Name: Matrikelnummer:

Klausur Algorithmen II, 3.3.2011 Blatt 1 von 1?

Lösungsvorschlag

Aufgabe 1. Kleinaufgaben [14 Punkte]

a. Nennen Sie zwei Dinge, die im Algorithm Engineering im Gegensatz zu theoretischer Algorithmik anders gemacht werden. [2 Punkte]

Lösung

Implementierung, Experimente, reale Eingaben, realistische Maschinenmodelle, konstante Faktoren berücksichtigen, Einfachheit von Algorithmen, ...

Lösungsende

b. Was ist der Unterschied zwischen einem PTAS und einem FPTAS? [2 Punkte]

Lösung

Ein FPTAS muss eine polynomielle Laufzeit in $1/\varepsilon$ haben, die Laufzeit eines PTAS kann beliebig von ε abhängen.

Lösungsende

c. Gegeben sei ein Bitvektor, der die Operation *rank* in $O(1)$ Laufzeit unterstützt. Geben Sie eine Funktion *rank_interval* an, die die Anzahl 1en in einem beliebigen Intervall $[i, j]$ dieses Bitvektors in konstanter Zeit zurückgibt.
Hinweis: *rank(i)* gibt die Anzahl 1en im Intervall $[0, i)$ an.

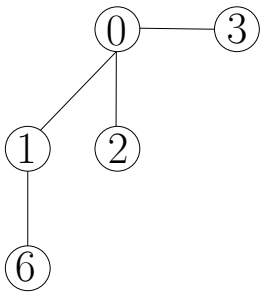
[2 Punkte]

Lösung

procedure rank_interval(i, j)
return rank($j+1$) - rank(i)

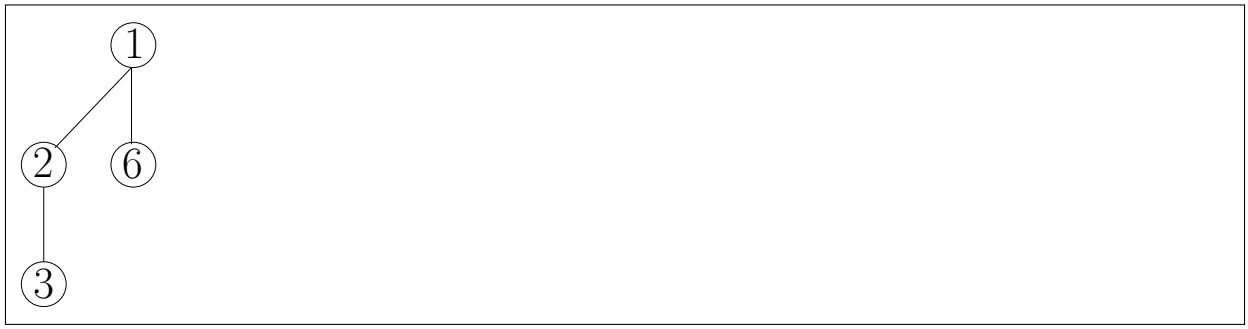
Lösungsende

d. Gegeben sei folgender Fibonacci-Heap. Geben Sie den resultierenden Fibonacci-Heap nach der Operation *delete*(0) an.



[2 Punkte]

Lösung



Lösungsende

(weitere Teilaufgaben auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 1

e. Die Ausführungszeit eines parallelen Algorithmus zum vergleichsbasierten Sortieren von n Elementen auf p Prozessoren sei:

$$T(p) := \Theta\left(\frac{n \log^2 n}{p}\right)$$

Geben Sie den absoluten Speedup (Beschleunigung) und die absolute Effizienz an. Wie muss die Prozessorzahl mit der Eingabegröße asymptotisch steigen, damit der Speedup konstant bleibt? Nehmen Sie immer den schlimmsten Fall an, und rechnen Sie im Θ-Kalkül.

[3 Punkte]

Lösung

Der absolute *Speedup* ist definiert als

$$S(p) = \frac{T_{\text{seq}}}{T(p)},$$

wobei p die Prozessorzahl, T_{seq} die asymptotische Ausführungszeit des besten sequentiellen Algorithmus ist. Die (absolute) Effizienz ist definiert als

$$E(p) = \frac{S(p)}{p} = \frac{T_{\text{seq}}}{pT(p)}.$$

Die besten sequentiellen Algorithmen für vergleichsbasiertes Sortieren haben Laufzeit $\Theta(n \log n)$. Damit ist der Speedup

$$S(p) = \frac{\Theta(n \log n)}{\Theta\left(\frac{n \log^2 n}{p}\right)} = \Theta\left(\frac{p}{\log n}\right)$$

und die Effizienz

$$E(p) = \frac{\Theta\left(\frac{p}{\log n}\right)}{p} = \Theta\left(\frac{1}{\log n}\right)$$

Aus konstantem Speedup

$$S(p) = \Theta\left(\frac{p}{\log n}\right) \stackrel{!}{=} \Theta(1)$$

folgt

$$p = \Theta(\log n)$$

Lösungsende

f. Sei k ein Parameter und n die Eingabegröße. Welche der folgenden Laufzeiten machen ein Problem *fixed-parameter-tractable*? Begründen Sie Ihre Antwort jeweils kurz.

1. $O(k^n \cdot \log n)$

2. $O(n \cdot k^6 \cdot n^{1.387536})$

3. $O(k \cdot n^{\log n})$

[3 Punkte]

Lösung

Ein parametrisiertes Problem Π heißt *fixed-parameter-tractable*, wenn es einen Lösungsalgorithmus zu Π mit Laufzeit $O(f(k) \cdot p(n))$ gibt. Dabei ist p ein Polynom, und f eine berechenbare Funktion, die nur von k abhängt.

1. Beide Faktoren hängen von n ab, und sie ergeben zusammen kein Polynom.
2. Ja, denn $n \cdot k^6 \cdot n^{1.387536} = k^6 \cdot n^{2.387536}$, was der FPT-Definition mit $f(k) = k^6$ und $p(n) = n^{2.387536}$ entspricht.
3. Nein, denn $n^{\log n}$ ist kein Polynom.

Lösungsende

Lösungsvorschlag

Aufgabe 2. Algorithmen-Entwurf: Skyline [12 Punkte]

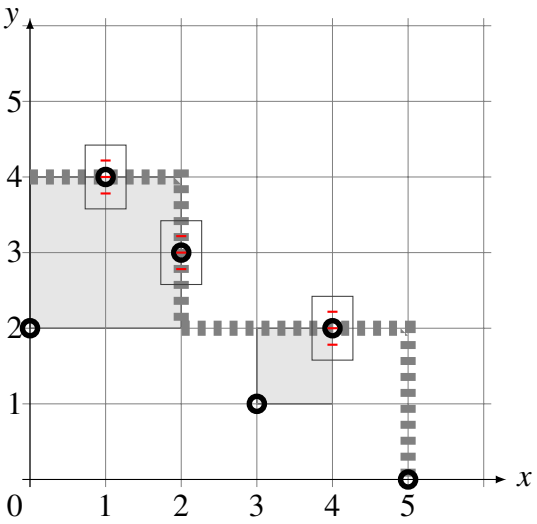
Gegeben sei eine Menge P von Paaren (2-Tupeln) (p_1, p_2) reeller Zahlen. Die *Skyline* $S(P)$ von P ist die Menge aller Elemente, die von keinem anderen Element dominiert werden, d. h. $S(P) := \{p \in P : \neg\exists q \in P : q \neq p \wedge q_1 \geq p_1 \wedge q_2 \geq p_2\}$.

a. Geben Sie die Skyline folgender Elemente als **Liste** von Paaren an:
$P := \{(2,3)(5,0)(4,2)(3,1)(0,2)(1,4)\}$

[3 Punkte]

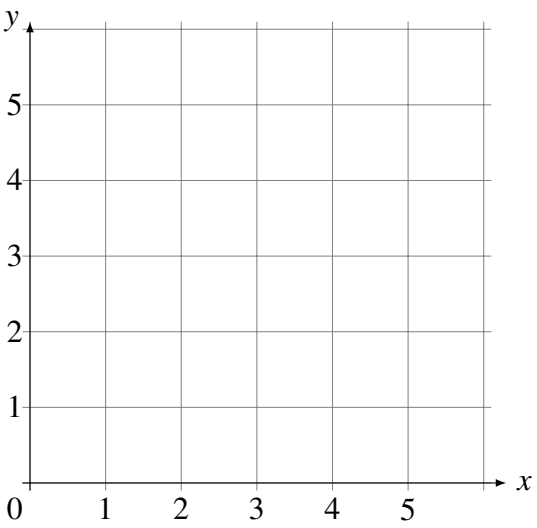
Lösung

$S(P) = \{(1,4)(2,3)(4,2)(5,0)\}$
$(3,1)$ wird von $(4,2)$ dominiert. $(0,2)$ wird (z. B.) von $(1,4)$ dominiert.



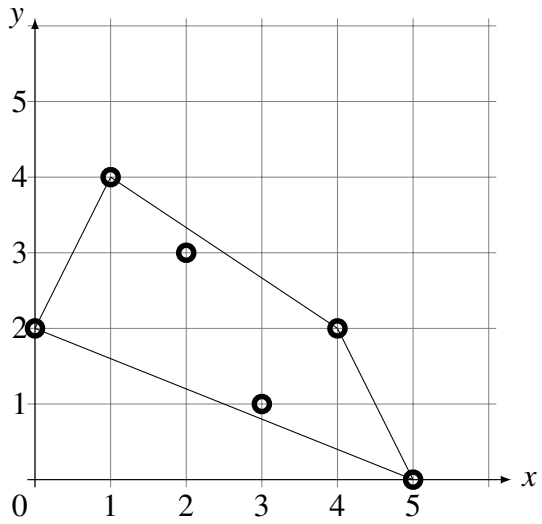
Lösungsende

b. Interpretieren Sie die (ursprünglichen) Paare aus **a.** als 2-dimensionale Koordinaten und geben Sie die konvexe Hülle als **Liste** von Paaren an. [3 Punkte]



Lösung

$S(P) = \{(5,0)(4,2)(0,2)(1,4)\}$
$(3,1)$ und $(2,3)$ sind nicht in der konvexen Hülle.



Lösungsende

(weitere Teilaufgabe auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 2

c. Geben Sie einen Algorithmus an, der $S(P)$ in $O(n \log n)$ Laufzeit berechnet.
Sie können dazu auch wie immer alle aus der Vorlesung bekannte Datenstrukturen und Algorithmen verwenden. Als Antwort werden wie immer Pseudo-Code oder eine natürlichsprachige Erklärung akzeptiert.
Begründen Sie, warum die geforderte Laufzeit erreicht wird. [6 Punkte]

Lösung

Die Paare werden als zweidimensionale Koordinaten $(p.x, p.y)$ interpretiert.

- Lösung mittels Sortieren

  procedure Skyline(P)
  sortiere P lexikographisch, absteigend
  $R := \{\}$
  $y = -\infty$
  for $p \in P$ **do**
  if $p.y > y$ **then**
  $R := R \cup \{p\}$
  $y := p.y$
  end if
  end for
  return R

  Die Schleife braucht $O(n)$ Zeit, also dominiert die Laufzeit von Sortieren, $O(\log n)$ Zeit. Damit liegt der Algorithmus im Rahmen.

- Lösung mittels Wavelet Tree

  procedure Skyline(P)
  konstruiere Wavelet Tree W aus P
  $R := \{\}$
  for $p \in P$ **do**
  if W.dominanceCount(p) = 0 **then**
  $R := R \cup \{p\}$
  end if
  end for
  return R

  Nachdem der Wavelet Tree in $O(n \log n)$ Zeit konstruiert wurde, braucht *intDominanceCount* n-mal $O(\log n)$ Zeit, was ebenfalls im Rahmen liegt.

Lösungsende

Lösungsvorschlag

Aufgabe 3. Online-Algorithmen: BahnCard [12 Punkte]

Gegeben ist das folgende Online-Problem:
Ein Bahn-Kunde reist n-mal mit dem Zug, $n \in \mathbb{N}_{\geq 0}$. Für den Kunden besteht die Möglichkeit, sich zum einem gewissen Zeitpunkt eine BahnCard 50 für 200 EUR zu kaufen. Ab diesem Zeitpunkt erhält der Kunde 50% Rabatt auf den regulären Preis, zahlt also nur die Hälfte. Er weiß aber nicht von vorneherein, zu welchen Zeitpunkten er Fahrkarten kaufen wird, auch nicht zu welchem Preis.
a. Geben Sie eine möglichst einfache Strategie an, die einen kompetitiven Faktor (competitive ratio) von 2 erreicht. [3 Punkte]

Lösung

Man kauft einfach nie eine BahnCard.
Optionale Begründung: Dann bezahlt man offensichtlich immer höchstens das Doppelte, also ist der kompetitive Faktor höchstens 2. Wenn OPT sofort eine Bahncard kauft, konvergiert das Verhältnis für die Gesamtsumme $P \to \infty$ gegen $\frac{P}{P/2+200} = 2$, also ist das tatsächlich der kompetitive Faktor.

Lösungsende

b. Wie gut ist die Strategie BLINDLINGS, sich sofort eine BahnCard 50 zu kaufen, im Vergleich zur optimalen Offline-Strategie OPT? [3 Punkte]

Lösung

Die Strategie BLINDLINGS ist beliebig viel schlechter als OPT. Fährt der Kunde gar nicht Bahn, kauft OPT weder eine Fahrkarten noch eine BahnCard, hat also Kosten 0 EUR. Die Strategie BLINDLINGS kauft aber sofort die BahnCard 50 für 200 EUR, hat also Kosten 200 EUR, und somit einen "kompetitiven Faktor" von $\frac{200}{0} = \infty$.

Lösungsende

(weitere Teilaufgabe auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 3

c. Ein Kunde verfährt nach der Strategie ABWARTEND. Er kauft die BahnCard 50% zu dem Zeitpunkt, wenn mit dem nächsten (gerade schon bekannten) Fahrkartenkauf die Gesamtsumme 400 EUR überschreiten würde. Er bekommt schon für diesen Kauf Rabatt.
Zeigen Sie, dass 1,5 eine obere Schranke für den kompetitiven Faktor dieser Strategie ist.
[6 Punkte]

Lösung

OPT unterscheidet zwei Möglichkeiten.

1. Die Summe aller Fahrpreise ist kleiner gleich 400 EUR. Dann kauft OPT keine Bahncard. In diesem Fall kauft ABWARTEND auch keine BahnCard, und hat somit die gleichen Kosten, $c \leq 1,5$ wird erfüllt.

2. Die Summe aller Fahrpreise P ist größer als 400 EUR. Dann kauft OPT gleich am Anfang eine Bahncard, hat also Gesamtkosten $200 + \frac{P}{2}$ EUR. ABWARTEND kauft ebenfalls eine BahnCard, aber erst, wenn die 400 EUR überschritten werden. Der Kunde kauft also insgesamt für höchstens 400 EUR Fahrkarten ohne Rabatt, die BahnCard, und die restlichen Fahrkarten mit Rabatt, hat also höchstens Gesamtkosten $400 + 200 + \frac{P-400}{2} = 400 + \frac{P}{2}$ EUR.

$$c \leq \frac{400 + P/2}{200 + P/2} < 1,5 \Leftrightarrow$$
$$400 + P/2 < 1,5(200 + P/2) \Leftrightarrow$$
$$400 + P/2 < 300 + 3P/4 \Leftrightarrow$$
$$100 < P/4 \Leftrightarrow$$
$$P > 400$$

Letzteres ist per Definition der Fall, also gilt im Umkehrschluss die Behauptung, da nur Äquivalenzumformungen vorgenommen wurden.

Lösungsende

Lösungsvorschlag

Aufgabe 4. Algorithmen-Entwurf: Schnittmenge [8 Punkte]

Gegeben seien zwei beliebig (un-) sortierte Listen L_1, L_2 mit jeweils Länge n. Aufgabe ist es, die Schnittmenge beider Listen zu berechnen, $L_1 \cap L_2$.

a. Angenommen, es gibt genügend Platz im Hauptspeicher.
Mittels einer Hash-Tabelle kann das Problem in erwarteter Zeit $O(n)$ gelöst werden. Zuerst wird jedes Element $\ell \in L_1$ in die Hashtabelle H eingefügt. Dann wird für jedes Element $\ell' \in L_2$ überprüft, ob es in H zu finden ist. Wenn ja, so wird es ausgegeben.
Welche asymptotische I/O-Komplexität ergibt sich im schlimmsten Fall für diesen Algorithmus im Externspeicher, d. h. wenn n größer als der Hauptspeicher ist? [3 Punkte]

Lösung

Trotz eventuellem Caching ergibt sich im schlimmsten Fall erwartet eine I/O-Operation pro Zugriff auf die externe Hashtabelle, also ingesamt $\Theta(n)$ I/Os.

Lösungsende

b. Angenommen, beide Listen passen nicht mehr in den Hauptspeicher. B sei die Blockgröße, und M die Größe des Hauptspeichers. Entwerfen Sie einen Algorithmus, der im Externspeicher für großes B und M deutlich weniger I/O-Operationen ausführt als der in Aufgabe **a.** angegebene. Geben Sie die asymptotische Anzahl I/O-Operationen für Ihren Algorithmus an. [5 Punkte]

Lösung

Per externem Mehrwege-Mischen werden beide Listen sortiert und die sortierten Liste mit einem angepassten Mischen auf gemeinsame Elemente überprüft. Die Zahl der I/O-Operationen für das Sortieren beträgt

$$\Theta\left(\frac{n}{B}\log_{M/B}\frac{n}{M}\right)$$

Das Mischen läuft in Zeit $\Theta(n/B)$ und wird somit vom Sortieren dominiert.

Lösungsende

Lösungsvorschlag

Aufgabe 5. Suffix-Arrays und -Bäume [6 Punkte]

Gegeben sei die Zeichenkette $S := \texttt{enemene\$}$
Es gilt $\texttt{\$} < \texttt{a} < \texttt{b} < \ldots < \texttt{z}$.

a. Geben Sie das Suffix-Array für S an.

[3 Punkte]

Lösung

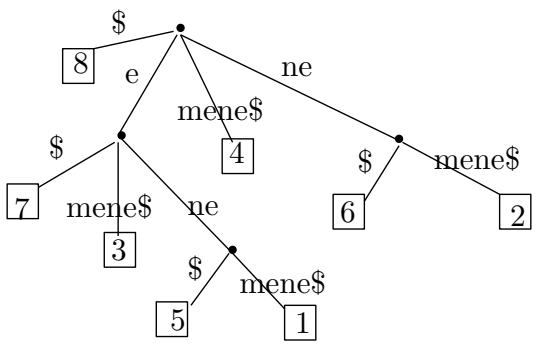
i	$SA[i]$	Suffix
1	8	$
2	7	e$
3	3	emene$
4	5	ene$
5	1	enemene$
6	4	mene$
7	6	ne$
8	2	nemene$

Akzeptiert werden Lösungen, die mindestens die Spalte $SA[i]$ oder die sortierte Liste der Suffixe angeben.

Lösungsende

b. Geben Sie den Suffix-Baum für S als kompaktierten Trie an. [3 Punkte]

Lösung



Lösungsende

Name: Matrikelnummer:

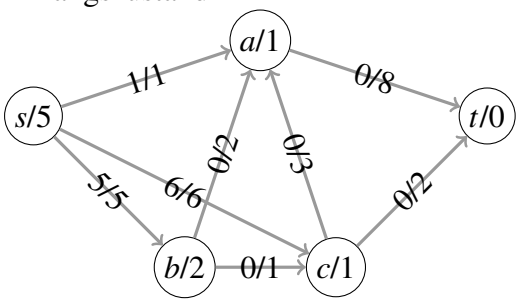
Klausur Algorithmen II, 3.3.2011 Blatt 12 von 12

Lösungsvorschlag

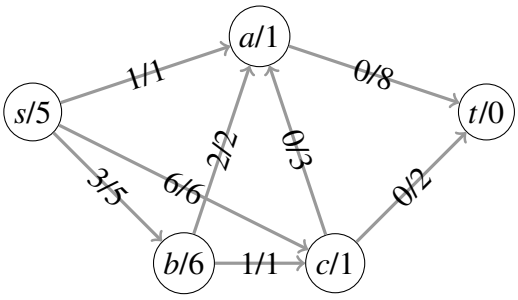
Aufgabe 6. Maximaler Fluss [8 Punkte]

Im folgenden sind zwei Zustände eines Preflow-Push-Algorithmus angegeben, ein Anfangs- und ein Endzustand.
Geben Sie eine mögliche Folge von vier Schritten (jeweils *push* oder *relabel* mit allen Parametern) des Algorithmus an, die den Anfangszustand in den Endzustand überführen. Fassen Sie dabei mehrfache direkt hintereinander folgende *relabel*-Operationen auf demselben Knoten zu einem Schritt zusammen.
In den Knoten steht „Knotenname“/„Level (d)“, and den Kanten „aktueller Fluss“/„Kapazität“. Die Kanten des Residualgraphen sowie der Exzess sind der Übersichtlichkeit halber weggelassen.

Anfangszustand

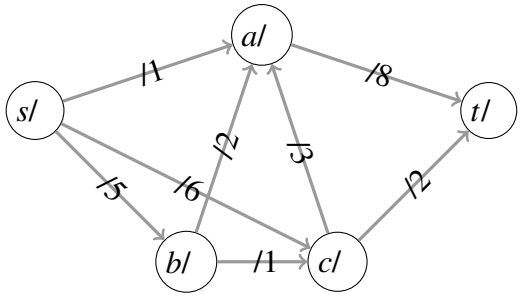


Endzustand

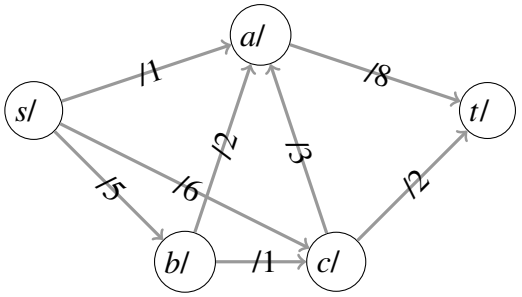


Lösung: (Eintragungen in die Graphen sind optional.)

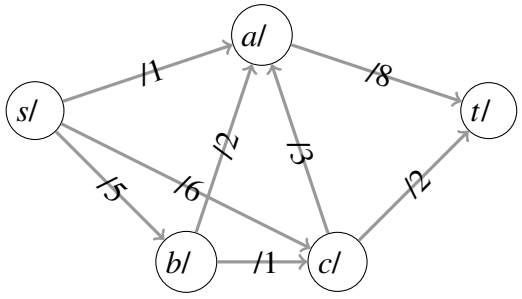
1. Schritt:



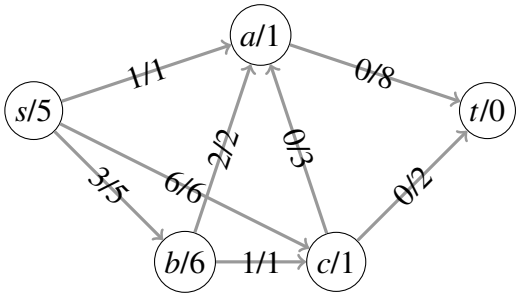
2. Schritt:



3. Schritt:

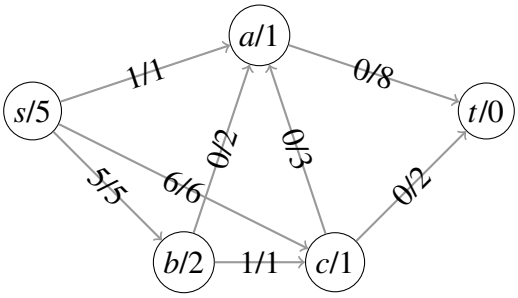


4. Schritt:

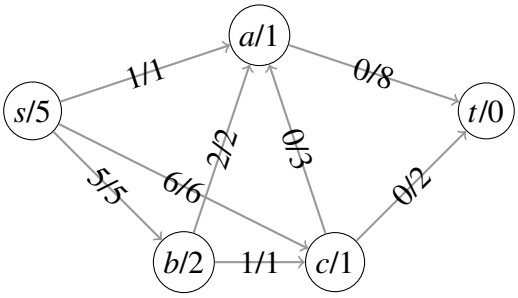


Lösung

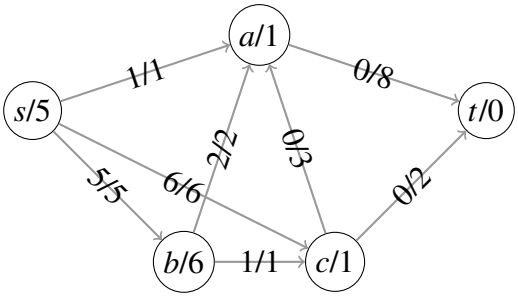
1. Schritt: push(b, c, 1)



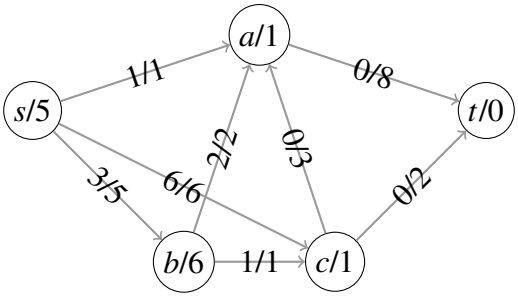
2. Schritt: push(b, a, 2)



3. Schritt: 4 $\times$ relabel(b)



4. Schritt: push(b, s, 2)



Lösungsende