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

WS 1999/2000
Semestralklausur
1. 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) $n^3 + 2^{256} \cdot n^2 = O(n^3)$

b) $\dfrac{9n^2 - 4n}{3n - 2\sqrt{n}} = \Omega(n)$

c) $3^{3n} = O(3^n)$

d) $n \log(n) = o(\log(n^7))$

Aufgabe 2 (4 Punkte)

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

a) der rekursive Mergesort und

b) Quicksort (wobei jeweils das rechteste Element als Pivot gewählt wird)

auf a operieren. Geben Sie dabei jeweils *alle* Zwischenschritte an.

Aufgabe 3 (5 Punkte)

Betrachten Sie den folgenden Algorithmus SORT, der auf $(x_0, \ldots, x_{n-1})$ wie folgt arbeitet: In der i-ten Phase ($i \in [1 : n-1]$) werden nacheinander für $j \in [1 : n-i]$ die Elemente an Position $j-1$ und j miteinander verglichen und anschließend vertauscht, wenn $x_{j-1} > x_j$. Im folgenden ist der Algorithmus noch einmal in Pseudo-Code angegeben:

```
SORT(int x[], int n)
{
    for (int i = 1; i < n; i++)
        for (int j = 1; j ≤ n − i; j++)
            if (x[j − 1] > x[j])
            {   /* Vertausche x_{j−1} mit x_j */
                int y = x[j − 1];
                x[j − 1] = x[j];
                x[j] = y;
            }
}
```

Begründen Sie in Worten, daß dieser Algorithmus die Folge korrekt sortiert, und bestimmen Sie die maximale Anzahl ausgeführter Vergleiche.

Aufgabe 4 (5 Punkte)

Eine Folge $(a_1, \ldots, a_n) \in \mathbb{Z}^n$ paarweiser verschiedener ganzer Zahlen heißt *alternierend*, wenn für alle $i \in [1 : \lfloor n/2 \rfloor]$ gilt, daß $a_{2i-1} < a_{2i}$ und $a_{2i} > a_{2i+1}$, also

$$a_1 \; < \; a_2 \; > \; a_3 \; < \; a_4 \; > \; a_5 \; < \; a_6 \; > \; \cdots$$

Beispiel: Die Folge $(1, 5, 3, 7, 2, 9, 4, 8)$ ist alternierend, die Folge $(1, 5, 3, 7, 2, 4, 8, 6)$ jedoch nicht.

Geben Sie einen Divide-and-Conquer-Algorithmus an (entweder in Worten oder als Pseudo-Code), der eine Folge $(x_1, \ldots, x_n)$ paarweiser verschiedener Zahlen so permutiert, daß sie eine alternierende Folge bilden. Sie dürfen hierbei annehmen, daß n eine Zweierpotenz ist, d.h. $n = 2^\ell$ für ein $\ell \in \mathbb{N}$.

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

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

— **Bitte wenden!** —

Aufgabe 5 (5 Punkte)

Sei $\mathcal{M}$ ein Algorithmus, der eine n-elementige Menge M nach dem Median $x \in M$ in zwei Teilmengen $\mathcal{M}(M) = (K, G)$ partitioniert, wobei

$$K = \{m \in M \,:\, m < x\} \qquad \text{sowie} \qquad G = \{m \in M \,:\, m \geq x\}.$$

Beachten Sie, daß die Elemente der Menge M paarweise verschieden sind. Weiterhin benötigt dieser Algorithmus $\mathcal{M}$ maximal $c \cdot n$ viele Vergleiche, wobei $0 < c \in \mathbb{R}$ eine Konstante ist.

Konstruieren Sie einen Algorithmus $\mathcal{S}$, der aus einer Menge mit n paarweise verschiedenen Elementen das Element mit Rang k mit $O(n)$ Vergleichen bestimmt. Dabei dürfen (sollen) Sie den Algorithmus $\mathcal{M}$ als Unterprogramm verwenden. Der Algorithmus $\mathcal{S}$ darf dabei selbst keine Vergleiche auf Elementen aus M vornehmen. Es sind nur Vergleiche durch den Aufruf von $\mathcal{M}$ erlaubt. Der Algorithmus soll dabei in Pseudo-Code angegeben werden. Sie dürfen hierbei annehmen, daß n eine Zweierpotenz ist, d.h. $n = 2^\ell$ für ein $\ell \in \mathbb{N}$.

Bestimmen Sie für Ihren Algorithmus $\mathcal{S}$ eine möglichst gute obere Schranke für die maximale Anzahl von ausgeführten Vergleichen.

Erinnerung: Ein Element x einer Menge M hat *Rang* r, wenn es genau r Elemente in M gibt, die kleiner gleich x sind, d.h. $|\{m \in M \,:\, m \leq x\}| = r$. Ein Element einer n-elementigen Menge heißt *Median*, wenn es Rang $\lceil \frac{n+1}{2} \rceil$ hat.

Hinweis: Vielleicht benötigen Sie die folgende Formel zur Abschätzung der Laufzeit Ihres Algorithmus $\mathcal{S}$:

$$\forall\, x \in [0,1) : \ \forall\, k, m \in \mathbb{N} : \quad \sum_{i=k}^{m} x^i = \frac{x^k - x^{m+1}}{1-x} \leq \frac{x^k}{1-x}.$$

Aufgabe 6 (4 Punkte)

Gegeben sei die folgende Hashfunktion $h(x) = x^3 \bmod 5$. Wir betrachten im folgenden das Universum $U = [0 : 99]$, wobei jeder Schlüssel aus U mit Wahrscheinlichkeit $\frac{1}{100}$ auftritt.

Ist h für U eine gut gewählte Hashfunktion? Ist h für U eine ideale Hashfunktion? Begründen Sie Ihre Antworten.

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

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

Aufgabe 7 (4 Punkte)

Fügen Sie die folgenden Schlüssel $1, 2, 3, 5, 6, 4$ in dieser Reihenfolge in einen AVL-Baum ein und löschen Sie zum Schluß den Schlüssel 5.

Geben Sie dabei *alle* Zwischenschritte sowohl vor als auch nach Rotationen an.

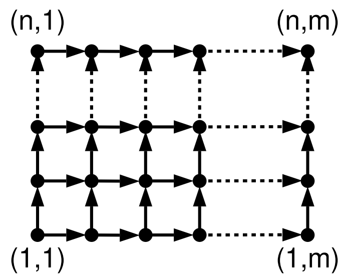
Aufgabe 8 (4 Punkte)

Gegeben seien zwei Zeichenreihen s und t, wobei s bzw. t die Länge m bzw. n haben. Gesucht ist die Länge des längsten Suffixes von s, der in t enthalten ist.

Geben Sie in Worten einen Algorithmus an, der dieses Problem löst und dabei mit maximal $O(m + n)$ Vergleichen auskommt. Begründen Sie in Worten die Korrektheit Ihres Algorithmus und Ihre Abschätzung über die Anzahl Vergleiche.

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. Für die Anzahl verschiedener Wege vom Punkt $(1, 1)$ zum Punkt (i, j) gilt dann die folgende Rekursionsgleichung:

$$A(i,j) = \begin{cases} A(i-1,j) + A(i,j-1) & \text{falls } i > 1 \text{ und } j > 1 \\ 1 & \text{falls } i = 1 \text{ oder } j = 1 \end{cases} .$$

Begründen Sie in Worten die Korrektheit dieser Rekursionsgleichung und geben Sie einen Algorithmus zur Bestimmung der Anzahl $A(n, m)$ mittels dynamischer Programmierung an.