Christoph Damerius
damerius@informatik.uni-hamburg.de

Version: 4. Dezember 2020
Abgabe: 14.12.2020

Algorithmen und Datenstrukturen

Aufgabenblatt 3

Übung 1

/9 P.

In Algorithmus 1 ist der Pseudocode des Algorithmus StoogeSort zu sehen.

Algorithmus 1: StoogeSort(A, i, j)

```
if A[i] > A[j]
    A[i] ↔ A[j]
if i + 1 ≥ j
    return
k ← ⌊(j − i + 1)/3⌋
StoogeSort(A, i, j − k) # sortiere ersten beiden Drittel
StoogeSort(A, i + k, j) # sortiere letzten beiden Drittel
StoogeSort(A, i, j − k) # sortiere ersten beiden Drittel
```

(a) Beweisen Sie die Korrektheit von StoogeSort. Das heißt, beweisen Sie dass der Aufruf StoogeSort$(A, 1, \text{length}(A))$ das Array A korrekt sortiert. Führen Sie dazu einen Induktionsbeweis über die Länge von A. (5 Punkte)

(b) Analysieren Sie die worst-case Laufzeit von StoogeSort im O-Kalkül. Nutzen Sie dazu eine geeignete Rekursionsgleichung. (4 Punkte)

Übung 2

/7 P.

(a) Sortieren Sie das Array $(20, 13, 8, 5, 2, 12, 9)$ mithilfe des QuickSort-Algorithmus aus der Vorlesung. Stellen Sie den Inhalt des Arrays nach jedem Aufruf von Partition dar. (3 Punkte)

(b) Welchen Einfluss auf die Laufzeit hat allgemein die Auswahl des Pivotelements bei QuickSort? Skizzieren Sie worst-case- und best-case-Eingaben für eine konkrete Auswahl des Pivotelements (z.B. immer am linken oder rechten Rand). Begründen Sie Ihre Antwort. (2 Punkte)

(c) Ein Sortieralgorithmus ist stabil, wenn er die Reihenfolge der Arrayeinträge mit gleichem Wert bewahrt.

Ist der QuickSort-Algorithmus aus der Vorlesung stabil? Begründen Sie Ihre Antwort. (2 Punkte)

Übung 3

______/4 P.

(a) Beschreiben Sie einen Algorithmus in Pseudocode, der zwei vollständige Max-Heaps gleicher Größe n vereinigt. Gehen Sie dazu davon aus, dass die Heaps keine gemeinsamen Elemente enthalten. (2 Punkte)

(b) Analysieren Sie die Laufzeit ihres Algorithmus. (2 Bonuspunkte, wenn sie beweisen können, dass ihr Algorithmus in $O(n)$ läuft.) (2 Punkte)

Übung 4

______/4 P.

Bonusaufgabe

(a) Gegeben sei eine $(n \times n)$-Matrix M mit natürlich-zahligen Einträgen. Wir betrachten nun eine Menge Z von n Zellen von M sodass Z aus jeder Spalte und Zeile von M genau eine Zelle enthält (siehe Abbildung 1).

Beschreiben Sie einen Algorithmus in Pseudocode, der solch eine Menge von Zellen Z mit der maximalen Summe für eine gegebene Matrix M bestimmt. (Hinweis: Benutzen Sie das Prinzip *divide-and-conquer.*; die Laufzeit ihres Algorithmus muss nicht polynomiell sein.) (2 Punkte)

(b) Analysieren Sie die worst-case-Laufzeit ihres Algorithmus'. (2 Punkte)

$$M = \begin{pmatrix} 8 & 5 & 17 & \boxed{13} & 9 & 1 \\ 2 & 5 & 9 & 1 & \boxed{4} & 6 \\ 12 & \boxed{13} & 15 & 2 & 3 & 7 \\ 4 & 2 & 2 & 4 & 1 & \boxed{4} \\ \boxed{11} & 11 & 5 & 9 & 4 & 10 \\ 2 & 7 & \boxed{16} & 9 & 7 & 5 \end{pmatrix}$$

Abbildung 1: Matrix M und blau eingefärbte Menge von Zellen Z mit $\sum_{z \in Z} z = 11 + 13 + 16 + 13 + 4 + 4 = 61$.