

## Klausur: Algorithmen und Datenstrukturen

Wintersemester 2004/05

Studiengänge BNC, BWM, GIn, EC, Mm

Name, Vorname:

Matrikelnummer:

Aufgabe	1	2	3	4	Σ
Punkte					

**Aufgabe 1:**

(2 + 2 + 2 Punkte)

Geben Sie möglichst scharfe asymptotische obere und untere Schranken für die unten definierte Funktion  $T(n)$  an. Dabei sei  $T(n)$  konstant für  $n \leq 3$ . Begründen Sie Ihre Lösungen.

(a)  $T(n) = 2T(n/2) + 2^{\log_2 n}$ .

(b)  $T(n) = 9T(n/3) + \log_2(n!)$ .

(c)  $T(n) = T(n/2) + n \log_2 n$ .

(d) **Zusatzaufgabe:**  $T(n) = T(n/2) + \log_2 n$ . (4 Punkte)

**Aufgabe 2:**

(2 + 4 + 2 Punkte)

Eine Hash-Tabelle mit den Positionen  $0, 1, \dots, 10$  und der Hash-Funktion  $g(k) = k \bmod 11$  sei wie folgt belegt:

0	1	2	3	4	5	6	7	8	9	10
65	24	51	118	58	103	79	70	85	19	174

Als Strategie zur Kollisionsauflösung werde quadratisches Sondieren mit den Parametern  $c_1 = c_2 = 1/2$  verwendet, d.h. die Hash-Funktion lautet

$$h(k, i) = [g(k) + \frac{1}{2}(i + i^2)] \bmod 11, \quad i = 0, 1, \dots, 10.$$

Die Hash-Tabelle sei am Anfang leer.

- (a) Bestimmen Sie die Sondierungsfolge jedes der elf Schlüssel in der Tabelle bis zu der Position, die der Schlüssel in der Tabelle einnimmt.
- (b) Geben Sie eine mögliche Reihenfolge an, in der die Schlüssel in die Tabelle eingefügt worden sein könnten. Geben Sie eine Begründung für die Reihenfolge der ersten fünf Elemente Ihrer Einfügefølge an.

(c) Sind weitere Reihenfolgen möglich ?

**Aufgabe 3:**

(6 Punkte)

Im Abschnitt 4.3 der Vorlesung über randomisierte Algorithmen wurde der Algorithmus RANDOMIZE-IN-PLACE vorgestellt, welcher gleichverteilte zufällige Permutationen generiert.

RANDOMIZE-IN-PLACE( $A, n$ )

```
for  $i \leftarrow 1$  to  $n$ 
  do vertausche  $A[i] \leftrightarrow A[\text{RANDOM}(i, n)]$ 
```

Betrachten Sie folgende Modifikation dieses Algorithmus:

PERMUTE-WITH-ALL( $A, n$ )

```
for  $i \leftarrow 1$  to  $n$ 
  do vertausche  $A[i] \leftrightarrow A[\text{RANDOM}(1, n)]$ 
```

Zeigen Sie, dass der so modifizierte Algorithmus keine gleichverteilten Permutationen erzeugt, indem Sie für  $n = 3$  die Wahrscheinlichkeiten aller möglichen Permutationen ermitteln.

**Aufgabe 4:**

(3 + 5 Punkte)

In Kapitel 11 der Vorlesung über dynamisches Programmieren wurde als Beispiel die Berechnung von längsten gemeinsamen Teilfolgen zweier (endlicher) Folgen  $X_m$  bzw.  $Y_n$  der Längen  $m$  bzw.  $n$  betrachtet. Dabei wurde untenstehender Algorithmus LCS-LENGTH vorgestellt. Hierin wird eine Tabelle  $c[i, j]$  aufgestellt, in welcher das  $(i, j)$ -te Element die Länge einer längsten gemeinsamen Teilfolge der Präfixe  $X_i$  und  $Y_j$  von  $X_m$  bzw.  $Y_n$  angibt.

LCS-LENGTH( $X, Y$ )

```
1  $m \leftarrow \text{length}[X]$ 
2  $n \leftarrow \text{length}[Y]$ 
3 for  $i \leftarrow 1$  to  $m$ 
4   do  $c[i, 0] \leftarrow 0$ 
5 for  $j \leftarrow 1$  to  $n$ 
6   do  $c[0, j] \leftarrow 0$ 
7 for  $i \leftarrow 1$  to  $m$ 
8   do for  $j \leftarrow 1$  to  $n$ 
9     do if  $x_i = y_j$ 
10      then  $c[i, j] \leftarrow c[i - 1, j - 1] + 1$ ;  $b[i, j] \leftarrow \text{“}\nwarrow\text{”}$ 
11      else if  $c[i - 1, j] \geq c[i, j - 1]$ 
12        then  $c[i, j] \leftarrow c[i - 1, j]$ ;  $b[i, j] \leftarrow \text{“}\uparrow\text{”}$ 
13        else  $c[i, j] \leftarrow c[i, j - 1]$ ;  $b[i, j] \leftarrow \text{“}\leftarrow\text{”}$ 
14 return  $c, b$ 
```

(a) Zeigen Sie, wie man den Speicherbedarf dieses Algorithmus reduzieren kann auf  $2 \cdot \min(m, n)$  Einträge der  $c$ -Tabelle plus  $O(1)$ .

(b) Verbessern Sie dies weiter auf  $\min(m, n) + O(1)$ .