



Julius-Maximilians-

**UNIVERSITÄT
WÜRZBURG**

Lehrstuhl für

INFORMATIK I

Algorithmen & Komplexität



Institut für Informatik

Algorithmische Graphentheorie

Sommersemester 2025

4. Vorlesung

Flussalgorithmen

Lehrstuhl für Informatik I

Alexander Wolff

Aufgabe

Indeed, it's an LP!

Gegeben ein gerichteter Graph G mit Knoten s und t sowie einer Kantenkapazitäten $c: E(G) \rightarrow \mathbb{R}_{>0}$.

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

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

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

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

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

maximiert.

lineare Zielfunktion!

Flussalgorithmen

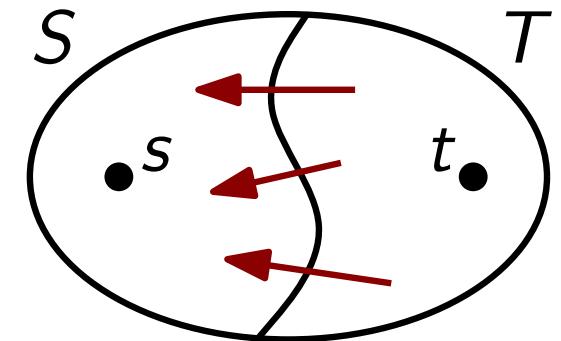
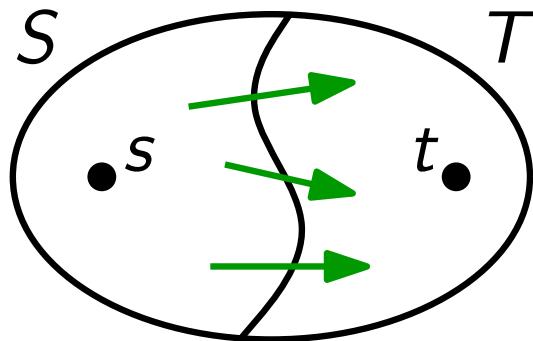
Kann man maximale Flüsse (= Spezialfall eines LPs) auch mit maßgeschneiderten kombinatorischen Algorithmen berechnen?

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

Nichts ist praktischer als eine gute Theorie

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

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

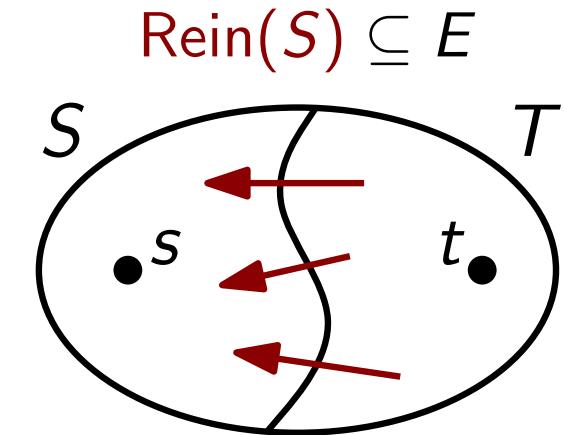
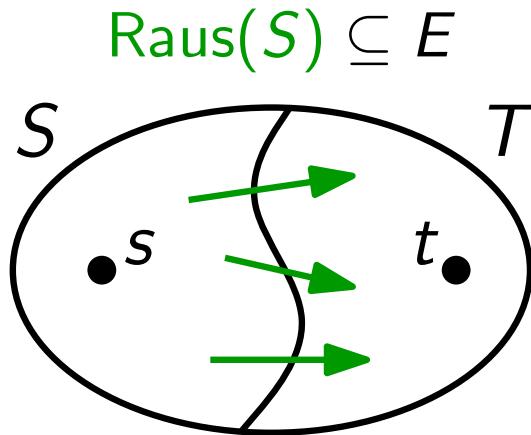


$$\begin{aligned} \text{Raus}(S) &= \{uv \in E(G) \mid u \in S, v \in T\} \\ &\quad \{uv \in E(G) \mid u \in T, v \in S\} = \text{Rein}(S) \end{aligned}$$

$$\left. \begin{array}{l} \text{Zufluss}_f(S) = f(\text{Rein}(S)) \\ \text{Abfluss}_f(S) = f(\text{Raus}(S)) \end{array} \right\} =: \text{Nettozufluss}_f(S)$$

Nettozuflüsse von Schnitten und Knoten

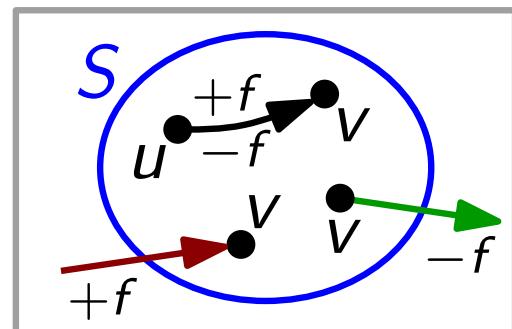
Zur Erinnerung: $\text{Nettozufluss}_f(S) := f(\text{Rein}(S)) - f(\text{Raus}(S))$



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

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

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



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

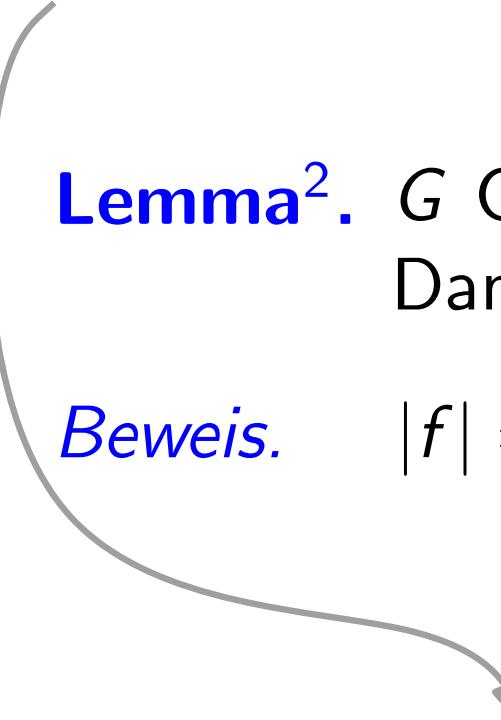
□

Noch mehr Schnitte

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

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

Beweis.



$$\begin{aligned}|f| &=_{\text{Def.}} \text{Nettozufluss}_f(t) \\&= \sum_{v \in T} \text{Nettozufluss}_f(v) \\&\quad \text{da } \text{Nettozufluss}_f(v) = 0 \text{ für alle } v \neq s, t \\&= \text{Nettozufluss}_f(T)\end{aligned}$$
□

Kapazität von Schnitten

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

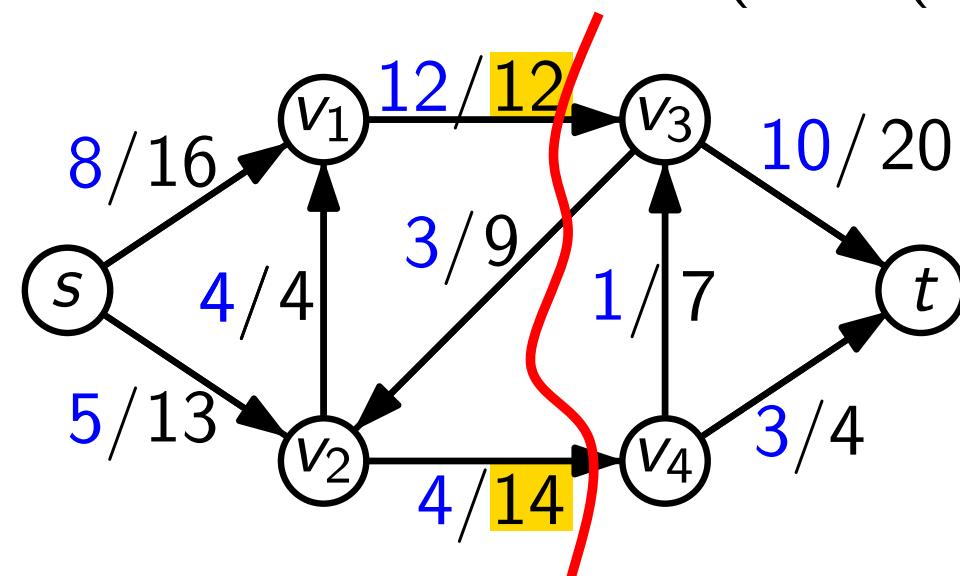
Def. G Graph mit Kap. $c: E(G) \rightarrow \mathbb{R}_{>0}$, (S, T) s - t -Schnitt.
Dann ist $c(S) := c(\text{Raus}(S))$ die *Kapazität* von (S, T) .

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. !!

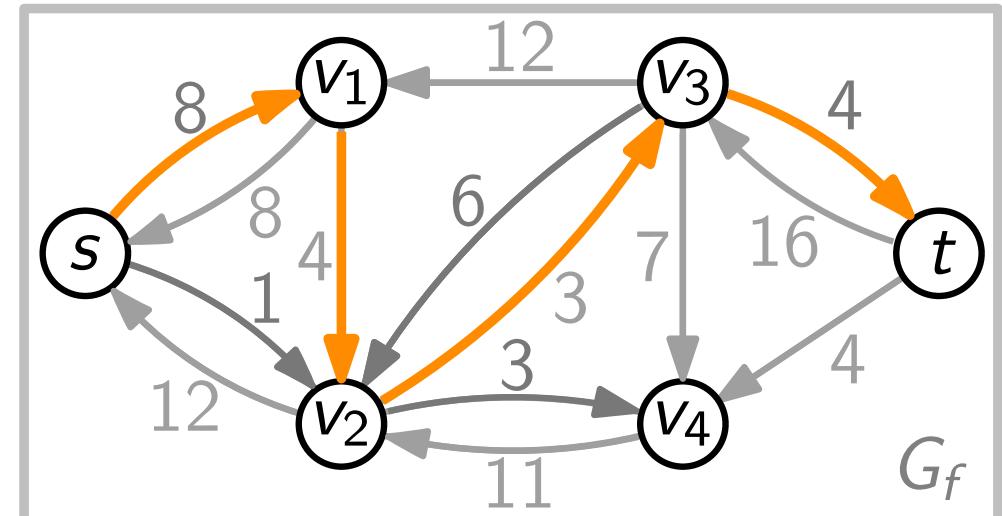
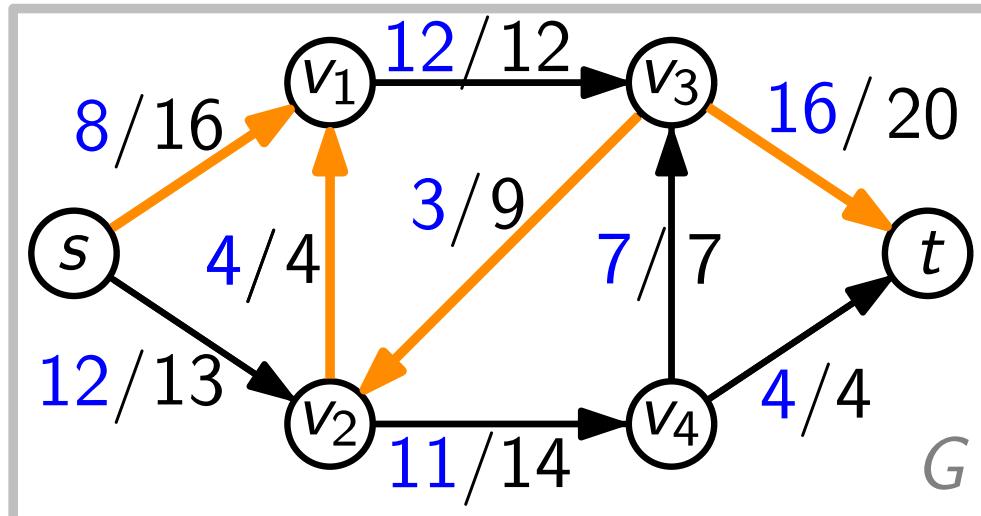
Residualnetz

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

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 $c_f(-e) := f(e)$



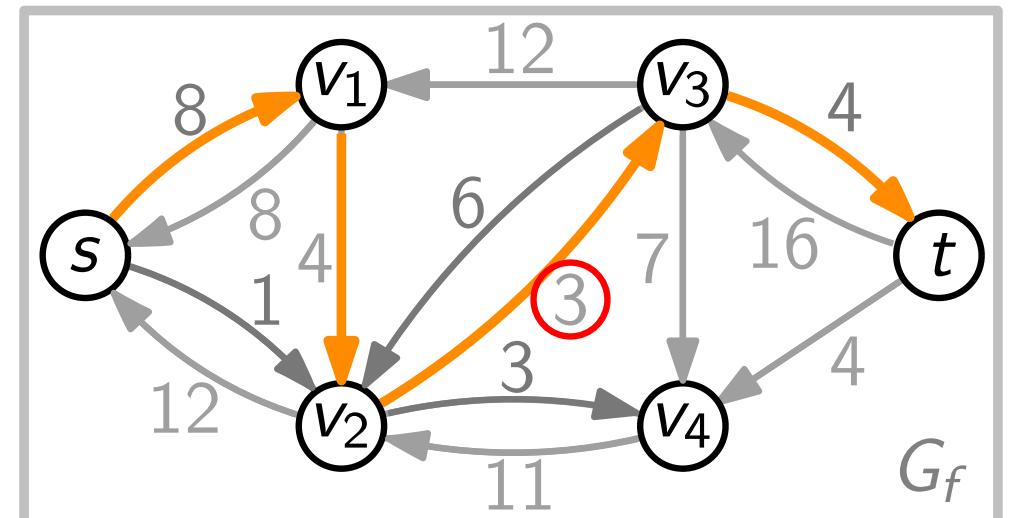
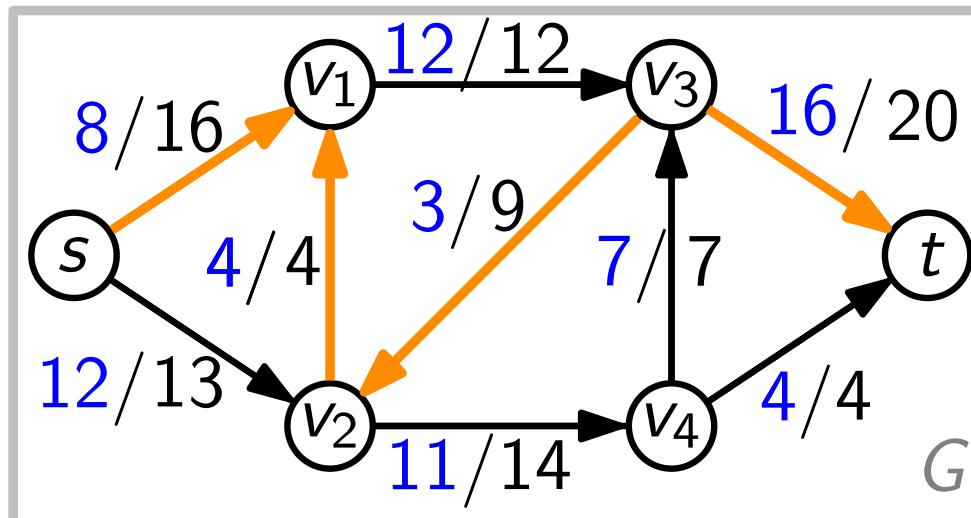
Augmentierende Wege

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

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

Satz (*vom augmentierenden [flussvergrößernden] Weg*). Ein zulässiger s - t -Fluss f in G ist maximal \Leftrightarrow es gibt keinen augmentierenden 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:

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 $|f| = c(S)$.

$(2) \Rightarrow (3)$: 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: $c(S) \stackrel{\text{Def.}}{=} c(\text{Raus}(S)) = f(\text{Raus}(S)) - f(\text{Rein}(S))$
 $= \text{Nettoabfluss}_f(S) = |f|$



Der Algorithmus von Ford & Fulkerson

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

foreach $uv \in E$ **do** } Initialisierung mit Null-Fluss
 $f_{uv} = 0$

while G_f enthält s - t -Weg W **do**

$$\Delta_W = \min_{uv \in W} c_f(uv) \quad \} \text{ Residualkapazität von } W$$

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**

$f_{uv} = f_{uv} + \Delta_W$

} Flussvergrößerung entlang 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**

└ $f_{uv} = 0$

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

Berechnung von $s-t$ -Wegen

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

Anz. Schleifendurchläufe

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

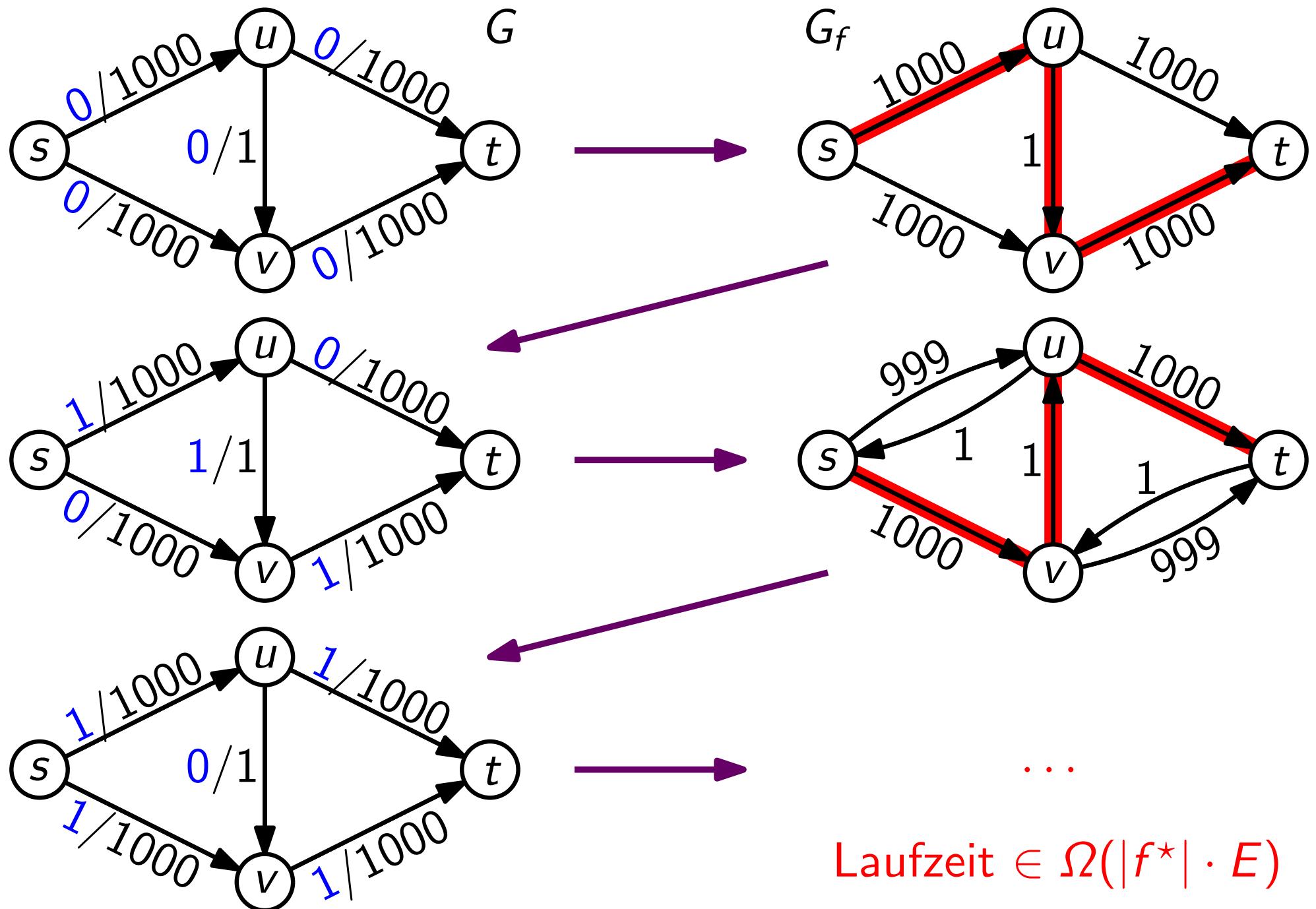
Laufzeit?

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

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

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

Beispiel



Der Algorithmus von Edmonds & Karp

~~EdmondsKarp~~

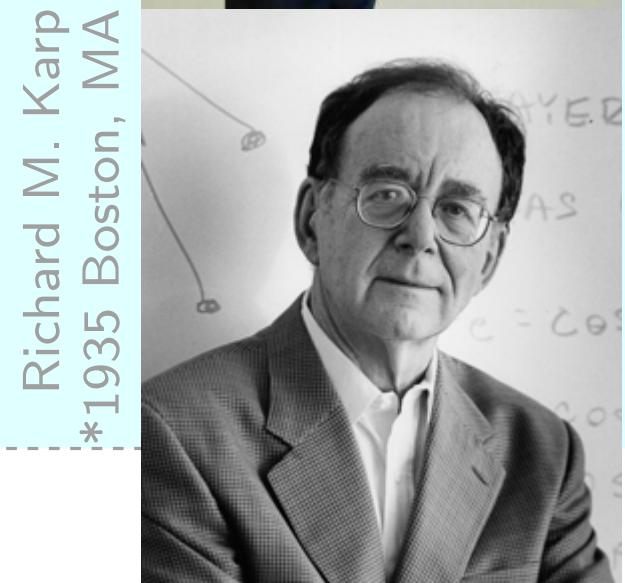
~~FordFulkerson~~(DirectedGraph $G = (V, E; c)$, Vertex s , Vertex t)

```

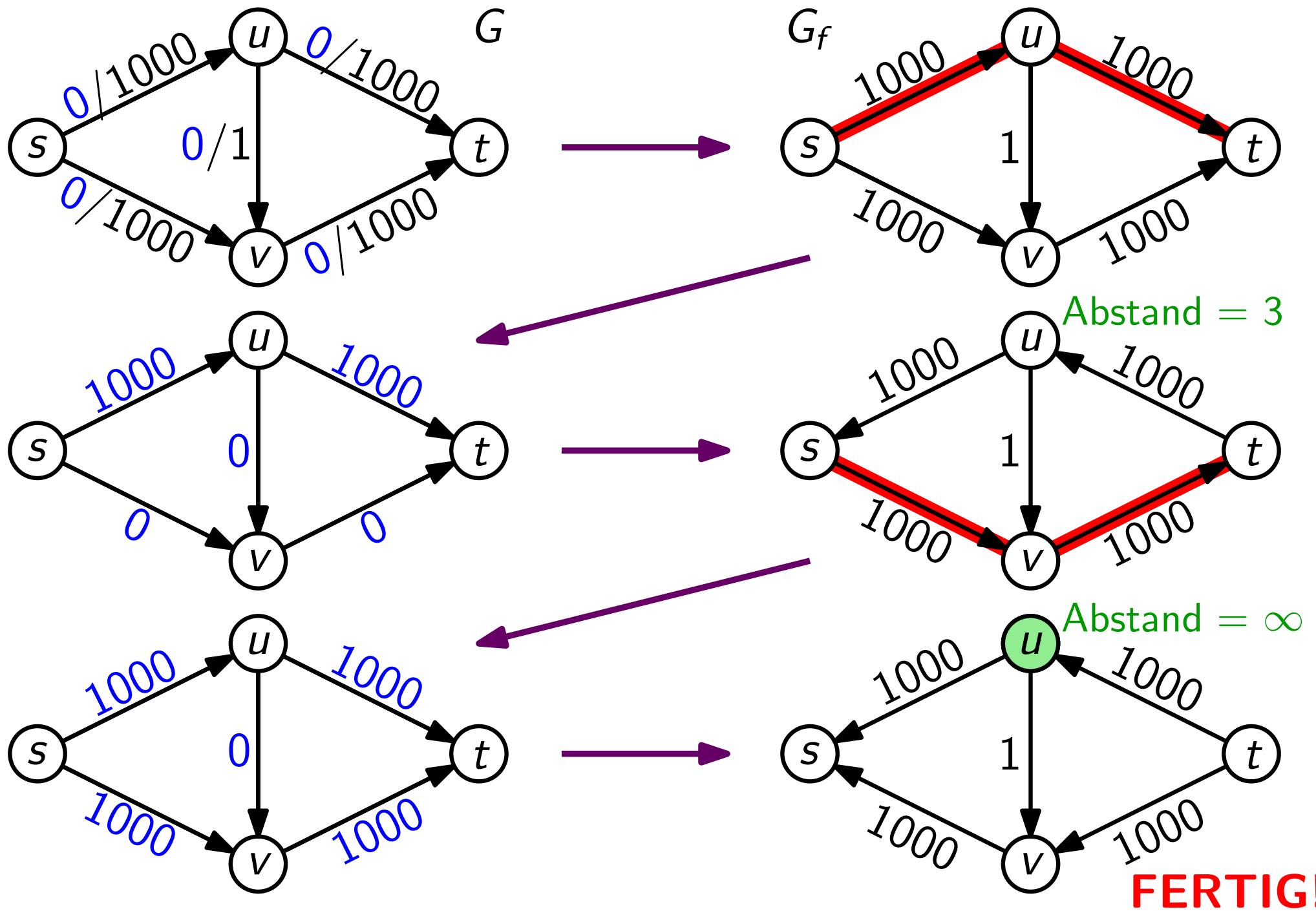
foreach  $uv \in E$  do
   $f_{uv} = 0$ 
while  $G_f$  enthält  $s$ - $t$ -Weg do
   $W = \text{kürzester } s\text{-}t\text{-Weg in } G_f$ 
   $\Delta_W = \min_{uv \in W} c_f(uv)$ 
  foreach  $uv \in W$  do
    if  $uv \in E$  then
       $f_{uv} = f_{uv} + \Delta_W$ 
    else
       $f_{vu} = f_{vu} - \Delta_W$ 
return  $f$ 

```

Jack R. Edmonds
*1934



Beispiel



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 $G_{f'}$.

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



$$\begin{aligned} \delta_f(s, v) &\leq \delta_f(s, u) + 1 \\ &\leq \delta_{f'}(s, u) + 1 \\ &= \delta_{f'}(s, v) \end{aligned}$$

Widerspruch zur Annahme, dass $\delta_{f'}(s, v) < \delta_f(s, v)$. 

Aber wie können wir $uv \notin E_f$ und $uv \in E_{f'}$ erklären??

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.

$$\Rightarrow \delta_f(s, v) = \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1 = \delta_{f'}(s, v) - 2$$

[u kein Schurke] [u liegt auf W vor v]

$$< \delta_{f'}(s, v). \text{ Widerspr. zur Ann. } \delta_{f'}(s, v) < \delta_f(s, v).$$



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. □

Beweis (Satz)

Satz. EdmondsKarp(G, s, t) führt $O(VE)$ Flussvergrößerungen durch.

Beweis. Sei f der aktuelle, nicht maximale Fluss in G . Sei W der kürzeste $s-t$ -Weg in G_f , entlang dem der Fluss vergrößert wird. Kante e auf W heißt *kritisch*, wenn $c_f(e) = \Delta_W$.

Zeige: Jede Kante uv 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 direkt vor dieser Vergrößerung (aber *nach* dem Zeitpunkt, als f der Fluss war.)

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

$$\begin{aligned}\delta_{f'}(s, u) &= \delta_{f'}(s, v) + 1 \\ \text{voriges Lemma} \quad &\geq \delta_f(s, v) + 1 \\ \text{uv auf } W \text{ in } G_f \quad &= \delta_f(s, u) + 1 + 1.\end{aligned}$$

Also steigt $\delta_{\square}(s, u)$, bis uv das nächste Mal kritisch ist, 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

Methode	Laufzeit $O(\cdot)$	Autoren
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 — " —
blocking flow	$VE \log(V^2/E + 2)$ $\min(V^{2/3}, E^{1/2}) \cdot E \cdot$ $\cdot \log(V^2/E + 2) \cdot \log C$,	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 + vertex capacities	$V \log V$ $V \log V$	Borradaile & Klein '06 Kaplan & Nussbaum '09