Grundzüge von Algorithmen und Datenstrukturen, WS 15/16: Lösungshinweise zum 3. Übungsblatt

Markus Bläser

Aufgabe 3.1 (Lösungshinweise)

a) Dann ist es sortiert!

b) Dies ergibt sich, indem man die Summanden mit $j < i < i + k$ auf beiden Seiten der Gleichung in der Definition von "k-approximativ sortiert" abzieht.

c) Man sortiert die Subarrays $a[1, k+1, 2k+1, \dots]$, $a[2, k+2, 2k+2, \dots]$, $\dots$ jeweils separat. Dies braucht Zeit $O(k \cdot \frac{n}{k} \log \frac{k}{n}) = O(n \log \frac{n}{k})$. Das gesamte Array ist k-approximativ sortiert, weil es die Bedingung aus b erfüllt.

Aufgabe 3.2 (Lösungshinweise) Cormen et. al. Abschnitt 6.5 "Priority queues", Seite 138.

Aufgabe 3.3 (Lösungshinweise)

a) Wir betrachten den Binärbaum der rekursiven Aufrufe ebenenweise. Die erste Ebene besteht aus dem Aufruf für das komplette Array. Dort wird das ganze Array durchlaufen und aufgeteilt, was höchstens cn Vergleiche benötigt, c sei eine geeignete Konstante.

   In der zweiten Ebene haben wir einen Aufruf für die linke Hälfte und einen für die rechte. Da links und rechts zusammen gerade wieder alle Elemente des Arrays untersucht werden (abzüglich des Pivotelements), werden in der zweiten Ebende zusammen auch wieder höchstens cn Vergleiche durchgeführt. Dies setzt sich fort für alle Ebenen.

   Da jeder Aufruf die Größe des zu betrachtenden Teils des Arrays um mindestens 1/10 reduziert, kann es rekursive Aufrufe höchstens bis zur Ebene $\log_{10/9} n$ geben. Denn spätestens in dieser Tiefe ist in jedem Pfad von Aufrufen ein einelementiger oder leere Bereich das Arrays zu untersuchen, was keinen weiteren rekursiven Aufruf benötigt.

   Also haben wir auf höchstens $\log_{10/9} n$ Ebenen jeweils cn Vergleiche, also insgesamt $O(n \log n)$.

b) Auf einem sortieren Array der Länge m macht Quicksort $\Theta(m^2)$ viele Vergleiche. Sei $m = n^{2/3}$. Wir setzen das $(m+1)$-größte Element an den Anfang des Array. Die ersten m Elemente dahinter, aufsteigend sortiert. Dahinter die übrigen Elemente so sortiert, dass Quicksort mit dieser Sortierung quasilineare Zeit braucht. (Ein solche

Sortierung kann man induktiv mit a) konstruieren.) Bei Aufruf von Quicksort wird das Array in zwei Teile geteilt, der Längen m und $n - m$. Auf dem ersten Teil wird dann Zeit $\Theta(m^2) = \Theta(n^{4/3})$ gebraucht. Auf dem zweiten Teil wird Zeit $\Theta((n-m)\log(n-m)) = \Theta(n \log n)$ gebraucht, was asymptotisch kleiner ist.