Universität Karlsruhe
Fakultät für Informatik
ITI Wagner

1. Übungsblatt

Ausgabe: 27. Oktober 2005
Abgabe: 7. November, 14 Uhr im ITI Wagner (Informatik-Hauptgebäude, 3. Stock)

Die Bearbeitung in Zweiergruppen ist ausdrücklich erwünscht.

Problem 1: Average-case-Laufzeit vs. Worst-case-Laufzeit **

Angelehnt an die Technik mit der ein holländischer Gartenzwerg Blumentöpfe sortiert sei folgender Algorithmus gegeben:

Algorithmus 1 : Gnomesort

```
Eingabe : Unsortiertes Array von n verschiedenen Zahlen A = (A[1], ..., A[n])
Ausgabe : Sortiertes Array A der Zahlen
i ← 2
solange i ≤ n tue
    Wenn A[i − 1] ≤ A[i]
        i ← i + 1
    sonst
        Tausche A[i − 1] und A[i]
        i ← i − 1
        Wenn i = 1
            i ← 2
```

(a) Berechnen sie die Worst-case-Laufzeit von Algorithmus 1, zählen sie dabei nur die Anzahl der nötigen Vergleichsoperationen.

(b) Wie muss das Eingabearray geartet sein um von allen Permutationen die höchste Laufzeit zu erzielen?

(c) Ein Gartenzwerg möchte die Average-case-Laufzeit von Algorithmus 1 bestimmen. Er nimmt an, dass bei Gleichverteilung aller Eingabepermutaionen A jeder Tausch von zwei Elementen mit einer Wahrscheinlichkeit von $1/2$ stattfindet. Somit wandert ein Element mit $p = 1/2$ um eine Position nach links, mit $p = 1/4$ um zwei Positionen, usw. Zeigen Sie, dass unter dieser Annahme eine Average-case-Laufzeit von $\Theta(n)$ resultiert.

(d) Argumentieren sie theoretisch, warum das Resultat $\Theta(n)$ des Gartenzwergs nicht korrekt sein kann. Erklären Sie wo sein Fehler liegt.

(e) Helfen Sie dem Gartenzwerg. Korrigieren sie zunächst seine fehlerhafte Annahme. Berechnen sie die korrekte (scharfe) Average-case-Laufzeit von Algorithmus 1.

Problem 2: Rekursive Suche *

Entwickeln Sie einen rekursiven Algorithmus, der testet, ob eine gegebene Zahl in einem gegebenen sortierten Array von Zahlen vorkommt. Notieren Sie den Algorithmus in Pseudocode und bestimmen Sie seine asymptotische Laufzeit im schlechtesten Fall.

Problem 3: Gewichteter Median ***

Der *gewichtete Median* von n Elementen $x_1, \ldots, x_n$ mit Gewichten $w_1, \ldots, w_n \in \mathbb{R}^+$ ist das Element x_k, das die Ungleichungen

$$\sum_{x_i<x_k} w_i \leq \frac{1}{2}\sum_{i=1}^{n} w_i \qquad \text{und} \qquad \sum_{x_i>x_k} w_i \leq \frac{1}{2}\sum_{i=1}^{n} w_i$$

erfüllt. Entwickeln Sie analog zum Algorithmus SELECT aus der Vorlesung einen Algorithmus, der in linearer Zeit den gewichteten Median berechnet, und beweisen sie dessen Laufzeit.

Problem 4: Laufzeit rekursiver Funktionen I *

Wenden Sie den Aufteilungs-Beschleunigunssatz an, um eine asymptotisch scharfe Schranke für $T(n) = 4 \cdot T(n/2) + n^3$ zu bestimmen.

Problem 5: Laufzeit rekursiver Funktionen II ***

Geben Sie den größten ganzzahligen Wert a an, so dass ein Algorithmus mit der Laufzeit

$$T_B(n) = a \cdot T_B(n/4) + n^2$$

im Θ-Kalkül asymptotisch schneller ist, als ein Algorithmus mit der Laufzeit

$$T_A(n) = 7 \cdot T_A(n/2) + n^2.$$

Problem 6: Amortisierte Analyse **

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

(a) Zeigen Sie zunächst eine Worst-case-Laufzeit von $O(n \log n)$ für den Zähler.

(b) Zeigen Sie mit Hilfe einer amortisierten Analyse eine schärfere Schranke.

Problem 7: Small-tree-Modifikation von weighted-UNION FIND ***

Ziel der *weighted*-Modifikation aus der Vorlesung (ohne Pfadkompression) ist es, FIND zu beschleunigen. Betrachten Sie im Vergleich folgende Alternative zur *weighted*-Modifikation:
Small-tree-Modifikation: Bei UNION *werden alle Knoten beider Mengen direkte Kinder der Wurzel. Somit hätte jeder Baum konstante Tiefe, und somit* FIND *die Laufzeit* $O(1)$.
Warum verbessert dies die asymptotische Laufzeit nicht? Zeigen Sie, welche asymptotische Laufzeit aus dieser alternativen Modifikation resultiert.