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

WS 1999/2000
Übungsblatt 6
14. Dez. 1999

Grundlegende Algorithmen

Abgabetermin: 21. Dezember

Aufgabe 1

Betrachten Sie den folgenden Sortieralgorithmus auf $n = k^m$ Elementen (für eine Konstante k). Zuerst werden die Elemente in k^{m-1} Gruppen erster Ordnung mit jeweils k Elementen aufgeteilt. Diese Gruppen erster Ordnung werden dann in k^{m-2} Gruppen zweiter Ordnung mit jeweils k Gruppen erster Ordnung aufgeteilt, u.s.w. Allgemein gibt es also k^{m-i} Gruppen i-ter Ordnung, die jeweils aus k Gruppen $(i-1)$-ter Ordnung gebildet wurden.

Zuerst wird in jeder Gruppe das Minimum gebildet. Dabei ist das Minimum einer Gruppe erster Ordnung das Minimum seiner Elemente. Das Minimum einer Gruppe i-ter Ordnung (für $i > 1$) ist das Minimum der Minima der Gruppen $(i-1)$-ter Ordnung, die in dieser Gruppe i-ter Ordnung enthalten sind. Damit ist das Minimum der Gruppe m-ter Ordnung genau das Minimum aller Elemente. Dieses wird nun entfernt und in allen Gruppen, die dieses Element enthielten, das Minimum neu gebildet.

Analysieren Sie diesen Algorithmus im Hinblick auf die Anzahl der benötigten Vergleiche.

Was hat dieser Algorithmus mit Heapsort zu tun?

Bitte wenden!

Aufgabe 2

Bei Quicksort werden im Aufteilungsschritt im allgemeinen zwei Teilfolgen erzeugt, die rekursiv mit Quicksort sortiert werden. Wir nennen diese beiden Folgen, die in einem Aufteilungsschritt generiert wurden, *befreundet*. Wir sagen eine Teilfolge *wartet auf ihre Sortierung*, wenn sie selbst noch nicht sortiert wurde und ihre befreundete Teilfolge noch sortiert wird.

Wieviele Teilfolgen können während eines Ablaufes von Quicksort im schlimmsten Falle auf ihre Sortierung warten? Geben Sie dazu ein Beispiel an.

Nehmen wir nun an, daß jeweils die kürzere der beiden befreundeten Folgen zuerst rekursiv sortiert wird. Zeigen Sie, daß dann maximal $O(\log(n))$ viele unsortierte Teilfolgen auf ihre Sortierung warten können.

Aufgabe 3

Geben Sie einen Divide-and-Conquer-Algorithmus zur Bestimmung des Elementes mit Rang k in einer Folge $(a_1, \ldots, a_n)$ paarweise verschiedener Elemente an, dessen Divide-Schritt der Aufteilungsprozedur in Quicksort stark ähnelt.

Hinweis: Schätzen Sie die Ränge der Elemente in den beiden im Aufteilungsschritt generierten Teilfolgen ab.

Erinnerung: Ein Element einer Folge von paarweise verschiedenen Elementen einer total geordneten Menge hat *Rang* k, wenn genau $k-1$ Elemente der Folge kleiner sind.