Algorithmen und Datenstrukturen
Kapitel 10
Flüsse

Frank Heitmann
heitmann@informatik.uni-hamburg.de

6. Januar 2016

Wiederholung

- Graphen
  - Grundlagen
    - Definition und Darstellung
    - Tiefen- und Breitensuche
  - Topologisches Sortieren
  - Starke Zusammenhangskomponenten
  - Minimale Spannbäume
    - Definition
    - Algorithmus von Kruskal
    - Algorithmus von Prim
  - Bestimmen der kürzesten Pfade
    - Definition und Varianten
    - (Breitensuche)
    - Algorithmus von Dijkstra
    - Algorithmus von Bellman-Ford
    - Algorithmus von Floyd-Warshall
  - Heute: Flüsse

Wiederholung

- Graphen
  - Grundlagen
    - Definition und Darstellung
    - Tiefen- und Breitensuche
  - Topologisches Sortieren
  - Starke Zusammenhangskomponenten
  - Minimale Spannbäume
    - Definition
    - Algorithmus von Kruskal
    - Algorithmus von Prim
  - Bestimmen der kürzesten Pfade
    - Definition und Varianten
    - (Breitensuche)
    - Algorithmus von Dijkstra
    - Algorithmus von Bellman-Ford
    - Algorithmus von Floyd-Warshall
  - Heute: Flüsse

Wiederholung

- Graphen
  - Grundlagen
    - Definition und Darstellung
    - Tiefen- und Breitensuche
  - Topologisches Sortieren
  - Starke Zusammenhangskomponenten
  - Minimale Spannbäume
    - Definition
    - Algorithmus von Kruskal
    - Algorithmus von Prim
  - Bestimmen der kürzesten Pfade
    - Definition und Varianten
    - (Breitensuche)
    - Algorithmus von Dijkstra
    - Algorithmus von Bellman-Ford
    - Algorithmus von Floyd-Warshall
  - Heute: Flüsse

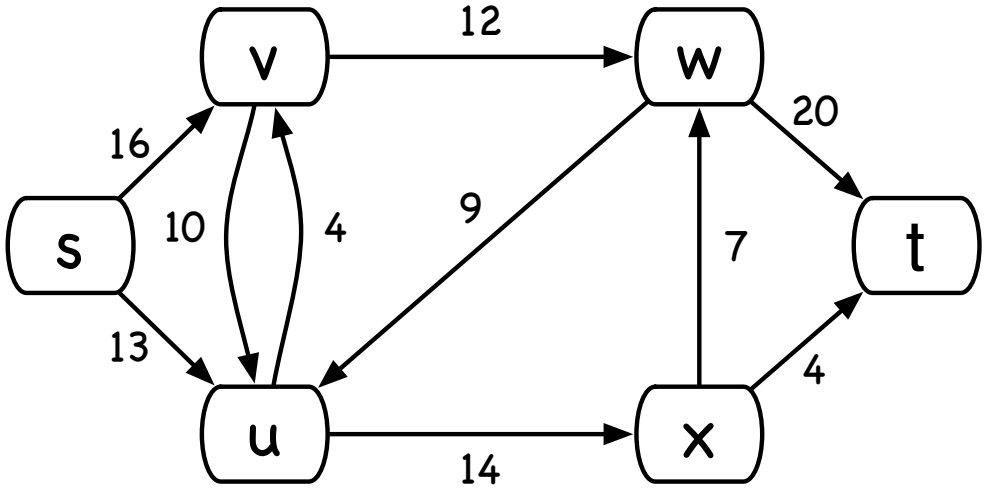
Wiederholung

- Graphen
  - Grundlagen
    - Definition und Darstellung
    - Tiefen- und Breitensuche
  - Topologisches Sortieren
  - Starke Zusammenhangskomponenten
  - Minimale Spannbäume
    - Definition
    - Algorithmus von Kruskal
    - Algorithmus von Prim
  - Bestimmen der kürzesten Pfade
    - Definition und Varianten
    - (Breitensuche)
    - Algorithmus von Dijkstra
    - Algorithmus von Bellman-Ford
    - Algorithmus von Floyd-Warshall
  - Heute: Flüsse

Wiederholung

- Graphen
  - Grundlagen
    - Definition und Darstellung
    - Tiefen- und Breitensuche
  - Topologisches Sortieren
  - Starke Zusammenhangskomponenten
  - Minimale Spannbäume
    - Definition
    - Algorithmus von Kruskal
    - Algorithmus von Prim
  - Bestimmen der kürzesten Pfade
    - Definition und Varianten
    - (Breitensuche)
    - Algorithmus von Dijkstra
    - Algorithmus von Bellman-Ford
    - Algorithmus von Floyd-Warshall
  - Heute: Flüsse

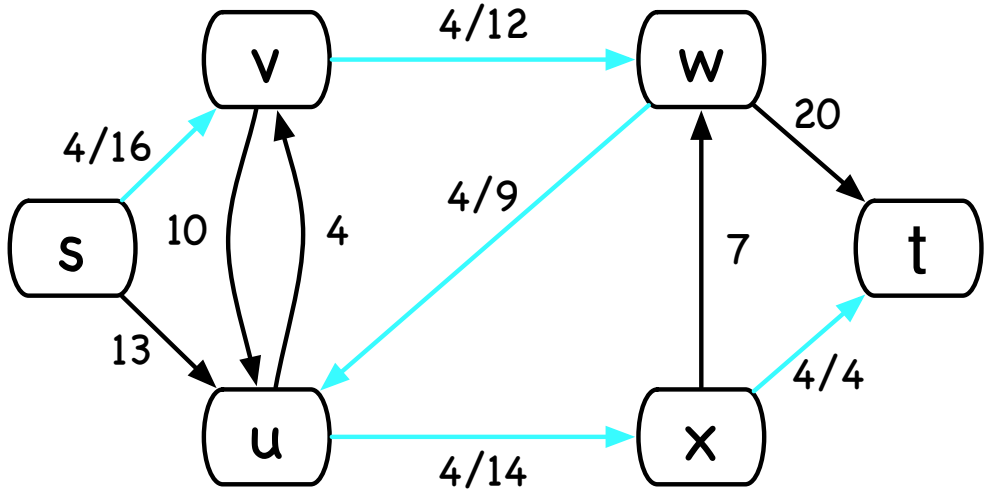
Motivation



Einen Graphen als 'Flussnetzwerk' zu interpretieren ist eine andere (neue) Art der Modellierung mit Graphen.

- $\Rightarrow$ Materialflüsse
- $\Rightarrow$ Logistik (Transportflüsse)
- $\Rightarrow$ Informationsflüsse
- $\Rightarrow$ ...

Begriffe und Definitionen



Definition (Flussnetzwerk)

Ein *Flussnetzwerk* ist ein gerichteter Graph $G = (V, E)$, in dem jeder Kante $(u, v) \in E$ zusätzlich eine nichtnegative *Kapazität* $c(u, v) \geq 0$ zugewiesen wird. (Ist $(u, v) \notin E$, so nehmen wir $c(u, v) = 0$ an.)
Es gibt die besonderen Knoten $s, t \in V$, die *Quelle* (s) und die *Senke* (t). Alle Knoten liegen auf Pfaden von der Quelle zur Senke.

Begriffe und Definitionen

Definition (Fluss)

Sei $G = (V, E)$ mit Kapazitätsfunktion c ein Flussnetzwerk, s und t wie oben. Ein *Fluss* in G ist eine Funktion $f : V \times V \to \mathbb{R}$, für die gilt:

1. Kapazitätsbeschränkung: $f(u, v) \leq c(u, v) \quad \forall u, v \in V$
2. Asymmetrie: $f(u, v) = -f(v, u) \quad \forall u, v \in V$
3. Flusserhaltung:

$$\sum_{v \in V} f(u, v) = 0 \quad \forall u \in V \setminus \{s, t\}$$

Das Problem des maximalen Flusses

Definition (Problem des maximalen Flusses)

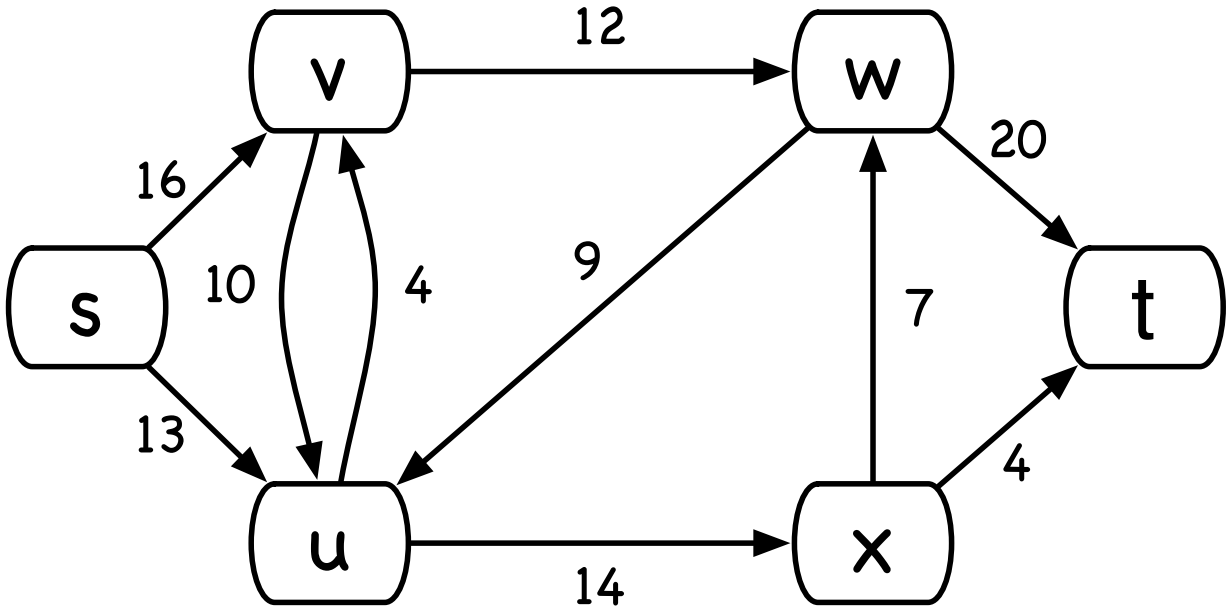
Eingabe: Ein Flussnetzwerk G mit Kapazitätsfunktion c und Quelle s und Senke t.
Gesucht: Ein Fluss mit maximalen Wert. Dabei ist der *Wert eines Flusses* definiert durch

$$|f| = \sum_{v \in V} f(s, v).$$

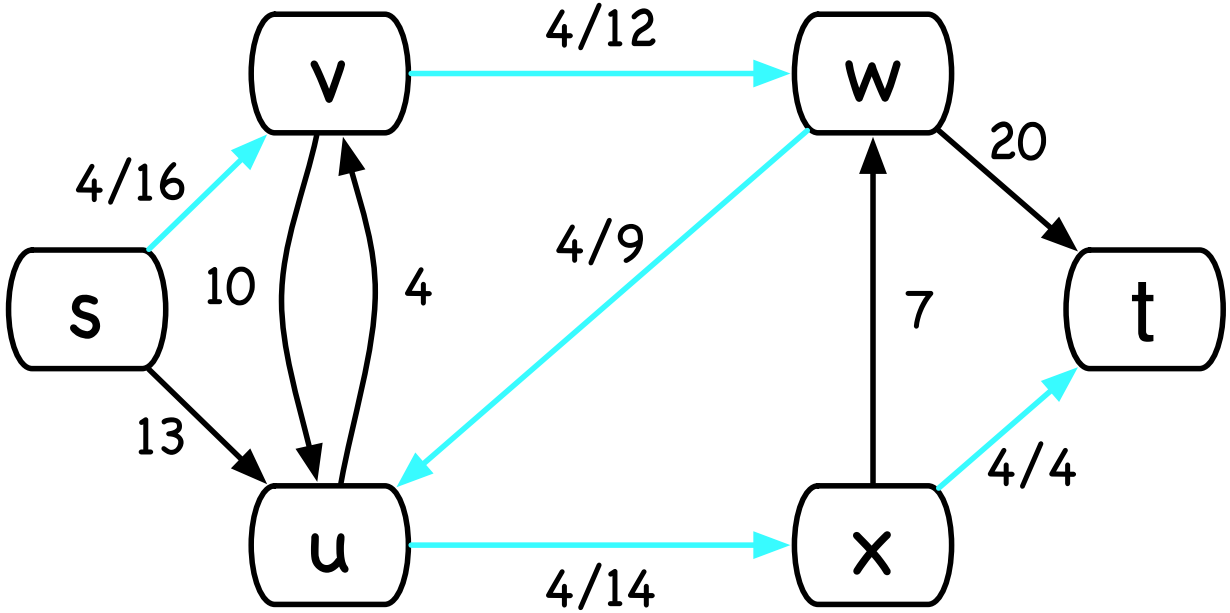
Anmerkung

Dieses Problem tritt bei den oben angesprochenen Anwendungen (Logistik, Material-/Informationsflüsse, ...) auf.

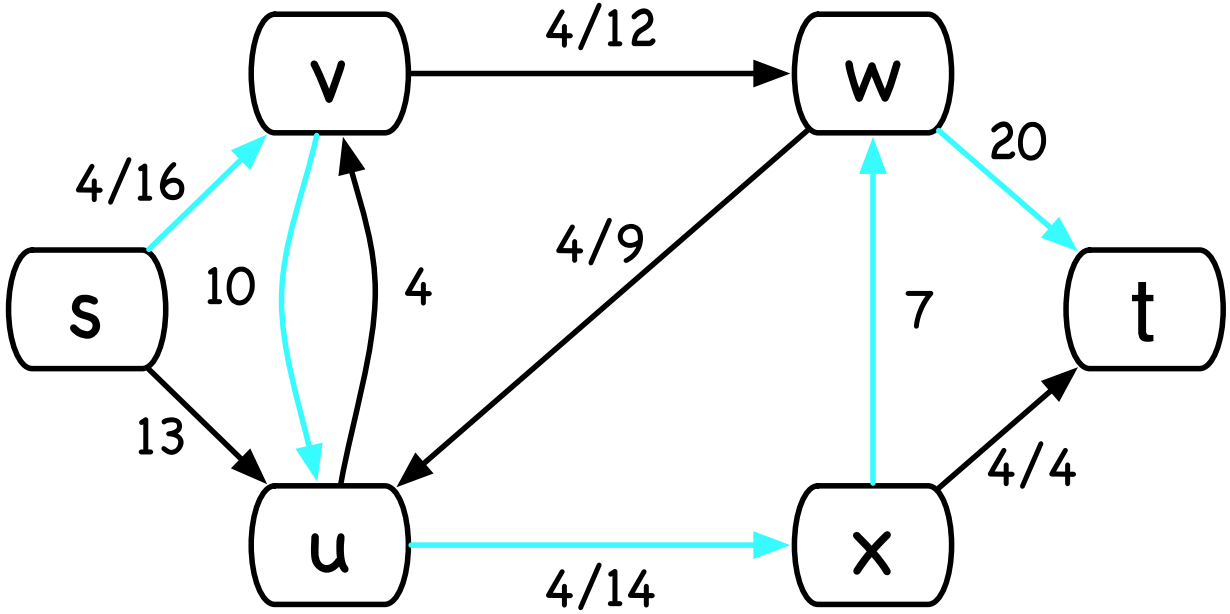
Maximaler Fluss - Beispiel



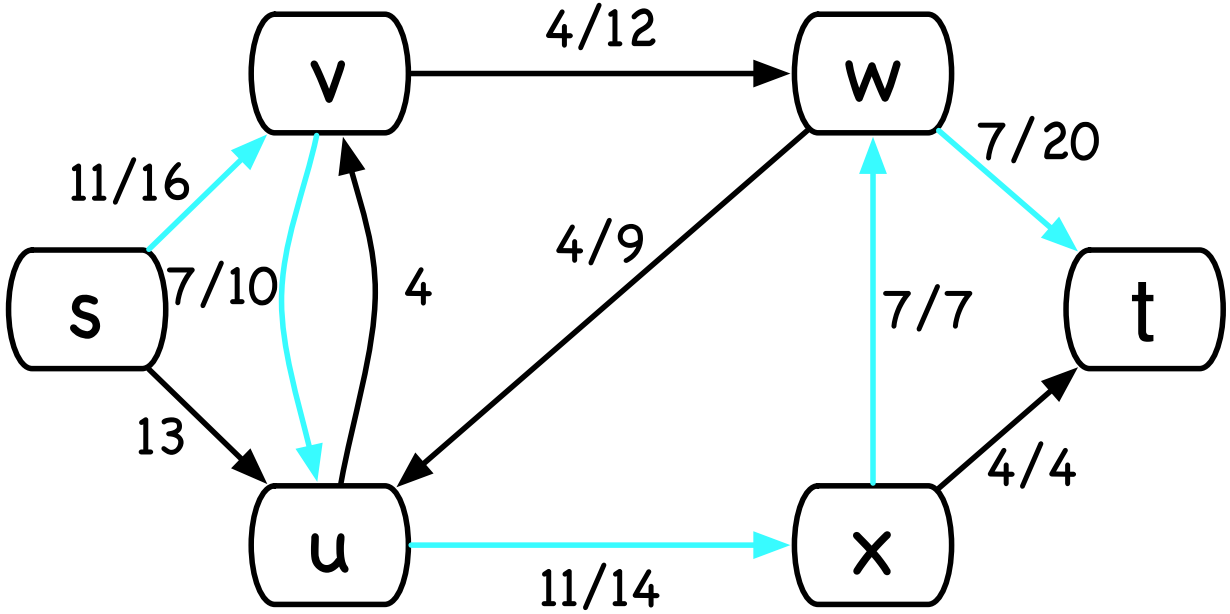
Maximaler Fluss - Beispiel



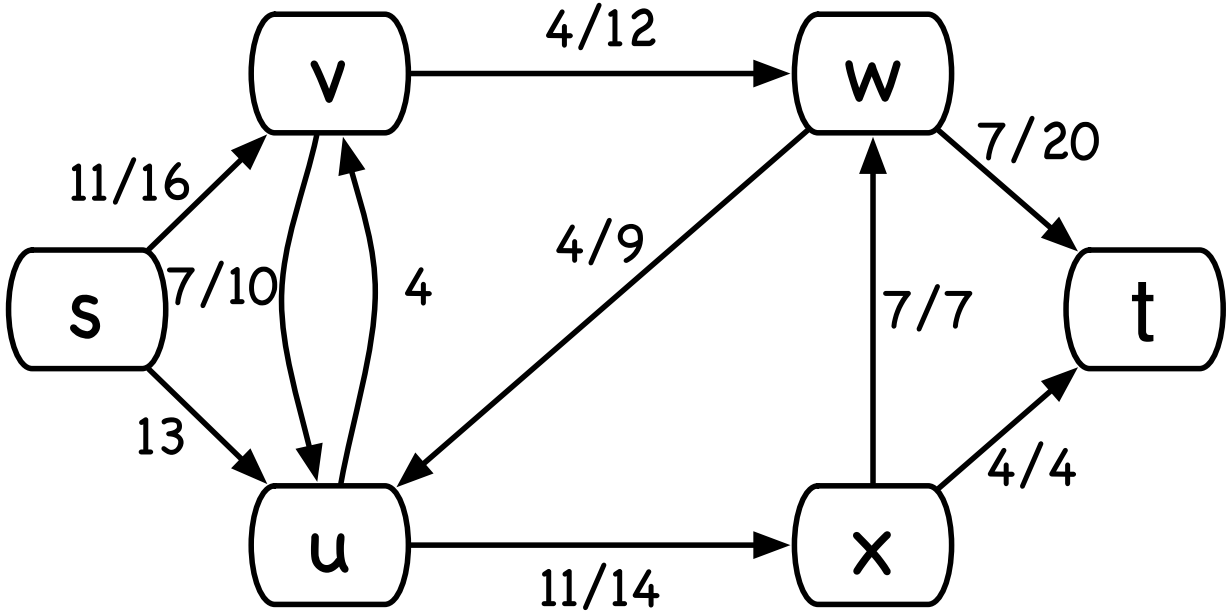
Maximaler Fluss - Beispiel



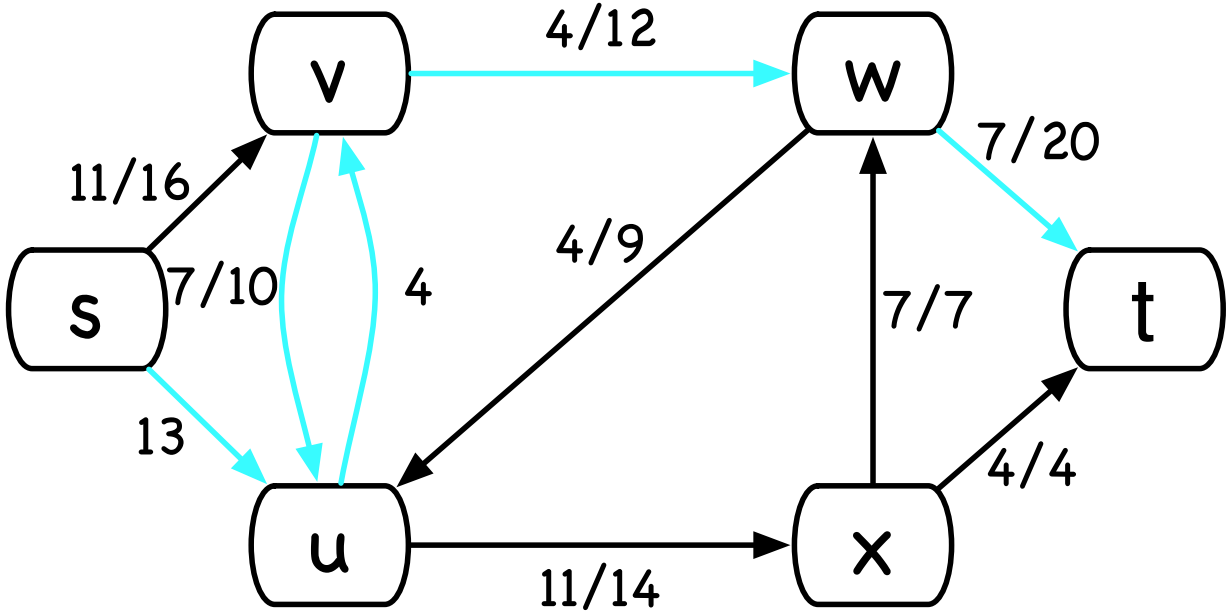
Maximaler Fluss - Beispiel



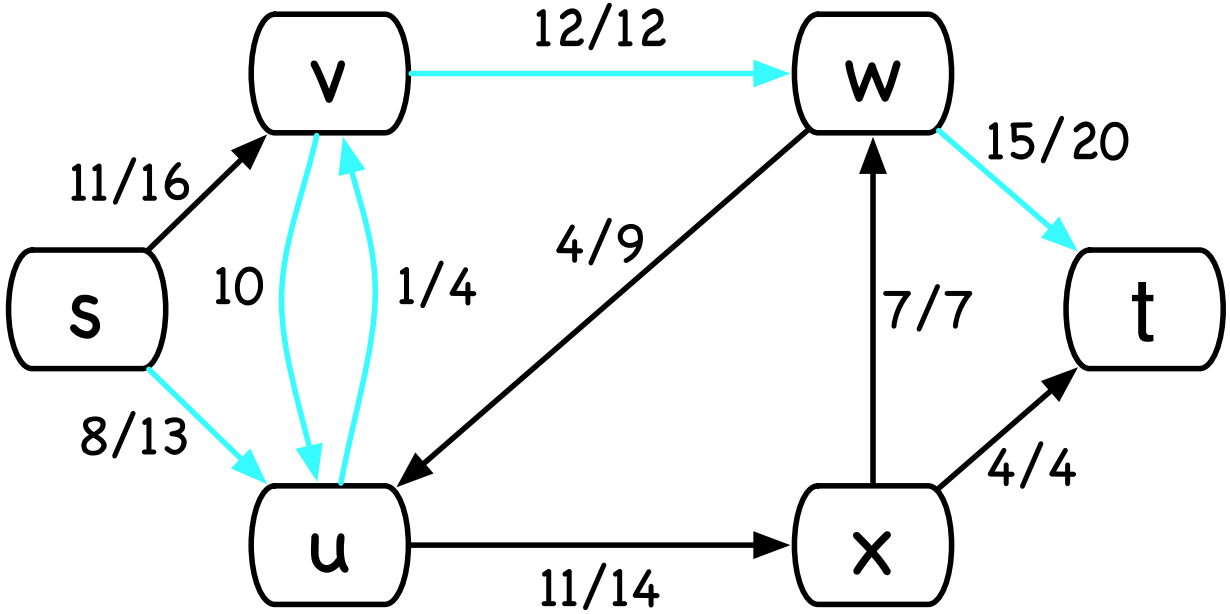
Maximaler Fluss - Beispiel



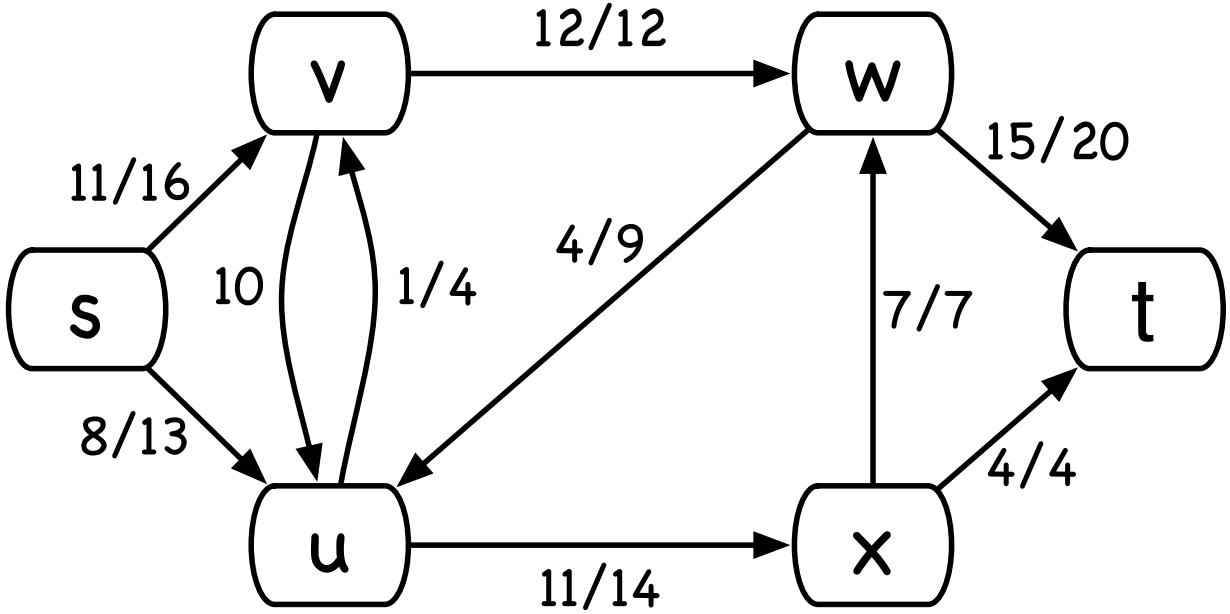
Maximaler Fluss - Beispiel



Maximaler Fluss - Beispiel



Maximaler Fluss - Beispiel



Flusseigenschaften

Satz

1. *Ist* $(u, v), (v, u) \notin E$, *so gilt* $f(u, v) = f(v, u) = 0$.
2. *Sei*

$$\sum_{\substack{u \in V \\ f(u,v) > 0}} f(u, v)$$

der gesamte positive Fluss in einen Knoten v *und analog der gesamte positive Fluss aus einem Knoten, dann ist der gesamte positive Fluss in einen Knoten* $v \in V \setminus \{s, t\}$ *gleich dem positiven Fluss aus diesem Knoten. ('Eintretender Fluss ist gleich austretender Fluss.')*

Beweis.

Mündlich / Zur Übung... □

Flusseigenschaften

Satz

1. *Ist* $(u, v), (v, u) \notin E$, *so gilt* $f(u, v) = f(v, u) = 0$.
2. *Sei*

$$\sum_{\substack{u \in V \\ f(u,v) > 0}} f(u, v)$$

der gesamte positive Fluss in einen Knoten v *und analog der gesamte positive Fluss aus einem Knoten, dann ist der gesamte positive Fluss in einen Knoten* $v \in V \setminus \{s, t\}$ *gleich dem positiven Fluss aus diesem Knoten. ('Eintretender Fluss ist gleich austretender Fluss.')*

Beweis.

Mündlich / Zur Übung... □

Flusseigenschaften

Satz

1. *Ist* $(u, v), (v, u) \notin E$, *so gilt* $f(u, v) = f(v, u) = 0$.
2. *Sei*

$$\sum_{\substack{u \in V \\ f(u,v) > 0}} f(u, v)$$

der gesamte positive Fluss in einen Knoten v *und analog der gesamte positive Fluss aus einem Knoten, dann ist der gesamte positive Fluss in einen Knoten* $v \in V \setminus \{s, t\}$ *gleich dem positiven Fluss aus diesem Knoten. ('Eintretender Fluss ist gleich austretender Fluss.')*

Beweis.

Mündlich / Zur Übung... □

Implizite Summennotation

f hat zwei Knoten als Argumente. Wir führen eine *implizite Summennotation* ein, um auch mit Mengen von Knoten arbeiten zu können. Wir setzen

$$f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y).$$

Die Flusserhaltung lässt sich so formulieren als

$$f(u, V) = 0 \quad \forall u \in V - \{s, t\}.$$

Ferner schreiben wir in der impliziten Summennotation auch z.B. $V - s$ statt $V - \{s\}$. Beispiel: $f(s, V - s) = f(s, V)$

Implizite Summennotation

f hat zwei Knoten als Argumente. Wir führen eine *implizite Summennotation* ein, um auch mit Mengen von Knoten arbeiten zu können. Wir setzen

$$f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y).$$

Die Flusserhaltung lässt sich so formulieren als

$$f(u, V) = 0 \quad \forall u \in V - \{s, t\}.$$

Ferner schreiben wir in der impliziten Summennotation auch z.B. $V - s$ statt $V - \{s\}$. Beispiel: $f(s, V - s) = f(s, V)$

Implizite Summennotation

f hat zwei Knoten als Argumente. Wir führen eine *implizite Summennotation* ein, um auch mit Mengen von Knoten arbeiten zu können. Wir setzen

$$f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y).$$

Die Flusserhaltung lässt sich so formulieren als

$$f(u, V) = 0 \quad \forall u \in V - \{s, t\}.$$

Ferner schreiben wir in der impliziten Summennotation auch z.B. $V - s$ statt $V - \{s\}$. Beispiel: $f(s, V - s) = f(s, V)$

Implizite Summennotation

f hat zwei Knoten als Argumente. Wir führen eine *implizite Summennotation* ein, um auch mit Mengen von Knoten arbeiten zu können. Wir setzen

$$f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y).$$

Die Flusserhaltung lässt sich so formulieren als

$$f(u, V) = 0 \quad \forall u \in V - \{s, t\}.$$

Ferner schreiben wir in der impliziten Summennotation auch z.B. $V - s$ statt $V - \{s\}$. Beispiel: $f(s, V - s) = f(s, V)$

Implizite Summennotation

f hat zwei Knoten als Argumente. Wir führen eine *implizite Summennotation* ein, um auch mit Mengen von Knoten arbeiten zu können. Wir setzen

$$f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y).$$

Die Flusserhaltung lässt sich so formulieren als

$$f(u, V) = 0 \quad \forall u \in V - \{s, t\}.$$

Ferner schreiben wir in der impliziten Summennotation auch z.B. $V - s$ statt $V - \{s\}$. Beispiel: $f(s, V - s) = f(s, V)$

Ein wichtiges Lemma / Weitere Flusseigenschaften

Satz (Lemma 26.1 im (Cormen))

Sei $G = (V, E)$ mit c ein Flussnetzwerk und f ein Fluss in G. Es gilt

1. $f(X, X) = 0 \quad \forall X \subseteq V$
2. $f(X, Y) = -f(Y, X) \quad \forall X, Y \subseteq V$
3. $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$ *und* $f(Z, X \cup Y) = f(Z, X) + f(Z, Y) \quad \forall X, Y, Z \subseteq V$ *mit* $X \cap Y = \emptyset$

Ein wichtiges Lemma / Weitere Flusseigenschaften

Satz (Lemma 26.1 im (Cormen))

Sei $G = (V, E)$ mit c ein Flussnetzwerk und f ein Fluss in G. Es gilt

2. $f(X, Y) = -f(Y, X) \quad \forall X, Y \subseteq V$

Beweis.

$$f(X, Y) = $$

$$= $$

$$= $$

□

Ein wichtiges Lemma / Weitere Flusseigenschaften

Satz (Lemma 26.1 im (Cormen))

Sei $G = (V, E)$ mit c ein Flussnetzwerk und f ein Fluss in G. Es gilt

2. $f(X, Y) = -f(Y, X) \quad \forall X, Y \subseteq V$

Beweis.

$$f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y)$$

$$=$$

$$=$$

□

Ein wichtiges Lemma / Weitere Flusseigenschaften

Satz (Lemma 26.1 im (Cormen))

Sei $G = (V, E)$ mit c ein Flussnetzwerk und f ein Fluss in G. Es gilt

2. $f(X, Y) = -f(Y, X) \quad \forall X, Y \subseteq V$

Beweis.

$$\begin{aligned} f(X, Y) &= \sum_{x \in X} \sum_{y \in Y} f(x, y) \\ &= \sum_{x \in X} \sum_{y \in Y} -f(y, x) \\ &= \end{aligned}$$

□

Ein wichtiges Lemma / Weitere Flusseigenschaften

Satz (Lemma 26.1 im (Cormen))

Sei $G = (V, E)$ mit c ein Flussnetzwerk und f ein Fluss in G. Es gilt

2. $f(X, Y) = -f(Y, X) \quad \forall X, Y \subseteq V$

Beweis.

$$
\begin{aligned}
f(X, Y) &= \sum_{x \in X} \sum_{y \in Y} f(x, y) \\
&= \sum_{x \in X} \sum_{y \in Y} -f(y, x) = -\sum_{x \in X} \sum_{y \in Y} f(y, x) \\
&=
\end{aligned}
$$

□

Ein wichtiges Lemma / Weitere Flusseigenschaften

Satz (Lemma 26.1 im (Cormen))

Sei $G = (V, E)$ mit c ein Flussnetzwerk und f ein Fluss in G. Es gilt

2. $f(X, Y) = -f(Y, X) \quad \forall X, Y \subseteq V$

Beweis.

$$
\begin{aligned}
f(X, Y) &= \sum_{x \in X} \sum_{y \in Y} f(x, y) \\
&= \sum_{x \in X} \sum_{y \in Y} -f(y, x) = -\sum_{x \in X} \sum_{y \in Y} f(y, x) \\
&= -\sum_{y \in Y} \sum_{x \in X} f(y, x)
\end{aligned}
$$

□

Ein wichtiges Lemma / Weitere Flusseigenschaften

Satz (Lemma 26.1 im (Cormen))

Sei $G = (V, E)$ mit c ein Flussnetzwerk und f ein Fluss in G. Es gilt

2. $f(X, Y) = -f(Y, X) \quad \forall X, Y \subseteq V$

Beweis.

$$\begin{aligned} f(X,Y) &= \sum_{x \in X} \sum_{y \in Y} f(x,y) \\ &= \sum_{x \in X} \sum_{y \in Y} -f(y,x) = -\sum_{x \in X} \sum_{y \in Y} f(y,x) \\ &= -\sum_{y \in Y} \sum_{x \in X} f(y,x) = -f(Y,X) \end{aligned}$$

□

Ein wichtiges Lemma / Weitere Flusseigenschaften

Satz (Lemma 26.1 im (Cormen))

Sei $G = (V, E)$ mit c ein Flussnetzwerk und f ein Fluss in G. Es gilt

1. $f(X, X) = 0 \quad \forall X \subseteq V$
3. $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$ *und* $f(Z, X \cup Y) = f(Z, X) + f(Z, Y) \quad \forall X, Y, Z \subseteq V$ *mit* $X \cap Y = \emptyset$

Beweis.

Mündlich / Zur Übung... □

Nutzung des Lemmas

Satz

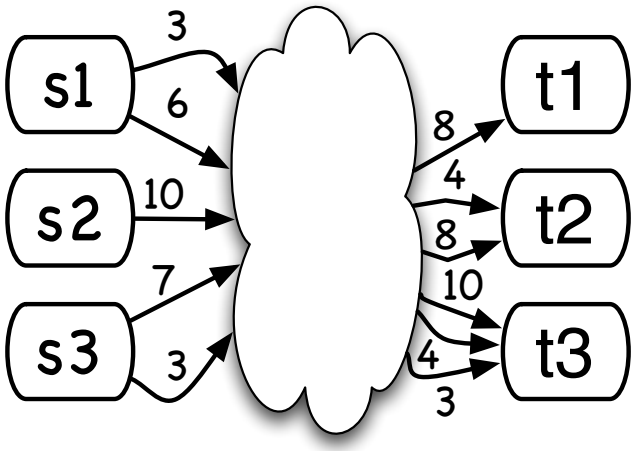
$|f| = f(V, t)$

Beweis.

$$\begin{aligned} |f| &= f(s, V) \\ &= f(V, V) - f(V - s, V) \\ &= -f(V - s, V) \\ &= f(V, V - s) \\ &= f(V, t) + f(V, V - s - t) \\ &= f(V, t) \end{aligned}$$

□

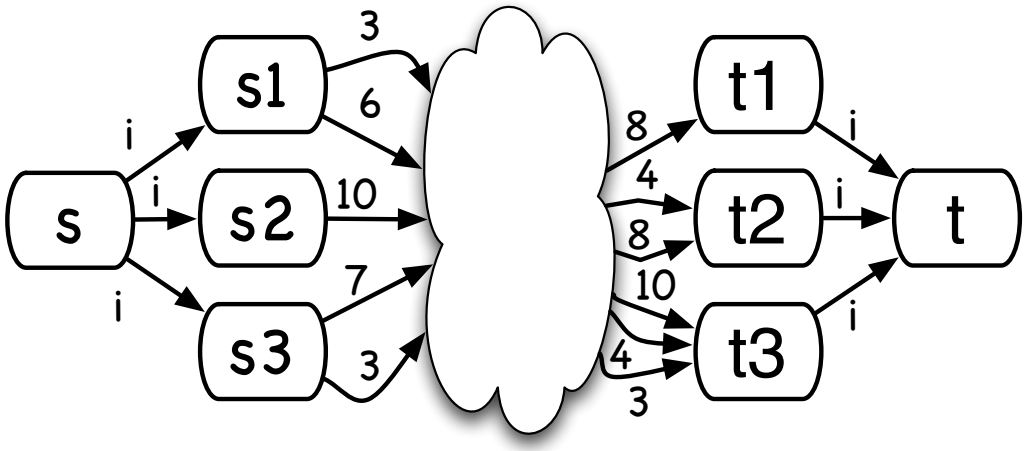
Exkurs: Multiple Quellen und Senken



Anmerkung

Das Problem mit mehreren Quellen und mehreren Senken kann auf den Fall einer Quelle und einer Senke zurückgeführt werden, indem eine *Superquelle* und eine *Supersenke* eingeführt wird.

Exkurs: Multiple Quellen und Senken



Anmerkung

Das Problem mit mehreren Quellen und mehreren Senken kann auf den Fall einer Quelle und einer Senke zurückgeführt werden, indem eine *Superquelle* und eine *Supersenke* eingeführt wird.

Inhaltsskizze

Die *Ford-Fulkerson-Methode* löst das Problem des maximalen Flusses. Sie umfasst verschieden Implementierungen (Algorithmen) mit unterschiedlichen Laufzeiten. Kern der Methode sind drei wichtie Ideen, die auch an anderer Stelle von Bedeutung sind:

- Restnetzwerke
- Erweiterungspfade
- Schnitte

Restnetzwerke: Definition

Definition

Sei $G = (V, E)$ mit c ein Flussnetzwerk und f ein Fluss in G. Wir definieren zu je zwei Knoten u, v die *Restkapazität* durch

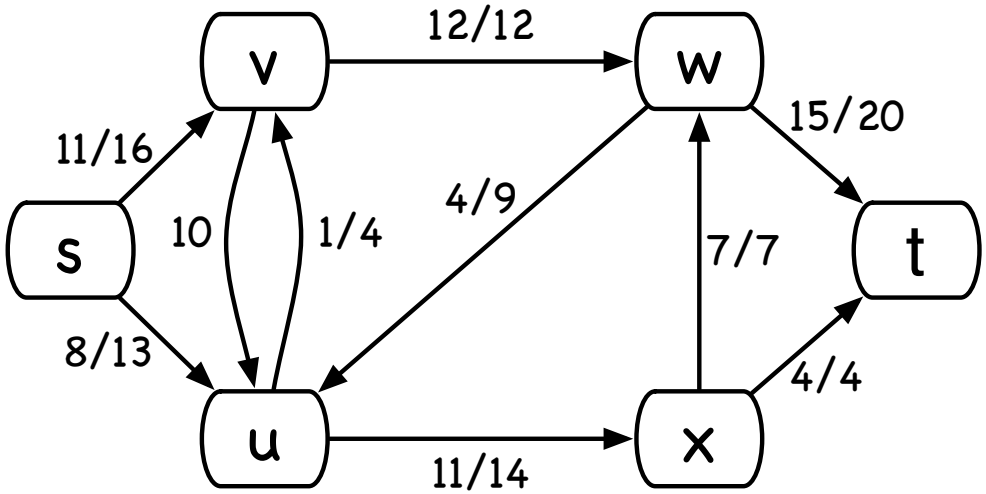
$$c_f(u, v) = c(u, v) - f(u, v).$$

Das Restnetzwerk ist dann gegeben durch $G_f = (V, E_f)$ und c_f, wobei

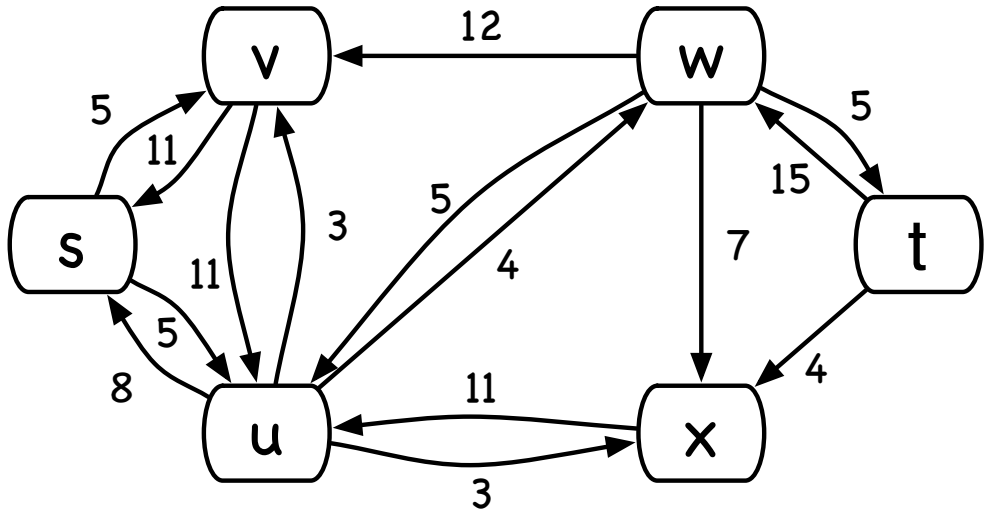
$$E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}$$

die Menge der *Restkanten* ist.
Das Restnetzwerk besteht also aus jenen Kanten, die noch mehr Fluss aufnehmen können.

Restnetzwerke: Beispiel



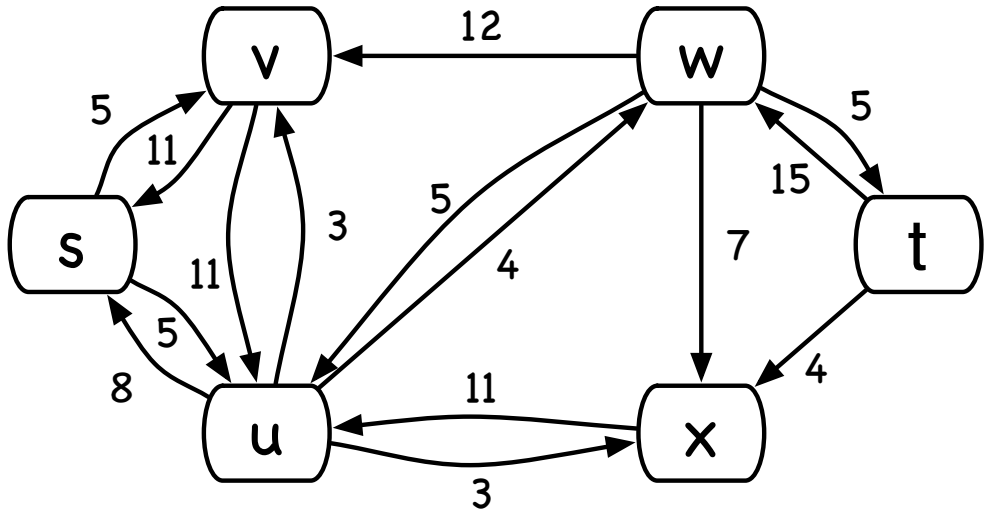
Restnetzwerke: Beispiel



Anmerkung

Man beachte, dass mit $f(u, v) = x$ dann $f(v, u) = -x$ gilt, was ggf. zu einer weiteren Kante im Restnetzwerk führt!

Restnetzwerke: Beispiel



Anmerkung

Man beachte, dass mit $f(u, v) = x$ dann $f(v, u) = -x$ gilt, was ggf. zu einer weiteren Kante im Restnetzwerk führt!

Restnetzwerke: Eigenschaften

Satz (Lemma 26.2 im (Cormen))

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk und f ein Fluss in G. G_f sei das durch den Fluss induzierte Restnetzwerk und f' ein Fluss in G_f. Dann ist die Flusssumme $f + f'$ ein Fluss in G mit $|f + f'| = |f| + |f'|$.

Anmerkung

Die Flusssumme $f + f'$ ist definiert durch

$$(f + f')(u, v) = f(u, v) + f'(u, v).$$

Restnetzwerke: Eigenschaften

Satz (Lemma 26.2 im (Cormen))

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk und f ein Fluss in G. G_f sei das durch den Fluss induzierte Restnetzwerk und f' ein Fluss in G_f. Dann ist die Flusssumme $f + f'$ ein Fluss in G mit $|f + f'| = |f| + |f'|$.

Beweis

Zu zeigen ist, dass $f + f'$ die drei Eigenschaften eines Flusses hat (Kapazitätsbeschränkung, Asymmetrie, Flusserhaltung) und dass $|f + f'| = |f| + |f'|$ gilt. Wir zeigen die ersten beiden Eigenschaften. Die dritte und vierte zur Übung/zum Nachlesen. ...

Restnetzwerke: Eigenschaften

Satz (Lemma 26.2 im (Cormen))

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk und f ein Fluss in G. G_f sei das durch den Fluss induzierte Restnetzwerk und f' ein Fluss in G_f. Dann ist die Flusssumme $f + f'$ ein Fluss in G mit $|f + f'| = |f| + |f'|$.

Beweis

Zu zeigen ist, dass $f + f'$ die drei Eigenschaften eines Flusses hat (Kapazitätsbeschränkung, Asymmetrie, Flusserhaltung) und dass $|f + f'| = |f| + |f'|$ gilt. Wir zeigen die ersten beiden Eigenschaften. Die dritte und vierte zur Übung/zum Nachlesen. ...

Restnetzwerke: Eigenschaften

Satz (Lemma 26.2 im (Cormen))

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk und f ein Fluss in G. G_f sei das durch den Fluss induzierte Restnetzwerk und f' ein Fluss in G_f. Dann ist die Flusssumme $f + f'$ ein Fluss in G mit $|f + f'| = |f| + |f'|$.

Beweis

Kapazitätsbeschränkung: Wir müssen $(f' + f)(u, v) \leq c(u, v)$ zeigen. Mit $f'(u, v) \leq c_f(u, v)$ (da f' ein Fluss in G_f ist) und $c_f(u, v) = c(u, v) - f(u, v)$ (nach Definition) folgt $(f + f')(u, v) = f(u, v) + f'(u, v) \leq f(u, v) + c(u, v) - f(u, v) = c(u, v)$. ...

Restnetzwerke: Eigenschaften

Satz (Lemma 26.2 im (Cormen))

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk und f ein Fluss in G. G_f sei das durch den Fluss induzierte Restnetzwerk und f' ein Fluss in G_f. Dann ist die Flusssumme $f + f'$ ein Fluss in G mit $|f + f'| = |f| + |f'|$.

Beweis

Kapazitätsbeschränkung: Wir müssen $(f' + f)(u, v) \leq c(u, v)$ zeigen. Mit $f'(u, v) \leq c_f(u, v)$ (da f' ein Fluss in G_f ist) und $c_f(u, v) = c(u, v) - f(u, v)$ (nach Definition) folgt $(f + f')(u, v) = f(u, v) + f'(u, v) \leq f(u, v) + c(u, v) - f(u, v) = c(u, v)$. ...

Restnetzwerke: Eigenschaften

Satz (Lemma 26.2 im (Cormen))

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk und f ein Fluss in G. G_f sei das durch den Fluss induzierte Restnetzwerk und f' ein Fluss in G_f. Dann ist die Flusssumme $f + f'$ ein Fluss in G mit $|f + f'| = |f| + |f'|$.

Beweis

Kapazitätsbeschränkung: Wir müssen $(f' + f)(u, v) \leq c(u, v)$ zeigen. Mit $f'(u, v) \leq c_f(u, v)$ (da f' ein Fluss in G_f ist) und $c_f(u, v) = c(u, v) - f(u, v)$ (nach Definition) folgt $(f + f')(u, v) = f(u, v) + f'(u, v) \leq f(u, v) + c(u, v) - f(u, v) = c(u, v)$. ...

Restnetzwerke: Eigenschaften

Satz (Lemma 26.2 im (Cormen))

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk und f ein Fluss in G. G_f sei das durch den Fluss induzierte Restnetzwerk und f' ein Fluss in G_f. Dann ist die Flusssumme $f + f'$ ein Fluss in G mit $|f + f'| = |f| + |f'|$.

Beweis

Asymmetrie: $(f + f')(u, v) = f(u, v) + f'(u, v) = -f(v, u) - f'(v, u) = -(f(v, u) + f'(v, u)) = -(f + f')(v, u)$. Wir nutzen hier lediglich die Asymmetrie der einzelnen Flüsse (und die Definition der Flusssumme). ...

Restnetzwerke: Eigenschaften

Satz (Lemma 26.2 im (Cormen))

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk und f ein Fluss in G. G_f sei das durch den Fluss induzierte Restnetzwerk und f' ein Fluss in G_f. Dann ist die Flusssumme $f + f'$ ein Fluss in G mit $|f + f'| = |f| + |f'|$.

Beweis

Asymmetrie: $(f + f')(u, v) = f(u, v) + f'(u, v) = -f(v, u) - f'(v, u) = -(f(v, u) + f'(v, u)) = -(f + f')(v, u)$. Wir nutzen hier lediglich die Asymmetrie der einzelnen Flüsse (und die Definition der Flusssumme). ...

Restnetzwerke: Eigenschaften

Satz (Lemma 26.2 im (Cormen))

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk und f ein Fluss in G. G_f sei das durch den Fluss induzierte Restnetzwerk und f' ein Fluss in G_f. Dann ist die Flusssumme $f + f'$ ein Fluss in G mit $|f + f'| = |f| + |f'|$.

Beweis.

Noch zu zeigen: $\sum_{v \in V}(f + f')(u, v) = 0$ (Flusserhaltung) und $|f + f'| = |f| + |f'|$. Mündlich / Zur Übung / Zum Nachlesen. □

Restnetzwerke: Zusammenfassung

- Intuitive Definition: Das Restnetzwerk zu einem gegebenen Flussnetzwerk *und* einem Fluss besteht aus jenen Kanten, die mehr Fluss aufnehmen können.
  - Formal sind die Restkapazität und die Restkanten wichtig. Man beachte, dass *neue Kanten* entstehen können (bzw. streng formal: Kanten, deren Kapazität vorher 0 war, kann nun größer als 0 werden.
- Wichtige Eigenschaft: Findet man in einem Restnetzwerk G_f (f ist ein Fluss in dem Flussnetzwerk G) einen weiteren Fluss f', so ist $f + f'$ erneut ein Fluss *in* G (!) mit dem Wert $|f| + |f'| (> |f|!)$.

Restnetzwerke: Zusammenfassung

- Intuitive Definition: Das Restnetzwerk zu einem gegebenen Flussnetzwerk *und* einem Fluss besteht aus jenen Kanten, die mehr Fluss aufnehmen können.
  - Formal sind die Restkapazität und die Restkanten wichtig. Man beachte, dass *neue Kanten* entstehen können (bzw. streng formal: Kanten, deren Kapazität vorher 0 war, kann nun größer als 0 werden.
- Wichtige Eigenschaft: Findet man in einem Restnetzwerk G_f (f ist ein Fluss in dem Flussnetzwerk G) einen weiteren Fluss f', so ist $f + f'$ erneut ein Fluss *in* G (!) mit dem Wert $|f| + |f'| (> |f|!)$.

Restnetzwerke: Zusammenfassung

- Intuitive Definition: Das Restnetzwerk zu einem gegebenen Flussnetzwerk *und* einem Fluss besteht aus jenen Kanten, die mehr Fluss aufnehmen können.
  - Formal sind die Restkapazität und die Restkanten wichtig. Man beachte, dass *neue Kanten* entstehen können (bzw. streng formal: Kanten, deren Kapazität vorher 0 war, kann nun größer als 0 werden.
- Wichtige Eigenschaft: Findet man in einem Restnetzwerk G_f (f ist ein Fluss in dem Flussnetzwerk G) einen weiteren Fluss f', so ist $f + f'$ erneut ein Fluss *in* G (!) mit dem Wert $|f| + |f'| (> |f|!)$.

Restnetzwerke: Zusammenfassung

- Intuitive Definition: Das Restnetzwerk zu einem gegebenen Flussnetzwerk *und* einem Fluss besteht aus jenen Kanten, die mehr Fluss aufnehmen können.
  - Formal sind die Restkapazität und die Restkanten wichtig. Man beachte, dass *neue Kanten* entstehen können (bzw. streng formal: Kanten, deren Kapazität vorher 0 war, kann nun größer als 0 werden.
- Wichtige Eigenschaft: Findet man in einem Restnetzwerk G_f (f ist ein Fluss in dem Flussnetzwerk G) einen weiteren Fluss f', so ist $f + f'$ erneut ein Fluss *in* G (!) mit dem Wert $|f| + |f'| (> |f|!)$.

Erweiterungspfade: Definition

Definition

Sei $G = (V, E)$ mit c ein Flussnetzwerk, f ein Fluss in G und G_f das Restnetzwerk. Ein *Erweiterungspfad* ist ein einfacher Pfad von s nach t in G_f.
Mit

$$c_f(p) = \min\{c_f(u, v) \mid (u, v) \text{ ist eine Kante von } p\}$$

wird die *Restkapazität* eines Erweiterungspfades p bezeichnet.

Pause to Ponder

Die Idee

Erweiterungspfade: Definition

Definition

Sei $G = (V, E)$ mit c ein Flussnetzwerk, f ein Fluss in G und G_f das Restnetzwerk. Ein *Erweiterungspfad* ist ein einfacher Pfad von s nach t in G_f.
Mit

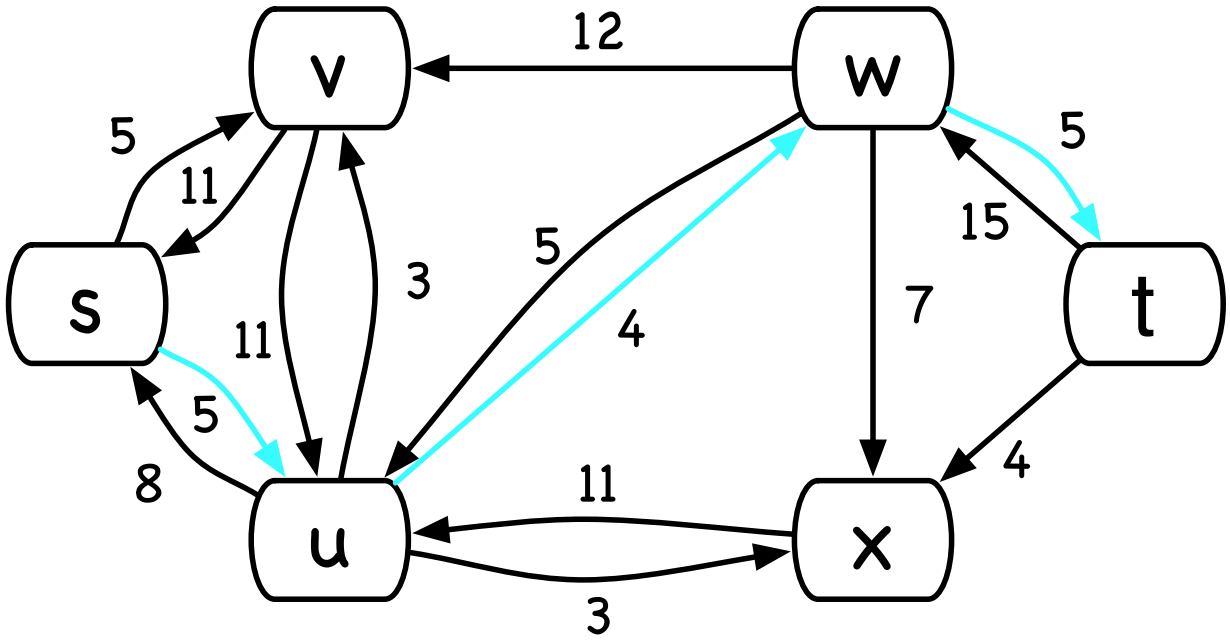
$$c_f(p) = \min\{c_f(u, v) \mid (u, v) \text{ ist eine Kante von } p\}$$

wird die *Restkapazität* eines Erweiterungspfades p bezeichnet.

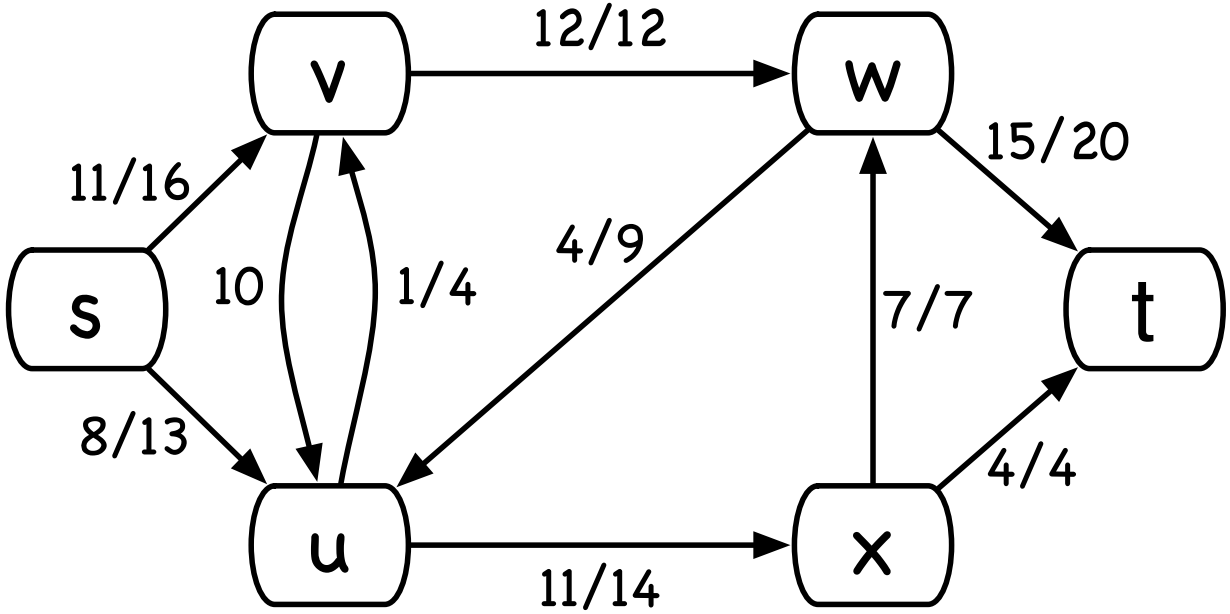
Pause to Ponder

Die Idee der Ford-Fulkerson-Methode: Bestimme so lange wie möglich Erweiterungspfade und erhöhe den Fluss jeweils um die Restkapazität.

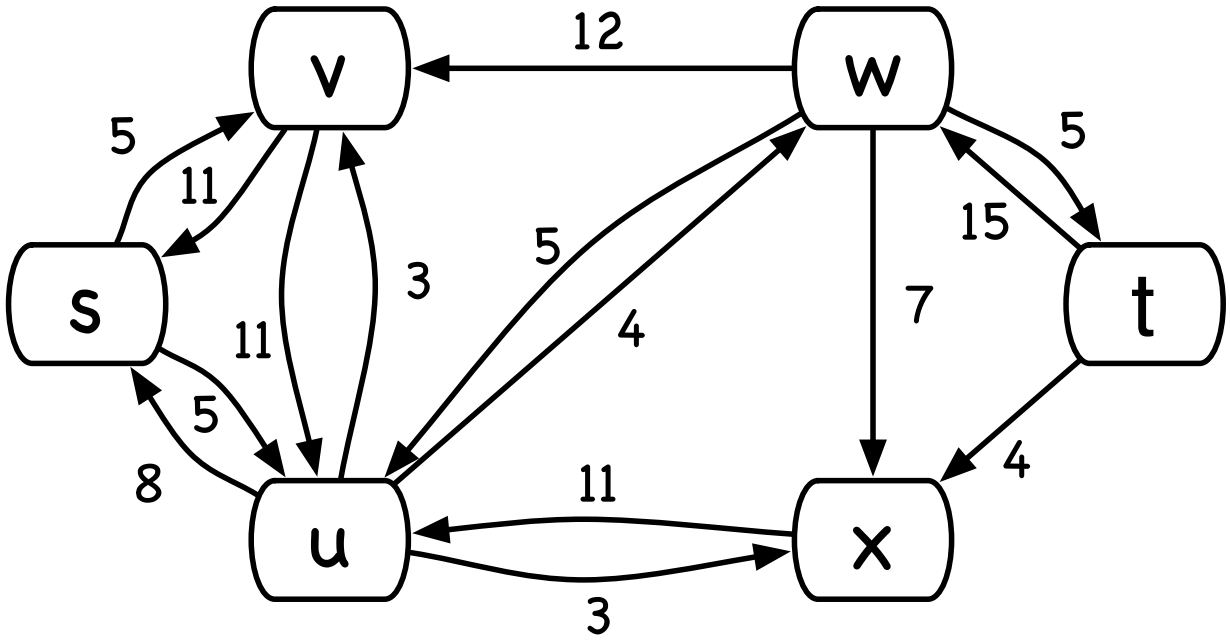
Erweiterungspfade: Beispiel



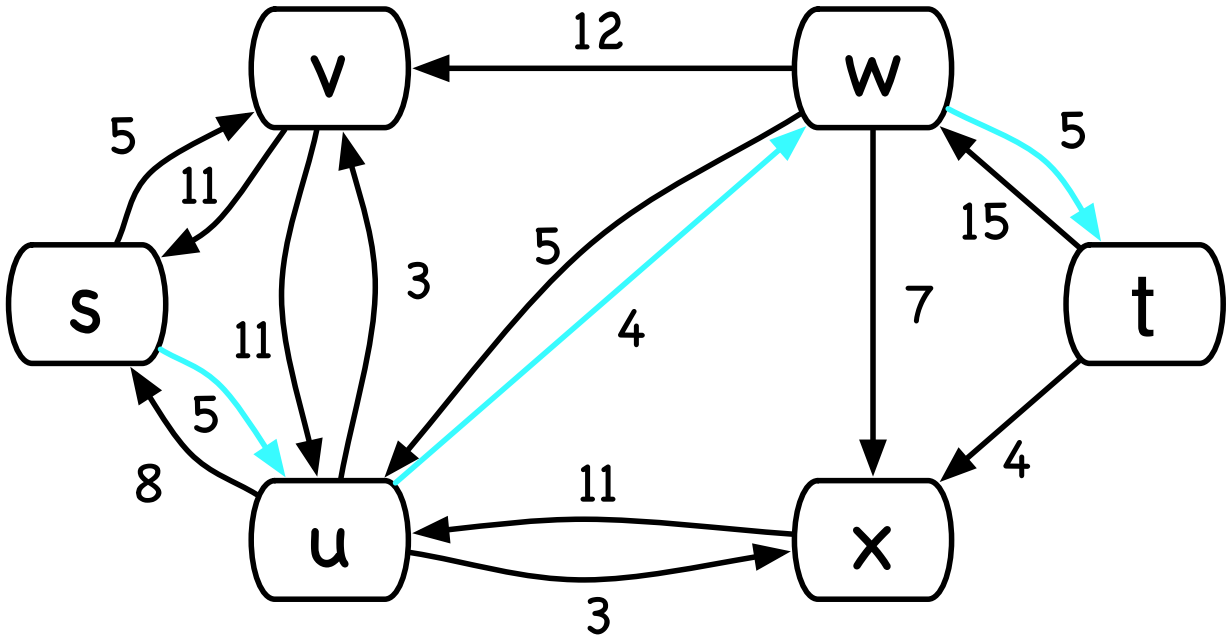
Erweiterungspfade: Beispiel



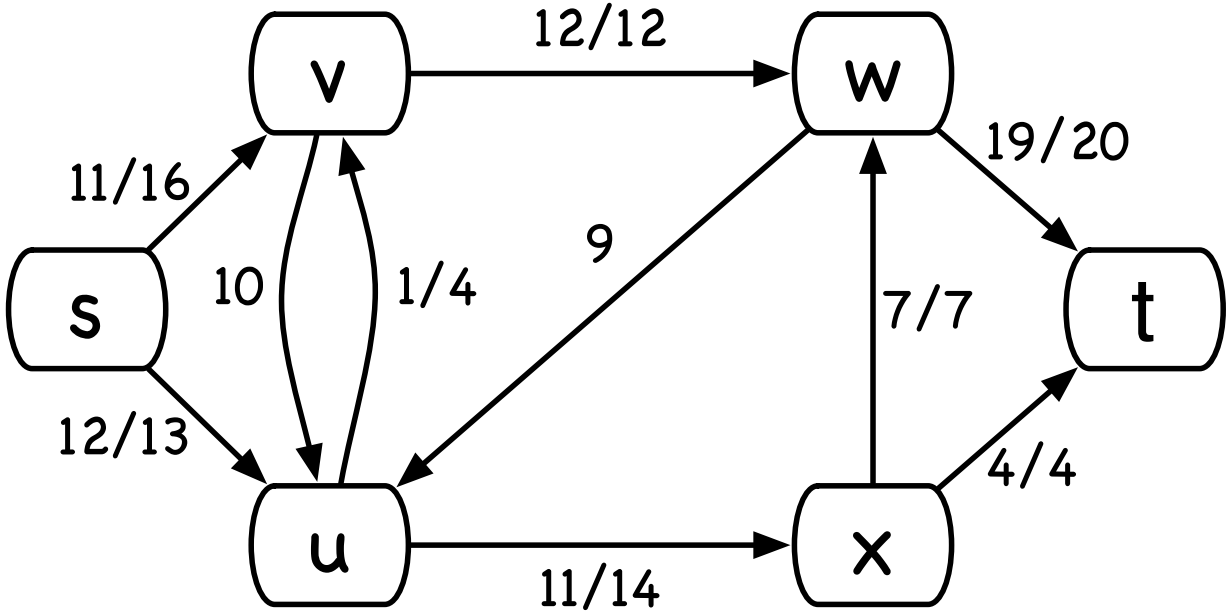
Erweiterungspfade: Beispiel



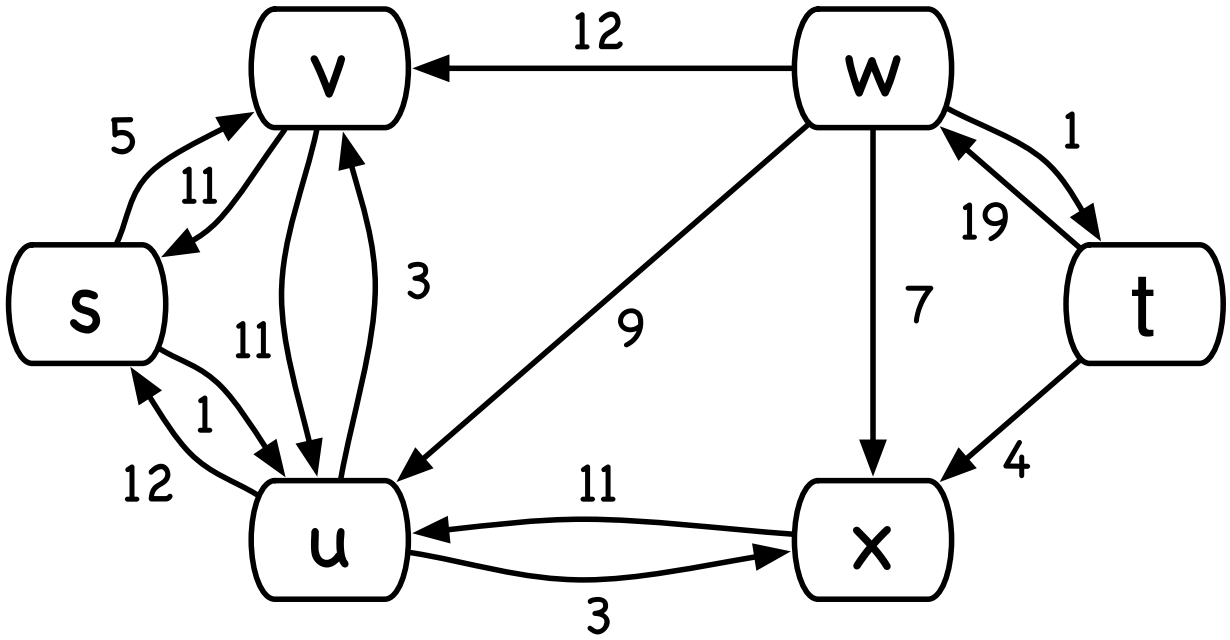
Erweiterungspfade: Beispiel



Erweiterungspfade: Beispiel



Erweiterungspfade: Beispiel



Erweiterungspfade: Eigenschaften

Satz (Lemma 26.3 im (Cormen))

Sei $G = (V, E)$ mit c ein Flussnetzwerk, f ein Fluss in G und p ein Erweiterungspfad in G_f. Sei $f_p : V \times V \to \mathbb{R}$ eine Funktion mit

$$f_p(u,v) = \begin{cases} c_f(p) & \text{falls } (u,v) \text{ zu } p \text{ gehört,} \\ -c_f(p) & \text{falls } (v,u) \text{ zu } p \text{ gehört,} \\ 0 & \text{sonst.} \end{cases}$$

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

Restnetzwerke: Eigenschaften (Wdh.)

Satz (Lemma 26.2 im (Cormen))

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk und f ein Fluss in G. G_f sei das durch den Fluss induzierte Restnetzwerk und f' ein Fluss in G_f. Dann ist die Flusssumme $f + f'$ ein Fluss in G mit $|f + f'| = |f| + |f'|$.

Erweiterungspfade: Eigenschaften

Satz (Korollar 26.4 im (Cormen))

Sei $G = (V, E)$ mit c ein Flussnetzwerk, f ein Fluss in G und p ein Erweiterungspfad in G_f. Sei f_p wie eben definiert und sei $f' = f + f_p$, dann ist f' ein Fluss in G mit $|f'| = |f| + |f_p| > |f|$.

Beweis.

Folgt sofort aus den obigen beiden Sätzen. □

Restnetzwerke und Erweiterungspfade: Zusammenfassung

- Intuitive Definition: Das *Restnetzwerk* zu einem gegebenen Flussnetzwerk *und* einem Fluss besteht aus jenen Kanten, die mehr Fluss aufnehmen können. Ein *Erweiterungspfad* ist ein einfacher Pfad von s nach t im Restnetzwerk.
- Findet man in einem Restnetzwerk G_f (f ist ein Fluss in dem Flussnetzwerk G) einen weiteren Fluss f', so ist $f + f'$ erneut ein Fluss in G mit dem Wert $|f| + |f'| > |f|$.
- Letztere gilt insb. auch, wenn man über einen Erweiterungspfad so viel wie möglich 'leitet' (Restkapazität).

Die Ford-Fulkerson-Methode

Die Ford-Fulkerson-Methode:

0. Sei $f = 0$.
1. Bestimme $G_f = (V, E_f)$ und c_f.
2. Bestimme einen Erweiterungspfad p. (Gibt es keinen, gehe zu 5.)
3. Erhöhe f entlang p um $c_f(p)$.
4. Gehe zu 1.
5. f ist ein maximaler Fluss.

Pause to Ponder

Terminiert dieses Verfahren? Ist f wirklich maximal?

Die Ford-Fulkerson-Methode

Die Ford-Fulkerson-Methode:

0. Sei $f = 0$.
1. Bestimme $G_f = (V, E_f)$ und c_f.
2. Bestimme einen Erweiterungspfad p. (Gibt es keinen, gehe zu 5.)
3. Erhöhe f entlang p um $c_f(p)$.
4. Gehe zu 1.
5. f ist ein maximaler Fluss.

Pause to Ponder

Terminiert dieses Verfahren? Ist f wirklich maximal?

Schnitte: Motivation

Das *maximaler-Fluss-minimaler-Schnitt-Theorem* besagt, dass ein Fluss maximal ist gdw. das Restnetzwerk keinen Erweiterungspfad enthält.

Dieses Theorem ist genau, was wir brauchen! Zum Beweis benötigen wir Schnitte...

Schnitte: Motivation

Das *maximaler-Fluss-minimaler-Schnitt-Theorem* besagt, dass ein Fluss maximal ist gdw. das Restnetzwerk keinen Erweiterungspfad enthält.

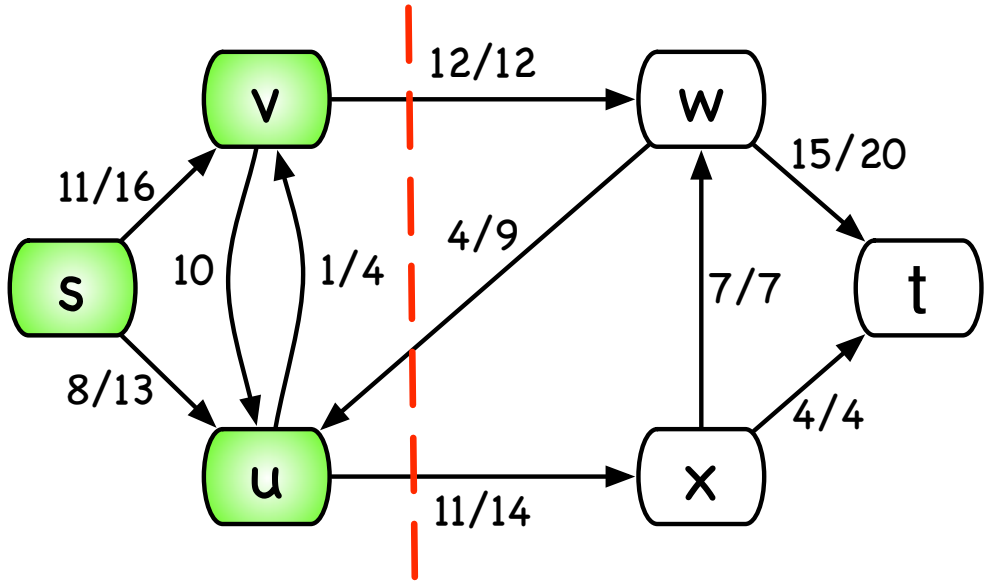
Dieses Theorem ist genau, was wir brauchen! Zum Beweis benötigen wir Schnitte...

Schnitte: Definition

Definition (Schnitt)

Ein *Schnitt* (S, T) eines Flussnetzwerkes $G = (V, E)$ ist eine Partionierung von V in S und $T = V - S$, wobei $s \in S$ und $t \in T$ gilt.
Der *Nettofluss* und die *Kapazität* eines Schnittes (S, T) ist durch $f(S, T)$ bzw. $c(S, T)$ definiert.
Ein *minimaler Schnitt* ist ein Schnitt, dessen Kapazität die kleinste von allen Schnitten des Netzwerkes ist.

Schnitte: Beispiel



Anmerkung

Beim Nettofluss wird ein (positiver) Fluss von S nach T addiert, ein (positiver) Fluss von T nach S subtrahiert. Bei der Kapazität sind nur Kanten von S nach T relevant.

Schnitte: Eigenschaften

Satz (Lemma 26.5 im (Cormen))

Seien G ein Flussnetzwerk, c, s, t wie üblich, f ein Fluss und (S, T) ein Schnitt von G. Dann gilt $f(S, T) = |f|$. (D.h. der Nettofluss durch beliebige Schnitte ist immer gleich, nämlich gleich dem Wert des Flusses.)

Beweis.

$f(S, T) = f(S, V) - f(S, S) = f(S, V) = f(s, V) + f(S - s, V) = f(s, V) = |f|$. □

Schnitte: Eigenschaften

Satz (Lemma 26.5 im (Cormen))

Seien G ein Flussnetzwerk, c, s, t wie üblich, f ein Fluss und (S, T) ein Schnitt von G. Dann gilt $f(S, T) = |f|$. (D.h. der Nettofluss durch beliebige Schnitte ist immer gleich, nämlich gleich dem Wert des Flusses.)

Beweis.

$f(S, T) = f(S, V) - f(S, S) = f(S, V) = f(s, V) + f(S - s, V) = f(s, V) = |f|$. □

Schnitte: Eigenschaften

Satz (Lemma 26.5 im (Cormen))

Seien G ein Flussnetzwerk, c, s, t wie üblich, f ein Fluss und (S, T) ein Schnitt von G. Dann gilt $f(S, T) = |f|$. (D.h. der Nettofluss durch beliebige Schnitte ist immer gleich, nämlich gleich dem Wert des Flusses.)

Beweis.

$f(S, T) = f(S, V) - f(S, S) = f(S, V) = f(s, V) + f(S - s, V) = f(s, V) = |f|$. □

Schnitte: Eigenschaften

Satz (Lemma 26.5 im (Cormen))

Seien G ein Flussnetzwerk, c, s, t wie üblich, f ein Fluss und (S, T) ein Schnitt von G. Dann gilt $f(S, T) = |f|$. (D.h. der Nettofluss durch beliebige Schnitte ist immer gleich, nämlich gleich dem Wert des Flusses.)

Beweis.

$f(S, T) = f(S, V) - f(S, S) = f(S, V) = f(s, V) + f(S - s, V) = f(s, V) = |f|$. □

Schnitte: Eigenschaften

Satz (Lemma 26.5 im (Cormen))

Seien G ein Flussnetzwerk, c, s, t wie üblich, f ein Fluss und (S, T) ein Schnitt von G. Dann gilt $f(S, T) = |f|$. (D.h. der Nettofluss durch beliebige Schnitte ist immer gleich, nämlich gleich dem Wert des Flusses.)

Beweis.

$f(S, T) = f(S, V) - f(S, S) = f(S, V) = f(s, V) + f(S - s, V) = f(s, V) = |f|$. □

Schnitte: Eigenschaften

Satz (Lemma 26.5 im (Cormen))

Seien G ein Flussnetzwerk, c, s, t wie üblich, f ein Fluss und (S, T) ein Schnitt von G. Dann gilt $f(S, T) = |f|$. (D.h. der Nettofluss durch beliebige Schnitte ist immer gleich, nämlich gleich dem Wert des Flusses.)

Beweis.

$f(S, T) = f(S, V) - f(S, S) = f(S, V) = f(s, V) + f(S - s, V) = f(s, V) = |f|$. □

Schnitte: Eigenschaften

Satz

In einem Flussnetzwerk G ist der Wert eines beliebigen Flusses durch die Kapzität jedes beliebigen Schnittes nach oben beschränkt.

Beweis.

Sei (S, T) ein Schnitt von G und f ein beliebiger Fluss. Wir wollen $|f| \leq c(S, T)$ zeigen.

$$\begin{aligned} |f| &= f(S, T) \\ &= \sum_{u \in S} \sum_{v \in T} f(u, v) \leq \sum_{u \in S} \sum_{v \in T} c(u, v) \\ &= c(S, T) \end{aligned}$$

Schnitte: Eigenschaften

Satz

In einem Flussnetzwerk G ist der Wert eines beliebigen Flusses durch die Kapzität jedes beliebigen Schnittes nach oben beschränkt.

Beweis.

Sei (S, T) ein Schnitt von G und f ein beliebiger Fluss. Wir wollen $|f| \leq c(S, T)$ zeigen.

$$\begin{aligned} |f| &= f(S, T) \\ &= \sum_{u \in S} \sum_{v \in T} f(u, v) \leq \sum_{u \in S} \sum_{v \in T} c(u, v) \\ &= c(S, T) \end{aligned}$$

□

Zum Max-Flow-Min-Cut-Theorem

Satz

In einem Flussnetzwerk G ist der Wert eines beliebigen Flusses durch die Kapzität jedes beliebigen Schnittes nach oben beschränkt.

Anmerkung

Der *maximale* Fluss muss folglich durch die Kapazität eines *minimalen* Schnitts beschränkt sein (da oben beide beliebig, aber die Schranke stets gilt). - Das wichtige Max-Flow-Min-Cut-Theorem sagt aus, dass diese Werte tatsächlich *gleich* sind.

Max-Flow-Min-Cut-Theorem

Satz

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk, f ein Fluss in G. Die folgenden Bedingungen sind äquivalent:

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

Beweis

(1) $\Rightarrow$ (2): Widerspruchsbeweis... mündlich/zur Übung/zum Nachlesen...

Max-Flow-Min-Cut-Theorem

Satz

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk, f ein Fluss in G. Die folgenden Bedingungen sind äquivalent:

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

Beweis

(1) $\Rightarrow$ (2): Widerspruchsbeweis... mündlich/zur Übung/zum Nachlesen...

Max-Flow-Min-Cut-Theorem

Satz

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

Beweis

(2) $\Rightarrow$ (3): G_f enthält keinen Pfad von s nach t. Definiere:

$$S := \{v \in V \mid \text{ es gibt einen Pfad in } G_f \text{ von } s \text{ nach } v\}$$

und $T := V - S$. (S, T) ist ein Schnitt (warum?), ferner muss $f(u, v) = c(u, v)$ für $u, v \in V$ mit $u \in S$ und $v \in T$ gelten, da sonst $(u, v) \in E_f$ wäre und somit $v \in S$ gelten müsste. Mit den zwei vorherigen Sätzen folgt dann $|f| = f(S, T) = c(S, T)$.

Max-Flow-Min-Cut-Theorem

Satz

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

Beweis

(2) $\Rightarrow$ (3): G_f enthält keinen Pfad von s nach t. Definiere:

$$S := \{v \in V \mid \text{ es gibt einen Pfad in } G_f \text{ von } s \text{ nach } v\}$$

und $T := V - S$. (S, T) ist ein Schnitt (warum?), ferner muss $f(u, v) = c(u, v)$ für $u, v \in V$ mit $u \in S$ und $v \in T$ gelten, da sonst $(u, v) \in E_f$ wäre und somit $v \in S$ gelten müsste. Mit den zwei vorherigen Sätzen folgt dann $|f| = f(S, T) = c(S, T)$.

Max-Flow-Min-Cut-Theorem

Satz

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

Beweis

(2) $\Rightarrow$ (3): G_f enthält keinen Pfad von s nach t. Definiere:

$$S := \{v \in V \mid \text{ es gibt einen Pfad in } G_f \text{ von } s \text{ nach } v\}$$

und $T := V - S$. (S, T) ist ein Schnitt (warum?), ferner muss $f(u, v) = c(u, v)$ für $u, v \in V$ mit $u \in S$ und $v \in T$ gelten, da sonst $(u, v) \in E_f$ wäre und somit $v \in S$ gelten müsste. Mit den zwei vorherigen Sätzen folgt dann $|f| = f(S, T) = c(S, T)$.

Max-Flow-Min-Cut-Theorem

Satz

1. *f ist ein maximaler Fluss in G.*
3. *Es ist $|f| = c(S, T)$ für einen Schnitt (S, T) von G.*

Beweis.

(3) $\Rightarrow$ (1): Wir haben vorhin $|f| \leq c(S, T)$ für alle Schnitte (S, T) bewiesen. Aus $|f| = c(S, T)$ folgt also, dass f ein maximaler Fluss sein muss. $\square$

Max-Flow-Min-Cut-Theorem

Satz

1. *f ist ein maximaler Fluss in G.*
3. *Es ist $|f| = c(S, T)$ für einen Schnitt (S, T) von G.*

Beweis.

(3) $\Rightarrow$ (1): Wir haben vorhin $|f| \leq c(S, T)$ für alle Schnitte (S, T) bewiesen. Aus $|f| = c(S, T)$ folgt also, dass f ein maximaler Fluss sein muss. $\square$

Schnitte - Zusammenfassung

- Schnitte sind ein Hilfsmittel für das Max-Flow-Min-Cut-Theorem, das aussagt, dass der maximale Fluss in einem Flussnetzwerk der Kapazität eines minimalen Schnittes entspricht.
- Daraus (und aus der ausführlicheren Formulierung des Theorems oben) folgt die Korrektheit der Ford-Fulkerson-Methode.

Die Ford-Fulkerson-Methode

Die Ford-Fulkerson-Methode:

0. Sei $f = 0$.
1. Bestimme $G_f = (V, E_f)$ und c_f.
2. Bestimme einen Erweiterungspfad p. (Gibt es keinen, gehe zu 5.)
3. Erhöhe f entlang p um $c_f(p)$.
4. Gehe zu 1.
5. f ist ein maximaler Fluss.

Pause to Ponder

Wo haben wir hier Stellschrauben?

Die Ford-Fulkerson-Methode

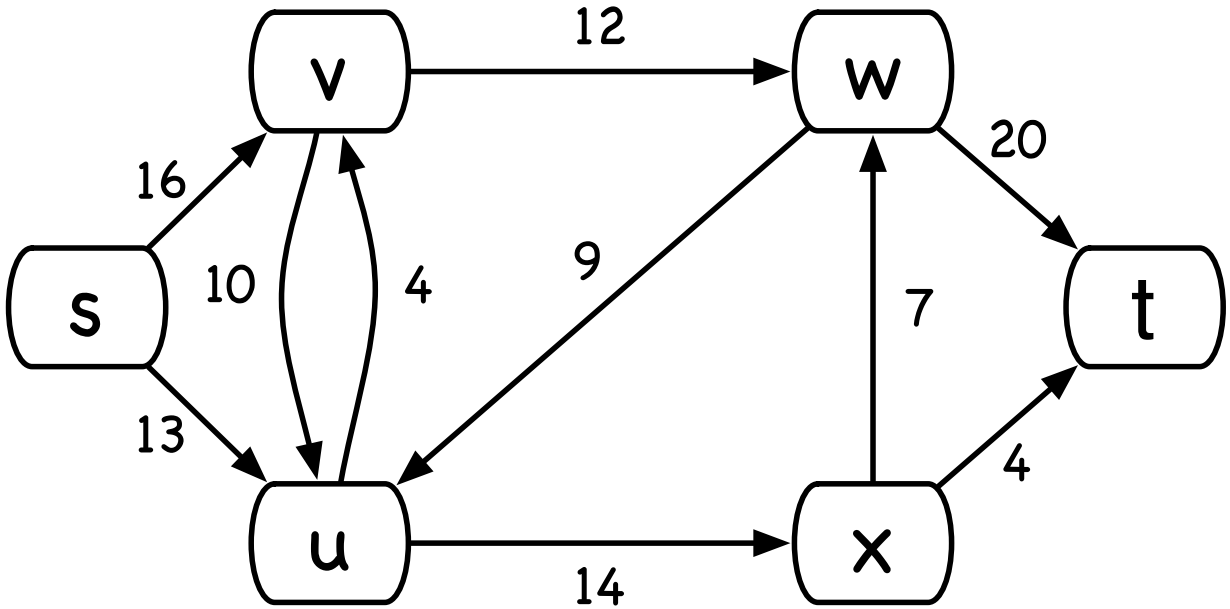
Die Ford-Fulkerson-Methode:

0. Sei $f = 0$.
1. Bestimme $G_f = (V, E_f)$ und c_f.
2. Bestimme einen Erweiterungspfad p. (Gibt es keinen, gehe zu 5.)
3. Erhöhe f entlang p um $c_f(p)$.
4. Gehe zu 1.
5. f ist ein maximaler Fluss.

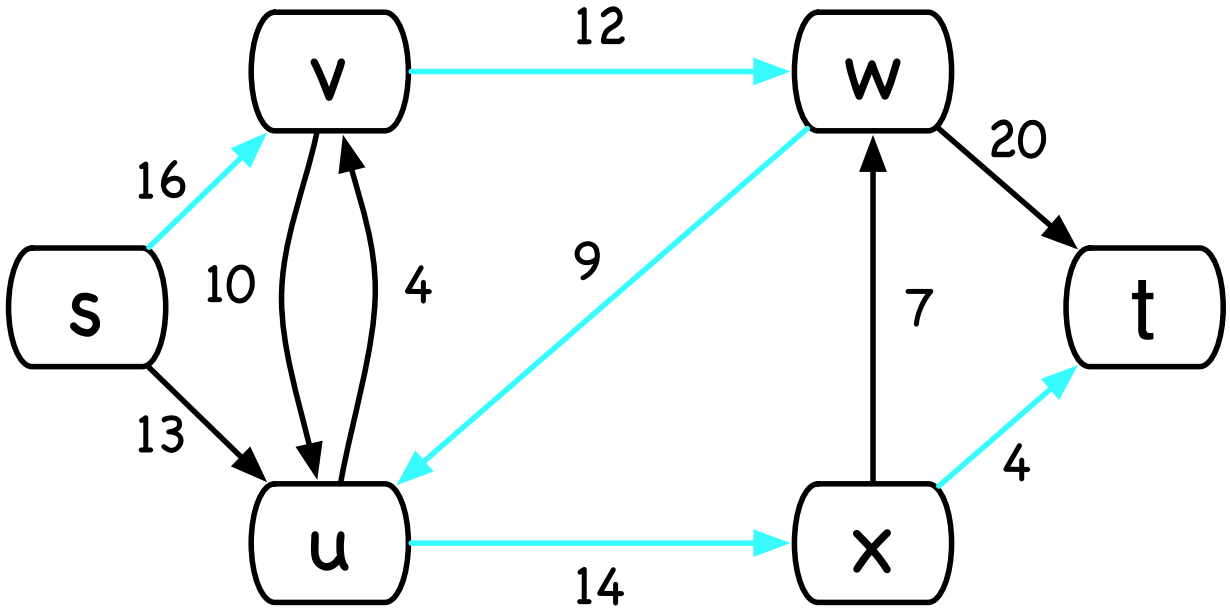
Pause to Ponder

Wo haben wir hier Stellschrauben?

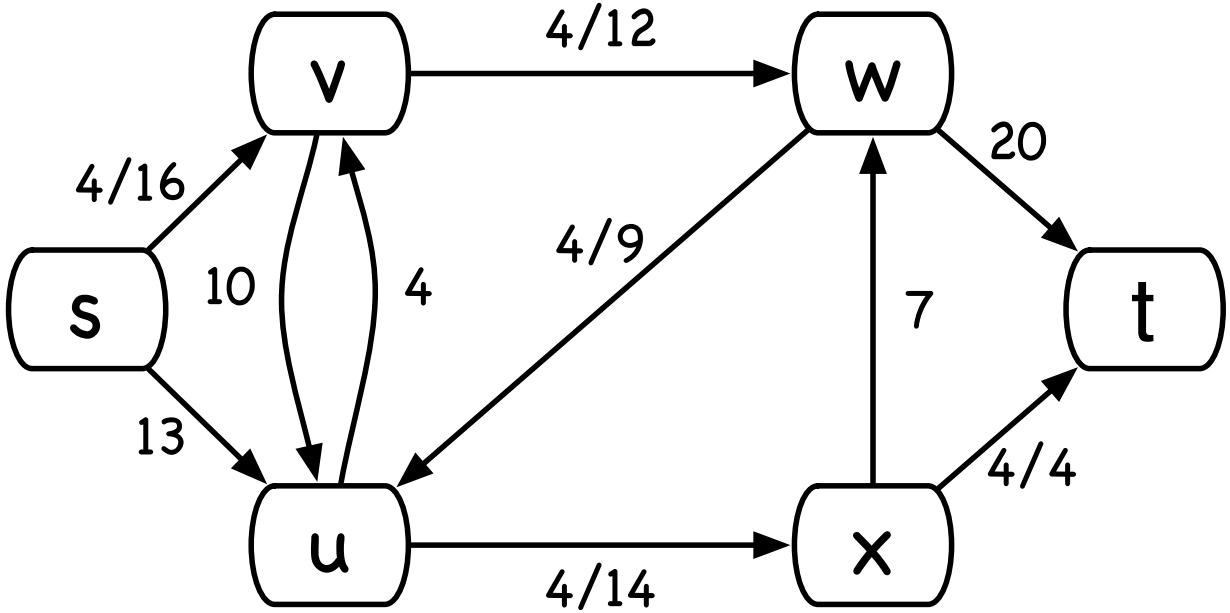
Ford-Fulkerson: Beispiel



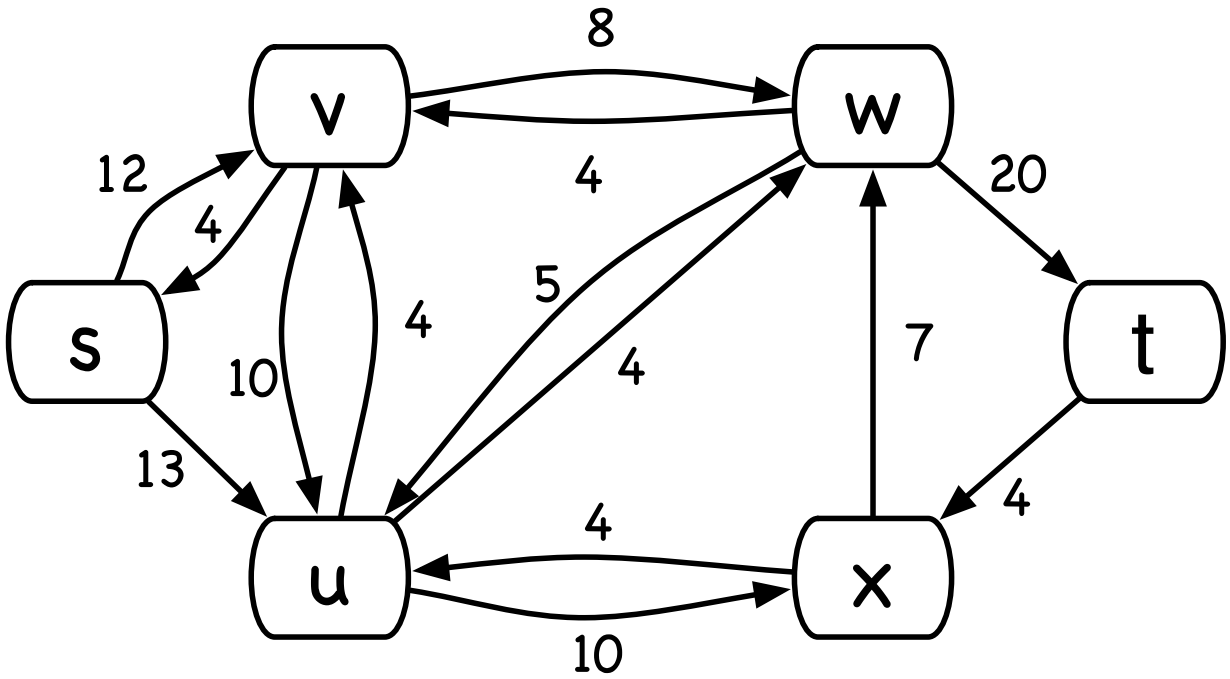
Ford-Fulkerson: Beispiel



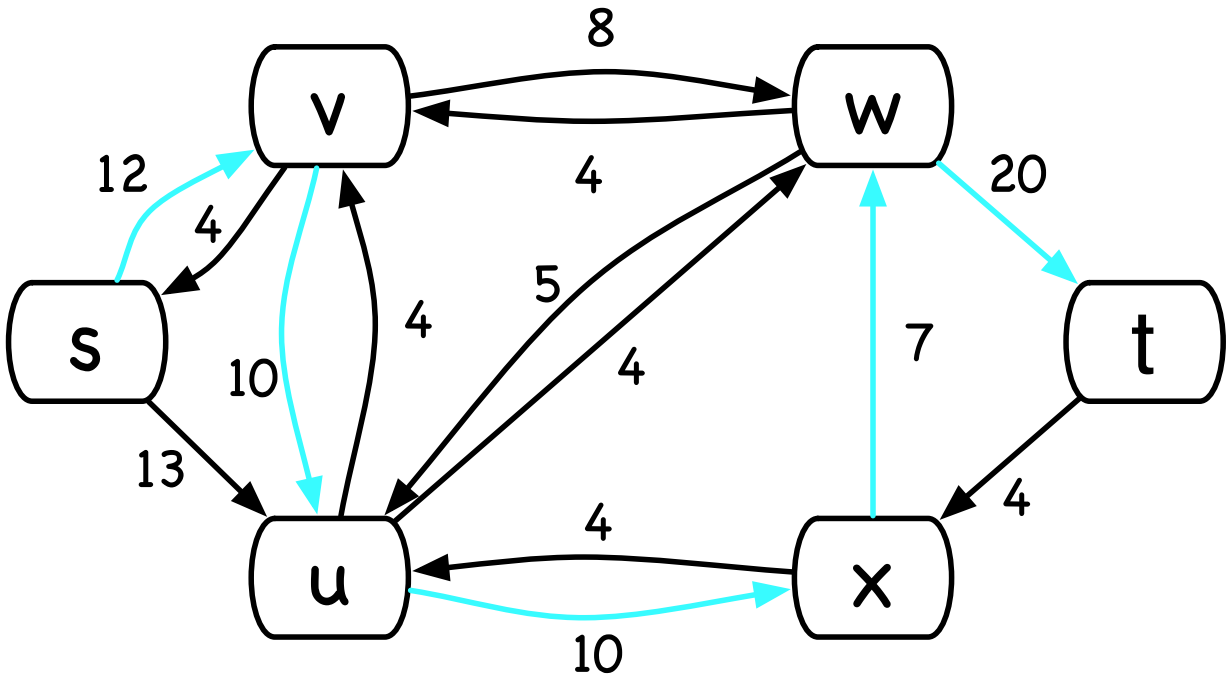
Ford-Fulkerson: Beispiel



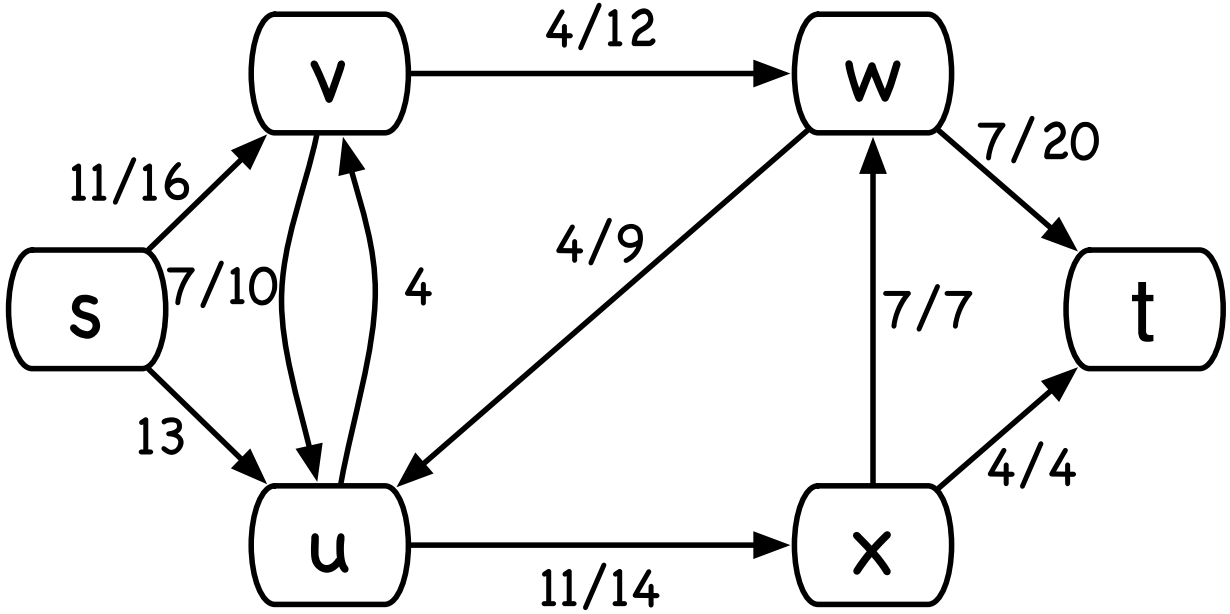
Ford-Fulkerson: Beispiel



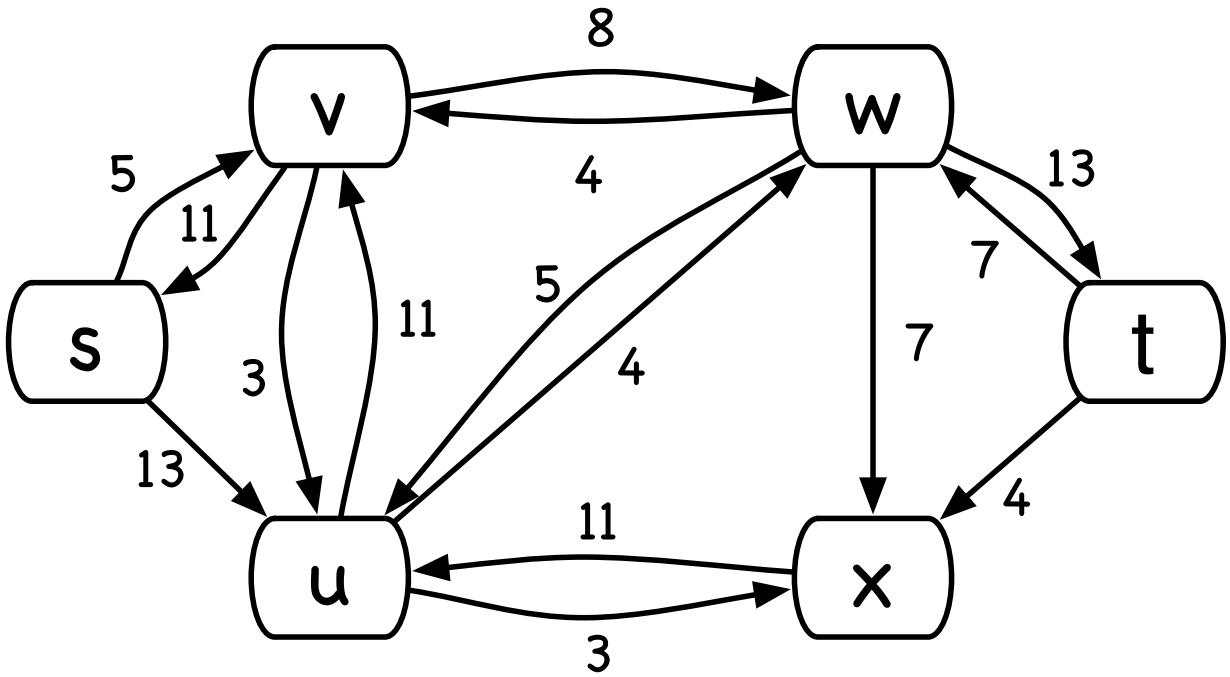
Ford-Fulkerson: Beispiel



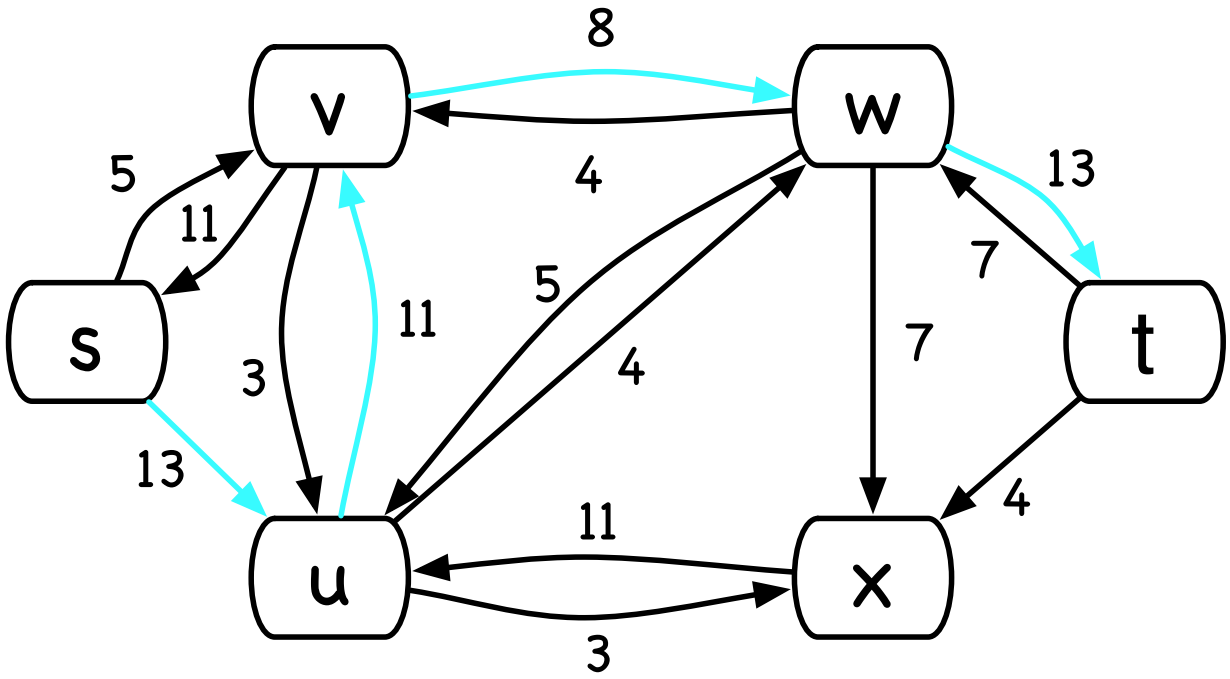
Ford-Fulkerson: Beispiel



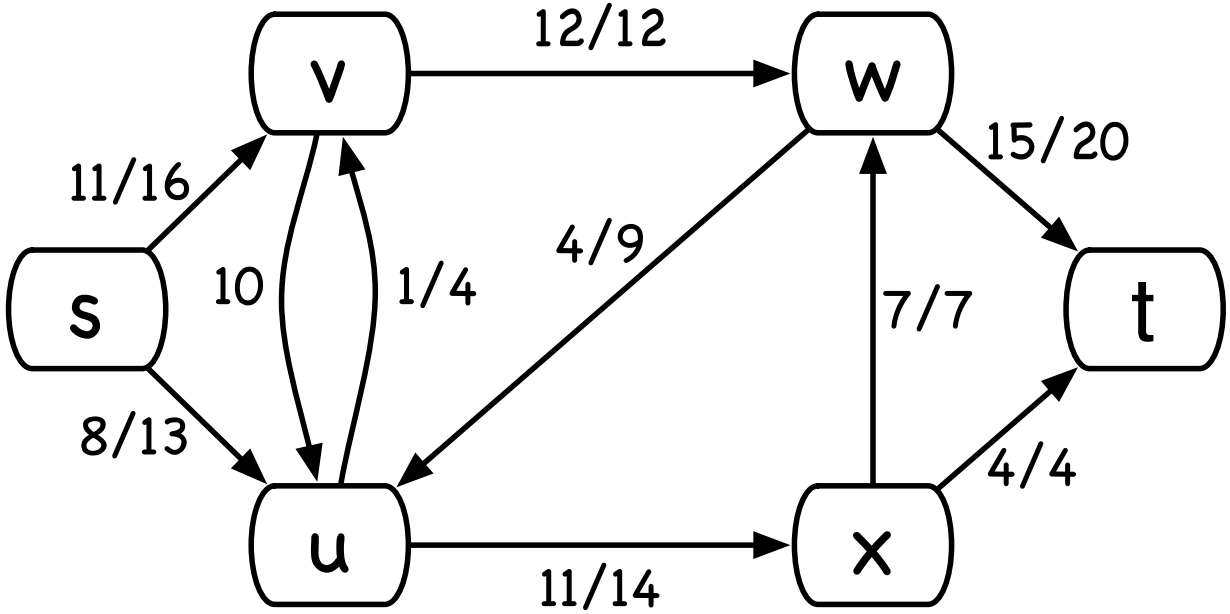
Ford-Fulkerson: Beispiel



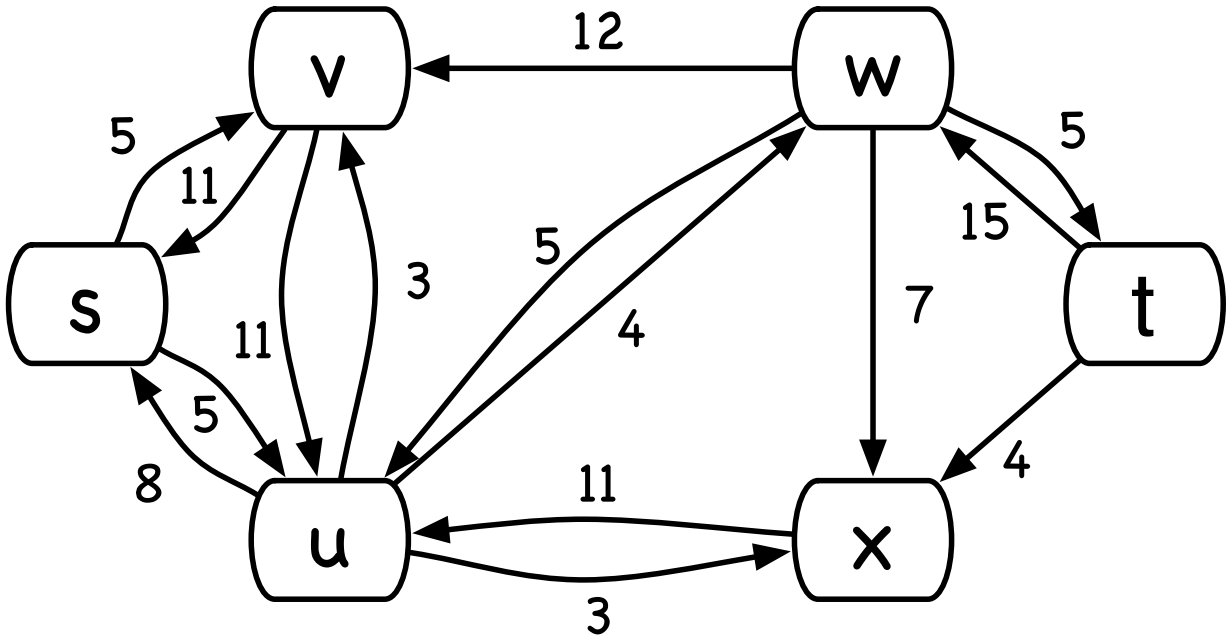
Ford-Fulkerson: Beispiel



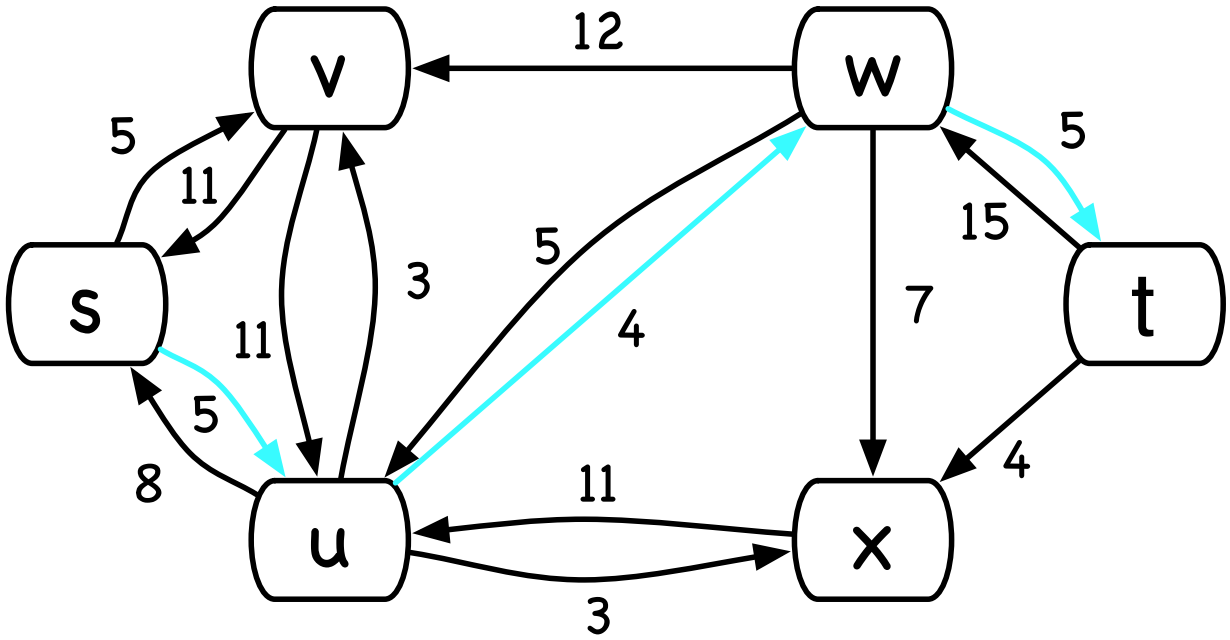
Ford-Fulkerson: Beispiel



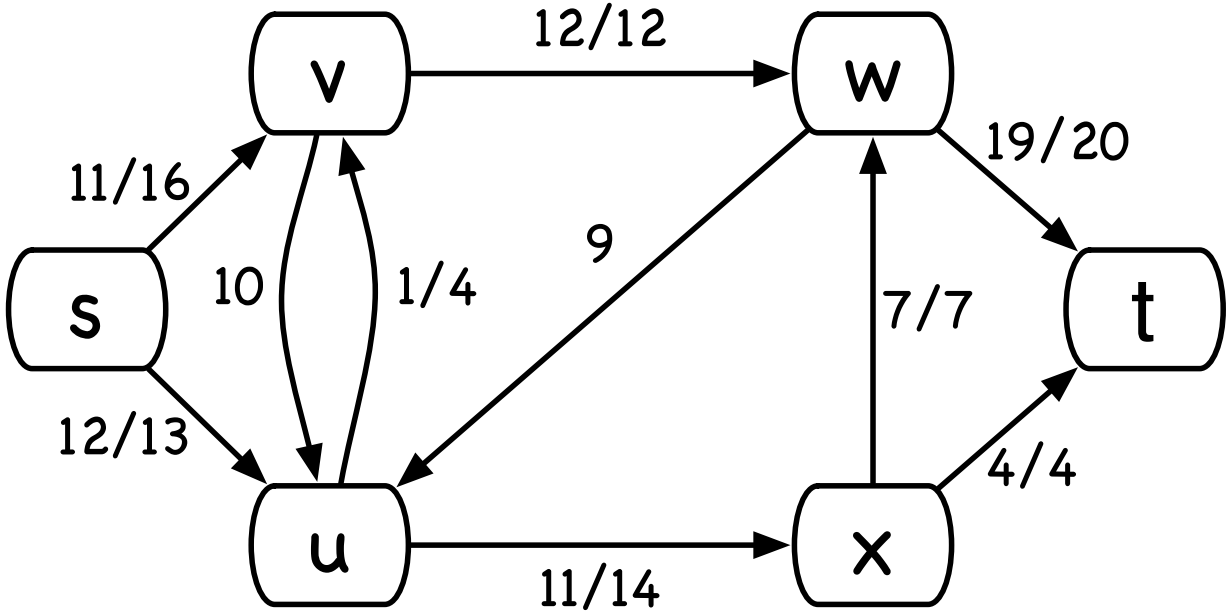
Ford-Fulkerson: Beispiel



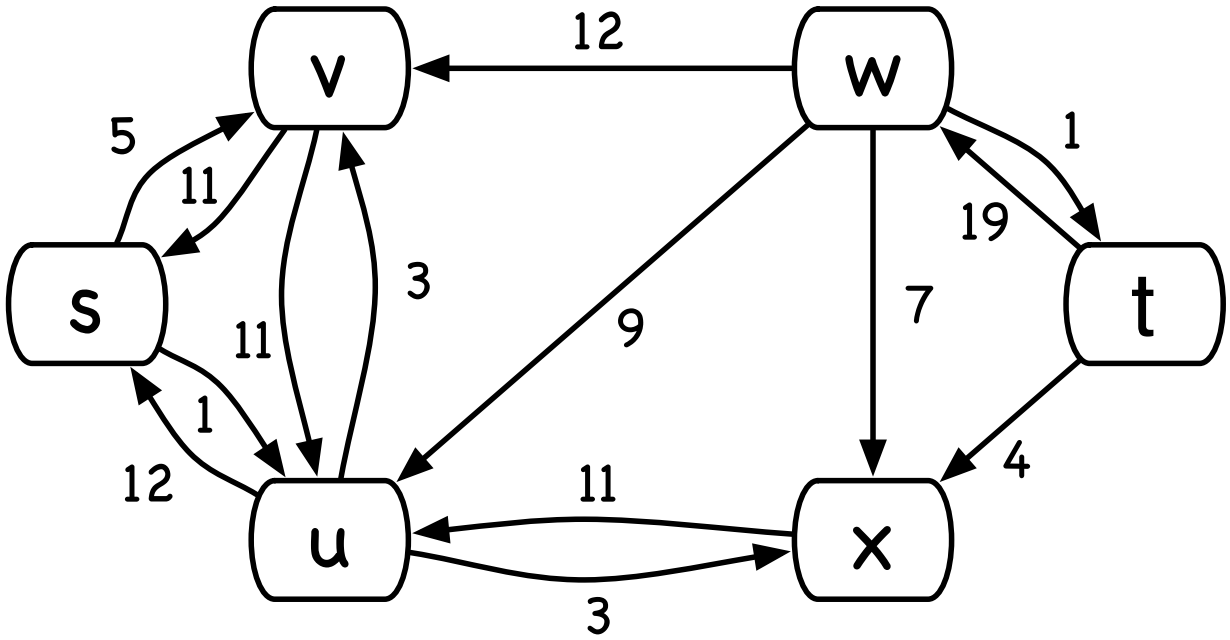
Ford-Fulkerson: Beispiel



Ford-Fulkerson: Beispiel



Ford-Fulkerson: Beispiel



Ford-Fulkerson-Basisalgorithmus

Algorithmus 1 FordFulkerson(G, s, t)

1: **for** alle $(u, v) \in E(G)$ **do**
2: $\quad f[u, v] = 0, f[v, u] = 0$
3: **end for**
4: **while** es gibt einen Pfad p von s nach t in G_f **do**
5: $\quad c_f(p) = \min\{c_f(u, v) \mid (u, v)$ gehört zu $p\}$
6: $\quad$ **for** alle Kanten (u, v) von p **do**
7: $\quad\quad f[u, v] = f[u, v] + c_f(p)$
8: $\quad\quad f[v, u] = -f[u, v]$
9: $\quad$ **end for**
10: **end while**

Ford-Fulkerson-Basisalgorithmus - Analyse

Analyse:

- Die for-Schleife ist in $\Theta(E)$.
- Die while-Schleife wird maximal $|f^*|$-mal ausgeführt, wobei f^* ein maximaler Fluss ist.
- Nutzen wir sinnvolle Datenstrukturen und ermitteln p in $O(E)$ (z.B. mittels Breiten- oder Tiefensuche), so ist jede Iteration der while-Schleife in $O(E)$.
- Insgesamt ist die Routine damit in $O(E \cdot |f^*|)$.

Anmerkung

Wir gehen dabei von ganzzahligen oder rationalen Kapazitäten aus. Bei irrationalen Kantenkapazitäten kann Unsinn passieren, was aber faktisch nur von mathematischem Interesse ist.

Ford-Fulkerson-Basisalgorithmus - Analyse

Analyse:

- Die for-Schleife ist in $\Theta(E)$.
- Die while-Schleife wird maximal $|f^*|$-mal ausgeführt, wobei f^* ein maximaler Fluss ist.
- Nutzen wir sinnvolle Datenstrukturen und ermitteln p in $O(E)$ (z.B. mittels Breiten- oder Tiefensuche), so ist jede Iteration der while-Schleife in $O(E)$.
- Insgesamt ist die Routine damit in $O(E \cdot |f^*|)$.

Anmerkung

Wir gehen dabei von ganzzahligen oder rationalen Kapazitäten aus. Bei irrationalen Kantenkapazitäten kann Unsinn passieren, was aber faktisch nur von mathematischem Interesse ist.

Ford-Fulkerson-Basisalgorithmus - Analyse

Analyse:

- Die for-Schleife ist in $\Theta(E)$.
- Die while-Schleife wird maximal $|f^*|$-mal ausgeführt, wobei f^* ein maximaler Fluss ist.
- Nutzen wir sinnvolle Datenstrukturen und ermitteln p in $O(E)$ (z.B. mittels Breiten- oder Tiefensuche), so ist jede Iteration der while-Schleife in $O(E)$.
- Insgesamt ist die Routine damit in $O(E \cdot |f^*|)$.

Anmerkung

Wir gehen dabei von ganzzahligen oder rationalen Kapazitäten aus. Bei irrationalen Kantenkapazitäten kann Unsinn passieren, was aber faktisch nur von mathematischem Interesse ist.

Ford-Fulkerson-Basisalgorithmus - Analyse

Analyse:

- Die for-Schleife ist in $\Theta(E)$.
- Die while-Schleife wird maximal $|f^*|$-mal ausgeführt, wobei f^* ein maximaler Fluss ist.
- Nutzen wir sinnvolle Datenstrukturen und ermitteln p in $O(E)$ (z.B. mittels Breiten- oder Tiefensuche), so ist jede Iteration der while-Schleife in $O(E)$.
- Insgesamt ist die Routine damit in $O(E \cdot |f^*|)$.

Anmerkung

Wir gehen dabei von ganzzahligen oder rationalen Kapazitäten aus. Bei irrationalen Kantenkapazitäten kann Unsinn passieren, was aber faktisch nur von mathematischem Interesse ist.

Ford-Fulkerson-Basisalgorithmus - Analyse

Analyse:

- Die for-Schleife ist in $\Theta(E)$.
- Die while-Schleife wird maximal $|f^*|$-mal ausgeführt, wobei f^* ein maximaler Fluss ist.
- Nutzen wir sinnvolle Datenstrukturen und ermitteln p in $O(E)$ (z.B. mittels Breiten- oder Tiefensuche), so ist jede Iteration der while-Schleife in $O(E)$.
- Insgesamt ist die Routine damit in $O(E \cdot |f^*|)$.

Anmerkung

Wir gehen dabei von ganzzahligen oder rationalen Kapazitäten aus. Bei irrationalen Kantenkapazitäten kann Unsinn passieren, was aber faktisch nur von mathematischem Interesse ist.

Ford-Fulkerson-Basisalgorithmus - Analyse

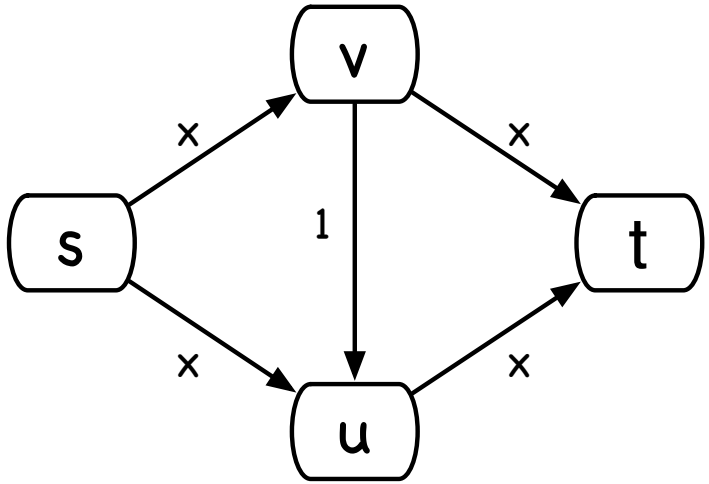
Analyse:

- Die for-Schleife ist in $\Theta(E)$.
- Die while-Schleife wird maximal $|f^*|$-mal ausgeführt, wobei f^* ein maximaler Fluss ist.
- Nutzen wir sinnvolle Datenstrukturen und ermitteln p in $O(E)$ (z.B. mittels Breiten- oder Tiefensuche), so ist jede Iteration der while-Schleife in $O(E)$.
- Insgesamt ist die Routine damit in $O(E \cdot |f^*|)$.

Anmerkung

Wir gehen dabei von ganzzahligen oder rationalen Kapazitäten aus. Bei irrationalen Kantenkapazitäten kann Unsinn passieren, was aber faktisch nur von mathematischem Interesse ist.

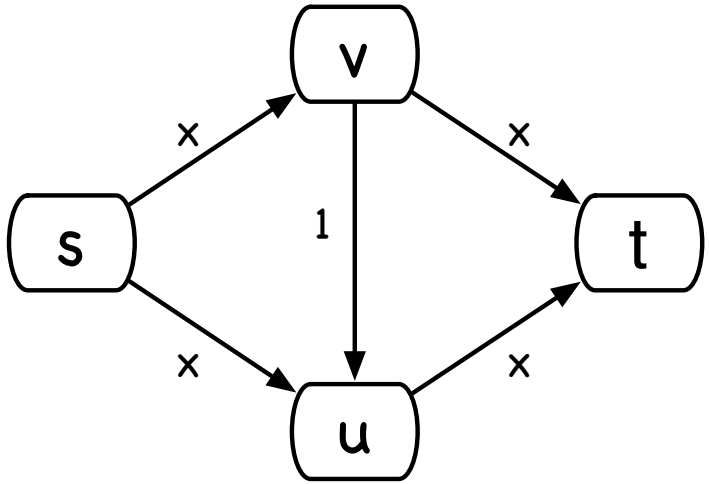
Ford-Fulkerson - Ein Problemfall



Beobachtung

Hier kann bei wiederholter ungünstiger Wahl von p die obere Laufzeitgrenze von $O(E \cdot |f^*|)$ erreicht werden, obwohl zwei Iterationen reichen würden. (Ist z.B. $x = 100$ und wird wiederholt s-v-u-t und s-u-v-t (im Restnetzwerk!) gewählt, braucht man 200 Iterationen (statt 2).)

Ford-Fulkerson - Ein Problemfall



Beobachtung

Hier kann bei wiederholter ungünstiger Wahl von p die obere Laufzeitgrenze von $O(E \cdot |f^*|)$ erreicht werden, obwohl zwei Iterationen reichen würden. (Ist z.B. $x = 100$ und wird wiederholt s-v-u-t und s-u-v-t (im Restnetzwerk!) gewählt, braucht man 200 Iterationen (statt 2).)

Edmonds-Karp-Algorithmus

Der Edmonds-Karp-Algorithmus:

- Ermittle einen Erweiterungspfad p in der Ford-Fulkerson-Methode durch eine Breitensuche, d.h. p ist ein *kürzester Pfad* von s nach t im Restnetzwerk (wobei jede Kante das Gewicht 1 hat).

Satz

Die Anzahl der durch den Edmonds-Karp-Algorithmus vorgenommenen Flusserweiterungen ist in $O(V \cdot E)$. Da jede Iteration (der Prozedur FordFulkerson) in Zeit $O(E)$ implementiert werden kann, ist der Edmonds-Karp-Algorithmus in $O(V \cdot E^2)$.

Anmerkung

Wenn auch hier etwas unscheinbar auf nur einer Folie platziert, ist dies ein sehr wichtiges Resultat!

Edmonds-Karp-Algorithmus

Der Edmonds-Karp-Algorithmus:

- Ermittle einen Erweiterungspfad p in der Ford-Fulkerson-Methode durch eine Breitensuche, d.h. p ist ein *kürzester Pfad* von s nach t im Restnetzwerk (wobei jede Kante das Gewicht 1 hat).

Satz

Die Anzahl der durch den Edmonds-Karp-Algorithmus vorgenommenen Flusserweiterungen ist in $O(V \cdot E)$. Da jede Iteration (der Prozedur FordFulkerson) in Zeit $O(E)$ implementiert werden kann, ist der Edmonds-Karp-Algorithmus in $O(V \cdot E^2)$.

Anmerkung

Wenn auch hier etwas unscheinbar auf nur einer Folie platziert, ist dies ein sehr wichtiges Resultat!

Ausblick und Anwendungen

Anmerkung

Maximale Flüsse sind nicht nur für das Problem an sich interessant, sondern treten oft auch bei anderen Problemen auf, die also als Flussprobleme formuliert werden können. Ein Beispiel ist die Bestimmung eines maximalen bipartiten Matchings. Wir gehen hier nicht mehr darauf ein (wer mag: Kapitel 26.3 im [Cormen]).

Ausblick (für Interessierte)

Den Beweis für den Edmonds-Karp-Algorithmus findet man im [Cormen] (zwei Seiten). Zudem kann die Laufzeit mit sogenannten *Push/Relabel-Algorithmen* weiter auf $O(V^2 \cdot E)$ und sogar $O(V^3)$ verbessert werden. Mehr dazu findet man auch im [Cormen]. - Und es geht sogar noch schneller (siehe Kapitelbemerkungen und Literaturhinweise im [Cormen]) ...

Ausblick und Anwendungen

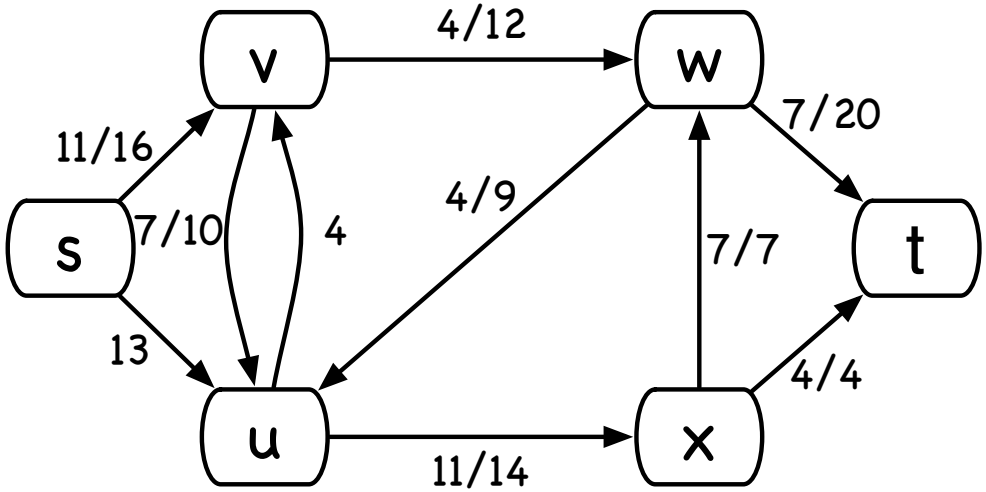
Anmerkung

Maximale Flüsse sind nicht nur für das Problem an sich interessant, sondern treten oft auch bei anderen Problemen auf, die also als Flussprobleme formuliert werden können. Ein Beispiel ist die Bestimmung eines maximalen bipartiten Matchings. Wir gehen hier nicht mehr darauf ein (wer mag: Kapitel 26.3 im [Cormen]).

Ausblick (für Interessierte)

Den Beweis für den Edmonds-Karp-Algorithmus findet man im [Cormen] (zwei Seiten). Zudem kann die Laufzeit mit sogenannten *Push/Relabel-Algorithmen* weiter auf $O(V^2 \cdot E)$ und sogar $O(V^3)$ verbessert werden. Mehr dazu findet man auch im [Cormen]. - Und es geht sogar noch schneller (siehe Kapitelbemerkungen und Literaturhinweise im [Cormen]) ...

Begriffe und Definitionen



Definition (Flussnetzwerk)

Ein *Flussnetzwerk* ist ein gerichteter Graph $G = (V, E)$, in dem jeder Kante $(u, v) \in E$ zusätzlich eine nichtnegative *Kapazität* $c(u, v) \geq 0$ zugewiesen wird. (Ist $(u, v) \notin E$, so nehmen wir $c(u, v) = 0$ an.)
Es gibt die besonderen Knoten $s, t \in V$, die *Quelle* (s) und die *Senke* (t). Alle Knoten liegen auf Pfaden von der Quelle zur Senke.

Begriffe und Definitionen

Definition (Fluss)

Sei $G = (V, E)$ mit Kapazitätsfunktion c ein Flussnetzwerk, s und t wie oben. Ein *Fluss* in G ist eine Funktion $f : V \times V \to \mathbb{R}$, für die gilt:

1. Kapazitätsbeschränkung: $f(u, v) \leq c(u, v) \quad \forall u, v \in V$
2. Asymmetrie: $f(u, v) = -f(v, u) \quad \forall u, v \in V$
3. Flusserhaltung:

$$\sum_{v \in V} f(u, v) = 0 \quad \forall u \in V \setminus \{s, t\}$$

Das Problem des maximalen Flusses

Definition (Problem des maximalen Flusses)

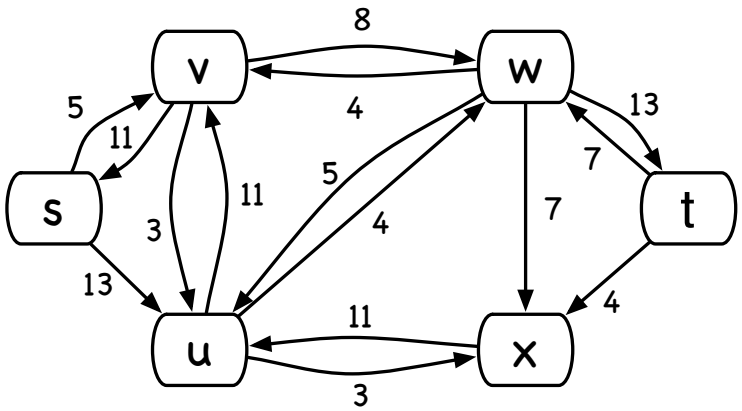
Eingabe: Ein Flussnetzwerk G mit Kapazitätsfunktion c und Quelle s und Senke t.
Gesucht: Ein Fluss mit maximalen Wert. Dabei ist der *Wert eines Flusses* definiert durch

$$|f| = \sum_{v \in V} f(s, v).$$

Anmerkung

Dieses Problem tritt bei vielen Anwendungen (Logistik, Material-/Informationsflüsse, ...) direkt auf, oft lassen sich ganz anders erscheinende Probleme aber auch als Flussprobleme formulieren und dann so lösen.

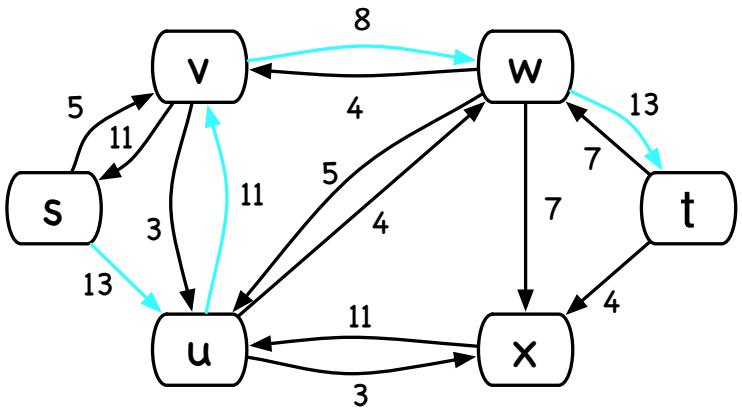
Restnetzwerke: Definition



Definition

Sei $G = (V, E)$ mit c ein Flussnetzwerk und f ein Fluss in G. Wir definieren zu je zwei Knoten u, v die *Restkapazität* durch $c_f(u, v) = c(u, v) - f(u, v)$. Das Restnetzwerk ist dann gegeben durch $G_f = (V, E_f)$ und c_f, wobei $E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}$ die Menge der *Restkanten* ist. Das Restnetzwerk besteht also aus jenen Kanten, die noch mehr Fluss aufnehmen können.

Erweiterungspfade: Definition



Definition

Sei $G = (V, E)$ mit c ein Flussnetzwerk, f ein Fluss in G und G_f das Restnetzwerk. Ein *Erweiterungspfad* ist ein einfacher Pfad von s nach t in G_f. Mit

$$c_f(p) = \min\{c_f(u, v) \mid (u, v) \text{ ist eine Kante von } p\}$$

wird die *Restkapazität* eines Erweiterungspfades p bezeichnet.

Restnetzwerke und Erweiterungspfade: Zusammenfassung

- Intuitive Definition: Das *Restnetzwerk* zu einem gegebenen Flussnetzwerk *und* einem Fluss besteht aus jenen Kanten, die mehr Fluss aufnehmen können. Ein *Erweiterungspfad* ist ein einfacher Pfad von s nach t im Restnetzwerk.
- Findet man in einem Restnetzwerk G_f (f ist ein Fluss in dem Flussnetzwerk G) einen weiteren Fluss f', so ist $f + f'$ erneut ein Fluss in G mit dem Wert $|f| + |f'| > |f|$.
- Letztere gilt insb. auch, wenn man über einen Erweiterungspfad so viel wie möglich 'leitet' (Restkapazität).

Die Ford-Fulkerson-Methode

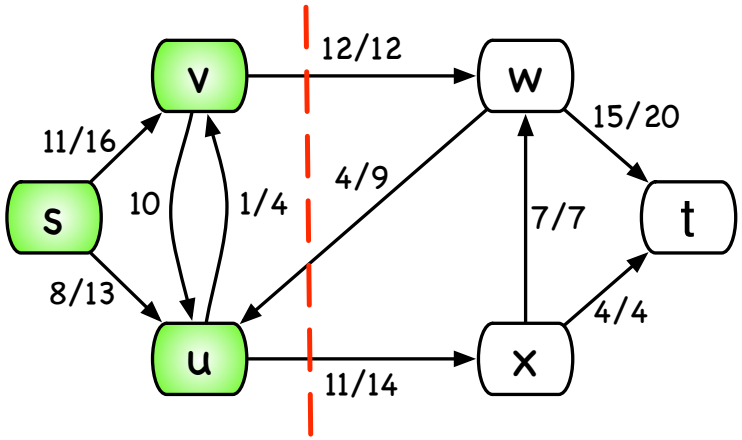
Die Ford-Fulkerson-Methode:

0. Sei $f = 0$.
1. Bestimme $G_f = (V, E_f)$ und c_f.
2. Bestimme einen Erweiterungspfad p. (Gibt es keinen, gehe zu 5.)
3. Erhöhe f entlang p um $c_f(p)$.
4. Gehe zu 1.
5. f ist ein maximaler Fluss.

Anmerkung

Die Idee: Bestimme so lange wie möglich Erweiterungspfade und erhöhe den Fluss jeweils um die Restkapazität. Laufzeit: $O(E \cdot |f^*|)$, wobei f^* ein maximaler Fluss ist.

Schnitte: Definition



Definition (Schnitt)

Ein *Schnitt* (S, T) eines Flussnetzwerkes $G = (V, E)$ ist eine Partionierung von V in S und $T = V - S$, wobei $s \in S$ und $t \in T$ gilt. Der *Nettofluss* und die *Kapazität* eines Schnittes (S, T) ist durch $f(S, T)$ bzw. $c(S, T)$ definiert. Ein *minimaler Schnitt* ist ein Schnitt, dessen Kapazität die kleinste von allen Schnitten des Netzwerkes ist.

Max-Flow-Min-Cut-Theorem

Satz

Sei $G = (V, E)$ mit c, s, t ein Flussnetzwerk, f ein Fluss in G. Die folgenden Bedingungen sind äquivalent:

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

Anmerkung

Mit diesem Satz folgt die Korrektheit der Ford-Fulkerson-Methode

Edmonds-Karp-Algorithmus

Der Edmonds-Karp-Algorithmus:

- Ermittle einen Erweiterungspfad p in der Ford-Fulkerson-Methode durch eine Breitensuche, d.h. p ist ein *kürzester Pfad* von s nach t im Restnetzwerk (wobei jede Kante das Gewicht 1 hat).

Satz

Die Anzahl der durch den Edmonds-Karp-Algorithmus vorgenommenen Flusserweiterungen ist in $O(V \cdot E)$. Da jede Iteration (der Prozedur FordFulkerson) in Zeit $O(E)$ implementiert werden kann, ist der Edmonds-Karp-Algorithmus in $O(V \cdot E^2)$.

Wiederholung

- Graphen
  - Grundlagen
    - Definition und Darstellung
    - Tiefen- und Breitensuche
  - Topologisches Sortieren
  - Starke Zusammenhangskomponenten
  - Minimale Spannbäume
    - Definition
    - Algorithmus von Kruskal
    - Algorithmus von Prim
  - Bestimmen der kürzesten Pfade
    - Definition und Varianten
    - (Breitensuche)
    - Algorithmus von Dijkstra
    - Algorithmus von Bellman-Ford
    - Algorithmus von Floyd-Warshall
  - Flüsse