Technische Universität Berlin **SS 2017**
Fakultät II, Institut für Mathematik
Sekretariat MA 6–2, Antje Schulz
Prof. Dr. Michael Joswig
Sven Jäger, Georg Loho, Dr. Frank Lutz

Modulklausur zur Computerorientierten Mathematik I+II

25.07.2017, Beginn: 08:15 Uhr, Bearbeitungszeit: 150 Minuten

Nachname, Vorname: _____

Matrikelnummer: _____

Studiengang: _____

Es sind keine Hilfsmittel erlaubt. Bitte nicht mit Bleistift oder mit roter Farbe schreiben!

*Die Klausur ist mit mindestens **36** (ITM/NidI: **28**) **Punkten** bestanden.*

1	2	3	4	5	6	7	8	Summe
9	9	13	11	10	10	6	12	80

1. Aufgabe $(3 + 2 + 4$ Punkte$)$

(a) (i) Zeichen Sie den einfachen ungerichteten Graphen G mit Inzidenzmatrix

$$I_G = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

und beschriften Sie seine Knoten.

(ii) Geben Sie die Adjazenzmatrix von G mit den zugehörigen Beschriftungen an.

(b) Sei G ein einfacher ungerichteter Graph mit n Knoten. Geben Sie zwei weitere äquivalente Bedingungen an, unter denen G ein Baum ist:

(i) G ist kreisfrei und zusammenhängend.

(ii) G ist kreisfrei mit $n - 1$ Kanten.

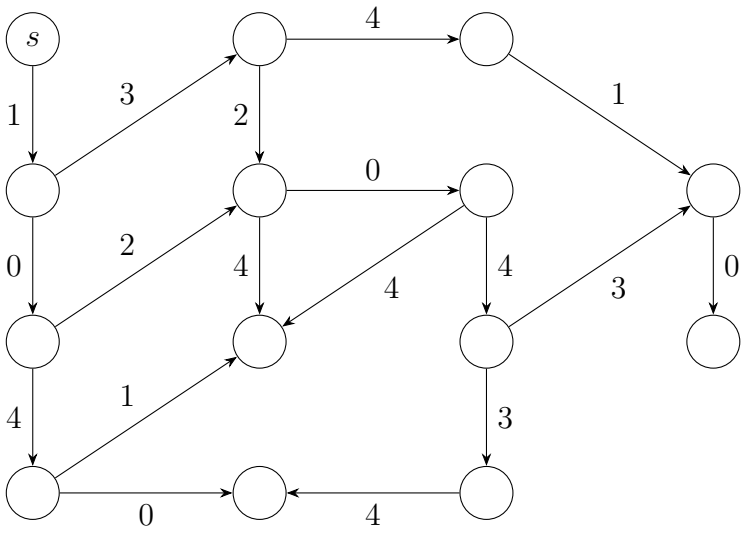
(iii) _____

_____

(iv) _____

_____

(c) Bestimmen Sie in folgendem Graphen die kürzesten Wege von s zu jedem anderen Knoten, indem Sie alle Kanten des Baums der kürzesten Wege in dem Graphen markieren. Schreiben Sie außerdem in jeden Knoten ungleich s die Nummer, als wievielter der Knoten aus der Prioritätswarteschlange in Dijkstras Algorithmus extrahiert wird.



2. Aufgabe (2 + 4 + 3 Punkte)

(a) Sei $(E, \mathcal{I})$ ein Unabhängigkeitssystem (über der endlichen Grundmenge E). Geben Sie eine Bedingung an, so dass $(E, \mathcal{I})$ zu einem Matroid wird.

(b) Betrachten Sie das Matroid $(E, \mathcal{I})$, wobei $E = \{a, b, c, d, e\}$ die Menge der Spalten der reellen Matrix

$$A = \begin{array}{c} \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{pmatrix} 2 & 3 & 0 & 0 & 3 \\ 1 & 1 & 2 & -1 & 3 \\ 0 & 5 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

bezeichne und $\mathcal{I}$ die Menge der linear unabhängigen Teilmengen von E ist. Geben Sie sämtliche Basen und Kreise des Matroids $(E, \mathcal{I})$ an.

(c) Gegeben sei ein Matroid $(E, \mathcal{I})$ mit $E = \{a, b, c, d\}$ und Basen

$$B \;=\; \{\, \{a, b, c\}, \{a, b, d\}, \{a, c, d\} \,\}.$$

Zeichnen Sie einen Graphen $G = (V, E)$ mit Kantenlabels E, so dass $(E, \mathcal{I})$ das graphische Matroid von G ist.

3. Aufgabe $(4 + 3 + 3 + 3$ Punkte)

(a) Geben Sie für zwei vergleichsbasierte Sortierverfahren (außer Quicksort und Merge-Sort) die Laufzeit im schlechtesten Fall in Θ-Notation an.

Verfahren	Laufzeit im Worst-Case

(b) Führen Sie Merge-Sort durch, um folgende Liste aufsteigend zu sortieren:

$$[3, 5, 4, 1, 6, 2, 9, 7].$$

Geben Sie hierbei Zwischenergebnisse an und achten Sie insbesondere darauf, dass die Abarbeitungsreihenfolge der Operationen ersichtlich ist.

(c) Formulieren Sie den Aufteilungs-Beschleunigungs-Satz und erklären Sie kurz, wie sich daraus die Laufzeit von Merge-Sort in $\mathcal{O}$-Notation bestimmen lässt.

(d) Eine Liste heißt *sprunghaft*, wenn jedes Element entweder größer als alle vorhergehenden oder kleiner als alle vorhergehenden Elemente ist. Zeigen Sie, dass die Worst-Case Laufzeit jedes vergleichsbasierten Verfahrens, das eine gegebene Liste mit paarweise verschiedenen Zahlen so umordnet, dass sie sprunghaft ist, in $\Omega(n \log n)$ ist.

4. Aufgabe $(5 + 3 + 3$ Punkte)

In den in dieser Aufgabe betrachteten Codes sind alle Codewörter 0-1-Strings.

(a) (i) Wann heißt ein Code eindeutig dekodierbar?

(ii) Geben Sie ein Beispiel für einen Code, bei dem alle Zeichen durch unterschiedliche Codewörter kodiert werden, der aber nicht eindeutig dekodierbar ist.

(iii) Wann heißt ein Präfixcode optimal?

(b) Sei C ein Alphabet. Für einen Präfixcode T für C sei $M(T)$ die maximale Länge eines Codeworts in T. Zeigen Sie: Es gibt einen Blockcode T_B für C mit $M(T_B) \leq M(T)$ für alle Präfixcodes T für C.

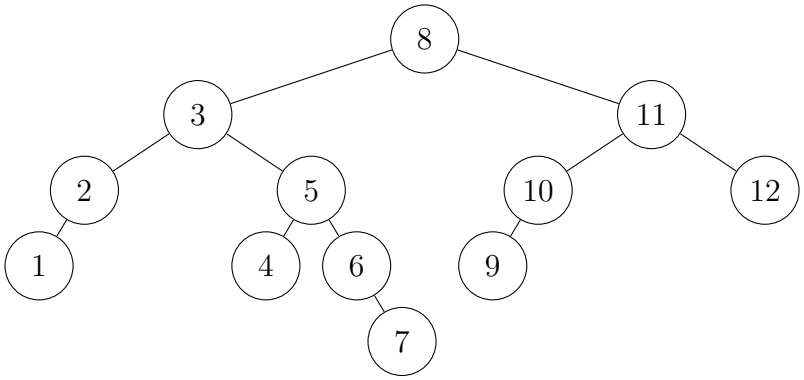
(c) Geben Sie den Huffman-Baum an, der einen optimalen Präfixcode für den folgenden Text (einschließlich der Leerzeichen) liefert. Nicht im Text vorkommende Zeichen müssen nicht kodiert werden. Notieren Sie die Häufigkeiten, die bei der Erstellung des Baums auftreten, an den entsprechenden Knoten (Blätter und innere Knoten).

`er␣isst␣seinen␣rest␣eis`

5. Aufgabe (2 + 4 + 4 Punkte)

(a) Geben Sie die Worst-Case-Laufzeiten der Operation `delete` für einen AVL-Baum und für einen Splay-Baum mit n Knoten in $\mathcal{O}$-Notation an.

(b) Aus folgendem AVL-Baum wird der Schlüssel 12 gelöscht. Um die AVL-Baum-Eigenschaft wiederherzustellen müssen diverse Rotationen durchgeführt werden. Geben Sie den Zustand nach jeder durchgeführten Rotation und die Bezeichnung der jeweiligen Rotationen an. Gegebenenfalls auftretende Doppelrotationen sollen als Einzelrotationen geschrieben werden.



(c) Erläutern Sie, wie das Prinzip der optimalen Substruktur in den Algorithmus zur Bestimmung eines optimalen statischen Suchbaums eingeht. Veranschaulichen Sie kurz das Prinzip der optimalen Substruktur an einem Beispiel mit 4 Knoten.

6. Aufgabe (4 + 2 + 4 Punkte)

Gegeben sei die folgende deterministische Turingmaschine mit Startzustand q_0, Zustandsmenge $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$, Eingabealphabet $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, Bandalphabet $\Gamma = \Sigma \cup \{\#, B\}$ und Menge akzeptierender Zustände $F = \{q_6\}$:

δ	a	b	c	#	B
q_0	$(q_1, \#, \mathtt{L})$	$(q_2, \#, \mathtt{L})$	$(q_3, \#, \mathtt{L})$	$(q_0, \#, \mathtt{R})$	$(q_5, B, \mathtt{L})$
q_1	$(q_1, \mathrm{a}, \mathtt{L})$	$(q_1, \mathrm{b}, \mathtt{L})$	$(q_1, \mathrm{c}, \mathtt{L})$	$(q_1, \#, \mathtt{L})$	$(q_4, \mathrm{a}, \mathtt{R})$
q_2	$(q_2, \mathrm{a}, \mathtt{L})$	$(q_2, \mathrm{b}, \mathtt{L})$	$(q_2, \mathrm{c}, \mathtt{L})$	$(q_2, \#, \mathtt{L})$	$(q_4, \mathrm{b}, \mathtt{R})$
q_3	$(q_3, \mathrm{a}, \mathtt{L})$	$(q_3, \mathrm{b}, \mathtt{L})$	$(q_3, \mathrm{c}, \mathtt{L})$	$(q_3, \#, \mathtt{L})$	$(q_4, \mathrm{c}, \mathtt{R})$
q_4	$(q_4, \mathrm{a}, \mathtt{R})$	$(q_4, \mathrm{b}, \mathtt{R})$	$(q_4, \mathrm{c}, \mathtt{R})$	$(q_0, \#, \mathtt{R})$	—
q_5	$(q_5, \mathrm{a}, \mathtt{L})$	$(q_5, \mathrm{b}, \mathtt{L})$	$(q_5, \mathrm{c}, \mathtt{L})$	$(q_5, B, \mathtt{L})$	$(q_6, B, \mathtt{R})$
q_6	—	—	—	—	—

Hier bedeutet „—“ in Zeile q und Spalte σ den Eintrag $(q, \sigma, \mathtt{N})$.

(a) Vervollständigen Sie die folgende Konfigurationsfolge:

$$
\begin{array}{lllllllll}
 & Bq_0\mathrm{cab} & \vdash & Bq_3B\#\mathrm{ab} & \vdash & \mathrm{c}q_4\#\mathrm{ab} & \vdash & \mathrm{c}\#q_0\mathrm{ab} \\
\vdash & & \vdash & & \vdash & & \vdash & \\
\vdash & \mathrm{ac}q_4\#\#\mathrm{b} & \vdash & \mathrm{ac}\#q_0\#\mathrm{b} & \vdash & \mathrm{ac}\#\#q_0\mathrm{b} & \vdash & \mathrm{ac}\#q_2\#\# \\
\vdash & \mathrm{ac}q_2\#\#\# & \vdash & \mathrm{a}q_2\mathrm{c}\#\#\# & \vdash & Bq_2\mathrm{ac}\#\#\# & \vdash & \\
\vdash & \mathrm{b}q_4\mathrm{ac}\#\#\# & \vdash & \mathrm{ba}q_4\mathrm{c}\#\#\# & \vdash & \mathrm{bac}q_4\#\#\# & \vdash & \\
\vdash & & \vdash & & \vdash & \mathrm{bac}\#\#q_5\# & \vdash & \mathrm{bac}\#q_5\# \\
\vdash & \mathrm{bac}q_5\# & \vdash & \mathrm{ba}q_5\mathrm{c} & \vdash & \mathrm{b}q_5\mathrm{ac} & \vdash & Bq_5\mathrm{bac} \\
\vdash & Bq_5B\mathrm{bac} & \vdash & Bq_6\mathrm{bac} & & & &
\end{array}
$$

(b) Die obige Turingmaschine liefert für $w \in \Sigma^*$ nach endlich vielen Schritten einen Output $f(w) \in \Sigma^*$. Beschreiben Sie die so definierte Funktion $f\colon \Sigma^* \to \Sigma^*$.

(c) Bestimmen Sie (mit kurzer Begründung) die exakte Anzahl der Konfigurationsübergänge der Maschine in Abhängigkeit von der Eingabelänge n.

Hinweis: Zählen Sie, wie oft die Turingmaschine die Laufrichtung ändert und wie viele Schritte sie jeweils in eine Richtung läuft.

7. Aufgabe (6 Punkte)

Bei den folgenden Teilaufgaben (a)–(f) ist jeweils genau eine der zur Auswahl stehenden Antworten zutreffend. Kennzeichnen Sie die jeweils korrekte Antwort durch das Setzen genau eines Kreuzes pro Teilaufgabe.

(a) Seien $f, g, h \colon \mathbb{N} \to \mathbb{N}$. Welche der folgenden Aussagen ist **wahr**?

- ☐ Ist $f \in \mathcal{O}(g)$ und $g \in \Omega(h)$, so ist $f \in \Theta(h)$.
- ☐ Ist $f \in \mathcal{O}(g)$ und $g \in \mathcal{O}(f)$, so ist $f = g$.
- ☐ Ist $f \in \mathcal{O}(g)$, so gibt es ein $N \in \mathbb{N}$ mit der Eigenschaft, dass es für alle $n \geq N$ ein $\alpha > 0$ gibt mit $f(n) \leq \alpha g(n)$.
- ☐ Ist $f \in \Omega(g)$, so gibt es für alle $\alpha > 0$ ein $N \in \mathbb{N}$, so dass für alle $n \geq N$ die Ungleichung $f(n) \geq \alpha g(n)$ gilt.

(b) Was ist die Binärdarstellung von $(43)_5$ (Darstellung zur Basis 5)?

☐ 101011 ☐ 11011 ☐ 111011 ☐ 10111

(c) Welche der folgenden Ausdrücke liefert in Python `True`?

- ☐ 0.1+(0.2+0.3)==(0.1+0.2)+0.3
- ☐ 0.1+(0.2+0.3)==(0.2+0.3)+0.1
- ☐ 100*(0.1+0.2)==100*0.1+100*0.2
- ☐ round(20.5)==21

(d) Sei T ein Baum mit n Knoten. Wie viele verschiedene Arboreszenzen mit zugrunde liegendem ungerichteten Graphen T gibt es (wobei jede Kante mit dem Paar (v, w) bestehend aus Start- und Endknoten identifiziert wird)?

☐ 1 ☐ 2 ☐ $n - 1$ ☐ n

(e) Welche der folgenden Aussagen ist **falsch**?

- ☐ Jede abzählbare Sprache ist rekursiv aufzählbar (semi-entscheidbar).
- ☐ Die Menge aller Turingmaschinen ist abzählbar.
- ☐ Die universelle Sprache ist rekursiv aufzählbar (semi-entscheidbar).
- ☐ Jede unentscheidbare Sprache ist abzählbar.

(f) Zum Speichern von drei Schlüsseln wird eine Hashtabelle der Länge 6 verwendet. Wir nehmen an, dass die Schlüssel $\boldsymbol{k}_1, \boldsymbol{k}_2$ und $\boldsymbol{k}_3$ zufällig gezogen werden und so verteilt sind, dass die Hashwerte $h(\boldsymbol{k}_1), h(\boldsymbol{k}_2), h(\boldsymbol{k}_3)$ unabhängig und gleichverteilt aus $\{0, 1, \ldots, 5\}$ sind. Diese drei Schlüssel werden nacheinander in die Hashtabelle eingefügt. Wie groß ist die Wahrscheinlichkeit, dass es dabei keine Kollision gibt?

☐ 1/36 ☐ 4/9 ☐ 5/9 ☐ 25/36

8. Aufgabe (7 + 3 + 2 Punkte)

(a) Zu einer Liste `liste` von paarweise verschiedenen positiven ganzen Zahlen kann man einen Graphen assoziieren, der die Elemente der Liste als Knoten hat. Zwei Zahlen in dem Graphen seien genau dann durch eine Kante verbunden, wenn die Differenz der Zahlen eine Primzahl ist. Das Gewicht einer Kante sei die kleinere der beiden Zahlen.

Schreiben Sie eine Funktion `mst(liste)` (sowie mögliche Hilfsfunktionen) in Python, die einen minimalen aufspannenden Baum des Graphen bestimmt, den man in oben beschriebender Weise mit der Liste `liste` assoziieren kann.

Sie dürfen im Folgenden annehmen, dass nur Instanzen auftreten, bei denen der Graph zusammenhängend ist.

Sie können hierbei die Klasse `Components` (siehe unten) und alle Python-Funktionen, die nicht importiert werden müssen, verwenden.

Ihr Algorithmus muss nicht optimale Laufzeit haben.

(b) Erläutern Sie die Laufzeit Ihrer Funktion `mst(liste)` in Abhängigkeit von den Eingabegrößen asymptotisch in $\mathcal{O}$-Notation. Bestimmen Sie dazu die Anzahlen der Aufrufe der Funktionen der Klasse `Components`. Bestimmen Sie ferner die Anzahl der zusätzlich benötigten elementaren Rechenoperationen. Beachten Sie, dass Python-Funktionen hierbei durchaus mehr als eine elementare Rechenoperation benötigen.

(c) Erläutern Sie kurz das algorithmische Prinzip, das hinter den Standard-Algorithmen zur Bestimmung eines minimalen Spannbaums steht. Welche Struktur ist durch dieses algorithmische Prinzip charakterisiert?

Gegeben sei die Klasse `Components` mit den folgenden Funktionen:

`__init__(mylist)` Initialisiert eine neue Instanz, die für jedes Element in `mylist` eine Zusammenhangskomponente erstellt.

`glue(i,j)` Vereinigt die Zusammenhangskomponenten, die `i` und `j` enthalten. Dabei sind `i` und `j` Elemente von `mylist`.

`find_component(i)` Der Rückgabewert ist für alle Elemente einer Zusammenhangskomponente gleich. Dabei ist `i` ein Element von `mylist`.