Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Moritz Maaß und Hanjo Täubig

Sommersemester 2005
Übungsblatt 10
24.06.2005

Informatik IV

Abgabetermin: 01.07.2005 **vor** *der Vorlesung*

Aufgabe 1 (10 Punkte)

Wieviele Vergleiche benötigt InsertionSort im bestmöglichen Fall?
Beweisen Sie, dass jedes Sortierverfahren mindestens so viele Vergleiche benötigt.

Beweisen Sie, dass InsertionSort zum Sortieren nicht mehr als $\binom{n}{2}$ Vergleiche benötigt.

Aufgabe 2 (10 Punkte)

Gegeben sei eine Matrix $A \in \mathbb{N}^{m \times m}$ zu der die n-te Potenz A^n berechnet werden soll. Geben Sie einen effizienten Algorithmus dazu an und schätzen Sie die Laufzeit im uniformen und logarithmischen Kostenmaß ab.

Aufgabe 3 (10 Punkte)

Betrachten Sie den folgenden Algorithmus, der auf einer Folge $(a_0, \ldots, a_{n-1})$ operiert.
BubbleSort: In der i-ten Phase $(i = 1, \ldots, n-1)$ werden nacheinander für $j = 1, \ldots, n-i$ die Elemente an Position $j-1$ und j miteinander verglichen und anschließend vertauscht, wenn $a_{j-1} > a_j$.
Zeigen Sie, dass dieser Algorithmus die Folge korrekt sortiert, und bestimmen Sie die Anzahl ausgeführter Vergleiche.

Aufgabe 4 (10 Punkte)

Geben Sie je ein Beispiel für eine möglichst ungünstige Eingabe für die Algorithmen SelectionSort, InsertionSort und BubbleSort an.
Wie wirkt sich bei den aus der Vorlesung bekannten Sortierverfahren eine richtige oder falsche Vorsortierung auf die Laufzeit aus?
Welche der Verfahren sind stabil (d.h. sie ändern die relative Ordnung von Elementen mit gleichen Schlüsseln nie)? Geben Sie ein konkretes Beispiel an, in dem nicht stabil sortiert wird.
Sortieren Sie die folgende Sequenz mit HeapSort und geben Sie die Zwischenschritte an:
18, 13, 15, 7, 10, 12, 8, 3, 1, 5.

Aufgabe 5 (10 Punkte)

Sei ein Array $A[0 \ldots n-1]$ mit n paarweise verschiedenen natürlichen Zahlen gegeben. Ein Paar (i, j) heißt *Inversion* (bezüglich der aufsteigenden Sortierung), wenn $i < j$, aber $A[i] > A[j]$.
Aus welchem Grund ist es sinnvoll, diese Größe zu betrachten?
Geben Sie einen Algorithmus an, der die Anzahl der Inversionen in einem Array $A[0 \ldots n-1]$ in Laufzeit $\mathcal{O}(n \log n)$ bestimmt.

Hinweis: Das Problem läßt sich durch Modifikation von MergeSort lösen.