Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Volker Heun

WS 1999/2000
Semestralklausur 2
29. März 2000

Grundlegende Algorithmen

Es sind insgesamt **40** Punkte zu erreichen.
Zum Bestehen sind **mindestens 17** Punkte erforderlich.
Bitte schreiben Sie **nicht** in *roter* oder *grüner* Farbe.

Die reine Arbeitszeit beträgt **150** Minuten.

Viel Glück und Erfolg!

Aufgabe 1 (4 Punkte)

Beweisen oder widerlegen Sie:

a) $2^{n+2^{256}} = O(2^n)$

b) $5^n \cdot 2^n = \Theta(7^n)$

c) $4^{\log(n)} = o(n^2)$

d) $\sum_{i=1}^{n} i \cdot \log(i) = \Theta(n^2 \log(n))$

Aufgabe 2 (4 Punkte)

Gegeben sei die Folge $a = (3, 9, 1, 7, 2, 8, 5, 4)$. Zeigen Sie, wie

a) der Heap bei Heapsort für die Folge a konstruiert wird und

b) wie dann Heapsort mit diesem Heap arbeitet.

Verwenden Sie sowohl zum Aufbau des Heaps als auch zum eigentlichen Sortieren die Prozedur `reheap` des Standard-Heapsorts.

Geben Sie dabei jeweils *alle* Zwischenschritte an.

Aufgabe 3 (5 Punkte)

Sei $A = (a_{i,j})$ eine $n \times n$-Matrix. Es wird zuerst jede Zeile von A aufsteigend sortiert, dann wird jede Spalte von A aufsteigend sortiert. Zeigen Sie, daß dann jede Zeile von A immer noch aufsteigend sortiert ist.

Aufgabe 4 (5 Punkte)

Gegeben sei eine sortierte Liste mit n Schlüsseln und eine unsortierte Liste mit m Schlüsseln. Wieviele Vergleiche werden von *jedem* vergleichsbasierten Sortieralgorithmus *im schlimmsten Fall* mindestens benötigt, um diese $n + m$ Schlüssel zu sortieren.

Nehmen Sie dabei an, daß die Schlüssel paarweise verschieden sind.

Aufgabe 5 (4 Punkte)

Eine Folge $(a_1, \ldots, a_n)$ ganzer Zahlen hat einen *Fixpunkt*, wenn es ein $i \in [1 : n]$ gibt, so daß $a_i = i$ gilt.

Geben Sie einen Divide-and-Conquer-Algorithmus an (entweder in Worten oder als Pseudo-Code), der feststellt, ob eine gegebene sortierte Folge $(x_1, \ldots, x_n)$ paarweiser verschiedener ganzer Zahlen einen Fixpunkt hat oder nicht.

Achtung: Ihr Algorithmus muß mit $o(n)$ Vergleichen auskommen!

Analysieren Sie Ihren Algorithmus bezüglich der Anzahl der ausgeführten Vergleiche auf den Elementen x_i.

— **Bitte wenden!** —

Aufgabe 6 (5 Punkte)

Wir betrachten im folgenden das Universum $U = [0:35]$. Gegeben sei die folgende Hashfunktion $h(k) = k \bmod 6$ für eine Hashtabelle der Größe 6. Wir betrachten die folgende erweiterte Hashfunktion, die das sogenannte kubische Sondieren zur Kollisionsauflösung verwendet:

$$h : U \times [0:5] \to [0:5]: \ h(k,i) = (h(k) + i^3) \bmod 6.$$

Die folgenden Fragen sollen nur für den Spezialfall beantwortet werden, in dem die Hashtabelle die Größe 6 hat.

Werden mit Hilfe des kubische Sondierens in jedem Fall leere Hashpositionen gefunden?

Wie sieht es mit der primären und sekundären Häufung bei Verwendung des kubischen Sondierens aus?

Begründen Sie Ihre Antworten.

Hinweis: Es gilt $x^3 \bmod n = (x \bmod n)^3 \bmod n$ und:

x	0	1	2	3	4	5
x^3	0	1	8	27	64	125

Aufgabe 7 (4 Punkte)

Ein Knoten x eines binären Baumes heißt *quasi-höhenbalanciert*, wenn sich die Höhen seiner beiden Teilbäume um maximal 2 unterscheiden. Ein binärer Baum heißt *quasi-höhenbalanciert*, wenn jeder seiner Knoten quasi-höhenbalanciert ist.

Zeigen Sie, daß ein quasi-höhenbalancierter Baum der Höhe $h \geq 0$ mindestens $(\frac{4}{3})^h$ Knoten besitzt.

Erinnerung: Die Höhe eines Baumes mit genau einem Knoten ist 0.

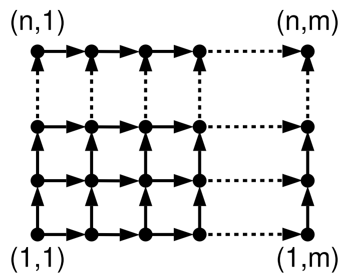
Aufgabe 8 (4 Punkte)

Fügen Sie die folgenden Schlüssel $1, 3, 4, 2, 5, 6, 8, 7, 9$ in dieser Reihenfolge in einen (2,3)-Baum ein. Löschen Sie anschließend die Schlüssel 4 und 6 in dieser Reihenfolge.

Geben Sie dabei *alle* Zwischenschritte sowohl vor als auch nach Verschmelzungen und Aufteilungen an.

Aufgabe 9 (5 Punkte)

Betrachten Sie ein Gitter mit n Zeilen und m Spalten.



Angenommen Sie befinden sich zu Beginn am Punkt $(1,1)$ und wollen zum Punkt (n,m). An jedem Punkt des Gitters dürfen Sie entweder nach rechts oder nach oben gehen. Weiterhin hat jede Kante von (i,j) nach (i',j') mit entweder $i' = i+1$ und $j' = j$ oder $i' = i$ und $j' = j+1$ ein Gewicht $g(i,j,i',j') \in \mathbb{R}_+$. Die *gewichtete Länge* eines Weges ist die Summe der Gewichte seiner Kanten.

Geben Sie eine Rekursionsgleichung für die kürzeste gewichtete Länge $L(i,j)$ eines Weges vom Punkt $(1,1)$ zum Punkt (i,j) an und begründen Sie diese.

Geben Sie einen Algorithmus zur Bestimmung der kürzesten gewichteten Länge $L(n,m)$ eines Weges von $(1,1)$ nach (n,m) mittels dynamischer Programmierung an.