Agenda

3. Graphentheorie

Flüsse in Netzwerken: Beispiele

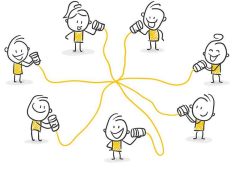
Wasserversorgung und -nutzung

Verkehr

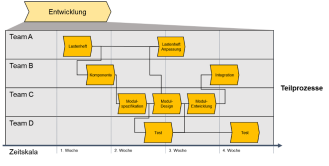
Stromnetze

Erdgas Pipelines

Kommunikation



Prozessplanung



Netzwerk: Definition

Ein Netzwerk ist ein Tupel *(V,E,s,t,u,c)* bestehend aus:

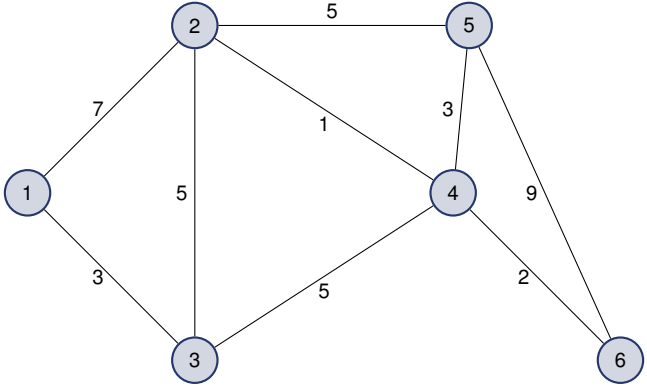
- einem endlichen, gerichteten Graphen (V,E)
  - V: Menge an Knoten (*engl.: vertices*)
  - E: Menge an Kanten (*engl.: edges*)
- mindestens zwei spezifische Knoten $s,t \in V$ um einen Fluss im Netzwerk darzustellen
  - s: Quelle (*engl.: source*)
  - t: Senke (*engl.: sink*)
- einer Kapazitätsfunktion $u : E \rightarrow R^+$
  - Ein Netzwerk wird auch "gewichteter Graph"genannt.
- Zusätzlich können den Kanten auch noch Kosten zugeordnet sein in Form einer Kostenfunktion $c : E \rightarrow R^+$

Adjazenz- und Inzidenzmatrizen sind essentiell um Netzwerke darzustellen

Wir werden uns in der Veranstaltung hauptsächlich mit zwei Fragestellungen auseinander setzen:

- Was ist der maximale Fluss zwischen zwei Knoten? (Ford-Fulkerson Algorithmus, Max-Flow, Min-Cut)
- Was ist die günstigste (kostenminimale) Fluss zwischen zwei Knoten? (Min-Cost)

Der Ford Fulkerson Algorithmus ist eine Methode um den maximalen Fluss in einem Netzwerk zwischen zwei Knoten zu ermitteln



Was ist der maximale Fluss zwischen den Knoten 1 und 6?

Lester Randolph Ford junior

Leben:

- *23.09.1927 (Houston) †27.02.2017
- Sohn von Lester Randolph Ford senior (Mathematiker)
- Arbeitete für CEIR Inc. und Rand Corporation

Hauptwerk:

- „Flows in Networks“ mit D.R. Fulkerson (1962)

Wirkung:

- Algorithmus von Ford und Fulkerson
- Bellman-Ford-Algorithmus

Delbert Ray Fulkerson

Leben:

- *14.08.1924 (Tamms, Illinois) †10.01.1976
- Studium und Ph.D. in Mathematik
- Arbeitete für die Rand Corporation

Hauptwerk:

- „Flows in Networks“ mit L.R. Ford junior (1962)

Wirkung:

- Algorithmus von Ford und Fulkerson
- Fulkerson Prize (Mathematik) nach ihm benannt

Ford Fulkerson Algorithmus

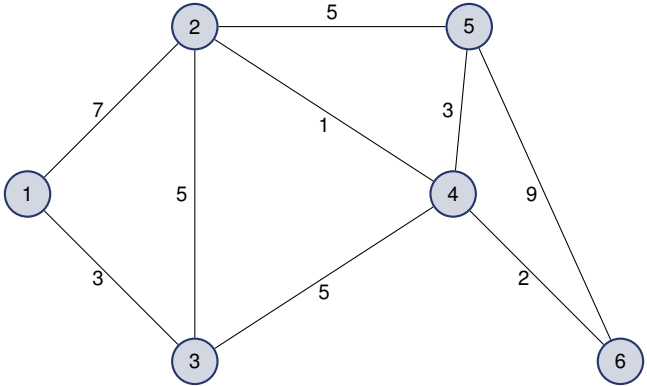
- Der Algorithmus bestimmt die maximale Kapazität eines Netzwerks.
- Für ein Netzwerk $G = (V, E, s, t, u)$ mit gegebenen Kantengewichten $u(e) = u(i, j)$ kann die maximale Kapazität bestimmt werden indem man die folgenden Schritte ausführt:

1. Initialisierung: Setze den Fluss auf jeder Kante auf 0.
2. Bilde den Residualgraphen für das Netzwerk. Direkt nach der Initialisierung sieht dieser genau wie der Originalgraph aus.
3. Finde einen Pfad P von s zu t für den es noch Kapazität im Residualgraphen gibt.
4. Bestimme die Kapazität $cap(P)$ des Pfads im Residualgraph.
5. Ziehe die Kapazität $cap(p)$ von allen Kanten im Residualgraphen ab.
6. Wiederhole den Prozess ab Schritt 3 bis kein Pfad mit positiver Kapazität mehr zwischen s und t im Residualgraphen existiert.

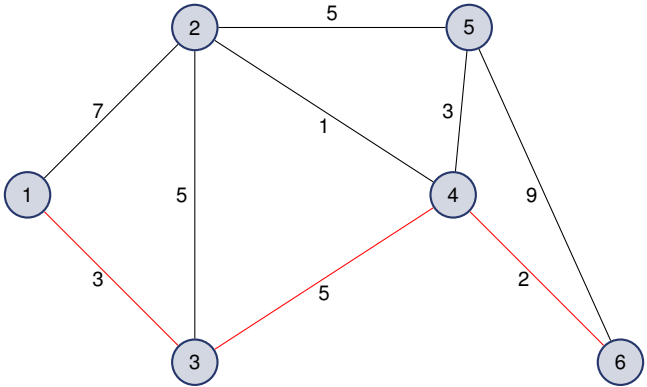
Der Maximale Fluss im Netzwerk ist die Summe aller $cap(p)$.

Ford Fulkerson Algorithmus I

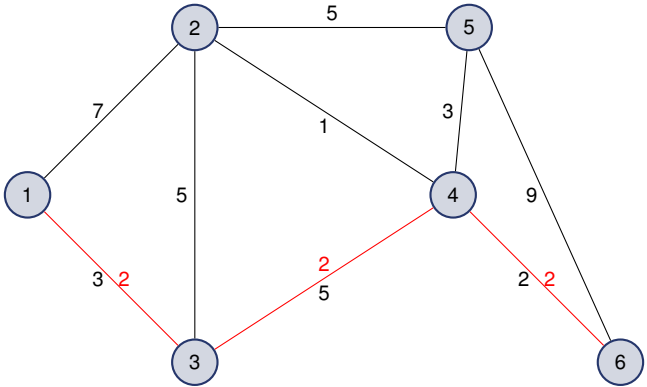
Schritt 1 + 2: Fluss aller Kanten auf 0 setzen und den Residualgraphen aufstellen. Sieht aus wie der Originalgraph.



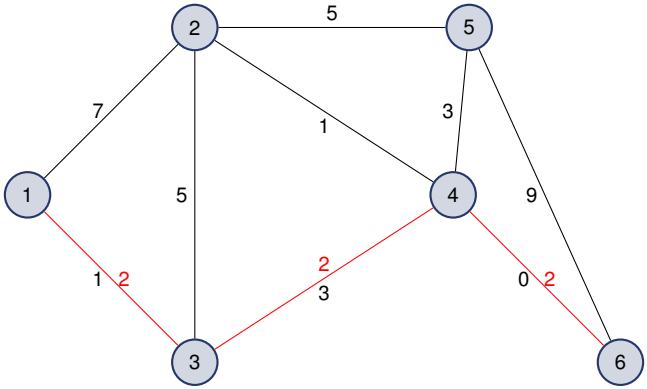
Schritt 3-5: Finde einen Pfad P von 1 zu 6 für den es noch Kapazität im Residualgraphen gibt. Bestimme die Kapazität und ziehe Sie von den Kantenkapazitäten ab.



Schritt 3-5: Finde einen Pfad P von 1 zu 6 für den es noch Kapazität im Residualgraphen gibt. Bestimme die Kapazität und ziehe Sie von den Kantenkapazitäten ab.

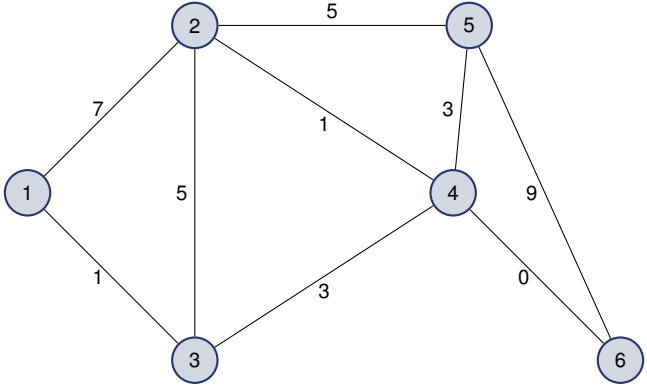


Schritt 3-5: Finde einen Pfad P von 1 zu 6 für den es noch Kapazität im Residualgraphen gibt. Bestimme die Kapazität und ziehe Sie von den Kantenkapazitäten ab.



Gesamtkapazität: 2 (P1)

Schritt 6: Wiederhole Schritte 3-5 solange es noch einen möglichen Pfad durch das Netzwerk gibt mit positiver Kapazität. Ansonsten Abbruch.

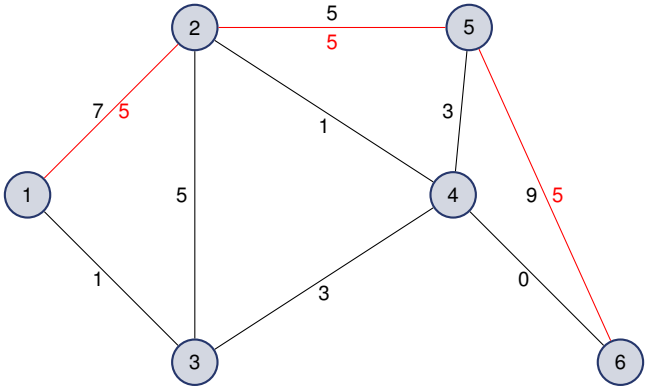


Gesamtkapazität: 2 (P1)

Gibt es noch einen möglichen Pfad mit Kapazität? –> Ja!

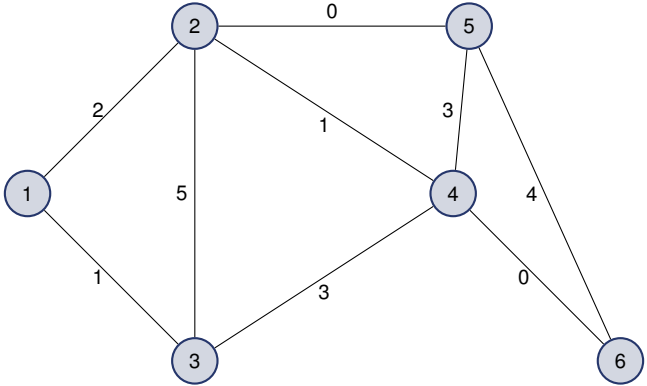
Ford Fulkerson Algorithmus III

Schritt 3-5: Finde einen Pfad P von 1 zu 6 für den es noch Kapazität im Residualgraphen gibt. Bestimme die Kapazität und ziehe Sie von den Kantenkapazitäten ab.



Gesamtkapazität: 2 (P1) + 5 (P2)

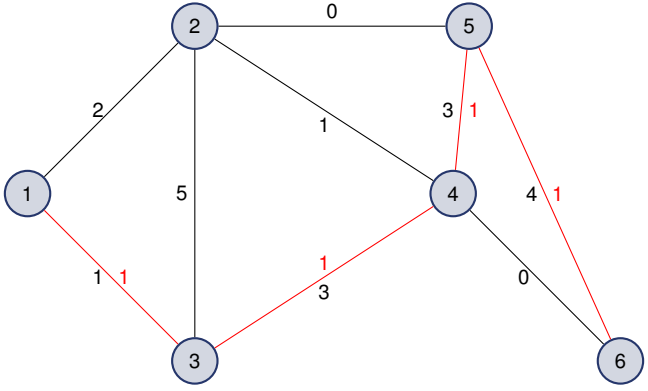
Schritt 6: Wiederhole Schritte 3-5 solange es noch einen möglichen Pfad durch das Netzwerk gibt mit positiver Kapazität. Ansonsten Abbruch.



Gesamtkapazität: 2 (P1) + 5 (P2)

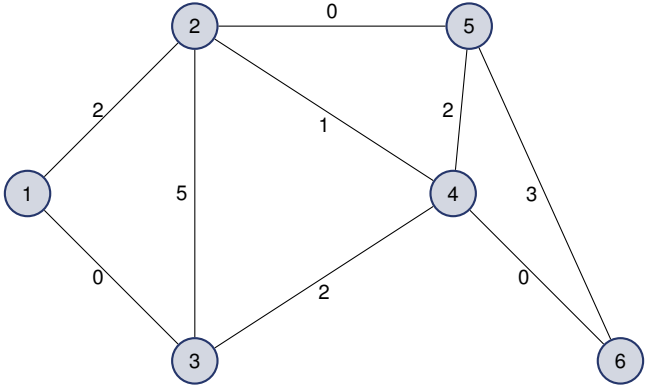
Gibt es noch einen möglichen Pfad mit Kapazität? –> Ja!

Schritt 3-5: Finde einen Pfad P von 1 zu 6 für den es noch Kapazität im Residualgraphen gibt. Bestimme die Kapazität und ziehe Sie von den Kantenkapazitäten ab.



Gesamtkapazität: 2 (P1) + 5 (P2) + 1 (P3)

Schritt 6: Wiederhole Schritte 3-5 solange es noch einen möglichen Pfad durch das Netzwerk gibt mit positiver Kapazität. Ansonsten Abbruch.

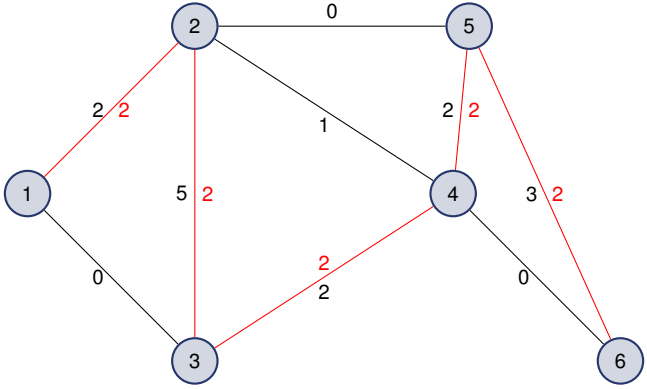


Gesamtkapazität: 2 (P1) + 5 (P2) + 1 (P3)

Gibt es noch einen möglichen Pfad mit Kapazität? –> Ja!

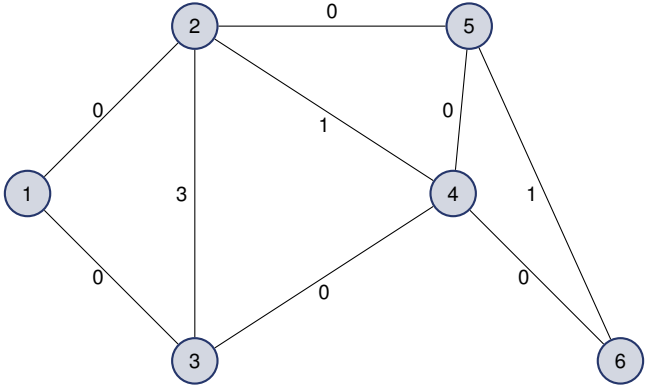
Ford Fulkerson Algorithmus V

Schritt 3-5: Finde einen Pfad P von 1 zu 6 für den es noch Kapazität im Residualgraphen gibt. Bestimme die Kapazität und ziehe Sie von den Kantenkapazitäten ab.



Gesamtkapazität: 2 (P1) + 5 (P2) + 1 (P3) + 2 (P4)

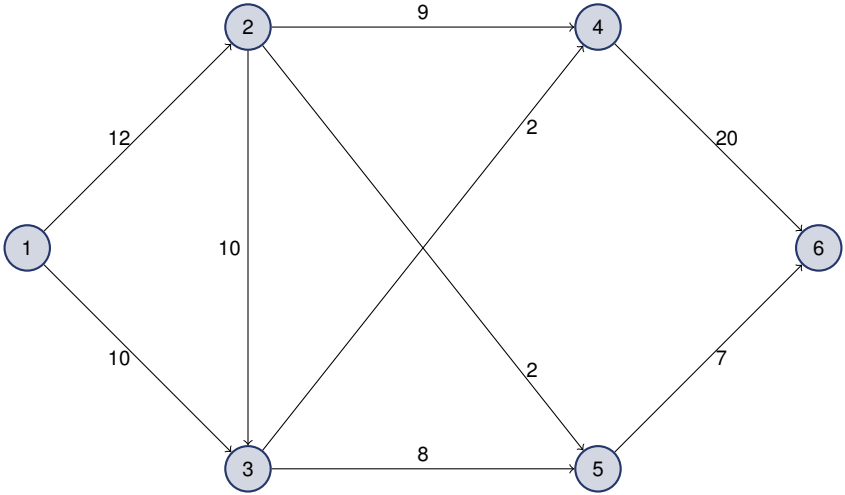
Schritt 6: Wiederhole Schritte 3-5 solange es noch einen möglichen Pfad durch das Netzwerk gibt mit positiver Kapazität. Ansonsten Abbruch.



Gesamtkapazität: 2 (P1) + 5 (P2) + 1 (P3) + 2 (P4)

Gibt es noch einen möglichen Pfad mit Kapazität? –> Nein! –> **Gesamtkapazität des Netzwerks = 10**

Flüsse in Netzwerken: Beispiel



Was ist der maximale Fluss durch das Netzwerk?

Fluss-Nebenbedingungen

Brutto-Flussfunktion $g : E \to R^+$, unter den folgenden beiden Nebenbedingungen

- Flusserhaltung (Kirchhoff's erstes Gesetz, 'Massenbilanz')

$$(\forall i \neq s,t) \sum_j \sum_{e \in e(i,j)} g(e) - \sum_j \sum_{e \in e(j,i)} g(e) = 0$$

- Kapazitäts-Nebenbedingung

$$(\forall e)\, g(e) \leq u(e)$$

Netto-Flussfunktion $f : E \to R^+$

$$f(i,j) = \sum_{e \in e(i,j)} g(e) - \sum_{e \in e(j,i)} g(e)$$

- f ist eine antisymetrische Funktion mit

$$f(i,j) = -f(j,i)$$

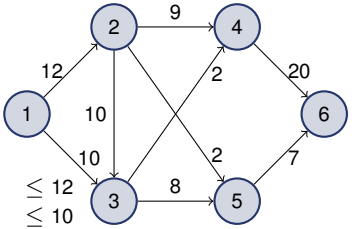
Die oben beschriebenen Nebenbedingungen gelten auch für Netto-Flussfunktionen!

Frage: Was ist der maximale Fluss, der durch das Netzwerk fließen kann?

- Trick: Fluss **aus** der Quelle wird maximiert.
- Kapazitäten der Kanten dürfen nicht überschritten werden.
- Fluss darf nicht verloren gehen (Massenbilanz).

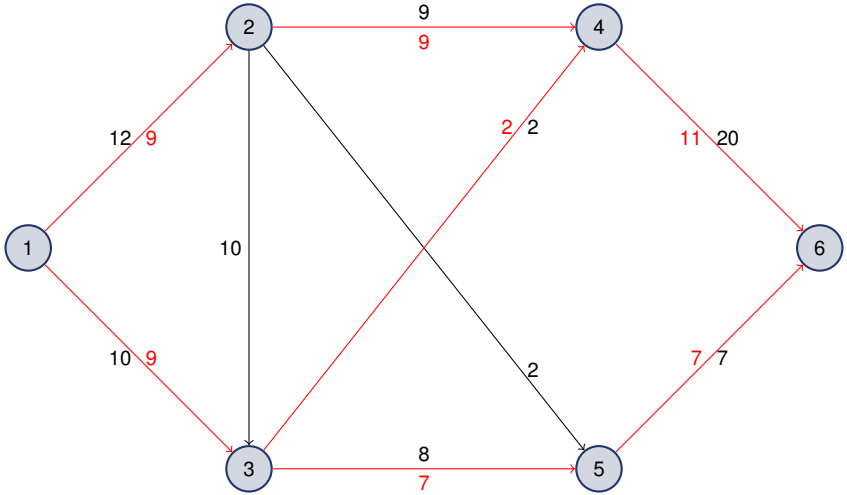
$$
\begin{aligned}
\max z = \quad & \sum_{j \in V:(s,j) \in E} g(s,j) & & & & \\
\text{s.t.} \quad & g(i,j) & & \leq & u(i,j) & \quad \forall (i,j) \in E \\
& g(i,j) & & = & 0 & \quad \forall (i,j) \notin E \\
& \sum_{j \in V:(i,j) \in E} g(i,j) & - \sum_{j \in V:(j,i) \in E} g(j,i) & = & 0 & \quad \forall i \in V \setminus \{s,t\} \\
& g(i,j) & & \geq & 0 & \quad \forall (i,j) \in E
\end{aligned}
$$

Flussmaximierungsproblem als LP



$$
\begin{array}{rrrrrrrrrrrl}
\max z = & g_{1,2} & +g_{1,3} & & & & & & & & & \\
\text{s.t.} & g_{1,2} & & & & & & & & & & \leq 12 \\
 & & g_{1,3} & & & & & & & & & \leq 10 \\
 & & & g_{2,3} & & & & & & & & \leq 10 \\
 & & & & g_{2,4} & & & & & & & \leq 9 \\
 & & & & & g_{2,5} & & & & & & \leq 2 \\
 & & & & & & g_{3,4} & & & & & \leq 2 \\
 & & & & & & & g_{3,5} & & & & \leq 8 \\
 & & & & & & & & g_{4,6} & & & \leq 20 \\
 & & & & & & & & & g_{5,6} & & \leq 7 \\
 & -g_{1,2} & & +g_{2,3} & +g_{2,4} & +g_{2,5} & & & & & & = 0 \\
 & & -g_{1,3} & -g_{2,3} & & & +g_{3,4} & +g_{3,5} & & & & = 0 \\
 & & & & -g_{2,4} & & -g_{3,4} & & +g_{4,6} & & & = 0 \\
 & & & & & -g_{2,5} & & -g_{3,5} & & +g_{5,6} & & = 0 \\
 & g_{1,2}, & g_{1,3}, & g_{2,3}, & g_{2,4}, & g_{2,5}, & g_{3,4}, & g_{3,5}, & g_{4,6}, & g_{5,6}, & & \geq 0
\end{array}
$$

Flüsse in Netzwerken: Beispiel



Maximaler Fluss: 18

Analog zu Bruttoflüssen können auch Nettoflüsse genutzt werden, um das Flussmaximierungsproblem zu formulieren.

$$
\begin{aligned}
\max z = \quad & \sum_{j \in V:(s,j)} f(s,j) & & & \\
\text{s.t.} \quad & f(i,j) & \leq & u(i,j) & \forall (i,j) \in E \\
& f(i,j) & = & 0 & \forall (i,j) \notin E \\
& \sum_{j \in V:(i,j) \in E} f(i,j) & = & 0 & \forall i \in V \setminus \{s,t\} \\
& f(i,j) & = & -f(j,i) & \forall (i,j) \in E \\
& f(i,j) & \in & \mathbb{R} & \forall (i,j) \in E
\end{aligned}
$$

Max Flow Min Cut Theorem

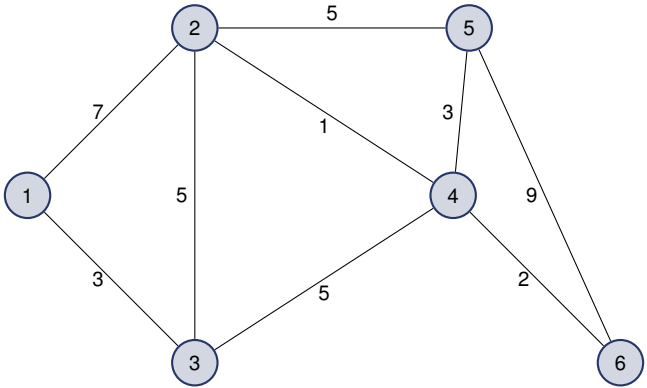
Cut: Ein Schnitt ("cut") eines Netzwerks ist eine Trennung (S,T) von V so dass

- S beinhaltet die Quelle s, T beinhaltet die Senke t
- Die Vereinigung von S und T ist V: $S \cup T = V$
- Der Schnitt von S und T ist leer: $S \cap T = \emptyset$

Max Flow Min Cut Theorem (auch Ford-Fulkerson Theorem genannt)
Die folgenden Aussagen sind äquivalent:

- f ist der maximale Fluss in G
- Es gibt einen Schnitt (S,T) in G mit der Kapazität von $|f|$

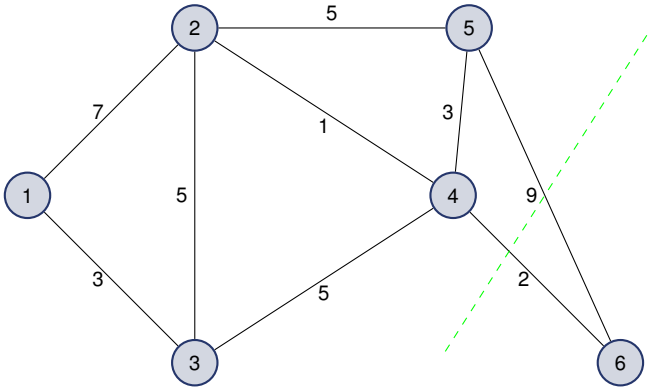
Min Cut in Netzwerken: Beispiel (I)



Wir suchen einen Schnitt durch das Netzwerk, sodass Quelle (1) und Senke (6) in unterschiedlichen Teilgraphen sind. Die Kapazität des Schnitts ist die Summe der Kantenkapazitäten die "durchtrennt"wurden.

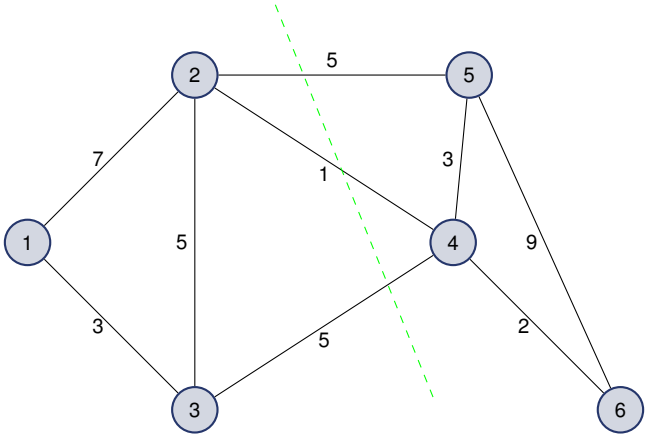
我们正在寻找一个网络的割，使得源节点（1）和汇节点（6）位于不同的子图中。割的容量是被“切断”的边的容量之和。

Min Cut in Netzwerken: Beispiel (I)



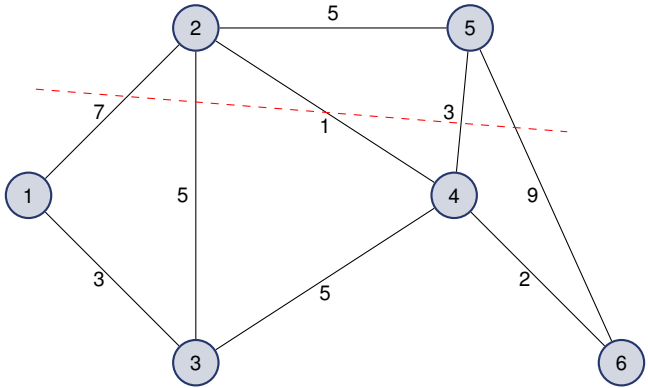
Erlaubter Schnitt, Kapazität = 11.

Min Cut in Netzwerken: Beispiel (I)



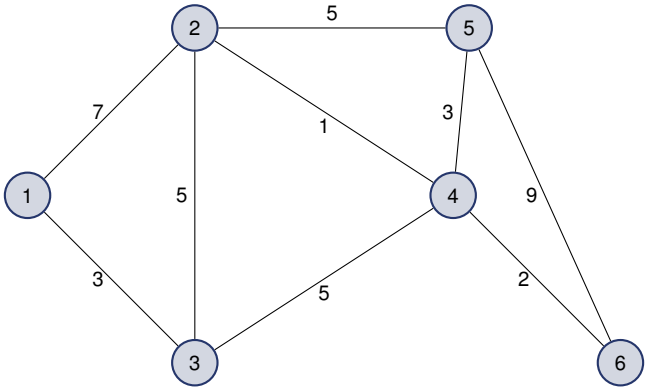
Erlaubter Schnitt, Kapazität = 11.

Min Cut in Netzwerken: Beispiel (I)



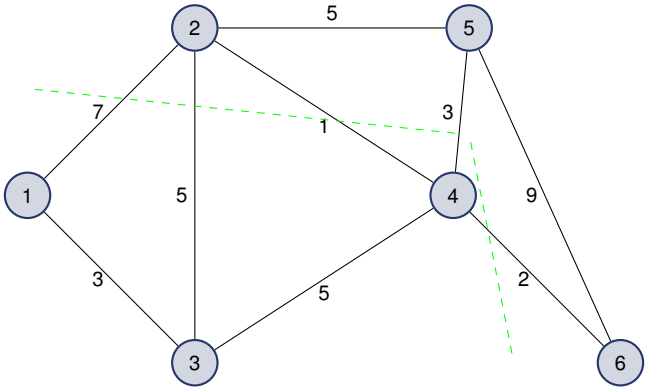
Nicht erlaubter Schnitt, Quelle und Senke in gleichem Teilgraphen!

Min Cut in Netzwerken: Beispiel (I)



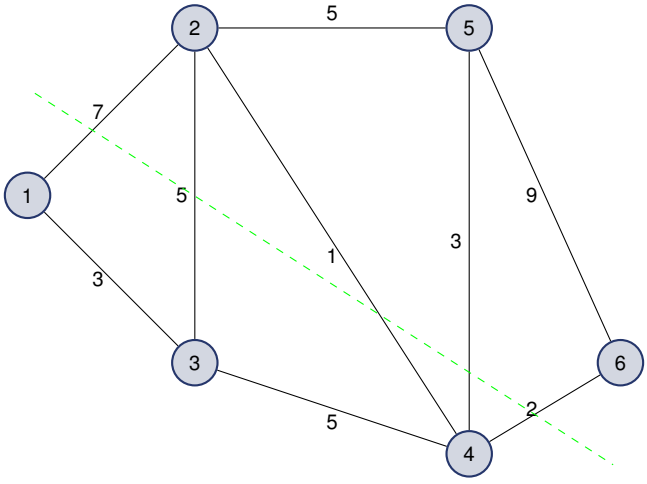
Frage: Ist es möglich eine Linie zu ziehen, sodass die Knoten (1), (3) und (4) in einem Teilgraphen sind und die Knoten (2), (5) und (6) in dem anderen?

Min Cut in Netzwerken: Beispiel (I)



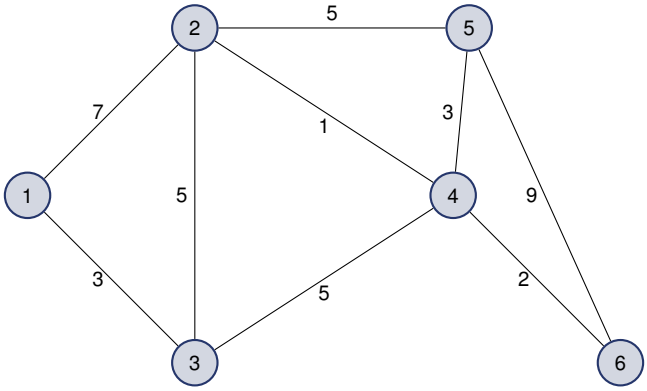
Graphisch: keine einzelne Linie aber trotzdem legitim!

Min Cut in Netzwerken: Beispiel (I)



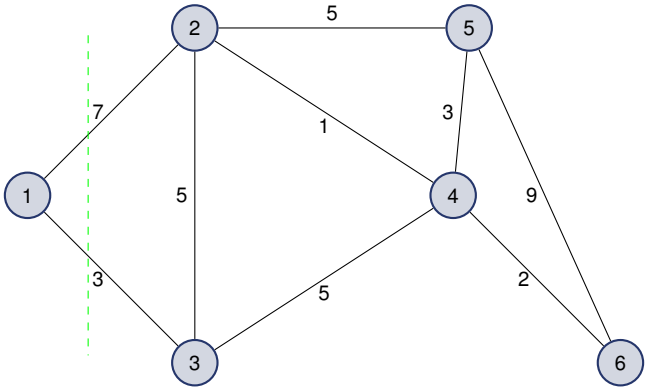
Man könnte den Knoten (4) auch "verschieben". Das Ergebnis wäre wieder eine einzige Gerade.

Min Cut in Netzwerken: Beispiel (I)



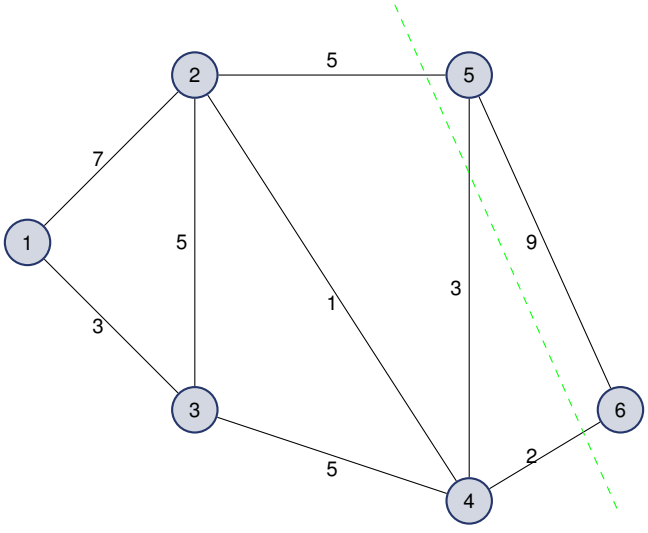
Frage: Was ist der Min-Cut in diesem Graphen und wo liegt er?

Min Cut in Netzwerken: Beispiel (I)



Kapazität von 10.

Min Cut in Netzwerken: Beispiel (I)



Minimale Kosten in Netzwerken: Motivation

Häufig wollen wir nicht wissen was der maximal mögliche Fluss in einem Netzwerk ist, sondern wie wir einen bestimmten Fluss kosteneffizient durch das Netzwerk fließen lassen können. Beispiele sind:

- Wie kann ich als Logistikunternehmen kostenoptimal meine Ware verteilen?
- Wie können wir eine kostenoptimale Stromversorgung sicherstellen?
- Wie kann man bei Großveranstaltungen Menschenflüsse optimal steuern?

Dabei geht es nicht um die Frage der reinen Machbarkeit sondern um die Optimierung bei gegebenen Rahmenbedingungen. Die folgenden Voraussetzungen müssen erfüllt sein:

- Es existiert eine Nachfrage nach dem betrachteten Gut in mindestens einem Knoten (Senke).
- Es existiert die Möglichkeit, das betrachtete Gut in mindestens einem Knoten zu produzieren (Quelle).
- Es existiert eine Kostenfunktion $c(e)$ für alle Kanten.
- Optional: Es existiert eine Kapazitätsfunktion $u(e)$ für alle Kanten.

第一张图片强调了在网络中寻找最小成本流的重要性。举例来说，在物流、电力供应或人群控制等不同的场景中，我们希望以最低的成本来满足特定的需求。在这些情况下，我们需要考虑以下几点：

- 网络中的需求点（比如消费者或事件地点）。
- 供应点（比如工厂或电站）。
- 每条边的成本函数 $c(e)$。
- 可选的是每条边的容量函数 $u(e)$。

Minimale Kosten in Netzwerken

- Sei (V,E,s,t,u,c) ein Netzwerk. Für jeden Knoten $i \in V$ benötigen wir eine Nachfrage $b(i)$
- Ein Knoten i ist ein...
  - ... Angebotsknoten, wenn $b(i) > 0$
  - ... Transportknoten, wenn $b(i) = 0$
  - ... Nachfrageknoten, wenn $b(i) < 0$
- Die Bruttoflussfunktion g ist zulässig, wenn die folgende Nebenbedingung erfüllt wird:

$$\sum_{j:(i,j)\in E} g(i,j) - \sum_{j:(j,i)\in E} g(j,i) = b(i)$$

- Zusätzlich definieren wir die Kosten des Flusses als:

$$c(g) = \sum_{(i,j)\in E} c(i,j)g(i,j)$$

第二张图片介绍了用于计算最小成本流的网络的数学模型。在这个模型中：

- $b(i)$ 表示每个节点 i 的供应或需求量（正数为供应，负数为需求）。
- 节点可以是供应点、需求点或中转点（分别对应 $b(i) > 0, b(i) < 0$ 或 $b(i) = 0$）。
- 流 $g(i,j)$ 是合法的，如果它满足所有节点的供应和需求平衡条件。
- 流的成本 $c(g)$ 是通过计算网络中所有边的流量与其成本函数的乘积之和来定义的。

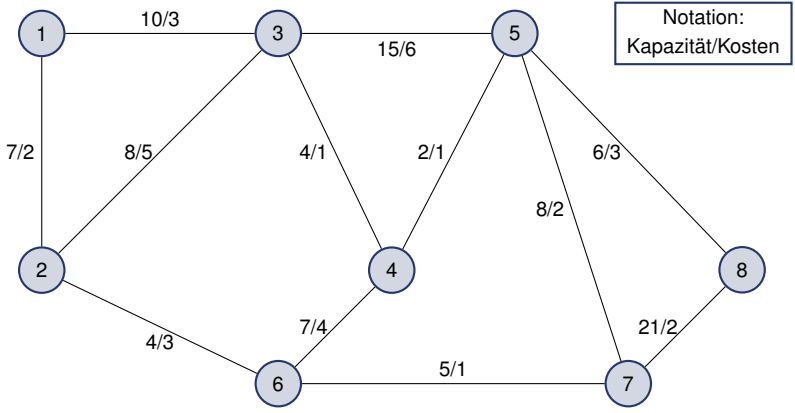
- Das minimale Kosten Problem in Netzwerken besteht daraus, den zulässigen Fluss mit den minimalen Gesamtkosten zu bestimmen.

$$
\begin{aligned}
\min z = \quad & \sum_{i,j \in V:(i,j) \in E} g(i,j) \cdot c(i,j) & & & \\
\text{s.t.} \quad & \sum_{j} g(i,j) & - \sum_{j} g(j,i) &= b(i) & \forall i \in V \\
& & g(i,j) &\leq u(i,j) & \forall (i,j) \\
& & g(i,j) &\geq 0 & \forall (i,j)
\end{aligned}
$$

Das minimale Kosten Problem beinhaltet die folgenden Elemente:

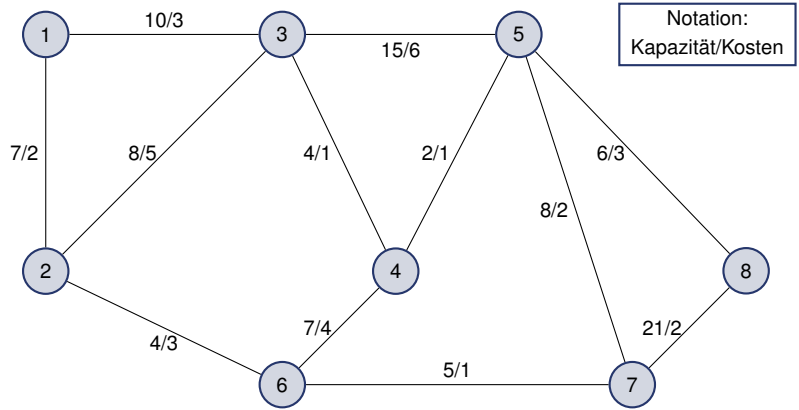
- Eine zu minimierende Kostenzielfunktion, bei der der Fluss auf einer Kante mit den entsprechenden Kosten multipliziert wird.
- Die Massenbilanz für jeden Knoten, bei der das $b(i)$ definiert ob es sich um einen Angebots-, Transport- oder Nachfrageknoten handelt.
- Eine Kapazitätsnebenbedingung für alle Kanten. Wenn die Kapazität der Kanten unendlich ist (aka: keine Kapazitätsbeschränkung) wird dennoch diese Nebenbedingung benötigt.
- Die Definition des Wertebereichs für g.

Min Cost in Netzwerken: Beispiel (I)



Wir suchen den kostenminimalen Fluss von 13 Einheiten von Knoten 1 zu Knoten 8.

Min Cost in Netzwerken: Beispiel (I)



Es wird nicht mehr sinnvoll sein, jede Gleichung einzeln aufzustellen. Dies würde in 1 (Zielfunktion) + 8 (Knoten-Massenbilanz) + 12 (Kanten-Kapazitäten) = 21 Gleichungen resultieren. Daher werden wir die kompakte Schreibweise verwenden.

Min Cost in Netzwerken: Beispiel (I)

Wir benötigen drei Parameter für die Darstellung unseres Problems:

- Kostenvektor $c(i,j)$

$$c(i,j) = \begin{pmatrix} 0 & 2 & 3 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 5 & 0 & 0 & 3 & 0 & 0 \\ 3 & 5 & 0 & 1 & 6 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 4 & 0 & 0 \\ 0 & 0 & 6 & 1 & 0 & 0 & 2 & 3 \\ 0 & 3 & 0 & 4 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 3 & 0 & 2 & 0 \end{pmatrix}$$

- Kapazitätsvektor $u(i,j)$

$$u(i,j) = \begin{pmatrix} 0 & 7 & 10 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 8 & 0 & 0 & 4 & 0 & 0 \\ 10 & 8 & 0 & 4 & 15 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 2 & 7 & 0 & 0 \\ 0 & 0 & 15 & 2 & 0 & 0 & 8 & 6 \\ 0 & 4 & 0 & 7 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 8 & 5 & 0 & 21 \\ 0 & 0 & 0 & 0 & 6 & 0 & 21 & 0 \end{pmatrix}$$

- Nachfrage $b(i)$

$$b(i)^T = \begin{pmatrix} 13 & 0 & 0 & 0 & 0 & 0 & 0 & -13 \end{pmatrix}$$

Min Cost in Netzwerken: Beispiel (I)

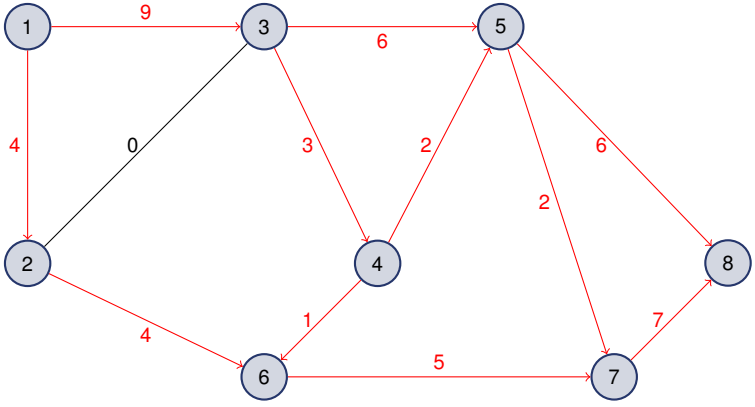
Die auf der vorherigen Folie definierten Parameter können nun genutzt werden, um die allgemeine Form des Minimalen Kosten Problems in Netzwerken aufzustellen.

$$
\begin{aligned}
\min z = & \sum_{i,j \in V:(i,j) \in E} g(i,j) \cdot c(i,j) & & & & \\
\text{s.t.} & \sum_{j} g(i,j) & - \sum_{j} g(j,i) & = & b(i) & \quad \forall i \in V \\
& & g(i,j) & \leq & u(i,j) & \quad \forall (i,j) \\
& & g(i,j) & \geq & 0 & \quad \forall (i,j)
\end{aligned}
$$

Die Nutzung der allgemeinen Form hat einige Vorteile

- Kompaktere Schreibweise: Es lässt sich einfacher verstehen, was im Modell passiert.
- Getrennte Pflege von Gleichungen und Daten, vor allem bei großen Modellen relevant.
- Weniger Fehleranfällig, da sichergestellt ist, dass alle Gleichungen identisch aufgebaut sind.
- Skalierbarkeit: Zusätzliche Knoten oder Kanten müssen lediglich in den Daten hinzugefügt werden, die Gleichungen werden automatisch aufgestellt.

Min Cost in Netzwerken: Beispiel (I)



Gesamtkosten = 133