Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
Sebastian Lamm, Demian Hespe

3. Übungsblatt zu Algorithmen II im WS 2018/2019

http://algo2.iti.kit.edu/AlgorithmenII_WS18.php
{sanders, lamm, hespe}@kit.edu

Musterlösungen

Aufgabe 1 (*Kleinaufgaben: Eigenschaften von Flüssen*)

a) Nach Vorlesung ist eine gültige Distanzfunktion $d(\cdot)$ für *Dinitz Algorithmus* gegeben durch:

- $d(t) = 0$
- $d(u) \leq d(v) + 1 \quad \forall\ (u, v) \in G_f$

Zeigen Sie, falls $d(s) \geq n$, existiert kein *augmentierender Pfad*.

b) In der Vorlesung wurde gezeigt, dass die Laufzeit von *Dinitz Algorithmus* für Graphen mit Kantengewichten gleich 1 (*unit edgeweights*) in $O((n+m)\sqrt{m})$ liegt. Vergleichen Sie diese Laufzeit zum *Ford Fulkerson Algorithmus*. Für welche Graphen mit *unit edgeweights* ist welcher der beiden Algorithmen schneller?

c) Sei $G = (V, E)$ ein gerichteter Graph, in dem maximale Flüsse berechnet werden sollen. Sei $e = (i, j) \in E$ ebenso wie $e' = (j, i) \in E$, d. h. G besitzt ein Paar entgegengesetzter Kanten. Außerdem sei $c(e) \geq c(e')$. Widerlegen Sie durch ein Gegenbeispiel:

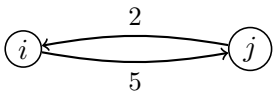
Entfernt man e' aus E und reduziert $c(e) := c(e) - c(e')$, ändert sich der maximale Fluss nicht, d. h. man kann entgegengesetzte Kanten a-priori (für beliebige s und t) gegeneinander aufrechnen.

Musterlösung:

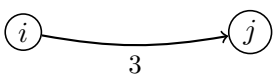
a) Ein Pfad in einem Graphen kann höchstens n unterschiedliche Knoten haben. Betrachte nun einen beliebigen augmentierenden Pfad in G_f. Für jede Kante auf diesem Weg wächst die Distanz für s höchstens um eins. Folglich kann ein augmentierender Pfad nur $d(s) \leq n - 1$ bedeuten.

b) Auf Graphen mit Kantengewichten gleich 1 ist die Laufzeit des *Ford Fulkerson Algorithmus* in $O(nm)$ (da $U = 1$!). *Dinics Algorithmus* ist damit schneller als der *Ford Fulkerson Algorithmus* falls $O((n+m)\sqrt{m}) < O(nm)$. Ausgerechnet ergibt sich $n > O(\frac{m}{\sqrt{m}-1}) = O(\sqrt{m})$.

c) Rechnet man die Kanten in folgendem Graph gegeneinander auf,



so ergibt sich



Für $s := j$ und $t := i$ ist kein Fluss mehr möglich, während im Originalgraphen ein maximaler Fluss von 2 möglich war. Dies ist somit ein Gegenbeispiel zur Behauptung.

Aufgabe 2 (*Rechnen: Segmentierung mit Flüßen*)

Wir betrachten einen einfachen Fall für Bildbearbeitung. Die Vorder-/Hintergrundsegmentierung. Das Ziel dieses Prozesses ist es, ein Bild in Vorder und Hintergrund zu zerlegen. Die Transformation dafür weißt jedem Pixel $p_{i,j}$ des Bildes einen Knoten $v_{i,j}$ im Graphen zu. Für jedes Paar von benachbarten Pixeln $p_{i,j}$ und $p_{k,l}$ ($|i-k|+|j-l|=1$) fügen wir eine ungerichtete Kante $(v_{i,j}, v_{k,l})$ ein. Zusätzlich fügen wir je einen Knoten s für Vordergrund (Quelle) und einen Knoten t für Hintergrund (Senke) ein. Von Knoten s existiert eine gerichtete Kante zu jedem Knoten $p_{i,j}$ und von jedem Knoten $p_{i,j}$ existiert eine gerichtete Kante zu Knoten t. Wir definieren darüber hinaus folgende Kantengewichte:

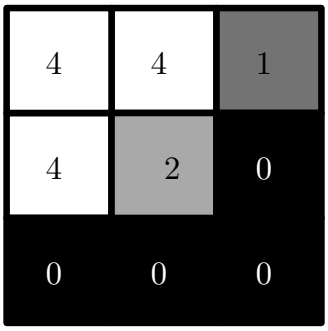
$$c(e=(u,v)) = \begin{cases} p_v(v) & u = s \\ p_h(u) & v = t \\ f(u,v) & \textit{sonst} \end{cases}$$

Wobei mit $p_v(x)$ die Wahrscheinlichkeit gegeben ist, dass x Vordergrundknoten ist, mit $p_h(x)$ die Wahrscheinlichkeit für einen Hintergrundknoten und mit $f(x,y)$ eine Penaltyfunktion für das Trennen der beiden Knoten x und y. Für ein Graustufenbild B definieren wir
$p_v(x,y) = B[x,y]^2$, $p_h(x,y) = (4 - B[x,y])^2$ sowie $f((x_1,y_1),(x_2,y_2)) = (4 - |B[x_1,y_1] - B[x_2,y_2]|)^2$.
Hinweis: Diese Modellierung ist nur ein Beispiel und keine allgemeingültige Modellierung. Sie soll nur verdeutlichen wie Flow Alglorithmen für andere Probleme eingesetzt werden können.

a) Geben Sie den Flussgraphen für das unten angegebene Graustufenbild an.

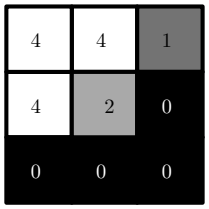
b) Führen Sie einen augmenting Path Algorithmus auf dem entstandenen Graphen aus.

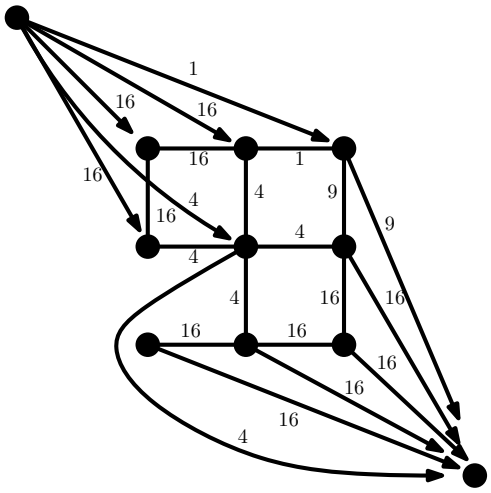
c) Wie würde die Segmentierung in Vorder- und Hintergrund im Bild als Ergebnis aussehen?



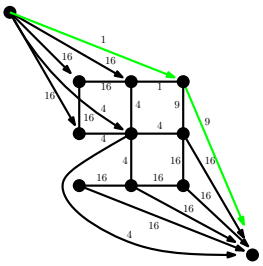
Musterlösung:

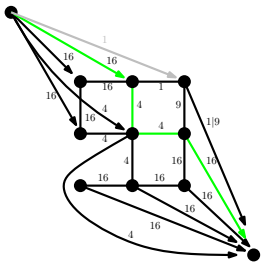
a) Als Transformation ergibt sich folgender Graph. Kanten ohne Kapazität wurden weggelassen.

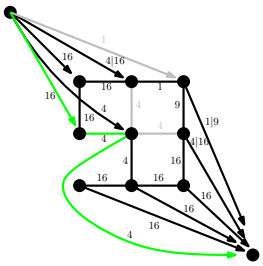


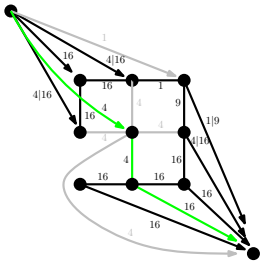


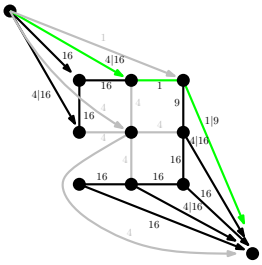
b) Der Algorithmus wird skizziert durch folgende Schritte:

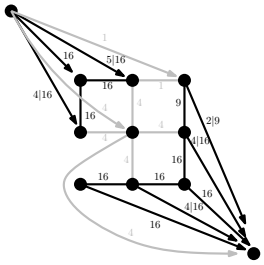


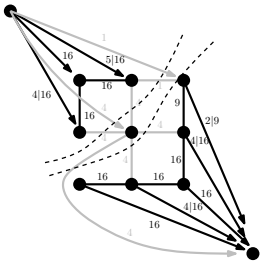




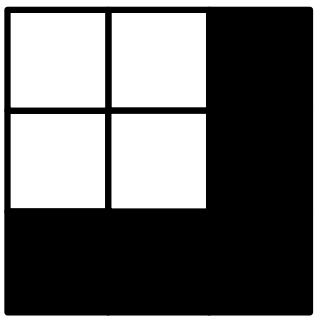


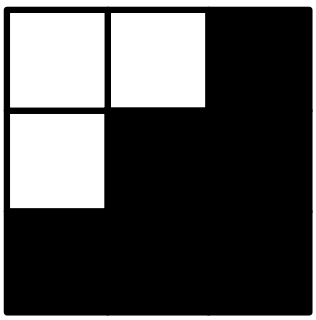






c) Die beiden gleichwertigen Lösungen nach unserer Modellierung sind:





Aufgabe 3 (*Analyse: Königs Theorem*)

Das *Theorem von König* besagt, dass in jedem bipartiten Graphen $G = (V,\ E)$ die Größe eines Matchings größter Wertigkeit (*maximum-cardinality matching*) gleich der Größe einer minimalen Knotenüberdeckung (*minimal vertex cover*) ist.

Vertex Cover:
Ein *Vertex Cover* ist definiert als eine Teilmenge der Knoten $S \subseteq V$, so dass für alle Kanten $e = (u, v) \in E$ gilt $u \in S \vee v \in S$. Ein *minimales Vertex Cover* besitzt unter allen korrekten die kleinste Teilmenge an Knoten S.

Matching:
Ein *Matching* ist definiert als eine Teilmenge von Kanten $S \subseteq E$, so dass jeder Knoten $v \in V$ Endpunkt von höchstens einer Kante in S ist. Eine *maximales Matching* besitzt unter allen korrekten die größte Teilmenge an Kanten S.

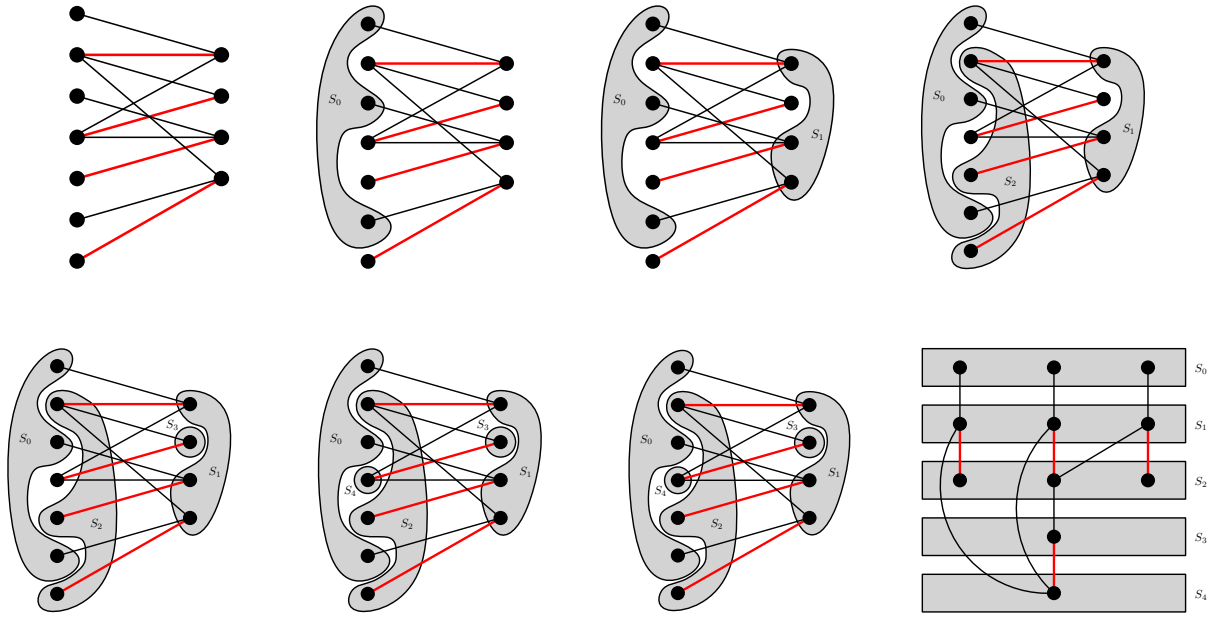
Bipartiter Graph:
Ein bipartiter Graph enthält ausschließlich Kanten zwischen disjunkten Teilmengen der Knotenmenge: $e \in E \leftrightarrow (u, v) \in S \times T,\ V = S \cup T,\ S \cap T = \varnothing$.

Beweisen Sie das *Theorem von König*.

Musterlösung:
Für einen bipartiten Graphen $G = (V, E)$ betrachten wir ein maximales Matching M der Größe $k = |M|$. Der Fall eines perfekten Matchings ist trivialerweise korrekt. Für den Fall eines nicht perfekten Matchings konstruieren wir einen Graphen mit mehreren Schichten S_i. Schicht S_0 definieren wir als alle Knoten, die zu keiner gematchten Kante inzident sind. Schicht S_i definieren wir über:

$$S_i = \begin{cases} v \in V : \exists e = (u, v) \in E \setminus M : u \in S_{i-1}, v \notin S_0, \dots S_{i-1} & \text{i ungerade} \\ v \in V : \exists e = (u, v) \in M : u \in S_{i-1}, v \notin S_0, \dots S_{i-1} & \text{i gerade} \end{cases}$$

Komponenten, die auf diese Weise nicht erreicht werden, formen in sich ein perfektes Matching und müssen für den Beweis nicht weiter betrachtet werden. Der Prozess ist hier bildlich dargestellt.



Da M ein maximales Matching ist, kann es in dem Graphen keinen alternierenden Weg geben, dessen Endpunkte ungematched sind. Daraus folgt, dass keine gematchte Kante innerhalb eines Layers S_{2i+1} existieren kann. Für eine Kante $(u, v) \in S_{2i+1}$ könnten wir sonst einen alternierenden Pfad $n_0 \in S_0, \dots, u, v, \dots n_{2(2i+1)+1} \in S_0$ finden. Dabei ist $n_0 \neq n_{2(2i+1)+1}$, da der Pfad ungerade Länge hat und das Matching wäre nicht maximal. Unter dem selben Argument kann keine ungematchte Kante zwischen zwei Knoten eines Layers S_{2i} existieren. Eine gematchte Kante innerhalb eines Layers S_{2i} kann aber ebenso nicht existieren, da jeder Knoten über eine eindeutige gematchte Kante zu einem Vorgängerlevel verbunden ist. Folglich hat jede gematchte Kante genau einen Endpunkt in einem Layer S_{2i+1}. Ebenso hat aber auch jede ungematchte Kante mindestens einen Endpunkt in einem Layer S_{2i+1}.
Folglich ist die Vereinigung über alle S_{2i+1} ein Vertex Cover da alle gematchten Kanten, aber auch ungemachten Kanten abgedeckt sind. Weiter besteht das Vertex Cover aus k Knoten, da jede der k gematchten Kanten genau einen Endpunkt im Vertex Cover besitzt und das Vertex Cover keine ungemachten Knoten beinhaltet. Ein Vertex Cover besteht aber aus mindestens k Knoten. Andernfalls gäbe es eine Kante im maximalen Matching, die vom Vertex Cover nicht überdeckt wäre, weil ein Knoten im Vertex Cover nur höchstens eine Kante aus dem Matching überdecken kann. Folglich ist die Vereinigung über alle S_{2i+1} ein Vertex Cover, das Vertex Cover besteht aus k Knoten und ist minimal da jedes Vertex Cover mindestens k Knoten beinhalten.

Aufgabe 4 (*Analyse+Entwurf+Rechnen: Grenzüberwachung*)

Eine (eindimensionale) Grenzlinie soll durch ein Sensornetz überwacht werden. Zu diesem Zweck wurde eine große Anzahl an Sensorknoten unregelmäßig an der Grenze ausgebracht. Jeder Knoten kann einen Bereich der Grenze für eine gewisse Zeit proportional zu seiner Batteriekapazität überwachen. Die Grenze gilt als vollständig gesichert, wenn jeder Abschnitt der Grenzlinie von mindestens einem Sensorknoten abgedeckt ist. Aufgrund der großen Menge an Knoten sind ihre Überwachungsbereiche stark überlappend. Daher müssen nicht immer alle Knoten aktiv sein, um eine vollständige Sicherung der Grenze zu gewährleisten. So kann Energie gespart werden und die maximale Dauer der Grenzsicherung erhöht werden.
Durch die unregelmäßige Anbringung der Knoten und durch große Fertigungstoleranzen in der Batteriekapazität und dem Überwachungsbereich (*man hat unbedingt beim billigsten Hersteller einkaufen müssen...*) ist zunächst nicht klar, wie lange die Grenze maximal vollständig gesichert werden kann. Glücklicherweise wurden die Positionen der Knoten und ihre jeweiligen Kapazitäten und Detektionsbereiche protokolliert und können verwendet werden, um diese Frage zu beantworten.

a) In der Vorlesung haben Sie Flussprobleme mit beschränkten Kantenkapazitäten $c(e)$ kennengelernt. Ebenso können Flussprobleme mit beschränkten Knotenkapazitäten $c(v)$ sinnvoll sein. In diesem Fall darf für einen gültigen Fluss die Summe der in den Knoten ankommenden bzw. ausgehenden Flüsse die Kapazität des Knotens nicht überschreiten. Außerdem muss wie bisher für jeden Knoten (außer der Quelle und Senke) die Summe der ankommenden Flüsse gleich der Summe der ausgehenden Flüsse sein.

   Erklären Sie, wie maximale Flüsse mit Knotenkapazitäten berechnet werden können. Begründen Sie kurz, warum Ihr Ansatz einen zulässigen und optimalen Fluss berechnet.

b) Konstruieren Sie ein Flussnetzwerk, das das oben beschriebene Problem der Bestimmung einer maximalen Dauer für die vollständige Grenzüberwachung lösen kann.

   Hinweis: Jeder Knoten entspricht einem Sensorknoten. Batteriekapazität kann als äquivalent zur Flussmenge betrachtet werden.

c) Erstellen Sie ein Flussnetz, das dem folgenden Sensornetz entspricht. Wie lange kann dieses Netz die Grenze im Bereich $[0, 13]$ überwachen? Welche Sensorknoten müssen wann aktiv sein?

   Format der Angaben: $x_{nodeID} = \{[begin\_range, end\_range], capacity\}$

   $x_1 = \{[0, 5], 4\}$
   $x_2 = \{[0, 7], 3\}$
   $x_3 = \{[4, 9], 2\}$
   $x_4 = \{[3, 8], 5\}$
   $x_5 = \{[8, 13], 5\}$
   $x_6 = \{[7, 11], 3\}$
   $x_7 = \{[11, 15], 2\}$

   Hinweis: Bevor Sie langwierig einen maximalen Fluss berechnen, versuchen Sie ihn durch *scharfes Hinschauen* zu bestimmen.

Musterlösung:

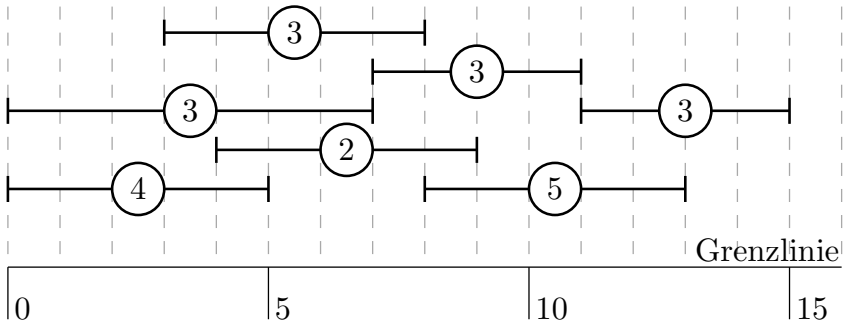
a) Man definiert einen neuen Flussgraph $G' = (V', E')$. Für jeden Knoten $v \in V$ fügt man zwei Knoten v_{in} und v_{out} sowie eine Kante (v_{in}, v_{out}) mit Kapazität $c(v_{in}, v_{out}) = c(v)$ in G' ein. Für jede Kante $(u, v) \in E$ fügt man eine neue Kante (u_{out}, v_{in}) in E' ein. Die Kapazität der Kante wird übernommen (bzw. auf ∞ gesetzt falls sie keine Kapazität hatte).

Nun berechnet man auf G' einen Fluss von s_{in} nach t_{out} und transferiert die Flusswerte zurück auf die Kanten in G. Der Fluss respektiert die Knotenkapazitäten, da sie in G' durch die Kanten (v_{in}, v_{out}) passend beschränkt wurden. Außerdem ist der Fluss optimal. Angenommen es gäbe noch einen augmentierenden Pfad in G, dann gäbe es auch einen in G' und der berechnete Fluss wäre kein maximaler Fluss in G': Die Restkapazitäten der ursprünglichen Kanten sind per Konstruktion gleich zu ihren Entsprechungen in G. Eine zu einem nicht voll ausgelasteten Knoten gehörende Kante hätte ebenfalls noch Restkapazität.

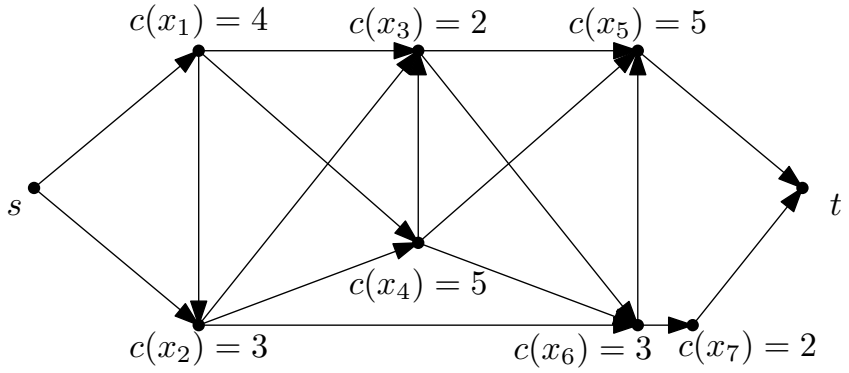
b) Das Flussnetz kann mit Hilfe von Knotenkapazitäten–wie in der letzten Teilaufgabe besprochen–konstruiert werden. Für jeden Sensorknoten x_i fügt man eine Knoten i mit Kapazität $c(i)$ gleich der Batteriekapazität des Sensorknotens ein. Außerdem fügt man eine Quelle s und eine Senke t ein. Anschließend fügt man Kanten (i, j) ein, wenn $x_i.end\_range \in [x_j.start\_range, x_j.end\_range]$ (für s und t entsprechen die *range* Werte dem Anfang und dem Ende des Grenzverlaufs). Alle Kanten sind ohne Kapazität.

Jeder Flusspfad durch das Netz entspricht einer Konfiguration von aktiven Sensorknoten, die den gesamten Grenzverlauf überwachen können und die Flussmenge der Überwachungsdauer für diese Konfiguration. Der gesamte Fluss entspricht der maximalen Überwachungsdauer.

c) Eingezeichnete Überwachungsbereiche und Batteriekapazitäten der Sensorknoten:



Sich ergebendes Flussnetzwerk:



Musterlösung:

c) Durch geschicktes Hinschauen muss man den Flussalgorithmus nicht ausführen und kann direkt eine maximale Überwachungsdauer von 7 ablesen. Diese wird durch folgende Knotenmengen erreicht, die jeweils gleichzeitig für die angegebene Dauer aktiv sind:

aktive Knoten	Dauer
x_1, x_4, x_5	4
x_2, x_4, x_5	1
x_2, x_6, x_7	2

Jede aktive Knotenmenge deckt offensichtlich den kompletten Bereich $[0, 13]$ ab. Fast alle Knoten verbrauchen ihre komplette Energie –aber auch nicht mehr– außer Knoten x_3 (gar nicht verwendet) und x_6 (noch 1 Restkapazität). Mit den restlichen Knoten kann keine weitere vollständige Überdeckung erreicht werden.

Aufgabe 5 (*Analyse: Eigenschaften von Flüssen*)

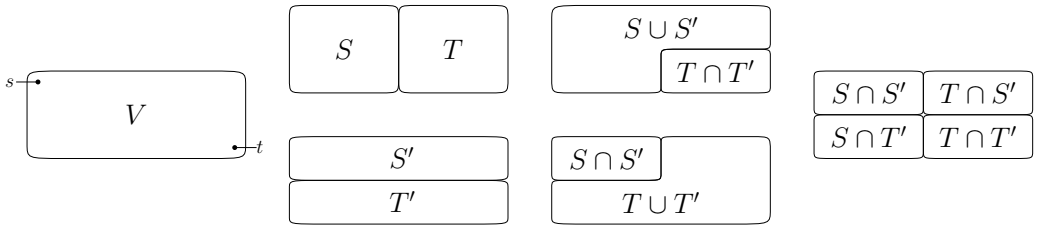
a) Seien (S, T) und (S', T') zwei minimale (s, t) Schnitte in einem Flußgraphen G. Zeigen oder widerlegen Sie, dass $(S \cup S', T \cap T')$ und $(S \cap S', T \cup T')$ auch minimale (s, t) Schnitte sind.

b) Sei (S, T) ein minimaler (s, t) Schnitt in einem Flussgraphen G. Zeigen oder widerlegen Sie, dass (S, T) ein minimaler (x, y) Schnitt ist f.a. $(x, y) \in S \times T$.

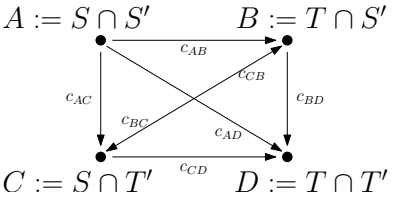
c) Zeigen Sie, dass für den *preflow-push Algorithmus* aus der Vorlesung mit beliebiger Wahl des nächsten Knotens $\mathcal{O}(n^2)$ die bestmögliche obere Schranke ist.

Musterlösung:

a) Der Graph wird in vier Bereiche aufgeteilt, $S \cap S'$, $T \cap S'$, $T \cap T'$ und $S \cap T'$ (siehe Abbildung).



Zur Anschauung definiert man einen Graphen, dessen Knoten der obigen Aufteilung entsprechen und dessen Kanten die Schnitte zwischen den Bereichen darstellen.



Damit ergeben sich folgende Werte für die Schnitte:

- (S, T) Schnitt: $c_{ST} = c_{AB} + c_{AD} + c_{CB} + c_{CD}$
- (S', T') Schnitt: $c_{S'T'} = c_{AC} + c_{AD} + c_{BC} + c_{BD}$
- $(S \cup S', T \cap T')$ Schnitt: $c_{S \cup S', T \cap T'} = c_{AD} + c_{BD} + c_{CD}$
- $(S \cap S', T \cup T')$ Schnitt: $c_{S \cap S', T \cup T'} = c_{AB} + c_{AC} + c_{AD}$

Da (S, T) und (S', T') minimale Schnitte sind, gilt $c_{ST} = c_{S'T'}$. Außerdem sind $c_{S \cup S', T \cap T'}$ und $c_{S \cap S', T \cup T'}$ jeweils größer gleich c_{ST} bzw. $c_{S'T'}$. Löst man die sich ergebenden Ungleichungen, erhält man $c_{BC} = c_{CB} = 0$ und damit $c_{AB} = c_{BD}$, $c_{AC} = c_{CD}$ (Rechnung siehe nächste Seite).

Es ergibt sich $c_{S \cup S', T \cap T'} = c_{ST} = c_{S \cup S', T \cap T'} = c_{S'T'}$.

Musterlösung:

a) (fortgesetzt)
Auflösen der Ungleichungen:

$$
\begin{aligned}
c_{S\cup S',T\cap T'} &\geq c_{ST} && (1)\\
c_{S\cup S',T\cap T'} &\geq c_{S'T'} && (2)\\
c_{S\cap S',T\cup T'} &\geq c_{ST} && (3)\\
c_{S\cap S',T\cup T'} &\geq c_{S'T'} && (4)
\end{aligned}
$$

$$
\begin{aligned}
c_{AD}+c_{BD}+c_{CD} &\geq c_{AB}+c_{AD}+c_{CB}+c_{CD} && (1)\\
c_{AD}+c_{BD}+c_{CD} &\geq c_{AC}+c_{AD}+c_{BC}+c_{BD} && (2)\\
c_{AB}+c_{AC}+c_{AD} &\geq c_{AB}+c_{AD}+c_{CB}+c_{CD} && (3)\\
c_{AB}+c_{AC}+c_{AD} &\geq c_{AC}+c_{AD}+c_{BC}+c_{BD} && (4)
\end{aligned}
$$

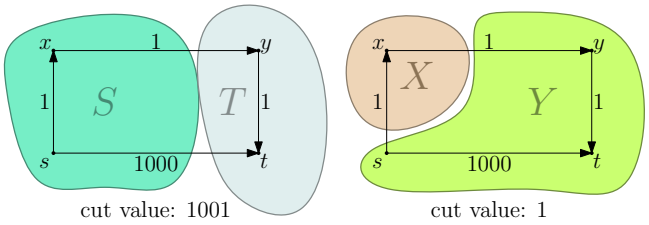
$$
\begin{aligned}
c_{BD} &\geq c_{AB}+c_{CB} && (1)\\
c_{CD} &\geq c_{AC}+c_{BC} && (2)\\
c_{AC} &\geq c_{CB}+c_{CD} && (3)\\
c_{AB} &\geq c_{BC}+c_{BD} && (4)
\end{aligned}
$$

Setze (2) in (3) ein und erhalte $c_{AC} \geq c_{CB}+c_{AC}+c_{BC} \Leftrightarrow 0 \geq c_{CB}+c_{BC}$. Da Kapazitäten nicht negativ sein können, gilt $c_{CB} = c_{BC} = 0$. Dies eingesetzt in die anderen Formeln liefert

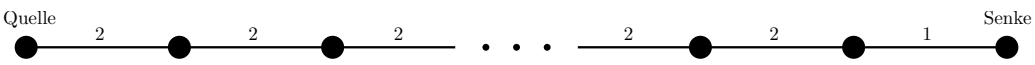
$$
\begin{aligned}
c_{BD} &\geq c_{AB} && (1)\\
c_{CD} &\geq c_{AC} && (2)\\
c_{AC} &\geq c_{CD} && (3)\\
c_{AB} &\geq c_{BD} && (4)
\end{aligned}
$$

und damit $c_{AB} = c_{BD}$, $c_{AC} = c_{CD}$.

b) Unten abgebildetes Flussnetzwerk mit dem eingezeichneten Schnitt ist ein Gegenbeispiel. Links ist ein minimaler (s,t) Schnitt mit Wert 1001 abgebildet. Der minimale (x,y) Schnitt mit dem Wert 1 ist rechts zu sehen.



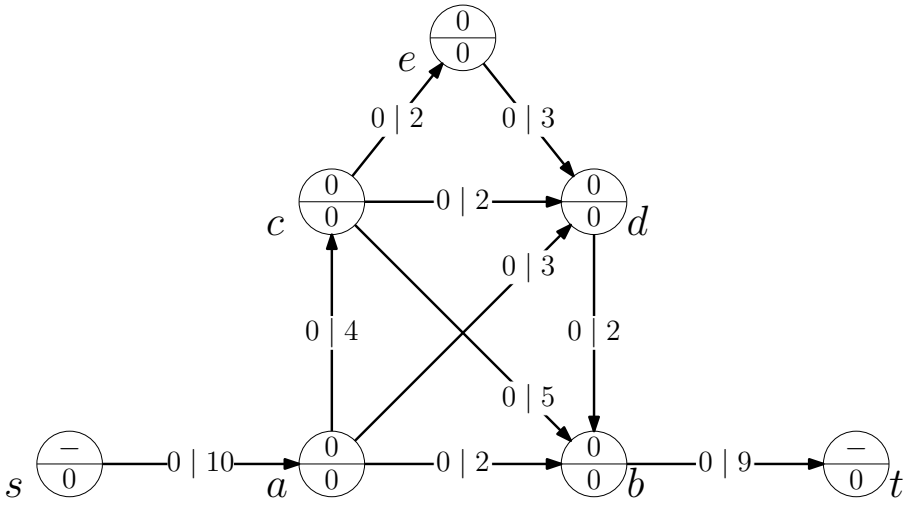
c) Wir betrachten folgenden Graphen:



Um einen Fluss auf diesem Graphen zu berechnen, muss der Fluss einmal durch den ganzen Graphen und wieder zurückfließen. Damit der zusätzliche Fluss vom vorletzten Knoten wieder in die Quelle zurückfließen kann, muss der Knoten mindestens Level $n+1$ haben. Somit ergibt sich über alle Knoten $\#relabel \geq \sum_{i=1}^{n-1} n+1 = n^2-1$.

Aufgabe 6 (*Rechnen: preflow-push Algorithmus*)

Gegeben sei folgender Flussgraph:



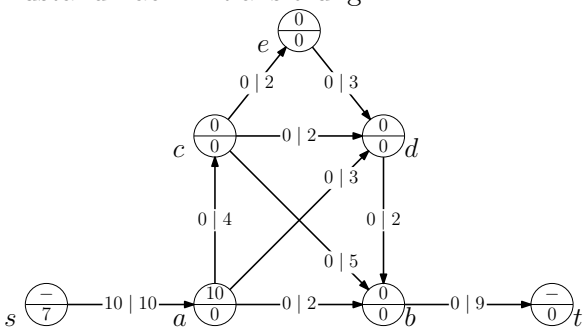
(das Haus vom Nikolaus mit Vorgarten und Hinterhof)

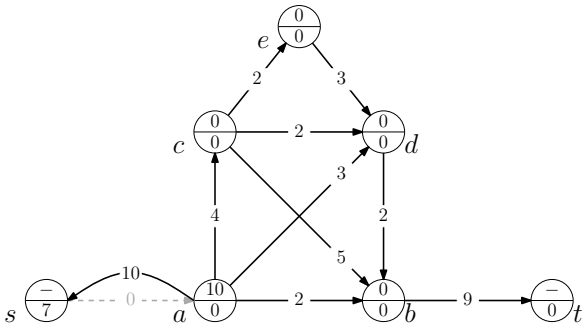
Knotenbeschriftung: Level (unten), Überschuss (oben)
Kantenbeschriftung: Fluss (vorne), Kapazität (hinten)

Bestimmen Sie den maximalen Fluss von s nach t mit dem generischen *preflow-push* Algorithmus.

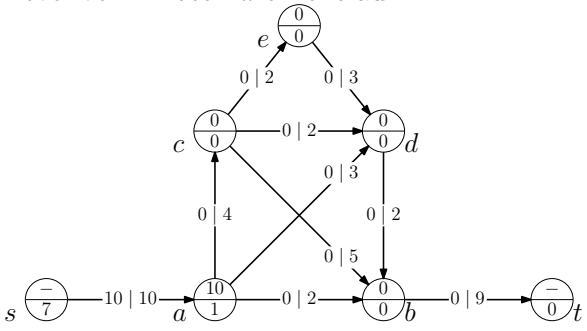
Musterlösung:
Im Folgenden wird der *preflow-push* Algorithmus aus der Vorlesung auf den Flussgraphen angewendet. Aktive Knoten werden zufällig ausgewählt. Es wird bei einem Knoten geblieben bis dessen gesamter Überschuss weggeschoben wurde. Dieser Ablauf ist *nicht* der schnellstmögliche!
Links ist der Zustand des Flussgraphen nach jedem Schritt zu sehen, rechts der des Residualgraphen.

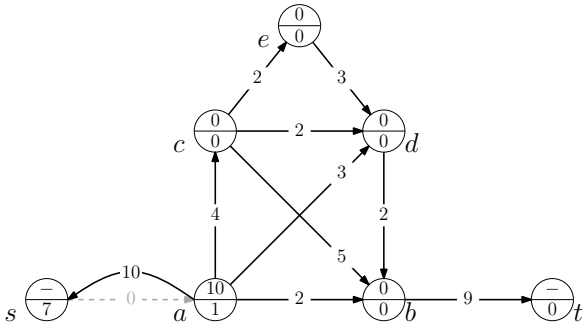
Zustand nach Initialisierung:





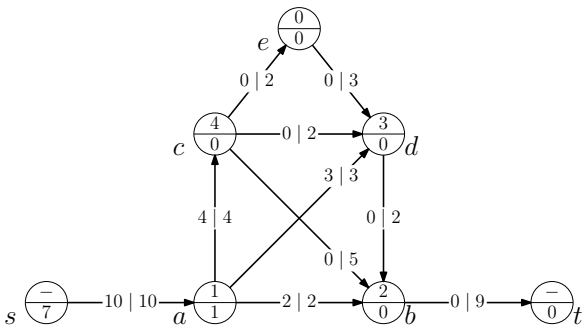
Level von Knoten a erhöht auf 1:

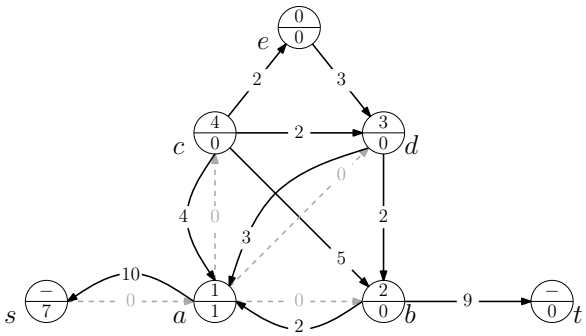




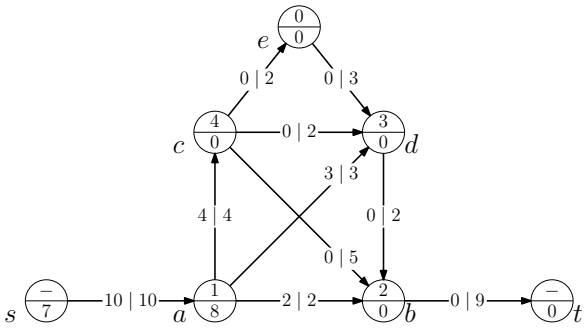
Musterlösung:

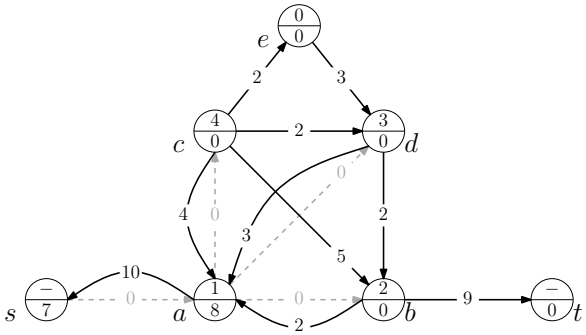
Fluss von a nach b, c und d geschoben:



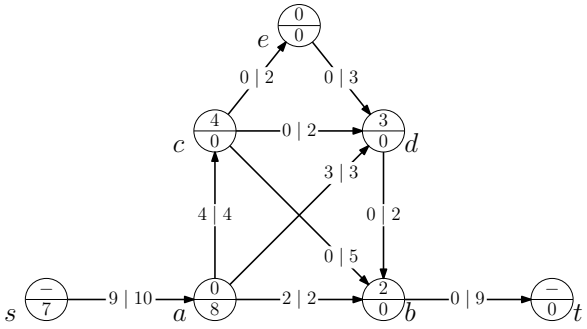


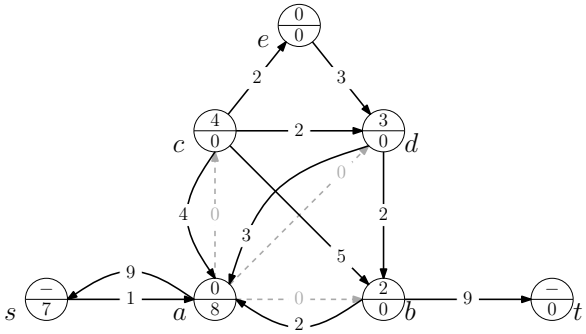
Level von Knoten a erhöht auf 8:



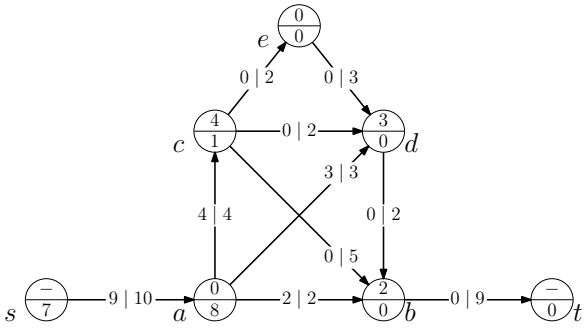


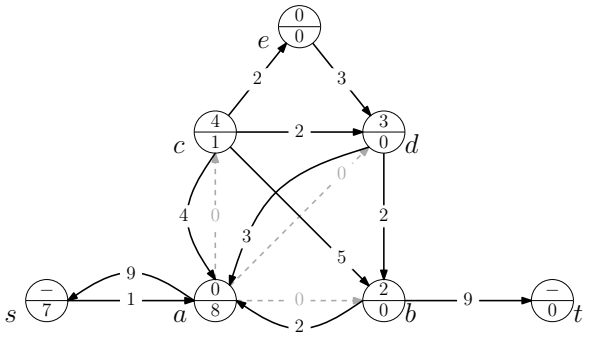
Fluss von a nach s geschoben:





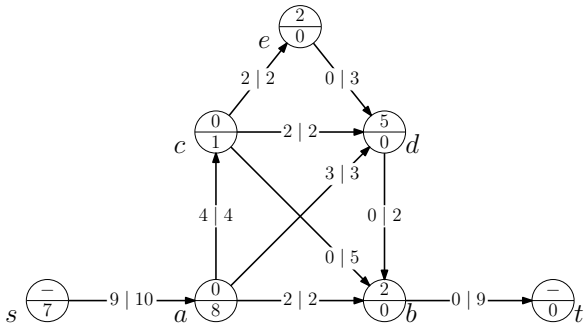
Level von Knoten c erhöht auf 1:

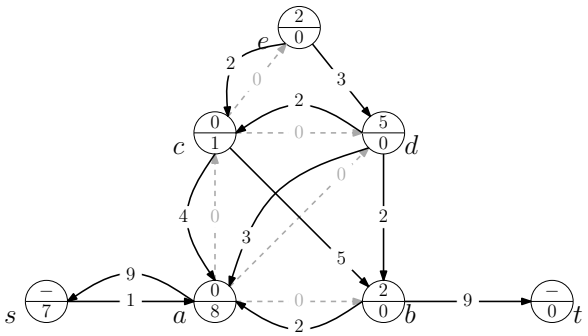




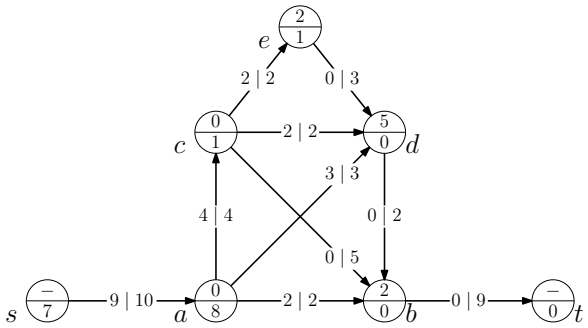
Musterlösung:

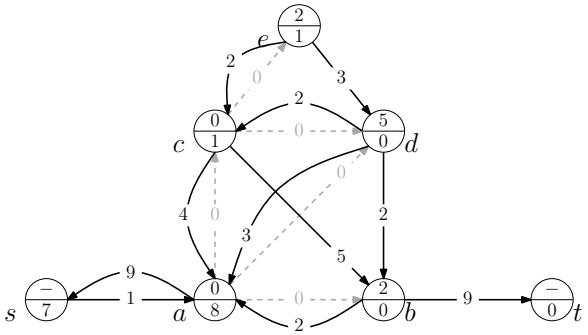
Fluss von c nach d und e geschoben:



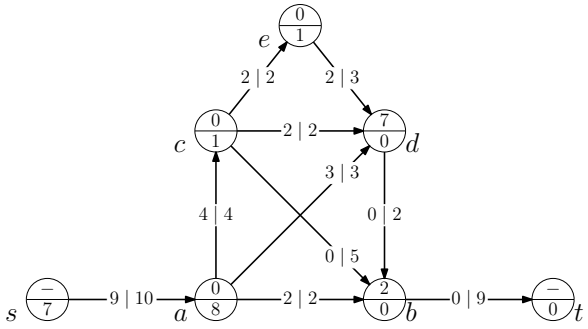


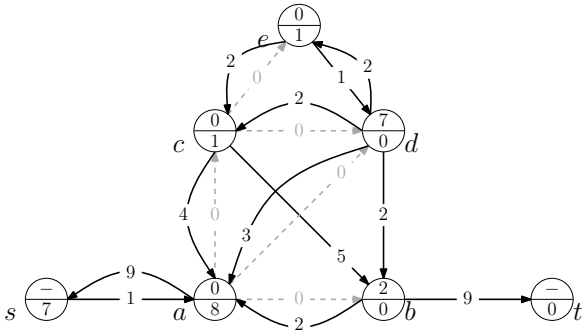
Level von Knoten e erhöht auf 1:



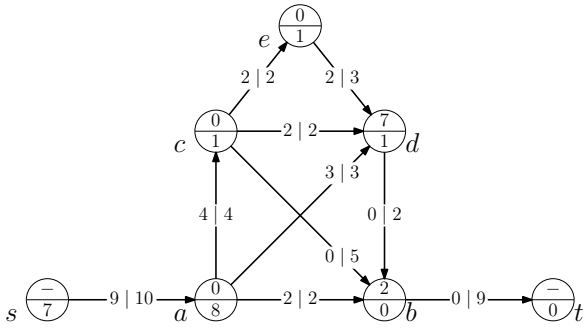


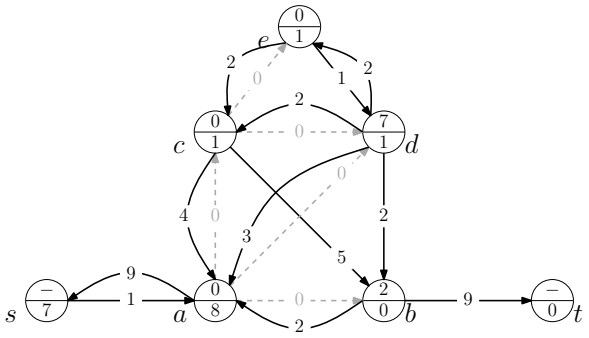
Fluss von e nach d geschoben:





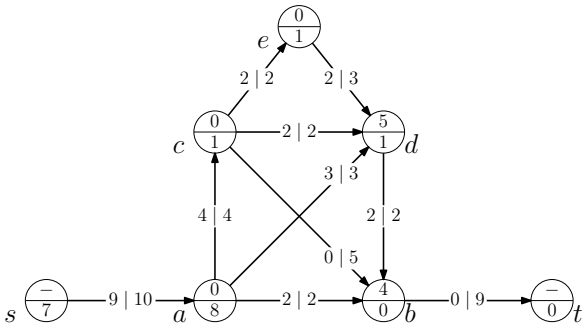
Level von Knoten d erhöht auf 1:

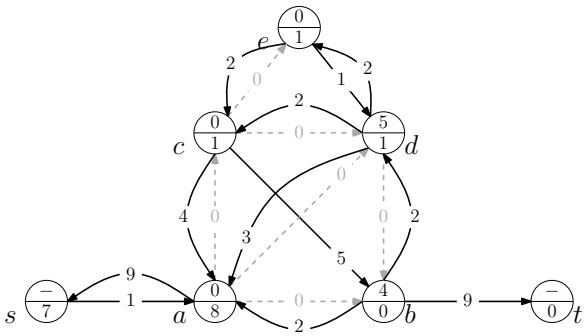




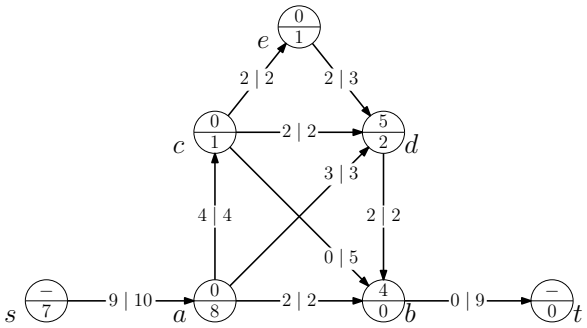
Musterlösung:

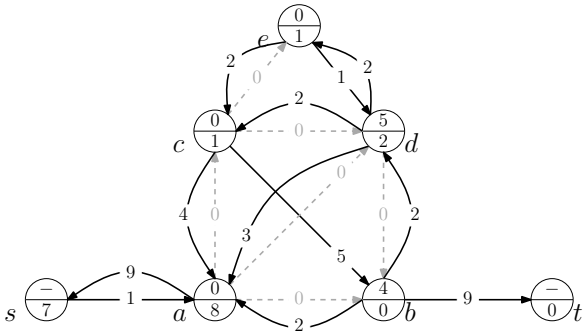
Fluss von d nach b geschoben:



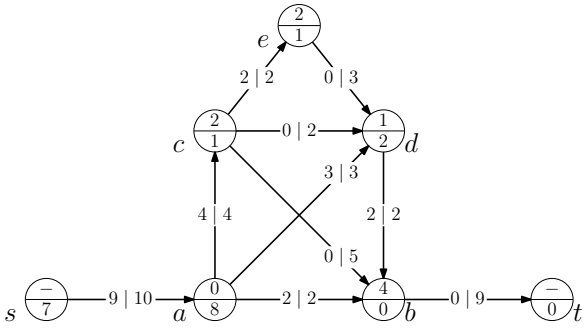


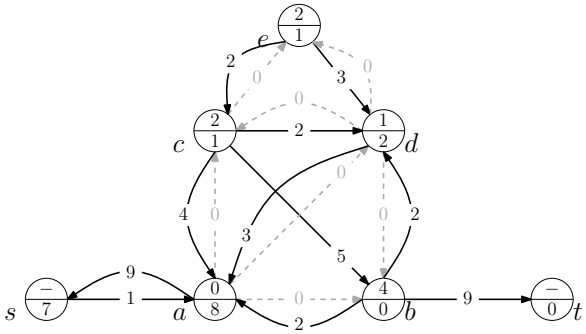
Level von Knoten d erhöht auf 2:



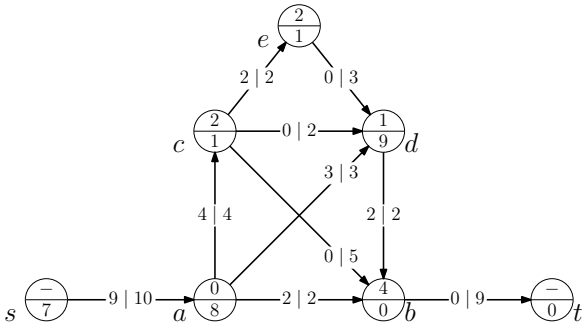


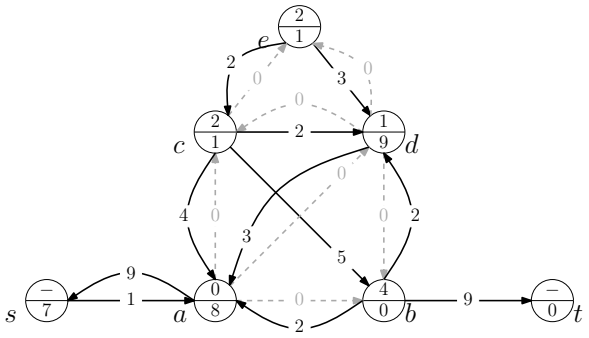
Fluss von d nach c und e geschoben:





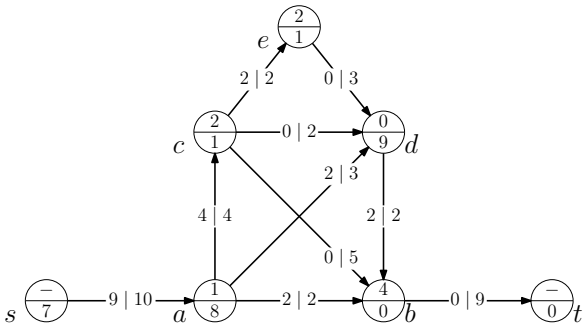
Level von Knoten d erhöht auf 9:

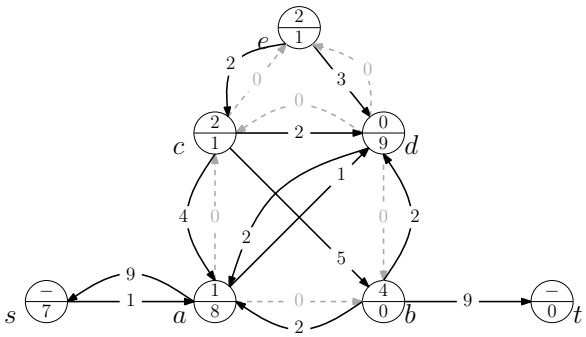




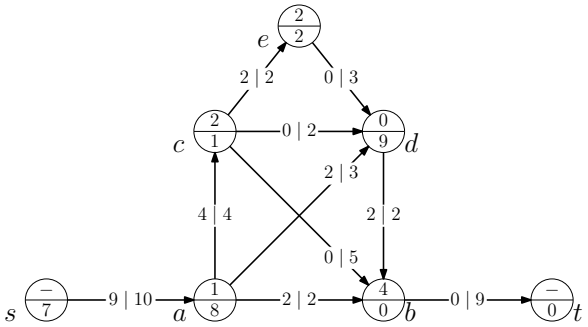
Musterlösung:

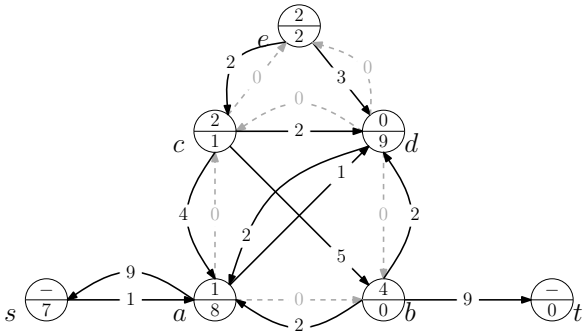
Fluss von d nach a geschoben:



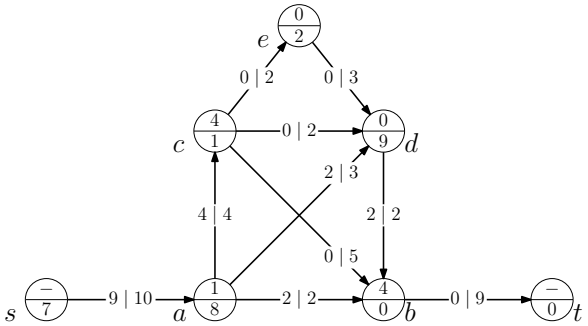


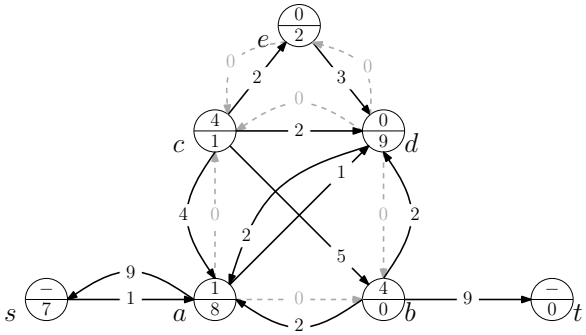
Level von Knoten e erhöht auf 2:



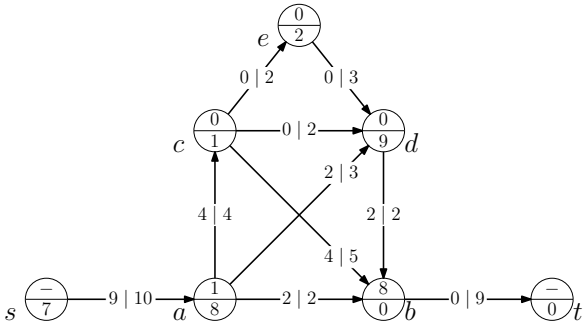


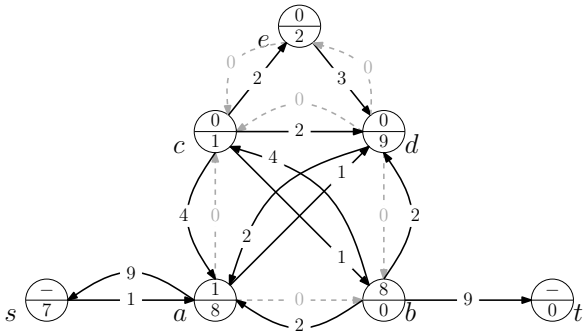
Fluss von e nach c geschoben:





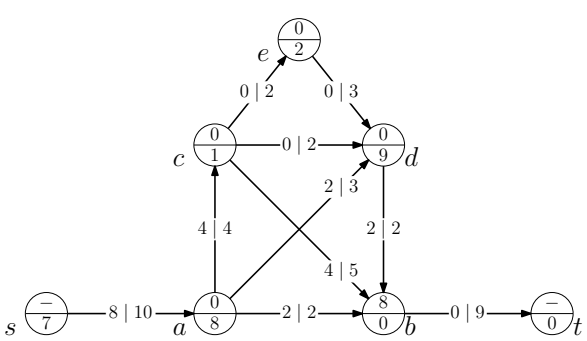
Fluss von c nach b geschoben:

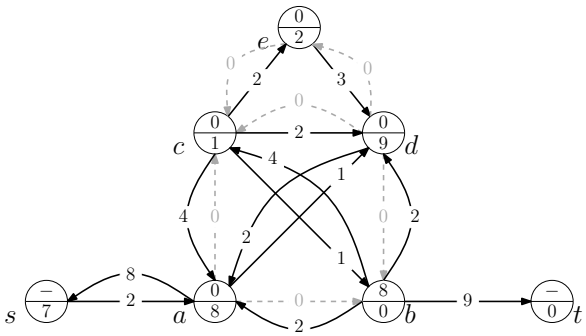




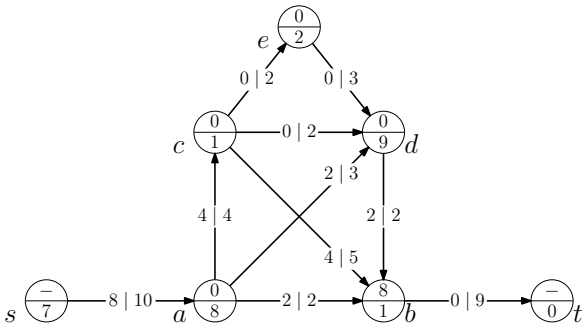
Musterlösung:

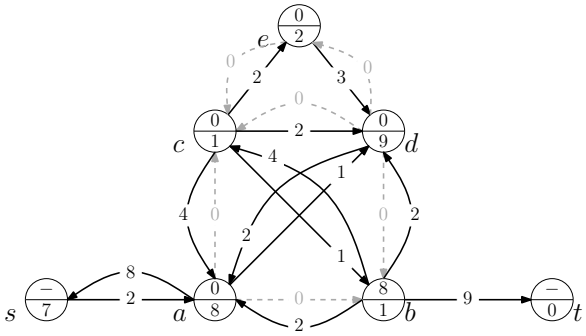
Fluss von a nach s geschoben:



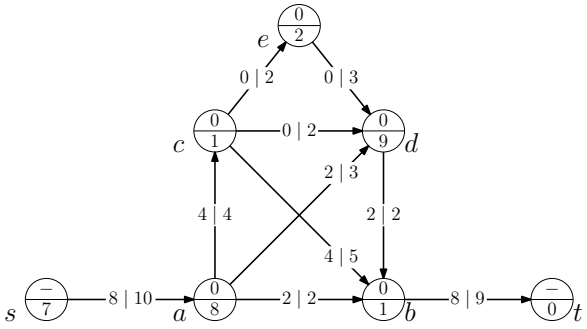


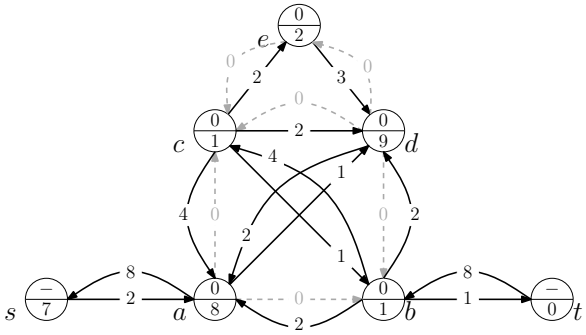
Level von Knoten b erhöht auf 1:





Fluss von b nach t geschoben:





Fertig!