

Programmierung und Algorithmen WS 25/26

Übungsblatt 9

Die Lösungen der Aufgaben sind bis zum 11.01.26, 23:59 Uhr abzugeben.

Die Besprechung der Aufgaben erfolgt in KW 3.

Aufgabe 1 (Binäre Suche)

4 Punkte

Implementieren Sie die rekursive Variante der binären Suche (siehe Kapitel 7, Folie 9) in Java. Gehen Sie davon aus, dass es sich bei der Folge F um ein Array aus ganzen Zahlen handelt und somit auch der Suchschlüssel k eine ganze Zahl ist. Sie können zudem davon ausgehen, dass das Feld F bereits sortiert ist.

Aufgabe 2 (Sortierv Verfahren)

10 Punkte

Zeigen Sie anhand der Zahlenfolge (21, 17, 14, 42, 6, 31, 2) den Ablauf der Sortierung mittels der Verfahren

- | | | |
|--------------------|--------------------|-----------------|
| (a) Insertion-Sort | (b) Selection-Sort | (c) Bubble-Sort |
| (d) Merge-Sort | (e) Quick-Sort | |

Veranschaulichen Sie die Abläufe¹ durch Angabe aller notwendigen Schritte und verwenden Sie dabei die Darstellung aus der Vorlesung. Geben Sie für Quicksort auch alle Schritte für *Partition* an und halten Sie sich an die in der Vorlesung vorgestellte Variante von Partition (siehe Kapitel 7, Folie 40).

Aufgabe 3 (Effizientes Sortieren)

5 Punkte

Gegeben sei ein Array A , welches die Zahlen $1, \dots, n$ in beliebiger (unsortierter) Reihenfolge enthält (anders ausgedrückt: A enthält eine *Permutation* der Zahlen $1, \dots, n$). Entwickeln Sie einen Algorithmus in Java, welcher A in-place mit einer Laufzeit von $\mathcal{O}(n)$ durch *Tauschoperationen* sortiert.

Aufgabe 4 (Sortieren mit linearem Aufwand)

5 + 1 Punkte

Ist bereits im Voraus ein begrenztes Intervall diskreter Werte bekannt, aus dem die n Werte eines zu sortierenden Feldes `int[] a` stammen können, ist es möglich `a` in linearer Zeit zu sortieren.

- (a) Schreiben Sie eine Java-Methode, die `a` zusammen mit einer Ganzzahl `int m` übergeben bekommt und die Werte in `a`, die alle aus dem Ganzzahlintervall $[0, m]$ stammen, sortiert. Die Laufzeit Ihres Algorithmus soll dabei in $\mathcal{O}(n+m)$ sein.

Hinweis: Anstatt Elemente zu vergleichen, könnten sie auch gezählt werden.

- (b) Warum lässt sich diese Methode nicht für Felder über *beliebigen* Fließkommazahlen einsetzen?

Bitte wenden!

¹Folgende interaktive Seite <https://mszula.github.io/visual-sorting/> zeigt eine recht unterhaltsame visuelle und akustische Darstellung verschiedener Sortieralgorithmen.

Aufgabe 5 (Suchen von Paaren)

7 + 1 Punkte

Gegeben sei ein Array A aus n ganzen Zahlen, welches keine Duplikate enthält.

- (a) Entwickeln Sie einen Algorithmus mit einer mittleren Laufzeit von $\mathcal{O}(n \log n)$ in Java, welcher bei Eingabe von $z \in \mathbb{Z}$ und A alle Paare $x, y \in A$ ermittelt, so dass $x + y = z$ gilt (sofern diese existieren).

Hinweis: Eventuell ist es hilfreich, das Array zu sortieren. Verwenden Sie dafür keine Methoden aus der Java-Klassenbibliothek!

- (b) Auf der Webseite zur Vorlesung finden Sie neben dem Übungsblatt eine Textdatei `sequence.txt`, welche eine Sequenz ganzer Zahlen enthält. Geben Sie alle Paare x, y aus dieser Sequenz an, für die $x + y = 42$ gilt.

Hinweis: Zum Einlesen von `sequence.txt` können Sie die Klasse `pua.FileUtils` verwenden:

```
1 import pua.FileUtils;
2 ...
3 String path = FileUtils.getPath(); // In der WebIDE stattdessen Pfad direkt angeben, z.B. "/"
referenceCode/files/blatt_09/sequence.txt"
4 int[] sequence = FileUtils.readIntArray(path);
```