Algorithmische Graphentheorie

Kapitel 10: Flüsse

Babeş-Bolyai Universität, Fachbereich Informatik,
Klausenburg

Flüsse

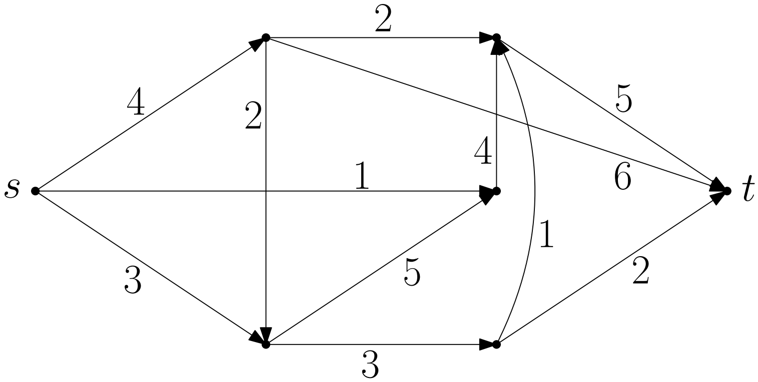
- In diesem Kapitel werden Bewertungen von Kanten als *maximale Kapazitäten* interpretiert, die über solch eine Kante pro Zeiteinheit transportiert werden können.
- Wir können uns einen Graphen als Versorgungsnetzwerk vorstellen, z.B. als Datennetz.
- Fragen: Welchen Durchsatz können wir erreichen? Wie viele Einheiten können wir von einem Knoten zu einem anderen Knoten pro Zeiteinheit transportieren? Welche Kanten bilden einen Engpass?
- Auch viele kombinatorische Probleme auf Graphen, wie Matchings in bipartiten Graphen, lassen sich leicht in ein geeignetes Flussproblem überführen und dort schnell lösen, wie wir sehen werden.

FLUSSNETZWERK

Ein *Flussnetzwerk* N ist ein Tupel $N = (G, c, s, t)$ bestehend aus:

1. $G = (V, A)$, einem gerichteten Graphen, wobei A die Menge der gerichteten Kanten ist – diese nennen wir häufig auch einfach Kanten, wenn aus dem Kontext klar ist, dass diese gerichtet sind; für eine gerichtete Kante von v nach w schreiben wir meistens (v, w) und manchmal einfach vw ($\neq wv$!),
2. einer *Kapazitätsfunktion* $c\colon A \to \mathbb{R}_{\geq 0}$ auf den gerichteten Kanten mit nicht negativen Werten und
3. $s, t \in V$, zwei ausgezeichneten Knoten, der *Quelle* (*source*) s und der *Senke* (*sink*) t mit $s \neq t$.

BEISPIEL



Fluss

Es sei $N = (G, c, s, t)$ ein Flussnetzwerk. Für einen Knoten $v \in V$ sei $A_{in}(v) := \{(u, v) \in A\}$ und $A_{out}(v) := \{(v, u) \in A\}$. Eine Abbildung $f : A \to \mathbb{R}$ heißt (s, t)-*Fluss* (*flow*) auf N, wenn die folgenden Bedingungen erfüllt sind:

1. *Kapazitätsbedingungen* (*capacity constraints*): $0 \leq f(e) \leq c(e)$ für alle $e \in A$, d.h. die Kapazität wird für keine Kante überschritten (in manchen Quellen werden Flüsse ohne diese Eigenschaft definiert, und jene Flüsse, die sie erfüllen, heißen *zulässig*) und
2. *Flusserhaltungsbedingungen* (*flow conservation constraints*): Es gilt das *1. kirchhoffsche Gesetz*:

$$\sum_{e \in A_{in}(v)} f(e) = \sum_{e \in A_{out}(v)} f(e)$$

für alle $v \in V \setminus \{s, t\}$, d.h. aus jedem Knoten fließt genau so viel heraus wie hinein, mit der möglichen Ausnahme der Quelle s und der Senke t.

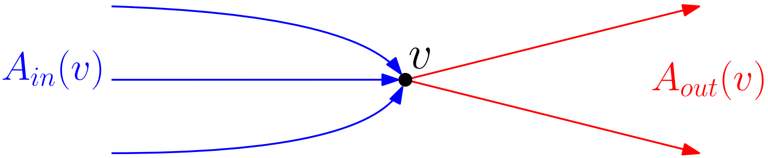
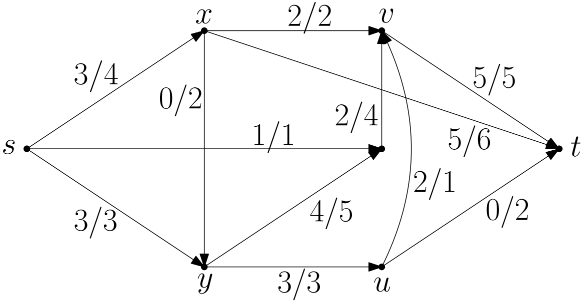


Illustration der Kantenmengen $A_{in}(v)$ und $A_{out}(v)$.

Wenn unmittelbar deutlich ist, wie s und t festgelegt sind, so schreiben wir statt (s,t)-Fluss auch einfach *Fluss*.

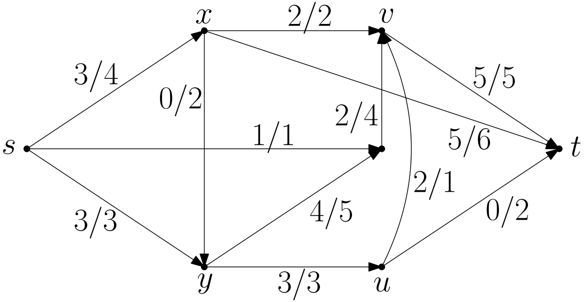
Beispiel

Frage. In folgendem Flussnetzwerk ist ein angeblicher (s,t)-Fluss f angegeben: jede Kante ist mit einem Paar Zahlen beschriftet: zunächst der Flusswert, gefolgt von der Kapazitätsgrenze. Ist f tatsächlich ein (s,t)-Fluss?



Beispiel

Frage. In folgendem Flussnetzwerk ist ein angeblicher (s,t)-Fluss f angegeben: jede Kante ist mit einem Paar Zahlen beschriftet: zunächst der Flusswert, gefolgt von der Kapazitätsgrenze. Ist f tatsächlich ein (s,t)-Fluss?



Nein: Durch Kante uv fließt mehr, als es die Kapazitätsgrenze erlaubt. Zudem erfüllt v nicht das 1. kirchhoffsche Gesetz.

Überschuss eines Knotens unter f

Für eine Funktion $f : A \to \mathbb{R}$ bezeichnen wir mit

$$\text{excess}_f(v) := \sum_{e \in A_{in}(v)} f(e) - \sum_{e \in A_{out}(v)} f(e)$$

den *Überschuss* von v unter f.

Frage. Was wissen wir über $\text{excess}_f(v)$ für jeden Knoten $v \in V \setminus \{s, t\}$, wenn f ein Fluss ist?

ÜBERSCHUSS EINES KNOTENS UNTER f

Für eine Funktion $f : A \to \mathbb{R}$ bezeichnen wir mit

$$\text{excess}_f(v) := \sum_{e \in A_{in}(v)} f(e) - \sum_{e \in A_{out}(v)} f(e)$$

den *Überschuss* von v unter f.

Frage. Was wissen wir über $\text{excess}_f(v)$ für jeden Knoten $v \in V \setminus \{s, t\}$, wenn f ein Fluss ist?

Durch das 1. kirchhoffsche Gesetz ist garantiert, dass $\text{excess}_f(v) = 0$ für alle $v \in V \setminus \{s, t\}$: es muss soviel in den Knoten v hineinfließen wie hinausfließt.

Wert von f

Für einen Fluss f eines Flussnetzwerks $N = (G, c, s, t)$ definieren wir

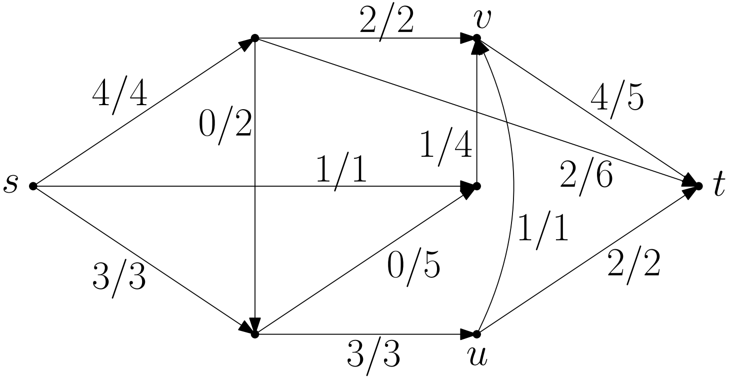
$$\Phi(f) := \text{excess}_f(t).$$

Der Wert $\Phi(f)$ heißt *Wert* des Flusses f auf N.

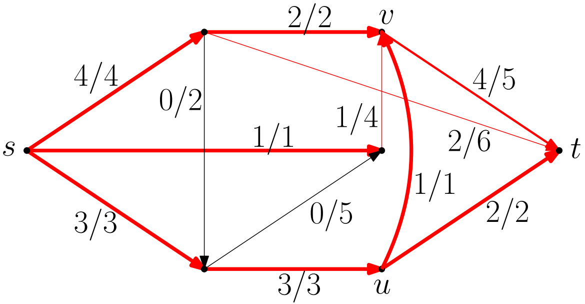
Ein Fluss f mit $\Phi(f) \geq \Phi(f')$ für alle Flüsse f' auf N heißt *maximaler* (s, t)*-Fluss* oder kurz *Maximalfluss* auf N. Das *Maximalflussproblem* besteht darin, zu einem gegebenen Flussnetzwerk einen Maximalfluss zu bestimmen.

Beispiel

Frage. In folgendem Flussnetzwerk ist ein angeblicher (s,t)-Fluss f angegeben. Ist f tatsächlich ein (s,t)-Fluss? Wenn ja, was ist $\Phi(f)$?

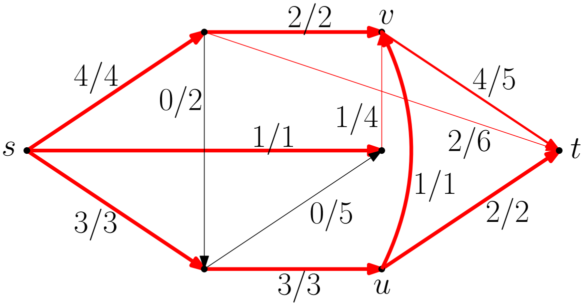


f erfüllt in der Tat alle Kapazitätsgrenzen und das 1. kirchhoffsche Gesetz.



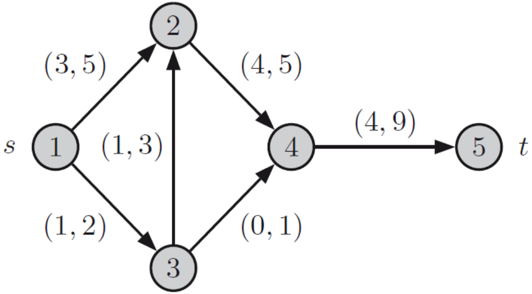
Der Fluss f ist in rot angegeben: fett markierte Kanten sind an ihrer Kapazitätsgrenze, dünner markierte Kanten nicht. Es ist leicht zu berechnen, dass der Flusswert $\Phi(f) = 8$ ist. Ist dieser maximal?

f erfüllt in der Tat alle Kapazitätsgrenzen und das 1. kirchhoffsche Gesetz.



Der Fluss f ist in rot angegeben: fett markierte Kanten sind an ihrer Kapazitätsgrenze, dünner markierte Kanten nicht. Es ist leicht zu berechnen, dass der Flusswert $\Phi(f) = 8$ ist. Ist dieser maximal? Ja, denn alle Kanten, die von s ausgehen, sind **voll ausgelastet**, es gibt also keine Möglichkeit, einen Fluss zu finden, der mehr transportiert. Wir werden dieses Argument auf den folgenden Folien formalisieren.

Für den Rest dieses Kapitels folgen wir grundsätzlich dem Buch von S. O. Krumke und H. Noltemeier, *Graphentheoretische Konzepte und Algorithmen* (3. Auflage), Kapitel 9 (Vieweg+Teubner Verlag, Springer Fachmedien Wiesbaden, 2012).



Jede Kante ist beschriftet mit $(f((u,v)), c((u,v)))$. Der Fluss f in dieser Abbildung hat Wert 4. **Frage.** Ist dieser maximal?

Ein *Schnitt* (*cut*) (A, B) in einem gerichteten oder ungerichteten Graphen G ist eine Partition von V (d.h. $A \cup B = V$ und $A \cap B = \emptyset$) mit $A \neq \emptyset$ und $B \neq \emptyset$.

Es besteht ein enger Zusammenhang zwischen Flüssen und Schnitten, den wir im Folgenden näher untersuchen werden. Dazu definieren wir zunächst einige Begriffe.

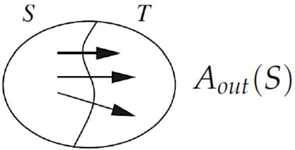
Ist (S, T) ein Schnitt mit $s \in S$ und $t \in T$, so nennen wir (S, T) einen (s, t)-*Schnitt*. Wie bei Flüssen lassen wir die explizite Referenz auf s und t weg, sofern keine Missverständnisse auftreten können.

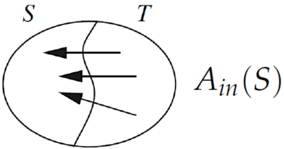
Es sei (S, T) ein Schnitt in einem gerichteten Graphen $G = (V, A)$. Dann definieren wir den *Vorwärtsteil* des Schnittes durch

$$A_{out}(S) := \{(v, w) \in A : v \in S \text{ und } w \in T\}$$

und den *Rückwärtsteil* des Schnittes (S, T) durch

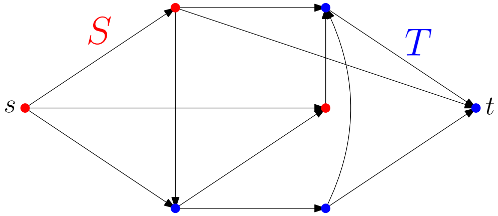
$$A_{in}(S) := \{(v, w) \in A : v \in T \text{ und } w \in S\}.$$





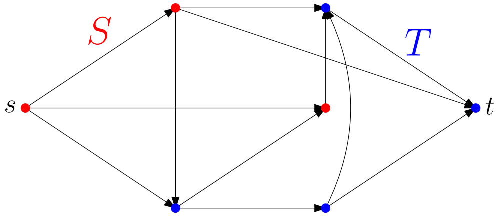
BEISPIEL

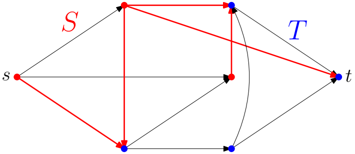
Aufgabe. Bestimmmen Sie $A_{out}(S)$ und $A_{in}(S)$. (S in rot, T in blau.)



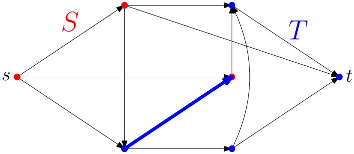
BEISPIEL

Aufgabe. Bestimmmen Sie $A_{out}(S)$ und $A_{in}(S)$. (S in rot, T in blau.)





$A_{out}(S)$ in rot.



$A_{in}(S)$ in blau.

Für $S \subseteq V$ definieren wir den *Überschuss* in S unter einer Funktion $f : A \to \mathbb{R}$ durch

$$\text{excess}_f(S) := f(A_{in}(S)) - f(A_{out}(S)),$$

wobei $f(A_{in}(S)) := \sum_{e \in A_{in}(S)} f(e)$ und $f(A_{out}(S)) := \sum_{e \in A_{out}(S)} f(e)$.

Das folgende Lemma zeigt, dass der Überschuss in S gleich der Summe der Überschüsse der Knoten in S ist.

Lemma. *Es sei $G = (V, A)$ ein gerichteter Graph, $f : A \to \mathbb{R}$ eine beliebige Funktion und $S \subseteq V$. Dann gilt*

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

Beweis

Es gilt nach Definition des Überschusses in einem Knoten:

$$\sum_{v\in S} \text{excess}_f(v) = \sum_{v\in S}\left(\sum_{e\in A_{in}(v)} f(e) - \sum_{e\in A_{out}(v)} f(e)\right)$$

Wenn für eine gerichtete Kante (x, y) in G sowohl x als auch y in S liegen, dann tritt der Term $f(e)$ in obiger Summe einmal positiv (für y) und einmal negativ (für x) auf, denn

$$(x, y) \in A_{out}(x) \cap A_{in}(y).$$

Die obige Summe reduziert sich also auf

$$\sum_{v\in S} \text{excess}_f(v) = \sum_{e\in A_{in}(S)} f(e) - \sum_{e\in A_{out}(S)} f(e) = \underbrace{f(A_{in}(S)) - f(A_{out}(S))}_{=:\ \text{excess}_f(S)}.$$

Lemma. *Ist f ein (s,t)-Fluss und (S,T) ein (s,t)-Schnitt in einem Graphen, so gilt*

$$\Phi(f) = f(A_{out}(S)) - f(A_{in}(S)).$$

Insbesondere folgt $\mathrm{excess}_f(t) = -\mathrm{excess}_f(s)$.

Dieses Lemma hat interessante Konsequenzen. Wir besprechen diese nachdem wir das Lemma beweisen.

BEWEIS (1/2)

Es gilt

$$\Phi(f) := \text{excess}_f(t) \overset{(1)}{=} \sum_{v \in T} \text{excess}_f(v) \overset{(2)}{=} \text{excess}_f(T)$$

$$:= f(A_{in}(T)) - f(A_{out}(T)) \overset{(3)}{=} f(A_{out}(S)) - f(A_{in}(S)),$$

wie behauptet.

(1) $\text{excess}_f(v) = 0$ für alle $v \in V \setminus \{s, t\}$ und $s \notin T$.
(2) Lemma Folie 16.
(3) $A_{in}(T) = A_{out}(S)$ und $A_{out}(T) = A_{in}(S)$.

Beweis (2/2)

Der zweite Teil des Lemmas ergibt sich aus dem ersten wie folgt. Es gilt per Definition (Folie 9)

$$\Phi(f) := \text{excess}_f(t).$$

Setzt man in den ersten Teil des Lemmas den speziellen Schnitt $(\{s\}, V \setminus \{s\})$ ein, so erhält man

$$\Phi(f) = f(A_{out}(s)) - f(A_{in}(s)).$$

Wieder gilt per Definition (Folie 8)

$$f(A_{in}(s)) - f(A_{out}(s)) = \sum_{e \in A_{in}(s)} f(e) - \sum_{e \in A_{out}(s)} f(e) =: \text{excess}_f(s).$$

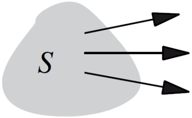
Insgesamt also

$$\text{excess}_f(t) =: \Phi(f) = f(A_{out}(s)) - f(A_{in}(s)) = -\text{excess}_f(s).$$

Ist (S, T) ein (s, t)-Schnitt im Graphen G mit Kapazitäten c (stets nicht negativ), so nennen wir

$$c(A_{out}(S)) = \sum_{e \in A_{out}(S)} c(e)$$

die *Kapazität* des Schnittes (S, T).



Kapazität eines Schnittes (S, T)

Wir nennen (S, T) einen *minimalen* (s, t)*-Schnitt*, falls er unter allen (s, t)-Schnitten minimale Kapazität besitzt.

- Intuitiv ist die Kapazität eines Schnittes (S, T) eine obere Schranke für den Flusswert eines Flusses f.
- Dass diese Intuition auch richtig ist, ergibt sich aus dem Lemma auf Folie 18.
- Es besagt, dass $\Phi(f) = f(A_{out}(S)) - f(A_{in}(S))$. Da $0 \leq f(e) \leq c(e)$ für alle $e \in A$, gilt

$$f(A_{out}(S)) = \sum_{e \in A_{out}(S)} f(e) \leq \sum_{e \in A_{out}(S)} c(e) = c(A_{out}(S)).$$

- Offensichtlich ist $f(A_{in}(S)) \geq 0$, also insgesamt

$$\Phi(f) \leq c(A_{out}(S)).$$

- Wir fassen dies in folgendem Lemma zusammen:

Lemma. *Ist f ein (s, t)-Fluss und (S, T) ein (s, t)-Schnitt, so gilt*

$$\Phi(f) \leq c(A_{out}(S)).$$

Da f und (S, T) beliebig wählbar sind, folgt:

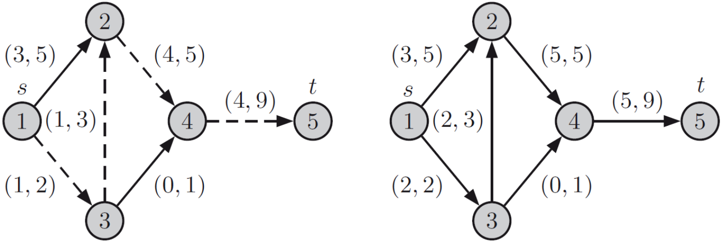
$$\max_{\substack{f \text{ ist} \\ (s,\,t)\text{-}\mathit{Fluss}}} \Phi(f) \leq \min_{\substack{(S,T) \text{ ist} \\ (s,\,t)\text{-}\mathit{Schnitt}}} c(A_{out}(S)).$$

In Worten: *Der maximale Flusswert ist höchstens so groß wie die Kapazität eines minimalen Schnittes.*

Korollar. *Falls f ein Fluss und (S, T) ein Schnitt sind, sodass der Flusswert $\Phi(f)$ gleich der Kapazität $c(A_{out}(S))$ des Schnittes ist, so ist f ein maximaler Fluss und (S, T) ein minimaler Schnitt.*

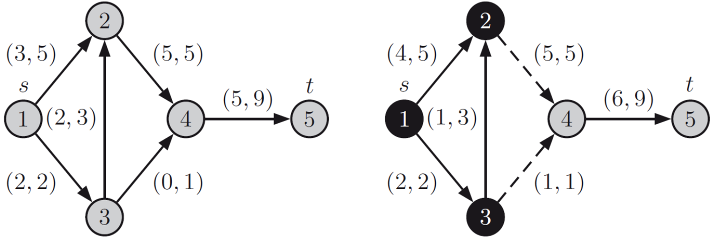
Residualnetze und flussvergrößernde Wege

Wenn wir den Fluss aus folgender Abbildung (links) betrachten, so sehen wir, dass für jede gerichtete Kante e des gestrichelt dargestellten Weges $P = \underbrace{1}_{s}, 3, 2, 4, \underbrace{5}_{t}$ die Ungleichung $f(e) < c(e)$ gilt. Wir können daher den Fluss auf P um $\delta_P := \min_{e \in A(P)}(c(e) - f(e)) = 1$ erhöhen, womit sich der Flusswert ebenfalls um δ_P auf 5 erhöht. Trivialerweise können wir also aus der Existenz eines Weges wie im Beispiel schließen, dass der gegebene Fluss nicht optimal ist. Der resultierende Fluss ist rechts dargestellt:



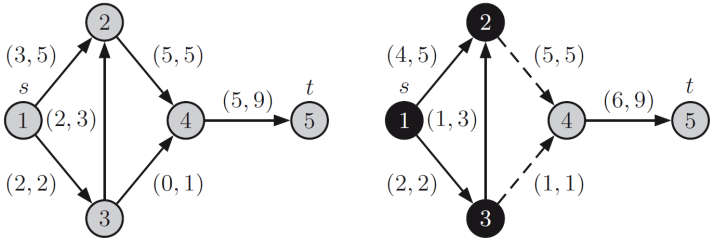
Jetzt existiert kein Weg mehr von s nach t, auf dem alle Kanten noch nicht ganz ausgelastet sind. Ist der erhaltene Fluss maximal? Oder mit anderen Worten: Ist die Nichtexistenz eines Weges mit Restkapazität nicht nur notwendig, sondern auch hinreichend für die Maximalität eines Flusses?

Frage wiederholt: Ist die Nichtexistenz eines Weges mit Restkapazität nicht nur notwendig, sondern auch **hinreichend** für die Maximalität eines Flusses?



Die Antwort ist: *Nein*. Die Abbildung rechts zeigt einen Fluss f mit Wert 6, womit der Fluss aus der Abbildung links, der den Flusswert 5 besitzt, nicht maximal sein kann. f ist übrigens ein maximaler Fluss, da der Schnitt $(\{s = 1, 2, 3\}, \{4, 5 = t\})$ (Kanten im Vorwärtsteil dieses minimalen Schnittes sind gestrichelt dargestellt) Kapazität 6 besitzt (s. Korollar Folie 24).

Wenn man die Flüsse aus der Abbildung vergleicht, so stellt man fest, dass sich der Fluss auf den Kanten $(1,2)$, $(3,4)$ und $(4,5)$ um eine Einheit erhöht hat, gleichzeitig der Flusswert auf $(3,2)$ um eins **gesunken** ist. Wir müssen offenbar auch die Möglichkeit des Fluss**abbaus** auf Kanten berücksichtigen.



In der folgenden Definition des Residualnetzes G_f lassen wir auch **untere** Kapazitätsschranken $\ell : E \to \mathbb{R}_{\geq 0}$ für den Fluss auf den Kanten zu und erlauben $c(e) = +\infty$ für eine Kante $e \in A$.

Residualnetz

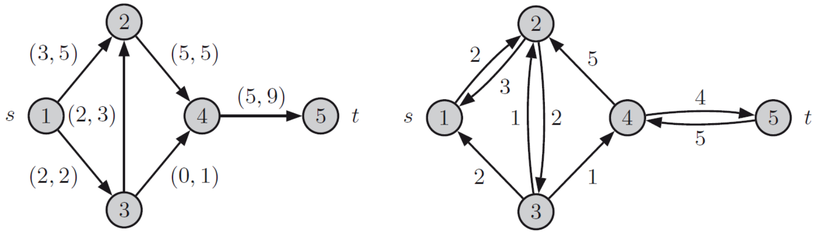
Es sei f ein Fluss in G und seien ℓ, c untere und obere Kapazitätsschranken für die gerichteten Kanten von G mit $0 \leq \ell(e) \leq c(e)$ für alle $e \in A$; man lässt in dieser allgemeinen Formulierung auch den Wert $c(e) = +\infty$ zu.

Das *Residualnetz* G_f besitzt die gleiche Knotenmenge wie G. Die Menge der gerichteten Kanten A_f im Residualnetz definiert sich wie folgt:

- Falls $e = (v, w) \in A$ und $f(e) < c(e)$, so enthält das Residualnetz G_f eine gerichtete Kante $+e = (v, w)$ sowie *Residualkapazität* $c_f(+e) := c(e) - f(e)$.
- Falls $e = (v, w) \in A$ und $f(e) > \ell(e)$, so enthält G_f eine gerichtete Kante $-e = (w, v)$ mit *Residualkapazität* $c_f(-e) := f(e) - \ell(e)$.

Wir werden uns in dieser Vorlesung stets auf den Fall $\ell(e) = 0$ und $c(e) < +\infty$ für alle $e \in A$ beschränken.

Residualnetz: Beispiel



Links: Ein Fluss f mit unterer Schranke $\ell(e) = 0$ für alle Kanten e. Jede Kante e ist mit dem Paar $(f(e), c(e))$ beschriftet.

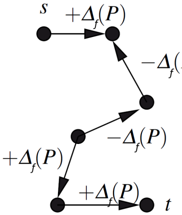
Rechts: Das zugehörige Residualnetz G_f. Für jede Kante e mit $f(e) < c(e)$ existiert in G_f eine Kante e mit Residualkapazität $c(e) - f(e)$, und für jede Kante $e = (v, w)$ mit $f(e) > \ell(e) = 0$ existiert in G_f eine Kante $-e =$ **(w, v)** mit Residualkapazität $f(e) - \ell(e) = f(e)$.

- Wir verwenden σ als Platzhalter für ein Vorzeichen, d.h. jede gerichtete Kante in G_f hat die Form σe für ein $e \in A$. Mit $-\sigma e$ bezeichnen wir dann die entsprechende inverse gerichtete Kante, also $-e = (w, v)$ für $+e := e = (v, w)$ und umgekehrt.
- Ein *st*-Weg P im Residualnetz G_f heißt *flussvergrößernder Weg* (*augmenting path*) für den Fluss f.
- Die Residualkapazität

$$\Delta_f(P) := \min_{\sigma e \in A(P)} c_f(\sigma e) > 0$$

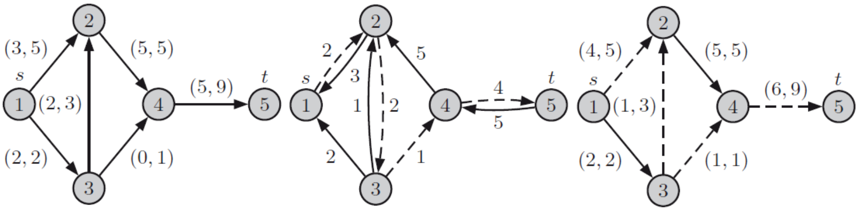
des Weges P ist die minimale Residualkapazität auf seinen gerichteten Kanten.

- Wir können nun den Fluss f längs P **erhöhen**: Falls die Kante e auf P liegt, so ist $f(e) + \Delta_f(P) \le c(e)$ und wir setzen $f'(e) := f(e) + \Delta_f(P)$.
- Liegt $-e$ auf P, so ist $f(e) \ge \Delta_f(P)$ und wir setzen $f'(e) := f(e) - \Delta_f(P)$.
- Für alle gerichteten Kanten e, die nicht auf P liegen, sei $f'(e) := f(e)$.
- Dann ist f' wieder ein Fluss in G und $\Phi(f') = \Phi(f) + \Delta_f(P) > \Phi(f)$.



Proposition. *Existiert ein flussvergrößernder Weg für f, so ist f kein maximaler Fluss.*

BEISPIEL (1/2)

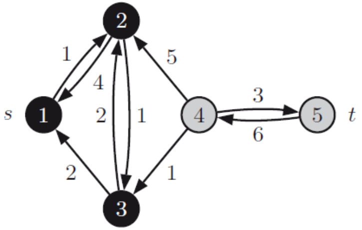


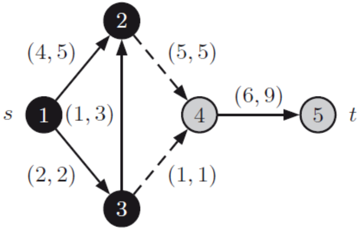
Links: Fluss mit Wert 5.

Mitte: Flussvergrößernder st-Weg P (gestrichelt) im Residualnetz G_f mit $\Delta_f(P) = 1$.

Rechts: Flusserhöhung längs P liefert neuen Fluss f' mit $\Phi(f') = \Phi(f) + \Delta_f(P) = \Phi(f) + 1$.

BEISPIEL (2/2)





Links: Das resultierende Residualnetz $G_{f'}$ hat **keinen Weg** von s nach t mehr. Die noch von der Quelle s im Residualnetz erreichbaren Knoten S sind schwarz hervorgehoben.

Rechts: Die im Residualnetz $G_{f'}$ von s aus erreichbaren Knoten S induzieren einen Schnitt (S, T) in G mit $c(A_{out}(S)) = 6 = \Phi(f')$. Die Kanten im Vorwärtsteil $A_{out}(S)$ sind gestrichelt dargestellt.

- Im Lemma von Folie 23 hatten wir gezeigt, dass die Kapazität jedes (s,t)-Schnittes eine obere Schranke für den Flusswert jedes zulässigen (s,t)-Flusses ist.
- Wir formulieren diese Eigenschaft nun mithilfe von Residualnetzen. Ausgangspunkt für dieses Lemma war die Identität

  $$\Phi(f) = f(A_{out}(S)) - f(A_{in}(S))$$

  aus dem Lemma von Folie 18, die für jeden Fluss f gilt.

- Es sei nun f^* ein maximaler (s, t)-Fluss und f ein beliebiger (s, t)-Fluss.
- Somit ist $\Phi(f^*) = \Phi(f) + \varepsilon$ für ein $\varepsilon \geq 0$.
- Nach dem Lemma von Folie 23 gilt dann für f^*

$$\Phi(f^*) = \Phi(f) + \varepsilon \leq c(A_{out}(S))$$

für jeden (s, t)-Schnitt (S, T).

$\Phi(f^*) = \Phi(f) + \varepsilon \leq c(A_{out}(S))$ und $\Phi(f) = f(A_{out}(S)) - f(A_{in}(S))$ liefern:

$$
\begin{aligned}
\varepsilon &\leq c(A_{out}(S)) - f(A_{out}(S)) + f(A_{in}(S)) \\
&= \sum_{e \in A_{out}(S)} (c(e) - f(e)) + \sum_{e \in A_{in}(S)} f(e) \\
&\overset{\text{im Residualnetz}}{=} \sum_{+e \in A_{out}(S)} c_f(+e) + \sum_{-e \in A_{out}(S)} c_f(-e) = c_f\left(A_{out}^{G_f}(S)\right),
\end{aligned}
$$

wobei

$$
A_{out}^{G_f}(S) := \{(v, w) \in A_f : v \in S \text{ und } w \in V \setminus S\}
$$

und A_f die Kantenmenge von G_f ist. Man beachte, dass S in Zeile 3 und 4 eine Teilmenge der Knotenmenge des ursprünglichen Flussnetzwerkes, in Zeile 5 jedoch die entsprechende Teilmenge der Knotenmenge des Residualnetzes G_f ist.

Hieraus folgt:

Lemma. *Ist f ein (s,t)-Fluss und f^* ein maximaler (s,t)-Fluss, so gilt*

$$\Phi(f^*) \leq \Phi(f) + c_f\left(A_{out}^{G_f}(S)\right)$$

für jeden (s,t)-Schnitt (S,T) in G_f.

DAS *Max-Flow Min-Cut* THEOREM

Zur Geschichte: Ford und Fulkerson schreiben in 1962:

"Determining a maximal steady state flow from one point to another in a network subject to capacity limitations on arcs [...] was posed to the authors in the spring of 1955 by T.E. Harris, who, in conjunction with General F.S. Ross (Ret.) had formulated a simplified model of railway traffic flow, and pinpointed this particular problem as the central one suggested by the model. It was not long after this until the main result, Theorem 5.1, which we call the max-flow min-cut theorem, was conjectured and established."

Satz (Ford und Fulkerson, 1956). *In einem gerichteten Graphen G mit Kapazitäten $c : A \to \mathbb{R}_{\geq 0}$ ist der Wert eines maximalen (s,t)-Flusses gleich der minimalen Kapazität eines (s,t)-Schnittes:*

$$\max_{\substack{f \text{ ist} \\ (s,t)\text{-}\textit{Fluss in } G}} \Phi(f) = \min_{\substack{(S,T) \text{ ist} \\ (s,t)\text{-}\textit{Schnitt in } G}} c(A_{out}(S)).$$

Dieser Satz ist bekannt als das *Max-Flow Min-Cut Theorem*. Er hat etliche wichtige und interessante kombinatorische Anwendungen. Als Beispiel geben wir etwas später Aussage und Beweis eines fundamentalen Satzes von Menger. Zunächst jedoch beweisen wir das Max-Flow Min-Cut Theorem und besprechen den Ford-Fulkerson-Algorithmus.

Beweis (1/3)

- Die elementare Proposition auf Folie 33 liefert eine **notwendige** Bedingung für die Maximalität eines Flusses f: Es darf kein flussvergrößernder Weg existieren. Wir zeigen nun, dass diese Bedingung auch **hinreichend** ist.
- Es sei f^* ein maximaler (s,t)-Fluss (die Existenz eines solchen Flusses folgt aus Stetigkeitsgründen). Nach der Proposition existiert kein flussvergrößernder Weg für f^*.
- Folglich ist t von s aus in G_{f^*} nicht erreichbar und die beiden Mengen

$$S := \{v \in V \ : \ v \text{ ist in } G_{f^*} \text{ von } s \text{ erreichbar}\}$$

$$T := \{v \in V \ : \ v \text{ ist in } G_{f^*} \text{ von } s \text{ nicht erreichbar}\}$$

sind nicht leer ($s \in S$ und $t \in T$) und definieren damit einen Schnitt (S, T).

BEWEIS (2/3)

- Es sei $e \in A_{out}(S)$ eine Kante im Vorwärtsteil des Schnittes. Dann gilt $f^*(e) = c(e)$, denn sonst wäre $+e = (v, w)$ eine Kante in G_{f^*} und w von s in G_{f^*} erreichbar im Widerspruch zu $w \in T$ (wir haben $v \in S$ und somit ist nach Definition von S der Knoten v von s aus in G_{f^*} erreichbar). Dies zeigt

$$f^*(A_{out}(S)) = c(A_{out}(S)).$$

- Analog muss für jede Kante $e = (v, w) \in A_{in}(S)$ gelten, dass $f^*(e) = 0$, da sonst $-e$ eine Kante in G_{f^*} wäre und mit w auch $v \in T$ von s aus erreichbar wäre – Widerspruch zur Definition von T. Also ist

$$f^*(A_{in}(S)) = 0.$$

Beweis (3/3)

- Aus den Gleichungen $f^*(A_{out}(S)) = c(A_{out}(S))$ und $f^*(A_{in}(S)) = 0$ folgt:

$$c(A_{out}(S)) = f^*(A_{out}(S)) - f^*(A_{in}(S)) \overset{\text{Lemma Folie 18}}{=} \Phi(f^*).$$

- Wegen des Korollars auf Folie 24 muss f^* ein maximaler Fluss und gleichzeitig (S, T) ein minimaler Schnitt sein, d.h. ein Schnitt mit minimaler Kapazität.

Der gerade geführte Beweis zeigt auch folgende Charakterisierung maximaler Flüsse:

Satz (Augmenting-Path-Theorem). *Ein* (s,t)*-Fluss* f *ist genau dann ein maximaler Fluss, wenn es keinen flussvergrößernden Weg gibt, d.h. keinen Weg von* s *nach* t *im Residualnetz* G_f.

Der Algorithmus von Ford und Fulkerson

- Ein gerichteter Graph G ist *schwach zusammenhängend*, wenn es in dem G zugrunde liegenden ungerichteten Graphen zwischen beliebigen zwei Knoten einen Weg gibt, dieser also zusammenhängend ist.
- Im Folgenden nehmen wir für die Analyse von Algorithmen zur Bestimmung maximaler Flüsse an, dass der gerichtete Graph G des Flussnetzes einfach und schwach zusammenhängend ist.
- Wir wiederholen das **Maximalflussproblem**: Es sei $G = (V, A)$ ein gerichteter Graph mit Kapazitäten $c : A \to \mathbb{R}_{\geq 0}$ und zwei Knoten $s, t \in V$ mit $s \neq t$. Gesucht ist ein maximaler (s, t)-Fluss.

- Das Augmenting-Path-Theorem motiviert in naheliegender Weise die Idee zu einem Algorithmus zur Bestimmung eines maximalen Flusses: Wir starten mit dem Nullfluss $f \equiv 0$ (d.h. f transportiert 0 Einheiten auf jeder Kante).
- Solange G_f einen Weg von s nach t besitzt, erhöhen wir den Fluss längs dieses Weges. Danach aktualisieren wir G_f.
- Dieser Algorithmus geht auf Ford und Fulkerson zurück. Wir bezeichnen ihn im Folgenden mit $\mathfrak{F}$.

Der Ford-Fulkerson-Algorithmus $\mathfrak{F}$

FORD-FULKERSON(G, c, s, t)

Input: Ein einfacher gerichteter Graph $G = (V, A)$ in Adjazenzlistendarstellung; eine nichtnegative Kapazitätsfunktion $c\colon A \to \mathbb{R}_+$, zwei Knoten $s, t \in V$.

Output: Ein maximaler (s,t)-Fluss f (falls c ganzzahlig ist bzw. bei »geschickter« Wahl der flussvergrößernden Wege).

```
Setze f(e) = 0 für alle e ∈ A, d.h. starte mit dem Nullfluss f ≡ 0.
while in G_f existiert ein Weg von s nach t do
    Wähle einen solchen Weg P.
    Setze Δ := min { c_f(σe) : σe ∈ A(P) }      { Residualkapazität des Weges P }
    Erhöhe f längs P um Δ Einheiten.
    Aktualisiere G_f.
```

- Ist $c : A \to \mathbb{N}$, so wird durch obigen Algorithmus der Fluss in jedem Erhöhungsschritt um einen ganzzahligen Betrag erhöht: Ist der aktuelle Fluss f ganzzahlig, so sind alle Residualkapazitäten als Differenzen von ganzen Zahlen wieder ganzzahlig.
- Somit ist jeder Fluss, der zwischenzeitlich entsteht, ganzzahlig.
- Außerdem erhöht sich der Fluss in jedem Schritt um mindestens 1.
- Es sei $C := \max\{c(e) : e \in A\}$ die größte auftretende Kapazität. Dann enthält der spezielle Schnitt $(\{s\}, V \setminus \{s\})$ höchstens $n - 1$ gerichtete Kanten (wobei $n := |V|$), von denen jede Kapazität höchstens C hat.
- Daher besitzt jeder Schnitt, und damit auch ein minimaler (s,t)-Schnitt, Kapazität höchstens $(n-1)C$.

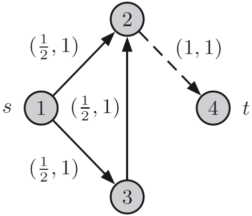
- Aus diesem Grund muss $\mathfrak{F}$ nach maximal $(n-1)C$ Iterationen terminieren, weil er keinen flussvergrößernden Weg mehr findet.
- Nach dem Augmenting-Path-Theorem ist der bei Abbruch gefundene Fluss – der nach unseren vorherigen Überlegungen ganzzahlig ist – dann auch maximal.
- Einen flussvergrößernden Weg im Residualnetz können wir etwa mit Hilfe der Tiefensuche (DFS) oder Breitensuche (BFS) in $O(n+m)$ Zeit finden. Damit haben wir folgenden Satz bewiesen:

Satz. *Sind alle Kapazitäten ganzzahlig, so bricht* $\mathfrak{F}$ *nach* $O(nC)$ *Vergrößerungsschritten und* $O((n+m)nC)$ *Zeit mit einem maximalen Fluss ab, der ganzzahlig ist. Hierbei bezeichnet* $C := \max\{c(e) : e \in A\}$ *die größte auftretende Kantenkapazität.*

Korollar (Ganzzahligkeitssatz). *Sind alle Kapazitäten ganzzahlig, so existiert immer ein maximaler Fluss, der ganzzahlig ist.*

Frage. Gilt sogar, dass bei ganzzahligen Kapazitäten *jeder* maximale Fluss ganzzahlig ist?

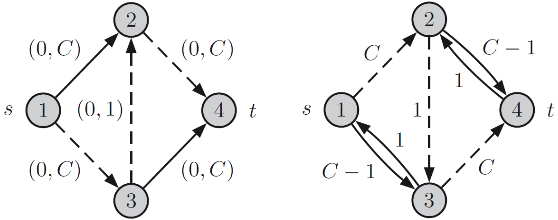
Nein, es stimmt **nicht**, dass jeder maximale Fluss ganzzahlig sein muss! Das Korollar zeigt nur, dass mindestens ein maximaler Fluss existiert, der zusätzlich ganzzahlig ist. Beispielsweise zeigt die Abbildung unten einen maximalen Fluss, der nicht ganzzahlig ist, obwohl alle Kapazitäten ganzzahlig sind.



Auch bei ganzzahligen Kapazitäten kann ein maximaler Fluss existieren, der nicht ganzzahlig ist. Die gestrichelte Kante ist der Vorwärtsteil eines minimalen Schnittes. Nach dem Korollar existiert aber mindestens ein ganzzahliger maximaler Fluss (hier z.B.: schicke 1 Einheit von s nach Knoten 2 nach t).

Es sollte bemerkt werden, dass $\mathfrak{F}$ bei ungeschickter Wegeauswahl extrem lange benötigen kann. Es gibt Beispiele, bei denen dann wirklich $\Omega(nC)$ Flusserhöhungen vorgenommen werden. Ein solches Beispiel folgt. Falls $C = 2^n$, so besitzt $\mathfrak{F}$ **exponentielle** Laufzeit.

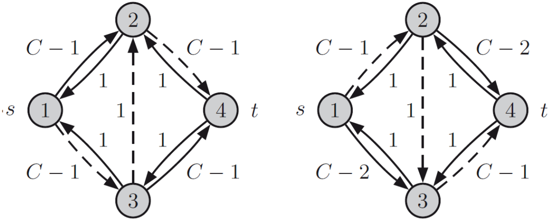
BEISPIEL (1/2)



Links: Das Ausgangsnetzwerk entspricht dem Residualnetzwerk G_f für den Nullfluss $f \equiv 0$ (also $f(e) = 0$ für alle $e \in A$). Der flussvergrößernde Weg 1324 hat Residualkapazität 1.

Rechts: Nach der Flusserhöhung dreht sich im Residualnetzwerk die Richtung der Kante (3,2) um. Der flussvergrößernde Weg 1234 im Residualnetzwerk hat wieder Residualkapazität 1.

Beispiel (2/2)



Links: Wählt man nun wieder den flussvergrößernden Weg 1324, so hat dieser wieder Residualkapazität 1.

Rechts: Auch im nächsten Schritt hat dann der Weg 1234 wieder Residualkapazität 1, sodass sich der Flusswert auch wieder um 1 erhöht.

Und so weiter.

Dieses Beispiel zeigt, dass $\mathfrak{F}$ bei ungeschickter Wahl des flussvergrößernden Weges $\Omega(nC)$ Iterationen benötigen kann. Im gerade besprochenen Beispiel werden abwechselnd die flussvergrößernden Wege 1324 und 1234 gewählt. In jeder Iteration erhöht sich der Flusswert um 1, der maximale Flusswert ist $2C$.

Sind die Kapazitäten rational, so gilt:

Satz. *Sind alle Kapazitäten* ***rationale*** *Zahlen, so bricht der Ford-Fulkerson-Algorithmus $\mathfrak{F}$ nach endlich vielen Iterationen ab.*

Beweis. Es sei K der gemeinsame Hauptnenner aller Kapazitäten. Dann sind während der Iteration des Algorithmus $\mathfrak{F}$, ähnlich wie im ganzzahligen Fall, alle zwischenzeitlichen Flusswerte auf den Kanten ganzzahlige Vielfache von $1/K$. Daraus folgt die Endlichkeit, da wiederum der Flusswert nach oben beschränkt ist.

Obwohl wir reelle Zahlen normalerweise nicht exakt in unserem Berechnungsmodell darstellen können, ist es für theoretische Zwecke zeitweise nützlich, auch irrationale Zahlen als Eingaben zuzulassen. In diesem Fall gehen wir davon aus, dass wir arithmetische Operationen auf reellen Zahlen ebenfalls in konstanter Zeit ausführen können.

Allerdings ist für den Ford-Fulkerson-Algorithmus $\mathfrak{F}$ auf Basis flussvergrößernder Wege im reellen Fall der **Abbruch nach endlich vielen Iterationen nicht gesichert**. Es gibt nicht allzu komplizierte Netzwerke, in denen bei ungeschickter Wahl der Wege der Algorithmus nicht nur nicht terminiert, sondern der **Grenzwert der erzeugten Flüsse kein maximaler Fluss** ist.

Zusammenfassend:

- Bei irrationalen Kapazitäten kann es vorkommen, dass der Ford-Fulkerson-Algorithmus $\mathfrak{F}$ immer weitere Flusserhöhungen vornimmt, ohne jemals zu terminieren. Desweiteren gibt es Netzwerke, in denen $\mathfrak{F}$ nicht gegen den Maximalflusswert konvergiert.
- Auch bei ganzzahligen Kapazitäten ist die Laufzeit von $\mathfrak{F}$ nicht polynomiell, $\mathfrak{F}$ terminiert jedoch stets und der Maximalflusswert wird gefunden. Dabei kann immer ein Maximalfluss gefunden werden, der auf jeder Kante eine ganzzahlige Quantität transportiert.
- Eine polynomielle Laufzeit erhält man aber, wenn man vergrößernde Wege *geschickt* wählt. Dazu betrachten wir im Laufe dieses Kapitels einen weiteren berühmten Algorithmus zur Bestimmung maximaler Flüsse.
- Zunächst jedoch diskutieren wir eine wichtige kombinatorische Anwendung des Max-Flow Min-Cut Theorems:

Satz von Menger

Es seien s und t verschiedene Knoten in einem Graphen G. Zwei st-Wege P und Q in G heißen *intern disjunkt*, falls $V(P) \cap V(Q) = \{s, t\}$.

In einem zusammenhängenden Graphen $G = (V, E)$ ist ein *Separator* (*separator* oder auch *vertex-cut*) eine Menge $S \subset V$, sodass $G - S$ nicht zusammenhängend ist.

Satz (Menger, 1927). *Es sei $G = (V, E)$ ein Graph und $s, t \in V$ mit $\{s, t\} \notin E$. Dann ist die Kardinalität eines kleinsten Separators S für den s und t in verschiedenen Zusammenhangskomponenten von $G - S$ liegen gleich der maximalen Anzahl intern disjunkter st-Wege.*

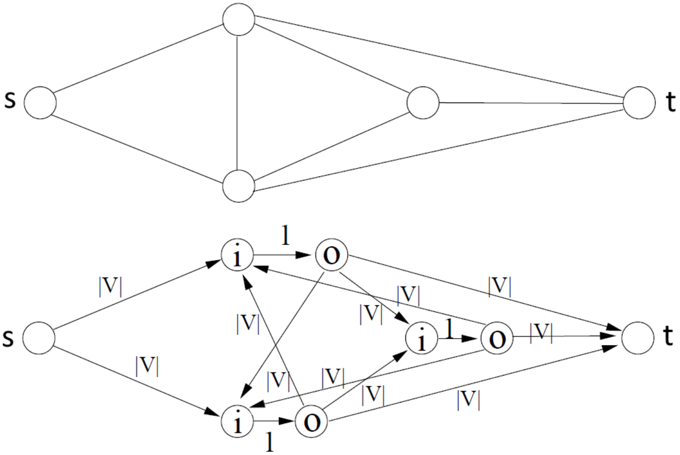
Aus diesem Satz folgt ein fundamentales Korollar.
Wiederholung: Ein Graph G der Ordnung $> k$ heißt *k-zusammenhängend* ist, falls für jede Menge $S \subset V(G)$ von höchstens $k-1$ Knoten gilt, dass $G - S$ zusammenhängend ist.

Korollar (Menger, 1927). *Ein Graph G ist k-zusammenhängend genau dann, wenn es zwischen beliebigen zwei Knoten v und w in G genau k intern disjunkte vw-Wege gibt.*

Beweis des mengerschen Satzes von Folie 59 (1/8)

- Wir sehen in $G = (V, E)$ die Knoten s und t als Quelle bzw. Senke.
- Für jeden Knoten $x \in V \setminus \{s, t\}$ führen wir zwei Knoten ein, x_{in} und x_{out}, und die gerichtete Kante (x_{in}, x_{out}) mit Kapazität 1.
- Für jede Kante $\{x, y\} \in E$ mit $\{x, y\} \cap \{s, t\} = \emptyset$ fügen wir die gerichteten Kanten (x_{out}, y_{in}) und (y_{out}, x_{in}) hinzu, für jede Kante $\{s, x\}$ die gerichtete Kante (s, x_{in}) und für jede Kante $\{t, x\}$ die gerichtete Kante (x_{out}, t). All diese Kanten bekommen eine sehr große Kapazität, z.B. $|V|$.
- Wir erhalten ein Netzwerk N.
- Folgende Abbildung illustriert die Konstruktion:

BEWEIS (2/8)



Abkürzend schreiben wir in dieser Figur i für x_{in} und o für x_{out}. Jede Kante der Form $x_{in}x_{out}$ = io hat Kapazität 1. Alle anderen Kanten haben Kapazität $|V|$.

Beweis (3/8)

- Jeder Weg $sx^1x^2\dots x^kt$ in G entspricht einem gerichteten Weg $sx^1_{in}x^1_{out}x^2_{in}x^2_{out}\dots x^k_{in}x^k_{out}t$ in N.
- Sind zwei Wege in G intern disjunkt, so sind die entsprechenden Wege in N ebenso intern disjunkt.
- Haben wir ℓ paarweise intern disjunkte st-Wege in G, dann können wir längs jedem der entsprechenden Wege in N einen Fluss mit Wert 1 transportieren, es existiert also ein Fluss auf N mit Wert mindestens ℓ.
- Haben wir einen Fluss mit Wert ℓ in N, so können wir ℓ paarweise intern disjunkte st-Wege in G finden. Diesen Zwischenschritt beweisen wir nun.

BEWEIS (4/8)

- Wir können annehmen, dass wir es mit einem Maximalfluss zu tun haben.
- Wir nutzen die Tatsache, dass es einen Maximalfluss gibt, bei dem auf jeder Kante eine Quantität $\in \mathbb{Z}$ transportiert wird (Folie 51).
- Wir können ebenso ausnutzen, dass es in N keinen gerichteten Kreis C gibt mit Wert $w > 0$ auf jeder Kante. Sollte dies doch der Fall sein, so können wir für alle Kanten $e \in E(C)$ den Flusswert von e auf 0 setzen und erhalten somit wieder einen Maximalfluss.

Beweis (5/8)

- Per Konstruktion von N sind alle gerichteten st-Wege in N derart, dass jede zweite Kante von der Form (x^i_{in}, x^i_{out}) ist, da es keine andere Möglichkeit gibt, einen Knoten x^i_{in} zu verlassen bzw. in einen Knoten x^i_{out} zu gelangen.
- Außerdem fließt für jeden ganzzahligen Fluss in N, per Konstruktion, entweder 0 oder 1 durch eine Kante der Form (x^i_{in}, x^i_{out}). Entweder benutzt ein Fluss eine Kante dieser Form, oder nicht (es gibt nichts dazwischen).
- Folgen wir nun in N gerichteten Wegen mit Wert 1 auf jeder Kante von s nach t, so können wir derart – z.B. durch Entfernen aller Kanten des st-Weges, sobald wir diesen abgeschlossen haben – so viele paarweise kantendisjunkte st-Wege in G finden, wie es Einheiten im (ganzzahligen) Maximalfluss gibt.
- Aufgrund der Konstruktion von N sind diese dann auch intern disjunkt in G.

BEWEIS (6/8)

- Haben wir also einen Maximalfluss mit Wert ℓ in N, so erhalten wir ℓ intern disjunkte *st*-Wege in G.
- Wähle nun ein $S \subset V$ mit $s \in S$, $t \in V \setminus S$, sodass

$$c(A_{out}(S)) := \sum_{e \in A_{out}(S)} c(e)$$

minimal ist, wobei

$$A_{out}(S) := \{(v, x) \in \underbrace{A(N)}_{\text{Kanten von } N} : v \in S, x \in V \setminus S\}.$$

Beweis (7/8)

- Betrachten wir Kanten aus $A_{out}(S)$, so wird deutlich, dass diese alle von der Form (x^i_{in}, x^i_{out}) sind:
- Nehmen wir s mitsamt seinen Nachbarn als Menge S, dann haben wir bereits einen (s, t)-Schnitt von obiger Form – solche Schnitte existieren also – und jedes S wofür auch nur eine Kante in $A_{out}(S)$ nicht von obiger Form ist, liefert bereits eine größere Kapazität als dieser Schnitt (denn Kapazitäten auf Kanten $\neq (x^i_{in}, x^i_{out})$ hatten wir auf $|V|$ gesetzt).
- Betrachten wir in G alle Knoten x^i mit $(x^i_{in}, x^i_{out}) \in A_{out}(S)$, dann erhalten wir eine Knotenmenge von Kardinalität $c(A_{out}(S)) = \ell$, die ein Separator ist dessen Entfernen s und t in verschiedenen Zusammenhangskomponenten hinterlässt.

Beweis (8/8)

- Zusammenfassend liefert dies, dass ein minimaler Separator in G, dessen Entfernen s und t in verschiedenen Zusammenhangskomponenten hinterlässt, gleich groß ist wie die minimale Kapazität $c(A_{out}(S))$ einer Menge S in N mit $s \in S$ und $t \notin S$.
- Nach dem Max-Flow Min-Cut Theorem ist diese minimale Kapazität gleich dem Maximalflusswert.
- Wir hatten bereits gezeigt, dass der Maximalflusswert in N und die Anzahl intern disjunkter st-Wege in G gleich ist.
- Somit ist die Größe eines minimalen Separators in G, dessen Entfernen s und t in verschiedenen Zusammenhangskomponenten hinterlässt, gleich der Anzahl intern disjunkter st-Wege in G und der Satz von Menger (Folie 59) ist bewiesen.

Der Algorithmus von Edmonds und Karp

Im Algorithmus $\mathfrak{F}$ bleibt zunächst offen, wie wir in Schritt 3 einen flussvergrößernden Weg wählen. Zur Flusserhöhung (und zum Beweis des Satzes und Korollars auf Folie 51) genügt es, irgendeinen flussvergrößernden Weg zu finden.

Dazu müssen wir im Residualnetz G_f einen st-Weg finden, bzw. feststellen, dass es keinen solchen Weg gibt. Wie bereits erwähnt können wir mithilfe der Breitensuche (BFS) diese Aufgabe in $O(n + m)$ Zeit lösen.

Da die Breitensuche immer einen kürzesten st-Weg (gemessen an der Anzahl der gerichteten Kanten) in G_f liefert, finden alle Flussvergrößerungen auf kürzesten Wegen statt. Diese Auswahl der flussvergrößernden Wege in $\mathfrak{F}$ liefert den Algorithmus von Edmonds und Karp:

DER ALGORITHMUS VON EDMONDS UND KARP

EDMONDS-KARP-MAXFLOW(G, c, s, t)

Input: Ein einfacher gerichteter Graph $G = (V, A)$ in Adjazenzlistendarstellung; eine nichtnegative Kapazitätsfunktion $c \colon A \to \mathbb{R}_+$, zwei Knoten $s, t \in V$.

Output: Ein maximaler (s, t)-Fluss f.

Setze $f(e) = 0$ für alle $e \in A$, d.h. starte mit dem Nullfluss $f \equiv 0$.
while in G_f existiert ein Weg von s nach t **do**
    Wähle einen kürzesten solchen Weg P.
    Setze $\Delta := \min\{\, c_f(\sigma e) : \sigma e \in A(P) \,\}$ { *Residualkapazität des Weges P* }
    Erhöhe f längs P um Δ Einheiten.
    Aktualisiere G_f.

Um die Laufzeit des Algorithmus von Edmonds und Karp zu beweisen, zeigen wir zunächst ein hilfreiches Lemma.

Lemma. *Es sei H ein gerichteter Graph und $s, t \in V(H)$. Wir bezeichnen mit* $\operatorname{dist}(s, t, H)$ *die Länge eines (bezüglich der Kantenanzahl) kürzesten st-Weges in H und mit $A_{st}(H)$ die Menge aller gerichteten Kanten von H, die auf (bezüglich der Kantenanzahl) kürzesten st-Wegen liegen. Es sei*

$$A_{st}(H)^{-1} := \{(v, u) : (u, v) \in A_{st}(H)\}.$$

Dann gilt für den Graphen H', der durch Hinzufügen aller gerichteten Kanten aus $A_{st}(H)^{-1}$ zu H entsteht:

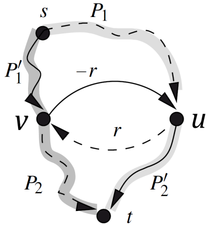
$$\operatorname{dist}(s, t, H') = \operatorname{dist}(s, t, H) \quad \textit{und} \quad A_{st}(H') = A_{st}(H).$$

Beweis (1/2)

- Per Induktion genügt es zu zeigen, dass sich $\text{dist}(s, t, H)$ und $A_{st}(H)$ nicht ändern, wenn wir für ein $(u, v) \in A_{st}(H)$ die inverse Kante (v, u) zu H hinzufügen.
- Falls die Behauptung falsch ist, so gibt es einen st-Weg

$$P' := P'_1(v, u)P'_2$$

(laufe also zuerst längs P'_1 von s bis v, verwende dann die Kante $-r := (v, u)$, und laufe anschließend längst P'_2 von u nach t) in H', welcher (v, u) benutzt, und dessen Länge höchstens $\text{dist}(s, t, H)$ ist.



Beweis (2/2)

- Wegen $(u, v) \in A_{st}(H)$ existiert aber auch ein *st*-Weg

$$P := P_1(u, v)P_2$$

  in H (und somit auch in H') der Länge $\mathrm{dist}(s, t, H)$, welcher (u, v) benutzt.
- Dann sind aber P_1P_2' und $P_1'P_2$ beides *st*-Wege in H, die zusammen höchstens

$$2 \cdot \mathrm{dist}(s, t, H) - 2$$

  Kanten besitzen.
- Folglich hat einer dieser Wege Länge höchstens $\mathrm{dist}(s, t, H) - 1$ im Widerspruch zur Definition von $\mathrm{dist}(s, t, H)$.

Satz. *Es sei G ein Netzwerk mit n Knoten und m Kanten und mit ganzzahligen, rationalen oder reellen Kapazitäten. Der Algorithmus von Edmonds und Karp terminiert nach $O(nm)$ Iterationen mit einem maximalen Fluss. Die Gesamtkomplexität des Algorithmus ist $O(nm^2)$.*

BEWEIS (1/6): BEHAUPTUNG 1

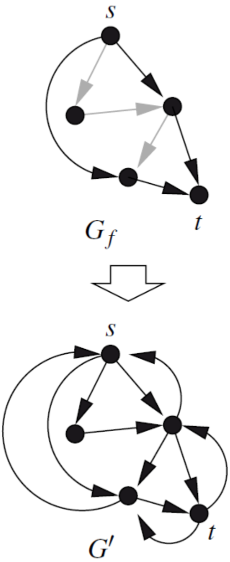
Es sei f der Fluss zu Beginn einer Iteration. Wiederholung: $A_{st}(G)$ ist die Menge aller gerichteten Kanten von G, die auf (bezüglich der Kantenanzahl) kürzesten st-Wegen liegen.

Behauptung 1. *Durch Hinzufügen aller Kanten aus $\{-\sigma e : \sigma e \in A_{st}(G_f)\}$ zum Residualnetz G_f entsteht der Graph G'. Es gilt in G':*

$$\operatorname{dist}(s,t,G') = \operatorname{dist}(s,t,G_f) \quad \textit{und} \quad A_{st}(G') = A_{st}(G_f).$$

Beweis. Lemma Folie 71 mit $H := G_f$ und $H' := G'$.

Beweis (2/6): Behauptung 1



Hinzufügen aller inversen Kanten zu den Kanten auf kürzesten st-Wegen in G_f (schwarze Kanten) liefert den Graphen G'.

Beweis (3/6): Behauptung 2

Es sei f der Fluss zu Beginn einer Iteration und f' der Fluss nach der Iteration, in der f längs eines kürzesten Weges im Residualnetz G_f erhöht wird.

Behauptung 2. *Es gilt*

$$\operatorname{dist}(s, t, G_{f'}) \geq \operatorname{dist}(s, t, G_f).$$

Falls Gleichheit gilt, so haben wir $A_{st}(G_{f'}) \subsetneq A_{st}(G_f)$.

Beweis. $G_{f'}$ ist ein Untergraph (:= induzierter Teilgraph) von G' (so wie in Behauptung 1 definiert), da nur gerichtete Kanten auf kürzesten st-Wegen zur Flusserhöhung benutzt werden. Daher gilt dann

$$\operatorname{dist}(s, t, G_{f'}) \geq \operatorname{dist}(s, t, G') = \operatorname{dist}(s, t, G_f),$$

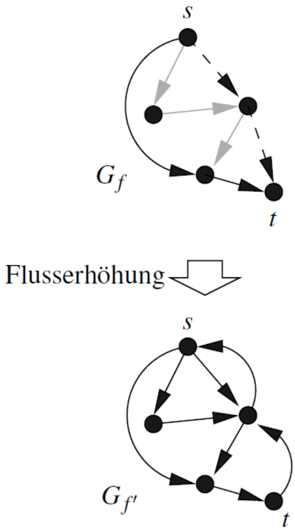
wobei sich die letzte Gleichheit aus Behauptung 1 ergibt.

Beweis (4/6): Behauptung 2

Falls Gleichheit gilt, so ist $A_{st}(G_{f'}) \subset A_{st}(G') = A_{st}(G_f)$, da $G_{f'}$ Untergraph von G' ist und somit jeder kürzeste st-Weg in $G_{f'}$ ein kürzester st-Weg in G' ist.

Es gilt strikte Inklusion, da bei der Flusserhöhung mindestens eine der gerichteten Kanten auf dem kürzesten Weg aus dem Residualnetz, also aus $A_{st}(G_f)$, verschwindet (s. nächste Abbildung zur Illustration) – schließlich wird das Minimum über die Residualkapazitäten (Folie 70, Schritt 4) in mindestens einer Kante des kürzesten Weges realisiert.

BEWEIS (5/6): BEHAUPTUNG 2

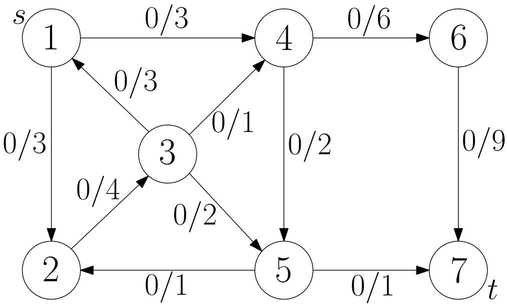


Flusserhöhung entlang eines kürzesten Weges in G_f (gestrichelt) ergibt ein neues Residualnetz $G_{f'}$, das ein Untergraph von G' ist.

Beweis (6/6)

- Aus Behauptung 2 folgt nun der Satz: Der Abstand von s zu t im Residualnetz ist monoton wachsend und nach oben durch $n - 1$ beschränkt.
- Falls der Abstand zwischen zwei Iterationen nicht echt anwächst, so schrumpft die Menge der gerichteten Kanten auf kürzesten st-Wegen.
- Dies kann hintereinander maximal $2m$ mal passieren, da jedes Residualnetz höchstens $2m$ gerichtete Kanten enthält.

Edmonds-Karp-Algorithmus: Beispiel (1/8)



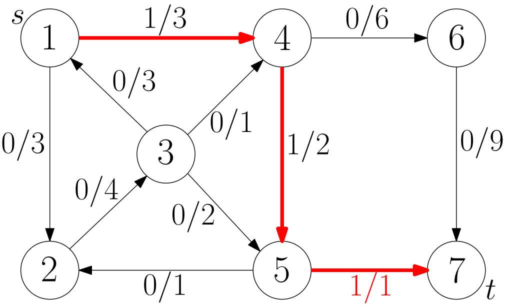
Gegeben sei obiges Netzwerk mit Quelle $s = 1$ und Senke $t = 7$. Auf jeder Kante notieren wir das Paar f/c, wobei f der derzeitige Fluss ist und c die Kantenkapazität. In jedem Schritt suchen wir nun **flussvergrößernde *st*-Wege minimaler Länge im Residualnetz**.

Edmonds-Karp-Algorithmus: Beispiel (2/8)

Wiederholung: Das *Residualnetz* G_f besitzt die gleiche Knotenmenge wie G. Die Menge der gerichteten Kanten A_f im Residualnetz definiert sich wie folgt:

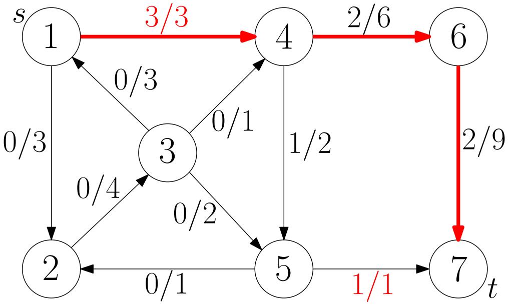
- Falls $e = (v, w) \in A$ und $f(e) < c(e)$, so enthält das Residualnetz G_f eine gerichtete Kante $+e = (v, w)$ sowie Residualkapazität $c_f(+e) := c(e) - f(e)$.
- Falls $e = (v, w) \in A$ und $f(e) > 0$, so enthält G_f eine gerichtete Kante $-e = (w, v)$ mit Residualkapazität $c_f(-e) := f(e)$.

EDMONDS-KARP-ALGORITHMUS: BEISPIEL (3/8)



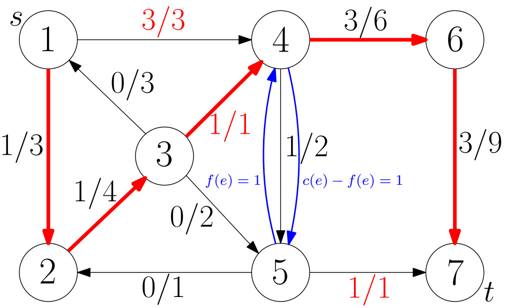
Flussvergrößernder minimaler *st*-Weg: 1457 (in rot). Länge: 3.
Erhöhung um: $\min(c_f((1,4)), c_f((4,5)), c_f((5,7))) = \min(3-0, 2-0, 1-0) = \min(3,2,1) = 1.$
Sind auf einer Kante Fluss und Kapazität gleich, so stellen wir dies mit roten Zahlen dar.

Edmonds-Karp-Algorithmus: Beispiel (4/8)



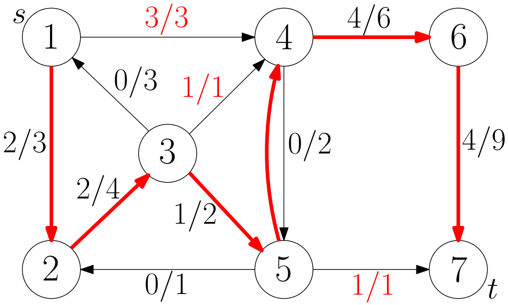
Flussvergrößernder minimaler *st*-Weg: 1467. Länge: 3.
Erhöhung um: $\min(c_f((1,4)), c_f((4,6)), c_f((6,7))) =$
$\min(3-1, 6-0, 9-0) = \min(2,6,9) = 2$.

Edmonds-Karp-Algorithmus: Beispiel (5/8)



Flussvergrößernder minimaler *st*-Weg: 123467. Länge: 5.
Erhöhung um:
$\min(c_f((1,2)), c_f((2,3)), c_f((3,4)), c_f((4,6)), c_f((6,7))) =$
$\min(3-0, 4-0, 1-0, 6-2, 9-2) = \min(3,4,1,4,7) = 1.$

Edmonds-Karp-Algorithmus: Beispiel (6/8)



Flussvergrößernder minimaler st-Weg: 1235467. Länge: 6.
Erhöhung um:
$\min(c_f((1,2)), c_f((2,3)), c_f((3,5)), c_f((5,4)), c_f((4,6)), c_f((6,7))) =$
$\min(3-1, 4-1, 2-0, 1-0, 6-3, 9-3) = \min(2,3,2,1,3,6) = 1.$

Edmonds-Karp-Algorithmus: Beispiel (7/8)

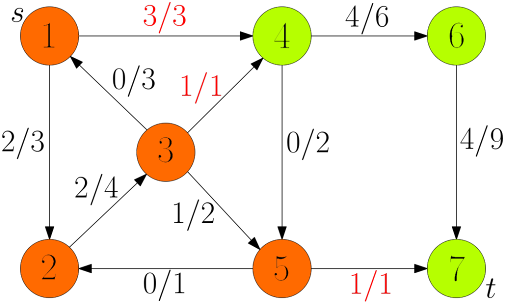
- Die Länge des (roten) flussvergrößernden Weges im Residualnetz nimmt nie ab.
- Diese Wege sind stets die kürzestmöglichen *st*-Wege.
- Der gefundene Flusswert 5 ist gleich der Kapazität eines minimalen (s, t)-Schnittes und somit nach dem Max-Flow Min-Cut Theorem maximal:
- Es gibt nur einen derartigen Schnitt in diesem Graphen, nämlich $(\{1, 2, 3, 5\}, \{4, 6, 7\})$, mit Kapazität

$$c((1,4)) + c((3,4)) + c((5,7)) = 3 + 1 + 1 = 5,$$

siehe nächste Folie.

Edmonds-Karp-Algorithmus: Beispiel (8/8)

Man beachte, dass der Algorithmus nicht nur einen Maximalfluss bestimmt, sondern zeitgleich anhand der maximal ausgelasteten Kanten einen minimalen (s,t)-Schnitt (den wir gerade angegeben haben und der hier mit orange/grün dargestellt ist):



0-1-Netzwerke

In diesem Abschnitt wird ein wichtiger Spezialfall von allgemeinen Netzwerken betrachtet: Netzwerke, in denen jede Kante die Kapazität 0 oder 1 hat. Solche Netzwerke nennt man *0-1-Netzwerke*. Sie treten in vielen Anwendungen auf. Auf 0-1-Netzwerken existieren maximale Flüsse mit speziellen Eigenschaften.

Satz. *Es sei G ein 0-1-Netzwerk. Dann existiert ein maximaler Fluss f, welcher auf jeder Kante den Wert* 1 *oder* 0 *hat. Ferner gibt es* $\Phi(f)$ *st-Wege, welche paarweise keine Kante gemeinsam haben. Die Kanten dieser Wege haben alle den Fluss* 1.

Anwendungen von Netzwerkalgorithmen: Matchings

Wiederholung: Es sei $G = (V, E)$ ein ungerichteter Graph. $M \subseteq E$ heißt *Matching* von G, falls die Kanten in M paarweise keine gemeinsamen Knoten haben. Ein Matching hat *maximale Kardinalität*, wenn es kein Matching mit mehr Kanten gibt. Ein Matching, dass in keinem anderen Matching echt enthalten ist, heißt *maximal*.

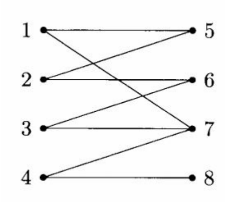
Ein Matching M heißt *perfekt*, falls jeder Knoten des Graphen mit einer Kante aus M inzident ist.

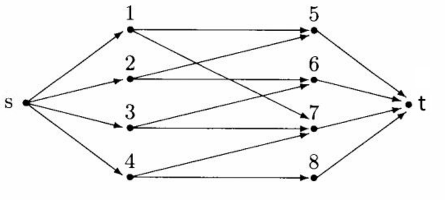
Ein maximales Matching M eines Graphen G kann durch keine weitere Kante vergrößert werden. In diesem Fall gibt es zu jeder Kante e_G von G eine Kante e_M aus M, sodass e_G und e_M mindestens einen Knoten teilen.

Matchings in bipartiten Graphen

- Es sei G ein bipartiter Graph mit Partitionsklassen V_1, V_2 (also $V = V_1 \cup V_2$) und N_G folgendes 0-1-Netzwerk:
- Die Knotenmenge V_G von N_G ist gleich $V \cup \{s, t\}$, d.h. es werden zwei neue Knoten eingeführt.
- Für jeden Knoten $v \in V_1$ gibt es in N_G eine Kante von s nach v mit Kapazität 1 und für jeden Knoten $w \in V_2$ eine Kante von w nach t mit Kapazität 1.
- Ferner gibt es für jede Kante $\{v, w\}$ von G eine gerichtete Kante von v nach w mit Kapazität 1 $(v \in V_1, w \in V_2)$.

Matchings in bipartiten Graphen





Ein bipartiter Graph G (rechts) und sein 0-1-Netzwerk N_G (links). Im rechten Bild hat jede gerichtete Kante Kapazität 1.

Ein Fluss heisst *binär*, wenn er auf jeder Kante den Wert 0 oder 1 hat. Man beachte, dass nicht jeder maximale Fluss auf einem 0-1-Netzwerk ein binärer Fluss ist.

Proposition. *Die Anzahl der Kanten in einem Matching maximaler Kardinalität eines bipartiten Graphen G ist gleich dem Wert eines maximalen Flusses auf N_G. Ist f ein maximaler binärer Fluss in N_G, so bilden die Kanten aus N_G mit Fluss* 1 *ein Matching maximaler Kardinalität in G.*

Beweis (1/2)

Es sei f ein maximaler binärer Fluss auf dem 0-1-Netzwerk N_G. Ferner sei M die Menge aller Kanten aus G, für die f durch die entsprechende Kante in N_G genau 1 Mengeneinheit transportiert.

Da jeder Knoten aus V_1 (G ist bipartiter Graph mit Partitionsklassen V_1, V_2) in N_G Eingangsgrad 1 und jeder Knoten aus V_2 in N_G Ausgangsgrad 1 hat, folgt aus der Flusserhaltungsbedingung, dass M ein Matching in G ist. Ferner enthält M genau $\Phi(f)$ Kanten.

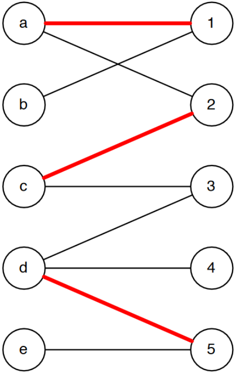
Beweis (2/2)

Es sei nun umgekehrt M ein Matching maximaler Kardinalität von G mit z Kanten. Dann lässt sich leicht ein Fluss f mit Wert z auf N_G konstruieren: Für jede Kante $\{v, w\} \in M$ ($v \in V_1, w \in V_2$) definiert man

$$f((s, v)) = f((v, w)) = f((w, t)) = 1$$

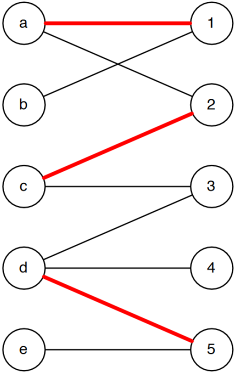
und $f(e) = 0$ für alle anderen Kanten e von N_G. Die Flusserhaltungsbedingung ist für f erfüllt, und es gilt $\Phi(f) = z$. Damit ist die Proposition bewiesen.

BEISPIEL (1/3)



Ein bipartiter Graph G und ein Matching M in rot. Ist dieses Matching maximal?

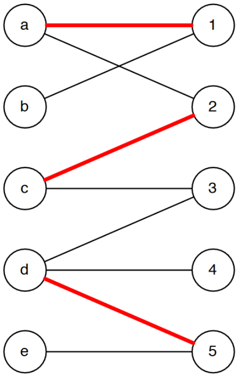
Beispiel (1/3)



Ein bipartiter Graph G und ein Matching M in rot. Ist dieses Matching maximal?

Ja. Hat M maximale Kardinalität?

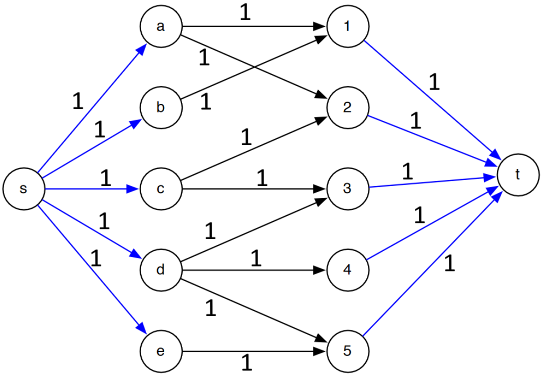
BEISPIEL (1/3)



Ein bipartiter Graph G und ein Matching M in rot. Ist dieses Matching maximal?

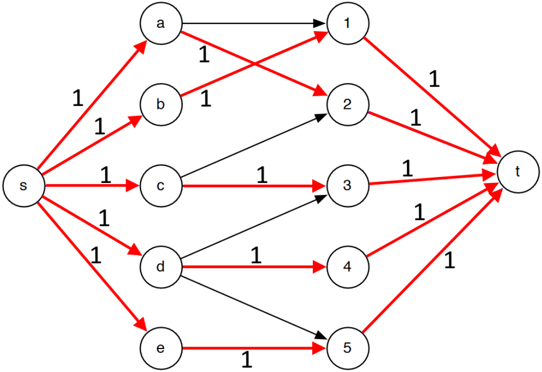
Ja. Hat M maximale Kardinalität? **Nein**; wir überprüfen dies indem wir das zugehörige Netzwerk N_G konstruieren:

BEISPIEL (2/3)



Das Hilfsnetzwerk N_G. Jede Kante hat Kapazität 1.

Beispiel (3/3)



Ein (offensichtlich) maximaler Fluss mit Wert 5. Somit enthält ein Matching mit maximaler Kardinalität in G genau 5 Kanten.