



Lineare und kombinatorische Optimierung

Übungsblatt 6

Gruppenübungen

Aufgabe 1. (Netzwerkeigenschaften)

(0 Punkte)

Sei $N = (D = (V, A), c, s, t)$ ein Netzwerk, also ein bewerteter Digraph D mit einer Quelle s und einer Senke t .

Welche der folgenden Aussagen sind wahr? Geben Sie entweder einen Beweis oder ein Gegenbeispiel an.

1. Wenn $x : A \rightarrow \mathbb{R}$ ein maximaler (s, t) -Fluss für N ist, dann gilt entweder $x_{uv} = 0$ oder $x_{uv} = c_{uv}$ für jeden Bogen $(u, v) \in A$.
2. N besitzt einen maximalen (s, t) -Fluss, für den entweder $x_{uv} = 0$ oder $x_{uv} = c_{uv}$ für jeden Bogen $(u, v) \in A$ gilt.
3. Wenn alle Kapazitäten verschieden sind, dann ist der minimale (s, t) -Schnitt eindeutig.
4. Wenn jede Kapazität mit einer positiven Zahl $\lambda \in \mathbb{R}$ multipliziert wird, dann bleibt jeder minimale (s, t) -Schnitt ein minimaler (s, t) -Schnitt des geänderten Netzwerks.
5. Wenn zu jeder Kapazität eine positive Zahl $\lambda \in \mathbb{R}$ addiert wird, dann bleibt jeder minimale (s, t) -Schnitt ein minimaler (s, t) -Schnitt des geänderten Netzwerks.

Lösung:

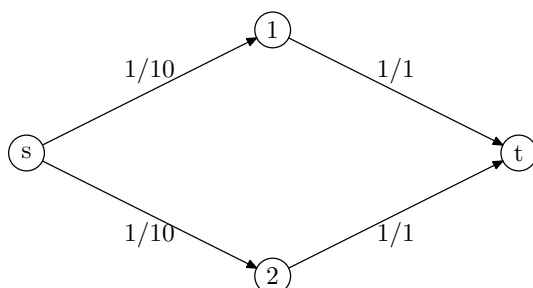


Abbildung 1: Graph 1

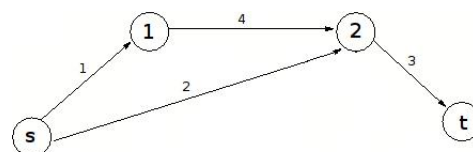


Abbildung 2: Graph 2

1. Diese Aussage ist falsch. Abbildung 1 zeigt ein Gegenbeispiel.

2. Diese Aussage ist auch falsch, was wir wieder an Graph 1 zeigen können. Ein maximaler Fluss hat Wert 2. Um diesen zu erhalten können wir die Werte an den aus s gehenden Kanten aber weder 0 noch 10 wählen.
3. Abbildung 2 zeigt ein Gegenbeispiel für diese Aussage. Sowohl der durch $S = \{s\}$ induzierte Schnitt wie auch der induzierte Schnitt durch $W = \{s, 1, 2\}$ haben Wert 3 und sind minimale Schnitte. Der minimale Schnitt ist also auch dann nicht eindeutig, wenn alle Kapazitäten verschieden sind.
4. Diese Aussage ist wahr: Der Wert eines durch die Knotenmenge W induzierten Schnittes ist gleich $c(\delta^+(W))$. Sei $\delta^+(S)$ ein minimaler Schnitt, also

$$c(\delta^+(S)) \leq c(\delta^+(W)) \text{ für alle Schnitte } \delta^+(W).$$

Dann gilt für das geänderte Netzwerk $(D = (V, A), \hat{c})$ mit $\lambda \in \mathbb{R}^+$ aber auch

$$\hat{c}(\delta^+(S)) = \lambda \cdot c(\delta^+(S)) \leq \lambda \cdot c(\delta^+(W)) = \hat{c}(\delta^+(W)) \text{ für alle Schnitte } \delta^+(W).$$

D.h. $\delta^+(S)$ ist auch ein minimaler Schnitt des geänderten Netzwerks.

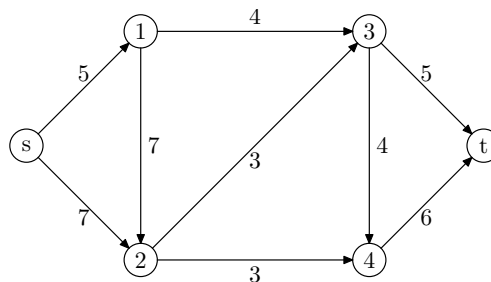
5. Diese Aussage ist wiederum falsch, wie man an Abbildung 2 erkennen kann. Wir haben gesehen, dass die durch $S = \{s\}$ und $W = \{s, 1, 2\}$ induzierten Schnitte minimale Schnitte mit Wert 3 sind. Addiert man nun z.B. $\lambda = 2$ zu allen Kapazitäten, dann hat $\delta^+(W)$ im geänderten Netzwerk den Wert 5, aber $\delta^+(S)$ hat den Wert 7, kann also kein minimaler Schnitt mehr sein.

Aufgabe 2. (Augmentierende Wege)

(0 Punkte)

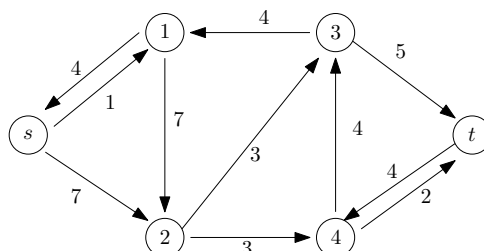
Wenden Sie den Augmentierende-Wege-Algorithmus von Ford–Fulkerson (Algorithmus 16) auf den unten abgebildeten Graphen an. Gehen Sie dabei wie folgt vor:

1. Starten Sie mit dem Nullfluss und augmentieren Sie den Fluss entlang des Weges $(s, 1, 3, 4, t)$.
2. Zeichnen Sie den Residualgraphen $G(x)$.
3. Augmentieren Sie den Fluss entlang passender augmentierender Wege. Geben Sie in jedem Schritt den augmentierenden Weg und die aktuellen Flusswerte x_{ij} an.
4. Geben Sie am Ende den maximalen Fluss und dessen Wert an.
5. Wie sieht der minimale (s, t) -Schnitt aus?



Lösung:

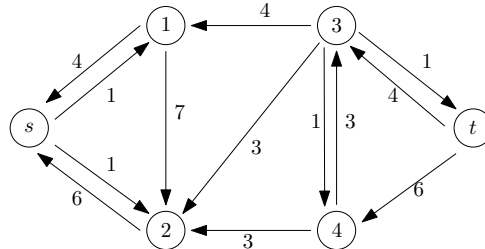
1. Entlang des Weges $s - 1 - 3 - 4 - t$ können wir um $\varepsilon = 4$ augmentieren. Wir erhalten den Fluss $x_{s1} = x_{13} = x_{34} = x_{4t} = 4$, $x_{s2} = x_{12} = x_{23} = x_{24} = x_{3t} = 0$.
2. Residualgraph $G(x)$ nach der ersten Augmentierung:



3. Weitere augmentierende Wege:

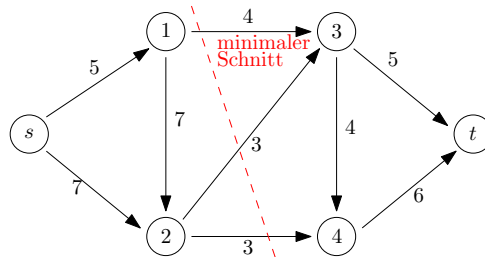
- (a) $s - 2 - 4 - t$ mit $\varepsilon = 2$ und Fluss $x_{s2} = x_{24} = 2$, $x_{s1} = x_{13} = x_{34} = 4$, $x_{4t} = 6$, $x_{12} = x_{23} = x_{3t} = 0$.
- (b) $s - 2 - 3 - t$ mit $\varepsilon = 3$ und Fluss $x_{s1} = x_{13} = x_{34} = 4$, $x_{s2} = 5$, $x_{24} = 2$, $x_{4t} = 6$, $x_{23} = x_{3t} = 3$, $x_{12} = 0$.
- (c) $s - 2 - 4 - 3 - t$ mit $\varepsilon = 1$ und Fluss $x_{s1} = x_{13} = x_{3t} = 4$, $x_{23} = x_{24} = x_{34} = 3$, $x_{s2} = x_{4t} = 6$, $x_{12} = 0$.

Im finalen Residualgraphen $G(x)$



gibt es nun keine weiteren augmentierenden Wege mehr und damit ist der Fluss maximal.

- 4. Der maximale Fluss ist $x_{s1} = x_{13} = x_{3t} = 4$, $x_{23} = x_{24} = x_{34} = 3$, $x_{s2} = x_{4t} = 6$, $x_{12} = 0$ mit $\text{VAL}(x) = 10$.
- 5. Der minimaler Schnitt hat ebenfalls den Wert 10 und trennt den Graphen in die Knotenmengen $\{s, 1, 2\}$ und $\{3, 4, t\}$.



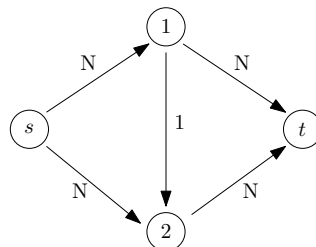
Aufgabe 3. (Laufzeit)

(0 Punkte)

Geben Sie ein Netzwerk $(D = (V, A), c, s, t)$ an, für das der Augmentierende-Wege-Algorithmus von Ford-Fulkerson im Worst-Case Fall eine exponentielle Laufzeit (in der Eingabegröße) benötigt. (Mit Begründung!)

Lösung:

Wir betrachten das folgende Netzwerk (D, c, s, t) , wobei $N \in \mathbb{Z}_+$:



Wenn wir in jeder Iteration einen augmentierenden $s-t$ -Weg der Länge 3 wählen, wird der Fluss in jeder Iteration nur um 1 zunehmen. Also brauchen wir $2N$ Iterationen, um einen maximalen Fluss zu bestimmen. Die Schritte 2)-5) werden also $O(N)$ -mal durchlaufen, die gesamte Laufzeit ist dann $O(mN)$, also nicht polynomial (dafür ist höchstens $O(m \log N)$ erlaubt).

Hinweis: Die Eingabegröße eines Graphens (Kodierungslänge) setzt sich zusammen aus $|V|$, $|A|$ und $\langle c_a \rangle \in O(\log c_a) \forall a \in A$. (vergleiche Kapitel 2 im Skript). Außerdem gilt:

$$N = 2^{\log N} \rightarrow O(N) = O(2^{\log N})$$

Damit ist die Laufzeit in der Eingabegröße (Kodierungslänge) des Graphens exponentiell.

Aufgabe 4. (Matchings und Maximale Flüsse)

(0 Punkte)

1. Ein *Matching* ist eine Menge von Kanten, so dass keine zwei Kanten einen gemeinsamen Endknoten haben.

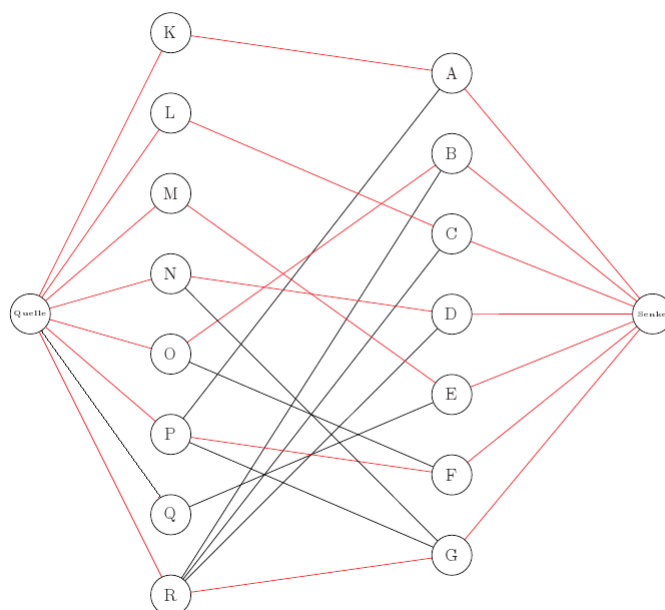
Gegeben sei ein bipartiter Graph $G = (V, E)$. Gesucht ist ein maximales Matching in G , d. h. ein Matching mit maximal vielen Kanten. Wie kann man dieses Problem als Maximal-Fluss-Problem modellieren?

2. Ein Personalchef hat mehrere Stellen zu besetzen: (K, L, M, N, O, P, Q, R). Er hat mehrere Bewerbungen vorliegen, wobei die Bewerber für mehrere Jobs geeignet sind (in Klammern für jeden angegeben): Alfred (K, P), Bruno (O, R), Cornelia (L, R), Dorit (N, R), Emil (Q, M), Frieda (O, P), Gundula (P, N, R). Finden Sie auf graphentheoretischem Wege eine Zuordnung, so dass möglichst viele der vakanten Stellen mit einem geeigneten Bewerber besetzt werden können. (Ein Algorithmus muss hier nicht im Detail durchgeführt werden.)

Lösung:

1. Das Matchingproblem wird auf das Flussproblem reduziert, indem man zwei zusätzliche Knoten, die Quelle s und die Senke t hinzufügt sowie Kanten zwischen der Quelle und allen Knoten aus der ersten Menge und zwischen denen aus der zweiten Menge und der Senke. Alle Kanten haben Kapazität Eins. Ein möglicher maximaler ganzzahliger Fluss entspricht einem maximalen Matching in diesem Graphen.
2. Das Problem wird als Graph aufgefasst, indem man die Berufe und die Personen als Knoten betrachtet und die Beziehung der Eignung als Kante zwischen der Person und den Berufen für die sie geeignet ist. Der entstandene Graph ist ein bipartiter Graph und die Aufgabe besteht nun darin, ein maximales Matching zu finden. Dieses können wir wie in Aufgabenteil a) angeben, mithilfe eines Max-Flow-Algorithmus' lösen.

Ein maximales Matching für das dargestellte Problem ist mit Rot eingetragen:

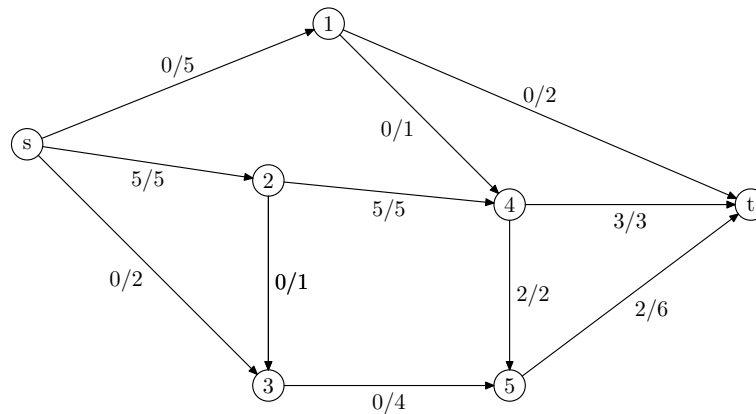


Hausübungen

Aufgabe 5. (Flüsse)

(2+1 Punkte)

Betrachten Sie folgendes Flussnetzwerk.

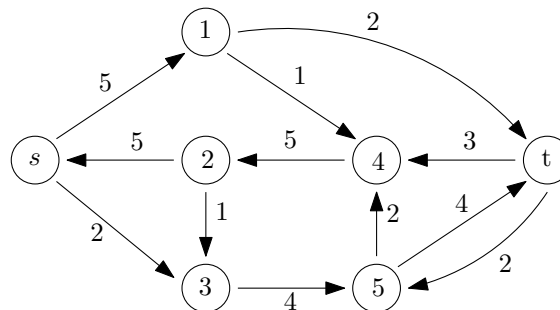


Die angegebenen Zahlen geben den aktuellen Fluss bzw. die Kapazität der jeweiligen Kante an.

1. Geben Sie für das Netzwerk den aktuellen Flusswert an. Berechnen Sie einen maximalen Fluss für das Netzwerk. Beginnen Sie mit dem aktuellen Fluss gemäß der Zeichnung. Geben Sie in jedem Schritt einen augmentierenden Weg und den Flusswert des aktualisierten Flusses an.
2. Geben Sie einen minimalen Schnitt für das Netzwerk an.

Lösung:

1. Der aktuelle Fluss hat den Wert 5.
Wir zeichnen den Residualgraphen:



Es gibt drei augmentierende Wege:

- $s - 3 - 5 - t$ Augmentiere um $\varepsilon = 2$
- $s - 1 - t$ Augmentiere um $\varepsilon = 2$
- $s - 1 - 4 - 2 - 3 - 5 - t$ Augmentiere um $\varepsilon = 1$

Der maximale Fluss hat den Wert $5 + 2 + 2 + 1 = 10$.

2. Der minimale Schnitt hat ebenfalls den Wert 10 und trennt den Graphen in die Knotenmengen $\{s, 1\}$ und $\{2, 3, 4, 5, t\}$.

Aufgabe 6. (Flusszerlegung)

(4 Punkte)

Beweisen Sie Lemma 6.3 der Vorlesung.

Lösung:

Wenn eine Zerlegung wie oben vorliegt, sind die Flusserhaltungsbedingungen erfüllt (da alle Wege und Kreise, die in einen Knoten ($\neq s, t$) hineinführen auch wieder hinausführen); da aber auch die Kapazitätsbedingungen per Voraussetzung erfüllt sind, folgt aus der Existenz einer Zerlegung bereits, dass x ein zulässiger (s, t) -Fluss ist.

Betrachten wir also die umgekehrte Richtung: Nehmen wir an, wir haben mit x einen zulässigen (s, t) -Fluss. Wir konstruieren nun eine Zerlegung wie oben; hierzu betrachten wir die Bögen a , die von einem v_0 (z.B. s) ausgehen und auf denen $x_a > 0$ ist. Von diesen Bögen wählen wir denjenigen mit maximalem Fluss aus, nennen ihn a_1 und kommen so zu einem neuen Knoten v_1 . Nun betrachten wir alle Bögen a , die von v_1 ausgehen und auf denen $x_a > 0$, wählen den mit maximalem Fluss, nennen ihn a_2 und gehen zu einem Knoten v_2 usw. Dies setzen wir fort, bis nach k Schritten entweder $v_k = t$ oder $v_k = v_i$ für ein $i < k$ gilt; einer dieser Fälle muss auftreten, da V endlich ist. Im ersten Fall haben wir einen Weg von s nach t gefunden. Betrachten wir nun das Minimum über alle a_i , so haben wir einen Weg P mit zugehörigem $\lambda = \min x_{a_i}$ gefunden. Analog haben wir im zweiten Fall einen Kreis C mit zugehörigem $\mu = \min x_{a_i}$ gefunden. Wenn wir nun den Fluss x auf allen Kanten a_i um λ bzw. μ reduzieren so können wir auf dem so reduzierten Graphen unsere Zerlegung fortsetzen, bis wir unseren Fluss x auf den Nullfluss reduziert haben. Dies zeigt auch die obere Schranke $p + q \leq |A|$.

Aufgabe 7. (Gerade und ungerade Flüsse)

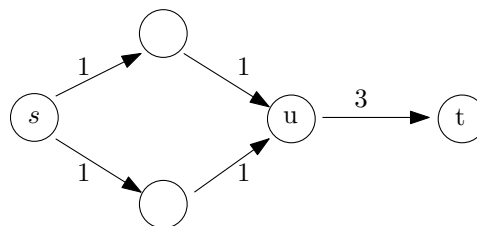
(2+1 Punkte)

Es sei $D = (V, A)$ ein Digraph mit Quelle s und Senke t und ganzzahligen Kapazitäten $c_a \geq 0$ für alle $a \in A$. Beweisen oder widerlegen Sie die folgenden Behauptungen:

1. Sind alle Kapazitäten gerade Zahlen, so existiert ein maximaler (s, t) -Fluss x , so dass x_a für alle Bögen $a \in A$ gerade ist.
2. Sind alle Kapazitäten ungerade Zahlen, so existiert ein maximaler (s, t) -Fluss x , so dass x_a für alle Bögen $a \in A$ ungerade ist.

Lösung:

1. Die Behauptung ist wahr. Der Nullfluss $x = 0$ ist gerade. Sind alle Kapazitäten des Flussnetzes (Netzwerks) ganzzahlig und gerade, dann wird in jedem Schritt, in dem es noch einen augmentierenden (s, t) -Weg P im Residualgraphen $G(x)$ gibt, der Fluss um einen geraden Wert $\varepsilon \geq 2$ entlang P augmentiert. Dabei bleiben die Flusswerte auf allen Bögen $a \in A$ gerade. Damit ist, wenn es keinen augmentierenden Weg mehr gibt, der maximale (s, t) -Fluss ebenfalls gerade.
2. Die Behauptung ist falsch. Im abgebildeten Flussnetz hat der maximale Fluss auf Bogen (u, t) den Wert $x_{ut} = 2$.



Aufgabe 8. (Experimente im All)

(3 Punkte)

Ein Professor soll mit einer Weltraumfähre ins All fliegen und dort einige Experimente durchführen. Es soll aus einer Menge von n Experimenten E_1, \dots, E_n gewählt werden. Das Durchführen eines Experiments E_i würde einen Ertrag von e_i Euro bringen. Wir haben eine Menge $W = \{W_1, \dots, W_m\}$ von Werkzeugen zur Verfügung und jedes Experiment E_i benötigt eine bestimmte Teilmenge $K_i \subseteq W$ von diesen Werkzeugen zur Durchführung. Ein Werkzeug darf gleichzeitig von mehreren Experimenten verwendet werden. Das Mitnehmen von einem Werkzeug W_i verursacht Kosten von w_i Euro. Geben Sie einen Algorithmus an, um zu bestimmen, welche Werkzeuge mitgenommen und welche Experimente

durchgeführt werden sollen, damit der Gewinn – also die Gesamteinnahmen minus den Gesamtkosten – maximal wird. Begründen Sie die Korrektheit dieses Algorithmus.

Lösung:

Sei $E = \{E_1, \dots, E_n\}$ die Menge der Experimente. Wir konstruieren einen gerichteten Graphen $D = (V, A)$ mit Kapazitäten c wie folgt:

$$\begin{aligned} V &= E \cup W \cup \{s, t\} \\ A &= \{(s, E_i) : 1 \leq i \leq n\} \cup \{(W_i, t) : 1 \leq i \leq m\} \cup \{(E_i, W_j) : W_j \in K_i\} \\ c_{uv} &= \begin{cases} e_i & \text{falls } u = s \text{ und } v = E_i \\ w_i & \text{falls } u = W_i \text{ und } v = t \\ \infty & \text{sonst} \end{cases} \end{aligned}$$

Sei nun U eine Knotenmenge, so dass $\delta^+(U)$ ein minimaler (s, t) -Schnitt in D ist. Man sieht leicht ein, dass die Kapazität von diesem Schnitt endlich sein muss und deshalb keine Kante der Form (E_i, W_j) enthalten kann. Insbesondere folgt hieraus, dass aus $E_i \in U$ folgt $K_i \subset U$, d.h. wenn ein Experiment in U aufgenommen wird, werden auch alle benötigten Werkzeuge aufgenommen. Wir behaupten, der Professor sollte genau die Experimente in U durchführen und die Werkzeuge in U mit sich führen. Man kann sich nämlich überlegen, dass die Kapazität eines solchen Schnittes gleich der Summe der Werkzeugkosten plus den verlorenen Einnahmen aus den Experimenten ist, die nicht durchgeführt werden. Wenn wir also den Schnitt minimieren, minimieren wir diese Kosten und Verluste und maximieren dadurch den Gewinn.

Aufgabe 9. (Programmieraufgabe: Augmentierende-Wege-Algorithmus)

(2+3 Punkte)

Laden Sie die Datei `ProgAufgabe06.py` aus Studon herunter und fügen Sie sie in ihr PyCharm ein.

In dieser Aufgabe werden Sie den Augmentierende-Wege-Algorithmus zur Bestimmung eines maximalen (s, t) -Flusses implementieren.

1. Implementieren Sie zunächst die Methode `residual_graph(G, x)`. Die Parameter der Methode sind ein gerichteter Graph G mit Kantenkapazitäten *capacity* und ein Dictionary x , das den aktuellen Fluss auf jeder Kante von G angibt ($x[v, w]$ ist der Fluss von v nach w).

Die Methode soll den aus dem Skript bekannten Residualgraphen $G(x)$ zurückgeben. Dabei ist folgendes zu beachten:

- Der Graph G darf dabei nicht verändert werden, das heißt der Rückgabewert ist ein neuer `networkx`-Digraph.
- Der Rückgabegraph muss zwei Kantenattribute *residual* und *forward* haben: *residual* ist die Bogenkapazität der Kante. *forward* ist ein boolescher Wert, der angibt, ob die Kante eine Vorwärtskante ist.

2. Implementieren Sie nun die Methode `max_flow(G, s, t)`, die unter Verwendung von `residual_graph` den Augmentierende-Wege-Algorithmus auf G anwendet, um einen maximalen (s, t) -Fluss zu berechnen. Zur Überprüfung der Existenz und zum Finden eines Weges von s nach t im Residualgraphen $Gx = \text{residual_graph}(G, x)$ dürfen Sie die `networkx`-Methoden `nx.hasPath(Gx, s, t)` und `nx.shortestPath(Gx, s, t)` verwenden.

- Der Graph G darf nicht verändert werden.
- Der Rückgabewert ist ein Tupel $x, \text{flow_value}$. Hierbei ist x ein Dictionary, das zu jeder Kante in G angibt, wie viel dort fließt (ungenutzte Kanten müssen im Dictionary mit Wert 0 auftreten!). *flow_value* ist der Wert dieses Flusses, sprich wie viel von s nach t fließt.

Nutzen Sie den beigefügten Testcode, um ihre Implementierung zu testen. Überprüfen Sie beispielsweise, ob der von ihrem Code ausgegebene Fluss tatsächlich zulässig ist (also die Kapazitätsschranken einhält).

Lösung:

Die Musterlösung der Programmieraufgabe finden Sie im StudOn in der Datei `ProgAufgabe06_Loesung.py`.

Gruppe 1: Abgabe der Hausaufgaben am 05.12.2017 in der Übung.
Gruppe 2+3: Abgabe der Hausaufgaben am 06.12.2017 in der jeweiligen Übung.