technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 11

Jonas Ellert
Mart Hagedoorn
Guangping Li

Sommersemester 2022
04. Mai

DAP2 Praktikum – Blatt 5

Abgabe: ab 09. Mai

Studienleistung

- Zum Bestehen des Praktikums muss jeder Teilnehmer die folgenden Leistungen erbringen:
  - Es müssen mindestens 50 Prozent der Punkte in den Kurzaufgaben erreicht werden.
  - Es müssen mindestens 50 Prozent der Punkte in den Langaufgaben erreicht werden.
- Im Krankheitsfall kann ein Testat bei Vorlage eines Attests in der folgenden Woche nachgeholt werden.
- Wenn ein Praktikumstermin auf einen Feiertag fällt, müssen Sie sich an einem beliebigen anderen Praktikumstermin in der gleichen Woche testieren lassen.
- **Hinweis:** Notieren Sie sich Ihre Punkte nach jedem Testat! Dies dient der eigenen Kontrolle. (Ihr Punktestand kann Ihnen während des Semesters nicht genannt werden.)

Wichtige Information (im Moodle verfügbar)

- Beachten Sie die Erklärung des **Ablaufs (Blatt A)**.
- Beachten Sie die **Regeln und Hinweise (Blatt R)** in der aktuellsten Version!
- Beachten Sie die **Hilfestellungen (Blatt H)** in der aktuellsten Version!

Kurzaufgabe 5.1: Frequenzen Zählen (3 Punkte)

Schreiben Sie ein Programm, das eine Liste A von Ganzzahlen $a_0, a_1, \ldots, a_{n-1}$ via Standard-In bekommt, und die Anzahl der Vorkommen jeder Zahl in der Liste A bestimmt. Dies soll in folgenden Schritten umgesetzt werden:

- Zuerst werden die Zahlen $a_0, a_1, \ldots, a_{n-1}$ in ein Array `data` eingelesen.
- Anschließend werden Maximum und Minimum des Arrays bestimmt. Implementieren Sie dafür die folgenden zwei Methoden:

  ```
  public static int getMin(int[] data)
  public static int getMax(int[] data) .
  ```

- Schreiben Sie eine Methode `count`, die ein Array `data` von Ganzzahlen sowie zwei zusätzliche Schlüssel `min` und `max` erhält. Ausgabe der Methode ist ein Array C der Länge $\texttt{max} - \texttt{min} + 1$, sodass $C[i]$ die Anzahl der Vorkommen von $i + \texttt{min}$ in der Liste A ist.

  ```
  public static int[] count(int[] A, int min, int max)
  ```

- Schreiben Sie eine `main`-Methode, welche die vorherigen Arbeitsschritte miteinander verbindet, und zu einer Ausgabe wie im unten stehenden Beispiel führt. Wie immer sollten Sie bei ungültigen Eingaben passende Fehlermeldungen ausgeben!

  Für die volle Punktzahl sollte Ihr Programm für das Zählen (abgesehen vom Lesen der Eingabe und Schreiben der Ausgabe) **nicht mehr als $\mathcal{O}(n)$ Rechenschritte benötigen.**

Beispielausgabe des Programms:

```
seq 4 > seq4.txt
seq 7 > seq7.txt
cat seq4.txt seq7.txt | java Counting.java
The minimum value: 1
The maximum value: 7
Frequencies:  [2, 2, 2, 2, 1, 1, 1]
```

Kurzaufgabe 5.2: Counting-Sort (3 Punkte)

- Für dieses Problem erhalten sie erneut eine Liste von Ganzzahlen $a_0, a_1, \ldots, a_{n-1}$ via Standard-In. Implementieren Sie den Algorithmus *Counting-Sort* aus der Vorlesung. (Den Pseudocode finden Sie im Skript.) Dieser Algorithmus soll die Liste der Ganzzahlen in *absteigende Reihenfolge* bringen. Dies geschieht mit einer Methode

  `public static int [] countingSort(int[] data)`.

  Zum Zählen der Frequenzen dürfen Sie ein Hilfsarray der Länge $\texttt{max} - \texttt{min} + 1$ verwenden (abgesehen von dem Ausgabearray sind keine weiteren Hilfsarrays erforderlich). Dabei sind `max` und `min` der maximale bzw. minimale Wert in der gegebenen List von Ganzzahlen.

  Überschreiben Sie `data` mit der absteigend sortierten Liste von Ganzzahlen, und geben Sie als Rückgabewert das Frequenzarray zurück.

  Für die volle Punktzahl sollte Ihr Programm (abgesehen vom Lesen der Eingabe und Schreiben der Ausgabe) **nicht mehr als $\mathcal{O}(n + \texttt{max} - \texttt{min})$ Rechenschritte benötigen.**

- Schreiben Sie eine geeignete `main`-Methode, welche das Frequenzarray und die absteigend sortierte Liste ausgibt. Wie immer sollten Sie bei ungültigen Eingaben passende Fehlermeldungen ausgeben!

  Beispielausgaben des Programms:

```
seq 4 > seq4.txt
seq 7 > seq7.txt
cat seq4.txt seq7.txt | java CountingSort.java
The minimum value: 1
The maximum value: 7
Before sorting:  [1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 7]
After sorting:  [7, 6, 5, 4, 4, 3, 3, 2, 2, 1, 1]
Frequencies:  [2, 2, 2, 2, 1, 1, 1]
```

Kurzaufgabe 5.3: Das Auswahlproblem (2 Punkte)

Auf Blatt 4 haben Sie sich bereits ausgiebig mit dem Auswahlproblem befasst. Dabei haben Sie Duplikate einfach ignoriert. Zum Beispiel sind sie auf Blatt 4 davon ausgegangen, dass 2 das 4-kleinste Element von $[1, 1, 2, 2, 2, 4, 5, 5, 7]$ ist. Jetzt sollen Sie eine Methode

```
public static int exactSelect(int[] data, int k)
```

schreiben, die das `k`-kleinste Element in einem Array findet, wobei Duplikate beachtet werden: Das 4-kleinste Element von $[1, 1, 2, 2, 2, 4, 5, 5, 7]$ ist dann 5. Allgemein ausgedrückt ist a das k-kleinste Element des Arrays, wenn es im Array genau $k - 1$ unterschiedliche Elemente gibt, die kleiner als a sind. Wenn es weniger als k unterschiedliche Elemente gibt, soll die Methode `Integer.MAX_VALUE` zurückgeben.

Verwenden Sie den Mechanismus von Counting-Sort, ohne dass Sie das Array tatsächlich sortieren. Sie dürfen wie in der vorherigen Aufgabe ein Hilfsarray der Länge $\texttt{max} - \texttt{min} + 1$ anlegen, und wieder nur $\mathcal{O}(n + \texttt{max} - \texttt{min})$ Rechenschritte benötigen.

Um das Ergebnis zu ermitteln, müssen Sie nicht wirklich wissen, wie oft jeder Wert vorkommt. Tatsächlich reicht es, wenn Sie sich für jeden möglichen Wert speichern, ob dieser überhaupt in der Eingabe vorkommt. Bei Ihrem Hilfsarray sollte es sich daher nicht um ein `int`-Array, sondern um ein `boolean`-Array handeln. (Dieses benötigt deutlich weniger Platz.)

Schreiben Sie eine geeignete `main`-Methode, sodass Ihr Programm eine Liste von Ganzzahlen via Standard-In, sowie ein Argument k erhält, und mit der Methode `exactSelect` den k-kleinsten Wert findet und ausgibt. Wie immer sollten Sie bei ungültigen Eingaben passende Fehlermeldungen ausgeben!