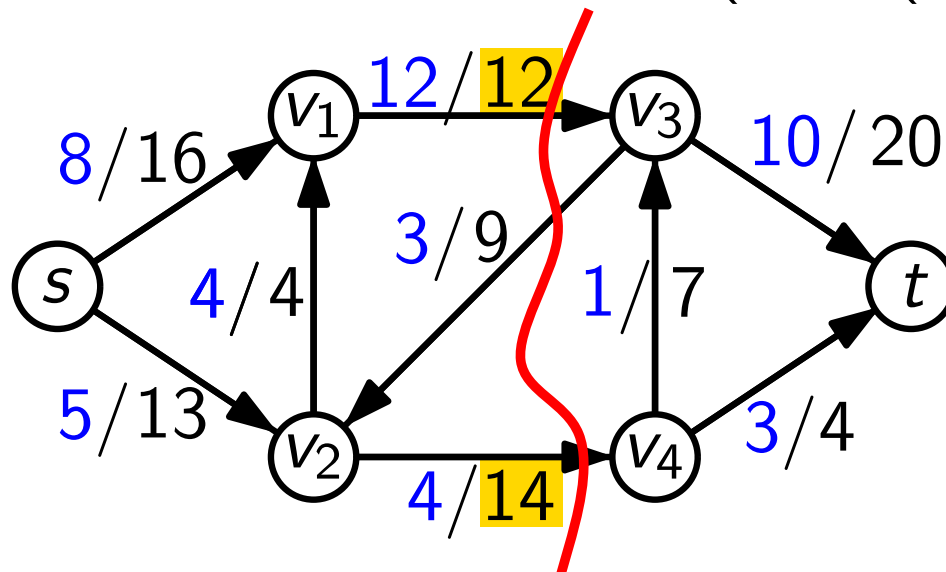
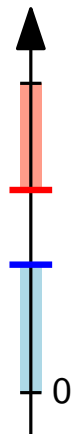
Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $\leq f(\text{Raus}(S)) \leq c(\text{Raus}(S)) = c(S)$ \square

Speziell:

$$\min_S c(S) \geq \max_f |f|$$



Korollar.

Wenn $|f| = c(S)$
 \Rightarrow

Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

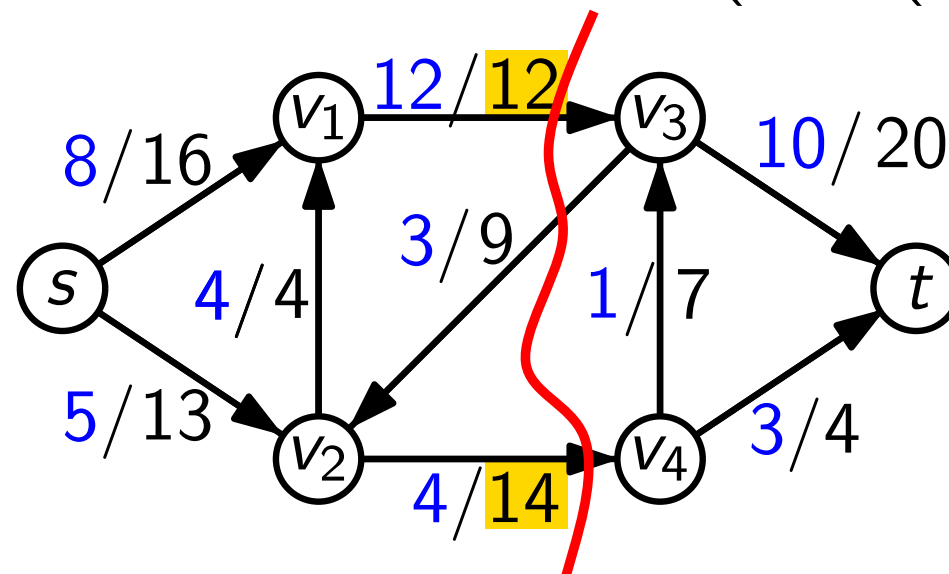
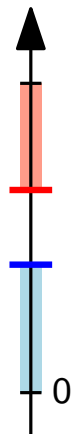
Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $\leq f(\text{Raus}(S)) \leq c(\text{Raus}(S)) = c(S)$ \square

Speziell:

$$\min_S c(S) \geq \max_f |f|$$



Korollar.

Wenn $|f| = c(S)$
 $\Rightarrow f$ max.,

Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

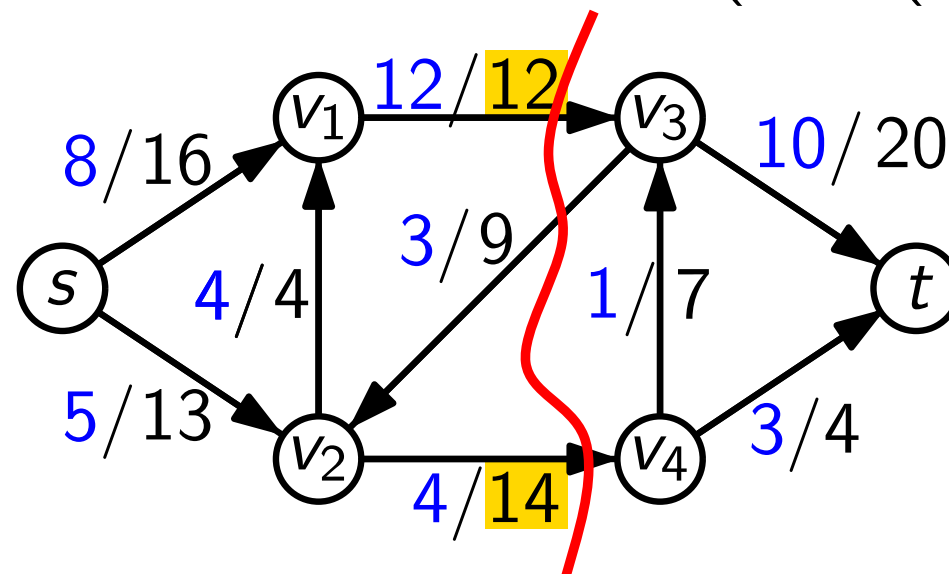
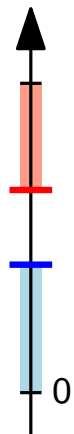
Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $\leq f(\text{Raus}(S)) \leq c(\text{Raus}(S)) = c(S) \quad \square$

Speziell:

$$\min_S c(S) \geq \max_f |f|$$



Korollar.

Wenn $|f| = c(S) \Rightarrow f$ max., (S, T) min.

Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

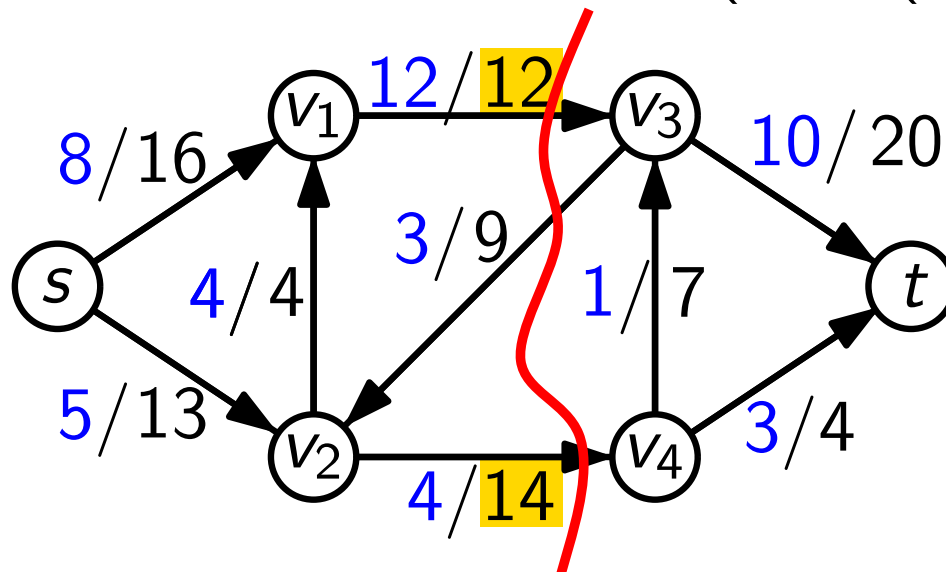
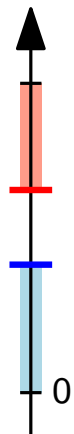
Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $\leq f(\text{Raus}(S)) \leq c(\text{Raus}(S)) = c(S) \quad \square$

Speziell:

$$\min_S c(S) \geq \max_f |f|$$



Korollar.

Wenn $|f| = c(S)$
 $\Rightarrow f$ max.,
 (S, T) min. !!

Kapazität von Schnitten

Lemma². G Graph, $s, t \in V$, f s - t -Fluss, (S, T) s - t -Schnitt.
Dann gilt $|f| = \text{Nettozufluss}_f(T) =: \text{Nettoabfluss}_f(S)$.

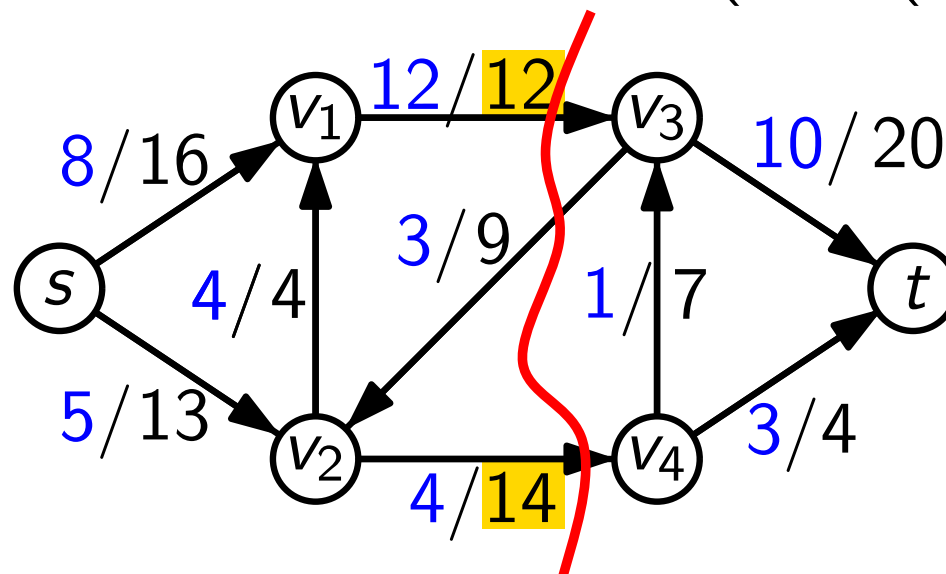
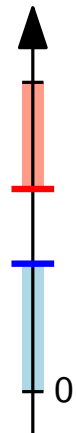
Def. G Graph mit Kap. $c: E \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

Lemma³. f zuläss. s - t -Fluss, (S, T) s - t -Schnitt $\Rightarrow |f| \leq c(S)$.

Beweis. $|f| = \text{Nettoabfluss}_f(S) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $\leq f(\text{Raus}(S)) \leq c(\text{Raus}(S)) = c(S)$ \square

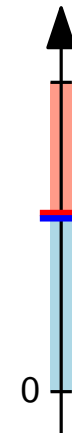
Speziell:

$$\min_S c(S) \geq \max_f |f|$$



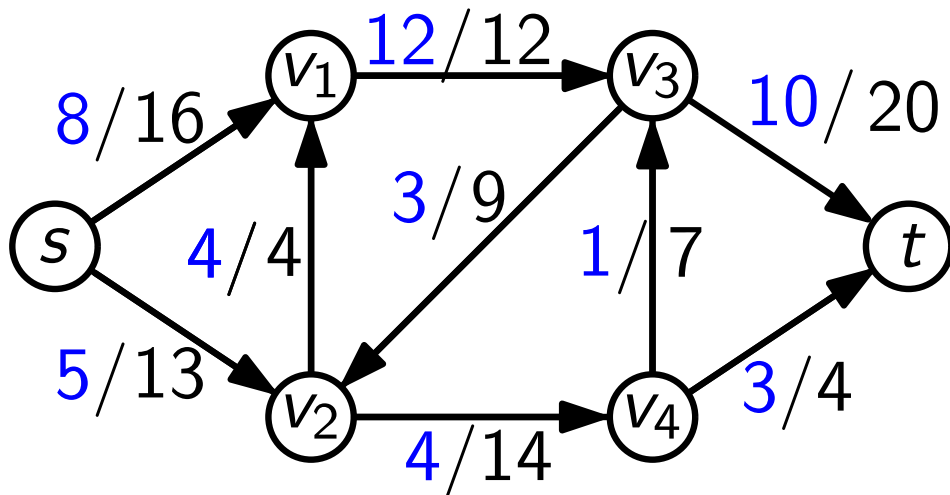
Korollar.

Wenn $|f| = c(S)$
 $\Rightarrow f$ max.,
 (S, T) min. !!



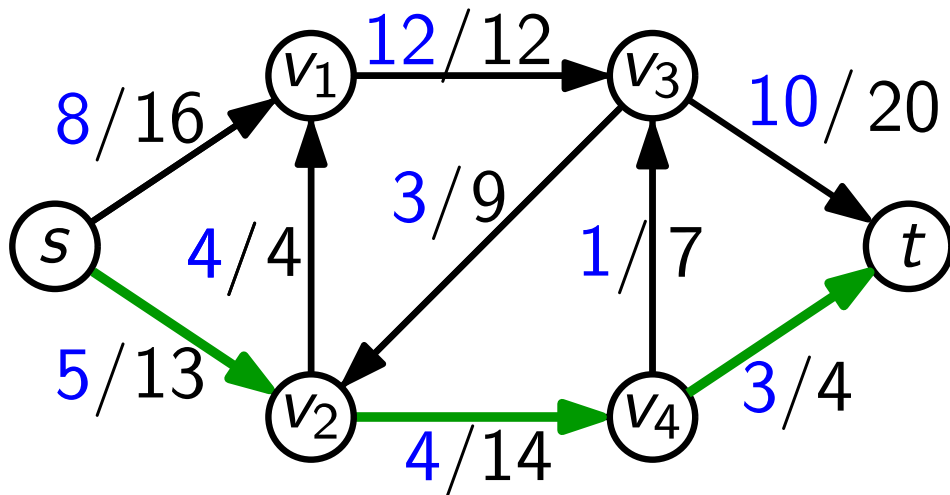
Residualnetz

Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.



Residualnetz

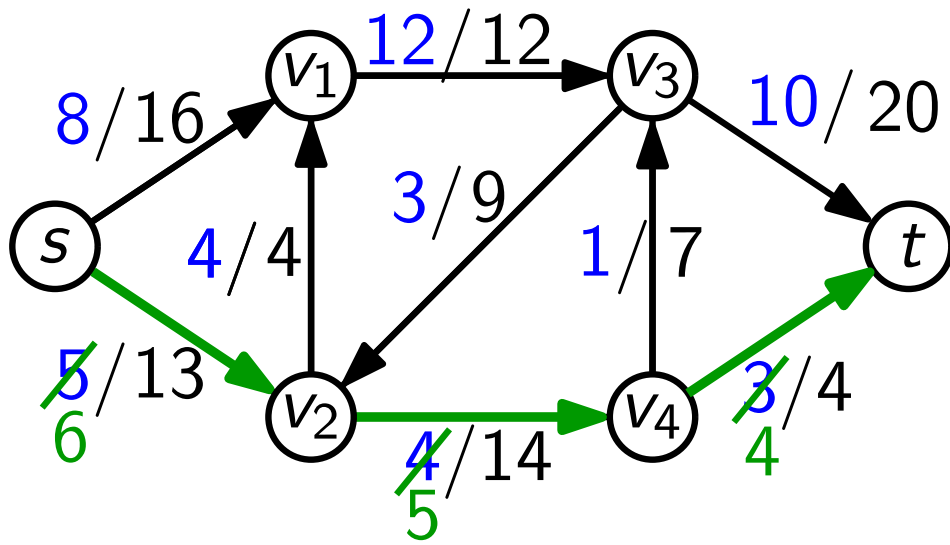
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.



Residualnetz

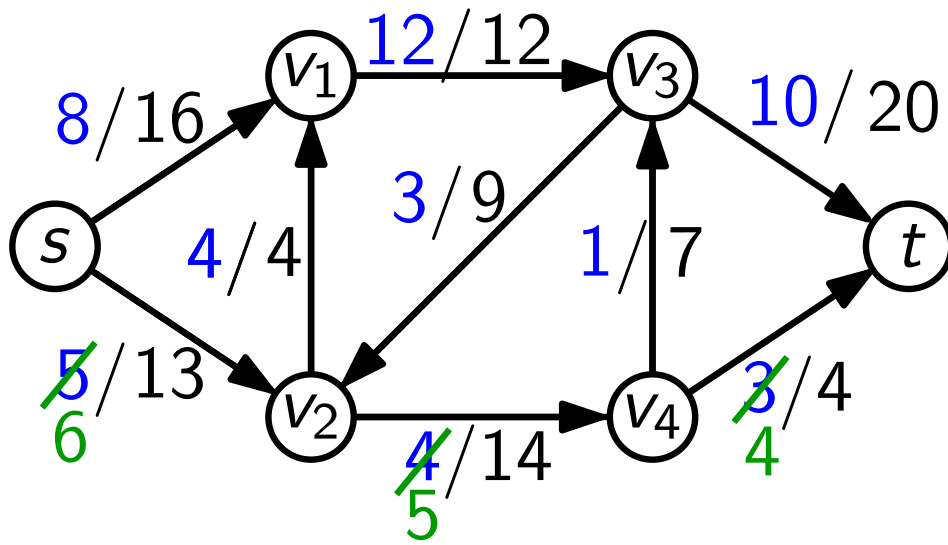
Beob.

Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.



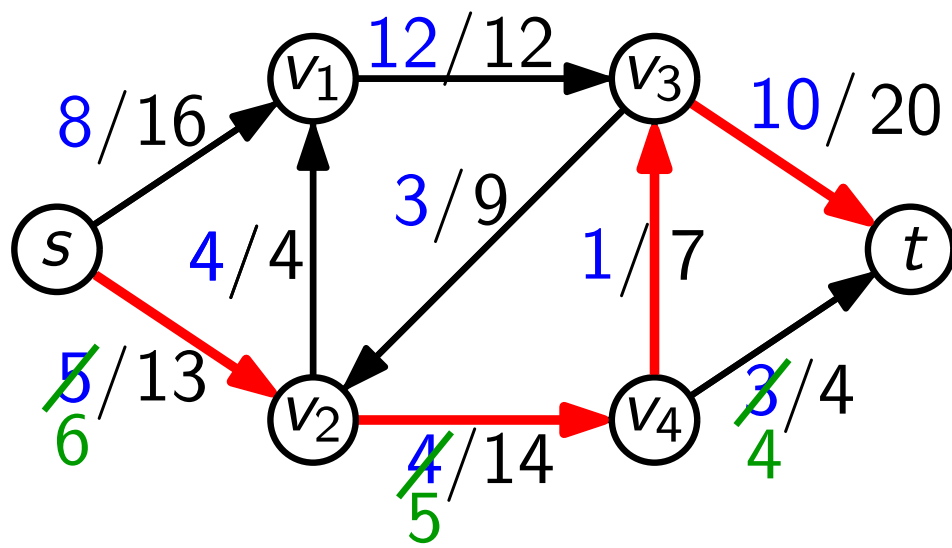
Residualnetz

Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.



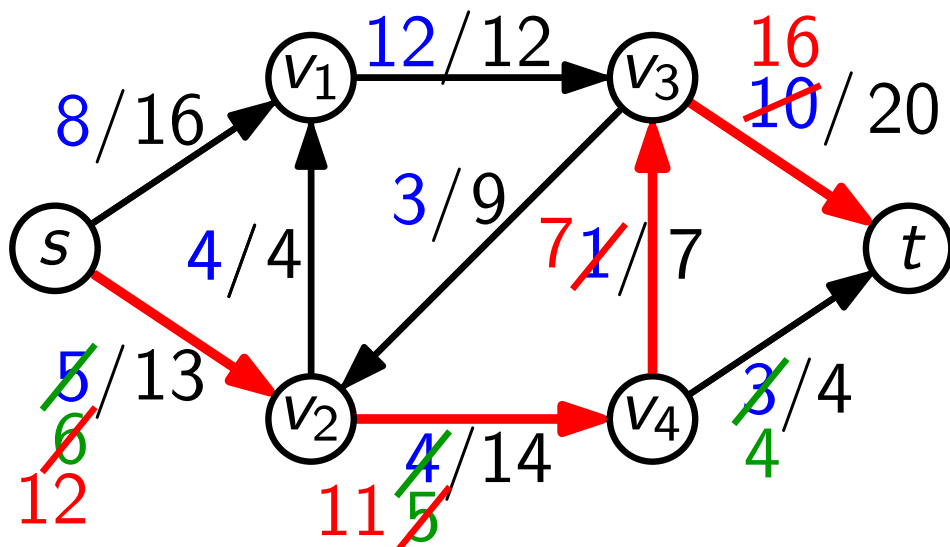
Residualnetz

Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.



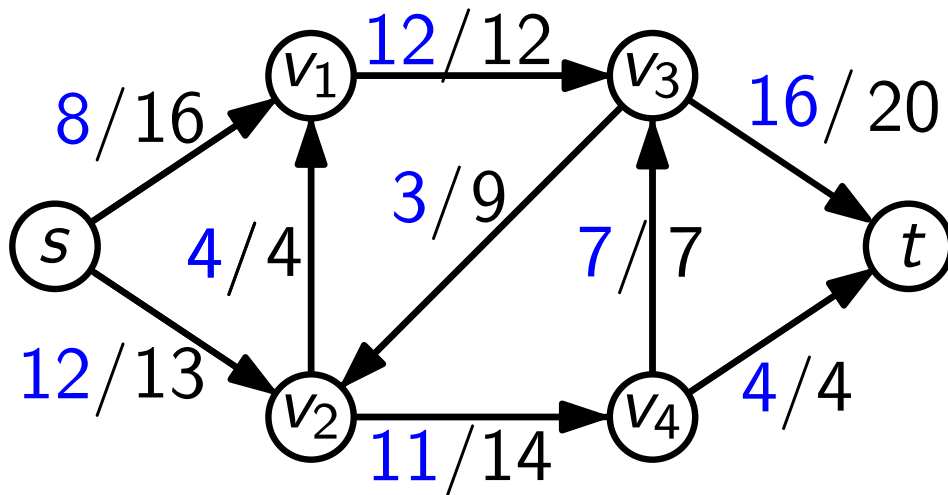
Residualnetz

Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.



Residualnetz

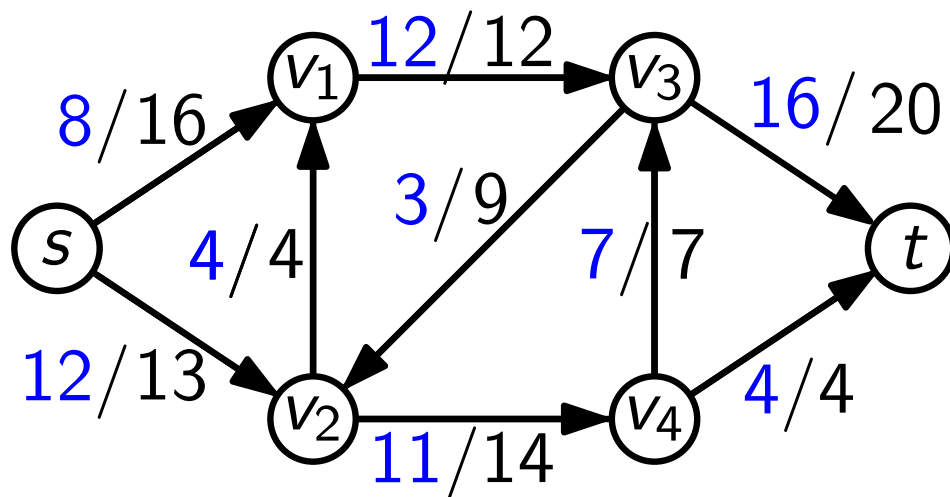
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.



Residualnetz

Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

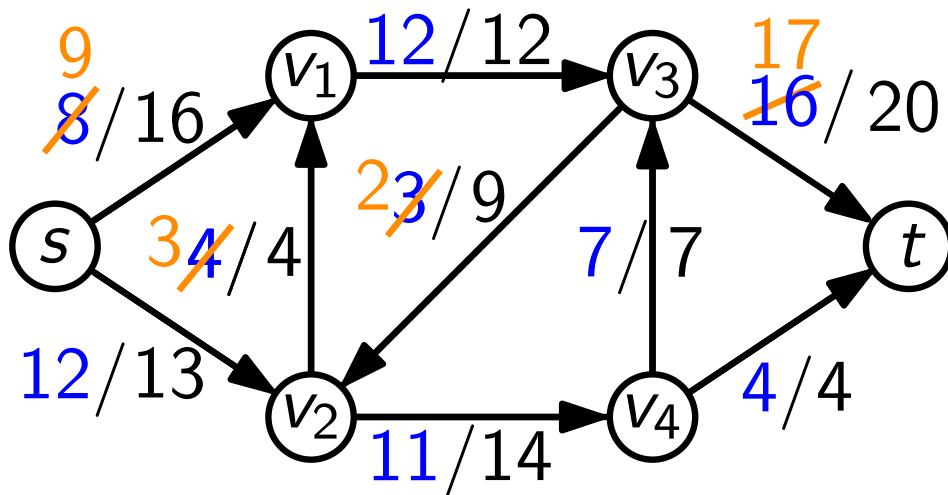
Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.



Residualnetz

Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.



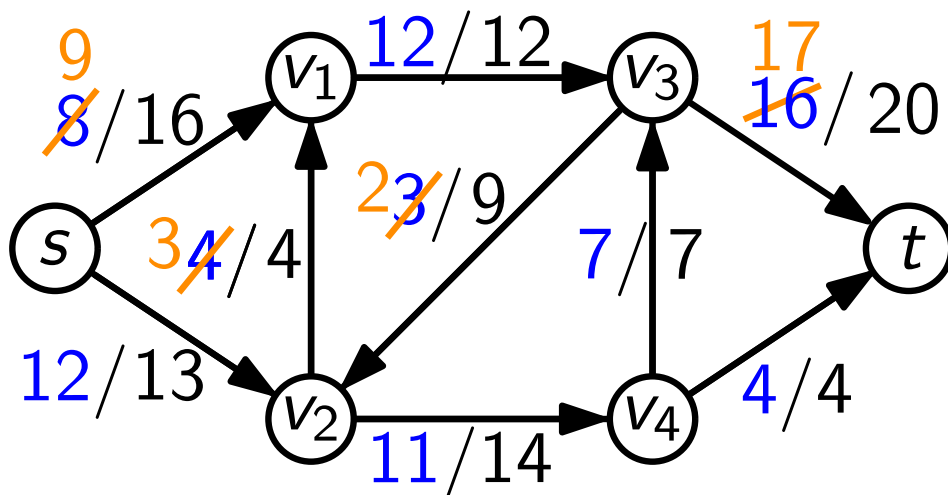
Residualnetz

Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$
- $-e := vu$ falls $f(e) > 0$



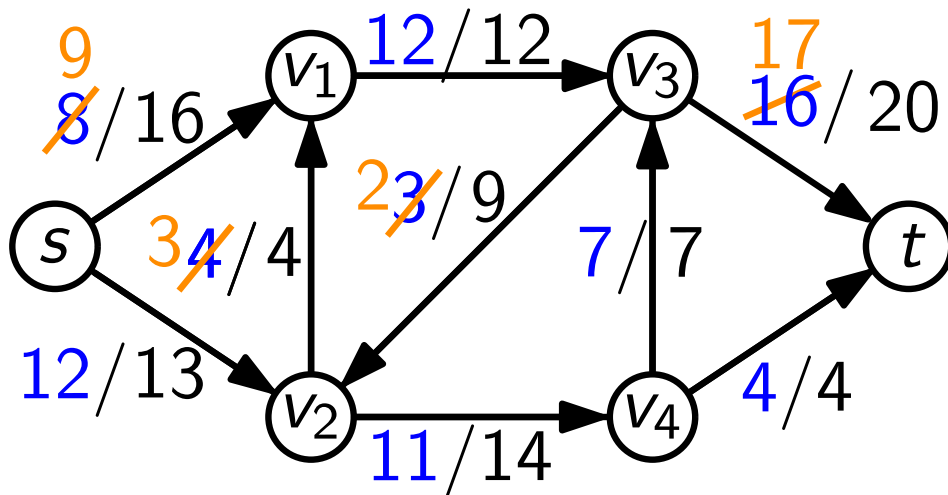
Residualnetz

Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$



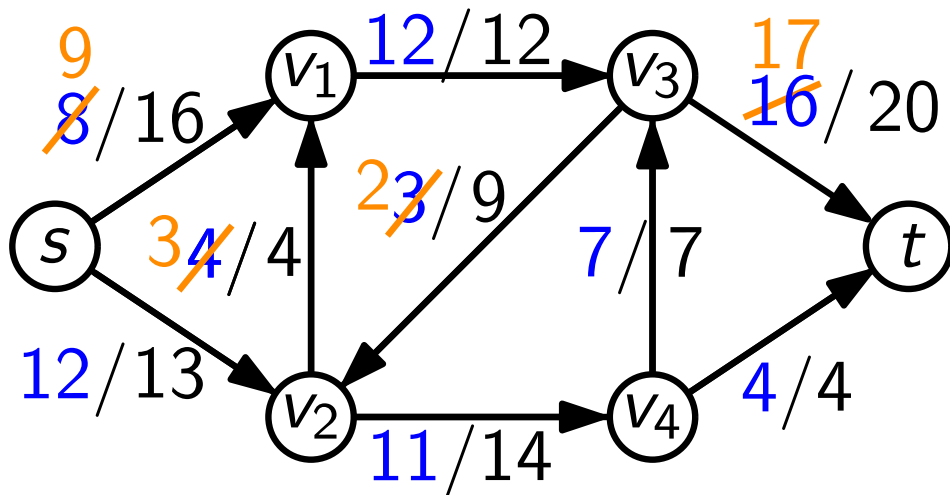
Residualnetz

Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$



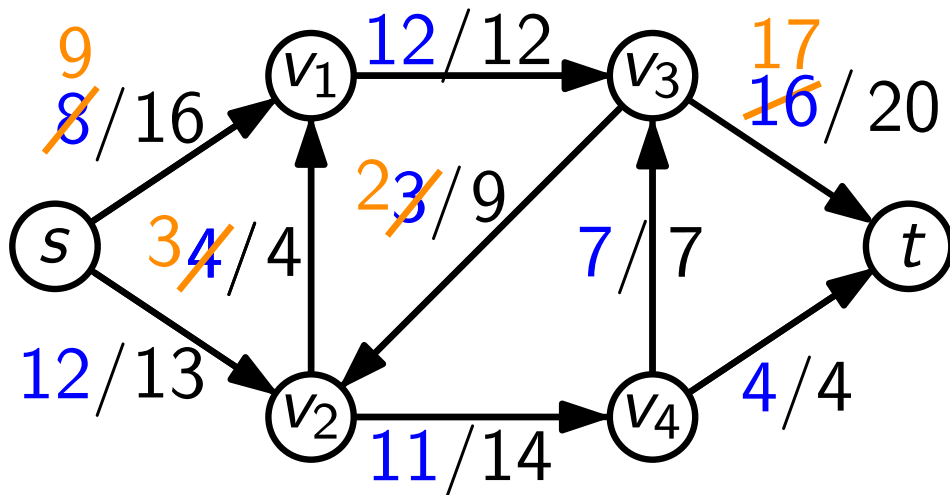
Residualnetz

Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$



Residualnetz

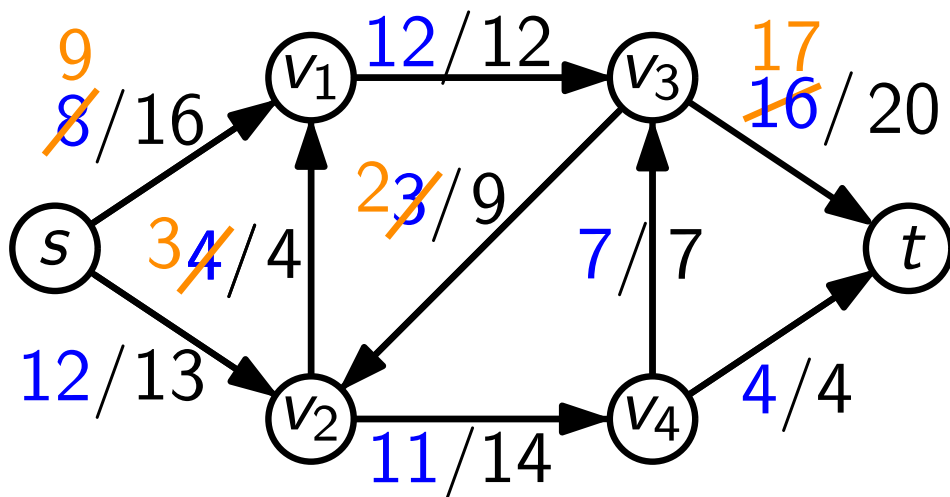
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

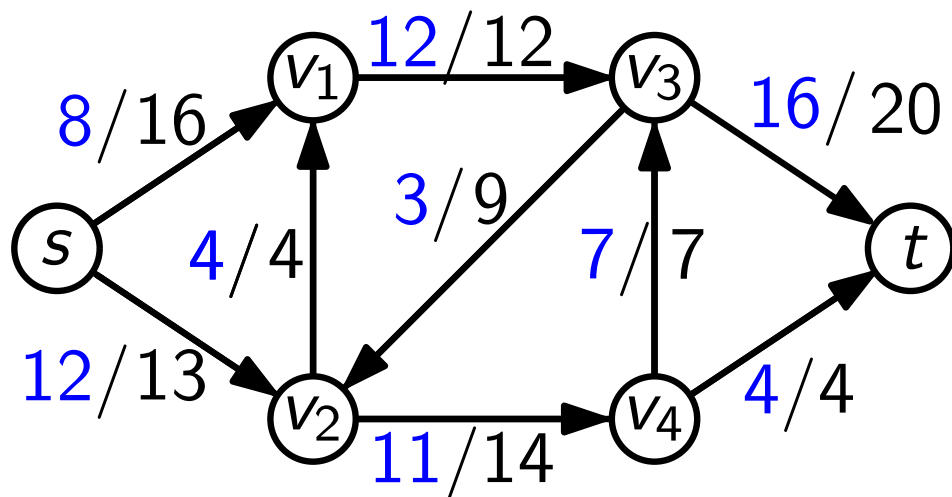
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

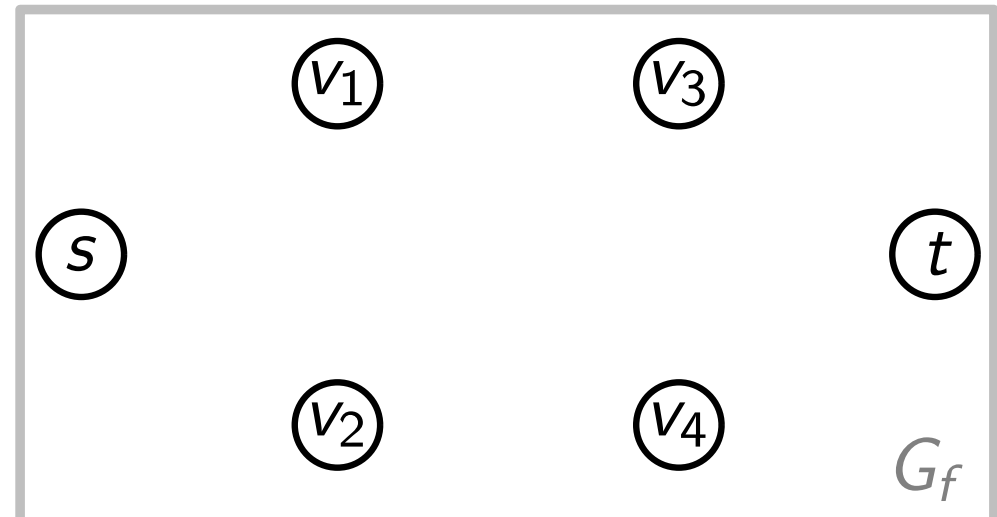
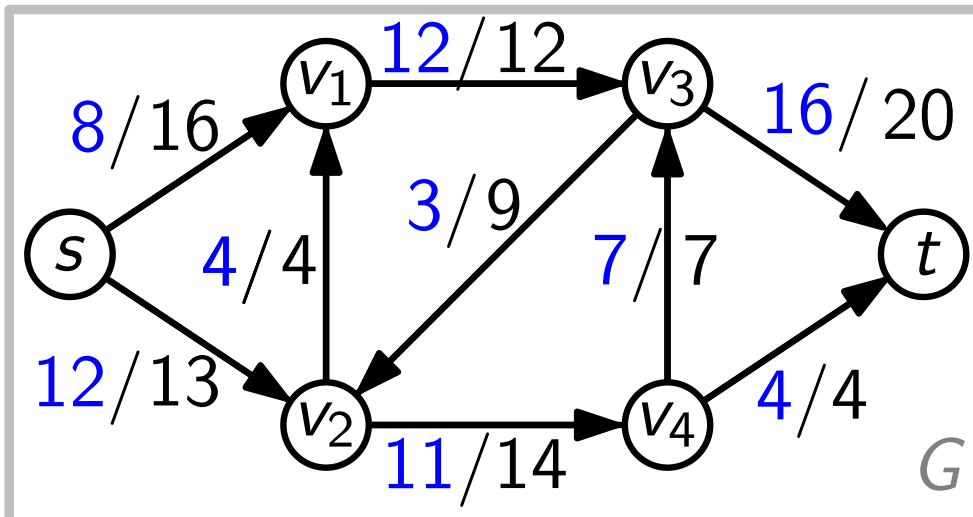
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

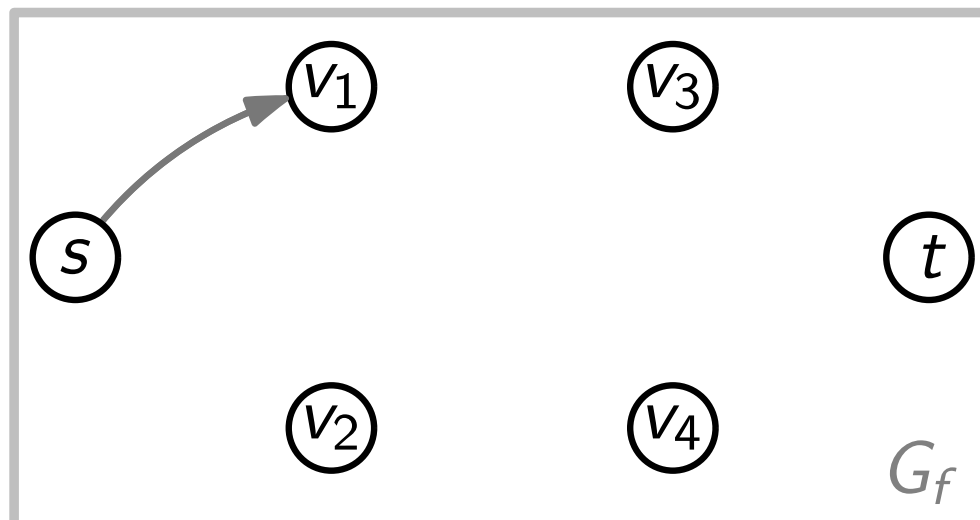
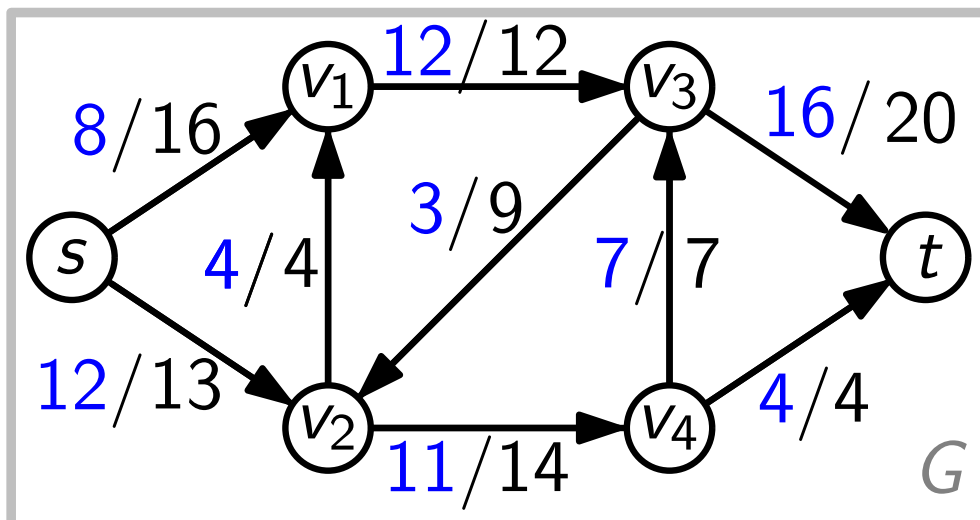
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

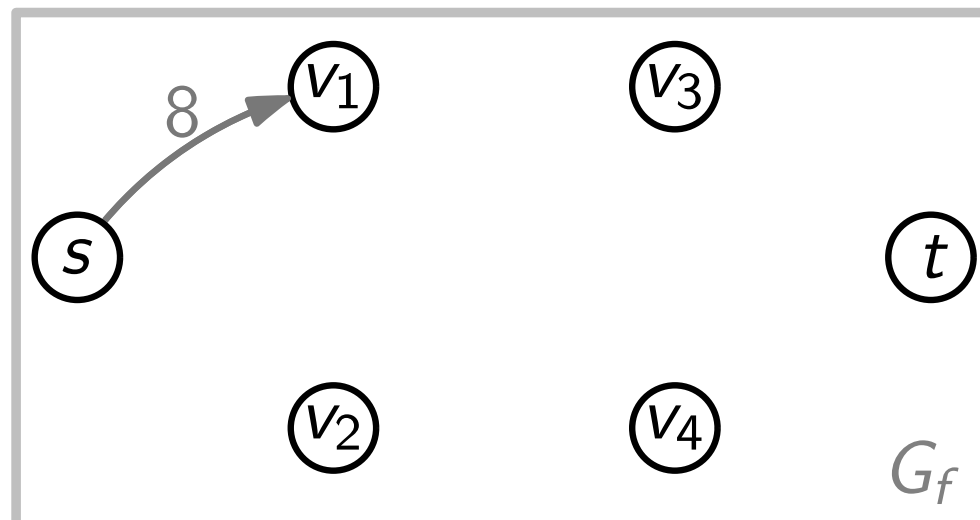
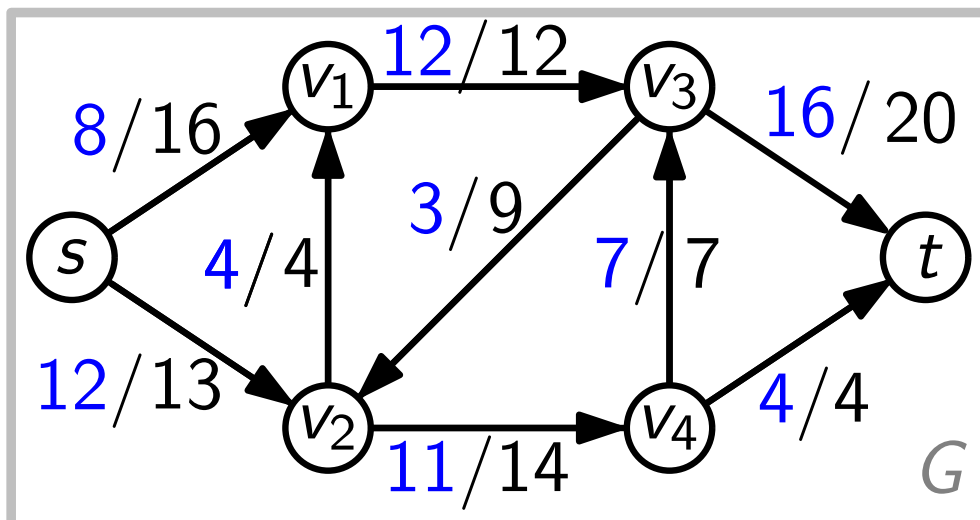
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

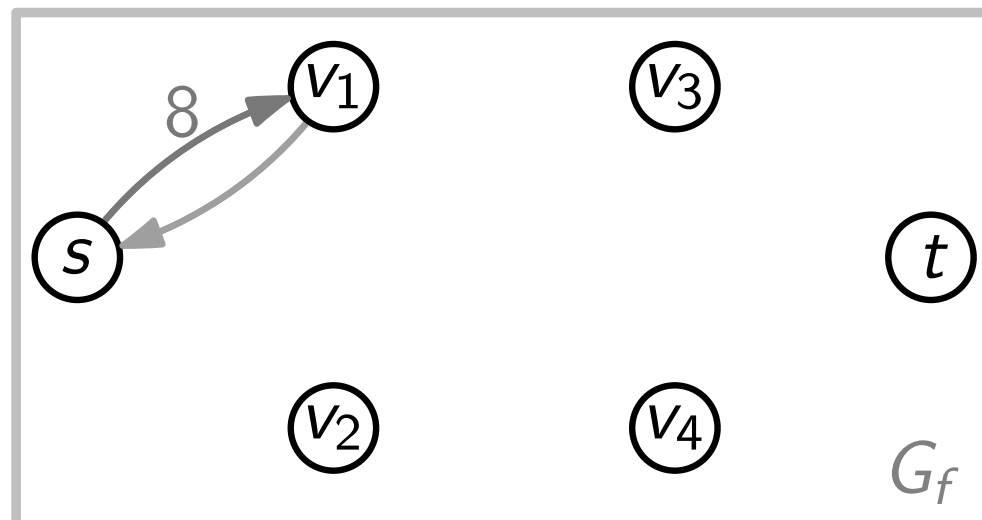
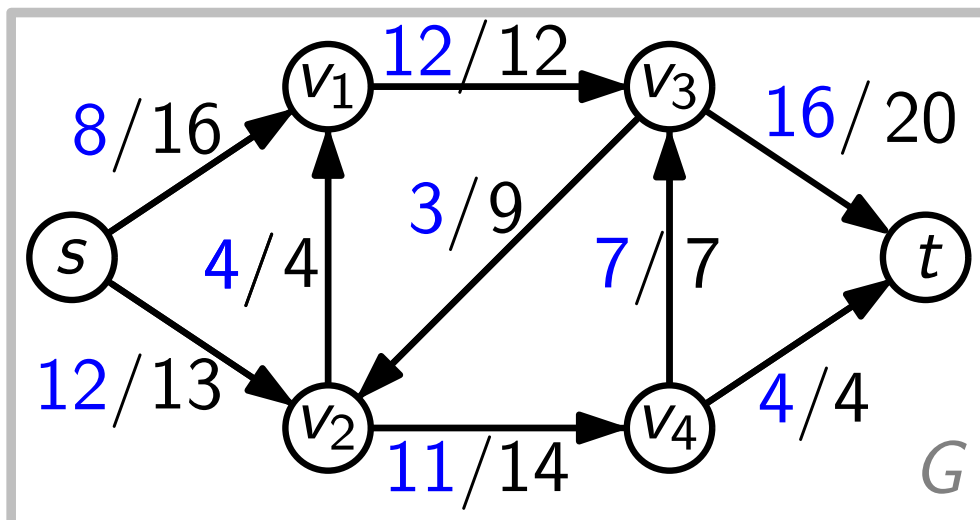
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

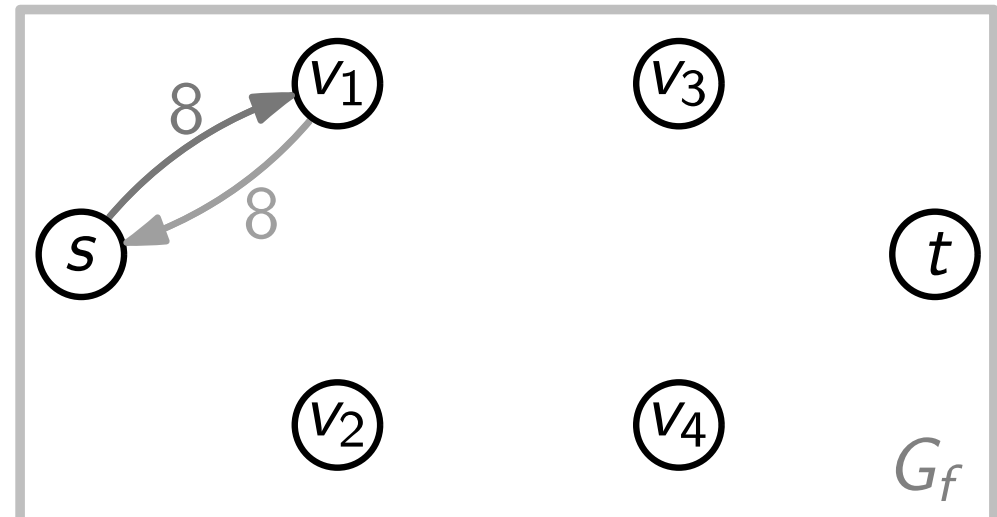
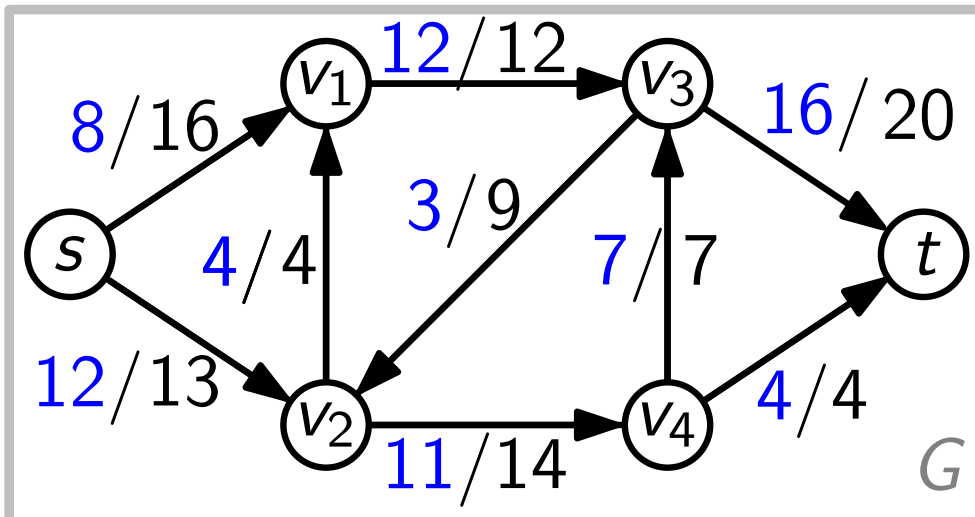
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

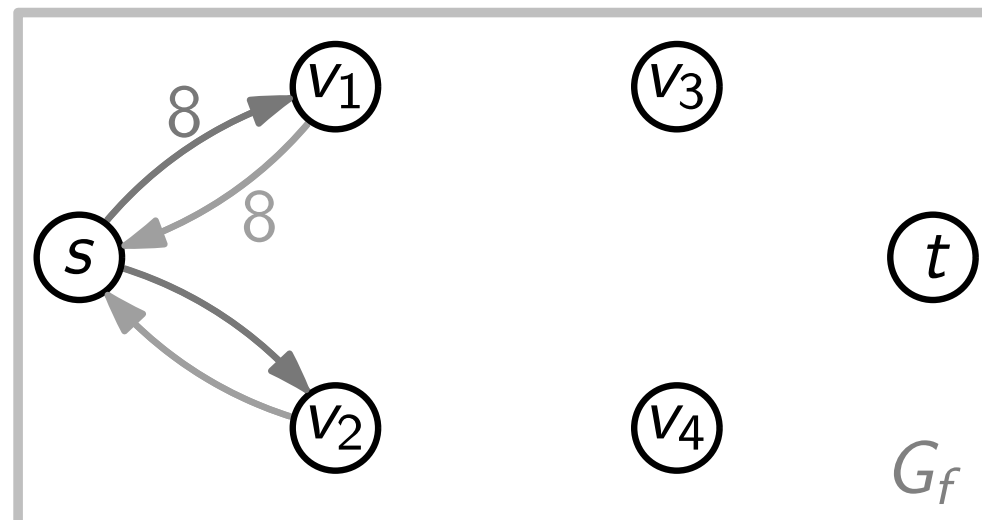
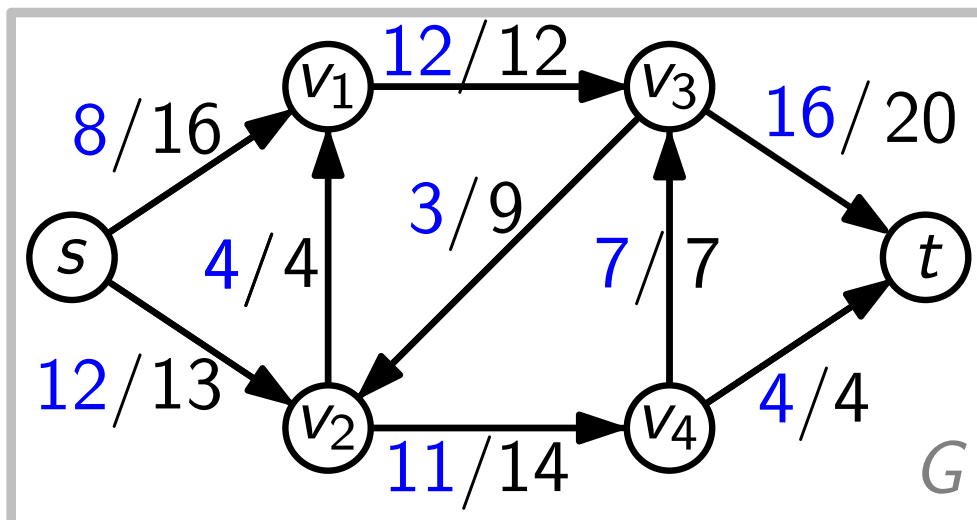
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

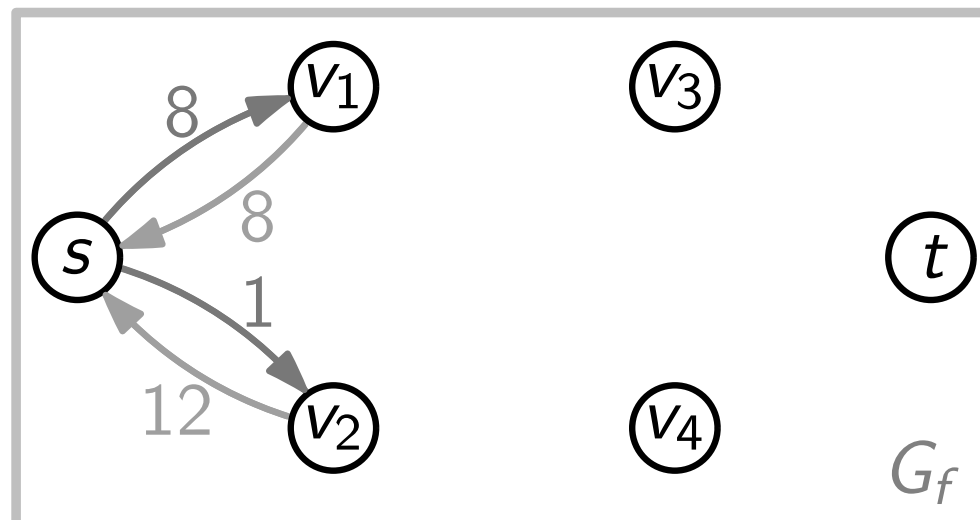
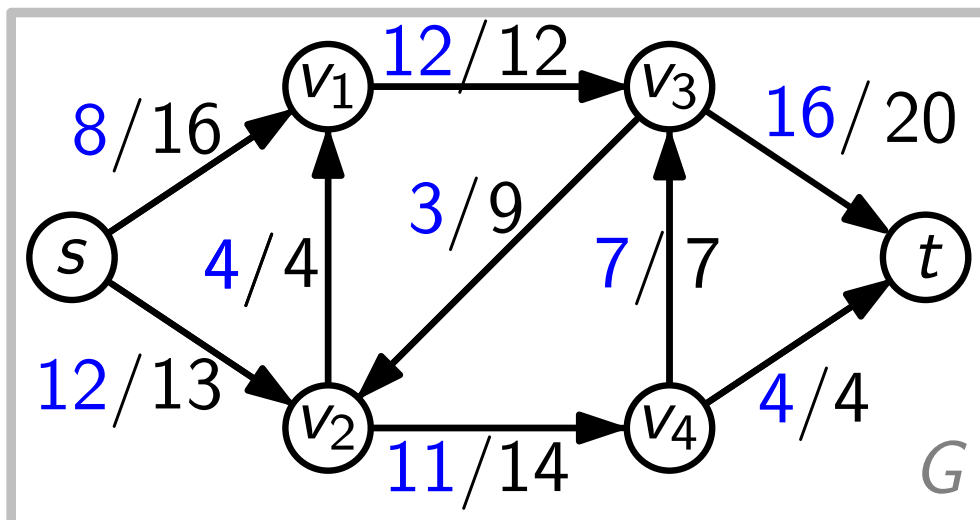
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

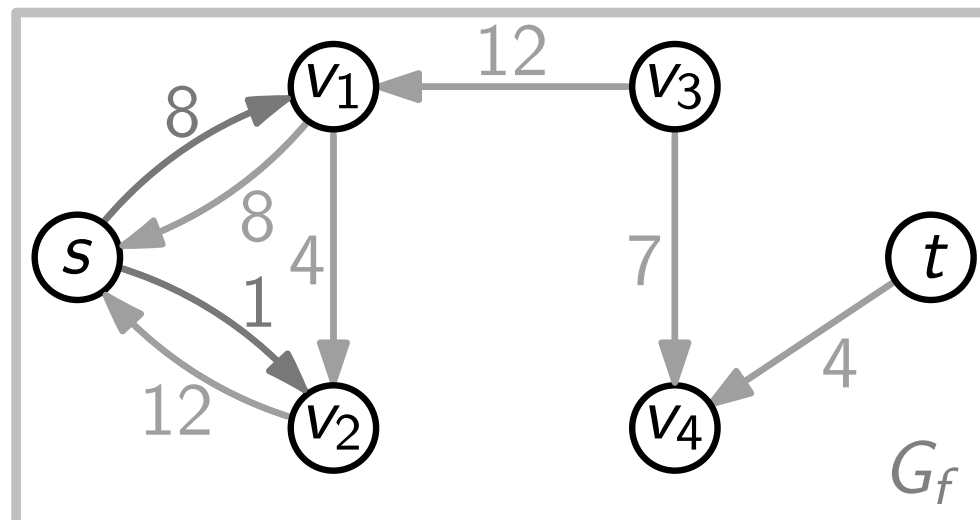
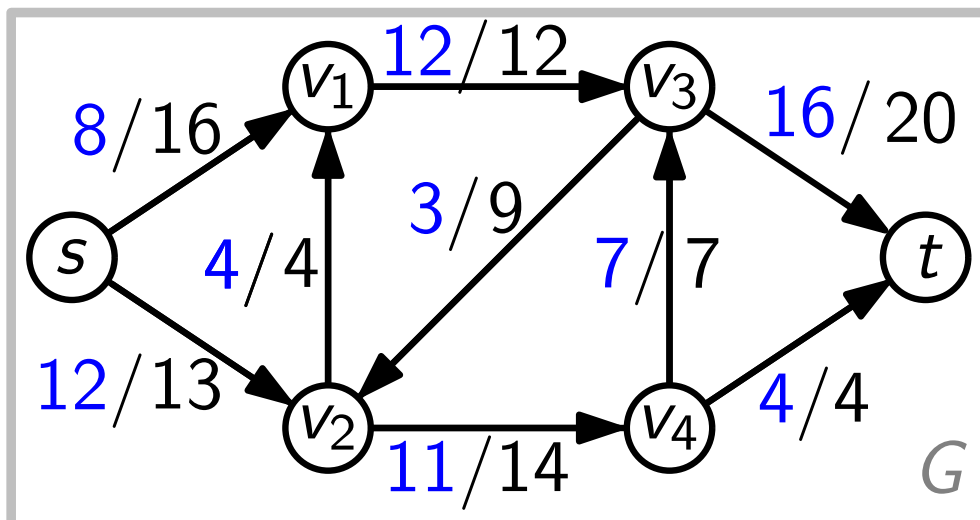
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

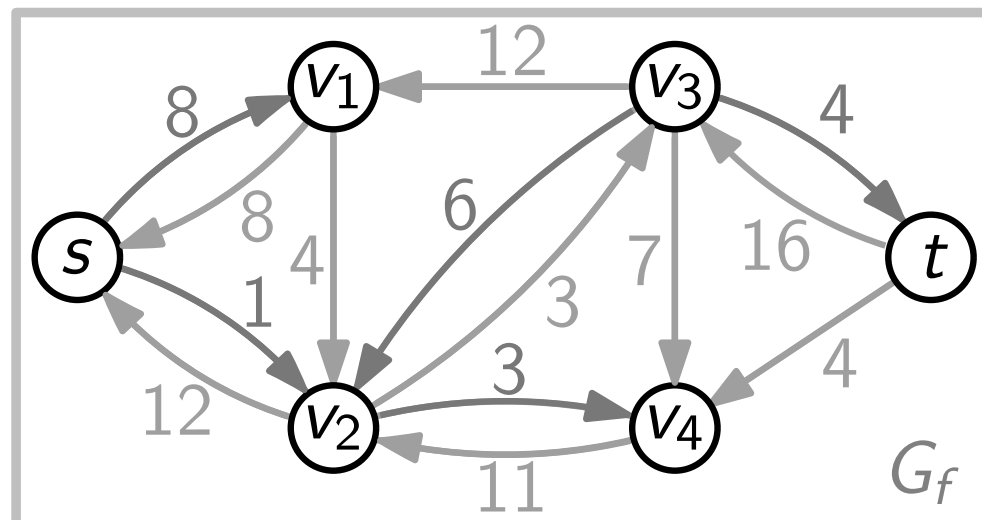
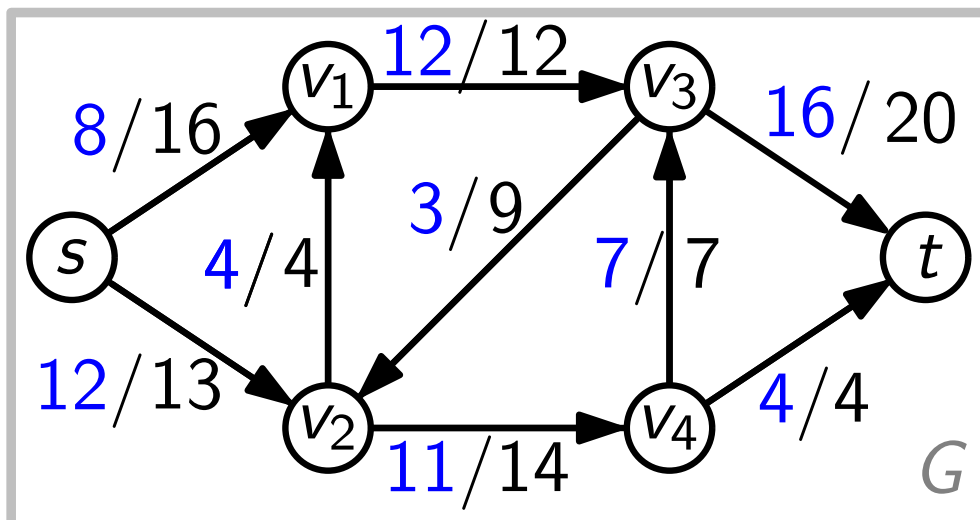
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

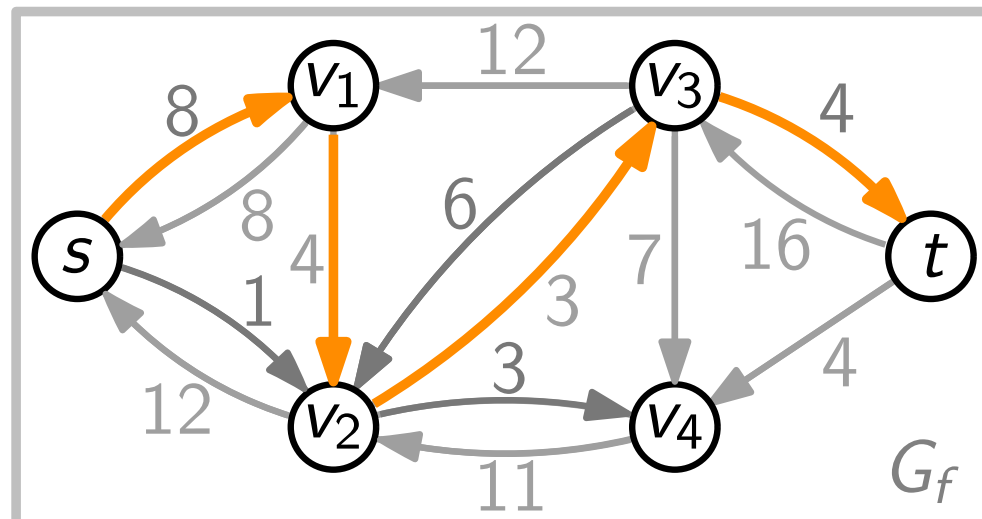
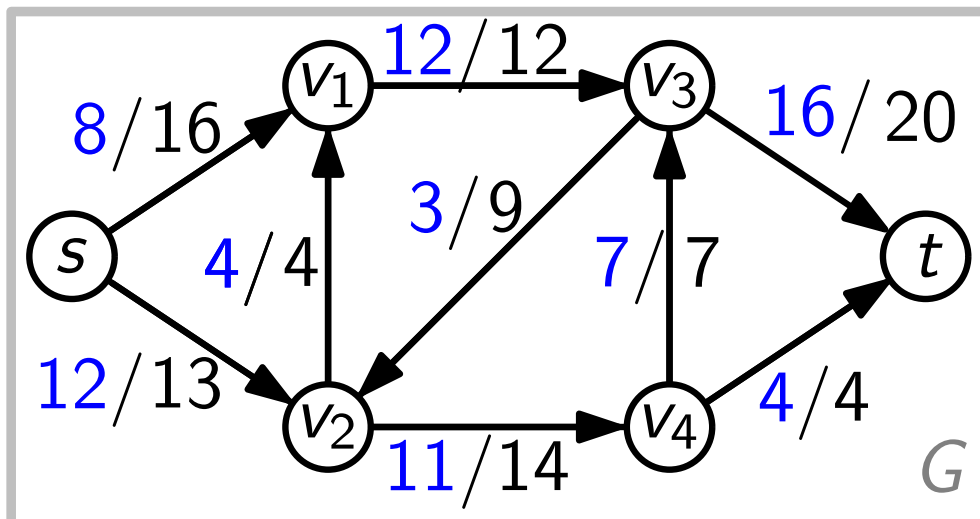
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Residualnetz

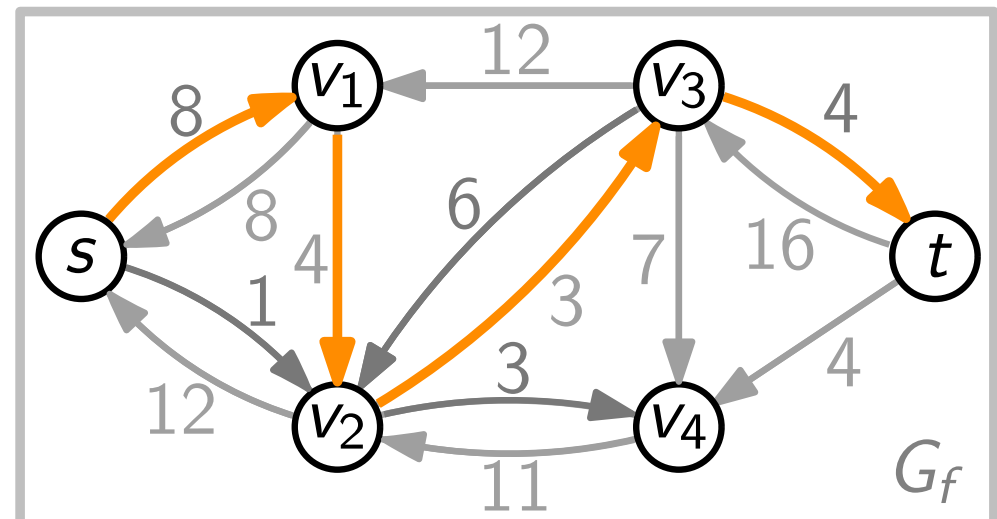
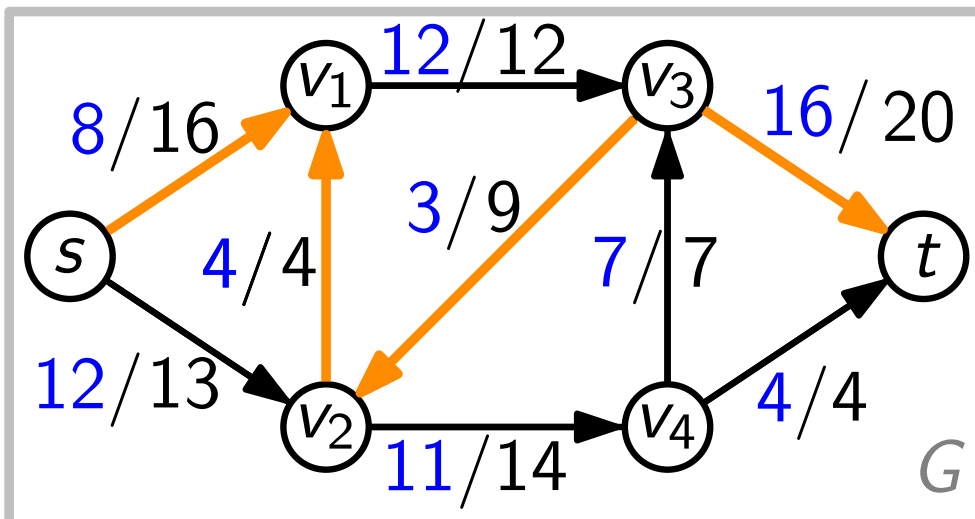
Beob. Falls es einen s - t -Weg gibt, bei dem auf keiner Kante die Kapazität ausgeschöpft ist, können wir f vergrößern.

Aber: Falls es **keinen** solchen s - t -Weg gibt, so ist f **nicht** unbedingt maximal.

Def. Der *Residualgraph* $G_f = (V, E_f)$ enthält für jede Kante $e = uv$ von $G = (V, E)$ die Kante(n)

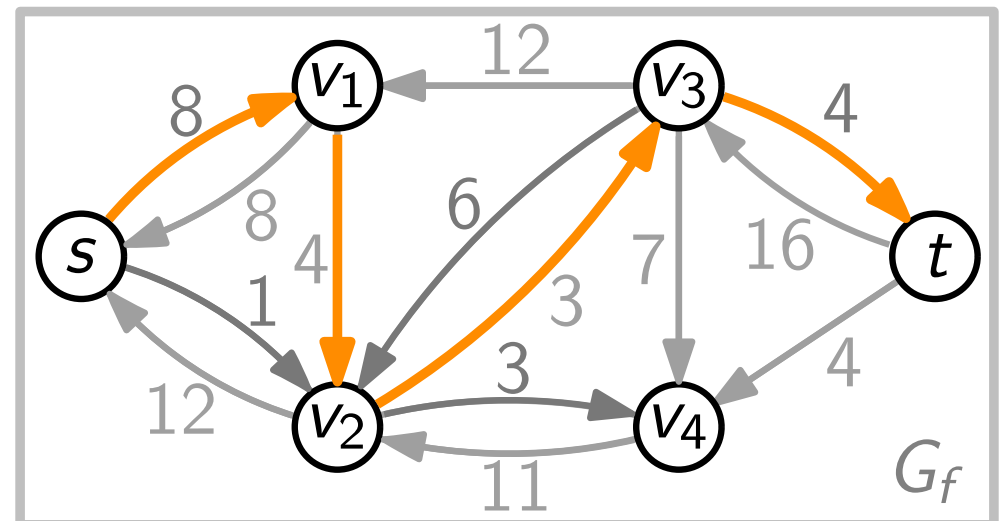
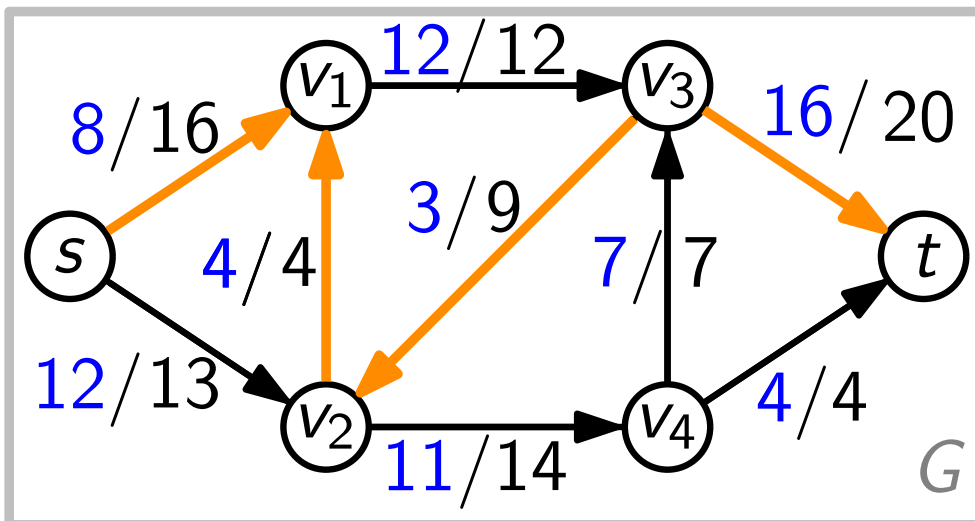
- $+e := uv$ falls $f(e) < c(e)$ mit $c_f(+e) := c(e) - f(e)$
- $-e := vu$ falls $f(e) > 0$ mit $c_f(-e) := f(e)$

Residualkapazitäten



Flussvergrößernde Wege

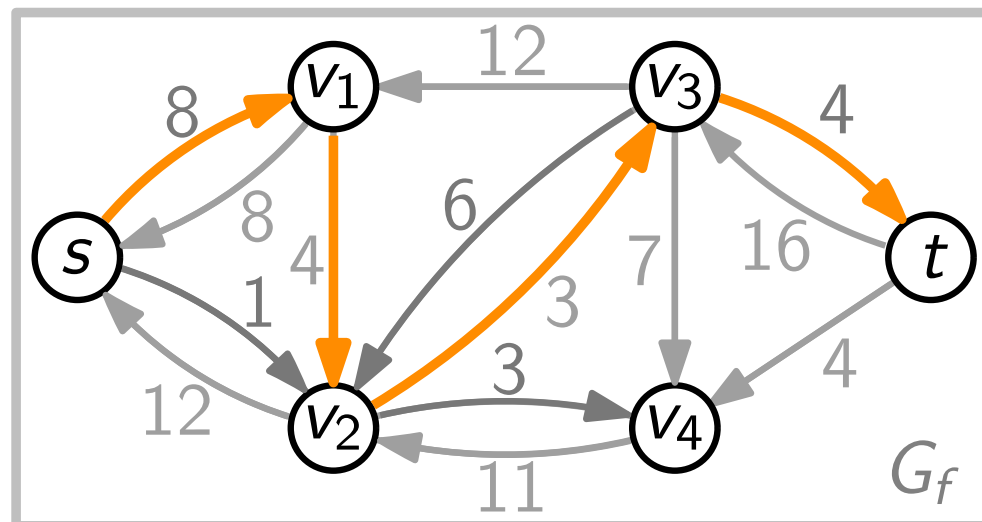
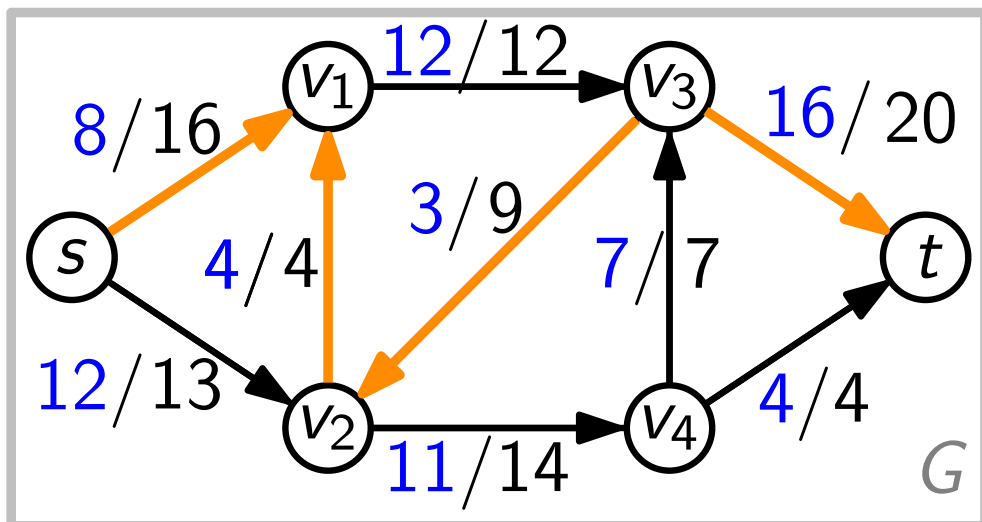
Def. Ein s - t -Weg W in G_f heißt *flussvergrößernder Weg* für f .



Flussvergrößernde Wege

Def. Ein s - t -Weg W in G_f heißt *flussvergrößernder Weg* für f . Die Residualkapazität von W ist

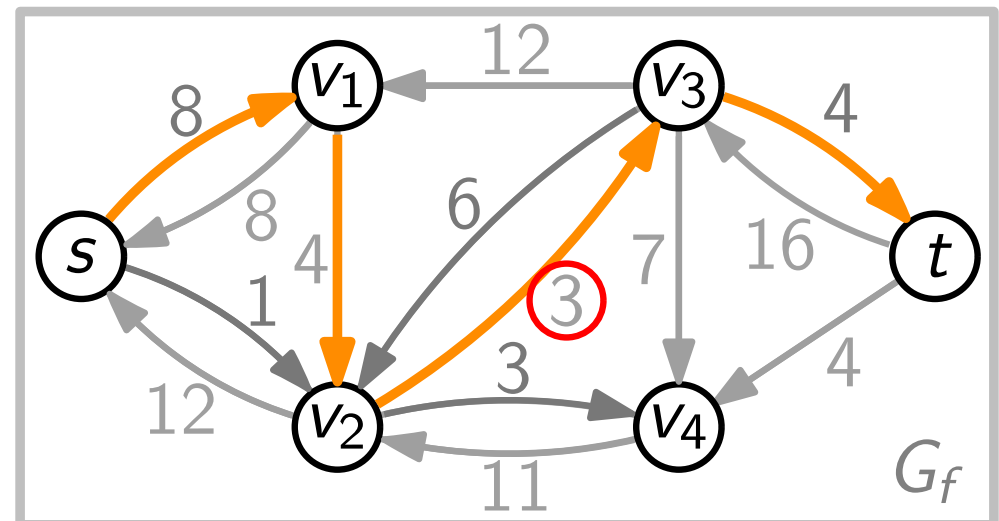
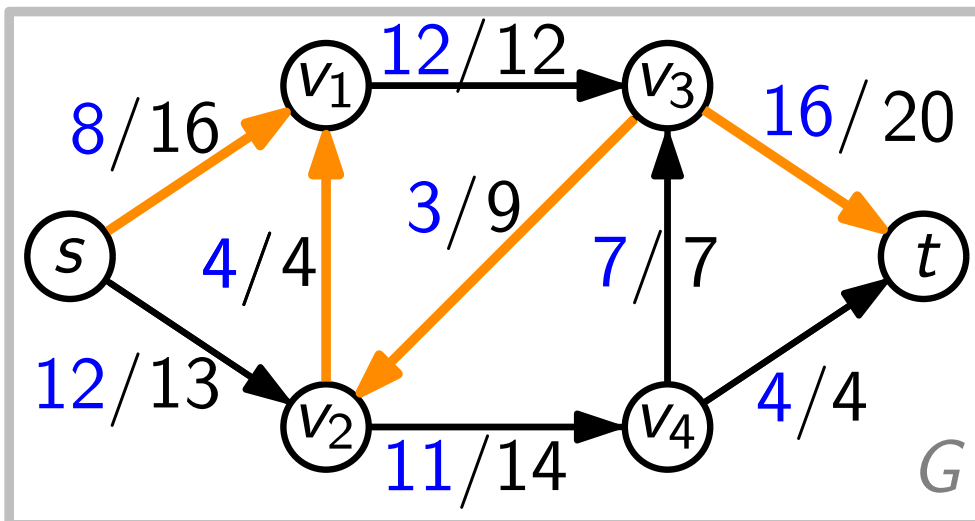
$$\Delta_W := \min_{e \in W} c_f(e),$$



Flussvergrößernde Wege

Def. Ein s - t -Weg W in G_f heißt *flussvergrößernder Weg* für f . Die Residualkapazität von W ist

$$\Delta_W := \min_{e \in W} c_f(e),$$

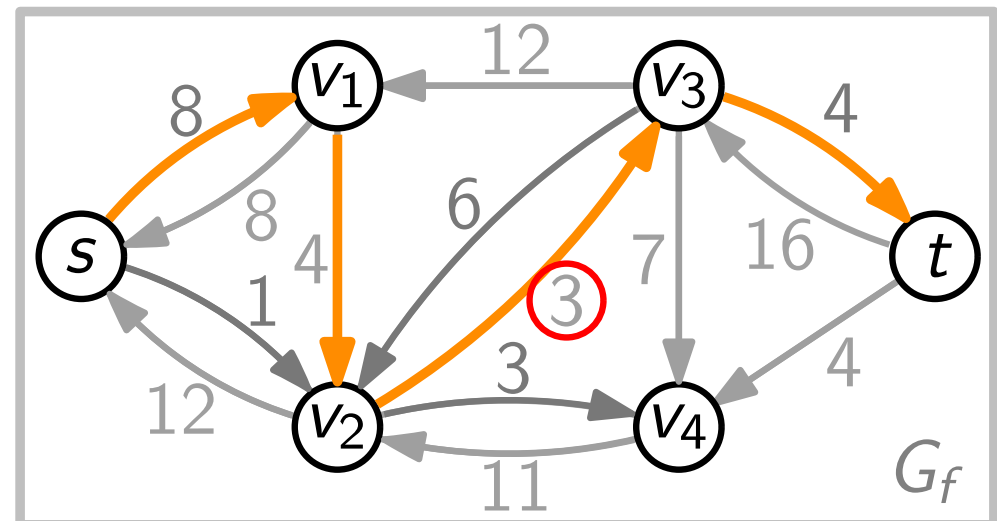
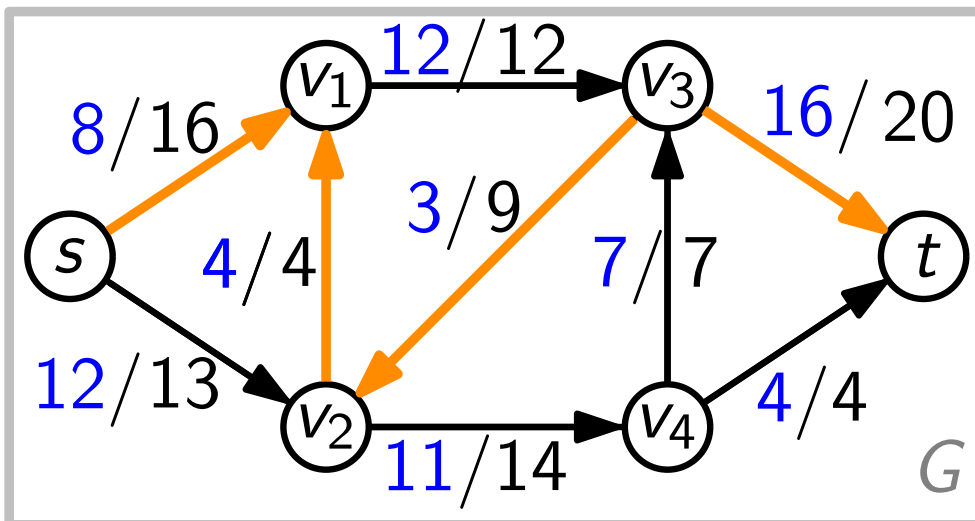


Flussvergrößernde Wege

Def. Ein s - t -Weg W in G_f heißt *flussvergrößernder Weg* für f . Die Residualkapazität von W ist

$$\Delta_W := \min_{e \in W} c_f(e),$$

Satz. (vom flussvergrößernden Weg).
Ein zulässiger s - t -Fluss f in G ist maximal \Leftrightarrow
es gibt keinen flussvergrößernden Weg in G_f .



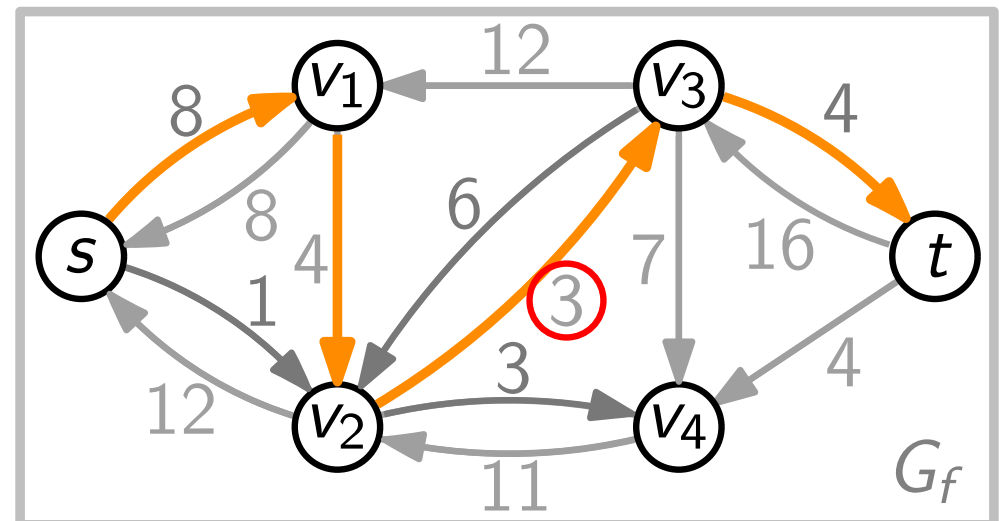
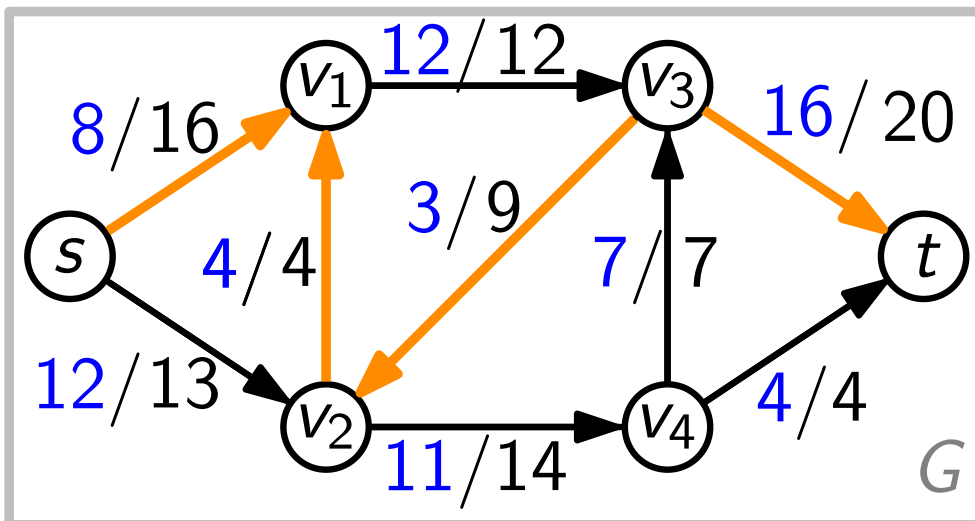
Flussvergrößernde Wege

Def. Ein s - t -Weg W in G_f heißt *flussvergrößernder Weg* für f . Die Residualkapazität von W ist

$$\Delta_W := \min_{e \in W} c_f(e),$$

Satz. (vom flussvergrößernden Weg).
Ein zulässiger s - t -Fluss f in G ist maximal \Leftrightarrow
es gibt keinen flussvergrößernden Weg in G_f .

(Wir beweisen ihn nicht; er folgt direkt aus dem nächsten Satz.)



Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.
Dann sind folgende Bedingungen äquivalent:

Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Dann sind folgende Bedingungen äquivalent:

1. f ist ein maximaler Fluss in G .
2. G_f enthält keine augmentierenden Wege.
3. Es gibt einen s - t -Schnitt (S, T) mit $|f| = c(S)$.

Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Dann sind folgende Bedingungen äquivalent:

1. f ist ein maximaler Fluss in G .
2. G_f enthält keine augmentierenden Wege.
3. Es gibt einen s - t -Schnitt (S, T) mit $|f| = c(S)$.

Kurz:

$$\max_{f \text{ zulässiger } s\text{-}t\text{-Fluss}} |f| = \min_{(S, T) \text{ } s\text{-}t\text{-Schnitt}} c(S)$$

Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Dann sind folgende Bedingungen äquivalent:

1. f ist ein maximaler Fluss in G .
2. G_f enthält keine augmentierenden Wege.
3. Es gibt einen s - t -Schnitt (S, T) mit $|f| = c(S)$.

Kurz:

$$\max_{f \text{ zulässiger } s\text{-}t\text{-Fluss}} |f| = \min_{(S, T) \text{ } s\text{-}t\text{-Schnitt}} c(S)$$

Beweis. (1) \Rightarrow (2):

Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Dann sind folgende Bedingungen äquivalent:

1. f ist ein maximaler Fluss in G .
2. G_f enthält keine augmentierenden Wege.
3. Es gibt einen s - t -Schnitt (S, T) mit $|f| = c(S)$.

Kurz:

$$\max_{f \text{ zulässiger } s\text{-}t\text{-Fluss}} |f| = \min_{(S, T) \text{ } s\text{-}t\text{-Schnitt}} c(S)$$

Beweis.

(1) \Rightarrow (2):

Ang. G_f enthält augmentierenden Weg.

Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Dann sind folgende Bedingungen äquivalent:

1. f ist ein maximaler Fluss in G .
2. G_f enthält keine augmentierenden Wege.
3. Es gibt einen s - t -Schnitt (S, T) mit $|f| = c(S)$.

Kurz:

$$\max_{f \text{ zulässiger } s\text{-}t\text{-Fluss}} |f| = \min_{(S, T) \text{ } s\text{-}t\text{-Schnitt}} c(S)$$

Beweis.

(1) \Rightarrow (2):

Ang. G_f enthält augmentierenden Weg.

Aber dann könnte f erhöht werden.

Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Dann sind folgende Bedingungen äquivalent:

1. f ist ein maximaler Fluss in G .
2. G_f enthält keine augmentierenden Wege.
3. Es gibt einen s - t -Schnitt (S, T) mit $|f| = c(S)$.

Kurz:

$$\max_{f \text{ zulässiger } s\text{-}t\text{-Fluss}} |f| = \min_{(S, T) \text{ } s\text{-}t\text{-Schnitt}} c(S)$$

Beweis.

(1) \Rightarrow (2):

Ang. G_f enthält augmentierenden Weg.

Aber dann könnte f erhöht werden.

Widerspruch zu $|f|$ maximal.

Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Dann sind folgende Bedingungen äquivalent:

1. f ist ein maximaler Fluss in G .
2. G_f enthält keine augmentierenden Wege.
3. Es gibt einen s - t -Schnitt (S, T) mit $|f| = c(S)$.

Kurz:

$$\max_{f \text{ zulässiger } s\text{-}t\text{-Fluss}} |f| = \min_{(S, T) \text{ } s\text{-}t\text{-Schnitt}} c(S)$$

Beweis.

(1) \Rightarrow (2):

Ang. G_f enthält augmentierenden Weg.

Aber dann könnte f erhöht werden.

Widerspruch zu $|f|$ maximal.

(3) \Rightarrow (1):

Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Dann sind folgende Bedingungen äquivalent:

1. f ist ein maximaler Fluss in G .
2. G_f enthält keine augmentierenden Wege.
3. Es gibt einen s - t -Schnitt (S, T) mit $|f| = c(S)$.

Kurz:

$$\max_{f \text{ zulässiger } s\text{-}t\text{-Fluss}} |f| = \min_{(S, T) \text{ } s\text{-}t\text{-Schnitt}} c(S)$$

Beweis.

(1) \Rightarrow (2):

Ang. G_f enthält augmentierenden Weg.

Aber dann könnte f erhöht werden.

Widerspruch zu $|f|$ maximal.

(3) \Rightarrow (1): Korollar \Rightarrow

Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Dann sind folgende Bedingungen äquivalent:

1. f ist ein maximaler Fluss in G .
2. G_f enthält keine augmentierenden Wege.
3. Es gibt einen s - t -Schnitt (S, T) mit $|f| = c(S)$.

Kurz:

$$\max_{f \text{ zulässiger } s\text{-}t\text{-Fluss}} |f| = \min_{(S, T) \text{ } s\text{-}t\text{-Schnitt}} c(S)$$

Beweis.

(1) \Rightarrow (2):

Ang. G_f enthält augmentierenden Weg.

Aber dann könnte f erhöht werden.

Widerspruch zu $|f|$ maximal.

(3) \Rightarrow (1): Korollar \Rightarrow

Erinnerung:

Korollar.

$$|f| = c(S)$$



f maximal

Das Max-Flow-Min-Cut-Theorem

Satz.

Sei f ein zulässiger s - t -Fluss in einem gerichteten Graphen G mit Kapazitäten $c: E \rightarrow \mathbb{R}_{>0}$.

Dann sind folgende Bedingungen äquivalent:

1. f ist ein maximaler Fluss in G .
2. G_f enthält keine augmentierenden Wege.
3. Es gibt einen s - t -Schnitt (S, T) mit $|f| = c(S)$.

Kurz:

$$\max_{f \text{ zulässiger } s\text{-}t\text{-Fluss}} |f| = \min_{(S, T) \text{ } s\text{-}t\text{-Schnitt}} c(S)$$

Beweis.

(1) \Rightarrow (2):

Ang. G_f enthält augmentierenden Weg.

Aber dann könnte f erhöht werden.

Widerspruch zu $|f|$ maximal.

(3) \Rightarrow (1): Korollar $\Rightarrow |f|$ maximal.

Erinnerung:

Korollar.

$$|f| = c(S)$$



f maximal

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3):

Zu zeigen: 2. G_f enthält keine augmentierenden Wege.
↓
3. Es gibt einen Schnitt (S, T) mit $|f| = c(S)$.

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

\Rightarrow $\left. \begin{array}{l} \\ \end{array} \right\}$ ist s - t -Schnitt.

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

Nun sei $e = uv \in \text{Rein}(S)$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

\Rightarrow

Nun sei $e = uv \in \text{Rein}(S)$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$

Nun sei $e = uv \in \text{Rein}(S)$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

Nun sei $e = uv \in \text{Rein}(S)$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow}$

Nun sei $e = uv \in \text{Rein}(S)$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S)$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S) \Rightarrow f(e) = 0$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S) \Rightarrow f(e) = 0$

$\stackrel{\Sigma}{\Rightarrow} f(\text{Rein}(S)) = 0$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S) \Rightarrow f(e) = 0$

$\stackrel{\Sigma}{\Rightarrow} f(\text{Rein}(S)) = 0$

Also: $c(S) \stackrel{\text{Def.}}{=} f(\text{Raus}(S)) + f(\text{Rein}(S)) = c(\text{Raus}(S))$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S) \Rightarrow f(e) = 0$

$\stackrel{\Sigma}{\Rightarrow} f(\text{Rein}(S)) = 0$

Also: $c(S) \stackrel{\text{Def.}}{=} c(\text{Raus}(S)) =$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S) \Rightarrow f(e) = 0$

$\stackrel{\Sigma}{\Rightarrow} f(\text{Rein}(S)) = 0$

Also: $c(S) \stackrel{\text{Def.}}{=} c(\text{Raus}(S)) =$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S) \Rightarrow f(e) = 0$

$\stackrel{\Sigma}{\Rightarrow} f(\text{Rein}(S)) = 0$

Also: $c(S) \stackrel{\text{Def.}}{=} c(\text{Raus}(S)) = f(\text{Raus}(S))$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S) \Rightarrow f(e) = 0$

$\stackrel{\Sigma}{\Rightarrow} f(\text{Rein}(S)) = 0$

Also: $c(S) \stackrel{\text{Def.}}{=} c(\text{Raus}(S)) = f(\text{Raus}(S))$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S) \Rightarrow f(e) = 0$

$\stackrel{\Sigma}{\Rightarrow} f(\text{Rein}(S)) = 0$

Also: $c(S) \stackrel{\text{Def.}}{=} c(\text{Raus}(S)) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $=$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S) \Rightarrow f(e) = 0$

$\stackrel{\Sigma}{\Rightarrow} f(\text{Rein}(S)) = 0$

Also: $c(S) \stackrel{\text{Def.}}{=} c(\text{Raus}(S)) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $= \text{Nettoabfluss}_f(S) \stackrel{\text{Lem. 2}}{=}$

Zu zeigen:

2. G_f enthält keine augmentierenden Wege.



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

(2) \Rightarrow (3): Es gibt keinen augmentierenden Weg in G_f .

\Rightarrow In G_f ist t von s aus nicht erreichbar.

$\Rightarrow \left. \begin{array}{l} S = \{v \mid v \text{ von } s \text{ erreichbar}\} \\ T = \{v \mid v \text{ von } s \text{ nicht erreichbar}\} \end{array} \right\}$ ist s - t -Schnitt.

Sei $e = uv \in \text{Raus}(S)$.

$\Rightarrow f(e) = c(e)$, sonst wäre v von s in G_f erreichbar.

$\stackrel{\Sigma}{\Rightarrow} f(\text{Raus}(S)) = c(\text{Raus}(S))$

Nun sei $e = uv \in \text{Rein}(S) \Rightarrow f(e) = 0$

$\stackrel{\Sigma}{\Rightarrow} f(\text{Rein}(S)) = 0$

Also: $c(S) \stackrel{\text{Def.}}{=} c(\text{Raus}(S)) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $= \text{Nettoabfluss}_f(S) \stackrel{\text{Lem. 2}}{=} |f|$ □

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

return f

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

} Initialisierung mit Null-Fluss

return f

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

} Initialisierung mit Null-Fluss

return f

} Rückgabe eines max. Flusses

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

} Initialisierung mit Null-Fluss

while G_f enthält s - t -Weg W **do**

└
return f

} Rückgabe eines max. Flusses

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

} Initialisierung mit Null-Fluss

while G_f enthält s - t -Weg W **do**

Schreiben Sie Ihren eigenen Code, der bei Abbruch der **while**-Schleife in f einen maximalen Fluss liefert.

return f

} Rückgabe eines max. Flusses

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

} Initialisierung mit Null-Fluss

while G_f enthält s - t -Weg W **do**

└ $\Delta_W = \min_{uv \in W} c_f(uv)$

} Residualkapazität von W

Schreiben Sie Ihren eigenen Code, der bei Abbruch der **while**-Schleife in f einen maximalen Fluss liefert.

return f

} Rückgabe eines max. Flusses

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

} Initialisierung mit Null-Fluss

while G_f enthält s - t -Weg W **do**

└ $\Delta_W = \min_{uv \in W} c_f(uv)$

} Residualkapazität von W

foreach $uv \in W$ **do**

└ **if** $uv \in E$ **then**

└ $f_{uv} = f_{uv} + \Delta_W$

else

└ $f_{vu} = f_{vu} - \Delta_W$

return f

} Rückgabe eines max. Flusses

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

} Initialisierung mit Null-Fluss

while G_f enthält s - t -Weg W **do**

└ $\Delta_W = \min_{uv \in W} c_f(uv)$

} Residualkapazität von W

└ **foreach** $uv \in W$ **do**

└ **if** $uv \in E$ **then**

└ | $f_{uv} = f_{uv} + \Delta_W$

└ **else**

└ | $f_{vu} = f_{vu} - \Delta_W$

} Flussvergrößerung entlang W

return f

} Rückgabe eines max. Flusses

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

} Initialisierung mit Null-Fluss

while G_f enthält s - t -Weg W **do**

└ $\Delta_W = \min_{uv \in W} c_f(uv)$

} Residualkapazität von W

└ **foreach** $uv \in W$ **do**

└ **if** $uv \in E$ **then**

└ | $f_{uv} = f_{uv} + \Delta_W$

└ **else**

└ | $f_{vu} = f_{vu} - \Delta_W$

} Flussvergrößerung entlang W

return f

} Rückgabe eines max. Flusses

Korrektheit?

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

} Initialisierung mit Null-Fluss

while G_f enthält s - t -Weg W **do**

└ $\Delta_W = \min_{uv \in W} c_f(uv)$

} Residualkapazität von W

└ **foreach** $uv \in W$ **do**

└ **if** $uv \in E$ **then**

└ | $f_{uv} = f_{uv} + \Delta_W$

└ **else**

└ | $f_{vu} = f_{vu} - \Delta_W$

} Flussvergrößerung entlang W

return f

} Rückgabe eines max. Flusses

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌ ┌ **if** $uv \in E$ **then**

┌ ┌ ┌ $f_{uv} = f_{uv} + \Delta_W$

┌ ┌ **else**

┌ ┌ ┌ $f_{vu} = f_{vu} - \Delta_W$

return f

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit?

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌└ **if** $uv \in E$ **then**

┌└└ $f_{uv} = f_{uv} + \Delta_W$

┌└ **else**

┌└└ $f_{vu} = f_{vu} - \Delta_W$

return f

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit?

1. $c: E \rightarrow \mathbb{N}$:

2. $\mathbb{Q}_{>0}$:

3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

┌ **foreach** $uv \in W$ **do**

┌ **if** $uv \in E$ **then**

┌ | $f_{uv} = f_{uv} + \Delta_W$

┌ **else**

┌ | $f_{vu} = f_{vu} - \Delta_W$

return f

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit? 1. $c: E \rightarrow \mathbb{N}$:

2. $\mathbb{Q}_{>0}$:

3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌└ **if** $uv \in E$ **then**

┌└└ $f_{uv} = f_{uv} + \Delta_W$

┌└ **else**

┌└└ $f_{vu} = f_{vu} - \Delta_W$

return f

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit? 1. $c: E \rightarrow \mathbb{N}$:

2. $\mathbb{Q}_{>0}$:

3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌ **if** $uv \in E$ **then**

└ $f_{uv} = f_{uv} + \Delta_W$

else

└ $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit? 1. $c: E \rightarrow \mathbb{N}$:

2. $\mathbb{Q}_{>0}$:

3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌└ **if** $uv \in E$ **then**

┌ | $f_{uv} = f_{uv} + \Delta_W$

┌└ **else**

┌ | $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

- Breitensuche
- Tiefensuche

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit? 1. $c: E \rightarrow \mathbb{N}$:

2. $\mathbb{Q}_{>0}$:
3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌└ **if** $uv \in E$ **then**

┌└└ $f_{uv} = f_{uv} + \Delta_W$

┌└ **else**

┌└└ $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

– Breitensuche } $O(E)$ Zeit
– Tiefensuche }

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit? 1. $c: E \rightarrow \mathbb{N}$:

2. $\mathbb{Q}_{>0}$:

3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

└ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

└└ **if** $uv \in E$ **then**

└└└ $f_{uv} = f_{uv} + \Delta_W$

└└ **else**

└└└ $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

– Breitensuche } $O(E)$ Zeit
– Tiefensuche }

Anz. Schleifendurchläufe

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit? 1. $c: E \rightarrow \mathbb{N}$:

2. $\mathbb{Q}_{>0}$:

3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌└ **if** $uv \in E$ **then**

┌ | $f_{uv} = f_{uv} + \Delta_W$

┌└ **else**

┌ | $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

– Breitensuche } $O(E)$ Zeit
– Tiefensuche }

Anz. Schleifendurchläufe

– in jedem Durchlauf wird f um ≥ 1 vergrößert

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit? 1. $c: E \rightarrow \mathbb{N}$:

2. $\mathbb{Q}_{>0}$:

3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌└ **if** $uv \in E$ **then**

┌ | $f_{uv} = f_{uv} + \Delta_W$

┌└ **else**

┌ | $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

– Breitensuche } $O(E)$ Zeit
– Tiefensuche }

Anz. Schleifendurchläufe

– in jedem Durchlauf wird f um ≥ 1 vergrößert
– max. $|f^*|$ Durchläufe, wobei f^* ein max. Fluss

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit? 1. $c: E \rightarrow \mathbb{N}$:

2. $\mathbb{Q}_{>0}$:

3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌└ **if** $uv \in E$ **then**

┌ | $f_{uv} = f_{uv} + \Delta_W$

┌└ **else**

┌ | $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

– Breitensuche } $O(E)$ Zeit
– Tiefensuche }

Anz. Schleifendurchläufe

– in jedem Durchlauf wird f um ≥ 1 vergrößert
– max. $|f^*|$ Durchläufe, wobei f^* ein max. Fluss

Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

Laufzeit?

1. $c: E \rightarrow \mathbb{N}$: $O(|f^*| \cdot E)$

2. $\mathbb{Q}_{>0}$:

3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌└ **if** $uv \in E$ **then**

┌ | $f_{uv} = f_{uv} + \Delta_W$

┌└ **else**

┌ | $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

– Breitensuche } $O(E)$ Zeit
– Tiefensuche }

Anz. Schleifendurchläufe

– in jedem Durchlauf wird f um ≥ 1 vergrößert
– max. $|f^*|$ Durchläufe, wobei f^* ein max. Fluss

Korrektheit?

Folgt aus Max-Flow-Min-Cut-Theorem

Laufzeit?

1. $c: E \rightarrow \mathbb{N}$: $O(|f^*| \cdot E)$

2. $\mathbb{Q}_{>0}$:

3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

foreach $uv \in W$ **do**

┌ **if** $uv \in E$ **then**

│ $f_{uv} = f_{uv} + \Delta_W$

else

└ $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

– Breitensuche } $O(E)$ Zeit
– Tiefensuche }

Anz. Schleifendurchläufe

– in jedem Durchlauf wird f um ≥ 1 vergrößert
– max. $|f^*|$ Durchläufe, wobei f^* ein max. Fluss

Korrektheit?

Folgt aus Max-Flow-Min-Cut-Theorem

Laufzeit?

1. $c: E \rightarrow \mathbb{N}$: $O(|f^*| \cdot E)$
2. $\mathbb{Q}_{>0}$: erweitern...
3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

foreach $uv \in W$ **do**

┌ **if** $uv \in E$ **then**

│ $f_{uv} = f_{uv} + \Delta_W$

else

└ $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

– Breitensuche } $O(E)$ Zeit
– Tiefensuche }

Anz. Schleifendurchläufe

– in jedem Durchlauf wird f um ≥ 1 vergrößert
– max. $|f^*|$ Durchläufe, wobei f^* ein max. Fluss

Korrektheit?

Folgt aus Max-Flow-Min-Cut-Theorem

Laufzeit?

1. $c: E \rightarrow \mathbb{N}$: $O(|f^*| \cdot E)$
2. $\mathbb{Q}_{>0}$: erweitern...
3. $\mathbb{R}_{>0}$:

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

foreach $uv \in W$ **do**

┌ **if** $uv \in E$ **then**

│ $f_{uv} = f_{uv} + \Delta_W$

else

└ $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

– Breitensuche } $O(E)$ Zeit
– Tiefensuche }

Anz. Schleifendurchläufe

– in jedem Durchlauf wird f um ≥ 1 vergrößert
– max. $|f^*|$ Durchläufe, wobei f^* ein max. Fluss

Korrektheit?

Folgt aus Max-Flow-Min-Cut-Theorem

Laufzeit?

1. $c: E \rightarrow \mathbb{N}$: $O(|f^*| \cdot E)$

2. $\mathbb{Q}_{>0}$: erweitern...

3. $\mathbb{R}_{>0}$: problematisch!

Der Algorithmus von Ford & Fulkerson

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

┌ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

┌└ **if** $uv \in E$ **then**

┌ | $f_{uv} = f_{uv} + \Delta_W$

┌└ **else**

┌ | $f_{vu} = f_{vu} - \Delta_W$

return f

Berechnung von s - t -Wegen

– Breitensuche } $O(E)$ Zeit
– Tiefensuche }

Anz. Schleifendurchläufe

– in jedem Durchlauf wird f um ≥ 1 vergrößert
– max. $|f^*|$ Durchläufe, wobei f^* ein max. Fluss

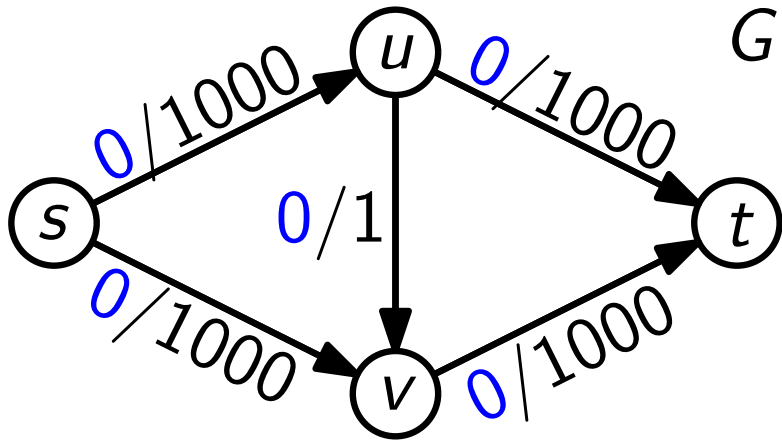
Korrektheit?

Folgt aus Max-Flow-
Min-Cut-Theorem

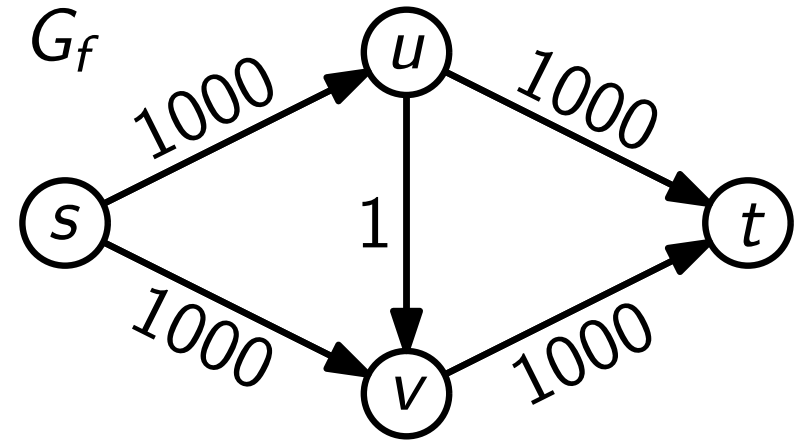
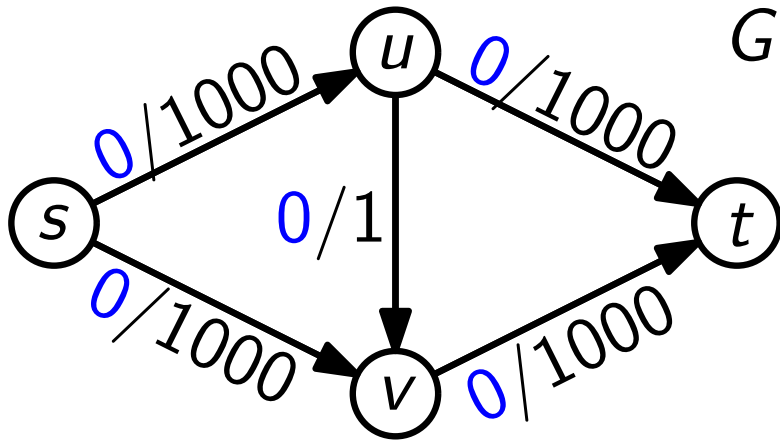
Laufzeit?

1. $c: E \rightarrow \mathbb{N}$: $O(|f^*| \cdot E)$
2. $\mathbb{Q}_{>0}$: erweitern...
3. $\mathbb{R}_{>0}$: problematisch!

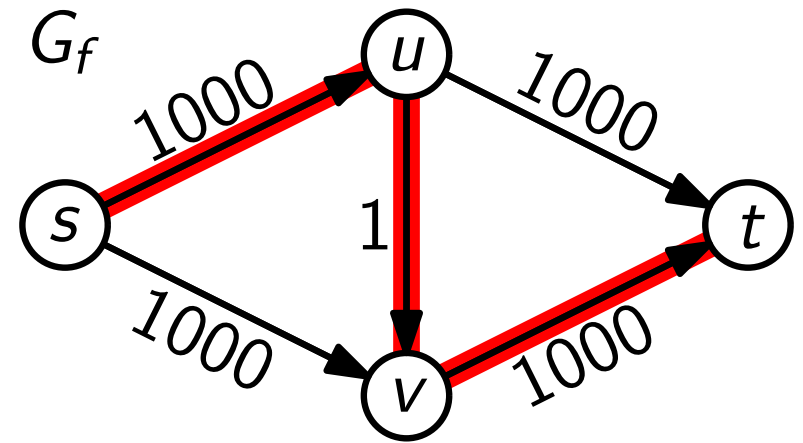
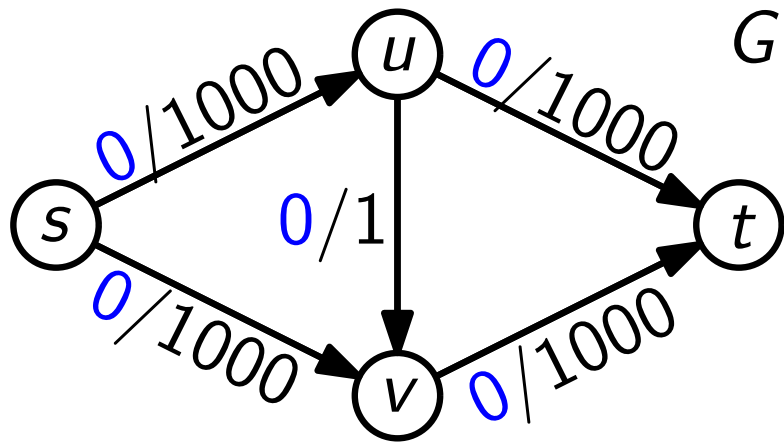
Beispiel



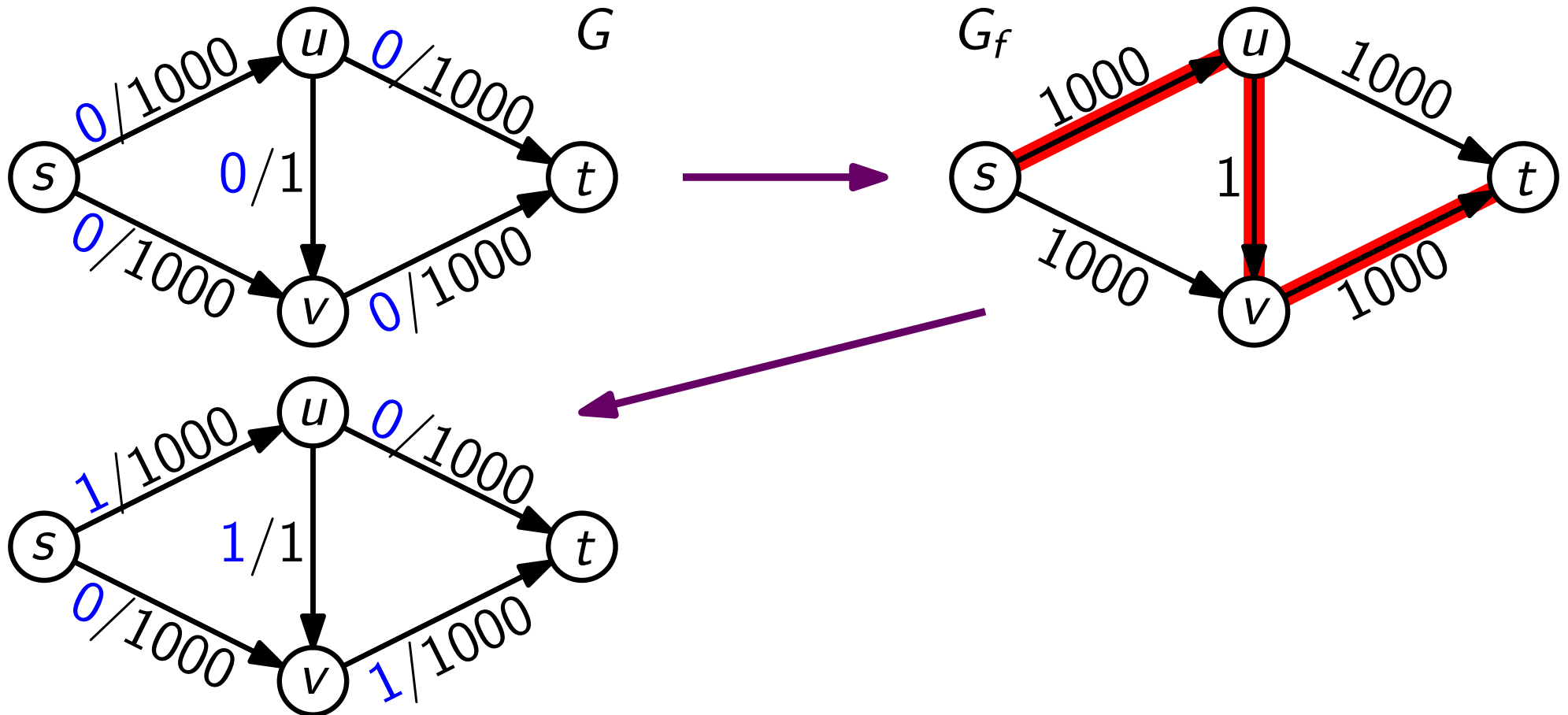
Beispiel



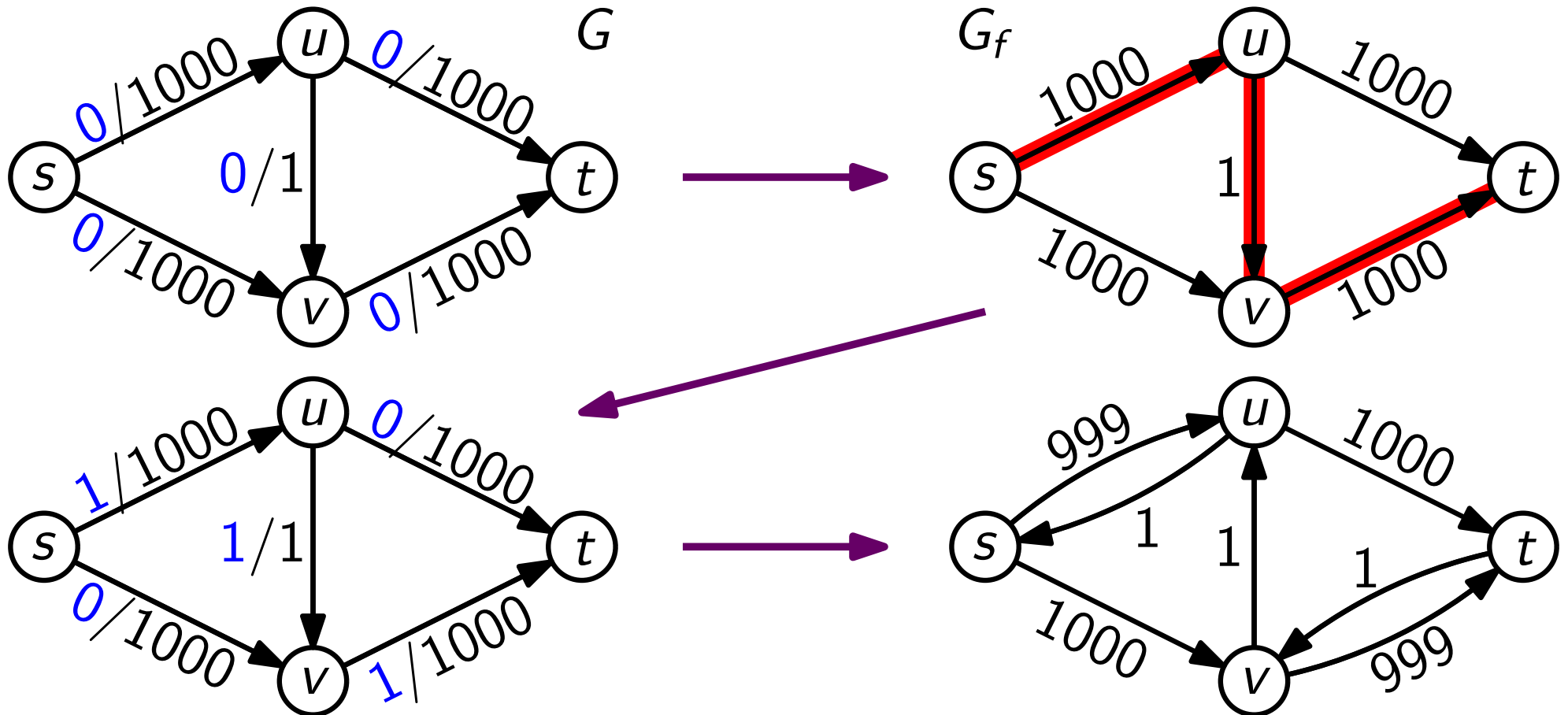
Beispiel



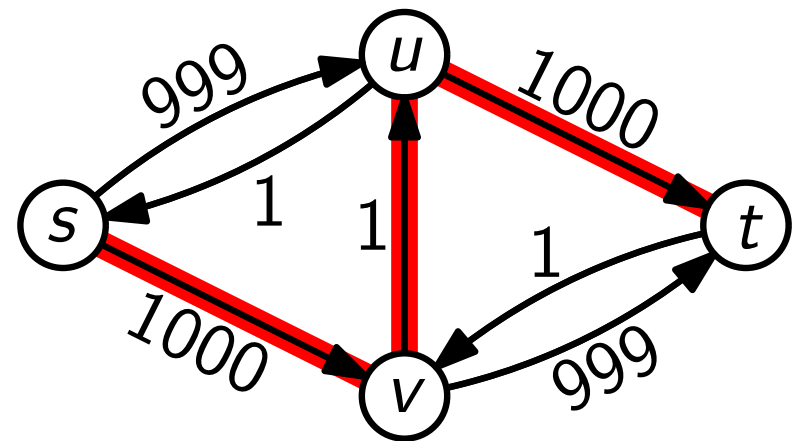
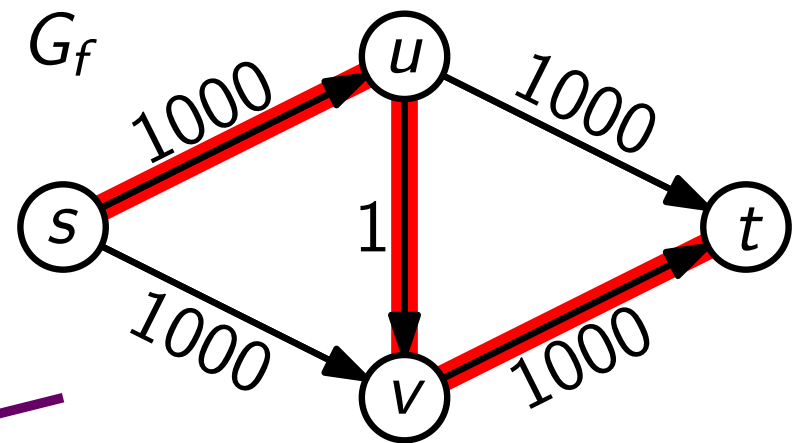
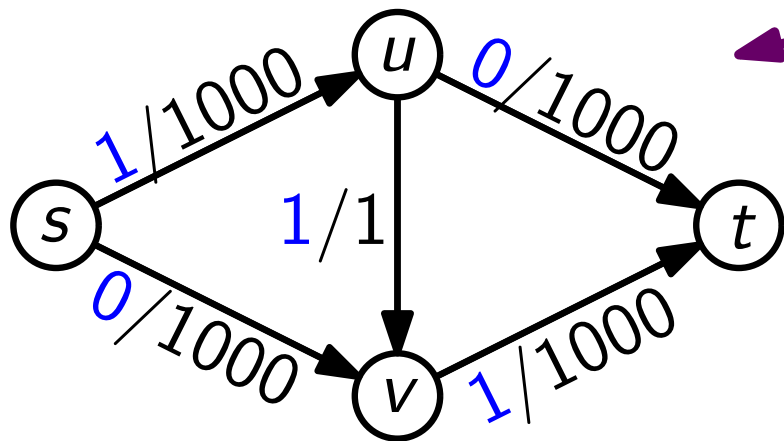
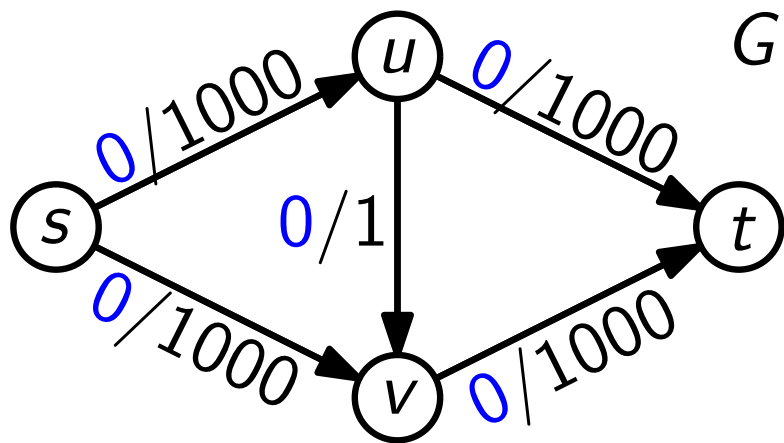
Beispiel



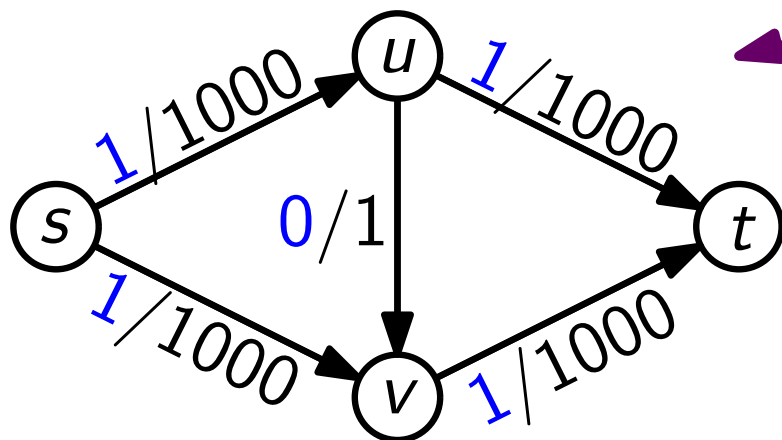
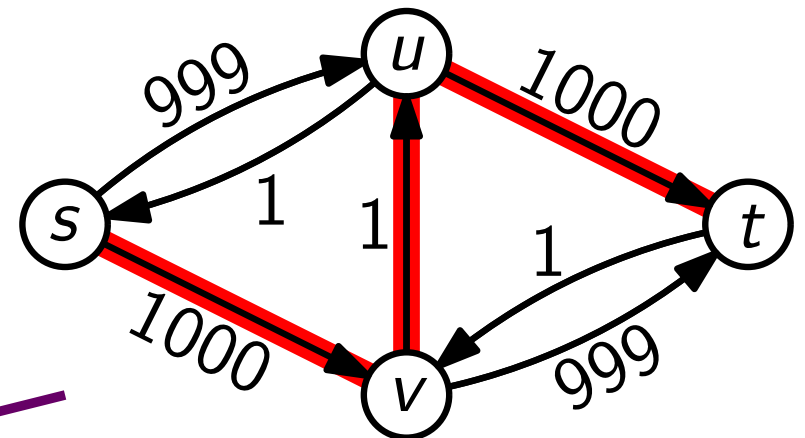
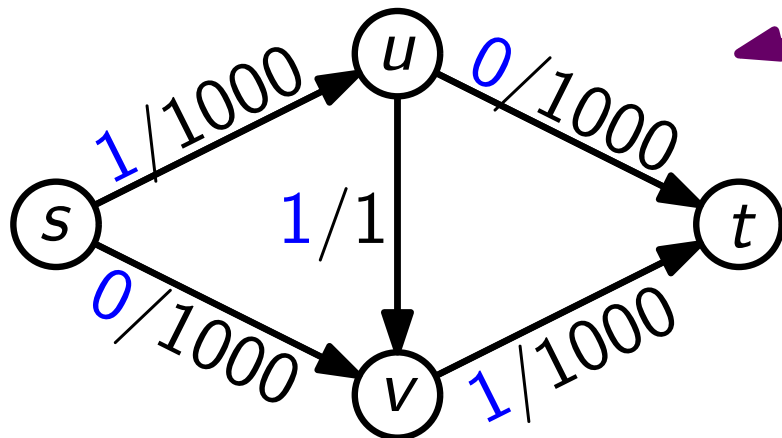
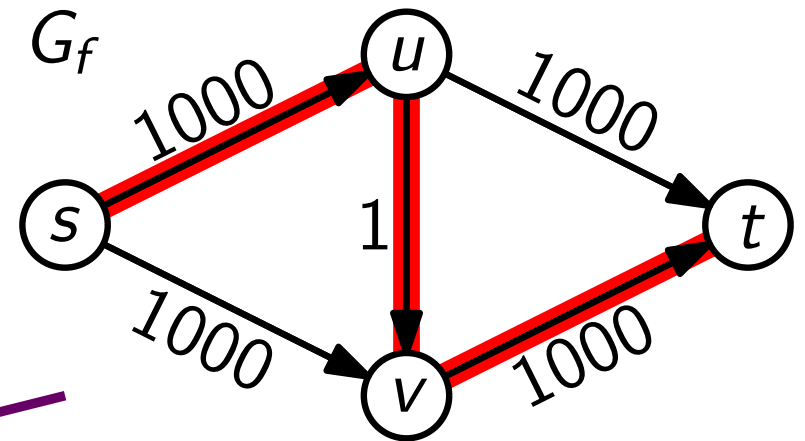
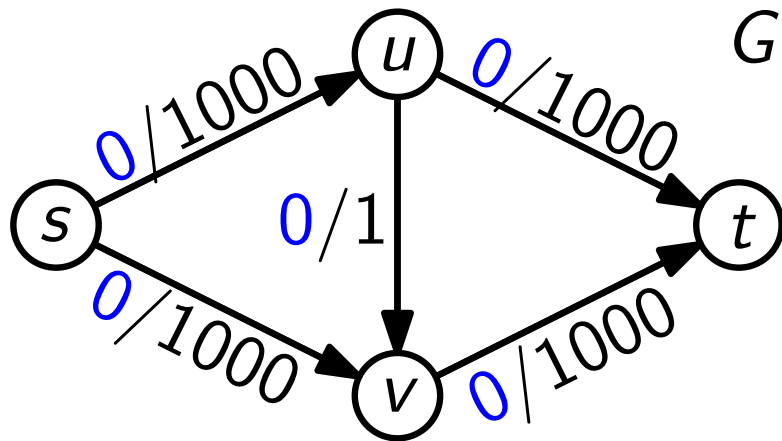
Beispiel



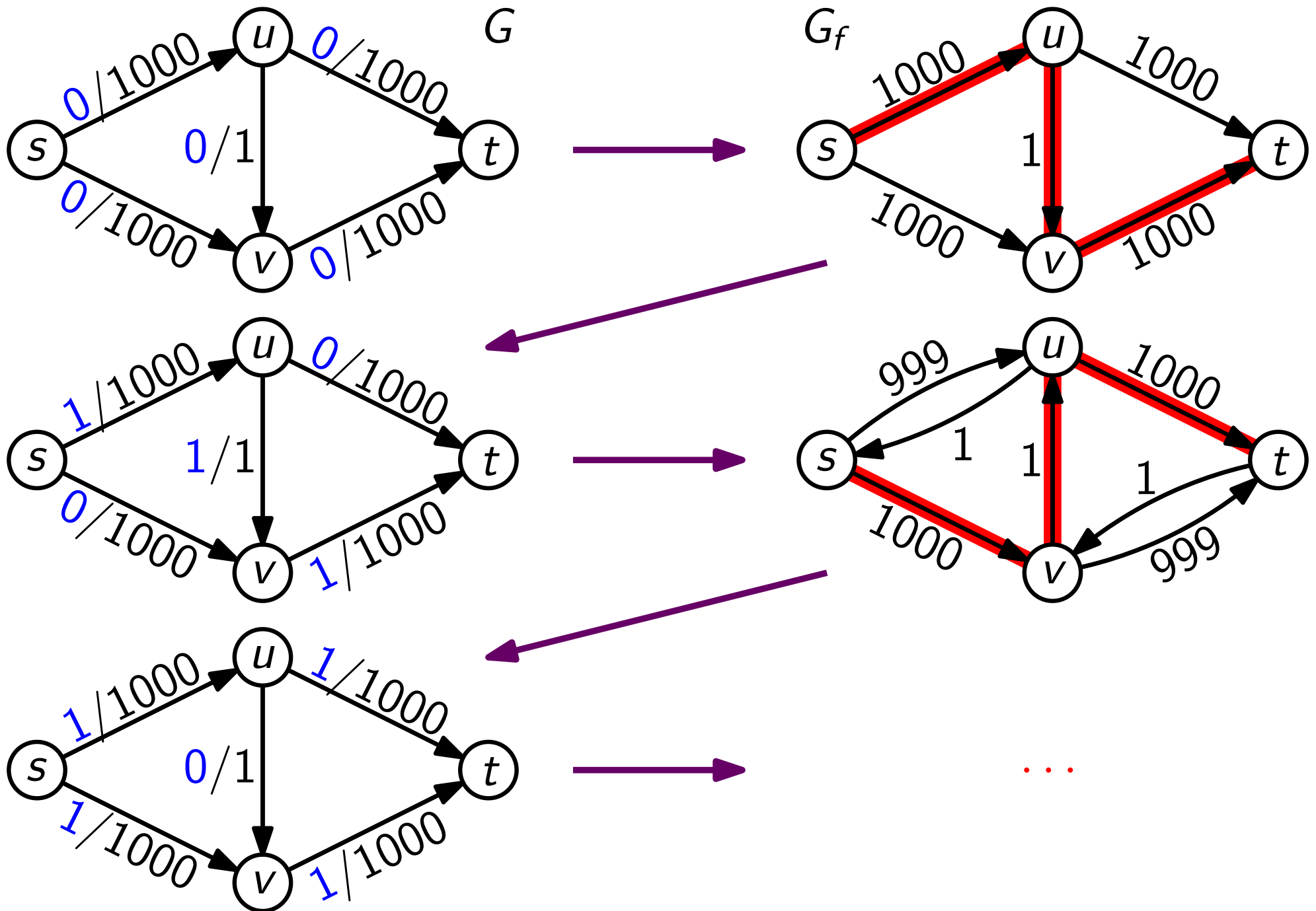
Beispiel



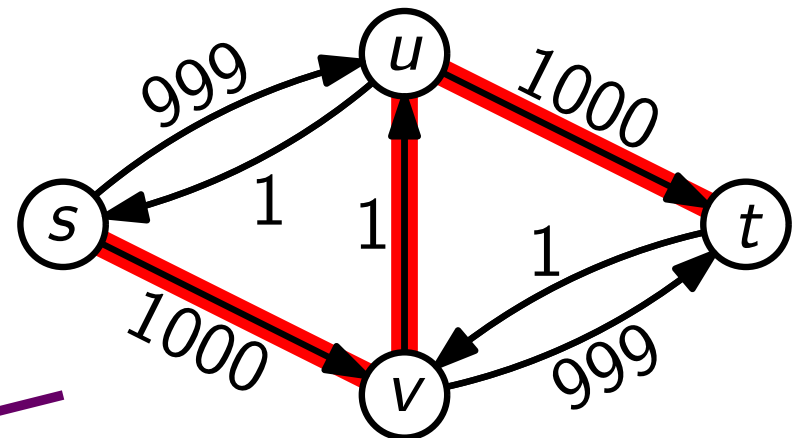
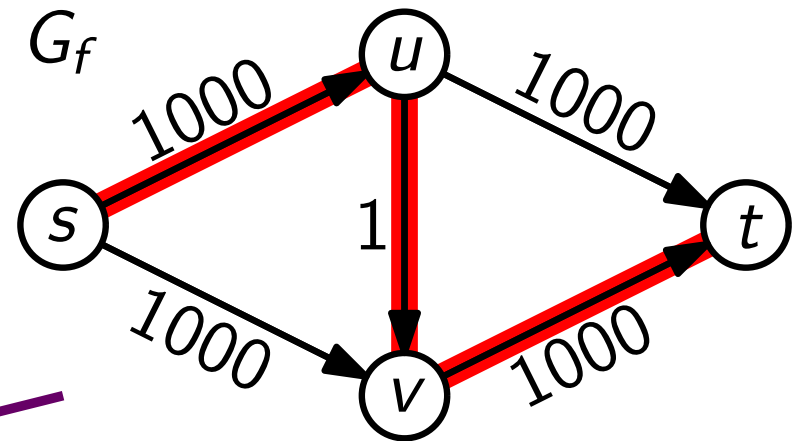
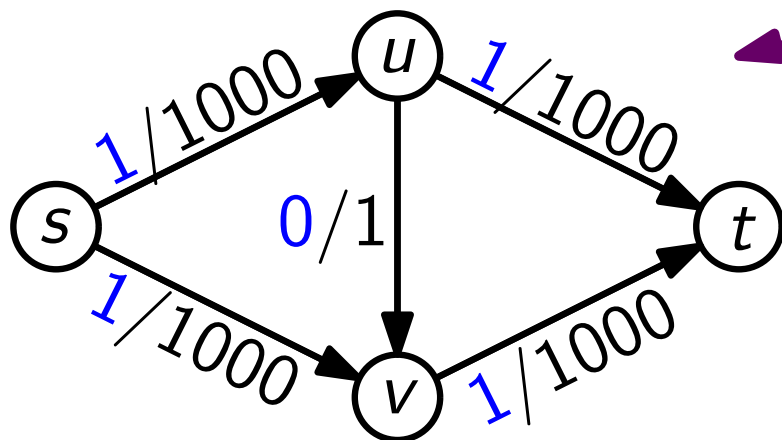
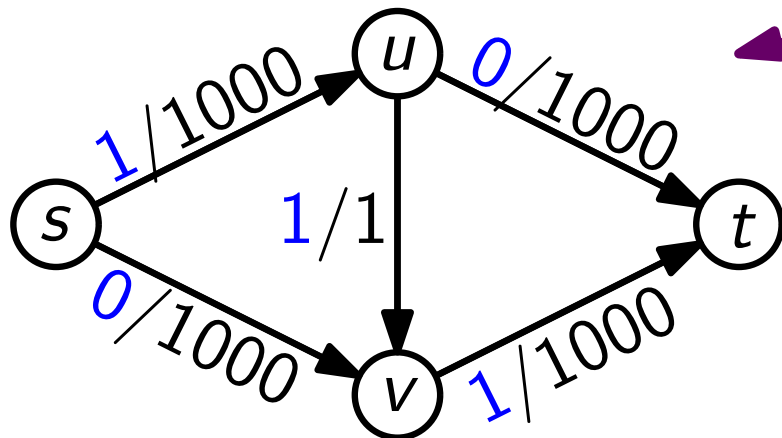
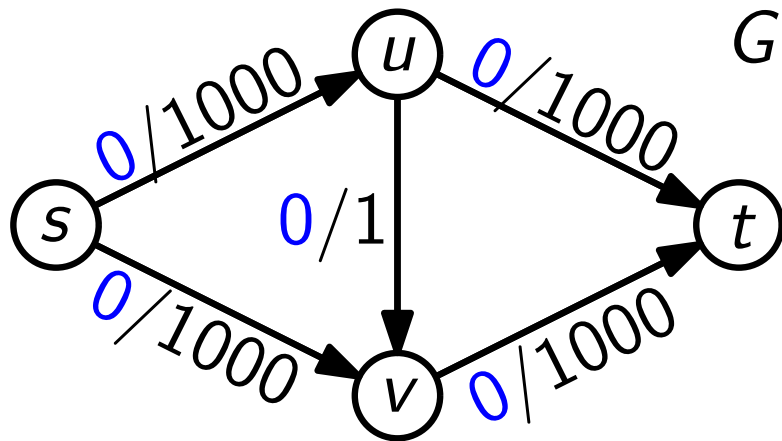
Beispiel



Beispiel



Beispiel



...

Laufzeit $\in \Omega(|f^*| \cdot E)$

Der Algorithmus von Edmonds & Karp

FordFulkerson(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg **do**

└ $W =$ s - t -Weg in G_f

└ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

└└ **if** $uv \in E$ **then**

└└└ $f_{uv} = f_{uv} + \Delta_W$

└└ **else**

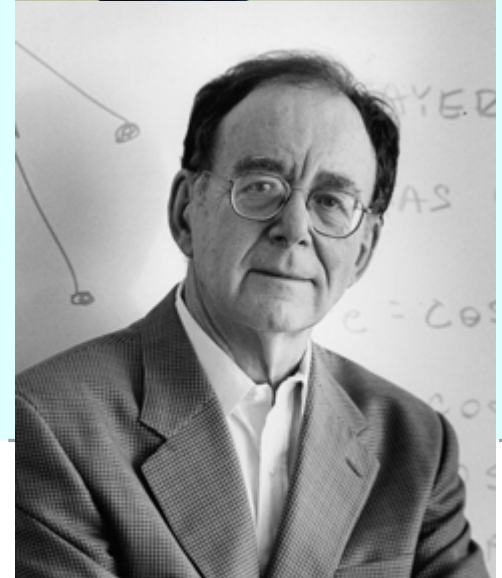
└└└ $f_{vu} = f_{vu} - \Delta_W$

return f

Jack R. Edmonds
*1934



Richard M. Karp
*1935 Boston, MA



Der Algorithmus von Edmonds & Karp

EdmondsKarp

~~FordFulkerson~~(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg **do**

└ $W =$ s - t -Weg in G_f

└ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

└└ **if** $uv \in E$ **then**

└└└ $f_{uv} = f_{uv} + \Delta_W$

└└ **else**

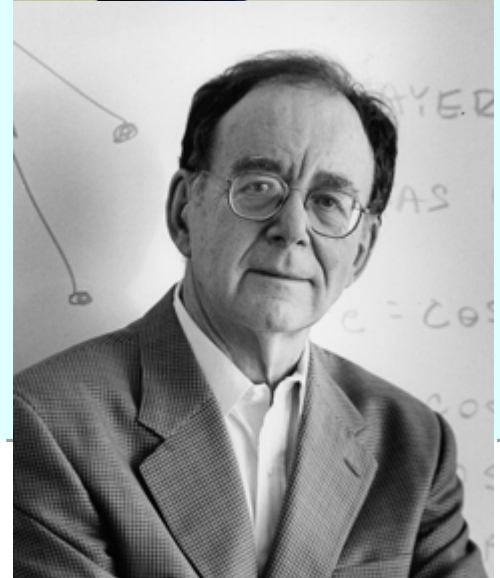
└└└ $f_{vu} = f_{vu} - \Delta_W$

return f

Jack R. Edmonds
*1934



Richard M. Karp
*1935 Boston, MA



Der Algorithmus von Edmonds & Karp

EdmondsKarp

~~FordFulkerson~~(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

foreach $uv \in E$ **do**

└ $f_{uv} = 0$

while G_f enthält s - t -Weg **do**

└ $W =$ kürzester s - t -Weg in G_f

└ $\Delta_W = \min_{uv \in W} c_f(uv)$

└ **foreach** $uv \in W$ **do**

└└ **if** $uv \in E$ **then**

└└└ $f_{uv} = f_{uv} + \Delta_W$

└└ **else**

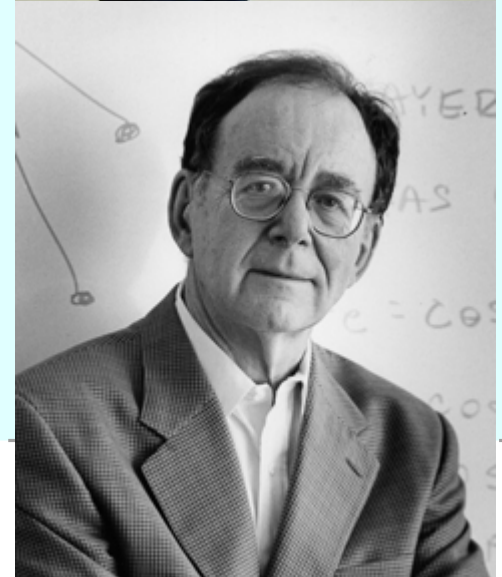
└└└ $f_{vu} = f_{vu} - \Delta_W$

return f

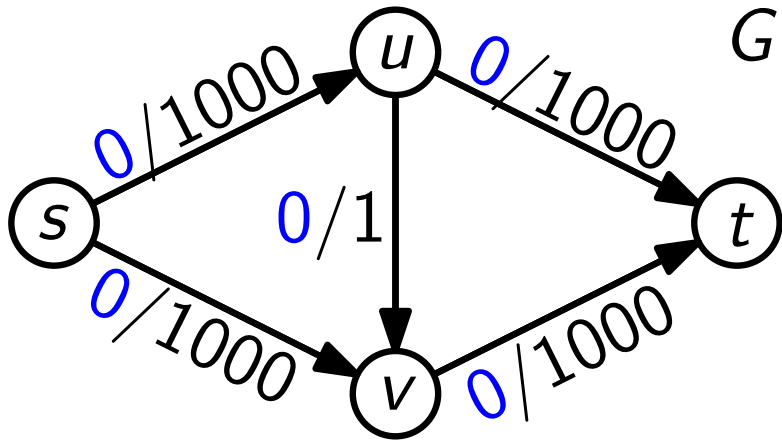
Jack R. Edmonds
*1934



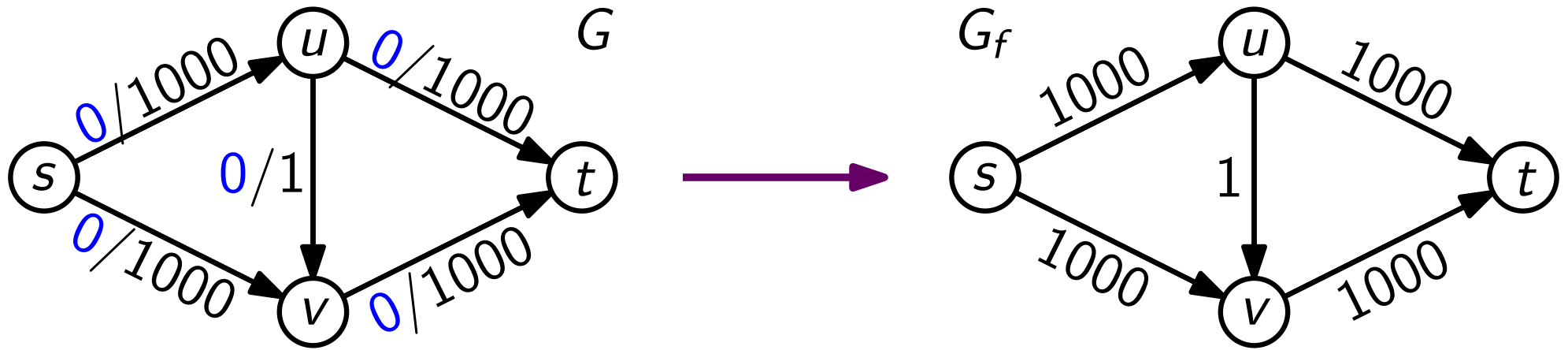
Richard M. Karp
*1935 Boston, MA



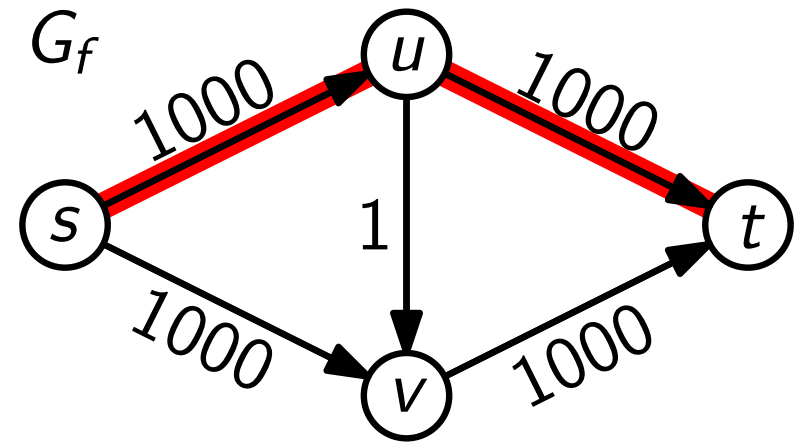
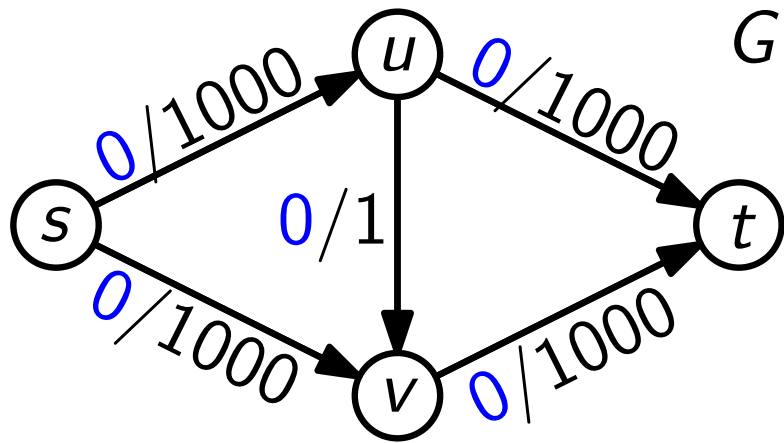
Beispiel



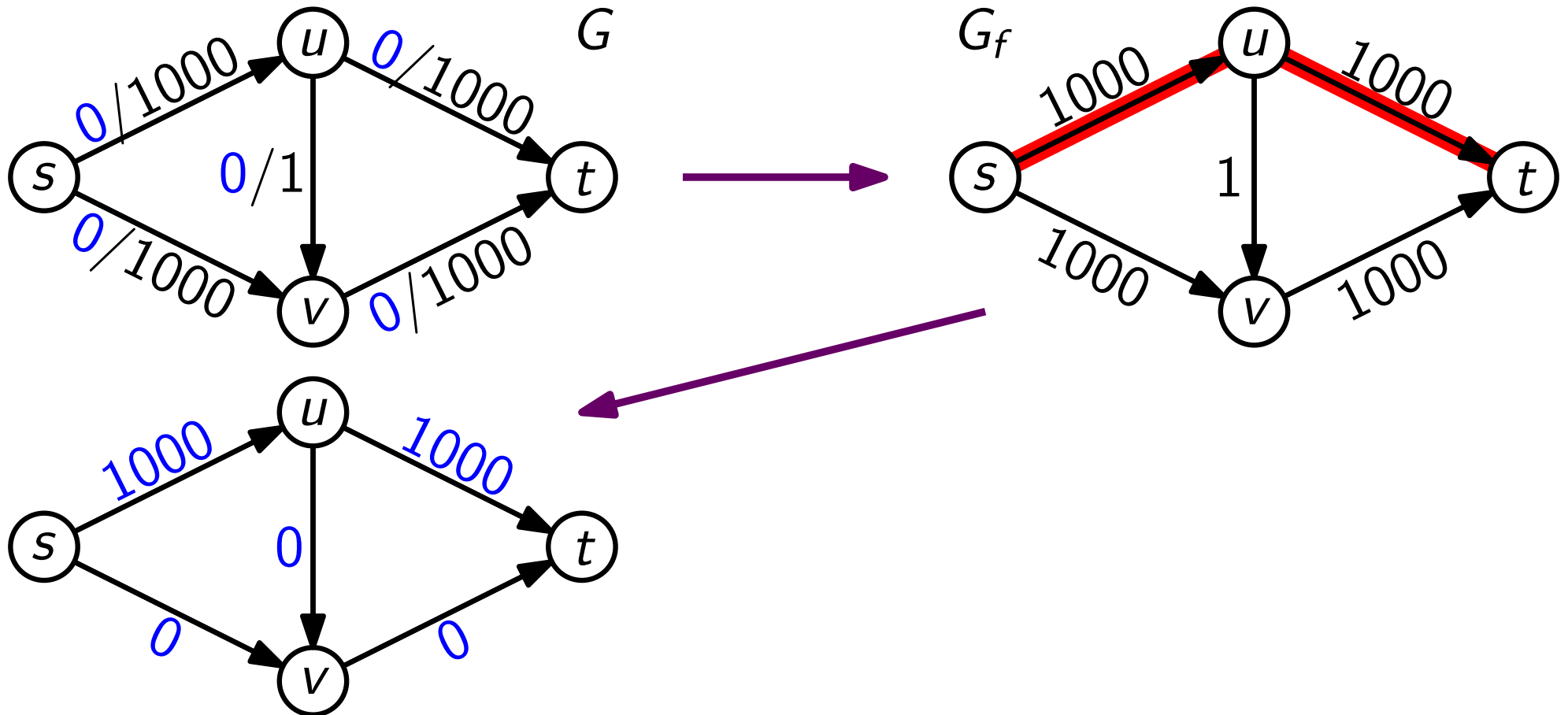
Beispiel



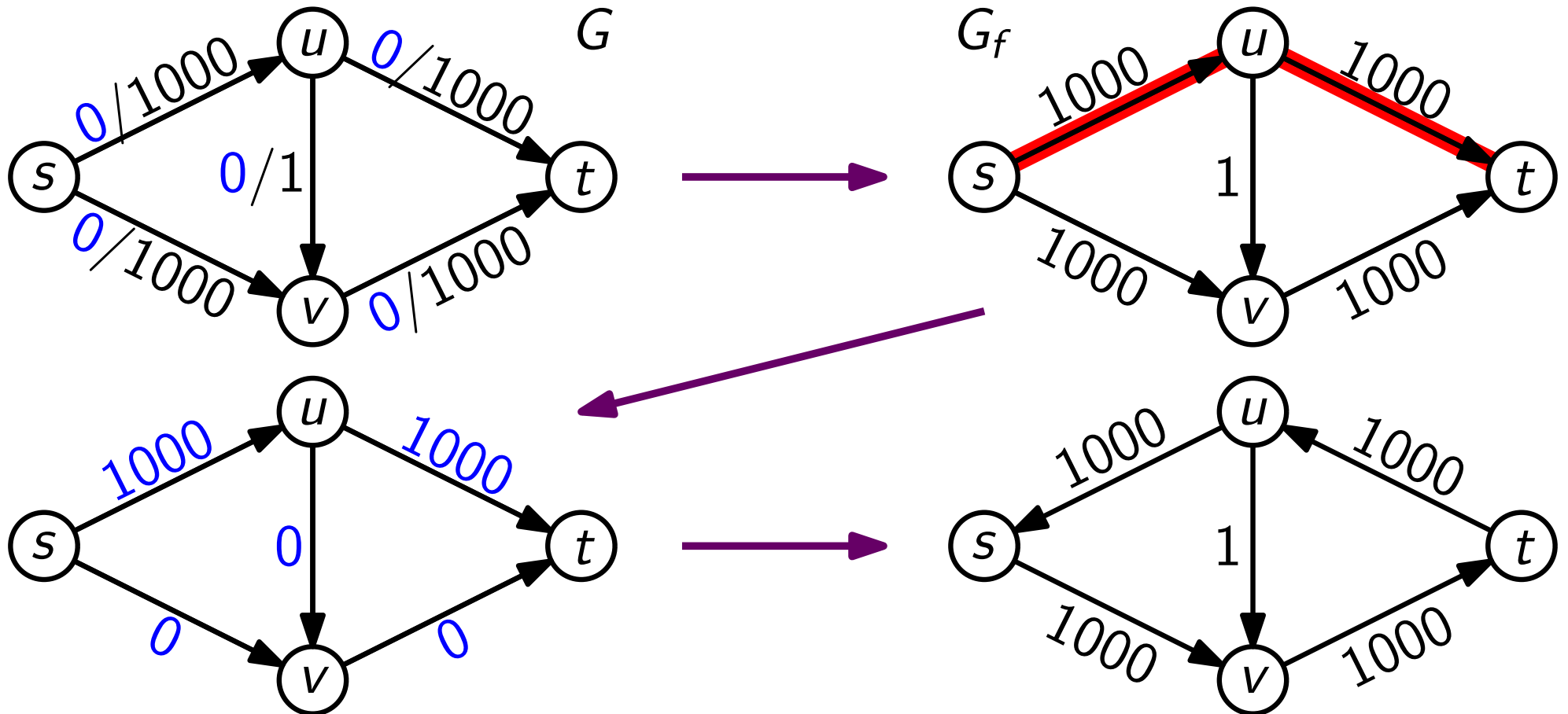
Beispiel



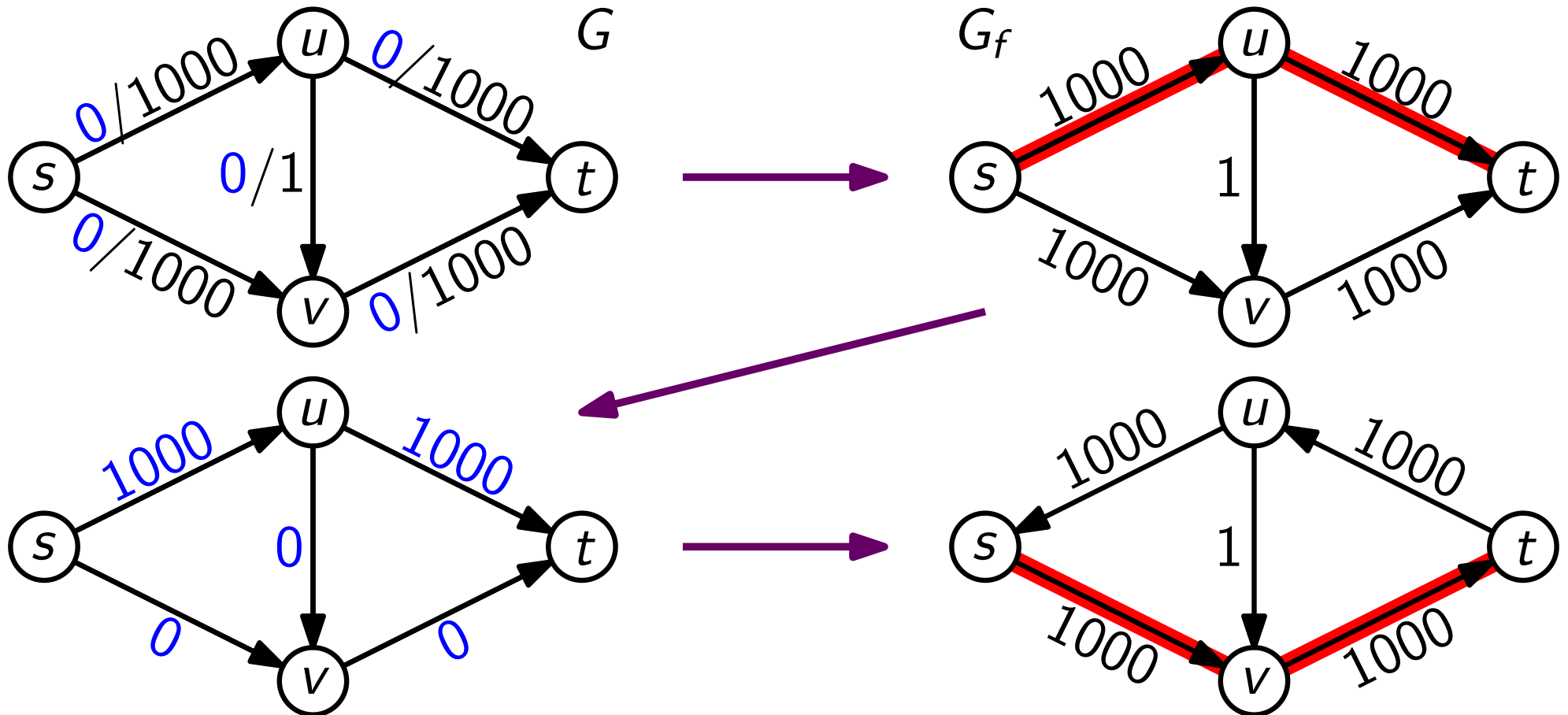
Beispiel



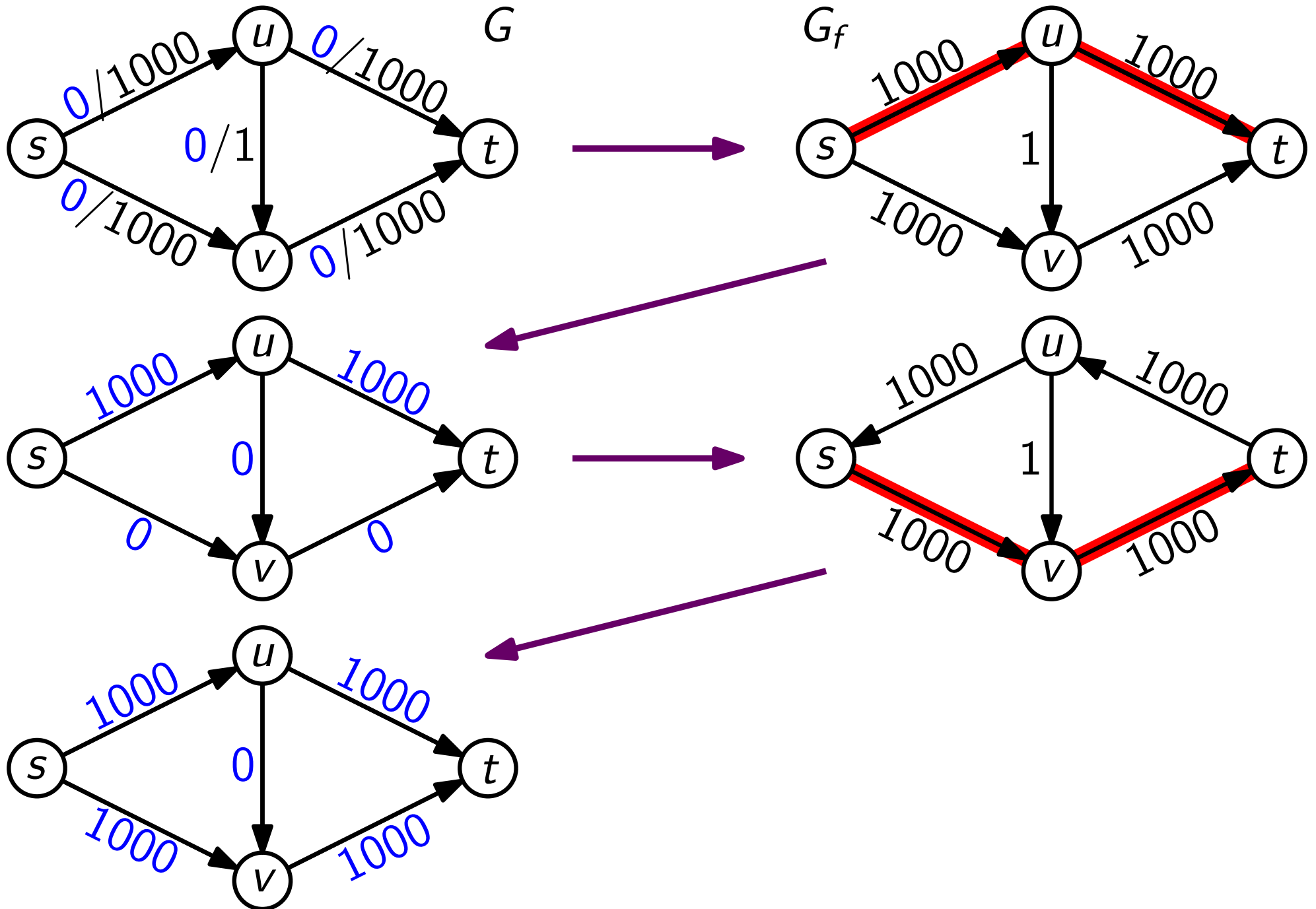
Beispiel



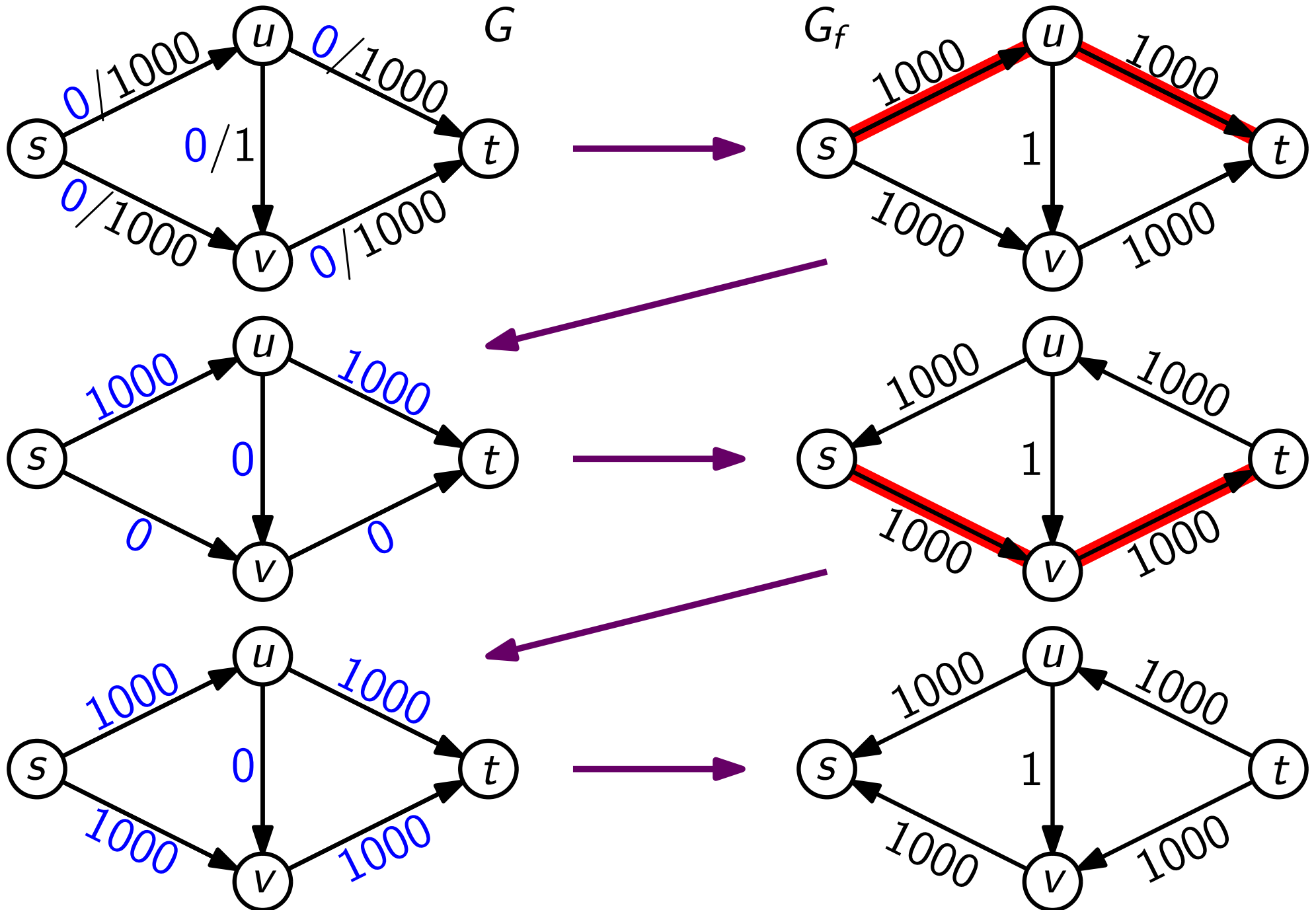
Beispiel



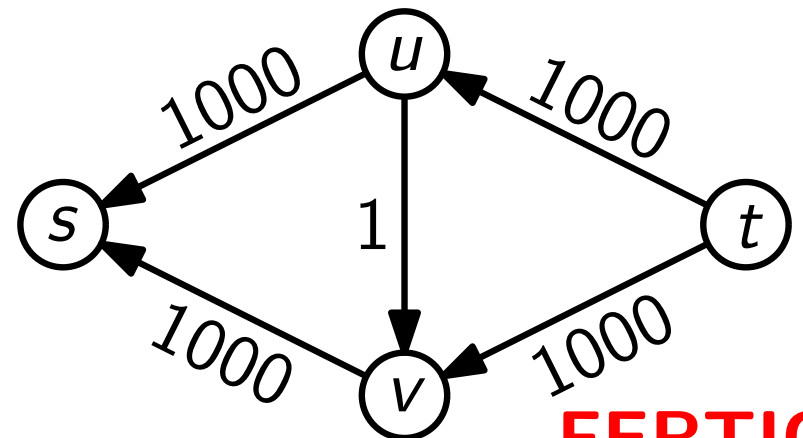
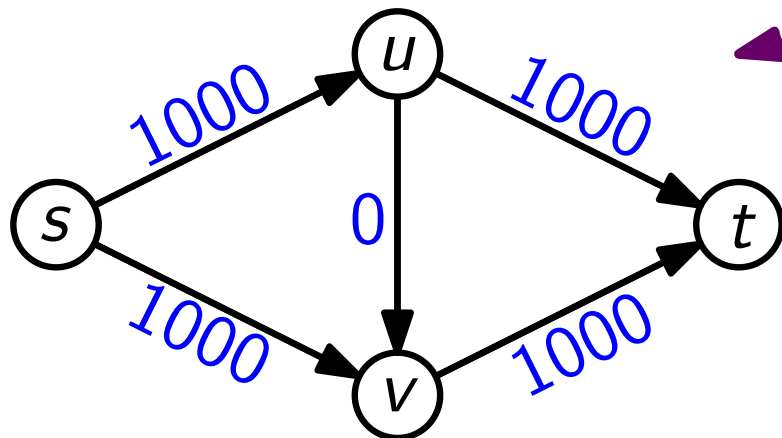
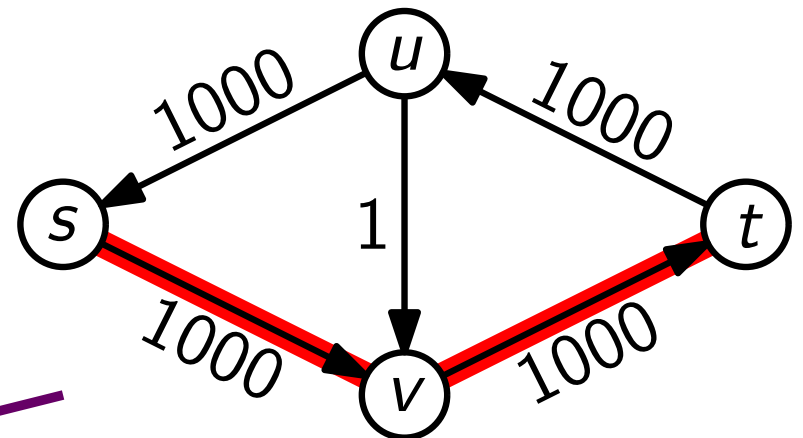
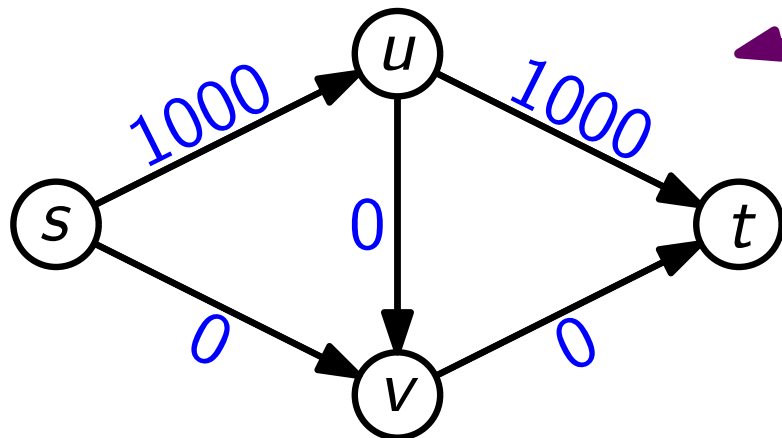
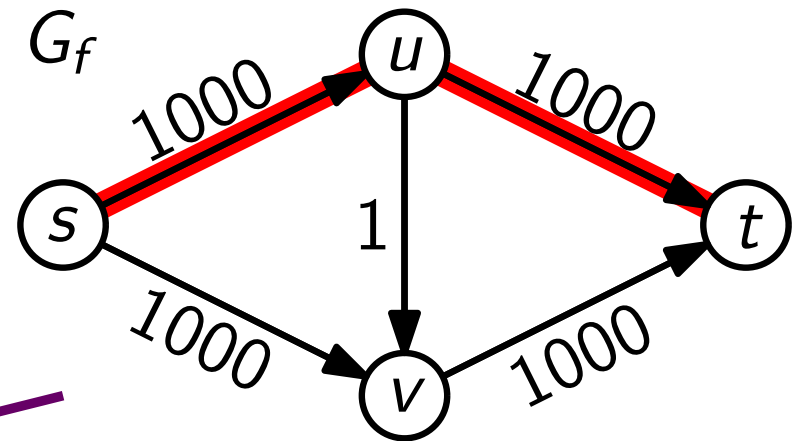
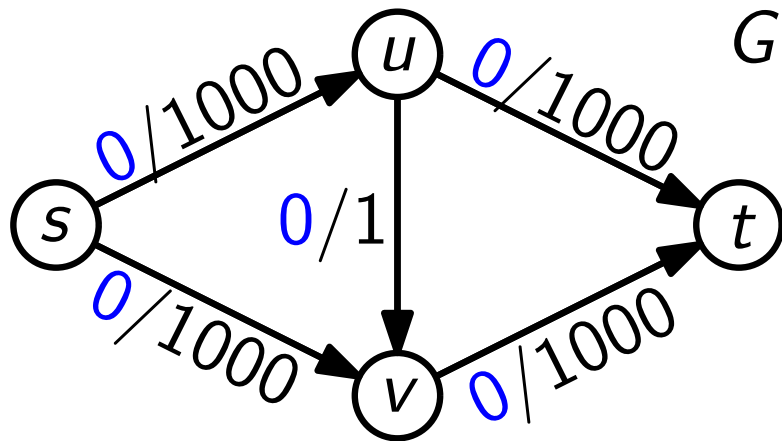
Beispiel



Beispiel

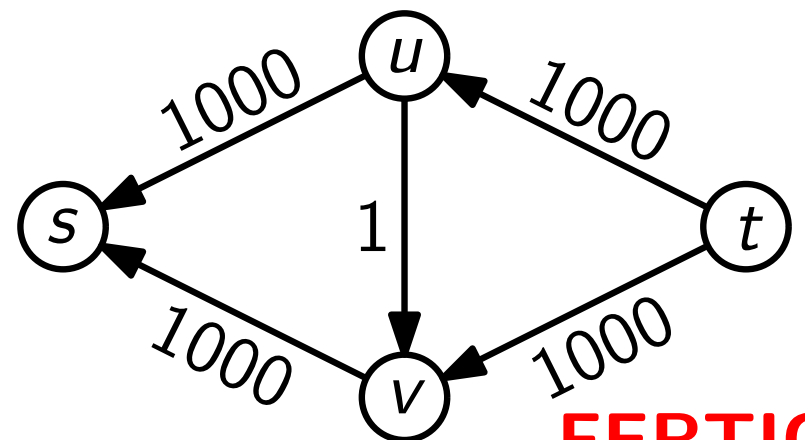
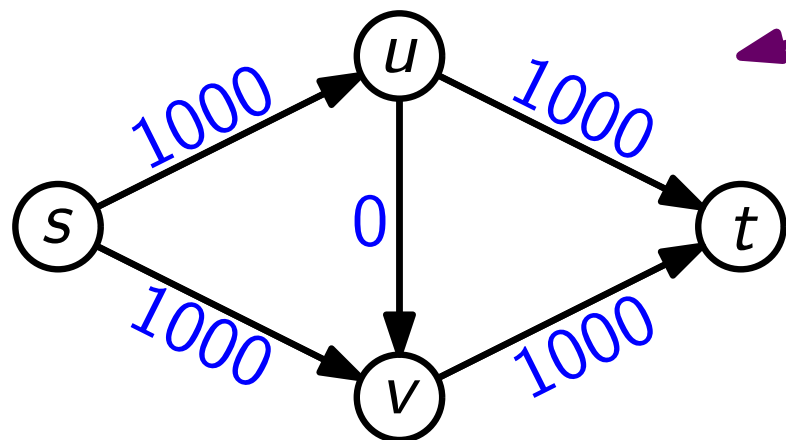
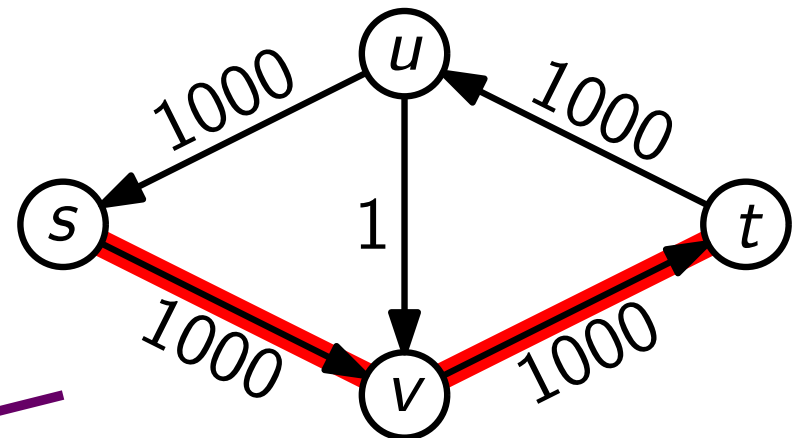
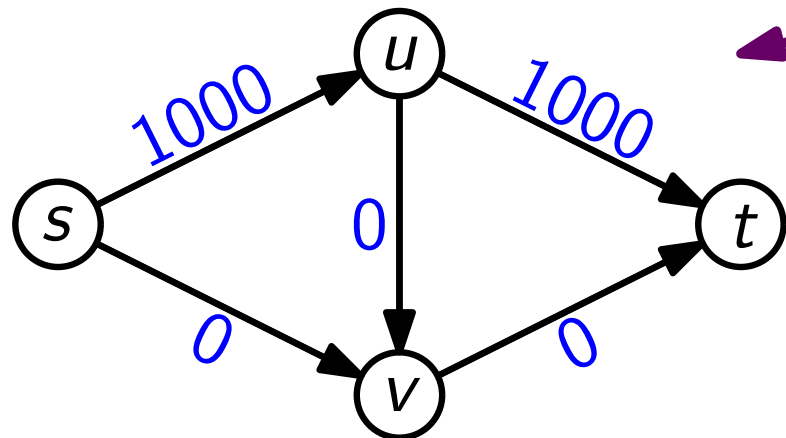
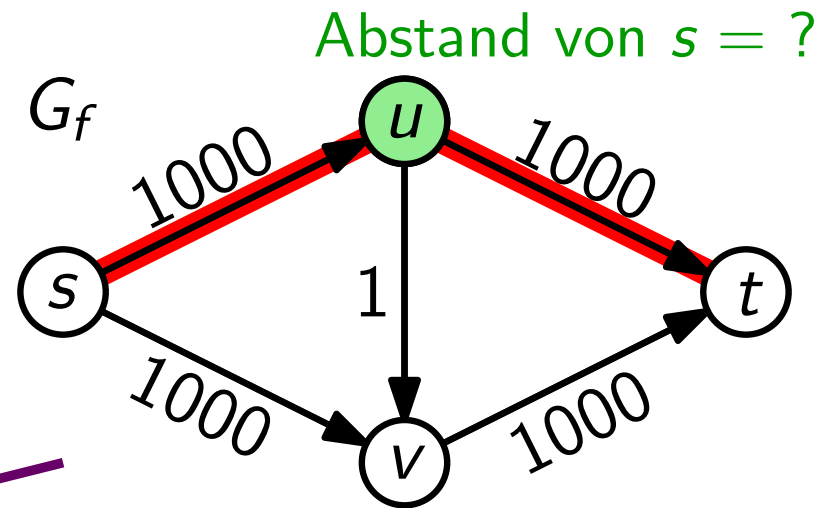
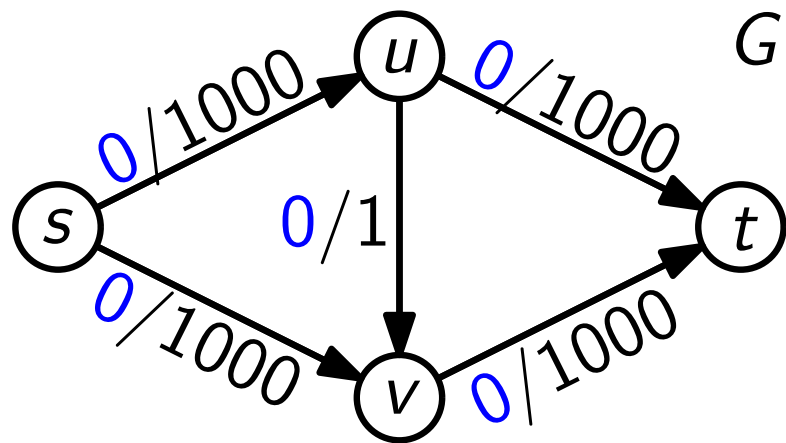


Beispiel



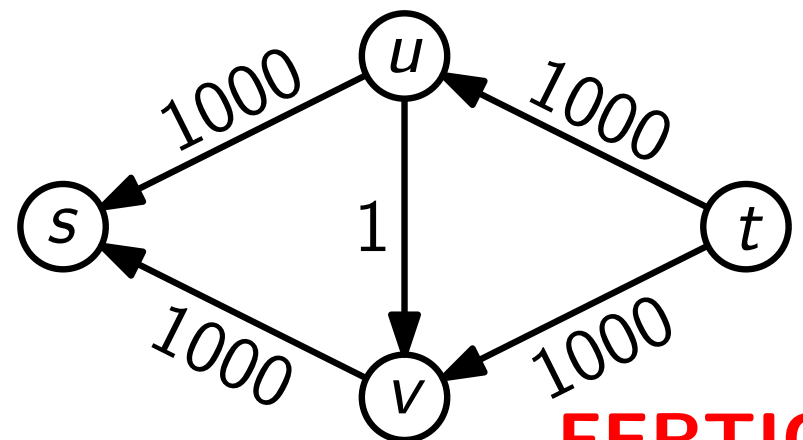
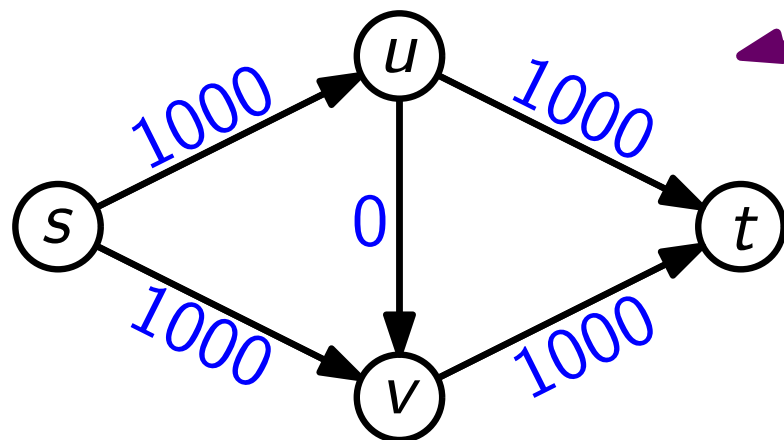
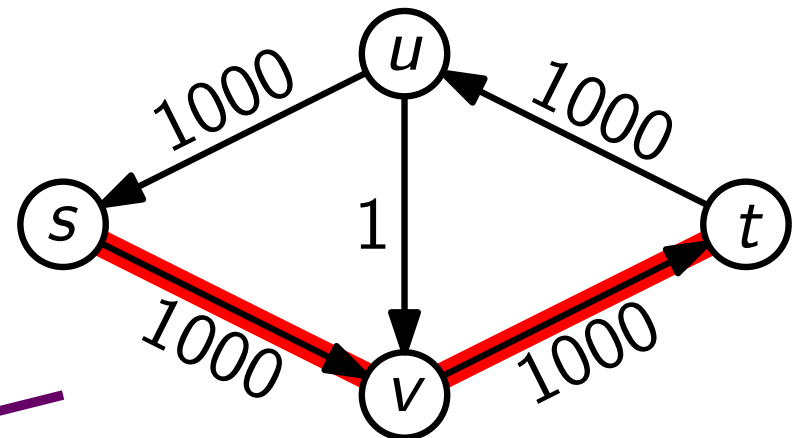
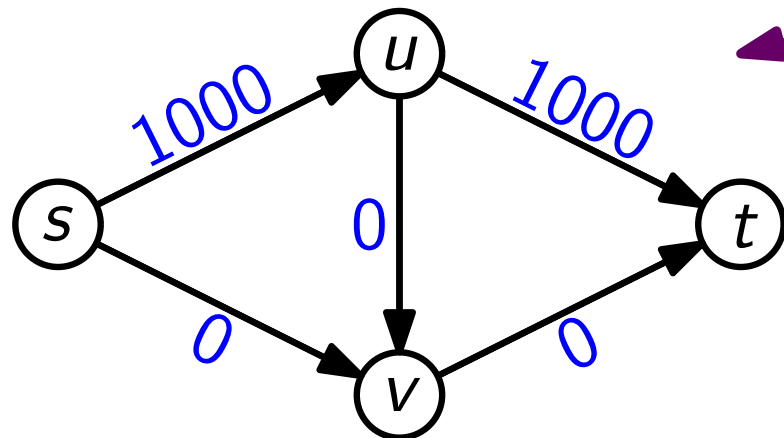
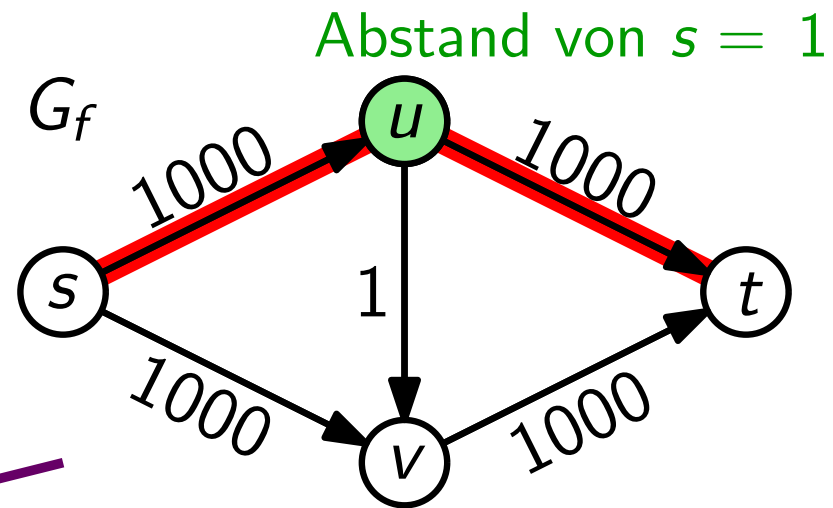
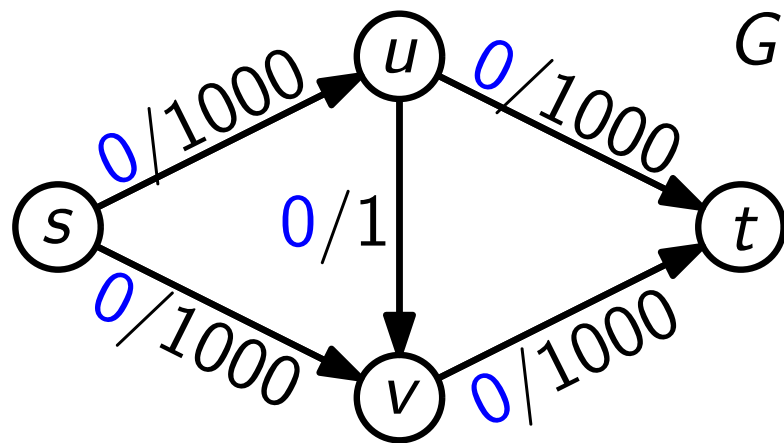
FERTIG!

Beispiel



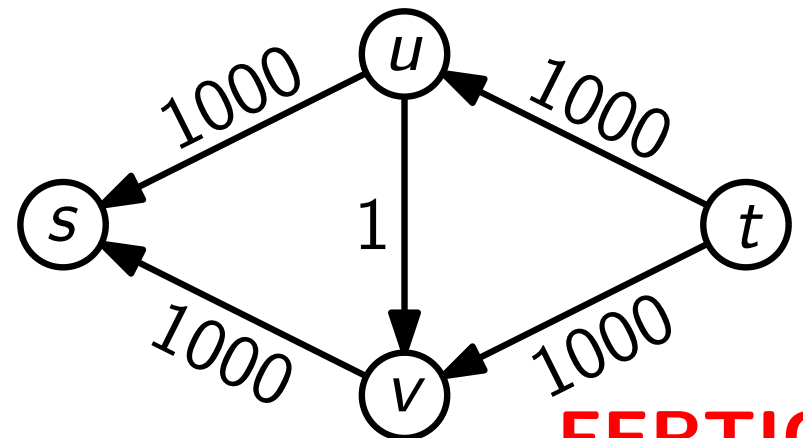
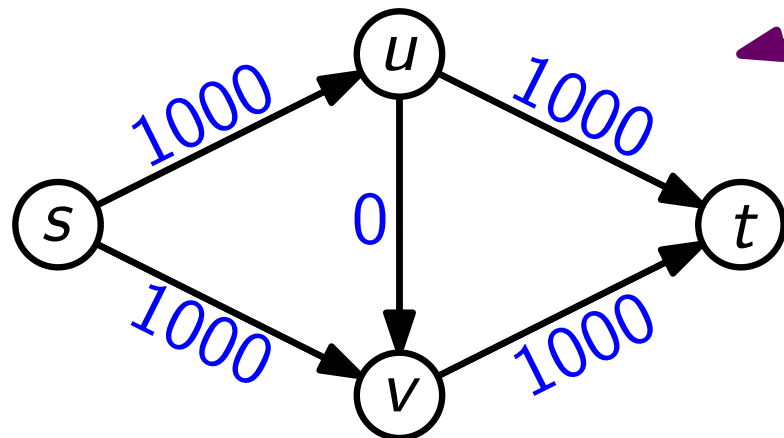
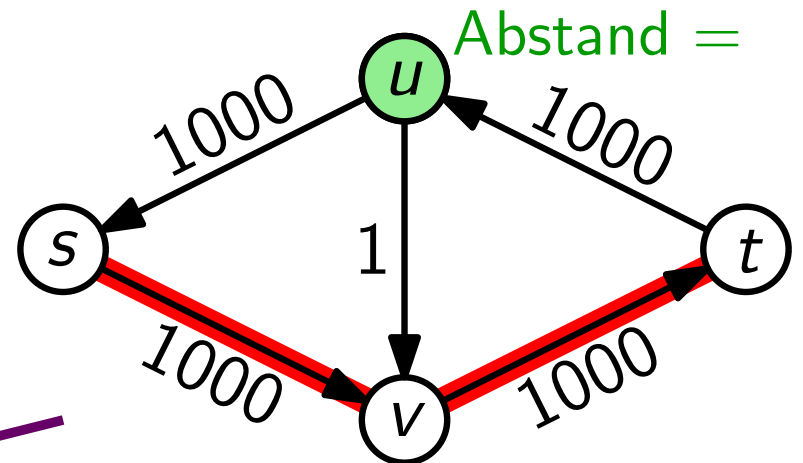
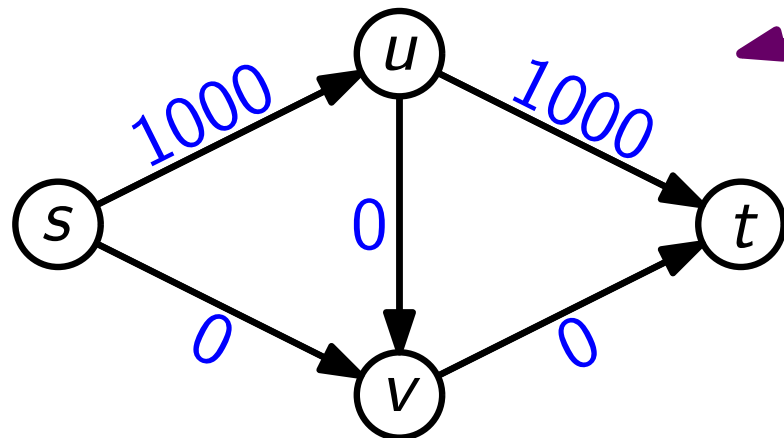
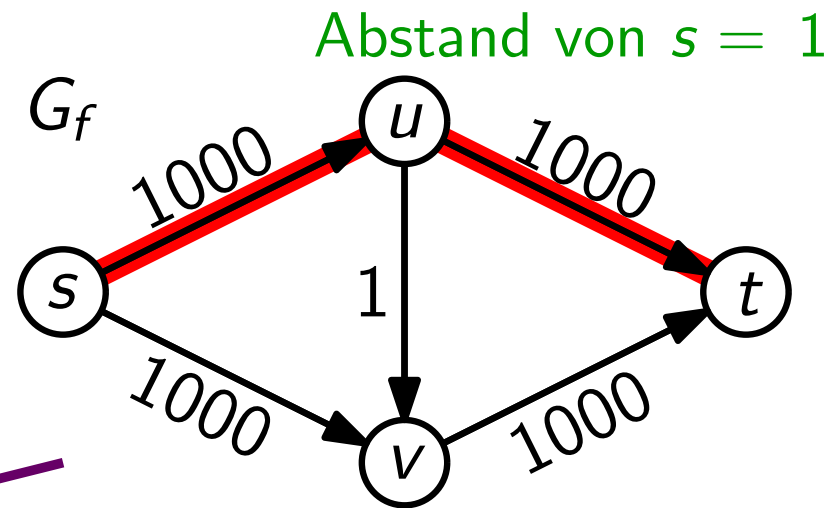
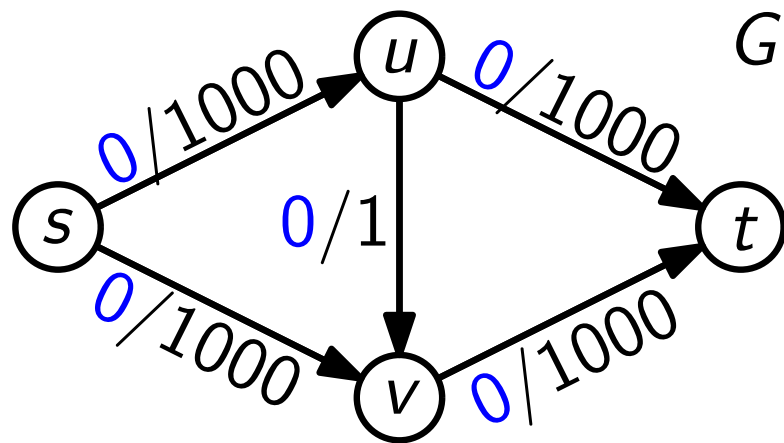
FERTIG!

Beispiel



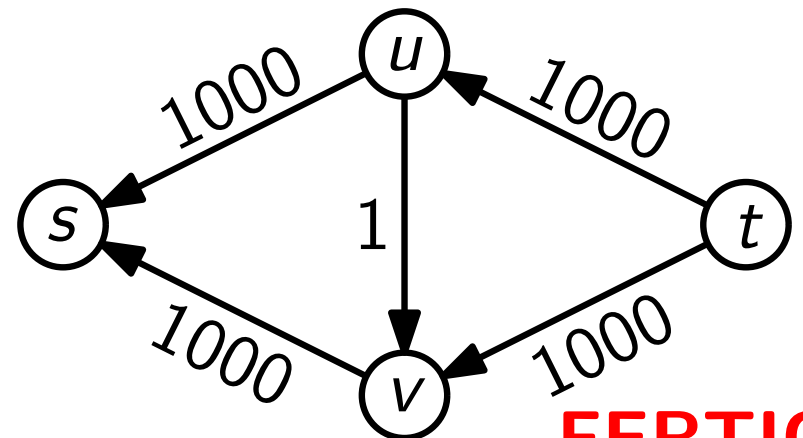
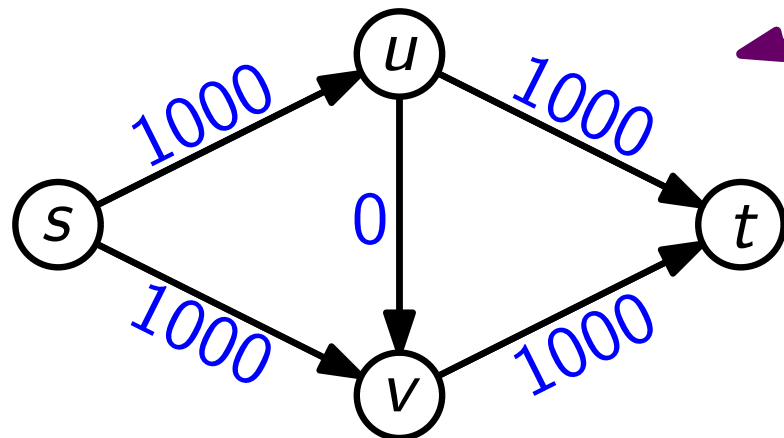
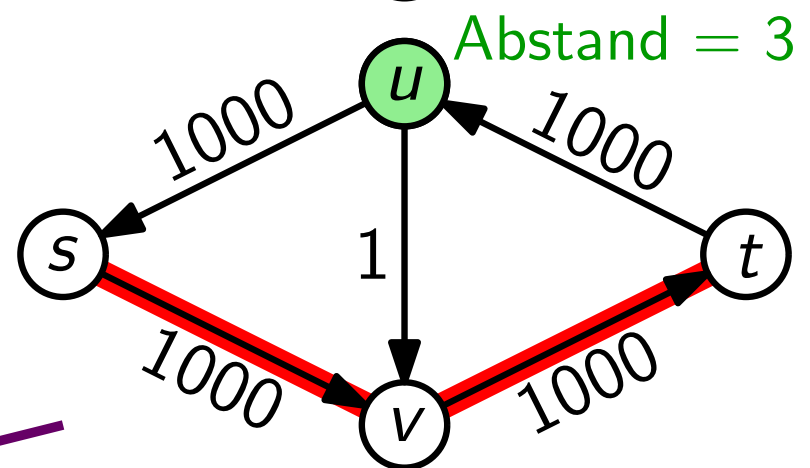
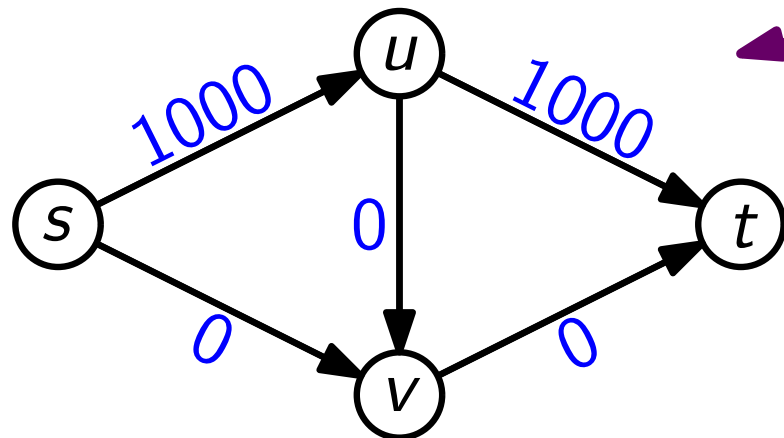
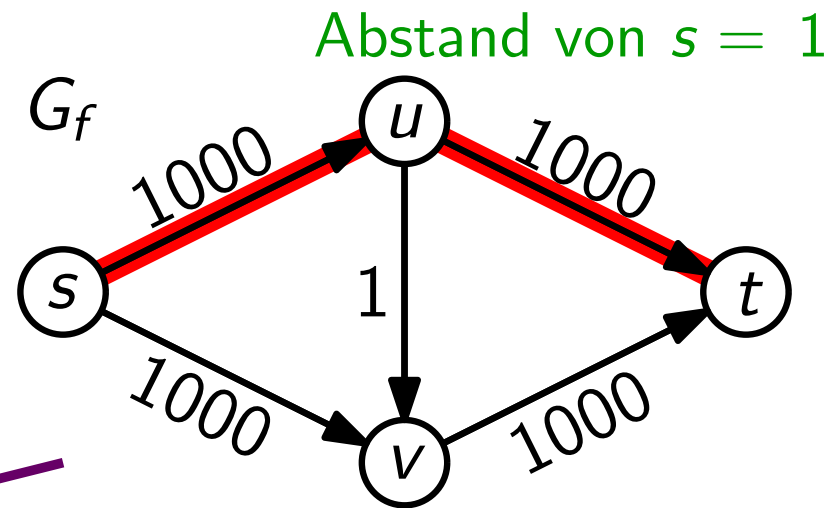
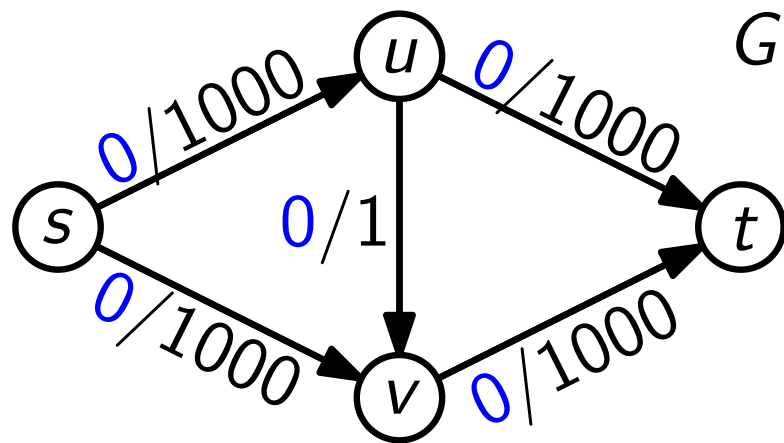
FERTIG!

Beispiel



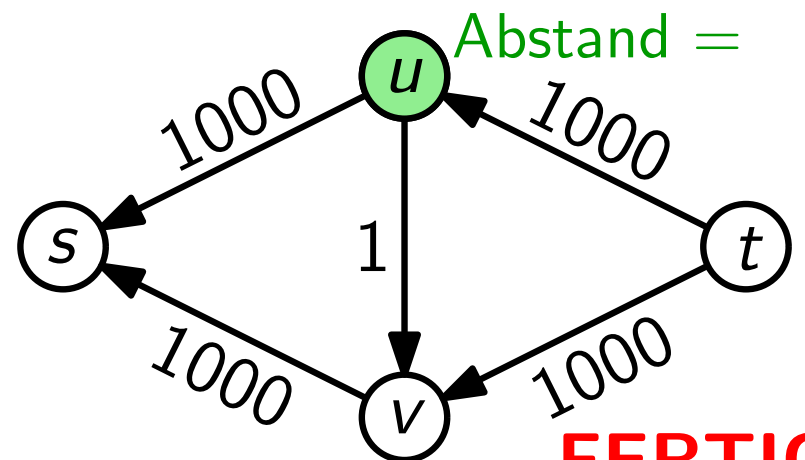
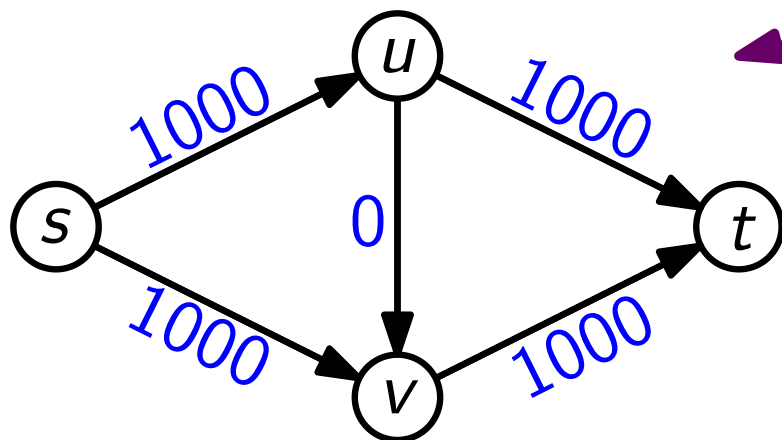
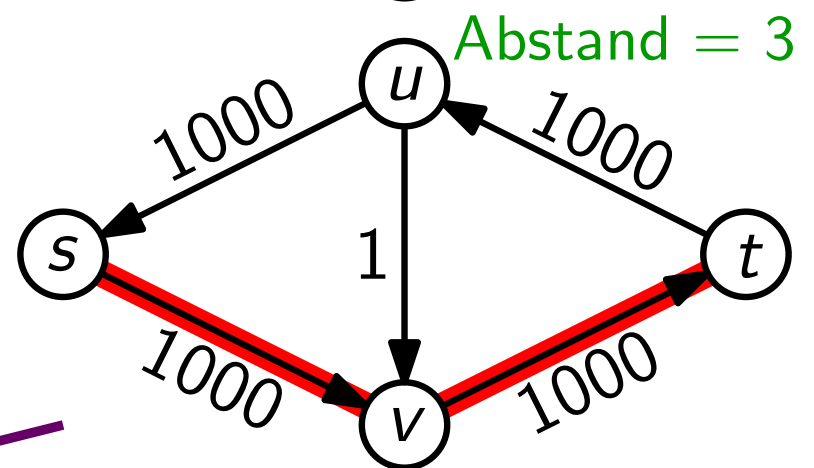
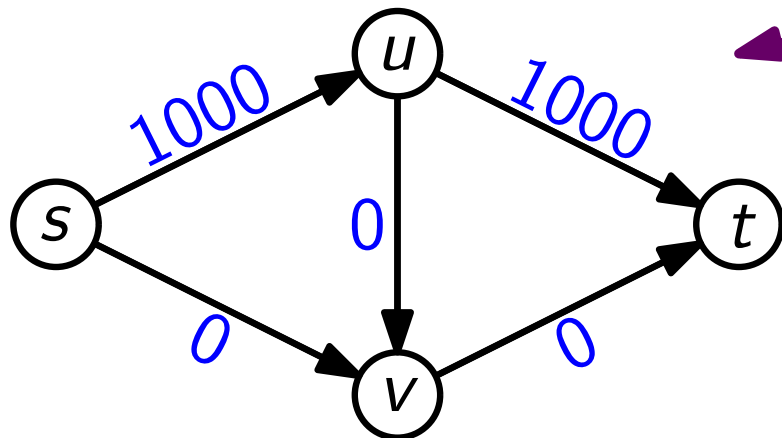
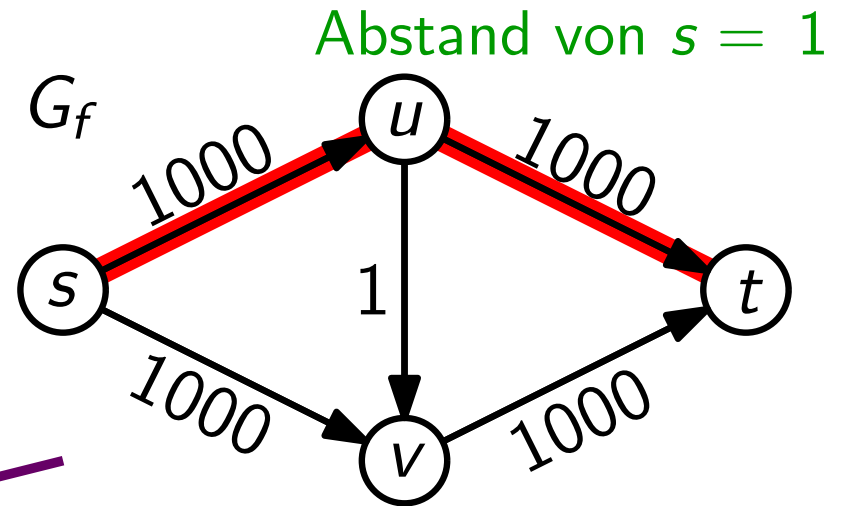
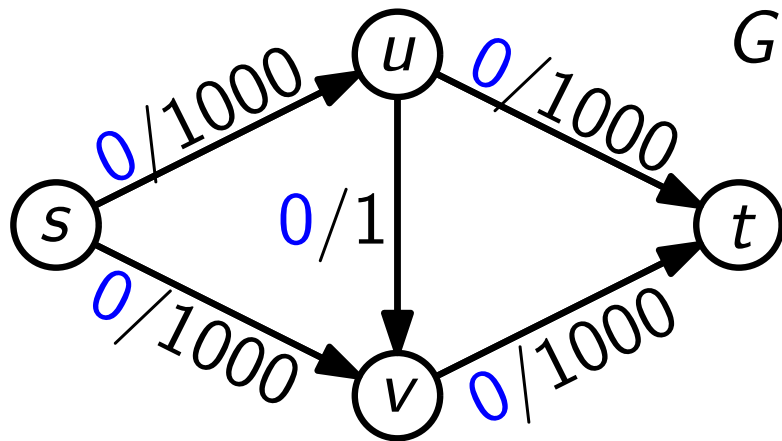
FERTIG!

Beispiel



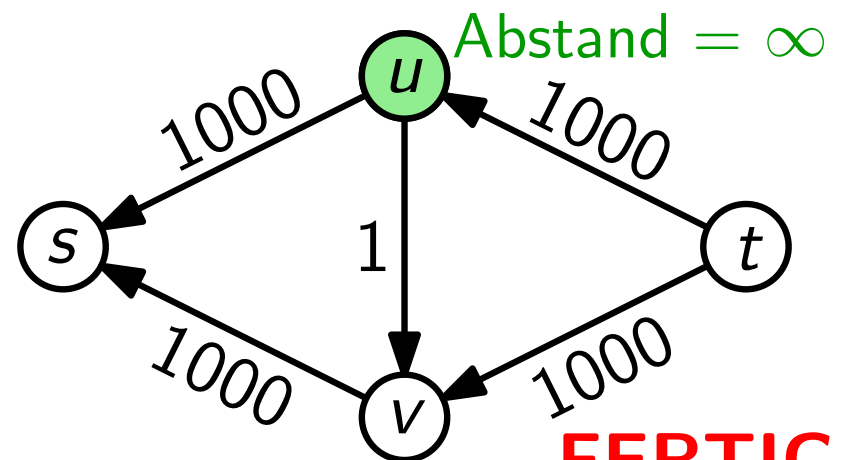
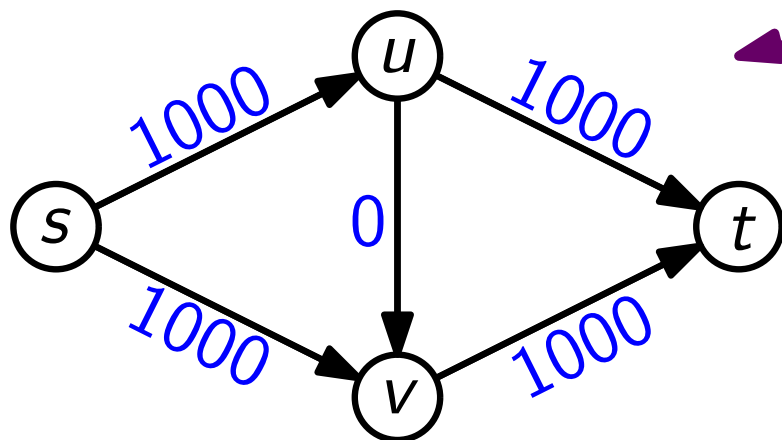
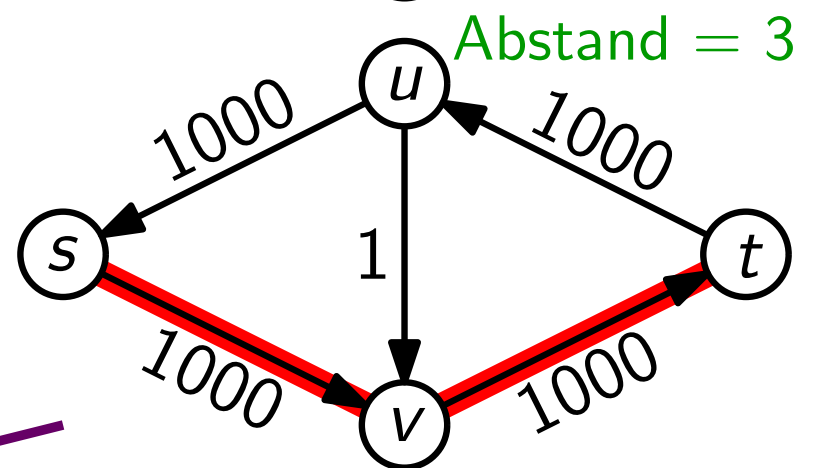
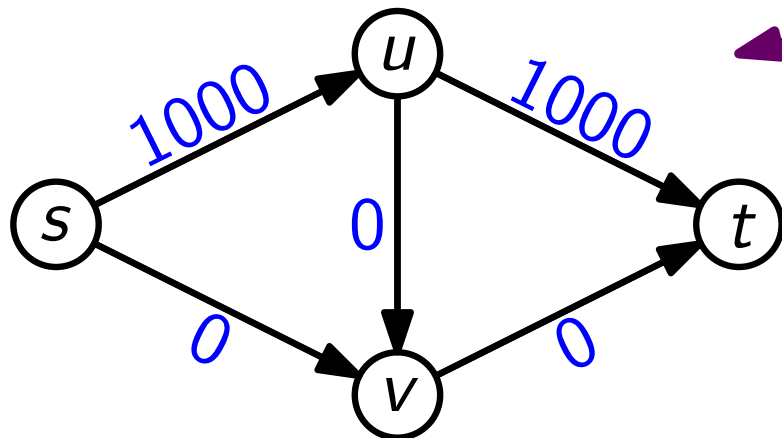
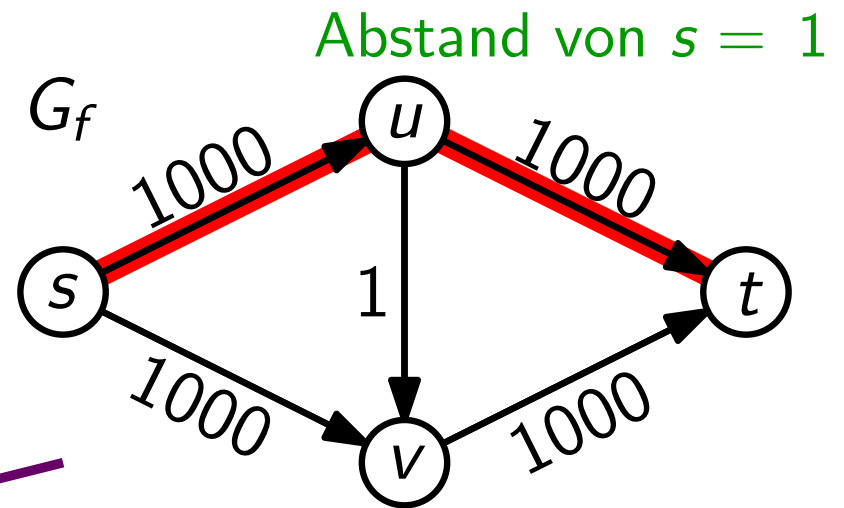
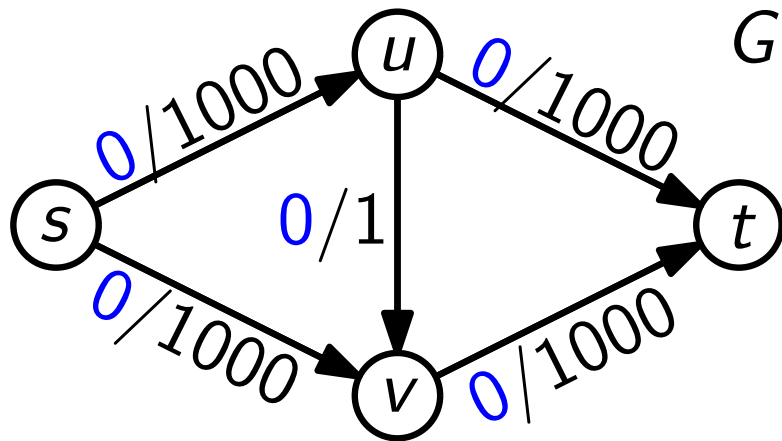
FERTIG!

Beispiel



FERTIG!

Beispiel



FERTIG!

Kürzeste Wege machen effiziente Algorithmen!

Def. Sei $\delta_f(u, v)$ die Länge (= Anz. Kanten) eines kürzesten u - v -Wegs in G_f .

Kürzeste Wege machen effiziente Algorithmen!

Def. Sei $\delta_f(u, v)$ die Länge (= Anz. Kanten) eines kürzesten u - v -Wegs in G_f .

Lemma. Während $\text{EdmondsKarp}(G, s, t)$ gilt für jeden Knoten $v \in V$, dass $\delta_f(s, v)$ mit jeder Flussvergrößerung monoton zunimmt.

Kürzeste Wege machen effiziente Algorithmen!

Def. Sei $\delta_f(u, v)$ die Länge (= Anz. Kanten) eines kürzesten u - v -Wegs in G_f .

Lemma. Während $\text{EdmondsKarp}(G, s, t)$ gilt für jeden Knoten $v \in V$, dass $\delta_f(s, v)$ mit jeder Flussvergrößerung monoton zunimmt.

Beweis.

Kürzeste Wege machen effiziente Algorithmen!

Def. Sei $\delta_f(u, v)$ die Länge (= Anz. Kanten) eines kürzesten u - v -Wegs in G_f .

Lemma. Während $\text{EdmondsKarp}(G, s, t)$ gilt für jeden Knoten $v \in V$, dass $\delta_f(s, v)$ mit jeder Flussvergrößerung monoton zunimmt.

Beweis. Annahme: es gibt einen Knoten v derart, dass $\delta_f(s, v)$ bei einer Vergrößerung von f *abnimmt*.

Kürzeste Wege machen effiziente Algorithmen!

Def. Sei $\delta_f(u, v)$ die Länge (= Anz. Kanten) eines kürzesten u - v -Wegs in G_f .

Lemma. Während $\text{EdmondsKarp}(G, s, t)$ gilt für jeden Knoten $v \in V$, dass $\delta_f(s, v)$ mit jeder Flussvergrößerung monoton zunimmt.

Beweis. Annahme: es gibt einen Knoten v derart, dass $\delta_f(s, v)$ bei einer Vergrößerung von f *abnimmt*.

Sei f der Fluss *vor* der Vergrößerung;
sei f' der Fluss *nach* der Vergrößerung.

Kürzeste Wege machen effiziente Algorithmen!

Def. Sei $\delta_f(u, v)$ die Länge (= Anz. Kanten) eines kürzesten u - v -Wegs in G_f .

Lemma. Während EdmondsKarp(G, s, t) gilt für jeden Knoten $v \in V$, dass $\delta_f(s, v)$ mit jeder Flussvergrößerung monoton zunimmt.

Beweis. Annahme: es gibt einen Knoten v derart, dass $\delta_f(s, v)$ bei einer Vergrößerung von f *abnimmt*.

Sei f der Fluss *vor* der Vergrößerung;
sei f' der Fluss *nach* der Vergrößerung.

Ab jetzt sei v unter den Knoten mit $\delta_{f'}(s, v) < \delta_f(s, v)$ einer mit *minimalem* Wert von $\delta_{f'}(s, v)$.

Kürzeste Wege machen effiziente Algorithmen!

Def. Sei $\delta_f(u, v)$ die Länge (= Anz. Kanten) eines kürzesten u - v -Wegs in G_f .

Lemma. Während $\text{EdmondsKarp}(G, s, t)$ gilt für jeden Knoten $v \in V$, dass $\delta_f(s, v)$ mit jeder Flussvergrößerung monoton zunimmt.

Beweis. Annahme: es gibt einen Knoten v derart, dass $\delta_f(s, v)$ bei einer Vergrößerung von f *abnimmt*.

Sei f der Fluss *vor* der Vergrößerung;
sei f' der Fluss *nach* der Vergrößerung.

„kleinster Schurke“

Ab jetzt sei v unter den Knoten mit $\delta_{f'}(s, v) < \delta_f(s, v)$ einer mit *minimalem* Wert von $\delta_{f'}(s, v)$.

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

\Rightarrow

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$ [u ist *kein* Schurke;
Abstand nimmt *nicht* ab.]

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$ [u ist *kein* Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$ [u ist *kein* Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$

Beweis.

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$ [u ist *kein* Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$

Beweis. Angenommen $uv \in E_f$

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$ [u ist *kein* Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$

Beweis. Angenommen $uv \in E_f \Rightarrow$

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$ [u ist kein Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$

Beweis. Angenommen $uv \in E_f \Rightarrow \delta_f(s, v) \leq \delta_f(s, u) + 1$

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

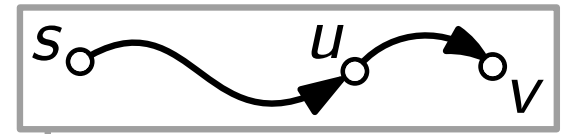
Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$

[u ist kein Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$



Beweis. Angenommen $uv \in E_f \Rightarrow \delta_f(s, v) \leq \delta_f(s, u) + 1$

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

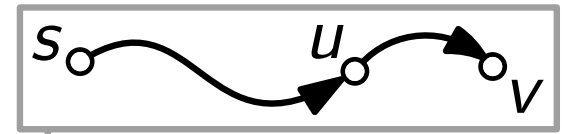
$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$

[u ist kein Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$

Beweis. Angenommen $uv \in E_f \Rightarrow \delta_f(s, v) \leq \delta_f(s, u) + 1$



Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$

*[u ist kein Schurke;
Abstand nimmt nicht ab.]*

Beh. $uv \notin E_f$

Beweis. Angenommen $uv \in E_f \Rightarrow \delta_f(s, v) \leq \delta_f(s, u) + 1$
 $\leq \delta_{f'}(s, u) + 1$



Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$

[u ist kein Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$

Beweis. Angenommen $uv \in E_f \Rightarrow \delta_f(s, v) \leq \delta_f(s, u) + 1$
 $\leq \delta_{f'}(s, u) + 1$



Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$

[u ist kein Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$

Beweis. Angenommen $uv \in E_f \Rightarrow \delta_f(s, v) \leq \delta_f(s, u) + 1$
 $\leq \delta_{f'}(s, u) + 1$
 $= \delta_{f'}(s, v)$



Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$

[u ist kein Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$

Beweis. Angenommen $uv \in E_f \Rightarrow \delta_f(s, v) \leq \delta_f(s, u) + 1$
 $\leq \delta_{f'}(s, u) + 1$
 $= \delta_{f'}(s, v)$



Widerspruch zur Annahme, dass $\delta_{f'}(s, v) < \delta_f(s, v)$. \triangle

Fortsetzung Beweis

Sei W ein kürzester s - v -Weg in $G_{f'}$.

Sei u der letzte Knoten vor v auf W .

$\Rightarrow uv \in E_{f'}$ und $\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1$. [Eig. kürzester Wege]

Nach Wahl von v gilt: $\delta_{f'}(s, u) \geq \delta_f(s, u)$

[u ist kein Schurke;
Abstand nimmt *nicht* ab.]

Beh. $uv \notin E_f$

Beweis. Angenommen $uv \in E_f \Rightarrow \delta_f(s, v) \leq \delta_f(s, u) + 1$
 $\leq \delta_{f'}(s, u) + 1$
 $= \delta_{f'}(s, v)$



Widerspruch zur Annahme, dass $\delta_{f'}(s, v) < \delta_f(s, v)$. \triangle

Aber wie können wir $uv \notin E_f$ und $uv \in E_{f'}$ erklären??

Fortsetzung II

1. Fall: $uv \in E$

2. Fall: $vu \in E$

Fortsetzung II

1. Fall: $uv \in E$
 $uv \notin E_f$ bedeutet

2. Fall: $vu \in E$

Fortsetzung II

1. Fall: $uv \in E$
 $uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

2. Fall: $vu \in E$

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet

2. Fall: $vu \in E$

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

2. Fall: $vu \in E$

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Da EdmondsKarp entlang kürzester Wege vergrößert, muss v Vorgänger von u auf einem kürzesten s - u -Weg in G_f sein.

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.


$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Da EdmondsKarp entlang kürzester Wege vergrößert, muss v Vorgänger von u auf einem kürzesten s - u -Weg in G_f sein.

$\Rightarrow \delta_f(s, v) =$



Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$


$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Da EdmondsKarp entlang kürzester Wege vergrößert, muss v Vorgänger von u auf einem kürzesten s - u -Weg in G_f sein.

$$\Rightarrow \delta_f(s, v) = \delta_f(s, u) - 1$$


Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Da EdmondsKarp entlang kürzester Wege vergrößert, muss v Vorgänger von u auf einem kürzesten s - u -Weg in G_f sein.

$$\Rightarrow \delta_f(s, v) = \delta_f(s, u) - 1 \leq$$

[u kein Schurke]

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Da EdmondsKarp entlang kürzester Wege vergrößert, muss v Vorgänger von u auf einem kürzesten s - u -Weg in G_f sein.

$$\Rightarrow \delta_f(s, v) = \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1$$

[u kein Schurke]

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$vu \notin E_f$ bedeutet $f(vu) = 0$.

$vu \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Da EdmondsKarp entlang kürzester Wege vergrößert, muss v Vorgänger von u auf einem kürzesten s - u -Weg in G_f sein.

$$\Rightarrow \delta_f(s, v) = \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1 =$$

[u kein Schurke]
[u liegt auf W vor v]

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Da EdmondsKarp entlang kürzester Wege vergrößert, muss v Vorgänger von u auf einem kürzesten s - u -Weg in G_f sein.

$$\Rightarrow \delta_f(s, v) \overset{[u \text{ kein Schurke}]}{=} \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1 \overset{[u \text{ liegt auf } W \text{ vor } v]}{=} \delta_{f'}(s, v) - 2$$

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Da EdmondsKarp entlang kürzester Wege vergrößert, muss v Vorgänger von u auf einem kürzesten s - u -Weg in G_f sein.

$$\Rightarrow \delta_f(s, v) \stackrel{[u \text{ kein Schurke}]}{=} \delta_f(s, u) - 1 \stackrel{[u \text{ liegt auf } W \text{ vor } v]}{\leq} \delta_{f'}(s, u) - 1 = \delta_{f'}(s, v) - 2$$

$<$

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$uv \notin E_f$ bedeutet $f(vu) = 0$.

$uv \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Da EdmondsKarp entlang kürzester Wege vergrößert, muss v Vorgänger von u auf einem kürzesten s - u -Weg in G_f sein.

$$\Rightarrow \delta_f(s, v) = \delta_f(s, u) - 1 \stackrel{[u \text{ kein Schurke}]}{\leq} \delta_{f'}(s, u) - 1 \stackrel{[u \text{ liegt auf } W \text{ vor } v]}{=} \delta_{f'}(s, v) - 2 < \delta_{f'}(s, v).$$

Fortsetzung II

1. Fall: $uv \in E$

$uv \notin E_f$ bedeutet $f(uv) = c(uv)$.

$uv \in E_{f'}$ bedeutet $f'(uv) < c(uv)$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

2. Fall: $vu \in E$

$vu \notin E_f$ bedeutet $f(vu) = 0$.

$vu \in E_{f'}$ bedeutet $f'(vu) > 0$.

\rightsquigarrow Flussvergrößerung entlang $vu \in E_f$

Der Fluss wird in beiden Fällen **entlang vu vergrößert**.

Da EdmondsKarp entlang kürzester Wege vergrößert, muss v Vorgänger von u auf einem kürzesten s - u -Weg in G_f sein.

$$\begin{aligned} \Rightarrow \delta_f(s, v) &= \overset{[u \text{ kein Schurke}]}{\delta_f(s, u) - 1} \leq \overset{[u \text{ liegt auf } W \text{ vor } v]}{\delta_{f'}(s, u) - 1} = \delta_{f'}(s, v) - 2 \\ &< \delta_{f'}(s, v). \quad \text{Widerspr. zur Ann. } \delta_{f'}(s, v) < \delta_f(s, v). \end{aligned}$$

□

Anzahl Flusserhöhungen & Laufzeit

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Anzahl Flusserhöhungen & Laufzeit

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Korollar. Der Edmonds-Karp-Algorithmus läuft in $O(VE^2)$ Zeit.

Anzahl Flusserhöhungen & Laufzeit

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Korollar. Der Edmonds-Karp-Algorithmus läuft in $O(VE^2)$ Zeit.

Beweis. Jede der $O(VE)$ Flussvergrößerungen benötigt $O(E)$ Zeit bei Anwendung von Breitensuche. \square

Beweis (Satz)

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis. Sei f der aktuelle Fluss in G .

Algorithmus vergrößere entlang kürzestem s - t -Weg W in G_f .

Beweis (Satz)

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis. Sei f der aktuelle Fluss in G .

Algorithmus vergrößere entlang kürzestem s - t -Weg W in G_f .

Kante uv auf W heißt *kritisch* in G_f , wenn $c_f(uv) = \Delta_W$.

Beweis (Satz)

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis. Sei f der aktuelle Fluss in G .

Algorithmus vergrößere entlang kürzestem s - t -Weg W in G_f .

Kante uv auf W heißt *kritisch* in G_f , wenn $c_f(uv) = \Delta_W$.

Zeige: Jede Kante kann höchstens $O(V)$ mal kritisch sein.

Beweis (Satz)

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis. Sei f der aktuelle Fluss in G .

Algorithmus vergrößere entlang kürzestem s - t -Weg W in G_f .

Kante uv auf W heißt *kritisch* in G_f , wenn $c_f(uv) = \Delta_W$.

Zeige: Jede Kante kann höchstens $O(V)$ mal kritisch sein.

$\delta_f(s, v) = \delta_f(s, u) + 1$, da uv auf kürzestem Weg W in G_f .

Beweis (Satz)

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis. Sei f der aktuelle Fluss in G .

Algorithmus vergrößere entlang kürzestem s - t -Weg W in G_f .

Kante uv auf W heißt *kritisch* in G_f , wenn $c_f(uv) = \Delta_W$.

Zeige: Jede Kante kann höchstens $O(V)$ mal kritisch sein.

$\delta_f(s, v) = \delta_f(s, u) + 1$, da uv auf kürzestem Weg W in G_f .

Nach Flussvergrößerung entlang W verschwindet uv aus G_f .

Beweis (Satz)

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis. Sei f der aktuelle Fluss in G .

Algorithmus vergrößere entlang kürzestem s - t -Weg W in G_f .

Kante uv auf W heißt *kritisch* in G_f , wenn $c_f(uv) = \Delta_W$.

Zeige: Jede Kante kann höchstens $O(V)$ mal kritisch sein.

$\delta_f(s, v) = \delta_f(s, u) + 1$, da uv auf kürzestem Weg W in G_f .

Nach Flussvergrößerung entlang W verschwindet uv aus G_f .

Die Kante uv erscheint erst wieder im Residualgraphen, nachdem Fluss entlang vu vergrößert wurde

Beweis (Satz)

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis. Sei f der aktuelle Fluss in G .

Algorithmus vergrößere entlang kürzestem s - t -Weg W in G_f .

Kante uv auf W heißt *kritisch* in G_f , wenn $c_f(uv) = \Delta_W$.

Zeige: Jede Kante kann höchstens $O(V)$ mal kritisch sein.

$\delta_f(s, v) = \delta_f(s, u) + 1$, da uv auf kürzestem Weg W in G_f .

Nach Flussvergrößerung entlang W verschwindet uv aus G_f .

Die Kante uv erscheint erst wieder im Residualgraphen,

nachdem Fluss entlang vu vergrößert wurde $\Rightarrow vu \in E_{f'}$

(f' = Fluss nach Vergrößerung)

Fortsetzung Beweis

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis (Forts.)

Da vu auf kürzestem flussvergrößerndem Weg in $G_{f'}$ liegt, gilt

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$$

Fortsetzung Beweis

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis (Forts.)

Da vu auf kürzestem flussvergrößerndem Weg in $G_{f'}$ liegt, gilt

$$\begin{aligned} \delta_{f'}(s, u) &= \delta_{f'}(s, v) + 1 \\ &\geq \end{aligned}$$

voriges Lemma

Fortsetzung Beweis

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis (Forts.)

Da vu auf kürzestem flussvergrößerndem Weg in $G_{f'}$ liegt, gilt

$$\begin{aligned}
 \delta_{f'}(s, u) &= \delta_{f'}(s, v) + 1 \\
 &\geq \delta_f(s, v) + 1 \\
 &=
 \end{aligned}$$

voriges Lemma

Fortsetzung Beweis

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis (Forts.)

Da vu auf kürzestem flussvergrößerndem Weg in $G_{f'}$ liegt, gilt

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$$

$$uv \text{ auf } W \text{ in } G_f \geq \delta_f(s, v) + 1$$

=

voriges Lemma

Fortsetzung Beweis

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis (Forts.)

Da vu auf kürzestem flussvergrößerndem Weg in $G_{f'}$ liegt, gilt

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$$

voriges Lemma

$$uv \text{ auf } W \text{ in } G_f \geq \delta_f(s, v) + 1$$

$$= \delta_f(s, u) + 1 + 1.$$

Fortsetzung Beweis

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis (Forts.)

Da vu auf kürzestem flussvergrößerndem Weg in $G_{f'}$ liegt, gilt

$$\begin{aligned} \delta_{f'}(s, u) &= \delta_{f'}(s, v) + 1 && \text{voriges Lemma} \\ uv \text{ auf } W \text{ in } G_f &\geq \delta_f(s, v) + 1 \\ &= \delta_f(s, u) + 1 + 1. \end{aligned}$$

Also steigt $\delta_{\square}(s, u)$ bei jeder Flusserhöhung um ≥ 2 .

Fortsetzung Beweis

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis (Forts.)

Da vu auf kürzestem flussvergrößerndem Weg in $G_{f'}$ liegt, gilt

$$\begin{aligned} \delta_{f'}(s, u) &= \delta_{f'}(s, v) + 1 && \text{voriges Lemma} \\ uv \text{ auf } W \text{ in } G_f &\geq \delta_f(s, v) + 1 \\ &= \delta_f(s, u) + 1 + 1. \end{aligned}$$

Also steigt $\delta_{\square}(s, u)$ bei jeder Flusserhöhung um ≥ 2 .

Kürzeste Wege sind *einfach*, d.h. besuchen jeden Knoten $\leq 1 \times$.

Fortsetzung Beweis

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis (Forts.)

Da vu auf kürzestem flussvergrößerndem Weg in $G_{f'}$ liegt, gilt

$$\begin{aligned} \delta_{f'}(s, u) &= \delta_{f'}(s, v) + 1 && \text{voriges Lemma} \\ uv \text{ auf } W \text{ in } G_f &\geq \delta_f(s, v) + 1 \\ &= \delta_f(s, u) + 1 + 1. \end{aligned}$$

Also steigt $\delta_{\square}(s, u)$ bei jeder Flusserhöhung um ≥ 2 .

Kürzeste Wege sind *einfach*, d.h. besuchen jeden Knoten $\leq 1 \times$.

$\Rightarrow \delta_{\square}(s, u) < |V|$, solange u von s erreicht werden kann.

Fortsetzung Beweis

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis (Forts.)

Da vu auf kürzestem flussvergrößerndem Weg in $G_{f'}$ liegt, gilt

$$\begin{aligned} \delta_{f'}(s, u) &= \delta_{f'}(s, v) + 1 && \text{voriges Lemma} \\ uv \text{ auf } W \text{ in } G_f &\geq \delta_f(s, v) + 1 \\ &= \delta_f(s, u) + 1 + 1. \end{aligned}$$

Also steigt $\delta_{\square}(s, u)$ bei jeder Flusserhöhung um ≥ 2 .

Kürzeste Wege sind *einfach*, d.h. besuchen jeden Knoten $\leq 1 \times$.

$\Rightarrow \delta_{\square}(s, u) < |V|$, solange u von s erreicht werden kann.

\Rightarrow Die Kante uv kann nur $O(V)$ mal kritisch sein. □

Kurze Geschichte der Berechnung max. Flüsse

<i>Methode</i>	<i>Laufzeit $O(\cdot)$</i>	<i>Autoren</i>
Allgemeine gerichtete Graphen		
shortest resid. s - t path	VE^2	Dinic '70, Ed. & Karp '72

Kurze Geschichte der Berechnung max. Flüsse

<i>Methode</i>	<i>Laufzeit $O(\cdot)$</i>	<i>Autoren</i>
Allgemeine gerichtete Graphen		
shortest resid. s - t path	VE^2	Dinic '70, Ed. & Karp '72
push relabel	V^2E	Goldberg '87

Kurze Geschichte der Berechnung max. Flüsse

<i>Methode</i>	<i>Laufzeit $O(\cdot)$</i>	<i>Autoren</i>
Allgemeine gerichtete Graphen		
shortest resid. s - t path	VE^2	Dinic '70, Ed. & Karp '72
push relabel	V^2E	Goldberg '87
relabel to front	V^3	Goldberg & Tarjan '88
	$VE \log(V^2/E + 2)$	— " —

Kurze Geschichte der Berechnung max. Flüsse

<i>Methode</i>	<i>Laufzeit $O(\cdot)$</i>	<i>Autoren</i>
Allgemeine gerichtete Graphen		
shortest resid. s - t path	VE^2	Dinic '70, Ed. & Karp '72
push relabel	V^2E	Goldberg '87
relabel to front	V^3	Goldberg & Tarjan '88
	$VE \log(V^2/E + 2)$	— " —
blocking flow	$\min(V^{2/3}, E^{1/2}) \cdot E \cdot$ $\cdot \log(V^2/E + 2) \cdot \log C,$	Goldberg & Rao '98 wobei $C = \sum_{e \in E} c(e)$

Kurze Geschichte der Berechnung max. Flüsse

<i>Methode</i>	<i>Laufzeit $O(\cdot)$</i>	<i>Autoren</i>
Allgemeine gerichtete Graphen		
shortest resid. s - t path	VE^2	Dinic '70, Ed. & Karp '72
push relabel	V^2E	Goldberg '87
relabel to front	V^3	Goldberg & Tarjan '88
	$VE \log(V^2/E + 2)$	— " —
blocking flow	$\min(V^{2/3}, E^{1/2}) \cdot E \cdot$ $\cdot \log(V^2/E + 2) \cdot \log C,$	Goldberg & Rao '98 wobei $C = \sum_{e \in E} c(e)$
new	VE	Orlin '13

Kurze Geschichte der Berechnung max. Flüsse

<i>Methode</i>	<i>Laufzeit $O(\cdot)$</i>	<i>Autoren</i>
Allgemeine gerichtete Graphen		
shortest resid. s - t path	VE^2	Dinic '70, Ed. & Karp '72
push relabel	V^2E	Goldberg '87
relabel to front	V^3	Goldberg & Tarjan '88
	$VE \log(V^2/E + 2)$	— " —
blocking flow	$\min(V^{2/3}, E^{1/2}) \cdot E \cdot$ $\cdot \log(V^2/E + 2) \cdot \log C,$	Goldberg & Rao '98 wobei $C = \sum_{e \in E} c(e)$
new	VE	Orlin '13
s-t-planare Graphen		
shortest path in dual	V	Hassin '81 + Henzinger et al. '97

Kurze Geschichte der Berechnung max. Flüsse

<i>Methode</i>	<i>Laufzeit $O(\cdot)$</i>	<i>Autoren</i>
Allgemeine gerichtete Graphen		
shortest resid. s - t path	VE^2	Dinic '70, Ed. & Karp '72
push relabel	V^2E	Goldberg '87
relabel to front	V^3	Goldberg & Tarjan '88
	$VE \log(V^2/E + 2)$	— " —
blocking flow	$\min(V^{2/3}, E^{1/2}) \cdot E \cdot$ $\cdot \log(V^2/E + 2) \cdot \log C,$	Goldberg & Rao '98 wobei $C = \sum_{e \in E} c(e)$
new	VE	Orlin '13
s-t-planare Graphen		
shortest path in dual	V	Hassin '81 + Henzinger et al. '97
Planare Graphen		
leftmost resid. s - t path	$V \log V$	Borradaile & Klein '06
+ vertex capacities	$V \log V$	Kaplan & Nussbaum '09