Otto-von-Guericke-Universität Magdeburg
Institut für Mathematische Optimierung
Prof. Dr. Robert Weismantel
Dr. Matthias Köppe, Gebäude 18, Raum 311

Sommersemester 2003

10. (letztes) Übungsblatt

Computerorientierte Mathematik

`http://www.math.uni-magdeburg.de/~mkoeppe/lehre/coma-2003`

Abgabe der Übungsaufgaben: Donnerstag, 26. Juni, zu Beginn der Übung

Organisatorischer Hinweis: Die Veranstaltung am Mittwochmorgen fällt ab sofort weg. Wer noch Programmieraufgaben abgeben möchte, bemühe sich um einen Termin.

32. Aufgabe **10 Punkte**

Sortieren Sie die Liste von Vektoren

$$\binom{1}{7}, \binom{37}{8}, \binom{6}{1}, \binom{4}{10}, \binom{12}{18}, \binom{23}{5}, \binom{47}{11}, \binom{6}{9}, \binom{20}{2}, \binom{8}{8}$$

mittels

(a) Mergesort,

(b) Quicksort

nach aufsteigender erster Komponente.

33. Aufgabe **10 Punkte**

(a) Gegeben sei eine Liste $a_1, \ldots, a_n$ von n Zahlen mit $a_i \in \{1, \ldots, n\}$. Geben Sie einen Algorithmus an, der die Liste in $O(n)$ elementaren Operationen sortiert.

Hinweis: Aus der Vorlesung ist bekannt, daß man im *worst case* $\Omega(n \log n)$ Vergleiche von Elementen benötigt, wenn man nur Vergleiche von Elementen zuläßt. Also sollte Ihr Verfahren etwas anderes machen, als Elemente nur zu vergleichen.

(b) Gegeben sei eine Liste $a_1, \ldots, a_n$ von n Zahlen mit $a_i \in \{1, \ldots, A\}$. Geben Sie einen Algorithmus an, der die Liste in $O(\min\{n \log n, \max\{A, n\}\})$ elementaren Operationen sortiert.

34. Aufgabe **14 Punkte**

Wir betrachten den in der Vorlesung kurz erwähnten Sortieralgorithmus *Insertion Sort*. Dabei ist ein Array A mit Einträgen $A[1], \ldots, A[n]$ gegeben. Der Algorithmus *Insertion Sort* arbeitet wie folgt:

(1) **for** $i \leftarrow 2$ **to** n **do**

(2) $\quad x \leftarrow A[i]$;

(3) $\quad$ Füge x an der richtigen Stelle innerhalb der Sequenz $A[1], \ldots, A[i]$ ein.

Die Bestimmung der korrekten Einfügeposition des Elements x in die Zielsequenz kann mittels

(a) *sequentieller Suche* oder (b) *binärer Suche*

erfolgen. Beschreiben Sie jeweils kurz beide Verfahren. Bestimmen Sie für beide Varianten die Anzahl $C(n)$ der Vergleiche von Arraykomponenten und die Anzahl $A(n)$ von Zuweisungen von Arraykomponenten, die der Algorithmus *Insertion Sort* im schlechtest möglichen Fall (*worst case*) benötigt, um ein Array der Länge n zu sortieren. Bestimmen Sie jeweils die korrekte Größenordnung dieser Anzahlen. Was ergibt sich für eine Laufzeit (im *worst case*) für dieses Sortierverfahren?

35. Aufgabe **8 Punkte**

Sei $S(n)$ die minimale Anzahl von Vergleichsoperationen, die ein Algorithmus im *worst case* benötigt, um n Elemente zu sortieren. Aus der Vorlesung ist die „informationstheoretische Schranke“ $n! \leq 2^{S(n)}$ oder

$$S(n) \geq \lceil \log n! \rceil \tag{1}$$

bekannt, aus der sich die Abschätzung $S(n) = \Omega(n \log n)$ ergibt.

Berechnen Sie die informationstheoretische Schranke (1) für $n = 1, \ldots, 8$. Bestimmen Sie außerdem (für $n = 1, \ldots, 8$) die genaue Anzahl $C(n)$ von Vergleichsoperationen, die *Insertion Sort* mit binärer Suche (aus Aufgabe 34) im *worst case* für die jeweilige Anzahl von Elementen benötigt.

Für welche n ist also *Insertion Sort* mit binärer Suche bestmöglich bezüglich der Anzahl der Vergleichsoperationen?

Zusatzaufgabe: Geben Sie ein Verfahren an, das 5 Elemente mit höchstens 7 Vergleichsoperationen sortiert. **10 mögliche Zusatzpunkte**