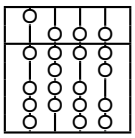
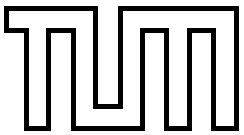


TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK



Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Christian Scheideler, S. Pott, H. Täubig

SS 2007
Midterm-Klausur
29.05.2007

Midterm-Klausur zu Grundlagen: Algorithmen und Datenstrukturen

Name	Vorname	Studiengang	Matrikelnummer
Hörsaal	Reihe	Sitzplatz	Unterschrift

Allgemeine Hinweise:

- **Bitte füllen Sie die oben angegeben Felder vollständig aus und unterschreiben Sie!**
- **Schreiben Sie nicht mit Bleistift oder in roter/grüner Farbe!**
- **Lösen Sie die Aufgaben auf dem freien Platz unter der Angabe (bzw. auf der Rückseite). Fordern Sie weiteres Papier von der Klausuraufsicht, falls Ihnen der Platz nicht ausreicht.**
- **Als Hilfsmittel ist ausschließlich ein handbeschriebenes DinA4-Blatt zugelassen. Bei Missachtung wird die gesamte Klausur mit null Punkten bewertet.**
- **Die Arbeitszeit beträgt 60 Minuten.**
- **Prüfen Sie, ob Sie alle 7 Seiten erhalten haben.**

Aufgabe 1 [5 Punkte]

Kreuzen Sie pro Teilaufgabe höchstens ein Kästchen an. Für ein falsches Kreuz gibt es einen halben Minuspunkt, für ein richtiges einen halben Pluspunkt. Wenn Sie kein Kreuz setzen, bekommen Sie auch keine Punkte. Eine negative Gesamtpunktzahl dieser Aufgabe wird zu 0 aufgerundet. Maximieren Sie Ihre Punktzahl!

a) $o(n) \subset \mathcal{O}(n^2)$ ☐ Richtig ☐ Falsch

b) $f(n) \in \Omega(g(n)) \Rightarrow f(n) \in \Theta(g(n))$ ☐ Richtig ☐ Falsch

c) $\mathcal{O}(g(n)) \cap \omega(g(n)) = \emptyset$ ☐ Richtig ☐ Falsch

d) Für jede positive Funktion $f(n)$ gilt: $g(n) + f(n) \in \mathcal{O}(g(n))$ ☐ Richtig ☐ Falsch

e) $\Theta(f(n)) = \{g(n) : \exists c > 0\ \exists n_0 \in \mathbb{N}_+\ \forall n \geq n_0 : g(n) = cf(n)\}$ ☐ Richtig ☐ Falsch

f) $T(\textbf{for } i := 0 \textbf{ to } 5 \textbf{ do } I) \in \mathcal{O}(T(I))$ ☐ Richtig ☐ Falsch

g) Sei $n = 3^k$ für $k \in \mathbb{N}_+$. Dann gilt
für $T(1) = 5$ und $T(n) = 3T(n/3) + n$: $\quad T(n) \in \mathcal{O}(n^2)$ ☐ Richtig ☐ Falsch

h) Für $n = 2^k$ mit $k \in \mathbb{N}_+$ gilt für
$S(1) = 2$ und $S(n) = 3S(n/2) + 3n$: $\quad S(n) \in \Theta(n^{\log_2 3})$ ☐ Richtig ☐ Falsch

i) Es lässt sich ein dynamisches Array konstruieren, bei dem die Operationen *pushBack* und *popBack* im worst case konstant sind. ☐ Richtig ☐ Falsch

j) Für alle $x \in \mathbb{R}_+$ gilt: $\left\lceil \sqrt{\lceil x \rceil} \right\rceil = \lceil \sqrt{x} \rceil$ ☐ Richtig ☐ Falsch

Aufgabe 2 [11 Punkte] **O-Notation**

a) Geben Sie den Wachstumsvergleich ($o(\cdot)$, $\omega(\cdot)$ oder $\Theta(\cdot)$) für die Funktionen $f(n) = \sqrt{n}$ und $g(n) = \log(n)$ an und begründen Sie Ihre Aussage!

b) Zeigen Sie: $n^3 \log(n^2) \in \mathcal{O}(n^{3\log(n^2)})$

c) Zeigen Sie: Für zwei positive Funktionen $f : \mathbb{N}_+ \to \mathbb{N}_+$ und $g : \mathbb{N}_+ \to \mathbb{N}_+$ gilt:

$$f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$$

d) Zeigen Sie: $n^{3\log(n^2)} \in \Omega(n^3 \log(n^2))$

Aufgabe 3 [12 Punkte] **Master-Theorem**

Geben Sie für folgende Rekurrenzen an, ob sie sich für ein $b \in \mathbb{R}$ mit $n = b^k$ ($k \in \mathbb{N}_0$) mit dem Master-Theorem aus der Vorlesung (ohne aufwändige Umformungen) lösen lassen. Wenn ja, geben Sie die entsprechende Laufzeitabschätzung an. Wenn nein, begründen Sie Ihre Antwort.

a) $A(1) = \pi$ und $A(n) = 3A(n/3) + 4n$ für $n > 1$

b) $B(n) = \begin{cases} n & n = 1 \\ 2B(n/2) + B(n/2) + n & n > 1 \end{cases}$

c) $C(1) = 2$ und $C(n) = C(n-2) + n + 2$ für $n > 1$

d) $D(1) = 27$ und $D(n) = 3n + D(2n/3)$ für $n > 1$

e) $E(1) = 4$ und $E(n) = 4 \cdot 2^n E(n/5) + n$ für $n > 1$

f) $F(1) = 1$ und $F(n) = cF(n/c) + cn^2$ für $n > 1$ und ein $c \in \mathbb{N}_+$

Aufgabe 4 [8 Punkte] **Master-Theorem**

Betrachten Sie den folgenden Algorithmus, der für eine beliebige Sequenz $a = \langle a_0, \ldots, a_{n-1} \rangle$ (mit $a_i \in \mathbb{N}_+$ für alle i) die Summe $\sum_{i=0}^{n-1} a_i$ ihrer Elemente berechnet. Dabei wurde für einen Teil der Zeilen der Zeitaufwand angegeben ($c, c' \in \mathbb{N}_+$).

Zeile		(Zeitaufwand)
1	**Procedure** $summe(a = \langle a_0, \ldots, a_{n-1} \rangle$: Sequenz)	
2	if (Länge von $a == 1$)	
3	return a_0	(1)
4	else	
5	$m := \lceil n/2 \rceil$	($c \cdot n$)
6	$b := \langle a_0, \ldots, a_{m-1} \rangle$	($c' \cdot m$)
7	$b' := \langle a_m, \ldots, a_{n-1} \rangle$	($c' \cdot (n - m)$)
8	return $summe(b) + summe(b')$	

a) Stellen Sie eine Rekurrenz $T(n)$ zur Laufzeitanalyse dieser Prozedur (in Abhängigkeit von der Länge n der Eingabefolge) auf. Dabei dürfen Sie den Zeitaufwand für den Vergleich in Zeile 2 und den Aufwand für die reine Addition in der letzten Zeile (8) vernachlässigen (nicht den Aufwand für die rekursiven Aufrufe!). Die Konstanten c und c' müssen Sie nicht näher bestimmen. Geben Sie alle Zwischenschritte an.

b) Lösen Sie die Rekurrenz mit Hilfe des (einfachen) Master-Theorems aus der Vorlesung für $n = 2^k$ ($k \in \mathbb{N}_0$). Geben Sie also eine Funktion $f(n)$ an, so dass $T(n) \in \Theta(f(n))$ gilt.

Aufgabe 5 [14 Punkte] **Amortisierte Analyse**

Auf der folgenden Seite sind zwei Datenstrukturen (**Queue1** und **Queue2**) angegeben. Beide stellen die Operationen *pushFirst* und *popLast* auf einer Sequenz von Elementen $s = \langle e_0, \ldots, e_n \rangle$ zur Verfügung. Begonnen wird mit einer leeren Sequenz $s_0 = \langle\rangle$. Für eine Sequenz $s = \langle e_0, \ldots, e_n \rangle$ gilt:

$$s \xrightarrow{pushFirst(e)} s' \quad \text{mit } s' = \langle e, e_0, \ldots, e_n \rangle$$

$$s \xrightarrow{popLast} s' \quad \text{mit } s' = \langle e_0, \ldots, e_{n-1} \rangle$$

In **Queue1** wird die Sequenz durch eine (einfach verkettete) Liste, in **Queue2** durch zwei (einfach verkettete) Listen repräsentiert.
Begründen Sie jeweils Ihre Antwort.

a) Geben Sie für die Operationen $pushFirst_1$, $popLast_1$, $pushFirst_2$, $popLast_2$ und $revers$ an, ob Sie im schlechtesten Fall konstante, logarithmische, lineare oder quadratische Laufzeit haben. (Dabei dürfen Sie für die verwendeten Listenoperationen (*insertBefore(e)*, *isEmpty*, *first*, *next*, *popFront* und die Abfrage „e hat Nachfolger“) konstante Laufzeit voraussetzen.)

b) Geben Sie zur ersten Datenstruktur **Queue1** eine Folge von m Operationen ($pushFirst_1$ und $popLast_1$) an, sodass die Laufzeit in $\Theta(m^2)$ liegt.

c) Geben Sie den Aufwand in Θ-Notation für Ihre Folge von m Operationen aus Aufgabenteil b) zur zweiten Datenstruktur **Queue2** an (wobei $pushFirst_1$ durch $pushFirst_2$ und $popLast_1$ durch $popLast_2$ ersetzt wird).

d) Zeigen Sie, dass der amortisierte Aufwand von $popLast_2$ (in **Queue2**) konstant ist.

Zusatz zu Aufgabe 5

Class *Queue1* **of** *Element*

l : **List** $\langle\rangle$ **of** *Element*

Procedure $pushFirst_1(e:\ Element)$

$l.insertBefore(e)$ // Fügt e an erster Stelle in die Liste ein.

Procedure $popLast_1$ // Löscht das letzte Element aus der Liste.

assert $(\neg l.isEmpty)$
$e := l.first$ // Erstes Element der Liste.
while e hat Nachfolger // Durchläuft alle Elemente
$e := e.next$
dispose e // Löscht auch alle Zeiger auf e

Class *Queue2* **of** *Element*

l_1 : **List** $\langle\rangle$ **of** *Element*

l_2 : **List** $\langle\rangle$ **of** *Element*

Procedure $pushFirst_2(e:\ Element)$

$l_1.insertBefore(e)$ // Fügt e an erster Stelle in die Liste l_1 ein.

Procedure $popLast_2$ // Löscht das erste Element aus der Liste l_2.

if $l_2.isEmpty$ **then** *revers*
if $\neg l_2.isEmpty$ **then** $l_2.popFront$

// *revers* verschiebt alle Elemente aus l_1 in l_2 und kehrt dabei die Reihenfolge um

Procedure *revers*

while $(\neg l_1.isEmpty)$
$e := l_1.first$
$l_2.insertBefore(e)$
$l_1.popFront$

Beispiel:

$$pushFirst_i(1);\ pushFirst_i(2);\ pushFirst_i(3);\ popLast_i;\ pushFirst_i(4);$$

Diese Folge von Operationen ergibt:

- für $i = 1$, also in **Queue1**: $l = \langle 4, 3, 2\rangle$
- für $i = 2$, also in **Queue2**: $l_1 = \langle 4\rangle$ und $l_2 = \langle 2, 3\rangle$