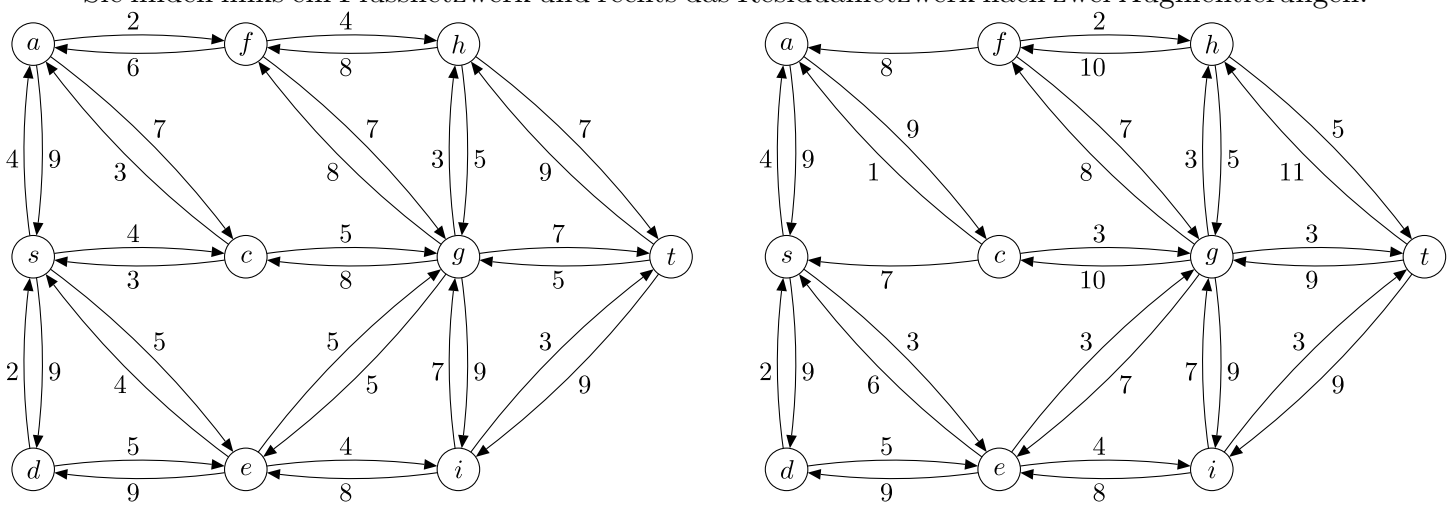
MUSTERLÖSUNG

Aufgabe K1 (6+4+11+4 Punkte)

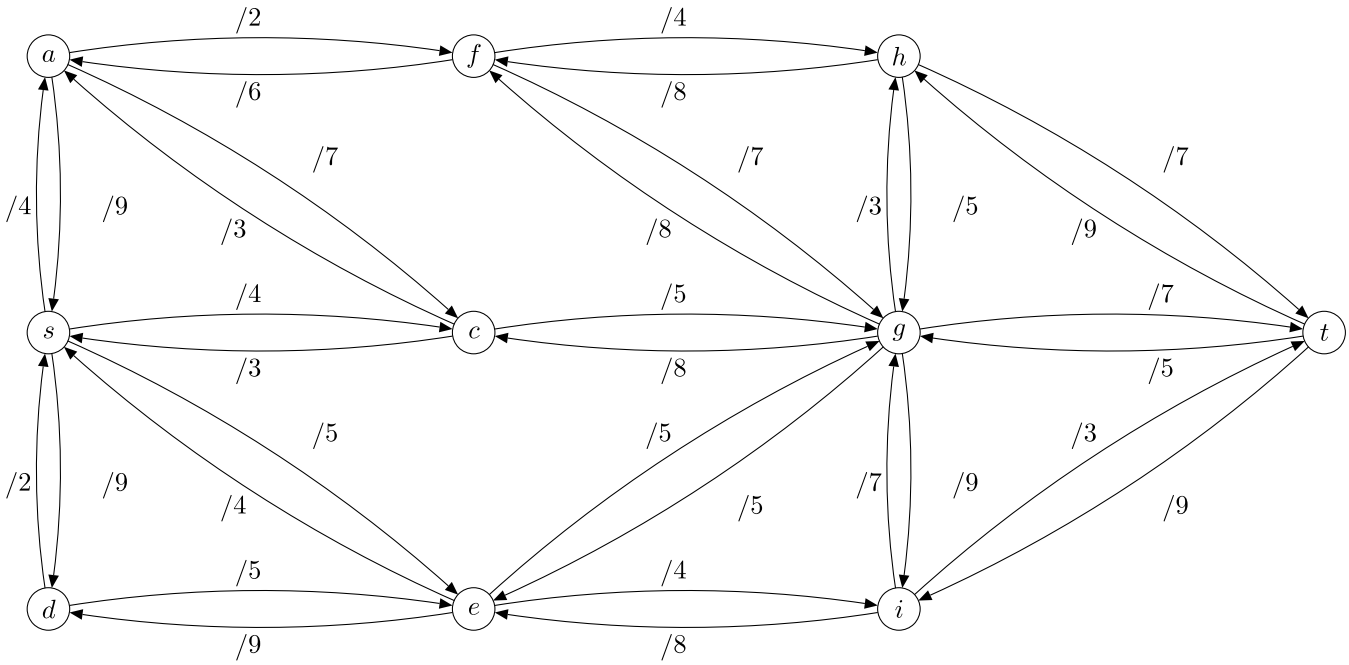
Sie finden links ein Flussnetzwerk und rechts das Residualnetzwerk nach zwei Augmentierungen:



a) Geben Sie die beiden augmentierenden Pfade an, indem Sie die zugehörigen Knoten in der entsprechenden Reihenfolge eintragen.

b) Was ist der Wert des Flusses, der zu dem obigen Residualnetzwerk gehört?

c) Finden Sie einen maximalen Fluss zu dem Flussnetzwerk links und zeichnen Sie ihn hier ein (keine Nullen, nur positive Werte). Im Anhang dieser Klausur sind einige Kopien der unten stehenden Grafik, die sie als Schmierpapier verwenden können.



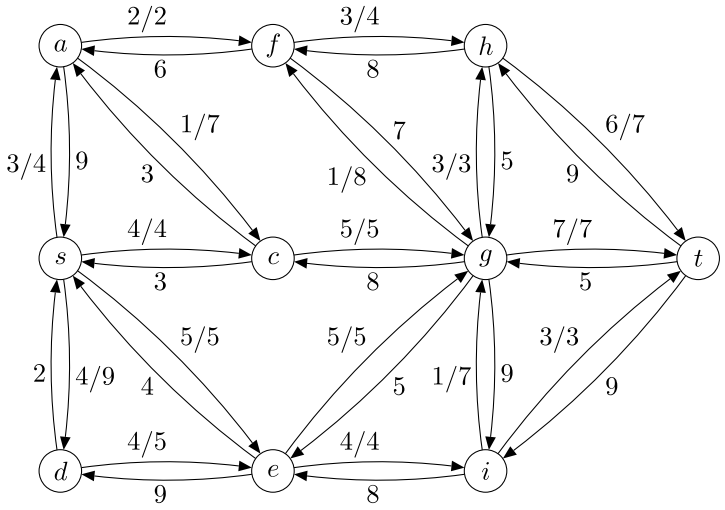
d) Geben Sie einen Schnitt (S, T) an, dessen Kapazität minimal ist:
$S = \{\}$, $T = \{\}$. Die Kapazität ist .

Lösungsvorschlag

a) *scgt* und *segcafht*.
Typische Fehler: Es wurden mehr als zwei augmentierende Pfade gefunden, oder solche, die in Kombination nicht den gesamten Fluss abdecken.

b) Der Wert des Flusses ist 6.

c)



Typische Fehler: Es wurde ein augmentierender Pfad übersehen, es wurde mehr Fluss über eine Kante geschickt als Kapazität vorhanden war.
Typische Kuriosität, die zu keinem Punktabzug führt: Es wurde zwischen zwei Knoten in beide Richtungen Fluss geschickt.

d) (S, T) mit $S = \{s, a, c, d, e\}$ und $T = \{f, g, h, i, t\}$. Die Kapazität des Schnitts ist 16.

Aufgabe K2 (18 Punkte)
Ein Gemischtwarenladen will durch einen besonderen Rabatt die Umsätze steigern. Der Laden bietet eine Menge von Artikeln $p_1, \ldots, p_n$ mit Preis $c_1, \ldots, c_n$ in Cent an. Wer an der Kasse zwei Artikel p_i und p_j mit $i \neq j$ und $i, j \in \{1, \ldots, n\}$ zum Kauf vorlegt, deren Gesamtpreis $c_i + c_j$ auf 11, 33, 55, 77 oder 99 Cent endet, erhält zusätzlich einen Gutschein. Dabei kann jede Person jeden Artikel nur einmal kaufen und nur einmal nutzen, um einen Gutschein zu erhalten.
Entwerfen Sie einen Algorithmus der eine Menge von Paaren an Artikeln findet, sodass jeder Artikel nur einmal in einem Paar vorkommt und die erhaltenen Gutscheine maximiert werden. Geben Sie außerdem an, warum ihr Algorithmus korrekt ist.

Lösungsvorschlag
Das Problem lässt sich leicht als Matchingproblem darstellen. Wir konstruieren einen bipartiten Graphen $G = (A, B, E)$ mit den folgenden Knoten $A = \{1 \leq i \leq n \mid c_i \text{ ist gerade}\}$ und $B = \{1 \leq i \leq n \mid c_i \text{ ist ungerade}\}$. Wir verbinden $i \in A$ mit $j \in B$, falls der Centbetrag von $c_i + c_j$ entweder 11, 33, 55, 77 oder 99 ist.
Es ist leicht einzusehen, dass ein maximales Matching in G genau eine Lösung unseres Problems darstellt. Formal wäre zu zeigen, dass jedes Matching eine Lösung unseres Problems darstellt und anders herum. Auf Grund der Konstruktion ist dies jedoch offensichtlich.

Aufgabe K3 (10+15 Punkte)

a) Gegeben sei eine beliebige, endliche Menge X.

Es sei $\mathcal{X} = \{I \in 2^X \mid |I| \bmod 2 = 0\}$.

Beweisen oder widerlegen Sie, dass $(X, \mathcal{X})$ ein Matroid ist.

b) Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit $|V| \geq 3$. Mit $G[F]$ bezeichnen wir den von der Kantenmenge $F \subseteq E$ induzierten Untergraphen. Dieser hat als Knoten alle Knoten des Graphen G und als Kantenmenge genau F.

Es sei $\mathcal{F} = \{\, F \subseteq E \mid G[F] \text{ ist kreisfrei und hat mindestens drei Komponenten} \,\}$.

Beweisen oder widerlegen Sie, dass $(E, \mathcal{F})$ ein Matroid ist.

Lösungsvorschlag

a) Dies gilt nicht. Wir widerlegen die Aussage mit der Menge $X = \{1, 2\}$ als Gegenbeispiel. Es gilt $\{1, 2\} \in \mathcal{X}$, aber $\{1\} \notin \mathcal{X}$. Daher ist die Struktur nicht hereditär und damit auch kein Matroid.

b) Wir müssen folgende zwei Eigenschaften aus der Vorlesung zeigen:

Ein Matroid $M = (S, I)$ besteht aus einer Basis S und einer Familie $I \subseteq 2^S$ von unabhängigen Mengen mit:

1. Falls $A \subseteq B$ und $B \in I$, dann $A \in I$ (M ist hereditary).
2. Falls $A, B \in I$ und $|A| < |B|$ dann gibt es ein $x \in B$ so dass $A \cup \{x\} \in I$ (M hat die Austauscheigenschaft).

Die erste Eigenschaft ist einfach zu sehen, da das Entfernen einer Kante niemals Kreise hinzufügt oder Komponenten verbindet.

Für die zweite Eingeschaften beachten wir, dass ein Wald mit k Kanten aus $|V| - k$ Bäumen besteht. Sei nun $G_A = (V, A)$ und $G_B = (V, B)$ mit $A, B \in I$ und $|A| < |B|$.

Demnach hat G_B mehr Kanten und damit weniger Bäume als G_A. Da G_B mindestens aus drei Bäumen besteht muss G_A mindestens aus vier Bäumen bestehen. Nehmen wir an jede Kante in B schließt einen Kreis in $G[A]$. Diesen Kreis gibt es nicht in $G[B]$. Das heißt, dass dieser Kreis in $G[A]$ mit (u, v) eine Kante enthält die nicht in B ist für jede Kante aus B. Daraus folgt, dass A mindestens so groß wie B ist (Widerspruch). Es ist also möglich eine weitere Kante aus B in A einzufügen so dass $A \in \mathcal{F}$ erhalten bleibt.

Aufgabe K4 (13+9 Punkte)

In einem Hotel in Mexiko kommen und gehen Gäste. Ihre Aufgabe ist es einen Überblick über die belegten Zimmer zu behalten. Ankommende Gäste können sich eine Zimmernummer wünschen, ansonsten belegen sie das *erste freie Zimmer*. Ein Gast kann jederzeit gehen; dabei teilt er seine Zimmernummer mit. (Es ist sichergestellt, dass ein Gast nie nach einem besetzen Raum fragt.)
Es gibt also die drei Anfragen:

- `checkin i` bedeutet, dass Zimmer `i` belegt wird.
- `checkout i` bedeutet, dass Zimmer `i` frei wird.
- `checkin mex` bedeutet, dass das erste freie Zimmer belegt wird. Dabei soll die vergebene Zimmernummer ausgegeben werden.

(a) Gehen Sie davon aus, dass Sie die Anzahl der Anfragen n schon zu Beginn wissen. Geben Sie einen Algorithmus an, der jede Anfrage in Zeit $O(\log n)$ bearbeitet wird und anfangs einmal $O(n \log n)$ Zeit zum Initialisieren braucht.

(b) *Wir empfehlen, diese Teilaufgabe erst dann zu bearbeiten, wenn Sie bereits mit der Bearbeitung der restlichen Klausur fertig sind.* Lösen Sie nun das gleiche Problem wie in Teilaufgabe (a), aber von Beginn an zu wissen, welchen Wert n hat.

Ein Beispiel für $n = 8$:

```
checkin 2
checkin mex
checkin 3
checkin 20 checkin mex
```

`checkout 4`
`checkout 2`
`checkin mex`
Die Ausgabe für die drei `checkin mex` Anfragen sind dann 1, 4 und 2.

Lösungsvorschlag

Wir behalten zu jedem Zeitpunkt die verteilten/nichtverteilten Zimmer. Wir speichern die aktuell belegten Zimmer als binären Suchbaum B, z.B. einem AVL-Baum, ab. Zusätzlich speichern wird die freien Räume im Interval 1 bis n als Suchbaum F ab. Anfangs ist B leer, und F mit den Räumen $1 \dots n$ gefüllt. Dies dauert maximal $O(n \log n)$.

Bei einer `checkin` x Anfrage wird x in B hinzugefügt und aus F entfernt (letzteres falls vorhanden/im Interval $1, \dots, n$). Entfernen und Hinzufügen dauert $O(\log n)$ in AVL-Bäumen. Die `checkout` Anfrage funktioniert genau umgekehrt. Bei `checkin mex` wird in logarithmischer Zeit nach dem Minimum in F gesucht und dann wird dieser wie oben eingecheckt.

Es reicht für `checkin mex` nur die ersten n Zimmer zu betrachten, da nach n Anfragen höchstens n Zimmer belegt sind, sodass das minimale freie Zimmer immer im Interval $1, \dots, n$ liegen muss.

Für Teil (b) können wir diesen Trick nicht direkt so umsetzen, da n zur Initalisierungszeit von F unbekannt ist. Wir werden den Trick so erweitern, dass das betrachtete Interval mitwächst: Natürlich ist nach einer Anfrage $n = 1$, nach zweien ist $n = 2$, usw. D.h. wir wollen F nach der i-ten Anfrage so haben, dass es alle freien Zimmer von 1 bis i enthält. Dies kann man bewerkstelligen, indem man nach der $i$ten Anfrage $i + 1$ in F hinzufügt, falls es nicht schon in B enthalten ist. Der Rest bleibt im Vergleich zu Teil (a) identisch.