Algorithmen und Datenstrukturen

Vorlesung #10 – Flussgraphen

Benjamin Blankertz

Lehrstuhl für Neurotechnologie, TU Berlin

benjamin.blankertz@tu-berlin.de

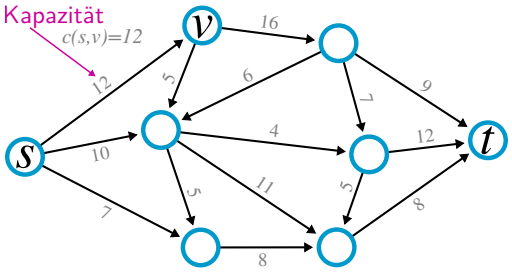
19 · Jun · 2019

Themen der heutigen Vorlesung

- **Flussgraphen**
- Schnitte durch Graphen
- Herausforderungen: maximaler Fluss (*max-flow*), minimaler Schnitt
- Fluss-vergrößernde Pfade
- Allgemeine *max-flow* Methode: **Ford-Fulkerson**
- Grenzen der Ford-Fulkerson Methode
- Geschickte Wahl der vergrößernden Pfade:
  - kürzeste Pfade: **Edmonds-Karp** Algorithmus
  - dicke Pfade: ***Capacity Scaling*** Algorithmus

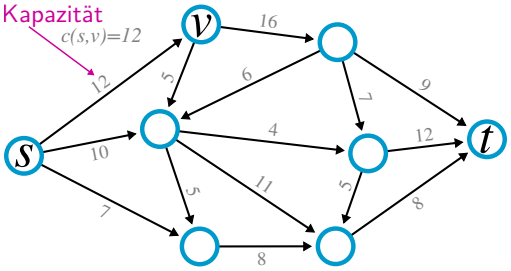
Flussgraphen

- Ein **Flussgraph** ist ein gewichteter Digraph $G = (V, E)$. Die Gewichte werden als Kapazitäten bezeichnet und sind positiv.
- Wir schreiben $c(v, w)$ für die Kapazität der Kante $v \to w$ und definieren $c(v, w) = 0$ für $v \to w \notin E$.



Flussgraphen

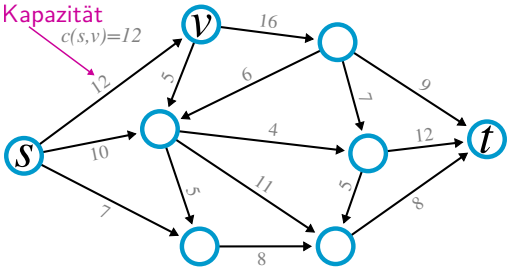
- Ein **Flussgraph** ist ein gewichteter Digraph $G = (V, E)$. Die Gewichte werden als Kapazitäten bezeichnet und sind positiv.
- Wir schreiben $c(v, w)$ für die Kapazität der Kante $v \to w$ und definieren $c(v, w) = 0$ für $v \to w \notin E$.



- Wir setzen voraus, dass es zwischen Knotenpaaren höchstens eine Kante gibt. Um Kanten in beiden Richtungen zu modellieren, wird bei einer der beiden Kanten ein Zwischenknoten eingefügt.
- Des Weiteren nehmen wir an, dass es eine ausgezeichnete **Quelle** s (Knoten mit Eingangsgrad 0) und eine ausgezeichnete **Senke** t (Knoten mit Ausgangsgrad 0).

Flussgraphen

- Ein **Flussgraph** ist ein gewichteter Digraph $G = (V, E)$. Die Gewichte werden als Kapazitäten bezeichnet und sind positiv.
- Wir schreiben $c(v, w)$ für die Kapazität der Kante $v \to w$ und definieren $c(v, w) = 0$ für $v \to w \notin E$.



- Wir setzen voraus, dass es zwischen Knotenpaaren höchstens eine Kante gibt. Um Kanten in beiden Richtungen zu modellieren, wird bei einer der beiden Kanten ein Zwischenknoten eingefügt.
- Des Weiteren nehmen wir an, dass es eine ausgezeichnete **Quelle** s (Knoten mit Eingangsgrad 0) und eine ausgezeichnete **Senke** t (Knoten mit Ausgangsgrad 0).
- Man stellt sich die Kanten am besten als Leitungen vor, durch die eine Flüssigkeit fließt. Ebenso kann z.B. der Fluss von Informationen durch Netzwerke modelliert werden.

Definition Fluss

- Ein **Fluss** (*flow*) ordnet jeder Kante des Digraphen einen Fluss zu, mittels einer Funktion

  $f : \boldsymbol{V} \times \boldsymbol{V} \to \mathbb{R}$ (wobei $f(v, w) = 0$ für $v \to w \notin \boldsymbol{E}$ gesetzt wird),

  die die folgenden beiden Bedingungen erfüllt:

Definition Fluss

- Ein **Fluss** (*flow*) ordnet jeder Kante des Digraphen einen Fluss zu, mittels einer Funktion

  $f : \boldsymbol{V} \times \boldsymbol{V} \to \mathbb{R}$ (wobei $f(v, w) = 0$ für $v \to w \notin \boldsymbol{E}$ gesetzt wird),

  die die folgenden beiden Bedingungen erfüllt:

- **Kapazitätsbeschränkung:** (*capacity constraint*) Der Fluss jeder Kante ist positiv und höchstens gleich der Kapazität der Kante:

  $$\forall v, w \in \boldsymbol{V} : \quad 0 \leq f(v, w) \leq c(v, w)$$

- **Flusserhaltung:** (*local equilibrium*) Für jeden Knoten außer Quelle und Senke ist der **Zufluss** (Summe vom Fluss der Kanten nach v) gleich dem **Abfluss**:

  $$\forall v \in \boldsymbol{V} - \{s, t\} : \quad \sum_{w \in \boldsymbol{V}} f(w, v) = \sum_{w \in \boldsymbol{V}} f(v, w)$$

Definition Fluss

- Ein **Fluss** (*flow*) ordnet jeder Kante des Digraphen einen Fluss zu, mittels einer Funktion

  $f : \boldsymbol{V} \times \boldsymbol{V} \to \mathbb{R}$ (wobei $f(v, w) = 0$ für $v \to w \notin \boldsymbol{E}$ gesetzt wird),

  die die folgenden beiden Bedingungen erfüllt:

- **Kapazitätsbeschränkung:** (*capacity constraint*) Der Fluss jeder Kante ist positiv und höchstens gleich der Kapazität der Kante:

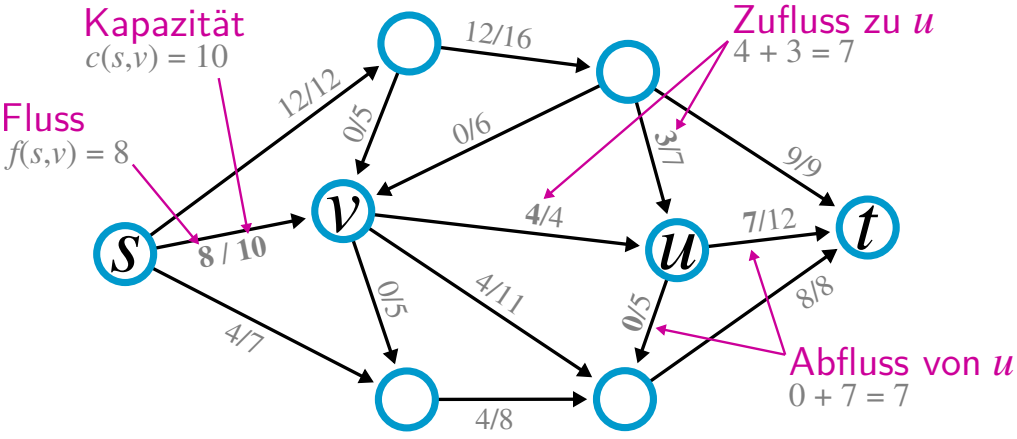
  $$\forall v, w \in \boldsymbol{V} : \quad 0 \leq f(v, w) \leq c(v, w)$$

- **Flusserhaltung:** (*local equilibrium*) Für jeden Knoten außer Quelle und Senke ist der **Zufluss** (Summe vom Fluss der Kanten nach v) gleich dem **Abfluss**:

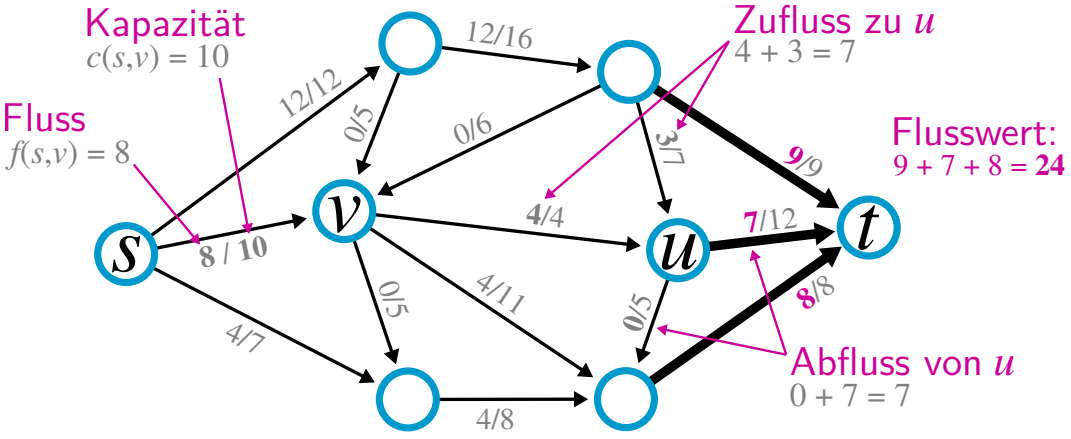
  $$\forall v \in \boldsymbol{V} - \{s, t\} : \quad \sum_{w \in \boldsymbol{V}} f(w, v) = \sum_{w \in \boldsymbol{V}} f(v, w)$$

- Der **Wert des Flusses** ist definiert als der Zufluss zur Senke: $|f| = \sum_{v \in \boldsymbol{V}} f(v, t)$.

Darstellung eines Flussgraphen mit einem Fluss



Darstellung eines Flussgraphen mit einem Fluss



- **Herausforderung:**
  Finde einen Fluss mit maximalem Wert (maximaler Fluss, *maxflow*)!

Strategie zur Bestimmung des maximalen Flusses

- Um den maximalen Fluss eines Flussgraphen zu bestimmen, startet man mit einem 0-Fluss ($f(v, w) = 0$ für alle v, w).
- Dann sucht man iterativ Pfade von s nach t, entlang derer der Fluss erhöht werden kann.

Strategie zur Bestimmung des maximalen Flusses

- Um den maximalen Fluss eines Flussgraphen zu bestimmen, startet man mit einem 0-Fluss ($f(v, w) = 0$ für alle v, w).
- Dann sucht man iterativ Pfade von s nach t, entlang derer der Fluss erhöht werden kann.
- Dazu führen wir als nächstes die augmentierenden, bzw. Fluss vergrößernden Pfade ein.

Fluss vergrößernde Pfade

- Ein **(Fluss) vergrößernder Pfad** (*augmenting path*) ist ein ungerichteter Pfad von s nach t, durch den der Flusswert vergrößert werden kann.
- Dabei bedeutet *ungerichtet*, dass eine gerichtete Flusskante auch in Gegenrichtung benutzt werden kann. Trotzdem denkt man den Pfad in Richtung von s nach t.
- Um den Flusswert vergrößern zu können, müssen zwei Bedingungen erfüllt sein:

Fluss vergrößernde Pfade

- Ein **(Fluss) vergrößernder Pfad** (*augmenting path*) ist ein ungerichteter Pfad von s nach t, durch den der Flusswert vergrößert werden kann.
- Dabei bedeutet *ungerichtet*, dass eine gerichtete Flusskante auch in Gegenrichtung benutzt werden kann. Trotzdem denkt man den Pfad in Richtung von s nach t.
- Um den Flusswert vergrößern zu können, müssen zwei Bedingungen erfüllt sein:

1. Bei Kanten in Pfadrichtung ist die Kapazität nicht ausgeschöpft. Hier kann der Fluss vergrößert werden.
2. Bei Kanten gegen Pfadrichtung ist der Fluss größer als 0. Hier kann der Fluss reduziert werden.

Fluss vergrößernde Pfade

- Ein **(Fluss) vergrößernder Pfad** (*augmenting path*) ist ein ungerichteter Pfad von s nach t, durch den der Flusswert vergrößert werden kann.
- Dabei bedeutet *ungerichtet*, dass eine gerichtete Flusskante auch in Gegenrichtung benutzt werden kann. Trotzdem denkt man den Pfad in Richtung von s nach t.
- Um den Flusswert vergrößern zu können, müssen zwei Bedingungen erfüllt sein:

1. Bei Kanten in Pfadrichtung ist die Kapazität nicht ausgeschöpft. Hier kann der Fluss vergrößert werden.
2. Bei Kanten gegen Pfadrichtung ist der Fluss größer als 0. Hier kann der Fluss reduziert werden.

- Der kritische Wert ist der kleinste Wert, um den der Fluss entlang des Pfades
  - auf Kanten in Pfadrichtung erhöht und
  - auf Kanten gegen Pfadrichtung reduziert werden kann.

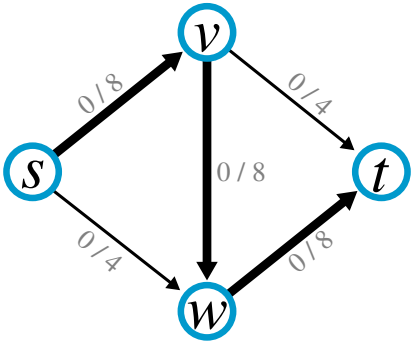
Fluss vergrößernde Pfade

- Ein **(Fluss) vergrößernder Pfad** (*augmenting path*) ist ein ungerichteter Pfad von s nach t, durch den der Flusswert vergrößert werden kann.
- Dabei bedeutet *ungerichtet*, dass eine gerichtete Flusskante auch in Gegenrichtung benutzt werden kann. Trotzdem denkt man den Pfad in Richtung von s nach t.
- Um den Flusswert vergrößern zu können, müssen zwei Bedingungen erfüllt sein:

1. Bei Kanten in Pfadrichtung ist die Kapazität nicht ausgeschöpft. Hier kann der Fluss vergrößert werden.
2. Bei Kanten gegen Pfadrichtung ist der Fluss größer als 0. Hier kann der Fluss reduziert werden.

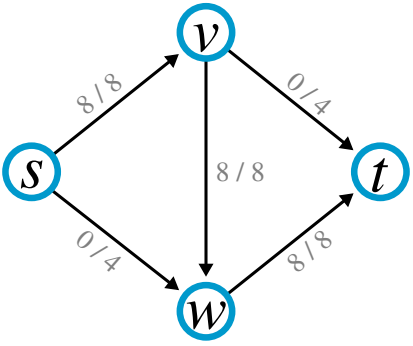
- Der kritische Wert ist der kleinste Wert, um den der Fluss entlang des Pfades
  - auf Kanten in Pfadrichtung erhöht und
  - auf Kanten gegen Pfadrichtung reduziert werden kann.
- Da die letzte Kante zu t in Pfadrichtung geht (da t eine Senke ist), wird der Flusswert um den kritischen Wert des Pfades erhöht.

Kleines Beispiel zur Flussumplanung



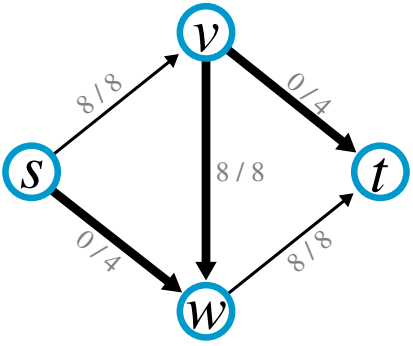
- Es wird ein vergrößernder Pfad gewählt

Kleines Beispiel zur Flussumplanung



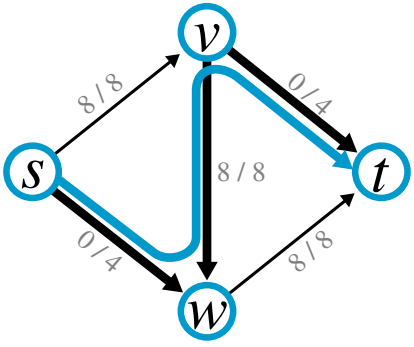
- Es wird ein vergrößernder Pfad gewählt
- und der Fluss entsprechend um 8 Einheiten erhöht.

Kleines Beispiel zur Flussumplanung



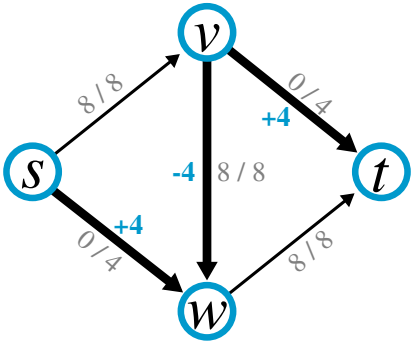
- Für eine Erhöhung um weitere 4 Einheiten, muss der Fluss von v nach w umgeplant werden.

Kleines Beispiel zur Flussumplanung



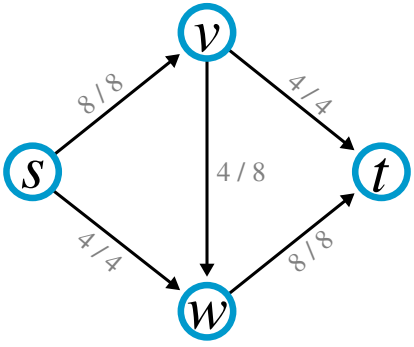
- Für eine Erhöhung um weitere 4 Einheiten, muss der Fluss von v nach w umgeplant werden.
- Der Fluss wird hier von 8 auf 4 vermindert (Kante $v \to w$ gegen Pfadrichtung)
- Auf den Kanten in Pfadrichtung ($s \to w$ und $v \to t$) wird der Fluss um 4 erhöht.

Kleines Beispiel zur Flussumplanung



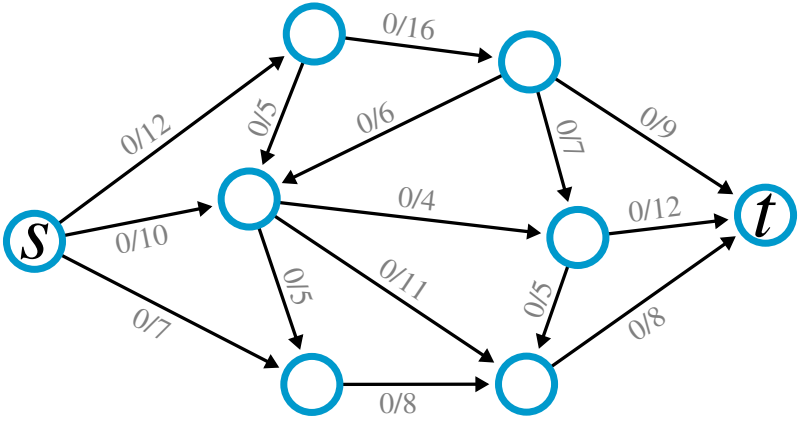
- Für eine Erhöhung um weitere 4 Einheiten, muss der Fluss von v nach w umgeplant werden.
- Der Fluss wird hier von 8 auf 4 vermindert (Kante $v \to w$ gegen Pfadrichtung)
- Auf den Kanten in Pfadrichtung ($s \to w$ und $v \to t$) wird der Fluss um 4 erhöht.
- Insgesamt entspricht dies einer Erhöhung des Flusses entlang des Pfades s–w–v–t um 4.

Kleines Beispiel zur Flussumplanung



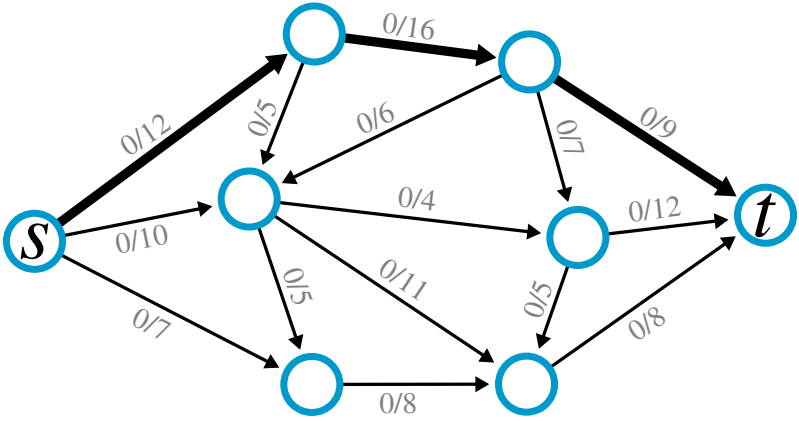
- Für eine Erhöhung um weitere 4 Einheiten, muss der Fluss von v nach w umgeplant werden.
- Der Fluss wird hier von 8 auf 4 vermindert (Kante $v \to w$ gegen Pfadrichtung)
- Auf den Kanten in Pfadrichtung ($s \to w$ und $v \to t$) wird der Fluss um 4 erhöht.
- Insgesamt entspricht dies einer Erhöhung des Flusses entlang des Pfades s–w–v–t um 4.

Vergrößernde Pfade

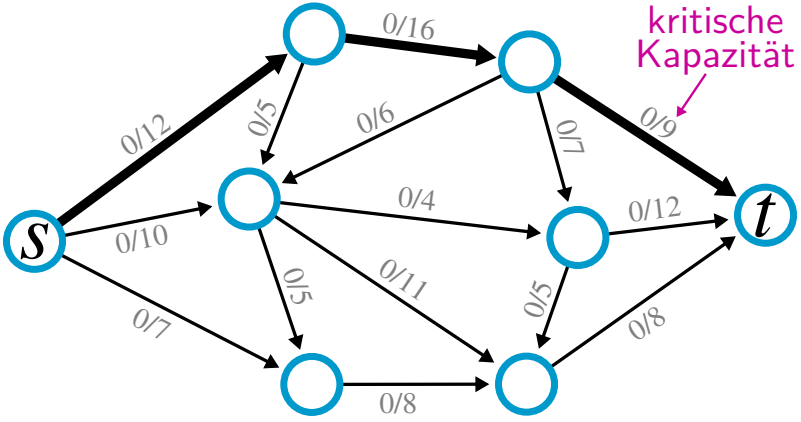


- Es gibt unterschiedliche Möglichkeiten vergrößernde Pfade zu wählen.
- Wir spielen wir eine Variante durch und fügen den Fluss des Pfades jeweils dem Fluss des Graphen hinzu.

Vergrößernde Pfade

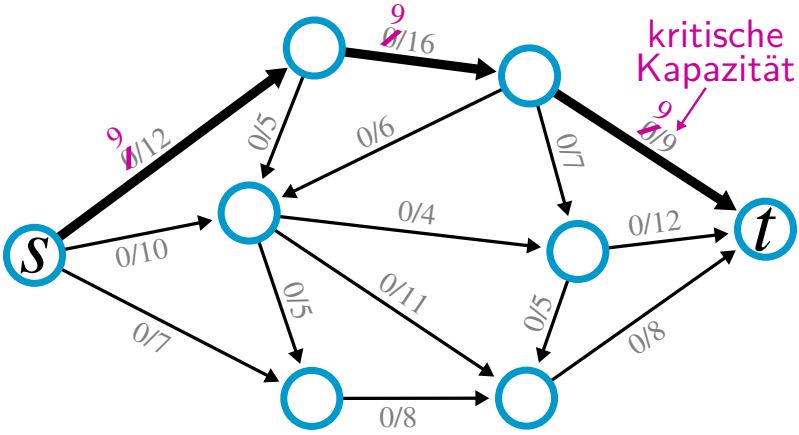


Vergrößernde Pfade

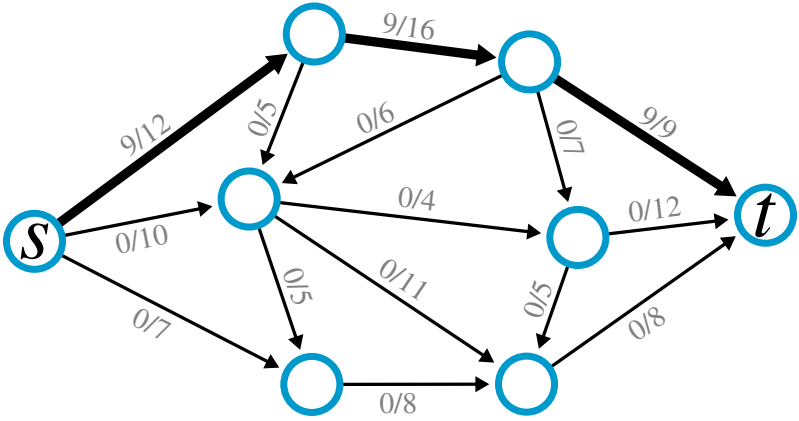


- Die **kritische Kapazität** (kritischer Wert) eines Pfades ist die kleinste freie Kapazität der benutzten Kanten.

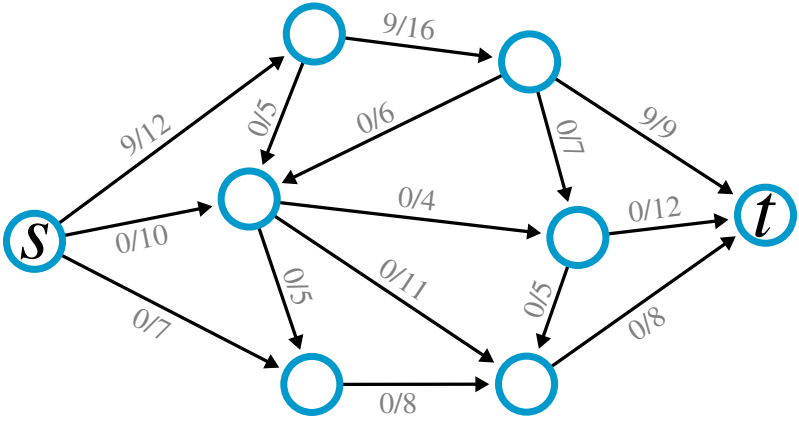
Vergrößernde Pfade



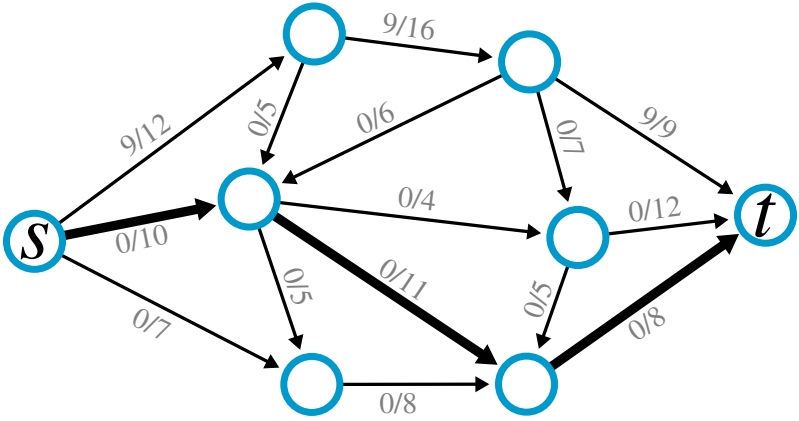
Vergrößernde Pfade



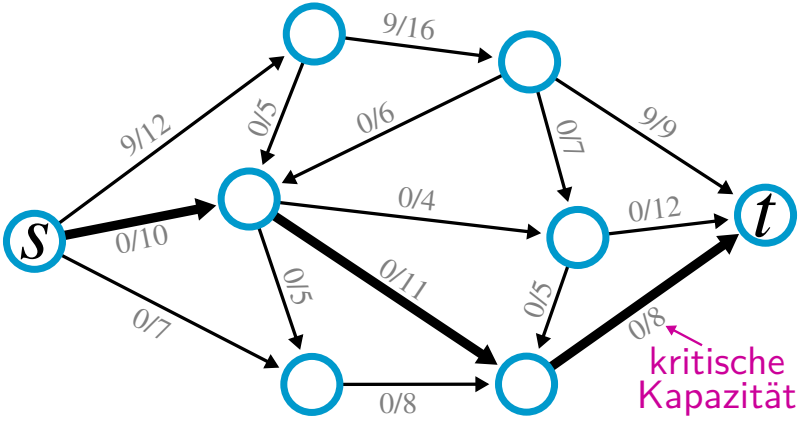
Vergrößernde Pfade



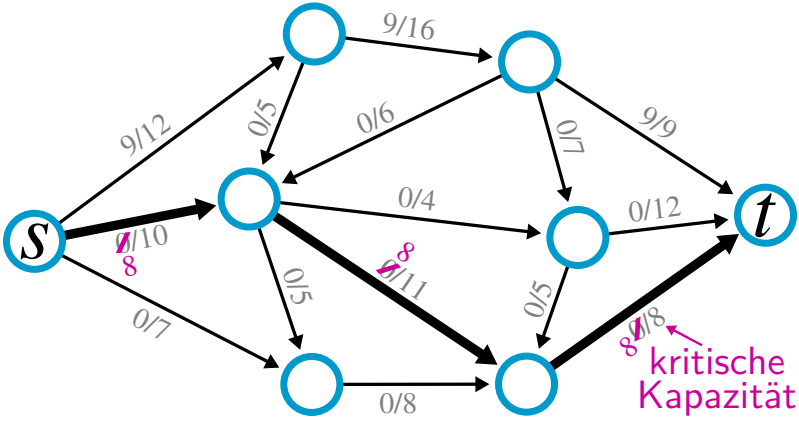
Vergrößernde Pfade



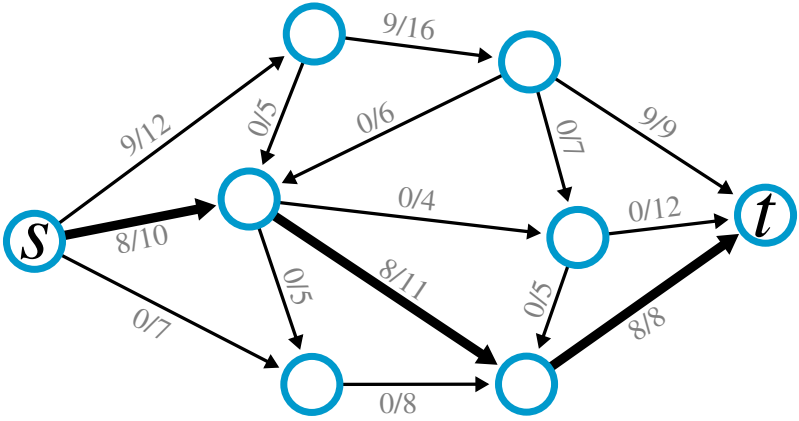
Vergrößernde Pfade



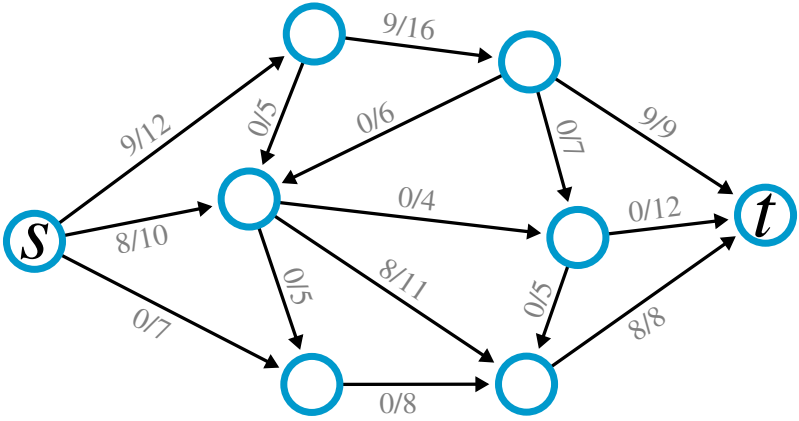
Vergrößernde Pfade



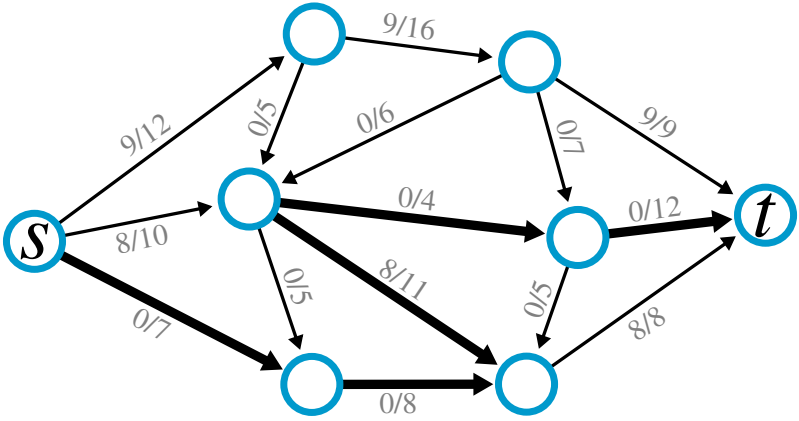
Vergrößernde Pfade



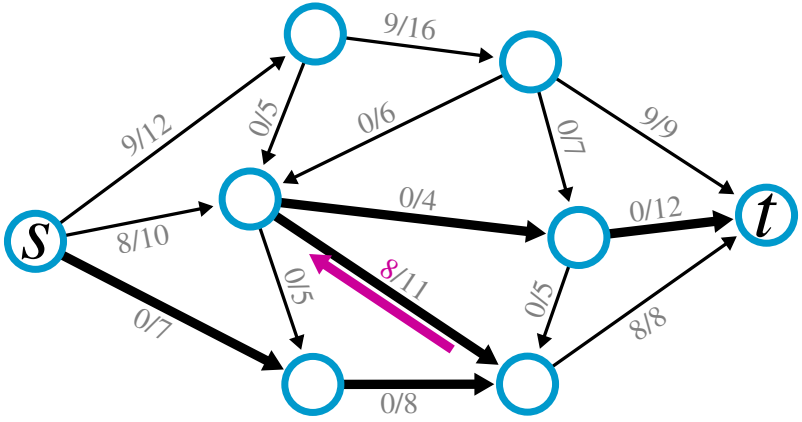
Vergrößernde Pfade



Vergrößernde Pfade

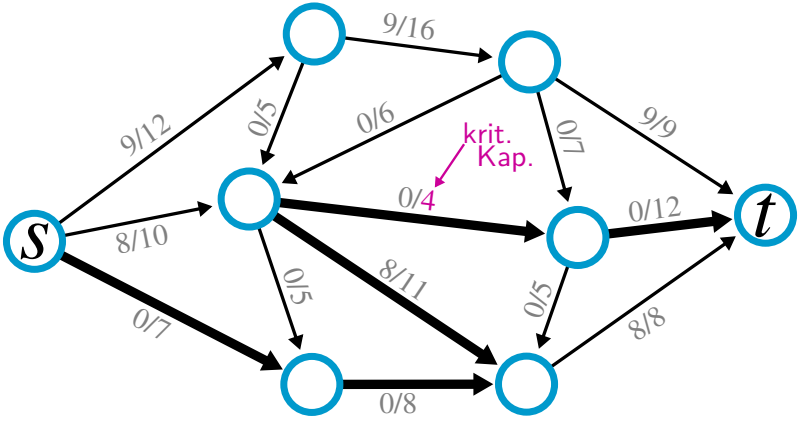


Vergrößernde Pfade

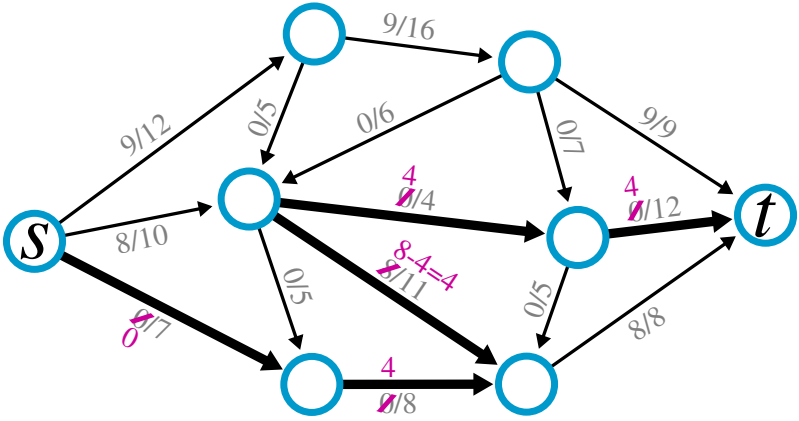


- Wenn eine Flusskante rückwärts durchlaufen wird, wird der Fluss der Pfadkante von dem Fluss der Flusskante abgezogen. Die freie Kapazität entspricht in diesem Fall also dem momentanen Fluss der Flusskante.
- Dies entspricht einer Umplanung eines früher gewählten Flusses.

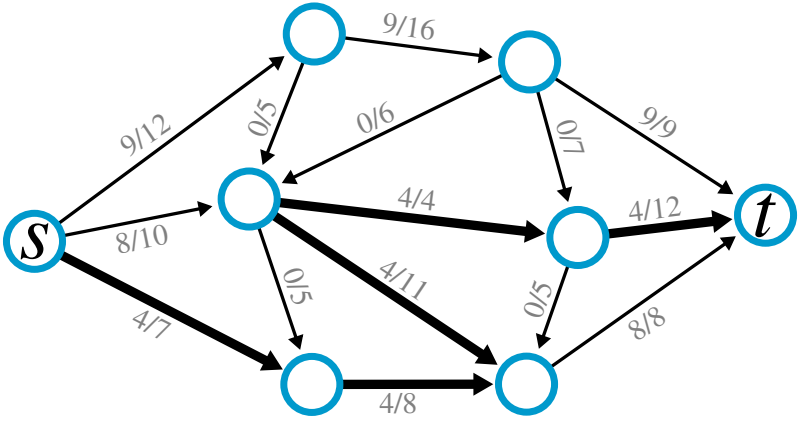
Vergrößernde Pfade



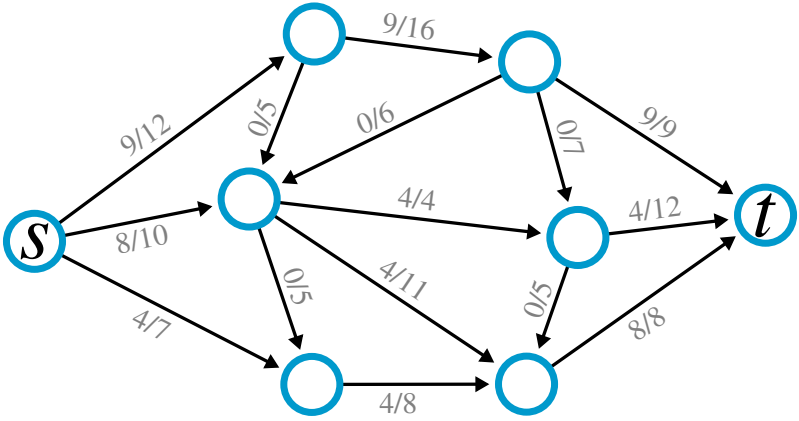
Vergrößernde Pfade



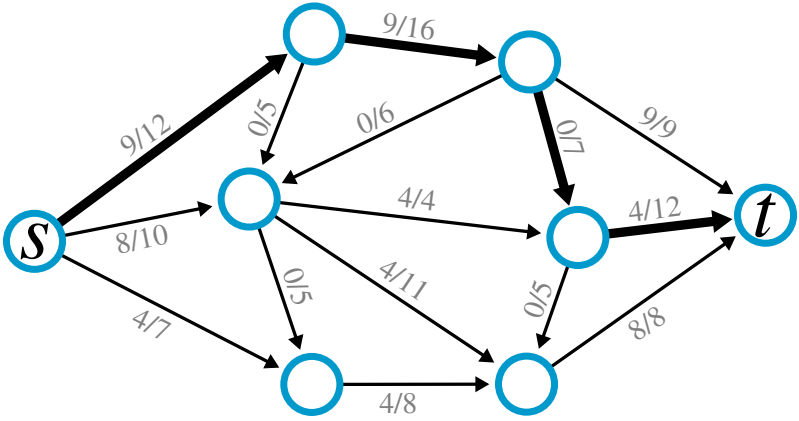
Vergrößernde Pfade



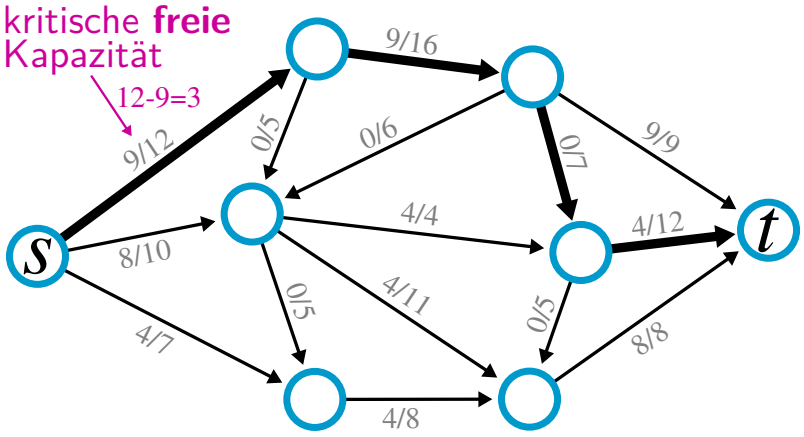
Vergrößernde Pfade



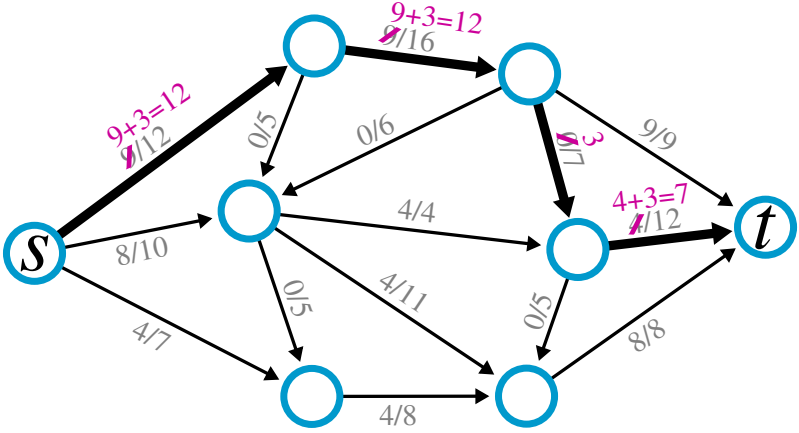
Vergrößernde Pfade



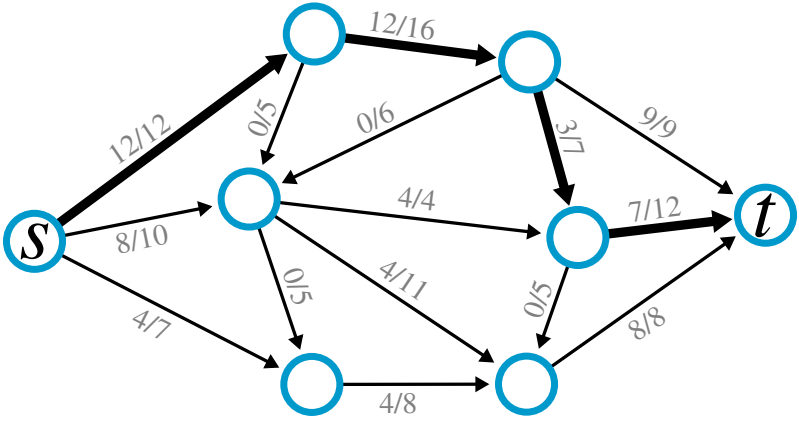
Vergrößernde Pfade



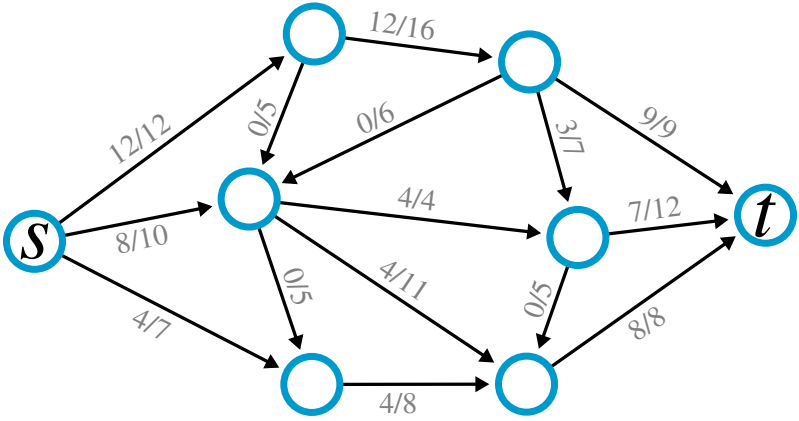
Vergrößernde Pfade



Vergrößernde Pfade



Vergrößernde Pfade

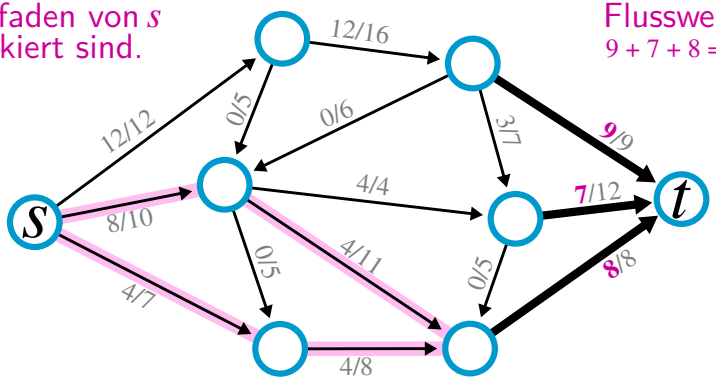


- Es gibt keine vergrößernden Pfade mehr.

Maximaler Fluss

- Wenn kein vergrößernder Pfad mehr existiert, ist der maximale Fluss gefunden (bisher intuitiv – Beweis folgt).
- Dies ist der Fall, wenn jeder Pfad von s nach t blockiert ist
  - durch eine Kante in Pfadrichtung ohne freie Kapazität $(f(v,w) = c(v,w))$ oder
  - durch eine Kante gegen Pfadrichtung ohne Fluss $(f(v,w) = 0)$.
- Dann ist Zufluss zu der Senke der maximale Fluss.

Kanten auf Pfaden von s die nicht blockiert sind.



Flusswert:
9 + 7 + 8 = **24**

Restgraphen

- Die algorithmische Behandlung von vergrößernden Pfaden wird durch die Einführung von **Restgraphen** (*residual graphs*) vereinfacht.
- Dieses Konzept integriert die beiden unterschiedlichen Anforderungen bezüglich Pfeilen in und gegen Pfadrichtung und beschreibt somit die Möglichkeiten zur Flussvergrößerung.

Restgraphen

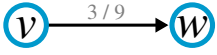
- Die algorithmische Behandlung von vergrößernden Pfaden wird durch die Einführung von **Restgraphen** (*residual graphs*) vereinfacht.
- Dieses Konzept integriert die beiden unterschiedlichen Anforderungen bezüglich Pfeilen in und gegen Pfadrichtung und beschreibt somit die Möglichkeiten zur Flussvergrößerung.
- Zu einem Flussgraph $G = (V, E)$ und Fluss f definieren wir einen gewichteten Digraphen $G_f = (V, E_f)$, genannt Restgraph (zu G, f), wobei die Gewichte die **Restkapazität** (*residual capacity*) sind:

$$rc(v,w) = \begin{cases} c(v,w) - f(v,w) & \text{falls } v{\to}w \in E \\ f(v,w) & \text{falls } w{\to}v \in E \end{cases}$$

- Es werden nur Kanten $v{\to}w$ berücksichtigt, deren Restkapazität $rc(v,w) > 0$ ist, also

$$E_f = \{v{\to}w \mid v,w \in V \ \ \& \ \ rc(v,w) > 0\}.$$

Flussgraph G



Restgraph G_f

Flüsse und ihre Restgraphen

- Für einen Pfeil $(v \to w \in G)$ des ursprünglichen Flussgraphen, gibt die Restkapazität
  - des Pfeils $v \to w \in G_f$ an, um wieviel der Fluss f vergrößert und
  - des Pfeils $w \to v \in G_f$ an, um wieviel der Fluss f verringert werden kann.

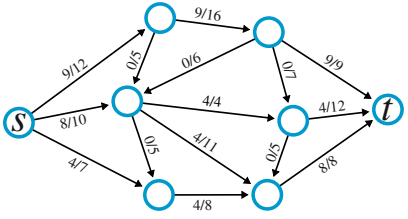
Flüsse und ihre Restgraphen

- Für einen Pfeil $(v \to w \in G)$ des ursprünglichen Flussgraphen, gibt die Restkapazität
  - des Pfeils $v \to w \in G_f$ an, um wieviel der Fluss f vergrößert und
  - des Pfeils $w \to v \in G_f$ an, um wieviel der Fluss f verringert werden kann.

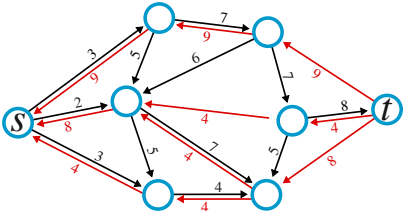
- Der kritische Wert eines Pfads im Restgraphen ist die kleinste Restkapazität seiner Kanten. Da der Restgraph nur Kanten $v \to w$ mit $rc(v,w) > 0$ enthält, ist der kritischer Wert immer > 0.

- Wenn es in dem Restgraphen G_f einen Pfad von s nach t gibt, kann der Fluss f entlang des Pfades um den kritischen Wert vergrößert werden.

Flussgraph G mit Fluss f



Restgraph G_f



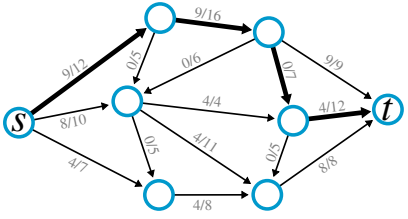
Flüsse und ihre Restgraphen

- Für einen Pfeil $(v \to w \in G)$ des ursprünglichen Flussgraphen, gibt die Restkapazität
  - des Pfeils $v \to w \in G_f$ an, um wieviel der Fluss f vergrößert und
  - des Pfeils $w \to v \in G_f$ an, um wieviel der Fluss f verringert werden kann.

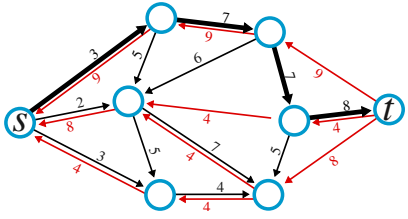
- Der kritische Wert eines Pfads im Restgraphen ist die kleinste Restkapazität seiner Kanten. Da der Restgraph nur Kanten $v \to w$ mit $rc(v,w) > 0$ enthält, ist der kritischer Wert immer > 0.

- Wenn es in dem Restgraphen G_f einen Pfad von s nach t gibt, kann der Fluss f entlang des Pfades um den kritischen Wert vergrößert werden.

Flussgraph G mit Fluss f

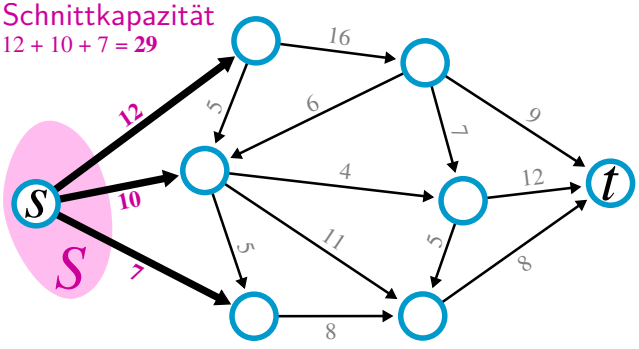


Restgraph G_f



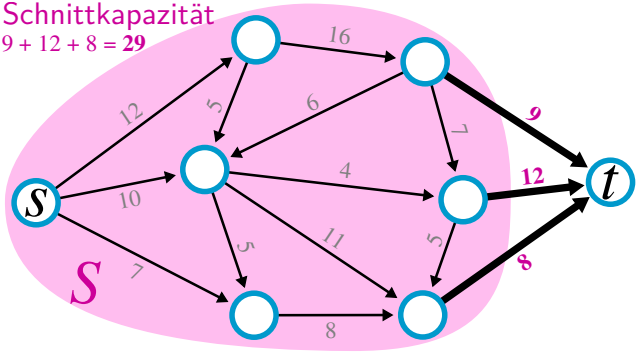
Schnitte und minimaler Schnitt

- Ein s von t trennender **Schnitt** (*cut*) teilt die Knoten eines Flussgraphen in zwei zusammenhängende, nicht-leere Teilmengen S und $T = V - S$, wobei die Quelle in S und die Senke in T ist: $s \in S$, $t \in T$.
- Die **Kapazität eines Schnittes** ist die Summe der Kapazitäten der kreuzenden Kanten die S verlassen. Kanten, die nach S hereinführen werden **nicht** gezählt.



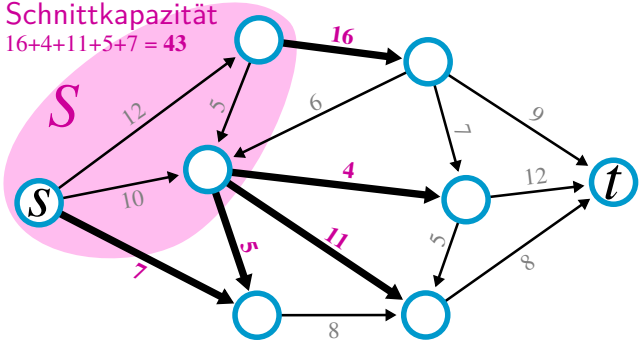
Schnitte und minimaler Schnitt

- Ein s von t trennender **Schnitt** (*cut*) teilt die Knoten eines Flussgraphen in zwei zusammenhängende, nicht-leere Teilmengen S und $T = V - S$, wobei die Quelle in S und die Senke in T ist: $s \in S$, $t \in T$.
- Die **Kapazität eines Schnittes** ist die Summe der Kapazitäten der kreuzenden Kanten die S verlassen. Kanten, die nach S hereinführen werden **nicht** gezählt.



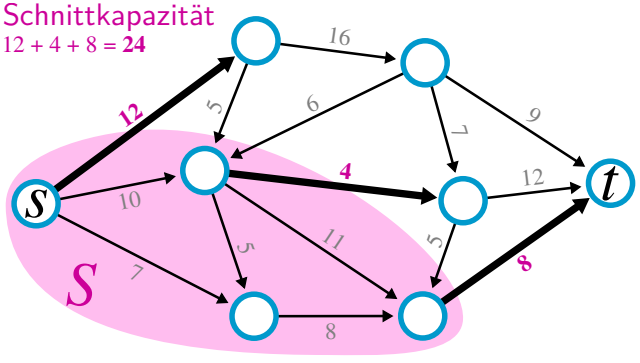
Schnitte und minimaler Schnitt

- Ein s von t trennender **Schnitt** (*cut*) teilt die Knoten eines Flussgraphen in zwei zusammenhängende, nicht-leere Teilmengen S und $T = V - S$, wobei die Quelle in S und die Senke in T ist: $s \in S$, $t \in T$.
- Die **Kapazität eines Schnittes** ist die Summe der Kapazitäten der kreuzenden Kanten die S verlassen. Kanten, die nach S hereinführen werden **nicht** gezählt.



Schnitte und minimaler Schnitt

- Ein s von t trennender **Schnitt** (*cut*) teilt die Knoten eines Flussgraphen in zwei zusammenhängende, nicht-leere Teilmengen S und $T = V - S$, wobei die Quelle in S und die Senke in T ist: $s \in S$, $t \in T$.
- Die **Kapazität eines Schnittes** ist die Summe der Kapazitäten der kreuzenden Kanten die S verlassen. Kanten, die nach S hereinführen werden **nicht** gezählt.



Herausforderung:
Finde einen Schnitt mit minimaler Kapazität (minimaler Schnitt, *mincut*)!

Bemerkung:
(Minimale) Schnitte werden für beliebige gewichtete Graphen betrachtet, nicht nur für Flussgraphen.

Fluss über einen Schnitt

- Wir definieren als **Fluss über einen Schnitt** $f(S)$ zu gegebenem Fluss f und Schnitt S die Summe über den Fluss aller kreuzenden Kanten. Dabei werden Kanten aus S positiv und Kanten nach S negativ gerechnet.

Fluss über einen Schnitt

- Wir definieren als **Fluss über einen Schnitt** $f(S)$ zu gegebenem Fluss f und Schnitt S die Summe über den Fluss aller kreuzenden Kanten. Dabei werden Kanten aus S positiv und Kanten nach S negativ gerechnet.
- Zur einfacheren Schreibweise in der Formeln schreiben wir auch $f(e)$ für den Fluss der Kante $e = v \to w$ und definieren die
- die Menge der kreuzenden Kanten, die nach S herein zeigen

$$E_S^{\leftarrow} = \{v \to w \in E \mid v \notin S \ \& \ w \in S\}$$

- und die Menge der kreuzenden Kanten, die aus S heraus zeigen

$$E_S^{\rightarrow} = \{v \to w \in E \mid v \in S \ \& \ w \notin S\}$$

- Mit diesen Definitionen erhalten wir die folgende Formel für den Fluss über den Schnitt S:

$$f(S) = \sum_{\substack{v \to w \in E \\ v \in S,\ w \notin S}} f(v,w) - \sum_{\substack{w \to v \in E \\ v \in S,\ w \notin S}} f(w,v) = \sum_{e \in E_S^{\rightarrow}} f(e) - \sum_{e \in E_S^{\leftarrow}} f(e)$$

Fluss über einen Schnitt

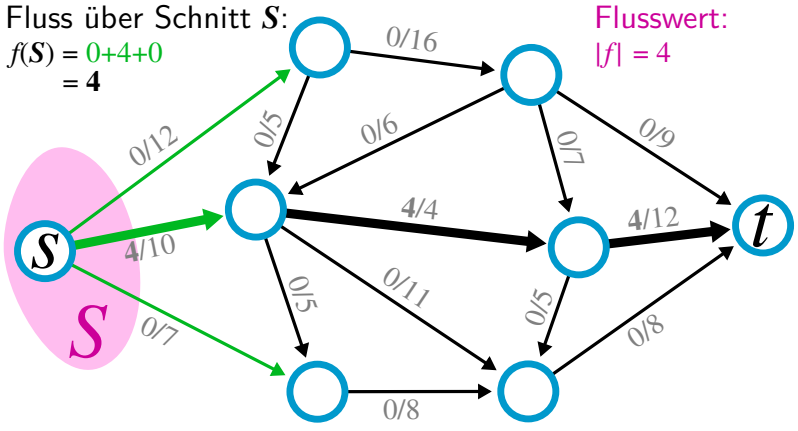
Schnitttheorem: Zusammenhang von Flüssen und Schnitten

Der Fluss ist über alle Schnitte derselbe. Er entspricht immer dem Flusswert.

Fluss über einen Schnitt

Schnitttheorem: Zusammenhang von Flüssen und Schnitten

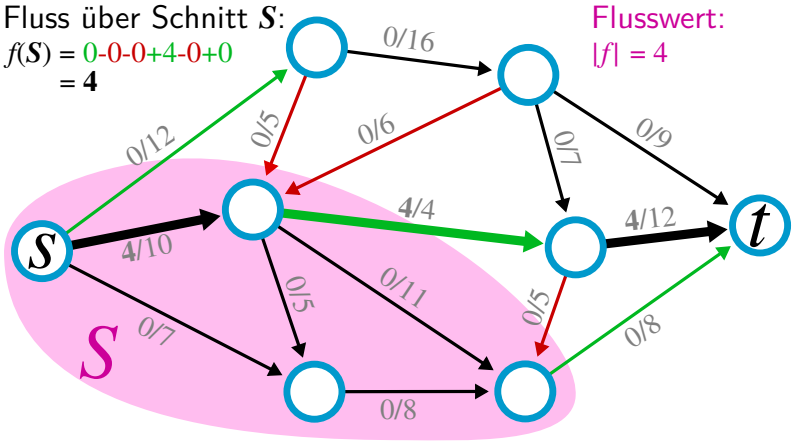
Der Fluss ist über alle Schnitte derselbe. Er entspricht immer dem Flusswert.



Fluss über einen Schnitt

Schnitttheorem: Zusammenhang von Flüssen und Schnitten

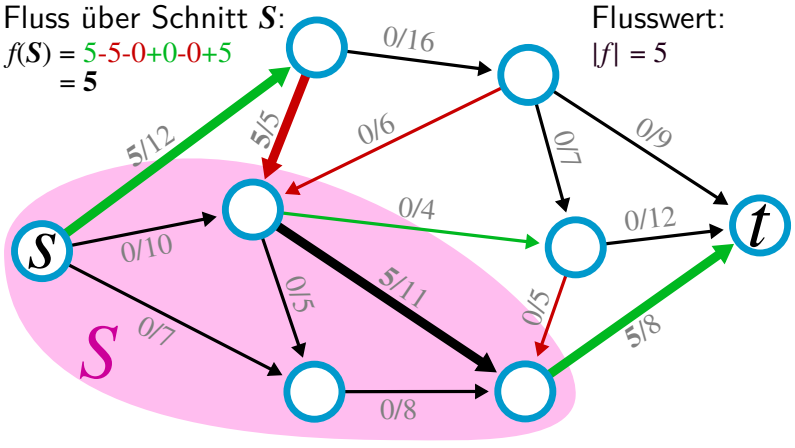
Der Fluss ist über alle Schnitte derselbe. Er entspricht immer dem Flusswert.



Fluss über einen Schnitt

Schnitttheorem: Zusammenhang von Flüssen und Schnitten

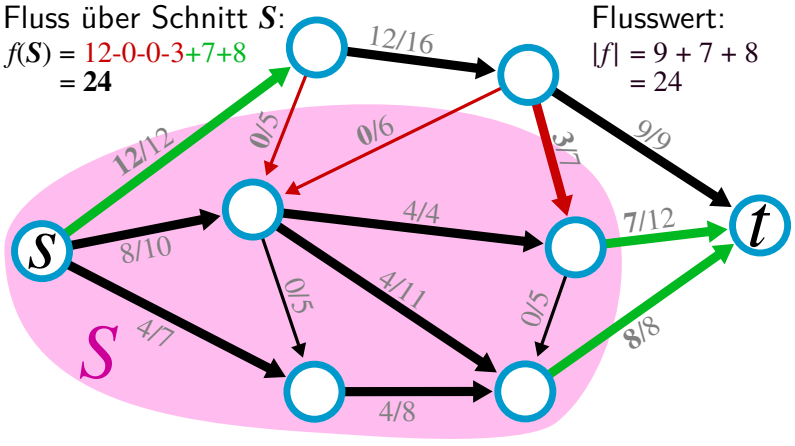
Der Fluss ist über alle Schnitte derselbe. Er entspricht immer dem Flusswert.



Fluss über einen Schnitt

Schnitttheorem: Zusammenhang von Flüssen und Schnitten

Der Fluss ist über alle Schnitte derselbe. Er entspricht immer dem Flusswert.



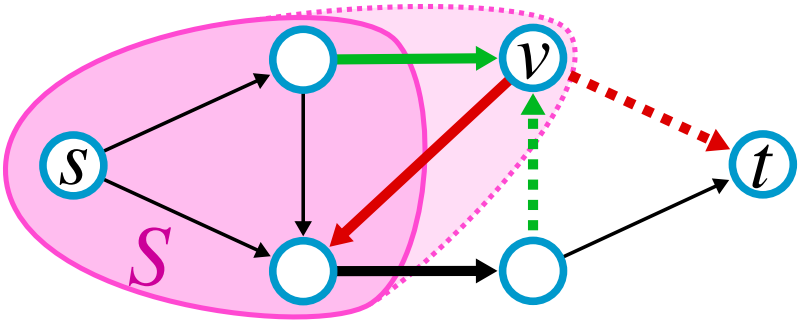
Beweis der Gleichheit des Flusses über alle Schnitte

- Wir beweisen durch Induktion nach $|\boldsymbol{T}|$, dass für alle Schnitte $(\boldsymbol{S}, \boldsymbol{T})$ gilt: $f(\boldsymbol{S}, \boldsymbol{T}) = |f|$.
- Für $|\boldsymbol{T}| = 1$ ist $\boldsymbol{T} = \{t\}$ und $\boldsymbol{S} = \boldsymbol{V} - \{t\}$. In diesem Fall entspricht der Flusswert per Definition dem Wert des Fluss über den Schnitt $(\boldsymbol{S}, \boldsymbol{T})$. Beides ist der Zufluss zu t, da t als Senke keine ausgehenden Kanten besitzt.
- Per IV wissen wir $f(\boldsymbol{S} + \{v\}, \boldsymbol{T}) = |f|$ und müssen zeigen, dass dies auch für $f(\boldsymbol{S}, \boldsymbol{T} + \{v\})$ gilt. Bei dem Übergang von Schnitt $(\boldsymbol{S} + \{v\}, \boldsymbol{T})$ zu $f(\boldsymbol{S}, \boldsymbol{T} + \{v\})$ passiert folgendes:

Beweis der Gleichheit des Flusses über alle Schnitte

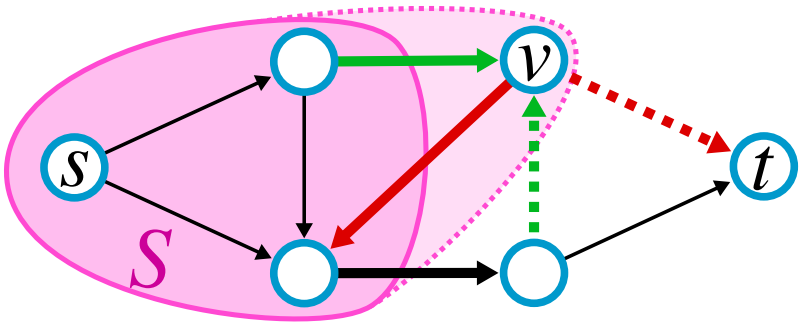
- Wir beweisen durch Induktion nach $|\boldsymbol{T}|$, dass für alle Schnitte $(\boldsymbol{S}, \boldsymbol{T})$ gilt: $f(\boldsymbol{S}, \boldsymbol{T}) = |f|$.
- Für $|\boldsymbol{T}| = 1$ ist $\boldsymbol{T} = \{t\}$ und $\boldsymbol{S} = \boldsymbol{V} - \{t\}$. In diesem Fall entspricht der Flusswert per Definition dem Wert des Fluss über den Schnitt $(\boldsymbol{S}, \boldsymbol{T})$. Beides ist der Zufluss zu t, da t als Senke keine ausgehenden Kanten besitzt.
- Per IV wissen wir $f(\boldsymbol{S} + \{v\}, \boldsymbol{T}) = |f|$ und müssen zeigen, dass dies auch für $f(\boldsymbol{S}, \boldsymbol{T} + \{v\})$ gilt. Bei dem Übergang von Schnitt $(\boldsymbol{S} + \{v\}, \boldsymbol{T})$ zu $f(\boldsymbol{S}, \boldsymbol{T} + \{v\})$ passiert folgendes:
- Es fällt weg (rot/grün gestrichelt): Fluss der Kanten von v nach $\boldsymbol{T}$ und der negativ gewichtete Fluss der Kanten von $\boldsymbol{T}$ nach v.
- Es kommt hinzu (rot/grün durchgezogen): Fluss der Kanten von $\boldsymbol{S}$ nach v und der negativ gewichtete Fluss der Kanten von v nach $\boldsymbol{S}$.

Beweis der Gleichheit des Flusses über alle Schnitte



- Es fällt weg (rot/grün gestrichelt): Fluss der Kanten von v nach $\boldsymbol{T}$ und der negativ gewichtete Fluss der Kanten von $\boldsymbol{T}$ nach v.
- Es kommt hinzu (rot/grün durchgezogen): Fluss der Kanten von $\boldsymbol{S}$ nach v und der negativ gewichtete Fluss der Kanten von v nach $\boldsymbol{S}$.

Beweis der Gleichheit des Flusses über alle Schnitte



- Es fällt weg (rot/grün gestrichelt): Fluss der Kanten von v nach $\boldsymbol{T}$ und der negativ gewichtete Fluss der Kanten von $\boldsymbol{T}$ nach v.
- Es kommt hinzu (rot/grün durchgezogen): Fluss der Kanten von $\boldsymbol{S}$ nach v und der negativ gewichtete Fluss der Kanten von v nach $\boldsymbol{S}$.
- In Summe kommt also der Zufluss nach v dazu (grün), und es wird der Abfluss von v abgezogen (rot). Nach der Flusserhaltungsbedingung (S. 3) ergibt dies 0. □

Beweis der Gleichheit des Flusses über alle Schnitte

- Wir beweisen durch Induktion nach $|\boldsymbol{T}|$, dass für alle Schnitte $(\boldsymbol{S}, \boldsymbol{T})$ gilt: $f(\boldsymbol{S}, \boldsymbol{T}) = |f|$.
- Für $|\boldsymbol{T}| = 1$ ist $\boldsymbol{T} = \{t\}$ und $\boldsymbol{S} = \boldsymbol{V} - \{t\}$. In diesem Fall entspricht der Flusswert per Definition dem Wert des Fluss über den Schnitt $(\boldsymbol{S}, \boldsymbol{T})$. Beides ist der Zufluss zu t, da t als Senke keine ausgehenden Kanten besitzt.
- Per IV wissen wir $f(\boldsymbol{S} + \{v\}, \boldsymbol{T}) = |f|$ und müssen zeigen, dass dies auch für $f(\boldsymbol{S}, \boldsymbol{T} + \{v\})$ gilt. Bei dem Übergang von Schnitt $(\boldsymbol{S} + \{v\}, \boldsymbol{T})$ zu $f(\boldsymbol{S}, \boldsymbol{T} + \{v\})$ passiert folgendes:
- Es fällt weg (rot/grün gestrichelt): Fluss der Kanten von v nach $\boldsymbol{T}$ und der negativ gewichtete Fluss der Kanten von $\boldsymbol{T}$ nach v.
- Es kommt hinzu (rot/grün durchgezogen): Fluss der Kanten von $\boldsymbol{S}$ nach v und der negativ gewichtete Fluss der Kanten von v nach $\boldsymbol{S}$.
- In Summe kommt also der Zufluss nach v dazu (grün), und es wird der Abfluss von v abgezogen (rot). Nach der Flusserhaltungsbedingung (S. 3) ergibt dies 0. □

Alternativer, rechnerischer Beweis

- Der Sachverhalt kann auch direkt, ohne Induktion, bewiesen werden.
- Wir nehmen die Definition des Fluss eines Schnittes und addieren den folgenden Term, der 0 ergibt, da beide Summen über alle Kanten innerhalb $\boldsymbol{T}$ laufen:

$$\sum_{\substack{v\to w\in \boldsymbol{E}\\ v\in \boldsymbol{T},\, w\in \boldsymbol{T}}} f(v,w) - \sum_{\substack{w\to v\in \boldsymbol{E}\\ v\in \boldsymbol{T},\, w\in \boldsymbol{T}}} f(w,v)$$

- Auf diese Weise erhalten wir nach Umsortieren der Summanden:

$$f(\boldsymbol{S},\boldsymbol{T}) = \sum_{\substack{v\to w\in \boldsymbol{E}\\ v\in \boldsymbol{S},\, w\in \boldsymbol{T}}} f(v,w) - \sum_{\substack{w\to v\in \boldsymbol{E}\\ v\in \boldsymbol{S},\, w\in \boldsymbol{T}}} f(w,v) \quad \text{Definition Fluss über Schnitt}$$

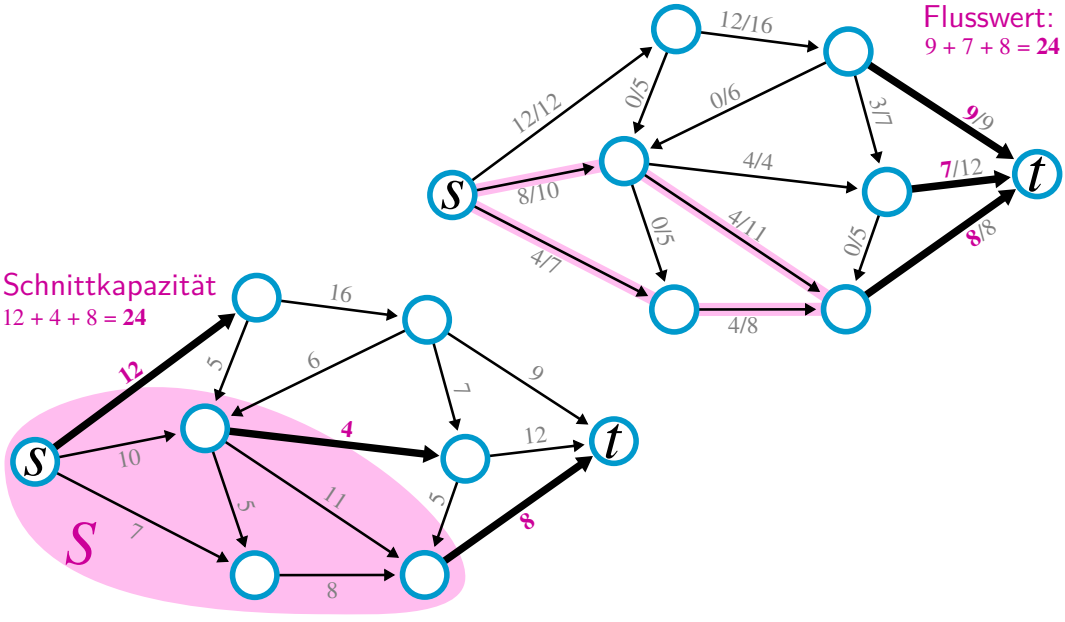
$$= \sum_{\substack{v\to w\in \boldsymbol{E}\\ v\in \boldsymbol{V},\, w\in \boldsymbol{T}}} f(v,w) - \sum_{\substack{w\to v\in \boldsymbol{E}\\ v\in \boldsymbol{V},\, w\in \boldsymbol{T}}} f(w,v)$$

Addition von obigem Term, $\boldsymbol{V} = \boldsymbol{S} \cup \boldsymbol{T}$ und Umsortieren

$$= \sum_{w\in \boldsymbol{T}} (\text{Zufluss zu } w - \text{Abfluss von } w) = |f|$$

Wegen Flusserhaltung S. 3 bleibt nur der Term für t.

Zusammenhang: Maximaler Fluss und minimaler Schnitt



Maximaler Fluss und minimaler Schnitt durch vergrößernde Pfade

Vergrößernde Pfade und maximaler Fluss

Ein Fluss f ist genau dann maximal, wenn es keine vergrößernden Pfade gibt.

Maximaler Fluss und minimaler Schnitt durch vergrößernde Pfade

Vergrößernde Pfade und maximaler Fluss

Ein Fluss f ist genau dann maximal, wenn es keine vergrößernden Pfade gibt.

- Der Beweis erfolgt dadurch, dass die Äquivalenz der folgenden drei Aussagen für einen Fluss f bewiesen wird:

1. Es gibt einen Schnitt, dessen Kapazität mit dem Wert von f übereinstimmt.
2. f ist ein maximaler Fluss.
3. Es gibt keinen vergrößernden Pfad für f.

Maximaler Fluss und minimaler Schnitt durch vergrößernde Pfade

Vergrößernde Pfade und maximaler Fluss

Ein Fluss f ist genau dann maximal, wenn es keine vergrößernden Pfade gibt.

- Der Beweis erfolgt dadurch, dass die Äquivalenz der folgenden drei Aussagen für einen Fluss f bewiesen wird:

1 Es gibt einen Schnitt, dessen Kapazität mit dem Wert von f übereinstimmt.

2 f ist ein maximaler Fluss.

3 Es gibt keinen vergrößernden Pfad für f.

- Die Äquivalenz von **1** und **2** ergibt auch folgenden Sachverhalt:

Maximaler Fluss und minimaler Schnitt

Der Wert des maximalen Flusses entspricht der Kapazität des minimalen Schnittes.

Beweis des Satzes über vergrößernde Pfade

1 Es gibt einen Schnitt mit $c(\boldsymbol{S}, \boldsymbol{T}) = |f| \Rightarrow$ **2** f ist maximaler Fluss

- **Beweis.** Für jeden Fluss f' gilt: $|f'| = f'(\boldsymbol{S}, \boldsymbol{T}) \leq c(\boldsymbol{S}, \boldsymbol{T}) = |f|$. Also ist der Flusswert $|f|$ maximal.

Beweis des Satzes über vergrößernde Pfade

1 Es gibt einen Schnitt mit $c(\boldsymbol{S},\boldsymbol{T}) = |f| \Rightarrow$ **2** f ist maximaler Fluss

- **Beweis.** Für jeden Fluss f' gilt: $|f'| = f'(\boldsymbol{S},\boldsymbol{T}) \leq c(\boldsymbol{S},\boldsymbol{T}) = |f|$. Also ist der Flusswert $|f|$ maximal.

2 f ist maximaler Schnitt $\Rightarrow$ **3** Es gibt keinen vergrößernden Pfad für f

- **Beweis.** Wir nehmen an, dass es einen vergrößernden Pfad für f gibt. Dann kann der Fluss entlang dieses Pfads um den kritischen Wert vergrößert werden, im Widerspruch zur Annahme, dass der Fluss f maximal ist.

Beweis des Satzes über vergrößernde Pfade

1 Es gibt einen Schnitt mit $c(\boldsymbol{S},\boldsymbol{T}) = |f|$ $\Rightarrow$ **2** f ist maximaler Fluss

- **Beweis.** Für jeden Fluss f' gilt: $|f'| = f'(\boldsymbol{S},\boldsymbol{T}) \leq c(\boldsymbol{S},\boldsymbol{T}) = |f|$. Also ist der Flusswert $|f|$ maximal.

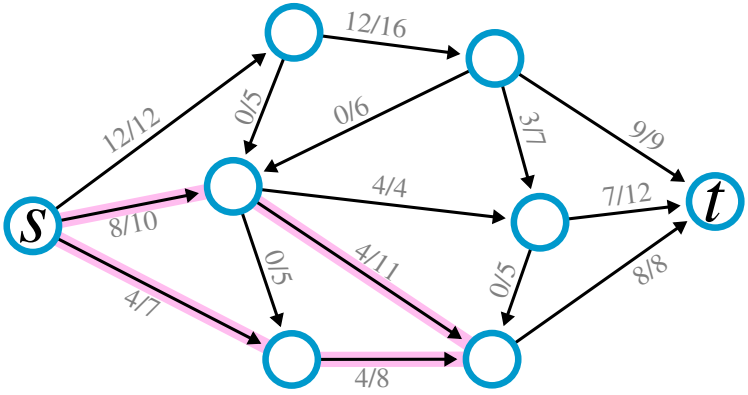
2 f ist maximaler Schnitt $\Rightarrow$ **3** Es gibt keinen vergrößernden Pfad für f

- **Beweis.** Wir nehmen an, dass es einen vergrößernden Pfad für f gibt. Dann kann der Fluss entlang dieses Pfads um den kritischen Wert vergrößert werden, im Widerspruch zur Annahme, dass der Fluss f maximal ist.

3 Kein vergrößernder Pfad für f $\Rightarrow$ **1** Es gibt Schnitt mit $c(\boldsymbol{S},\boldsymbol{T}) = |f|$

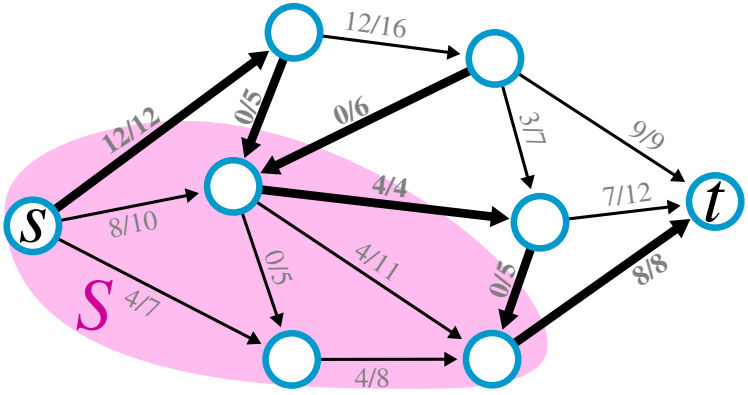
- **Beweis.** Wenn es keinen vergrößernden Pfad für f gibt, definiert die Menge $\boldsymbol{S}$ aller Knoten v, die von s im Restgraphen erreicht werden können, einen Schnitt. Nach dieser Definition sind gilt $rc(v,w) = 0$ für alle kreuzenden Kanten $v \to w$, also gilt $c(\boldsymbol{S},\boldsymbol{T}) = f(\boldsymbol{S},\boldsymbol{T})$. Weiter folgt mit dem Fluss-Lemma (S. 14): $c(\boldsymbol{S},\boldsymbol{T}) = f(\boldsymbol{S},\boldsymbol{T}) = |f|$. □

Illustration zum Beweis $3 \Rightarrow 1$



- Die von s im Restgraphen erreichbaren Pfade definieren einen Schnitt S.

Illustration zum Beweis $3 \Rightarrow 1$



- Die von s im Restgraphen erreichbaren Pfade definieren einen Schnitt S.
- Alle S verlassenden kreuzenden Kanten müssen voll gefüllt sein.
- Alle S betretenden kreuzenden Kanten müssen leer sein.
- Daher stimmt die Kapazität von S mit dem Fluss über S überein.

Allgemeine Methode zum Identifizieren des maximalen Flusses

Von **Ford-Fulkerson** wurde die Technik der vergrößernden Pfade entwickelt, um eine allgemeine Methode zum Identifizieren des maximalen Flusses in Flussgraphen anzugeben:

```
1 for each e in E
2   f(e) ←0
3 end
4 G_f ←Restgraph von G
5 while es gibt einen Pfad p in G_f do
6   cv ←min { rc(e) | e liegt auf Pfad p in G_f }
7   vergrößere f entlang p um cv
8   aktualisiere G_f
9 end
```

- Mit "*vergrößere f entlang p um cv*" ist das auf Seite 6 beschriebene Verfahren der vergrößernden Pfade gemeint:
  - Auf Kanten in Richtung des Pfades p wird der Fluss um cv erhöht und
  - auf Kanten gegen Richtung des Pfades p wird der Fluss um cv reduziert.

Korrektheit und Laufzeit der Ford-Fulkerson Methode

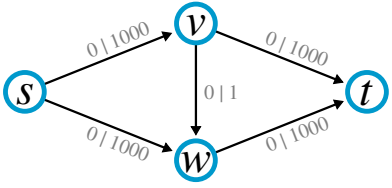
- **Korrektheit:** Wenn die allgemeine Ford-Fulkerson Methode terminiert, wissen wir nach dem Satz über vergrößernde Pfade (Seite 18), dass das Ergebnis ein maximaler Fluss ist.

Korrektheit und Laufzeit der Ford-Fulkerson Methode

- **Korrektheit:** Wenn die allgemeine Ford-Fulkerson Methode terminiert, wissen wir nach dem Satz über vergrößernde Pfade (Seite 18), dass das Ergebnis ein maximaler Fluss ist.
- Bevor wir die Laufzeit diskutieren, die die Terminierung impliziert, besprechen wir Beispiele, die der Methode Schwierigkeiten bereiten.
- Dabei ist zu beachten, dass bisher keine Strategie zur Auswahl der vergrößernden Pfade spezifiziert wurde.
- Die Beispiele beruhen auf einer 'unglücklichen' Reihenfolge.

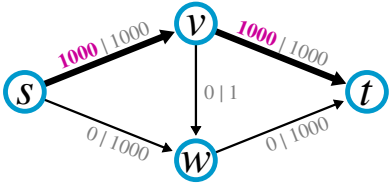
Kleiner Flussgraph mit potenziell langer Laufzeit

- Der abgebildete Flussgraph hat den maximalen Fluss von $|f^*| = 2.000$, der durch die Kombination der beiden Pfade $s \to v \to t$ und $s \to w \to t$ mit jeweils 1.000 Einheiten erreicht wird.



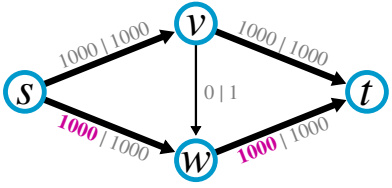
Kleiner Flussgraph mit potenziell langer Laufzeit

- Der abgebildete Flussgraph hat den maximalen Fluss von $|f^*| = 2.000$, der durch die Kombination der beiden Pfade $s \to v \to t$ und $s \to w \to t$ mit jeweils 1.000 Einheiten erreicht wird.



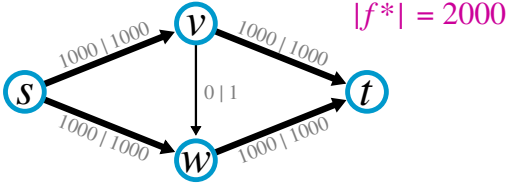
Kleiner Flussgraph mit potenziell langer Laufzeit

- Der abgebildete Flussgraph hat den maximalen Fluss von $|f^*| = 2.000$, der durch die Kombination der beiden Pfade $s \to v \to t$ und $s \to w \to t$ mit jeweils 1.000 Einheiten erreicht wird.



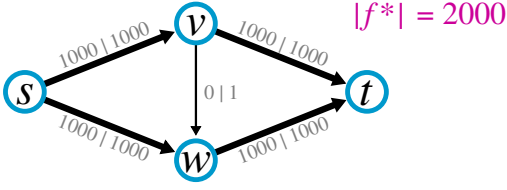
Kleiner Flussgraph mit potenziell langer Laufzeit

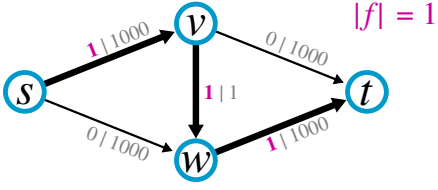
- Der abgebildete Flussgraph hat den maximalen Fluss von $|f^*| = 2.000$, der durch die Kombination der beiden Pfade $s \to v \to t$ und $s \to w \to t$ mit jeweils 1.000 Einheiten erreicht wird.



Kleiner Flussgraph mit potenziell langer Laufzeit

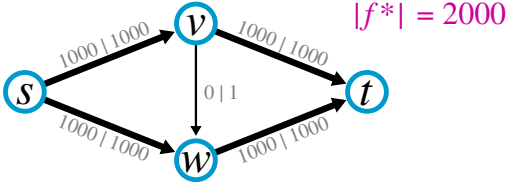
- Der abgebildete Flussgraph hat den maximalen Fluss von $|f^*| = 2.000$, der durch die Kombination der beiden Pfade $s \to v \to t$ und $s \to w \to t$ mit jeweils 1.000 Einheiten erreicht wird.
- Eine unglückliche Wahl der vergrößernden Pfade ist ein Wechsel von $s \to v \to w \to t$ und $s \to w \to v \to t$.
- Diese Pfade haben beide den kritischen Wert 1, so dass insgesamt 2.000 Iterationen nötig sind, um den maximalen Fluss $|f^*|$ zu erzeugen.

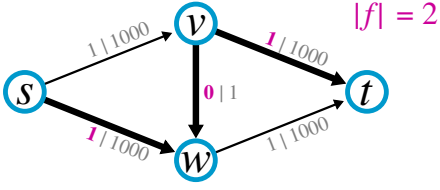




Kleiner Flussgraph mit potenziell langer Laufzeit

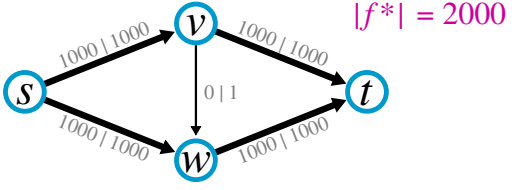
- Der abgebildete Flussgraph hat den maximalen Fluss von $|f^*| = 2.000$, der durch die Kombination der beiden Pfade $s \to v \to t$ und $s \to w \to t$ mit jeweils 1.000 Einheiten erreicht wird.
- Eine unglückliche Wahl der vergrößernden Pfade ist ein Wechsel von $s \to v \to w \to t$ und $s \to w \to v \to t$.
- Diese Pfade haben beide den kritischen Wert 1, so dass insgesamt 2.000 Iterationen nötig sind, um den maximalen Fluss $|f^*|$ zu erzeugen.

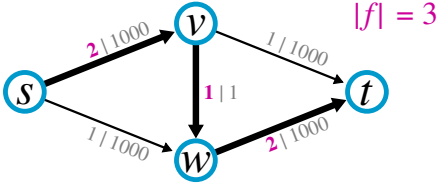




Kleiner Flussgraph mit potenziell langer Laufzeit

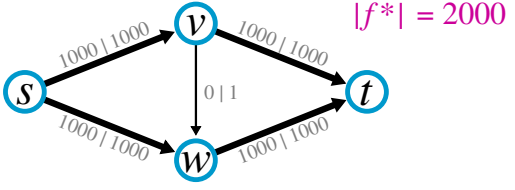
- Der abgebildete Flussgraph hat den maximalen Fluss von $|f^*| = 2.000$, der durch die Kombination der beiden Pfade $s \to v \to t$ und $s \to w \to t$ mit jeweils 1.000 Einheiten erreicht wird.
- Eine unglückliche Wahl der vergrößernden Pfade ist ein Wechsel von $s \to v \to w \to t$ und $s \to w \to v \to t$.
- Diese Pfade haben beide den kritischen Wert 1, so dass insgesamt 2.000 Iterationen nötig sind, um den maximalen Fluss $|f^*|$ zu erzeugen.

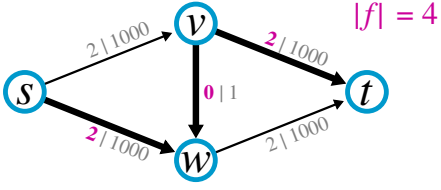




Kleiner Flussgraph mit potenziell langer Laufzeit

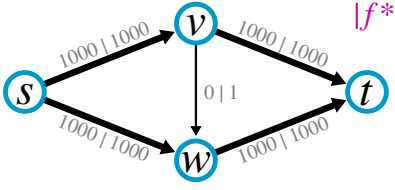
- Der abgebildete Flussgraph hat den maximalen Fluss von $|f^*| = 2.000$, der durch die Kombination der beiden Pfade $s \to v \to t$ und $s \to w \to t$ mit jeweils 1.000 Einheiten erreicht wird.
- Eine unglückliche Wahl der vergrößernden Pfade ist ein Wechsel von $s \to v \to w \to t$ und $s \to w \to v \to t$.
- Diese Pfade haben beide den kritischen Wert 1, so dass insgesamt 2.000 Iterationen nötig sind, um den maximalen Fluss $|f^*|$ zu erzeugen.



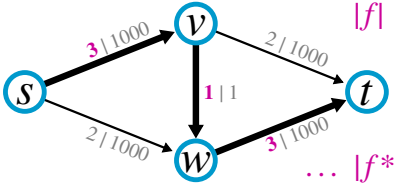


Kleiner Flussgraph mit potenziell langer Laufzeit

- Der abgebildete Flussgraph hat den maximalen Fluss von $|f^*| = 2.000$, der durch die Kombination der beiden Pfade $s \to v \to t$ und $s \to w \to t$ mit jeweils 1.000 Einheiten erreicht wird.
- Eine unglückliche Wahl der vergrößernden Pfade ist ein Wechsel von $s \to v \to w \to t$ und $s \to w \to v \to t$.
- Diese Pfade haben beide den kritischen Wert 1, so dass insgesamt 2.000 Iterationen nötig sind, um den maximalen Fluss $|f^*|$ zu erzeugen.
- Dieses Beispiel zeigt auch, dass die Laufzeit der Ford-Fulkerson Methode, sofern kein geeignetes Verfahren zur Pfadauswahl angegeben wird, nicht nur von der Struktur des Graph, sondern auch von seinen Kapazitäten abhängen kann.



$|f^*| = 2000$



$|f| = 5$

$\ldots \; |f^*| = 2000$

Kleiner Graph ohne Terminierung

- Es kommt noch schlimmer. Es gibt keine Garantie, dass die Ford-Fulkerson Methode überhaupt terminiert.
- Diese ungünstigen Fälle können allerdings nur auftreten, wenn es unter den Kapazitäten des Flussgraphen irrationale Zahlen gibt und die vergrößernden Pfade ungünstig gewählt werden.
- Ein entsprechendes Beispiel wird im Anhang auf Seite 43 diskutiert.

Kleiner Graph ohne Terminierung

- Es kommt noch schlimmer. Es gibt keine Garantie, dass die Ford-Fulkerson Methode überhaupt terminiert.
- Diese ungünstigen Fälle können allerdings nur auftreten, wenn es unter den Kapazitäten des Flussgraphen irrationale Zahlen gibt und die vergrößernden Pfade ungünstig gewählt werden.
- Ein entsprechendes Beispiel wird im Anhang auf Seite 43 diskutiert.
- Um die Laufzeit der allgemeinen Ford-Fulkerson Methode zu bestimmen, beschränken wir uns daher auf rationale Kapazität.
- Wir setzen sogar voraus, dass alle Kapazitäten des Flussgraphs in $\mathbb{N}^{>0}$ sind.
  Beliebige rationale Zahlen können mit dem KGV aller Nenner multipliziert werden, um einen äquivalenten Flussgraphen mit Kapazitäten in $\mathbb{N}^{>0}$ zu definieren.

Kleiner Graph ohne Terminierung

- Es kommt noch schlimmer. Es gibt keine Garantie, dass die Ford-Fulkerson Methode überhaupt terminiert.
- Diese ungünstigen Fälle können allerdings nur auftreten, wenn es unter den Kapazitäten des Flussgraphen irrationale Zahlen gibt und die vergrößernden Pfade ungünstig gewählt werden.
- Ein entsprechendes Beispiel wird im Anhang auf Seite 43 diskutiert.
- Um die Laufzeit der allgemeinen Ford-Fulkerson Methode zu bestimmen, beschränken wir uns daher auf rationale Kapazität.
- Wir setzen sogar voraus, dass alle Kapazitäten des Flussgraphs in $\mathbb{N}^{>0}$ sind.
  Beliebige rationale Zahlen können mit dem KGV aller Nenner multipliziert werden, um einen äquivalenten Flussgraphen mit Kapazitäten in $\mathbb{N}^{>0}$ zu definieren.
- Das vorige Beispiel (Graph mit langer Laufzeit, S. 23) zeigt, dass die Laufzeit nicht nur von der Größe des Graphen, sondern auch von den Kapazitäten abhängen kann.

Laufzeit der Ford-Fulkerson Methode

Die Ford-Fulkerson Methode benötigt für einen Flussgraphen, dessen Kapazitäten natürliche Zahlen sind, eine Laufzeit in $\boldsymbol{O}(E|f^*|)$, wobei f^* der maximale Fluss ist.

Beweis.

- Die Restkapazitäten sind immer ganze Zahlen, also auch der kritische Wert jedes vergrößernden Pfades.

Laufzeit der Allgemeinen Ford-Fulkerson Methode

Laufzeit der Ford-Fulkerson Methode

Die Ford-Fulkerson Methode benötigt für einen Flussgraphen, dessen Kapazitäten natürliche Zahlen sind, eine Laufzeit in $\boldsymbol{O}(E|f^*|)$, wobei f^* der maximale Fluss ist.

Beweis.

- Die Restkapazitäten sind immer ganze Zahlen, also auch der kritische Wert jedes vergrößernden Pfades.
- Der Fluss wird bei jeder Iteration der *while*-Schleife um den kritischen Wert erhöht, also um mindestens 1. Daher wird der maximale Fluss nach höchstens $|f^*|$ Durchläufen der *while*-Schleife erreicht.

Laufzeit der Allgemeinen Ford-Fulkerson Methode

Laufzeit der Ford-Fulkerson Methode

Die Ford-Fulkerson Methode benötigt für einen Flussgraphen, dessen Kapazitäten natürliche Zahlen sind, eine Laufzeit in $\boldsymbol{O}(E|f^*|)$, wobei f^* der maximale Fluss ist.

Beweis.

- Die Restkapazitäten sind immer ganze Zahlen, also auch der kritische Wert jedes vergrößernden Pfades.
- Der Fluss wird bei jeder Iteration der *while*-Schleife um den kritischen Wert erhöht, also um mindestens 1. Daher wird der maximale Fluss nach höchstens $|f^*|$ Durchläufen der *while*-Schleife erreicht.
- In jeder Iteration werden $\boldsymbol{O}(V + E_{\boldsymbol{R}}) = \boldsymbol{O}(E)$ Schritte benötigt, um einen Pfad im Restgraphen zu finden (z.B. durch Tiefensuche). Die Zeit, um die Restkapazität in Zeile 5 zu bestimmen und den Fluss in Zeile 6 zu vergrößern ist linear in $\boldsymbol{E}$.
- Insgesamt ergibt sich eine Laufzeit in $\boldsymbol{O}(E|f^*|)$. □

Eine spezifische Wahl der vergrößernden Pfade

- Alternativ kann die Laufzeit der allgemeinen Ford-Fulkerson Methode als $\boldsymbol{O}(EVC)$ angegeben werden, wobei C eine obere Schranke für die Kapazitäten ist, da $|f^*| \leq VC$.

Eine spezifische Wahl der vergrößernden Pfade

- Alternativ kann die Laufzeit der allgemeinen Ford-Fulkerson Methode als $\boldsymbol{O}(EVC)$ angegeben werden, wobei C eine obere Schranke für die Kapazitäten ist, da $|f^*| \leq VC$.
- Um eine Laufzeitschranke zu erzielen, die nur von der Größe des Graphen abhängt, muss man eine spezielle Strategie zur Auswahl des vergrößernden Pfades (= nicht-leerer Pfad im Restgraphen) anwenden.
- Folgende Strategien scheinen plausibel:

Eine spezifische Wahl der vergrößernden Pfade

- Alternativ kann die Laufzeit der allgemeinen Ford-Fulkerson Methode als $\boldsymbol{O}(EVC)$ angegeben werden, wobei C eine obere Schranke für die Kapazitäten ist, da $|f^*| \leq VC$.
- Um eine Laufzeitschranke zu erzielen, die nur von der Größe des Graphen abhängt, muss man eine spezielle Strategie zur Auswahl des vergrößernden Pfades (= nicht-leerer Pfad im Restgraphen) anwenden.
- Folgende Strategien scheinen plausibel:
  - Wähle einen vergrößernden Pfad mit wenigen Kanten
  - Wähle einen vergrößernden Pfad mit großem Fluss (großem kritischen Wert)

Der Edmonds-Karp Algorithmus (Pfad mit wenigen Kanten)

- Der **Edmonds-Karp Algorithmus** wählt als vergrößernden Pfad in der Ford-Fulkerson Methode einen Pfad, der die wenigsten Kanten hat.
- Man fängt mit einem leeren Fluss f an. Der Fluss wird iterativ vergrößert.
- Unter allen Pfaden von s nach t im Restgraphen $\boldsymbol{G}_f$ wird ein Pfad p mit der geringsten Anzahl von Kanten ausgesucht.

Der Edmonds-Karp Algorithmus (Pfad mit wenigen Kanten)

- Der **Edmonds-Karp Algorithmus** wählt als vergrößernden Pfad in der Ford-Fulkerson Methode einen Pfad, der die wenigsten Kanten hat.
- Man fängt mit einem leeren Fluss f an. Der Fluss wird iterativ vergrößert.
- Unter allen Pfaden von s nach t im Restgraphen $\boldsymbol{G}_f$ wird ein Pfad p mit der geringsten Anzahl von Kanten ausgesucht.
- Dies kann z.B. durch Breitensuche im Restgraphen geschehen.
- Dann wird der kritische Wert des Pfades p bestimmt. Dies ist die kleinste Restkapazität (Gewichte im Restgraphen $\boldsymbol{G}_f$) der Kanten des Pfades.
- Alle Kantengewichte des Restgraphen entlang p werden um diesen kritischen Wert verringert. Dann geht es weiter mit dem nächsten Iterationsschritt.

Der Edmonds-Karp Algorithmus (Pfad mit wenigen Kanten)

- Der **Edmonds-Karp Algorithmus** wählt als vergrößernden Pfad in der Ford-Fulkerson Methode einen Pfad, der die wenigsten Kanten hat.
- Man fängt mit einem leeren Fluss f an. Der Fluss wird iterativ vergrößert.
- Unter allen Pfaden von s nach t im Restgraphen $\boldsymbol{G}_f$ wird ein Pfad p mit der geringsten Anzahl von Kanten ausgesucht.
- Dies kann z.B. durch Breitensuche im Restgraphen geschehen.
- Dann wird der kritische Wert des Pfades p bestimmt. Dies ist die kleinste Restkapazität (Gewichte im Restgraphen $\boldsymbol{G}_f$) der Kanten des Pfades.
- Alle Kantengewichte des Restgraphen entlang p werden um diesen kritischen Wert verringert. Dann geht es weiter mit dem nächsten Iterationsschritt.
- Der Fluss f braucht dabei nicht explizit gespeichert zu werden. Alle benötigte Information ist in dem Restgraphen.

Pseudocode für den Edmonds-Karp Algorithmus

Listing 1: **Bestimmt den maximalen Fluss eines Flussgraphen G von Quelle s zur Senke t mit dem Edmonds-Karp Algorithmus**

```
1  G_f ←Restgraph von G für leeren Fluss f ≡ 0
2  while es gibt einen Pfad p von s nach t in G_f do
3     wähle Pfad p in G_f mit den wenigsten Kanten
4     cv ←min { rc(e) | e liegt auf Pfad p }
5     // aktualisiere G_f entlang p:
6     for all Knoten v, w mit v→w auf Pfad p in G_f
7        rc(v,w) ←rc(v,w) − cv
8        rc(w,v) ←rc(w,v) + cv
9     end
10 end
```

- Für die Implementierung bietet es sich an, alle Kanten in dem Restgraphen G_f zu speichern und belassen, auch wenn die Restkapazität einer Kante 0 ist.
- Dann dürfen bei der Wahl des Pfades in Zeilen 2 und 3 nur Kanten $v \to w$ mit $rc(v, w) > 0$ berücksichtigt werden.

Edmonds-Karp Algorithmus

Laufzeit des Edmonds-Karp Algorithmus

Der Edmonds Karp Algorithmus bestimmt den maximalen Fluss eines Flussgraphen in einer Laufzeit von $\boldsymbol{O}(E^2V)$.

- Die Korrektheit gilt als Spezialfall der Ford-Fulkerson Methode. Die Laufzeit folgt aus den beiden Lemmata, die auf den folgenden Seiten bewiesen werden.
- **Lemma 1:** Die Längen der kürzesten vergrößernden Pfade ist monoton steigend.
- **Lemma 2:** Spätestens nach E Iterationen steigt die Länge streng an.

Laufzeit des Edmonds-Karp Algorithmus

Der Edmonds Karp Algorithmus bestimmt den maximalen Fluss eines Flussgraphen in einer Laufzeit von $\boldsymbol{O}(E^2V)$.

- Die Korrektheit gilt als Spezialfall der Ford-Fulkerson Methode. Die Laufzeit folgt aus den beiden Lemmata, die auf den folgenden Seiten bewiesen werden.
- **Lemma 1:** Die Längen der kürzesten vergrößernden Pfade ist monoton steigend.
- **Lemma 2:** Spätestens nach E Iterationen steigt die Länge streng an.
- Da die Länge des kürzesten Pfades höchstens $V - 1$ beträgt, kann es nach den Lemmata maximal $E(V - 1)$-viele Iterationen $(= \text{Flussvergrößerungen})$ geben.
- Ein kürzester vergrößernder Pfad wird jeweils im Restgraphen mit Breitensuche in $\boldsymbol{O}(E)$ gefunden (beachte $E \geq V - 1$). Ebenso erfolgt die Aktualisierung des Restgraphen in $\boldsymbol{O}(E)$.
- Die Gesamtlaufzeit ist somit in $\boldsymbol{O}(E^2V)$. □

Lemma 1 für die Laufzeit von Edmonds-Karp

Lemma 1: Die Längen der kürzesten vergrößernden Pfade, die im Edmonds-Karp Algorithmus ausgewählt werden, ist (schwach) monoton steigend.

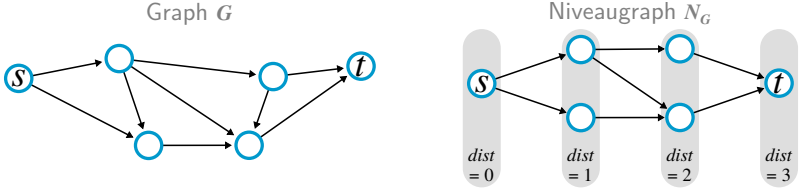
- Um das Lemma anschaulich zu beweisen, führen wir **Niveaugraphen** ein. Der Niveaugraph N_G zu G enthält genau dann einen Pfad $s \rightsquigarrow v$, wenn $s \rightsquigarrow v$ ein kürzester Pfad in G ist.

Lemma 1 für die Laufzeit von Edmonds-Karp

Lemma 1: Die Längen der kürzesten vergrößernden Pfade, die im Edmonds-Karp Algorithmus ausgewählt werden, ist (schwach) monoton steigend.

- Um das Lemma anschaulich zu beweisen, führen wir **Niveaugraphen** ein. Der Niveaugraph N_G zu G enthält genau dann einen Pfad $s \rightsquigarrow v$, wenn $s \rightsquigarrow v$ ein kürzester Pfad in G ist.
- Genauer: Sei *dist(v)* die Anzahl der Kanten des kürzesten Weges von der Quelle s nach v in $G = (V, E)$.
- Der Niveaugraph $N_G = (V, E_N)$ ist der Untergraph von G mit folgenden Kanten:

$$E_N = \{v \to w \mid dist(w) = dist(v) + 1\}$$

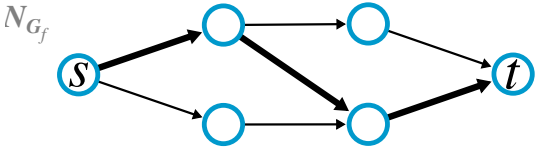


Beweis von Lemma 1 für die Laufzeit von Edmonds-Karp

- Sei f der Fluss vor und f' der Fluss nach einer Flussvergrößerung.
- Wir vergleichen die Niveaugraphen zu den Restgraphen $\boldsymbol{G}_f$ und $\boldsymbol{G}_{f'}$. Der vergrößernde Pfad p muss ein Pfad in $\boldsymbol{N}_{\boldsymbol{G}_f}$ sein.
- Durch die Flussvergrößerung können nur Kanten verändert werden, die in Pfad p benachbarte Knoten verbinden.

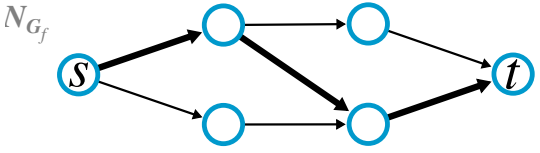
Beweis von Lemma 1 für die Laufzeit von Edmonds-Karp

- Sei f der Fluss vor und f' der Fluss nach einer Flussvergrößerung.
- Wir vergleichen die Niveaugraphen zu den Restgraphen $\boldsymbol{G}_f$ und $\boldsymbol{G}_{f'}$. Der vergrößernde Pfad p muss ein Pfad in $\boldsymbol{N}_{\boldsymbol{G}_f}$ sein.
- Durch die Flussvergrößerung können nur Kanten verändert werden, die in Pfad p benachbarte Knoten verbinden.
  - Kanten in Pfadrichtung können höchstens wegfallen. Dies passiert, wenn eine Kante $v \to w$ voll gefüllt wird: $f'(v, w) = c(v, w)$.
  - Kanten gegen Pfadrichtung $w \to v$ können neu entstehen. Dies passiert, wenn eine Kante $v \to w$ angefüllt wird, die vorher leer war: $f'(v, w) > f(v, w) = 0$.



Beweis von Lemma 1 für die Laufzeit von Edmonds-Karp

- Sei f der Fluss vor und f' der Fluss nach einer Flussvergrößerung.
- Wir vergleichen die Niveaugraphen zu den Restgraphen $\boldsymbol{G}_f$ und $\boldsymbol{G}_{f'}$. Der vergrößernde Pfad p muss ein Pfad in $\boldsymbol{N}_{\boldsymbol{G}_f}$ sein.
- Durch die Flussvergrößerung können nur Kanten verändert werden, die in Pfad p benachbarte Knoten verbinden.
  - Kanten in Pfadrichtung können höchstens wegfallen. Dies passiert, wenn eine Kante $v \to w$ voll gefüllt wird: $f'(v, w) = c(v, w)$.
  - Kanten gegen Pfadrichtung $w \to v$ können neu entstehen. Dies passiert, wenn eine Kante $v \to w$ angefüllt wird, die vorher leer war: $f'(v, w) > f(v, w) = 0$.
- Weder wegfallende Kanten, noch neue Kanten, die im Niveaugraphen rückwärts verlaufen würden, können zu kürzeren Wegen im Restgraphen $\boldsymbol{G}_{f'}$ führen. □



Lemma 2 für die Laufzeit von Edmonds-Karp

Lemma 2: Spätestens nach E Iterationen steigt die Länge streng an (also um einen Wert > 0)

Beweis.

- Der kürzeste vergrößerende Pfad p im Restpgrahen $\boldsymbol{G}_f$ ist im Niveaugraphen $\boldsymbol{N}_{\boldsymbol{G}_f}$ enthalten.
- Bei jeder Flussvergrößerung wird mindestens eine Kante des Pfades aus dem Restgraphen gelöscht.
- Dies ist diejenige Kante, bei der der kritische Wert erreicht wird (Kapazität erschöpft bei Kanten in Pfadrichtung im Flussgraphen, bzw. Fluss auf 0 reduziert bei Kanten gegen Pfadrichtung im Flussgraphen).

Lemma 2 für die Laufzeit von Edmonds-Karp

Lemma 2: Spätestens nach E Iterationen steigt die Länge streng an (also um einen Wert > 0)

Beweis.

- Der kürzeste vergrößerende Pfad p im Restpgrahen $\boldsymbol{G_f}$ ist im Niveaugraphen $\boldsymbol{N_{G_f}}$ enthalten.
- Bei jeder Flussvergrößerung wird mindestens eine Kante des Pfades aus dem Restgraphen gelöscht.
- Dies ist diejenige Kante, bei der der kritische Wert erreicht wird (Kapazität erschöpft bei Kanten in Pfadrichtung im Flussgraphen, bzw. Fluss auf 0 reduziert bei Kanten gegen Pfadrichtung im Flussgraphen).
- Nach spätestens E-vielen Flussvergrößerungen sind also alle Kanten aus dem Niveaugraphen gelöscht.
- Es gibt also keine vergrößernden Pfade dieser Länge mehr. Also muss die Länge um mindestens 1 ansteigen. □

Verbesserungen der Laufzeit von Edmonds-Karp

- Bisher: $\boldsymbol{O}(VE)$ viele Flussvergrößerungen, jeweils $\boldsymbol{O}(E)$, insgesamt $\boldsymbol{O}(E^2V)$.
- Es gibt Beispiele für Flussgraphen, bei denen die Anzahl der notwendigen Flussvergrößerungen tatsächlich in $\boldsymbol{\Theta}(VE)$ liegt, wenn immer ein kürzester vergrößernder Pfad gewählt wird. An dieser Schranke ist also in Edmonds-Karp nichts zu verbessern.
- Es kann aber die benötigte Zeit für Flussvergrößerungen reduziert werden.

Verbesserungen der Laufzeit von Edmonds-Karp

- Bisher: $\boldsymbol{O}(VE)$ viele Flussvergrößerungen, jeweils $\boldsymbol{O}(E)$, insgesamt $\boldsymbol{O}(E^2V)$.
- Es gibt Beispiele für Flussgraphen, bei denen die Anzahl der notwendigen Flussvergrößerungen tatsächlich in $\boldsymbol{\Theta}(VE)$ liegt, wenn immer ein kürzester vergrößernder Pfad gewählt wird. An dieser Schranke ist also in Edmonds-Karp nichts zu verbessern.
- Es kann aber die benötigte Zeit für Flussvergrößerungen reduziert werden.
- Der *blocking-flow* Algorithmus wurde in [Dinic 1970] vorgeschlagen, also *vor* der Veröffentlichung von Edmonds-Karp.
- Dabei werden Pfade in dem Niveaugraphen schrittweise aktualisiert, um jeweils den nächsten vergrößernden Pfad effizienter zu finden.
- Auf diese Weise lässt sich eine Laufzeit in $\boldsymbol{O}(EV^2)$ erreichen.

Verbesserungen der Laufzeit von Edmonds-Karp

- Bisher: $\boldsymbol{O}(VE)$ viele Flussvergrößerungen, jeweils $\boldsymbol{O}(E)$, insgesamt $\boldsymbol{O}(E^2V)$.
- Es gibt Beispiele für Flussgraphen, bei denen die Anzahl der notwendigen Flussvergrößerungen tatsächlich in $\boldsymbol{\Theta}(VE)$ liegt, wenn immer ein kürzester vergrößernder Pfad gewählt wird. An dieser Schranke ist also in Edmonds-Karp nichts zu verbessern.
- Es kann aber die benötigte Zeit für Flussvergrößerungen reduziert werden.
- Der *blocking-flow* Algorithmus wurde in [Dinic 1970] vorgeschlagen, also *vor* der Veröffentlichung von Edmonds-Karp.
- Dabei werden Pfade in dem Niveaugraphen schrittweise aktualisiert, um jeweils den nächsten vergrößernden Pfad effizienter zu finden.
- Auf diese Weise lässt sich eine Laufzeit in $\boldsymbol{O}(EV^2)$ erreichen.
- Mit dynamischen Bäumen [Sleator & Tarjan 1983] kann sogar eine Laufzeit in $\boldsymbol{O}(EV \log V)$ erzielt werden.

Der Kapazitätskontrolle Algorithmus (Pfad mit großem Fluss)

- Zurück zu der allgemeinen Ford-Fulkerson Methode und der Suche nach einer geeigneten Strategie zur Auswahl der vergrößernden Pfade.

Der Kapazitätskontrolle Algorithmus (Pfad mit großem Fluss)

- Zurück zu der allgemeinen Ford-Fulkerson Methode und der Suche nach einer geeigneten Strategie zur Auswahl der vergrößernden Pfade.
- Besonders sinnvoll scheint es, Pfade auszuwählen, die den Fluss maximal vergrößern.
- Eine entsprechende Pfadauswahl wurde auch von Edmonds und Karp vorgeschlagen, ist aber weniger effizient zu implementieren.

Der Kapazitätskontrolle Algorithmus (Pfad mit großem Fluss)

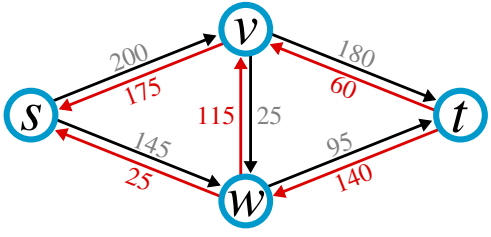
- Zurück zu der allgemeinen Ford-Fulkerson Methode und der Suche nach einer geeigneten Strategie zur Auswahl der vergrößernden Pfade.
- Besonders sinnvoll scheint es, Pfade auszuwählen, die den Fluss maximal vergrößern.
- Eine entsprechende Pfadauswahl wurde auch von Edmonds und Karp vorgeschlagen, ist aber weniger effizient zu implementieren.
- Geben wir uns also mit weniger zufrieden: Der vergrößernde Pfad erhöht den Fluss nicht maximal, aber relativ stark.
- Wir benutzen einen Parameter Δ zur Kapazitätskontrolle: Es werden nur Pfade mit einem Fluss $\geq \Delta$ gewählt.
- Wenn es keine solchen Pfade mehr gibt, wird Δ halbiert, und das Spiel wird fortgesetzt.
- Dieses Verfahren wird ***Capacity Scaling*** genannt.

Der *Capacity Scaling* Algorithmus

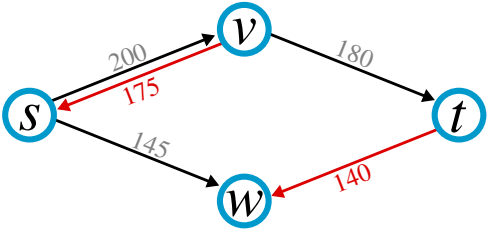
- Zu $\Delta > 0$ definieren wir den Subgraphen $\boldsymbol{G}_f(\Delta)$ des Restgraphens $\boldsymbol{G}_f$ der nur Kanten mit einer Restkapazität von mindestens Δ enthält:

$$\boldsymbol{E}_f(\Delta) = \{v \to w \in \boldsymbol{E} \mid rc(v, w) \geq \Delta\}$$

$\boldsymbol{G}_f$



$\boldsymbol{G}_f(\Delta)$ für Δ=128



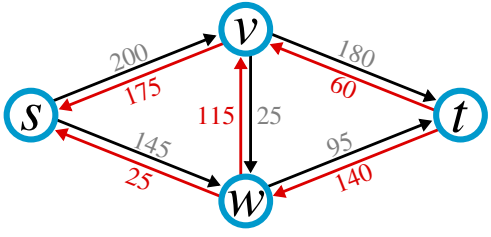
Der *Capacity Scaling* Algorithmus

- Zu $\Delta > 0$ definieren wir den Subgraphen $\boldsymbol{G}_f(\Delta)$ des Restgraphens $\boldsymbol{G}_f$ der nur Kanten mit einer Restkapazität von mindestens Δ enthält:

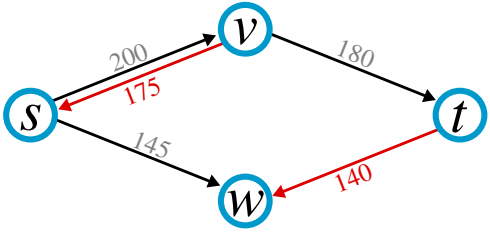
$$\boldsymbol{E}_f(\Delta) = \{v{\rightarrow}w \in \boldsymbol{E} \mid rc(v,w) \geq \Delta\}$$

- Mit jedem Pfad in $\boldsymbol{G}_f(\Delta)$ kann der Fluss f um mindestens Δ vergrößert werden.

$\boldsymbol{G}_f$



$\boldsymbol{G}_f(\Delta)$ für Δ=128



Pseudocode für den *Capacity Scaling* Algorithmus

Listing 2: **Bestimme den maximalen Fluss eines Flussgraphen G von Quelle s zur Senke t mit** *Capacity Scaling*

```
1  C ←maximale Kapazität von Kanten in G
2  Δ ←2^⌊log2(C)⌋ // kleinste 2-er Potenz ≤ C
3  G_f(Δ) ←Δ–Restgraph von G_f für leeren Fluss f ≡ 0
4  while Δ ≥ 1
5    while es gibt einen Pfad p von s nach t in G_f(Δ) do
6      cv ←min { rc(e) | e liegt auf Pfad p in G_f(Δ) }
7      // aktualisiere G_f(Δ) entlang p
8      for all Knoten v, w mit v→w auf Pfad p in G_f(Δ)
9        rc(v,w) ←rc(v,w) − cv
10       rc(w,v) ←rc(w,v) + cv
11     end
12   end
13   Δ ←Δ/2
14   aktualisiere G_f(Δ), indem neue Kanten hinzugefügt werden
15 end
```

- Bei der Aktualisierung von $G_f(\Delta)$ in Zeile 14 werden alle Kanten mit einer Restkapazität im Intervall $[\Delta, 2\Delta[$ zu dem vorigen Graphen $G_f(2\Delta)$ hinzugefügt.
- Es muss natürlich nur ein Restgraph gespeichert werden. Die Abhängigkeit von Δ in dem Code dient nur der inhaltlichen Einordnung.

Korrektheit des *Capacity Scaling* Algorithmus

Für einen Flussgraphen G mit ganzzahligen Kapazitäten bestimmt der *Capacity Scaling* Algorithmus 2 den maximalen Fluss.

Beweis.

- Wenn alle Kapazitäten ganzzahlig sind, dann gilt dies auch für alle Restkapazitäten.

Korrektheit des *Capacity Scaling* Algorithmus

Für einen Flussgraphen $\boldsymbol{G}$ mit ganzzahligen Kapazitäten bestimmt der *Capacity Scaling* Algorithmus 2 den maximalen Fluss.

Beweis.

- Wenn alle Kapazitäten ganzzahlig sind, dann gilt dies auch für alle Restkapazitäten.
- Für $\Delta = 1$ gilt also $\boldsymbol{G}_f(\Delta) = \boldsymbol{G}_f$.
- Daher folgt aus dem Satz über vergrößernde Pfade, Seite 18, dass der Fluss maximal ist, wenn es keine Pfade mehr von s nach t in $\boldsymbol{G}_f$ gibt. □

Laufzeit des *Capacity Scaling* Algorithmus

Der *Capacity Scaling* Algorithmus 2 bestimmt den maximalen Fluss in einer Laufzeit in $\boldsymbol{O}(E^2 \log \boldsymbol{C})$.

Beweis. (Beweise der Teilaussagen **1** – **3** folgen.)

1 Es gibt höchstens $1 + \lceil \log_2 C \rceil$ Skalierungsphasen
(= Durchläufe der *while*-Schleife, Zeile 4–15).

Laufzeit des *Capacity Scaling* Algorithmus

Laufzeit des *Capacity Scaling* Algorithmus

Der *Capacity Scaling* Algorithmus 2 bestimmt den maximalen Fluss in einer Laufzeit in $\boldsymbol{O}(E^2 \log \boldsymbol{C})$.

Beweis. (Beweise der Teilaussagen **1** – **3** folgen.)

1 Es gibt höchstens $1 + \lceil \log_2 C \rceil$ Skalierungsphasen
(= Durchläufe der *while*-Schleife, Zeile 4–15).

2 In jeder Skalierungsphase gibt es höchstens $2E$-viele Flussvergrößerungen
(= Durchläufe der *while*-Schleife, Zeilen 5–12).

Laufzeit des *Capacity Scaling* Algorithmus

Laufzeit des *Capacity Scaling* Algorithmus

Der *Capacity Scaling* Algorithmus 2 bestimmt den maximalen Fluss in einer Laufzeit in $\boldsymbol{O}(E^2 \log C)$.

Beweis. (Beweise der Teilaussagen **1** – **3** folgen.)

1 Es gibt höchstens $1 + \lceil \log_2 C \rceil$ Skalierungsphasen
(= Durchläufe der *while*-Schleife, Zeile 4–15).

2 In jeder Skalierungsphase gibt es höchstens $2E$-viele Flussvergrößerungen
(= Durchläufe der *while*-Schleife, Zeilen 5–12).

3 Jede Flussvergrößerung (Zeile 5–11) benötigt eine Laufzeit in $\boldsymbol{O}(E)$
Suchen eines Pfades p im Restgraphen und Erhöhung von $\boldsymbol{G}_f(\Delta)$ entlang p.

Laufzeit des *Capacity Scaling* Algorithmus

Der *Capacity Scaling* Algorithmus 2 bestimmt den maximalen Fluss in einer Laufzeit in $\boldsymbol{O}(E^2 \log C)$.

Beweis. (Beweise der Teilaussagen **1** – **3** folgen.)

1 Es gibt höchstens $1 + \lceil \log_2 C \rceil$ Skalierungsphasen
(= Durchläufe der *while*-Schleife, Zeile 4–15).

2 In jeder Skalierungsphase gibt es höchstens $2E$-viele Flussvergrößerungen
(= Durchläufe der *while*-Schleife, Zeilen 5–12).

3 Jede Flussvergrößerung (Zeile 5–11) benötigt eine Laufzeit in $\boldsymbol{O}(E)$
Suchen eines Pfades p im Restgraphen und Erhöhung von $\boldsymbol{G}_f(\Delta)$ entlang p.

- Insgesamt wird maximal $2E(1 + \lceil \log_2 C \rceil)$ mal eine Flussvergrößerung mit Laufzeit in $\boldsymbol{O}(E)$ ausgeführt. Die Aktualisierung in Zeile 14 kann ebenfalls in $\boldsymbol{O}(E)$ erfolgen.
- Die Gesamtlaufzeit ist in $\boldsymbol{O}(E^2 \log C)$. □

Beweise der Teilaussagen

Wir beweisen zunächste die folgende, zentrale Eigenschaft:

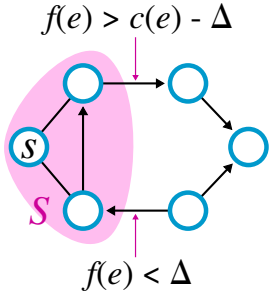
- **Lemma:** Am Ende einer Skalierungsiteration (Zeile 12) gilt $|f^*| \leq |f| + E\Delta$.
- Wir betrachten den Schnitt S der durch alle von s in $G_f(\Delta)$ erreichbaren Knoten definiert ist und zeigen $c(S, V - S) \leq |f| + E\Delta$ (siehe Schnitttheorem, Seite 14).

Beweise der Teilaussagen

Wir beweisen zunächste die folgende, zentrale Eigenschaft:

- **Lemma:** Am Ende einer Skalierungsiteration (Zeile 12) gilt $|f^*| \leq |f| + E\Delta$.
- Wir betrachten den Schnitt $\boldsymbol{S}$ der durch alle von s in $\boldsymbol{G}_f(\Delta)$ erreichbaren Knoten definiert ist und zeigen $c(\boldsymbol{S}, \boldsymbol{V} - \boldsymbol{S}) \leq |f| + E\Delta$ (siehe Schnitttheorem, Seite 14).
- Nach Definition ist $s \in \boldsymbol{S}$ und da am Ende einer Skalierungsiteration die *while*-Bedingung in Zeile 5 nicht erfüllt ist, ist t nicht in $\boldsymbol{G}_f(\Delta)$ erreichbar, also $t \notin \boldsymbol{S}$.

$$
\begin{aligned}
|f| &= \sum_{e \in \boldsymbol{E}^{\rightarrow}_{\boldsymbol{S}}} f(e) - \sum_{e \in \boldsymbol{E}^{\leftarrow}_{\boldsymbol{S}}} f(e) && \text{Schnitttheorem, S. 14} \\
&\geq \sum_{e \in \boldsymbol{E}^{\rightarrow}_{\boldsymbol{S}}} (c(e) - \Delta) - \sum_{e \in \boldsymbol{E}^{\leftarrow}_{\boldsymbol{S}}} \Delta && \text{Definition von } \boldsymbol{G}_f(\Delta): \\
&= \sum_{e \in \boldsymbol{E}^{\rightarrow}_{\boldsymbol{S}}} c(e) - \sum_{e \in \boldsymbol{E}^{\rightarrow}_{\boldsymbol{S}}} \Delta - \sum_{e \in \boldsymbol{E}^{\leftarrow}_{\boldsymbol{S}}} \Delta && \text{Umsortieren der Summen} \\
&\geq c(\boldsymbol{S}, \boldsymbol{V} - \boldsymbol{S}) - E\Delta && \text{Defintion von } c(\boldsymbol{S}, \boldsymbol{T})
\end{aligned}
$$



Beweise der Teilaussagen

1 Es gibt höchstens $1 + \lceil \log_2 C \rceil$ Skalierungsphasen
(= Durchläufe der *while*-Schleife, Zeile 4–15).

- Dies ist der Fall, da Δ anfänglich in $]C/2,\ C]$ gewählt und dann immer halbiert wird.

Beweise der Teilaussagen

1 Es gibt höchstens $1 + \lceil \log_2 C \rceil$ Skalierungsphasen
(= Durchläufe der *while*-Schleife, Zeile 4–15).

- Dies ist der Fall, da Δ anfänglich in $]C/2,\ C]$ gewählt und dann immer halbiert wird.

2 In jeder Skalierungsphase gibt es höchstens $2E$-viele Flussvergrößerungen
(= Durchläufe der *while*-Schleife, Zeilen 5–12).

- Am Anfang einer Skalierungsphase gilt nach dem Lemma $|f^*| \leq |f| + E2\Delta$ als Resultat der vorgegangenen Phase für 2Δ. Da jede Flussvergrößerung den Flusswert um Δ erhöht, kann die Anzahl $2E$ nicht überschreiten. Sonst würde der Flusswert $|f|$ über den Maximalwert $|f^*|$ steigen.

Beweise der Teilaussagen

1 Es gibt höchstens $1 + \lceil \log_2 C \rceil$ Skalierungsphasen
(= Durchläufe der *while*-Schleife, Zeile 4–15).

- Dies ist der Fall, da Δ anfänglich in $]C/2,\ C]$ gewählt und dann immer halbiert wird.

2 In jeder Skalierungsphase gibt es höchstens $2E$-viele Flussvergrößerungen
(= Durchläufe der *while*-Schleife, Zeilen 5–12).

- Am Anfang einer Skalierungsphase gilt nach dem Lemma $|f^*| \leq |f| + E2\Delta$ als Resultat der vorgegangenen Phase für 2Δ. Da jede Flussvergrößerung den Flusswert um Δ erhöht, kann die Anzahl $2E$ nicht überschreiten. Sonst würde der Flusswert $|f|$ über den Maximalwert $|f^*|$ steigen.

3 Jede Flussvergrößerung (Zeile 5–11) benötigt eine Laufzeit in $\boldsymbol{O}(E)$

- Suchen eines Pfades p im Restgraphen benötigt $\boldsymbol{O}(E)$ und Erhöhung von $\boldsymbol{G}_f(\Delta)$ entlang p ebenfalls.

Laufzeiten von *maxflow* Algorithmen

Die folgende Tabelle zeigt eine Übersicht über die Laufzeiten von *maxflow* Algorithmen für Flussgraphen mit ganzzahligen Kapazitäten.

Algorithmen zum Finden des Maximalen Flusses		
Algorithmus	***worst-case***	**alternativ**
Ford-Fulkerson	$\boldsymbol{O}(E\|f^*\|)$	$\boldsymbol{O}(EVC)$
Edmonds-Karp	$\boldsymbol{O}(E^2V)$	
blocking-flow	$\boldsymbol{O}(EV^2)$	
blocking-flow mit dynamischen Bäumen	$\boldsymbol{O}(EV \log V)$	
Capacity scaling	$\boldsymbol{O}(E^2 \log C)$	

C ist die maximale Kapazität, $|f^*|$ der maximale Fluss.

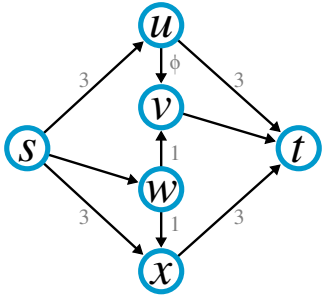
Anhang

Inhalt des Anhangs:

- Beispiel Flussgraph mit irrationalen Kapazitäten, für den die Ford-Fulkerson Methode bei ungünstiger Wahl der vergrößernden Pfade nicht terminiert: S. 43

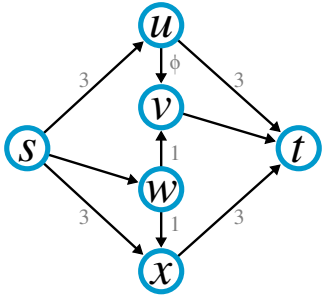
Kleiner Graph ohne Terminierung

- Sei $\phi = (\sqrt{5} - 1)/2$, das Verhältnis des goldenen Schnittes.
- Es gilt $\phi^2 = \left(\frac{\sqrt{5}-1}{2}\right)^2 = \frac{5-2\sqrt{5}+1}{4} = \frac{3-\sqrt{5}}{2} = 1 - \phi$

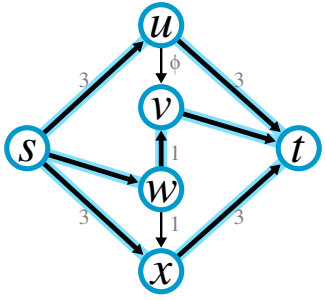


Kleiner Graph ohne Terminierung

- Sei $\phi = (\sqrt{5} - 1)/2$, das Verhältnis des goldenen Schnittes.
- Es gilt $\phi^2 = \left(\frac{\sqrt{5}-1}{2}\right)^2 = \frac{5-2\sqrt{5}+1}{4} = \frac{3-\sqrt{5}}{2} = 1 - \phi$
- und $\phi - \phi^2 = \phi(1-\phi) = \phi \cdot \phi^2 = \phi^3$.

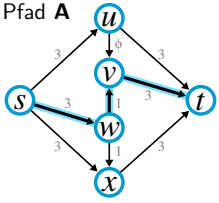


- Sei $\phi = (\sqrt{5} - 1)/2$, das Verhältnis des goldenen Schnittes.
- Es gilt $\phi^2 = \left(\frac{\sqrt{5}-1}{2}\right)^2 = \frac{5-2\sqrt{5}+1}{4} = \frac{3-\sqrt{5}}{2} = 1 - \phi$
- und $\phi - \phi^2 = \phi(1-\phi) = \phi \cdot \phi^2 = \phi^3$.
- Der abgebildete Graph hat einen maximalen Fluss von mindestens 7:
  Die Pfade $s - u - t$ und $s - x - t$ bringen jeweils 3 und $s - w - v - t$ bringt 1.

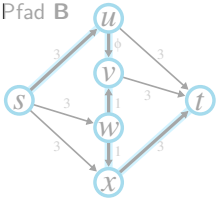


Kleiner Graph ohne Terminierung

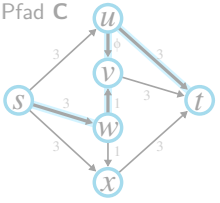
Pfad **A**



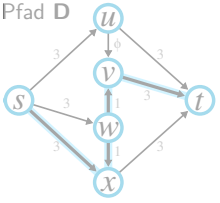
Pfad **B**



Pfad **C**



Pfad **D**

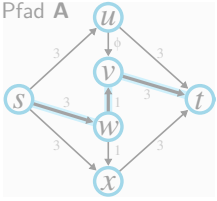


Folgende Wahl vergrößernder Pfade führt zu einer endlosen Sequenz:

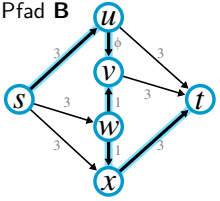
Pfad	**Fluss**	**Restkapazitäten** $u \to v$	$v \to w$	$w \to x$
A	1	ϕ	0	1

Kleiner Graph ohne Terminierung

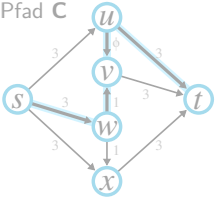
Pfad **A**



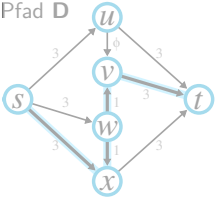
Pfad **B**



Pfad **C**



Pfad **D**

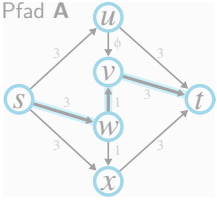


Folgende Wahl vergrößernder Pfade führt zu einer endlosen Sequenz:

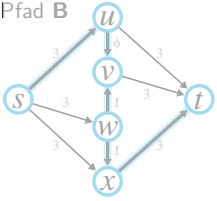
Pfad	**Fluss**	**Restkapazitäten** $u\to v$	$v\to w$	$w\to x$	
A	1	ϕ	0	1	
B	ϕ	0	ϕ	ϕ^2	da $1-\phi=\phi^2$

Kleiner Graph ohne Terminierung

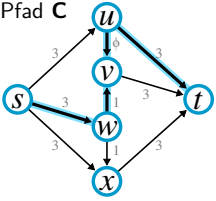
Pfad **A**



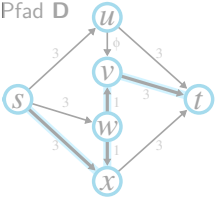
Pfad **B**



Pfad **C**



Pfad **D**



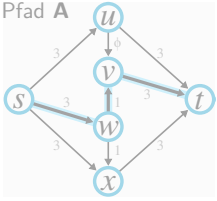
Folgende Wahl vergrößernder Pfade führt zu einer endlosen Sequenz:

Pfad	**Fluss**	**Restkapazitäten** $u \to v$	$v \to w$	$w \to x$	
A	1	ϕ	0	1	
B	ϕ	0	ϕ	ϕ^2	da $1 - \phi = \phi^2$
C	ϕ	ϕ	0	ϕ^2	

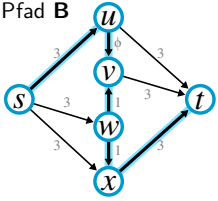
[Zwick 1995; Erickson, http://jeffe.cs.illinois.edu/teaching/algorithms]

Kleiner Graph ohne Terminierung

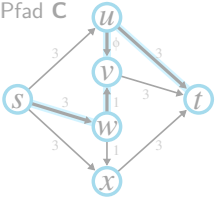
Pfad **A**



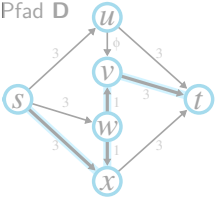
Pfad **B**



Pfad **C**



Pfad **D**



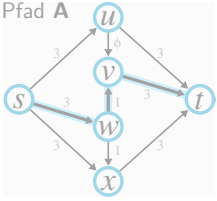
Folgende Wahl vergrößernder Pfade führt zu einer endlosen Sequenz:

Pfad	**Fluss**	**Restkapazitäten** $u\to v$	$v\to w$	$w\to x$	
A	1	ϕ	0	1	
B	ϕ	0	ϕ	ϕ^2	da $1-\phi=\phi^2$
C	ϕ	ϕ	0	ϕ^2	
B	ϕ^2	ϕ^3	ϕ^2	0	da $\phi-\phi^2=\phi^3$

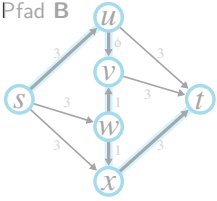
[Zwick 1995; Erickson, http://jeffe.cs.illinois.edu/teaching/algorithms]

Kleiner Graph ohne Terminierung

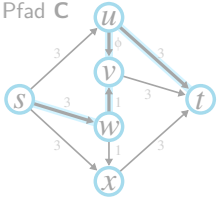
Pfad **A**



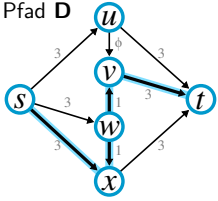
Pfad **B**



Pfad **C**



Pfad **D**

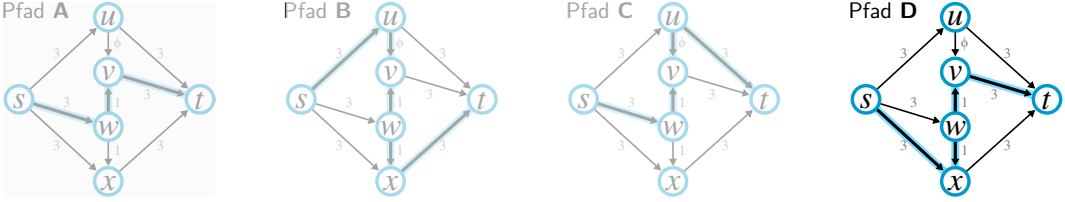


Folgende Wahl vergrößernder Pfade führt zu einer endlosen Sequenz:

Pfad	**Fluss**	**Restkapazitäten** $u\to v$	$v\to w$	$w\to x$	
A	1	ϕ	0	1	
B	ϕ	0	ϕ	ϕ^2	da $1-\phi=\phi^2$
C	ϕ	ϕ	0	ϕ^2	
B	ϕ^2	ϕ^3	ϕ^2	0	da $\phi-\phi^2=\phi^3$
D	ϕ^2	ϕ^3	0	ϕ^2	

[Zwick 1995; Erickson, http://jeffe.cs.illinois.edu/teaching/algorithms]

Kleiner Graph ohne Terminierung

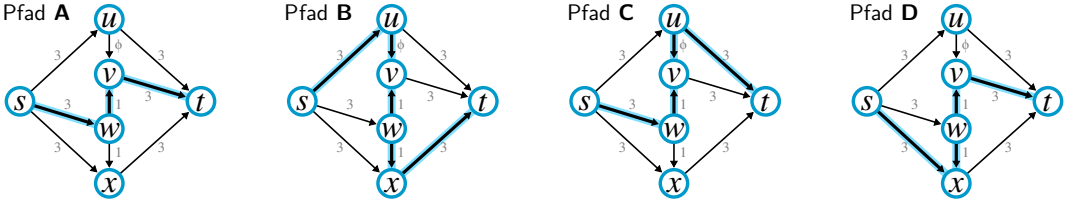


Folgende Wahl vergrößernder Pfade führt zu einer endlosen Sequenz:

Pfad	Fluss	Restkapazitäten $u{\to}v$	$v{\to}w$	$w{\to}x$	
A	1	ϕ	0	1	$\to\ \phi^{k+1}\quad 0\quad \phi^{k}$ (für $k=0$)
B	ϕ	0	ϕ	ϕ^2	da $1-\phi=\phi^2$
C	ϕ	ϕ	0	ϕ^2	
B	ϕ^2	ϕ^3	ϕ^2	0	da $\phi-\phi^2=\phi^3$
D	ϕ^2	ϕ^3	0	ϕ^2	$\to\ \phi^{k+3}\quad 0\quad \phi^{k+2}$ (für $k=0$)

[Zwick 1995; Erickson, http://jeffe.cs.illinois.edu/teaching/algorithms]

Kleiner Graph ohne Terminierung



Folgende Wahl vergrößernder Pfade führt zu einer endlosen Sequenz:

Pfad	Fluss	Restkapazitäten $u\to v$	$v\to w$	$w\to x$	
A	1	ϕ	0	1	$\to\ \phi^{k+1}\quad 0\quad \phi^k$ (für $k=0$)
B	ϕ	0	ϕ	ϕ^2	da $1-\phi=\phi^2$
C	ϕ	ϕ	0	ϕ^2	
B	ϕ^2	ϕ^3	ϕ^2	0	da $\phi-\phi^2=\phi^3$
D	ϕ^2	ϕ^3	0	ϕ^2	$\to\ \phi^{k+3}\quad 0\quad \phi^{k+2}$ (für $k=0$)
A + $K\cdot$ BCBD	$1+\sum_{k=1}^{2K} 2\phi^k$	ϕ^{2K+1}	0	ϕ^{2K}	

[Zwick 1995; Erickson, http://jeffe.cs.illinois.edu/teaching/algorithms]

Kleiner Graph ohne Terminierung

- Die Pfad Sequenz *A und dann immer wiederholend B, C, B, D* konvergiert nicht zum maximalen Fluss:

$$1 + 2\sum_{k=1}^{\infty} \phi^k = 1 + \frac{2}{1-\phi} = 4 + \sqrt{5} < 7$$

Kleiner Graph ohne Terminierung

- Die Pfad Sequenz *A und dann immer wiederholend B, C, B, D* konvergiert nicht zum maximalen Fluss:

$$1 + 2\sum_{k=1}^{\infty} \phi^k = 1 + \frac{2}{1-\phi} = 4 + \sqrt{5} < 7$$

- Zum vollständigen Beweis fehlen noch folgende (einfachen) Punkte:
- Zeige die Eigenschaften der Sequenz (siehe vorige Seite) durch Induktion nach k.
- Zeige insbesondere $1 - \phi^k = \phi^{k+1}$ und $\phi^k - \phi^{k+1} = \phi^{k+2}$.
- Prüfe bei der Induktion, dass die Kanten mit Kapazität 3 nicht über ihre Kapazitätsgrenzen gefüllt werden.

Literatur I

Generell:

- Ottmann T & Widmayer P. *Algorithmen und Datenstrukturen.* Springer Verlag, 5. Auflage; 2011. ISBN: 978-3827428042
- Kleinberg J, Tardos E. *Algorithm Design.* Pearson Education Limited; Auflage: Pearson New International Edition (30. Juli 2013). ISBN: 978-1292023946
- Cormen TH, Leiserson CE, Rivest R, Stein C. *Algorithmen - Eine Einführung.* De Gruyter Oldenbourg, 4. Auflage; 2013. ISBN: 978-3486748611

Anderes Vorlesungsmaterial:

- Wayne K. Vorlesung *Theory of Algorithms* (COS 423), Princeton University 2013. `https://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures.php`
- Erickson J, *Algorithms lecture notes*, `http://algorithms.wtf`.

Literatur II

Originalveröffentlichungen:

- Zwick U. *The smallest networks on which the ford-fulkerson maximum flow procedure may fail to terminate*. Theoretical computer science. 1995 Aug 21;148(1):165-70.
- Dinic EA. *Algorithm for solution of a problem of maximum flow in a network with power estimation*, Soviet Math. Dokl. 11 (5), 1277-1280, 1970.
- Sleator DD, Tarjan RE. *A data structure for dynamic trees*. Journal of computer and system sciences. 1983 Jun 1;26(3):362-91.
- Orlin JB. *Max flows in O(nm) time*. In: Symp. on Theory of Computing 2012 (pp. 765-774).

Danksagung I

Bei der Darstellung habe ich viele Ideen von den großartigen Folien von Kevin Wayne zu seiner Vorlesung *Theory of Algorithms* (COS 423, Princeton University 2013) aufgenommen. (Seine Vorlesung orientiert sich seinerseits an den Büchern von Kleinberg & Tardos und von Kozen.)

Index