Algorithmen und
Berechnungskomplexität I
Wintersemester 2022/23

Abgabe: 17.11.21 bis 09:00 Uhr

UNIVERSITÄT BONN

Prof. Dr. Anne Driemel
Frederik Brüning, Jan Eube
Institut für Informatik

Übungsblatt 5

Aufgabe 5.1: Sortieren

(2 + 2 + 3 = 7 Punkte)

Wir betrachten ein Array $A = a_1, ..., a_n$ an natürlichen Zahlen, wobei wir davon ausgehen, dass alle Einträge unterschiedlich sind. Wir bezeichnen den i-ten Eintrag des Arrays als *Extrempunkt* für ein $i \in \{2, ..., n-1\}$, wenn entweder gilt, dass a_i sowohl kleiner ist als a_{i-1} als auch a_{i+1} oder a_i größer ist, als a_{i-1} und a_{i+1}. Mit anderen Worten a_i soll nicht zwischen den beiden benachbarten Einträgen liegen. Des weiteren werden auch a_1 und a_n als Extrempunkte bezeichnet. Zum Verständnis ist in Abbildung 1 ein Array gezeichnet, bei dem die enthaltenen Extrempunkte markiert wurden. Unser Ziel ist es nun einen Algorithmus zu finden, welcher das Array möglichst effizient sortiert, falls die Anzahl der Extrempunkte nicht allzu groß ist.

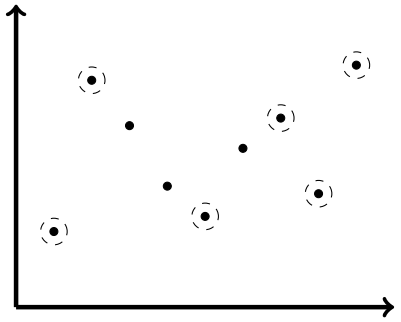

Abbildung 1: Beispiel für die Extrempunkte.

(a) Geben Sie einen Algorithmus an, welcher das Array A in Zeit $O(n \log(k))$ aufsteigend sortiert, wobei k die Anzahl der enthaltenen Extrempunkte ist.

(b) Begründen Sie die Korrektheit Ihres Algorithmus.

(c) Beweisen Sie, dass die Laufzeit, des in (a) entwickelten Algorithmus tatsächlich in $O(n \log(k))$ liegt.

Aufgabe 5.2: Quicksort

(2 + 4 = 6 Punkte)

(a) Zeigen Sie, dass *Quicksort* mit der PARTITION-Funktion aus der Vorlesung nicht stabil ist.

(b) Geben Sie an, wie *Quicksort* (und auch jeder andere nicht stabile vergleichsbasierte Sortieralgorithmus) stabil gemacht werden kann, ohne dessen Laufzeit und die prinzipielle Vorgehensweise zu verändern. Motivieren Sie kurz die Idee Ihres Ansatzes.

Aufgabe 5.3: Vergleichsbasiertes Sortieren

(7 Punkte)

Dem bekannten Wissenschaftler Professor G. Witzt ist es gelungen eine Hardware zu entwerfen, welche den folgenden verallgemeinerten Vergleichsoperator in konstanter Zeit unterstützt (Anmerkung: diese Aufgabe ist fiktiv und es wäre uns nicht bekannt, wie diese Funktion tatsächlich in konstanter Zeit implementiert werden könnte):

- IsSmaller(i,l,r) entscheidet für ein gegebenes Array $A = (a[1], ..., a[n])$ und drei Indexe i, l und r mit $l \leq r$, ob $a[i]$ kleiner gleich aller Elemente in $a[l], a[l+1], ..., a[r]$ ist. Falls ja gibt die Funktion *true* und ansonsten *false* zurück.

Beachten Sie, dass für zwei Indexe i und j gilt, dass IsSmaller(i,j,j) genau dann *true* zurückgibt, wenn $a[i] \leq a[j]$ und es sich demnach tatsächlich um eine Verallgemeinerung des klassischen Vergleichsoperators handelt.

Darauf angesprochen, wie nützlich diese Funktion denn am Ende tatsächlich sei, erwidert G. Witzt, dass man mit diesem neuen Vergleichsoperator ein vergleichsbasiertes Sortierverfahren entwerfen kann, welches ein beliebiges Array in $o(n \log(n))$ sortiert. Beurteilen Sie, ob diese Aussage stimmen kann. Falls ja geben sie einen Algorithmus mit Laufzeit in $o(n \log(n))$ an. Falls nein, beweisen Sie, dass auch mit diesem modifizierten Vergleichsoperator jeder Algorithmus im Worst Case $\Omega(n \log(n))$ Vergleiche durchführen muss.