technische universität dortmund

Fakultät für Informatik
Lehrstuhl 11

Erik Thordsen
Lena Krieg
Sebastian Konietzny

Sommersemester 2023
8. Mai

DAP2 Praktikum – Blatt 5

Abgabe: ab 15. 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.
- Ansonsten kann die Testierung **nur in der zugeteilten Gruppe** garantiert werden.
- Bitte bereiten Sie den Tester sowie in den Aufgaben angegebene Beispieltests vor, bevor Sie sich testieren lassen!

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!

Kommandozeile Tester

Sie finden im Moodle eine Datei `Test.jar`. Fuer das Testen Ihrer Lösung laden Sie diese herunter und geben den unten stehenden Kommandozeilen Befehl ein. **Es ist möglich, dass die Tests verzögert zur Verfügung gestellt oder vervollständigt werden!**

```
java -jar <path-to-moodle-jar>/Test.jar -s <path-to-solution> 5 -e
```

Kurzaufgabe 5.1: Frequenzen Zählen (4 Punkte)

In dieser Aufgabe erhalten Sie eine Liste von n Ganzzahlen $a_0, a_1, \ldots, a_{n-1}$ via Standard-In und sollen die Häufigkeit *aller* Ganzzahlen im Intervall $[\min(a_0, a_1, \ldots, a_{n-1}), \max(a_0, a_1, \ldots, a_{n-1})]$ bezüglich dieser Liste bestimmen.

Legen Sie eine Klasse `AufgabeB5A1` an, in welcher Sie folgende Methoden implementieren:

- Die Methode `public static int[] readInput()` soll alle Ganzzahlen aus Standard-In einlesen. Wie in den vorigen Wochen sollen nur Eingaben akzeptiert werden, die vollständig aus Ganzzahlen bestehen. Die eingelesenen Ganzzahlen sollen als `int[]` zurückgegeben werden. Entstehende `NumberFormatException` sollen von dieser Methode weitergeleitet werden.
- Der Konstruktor `public AufgabeB5A1(int[] data)` soll eine neue Instanz dieser Klasse erstellen und sich das übergebene `data` in einem Attribut mit dem selben Namen speichern.
- Die Methode `public int getMin()` soll das *Minimum* von `this.data` zurückgeben.
- Die Methode `public int getMax()` soll das *Maximum* von `this.data` zurückgeben.
- Die Methode `public int[] count()` soll ein Array C der Länge `this.getMax()` − `this.getMin()` + 1 zurückgeben, sodass $C[i]$ die Häufigkeit von i + `this.getMin()` in der Liste `this.data` enthält.
- Die Methode `public static void main(String[] args)` enthält wie immer das ausführbare Programm Ihrer Implementierung. Sie sollen hier die Zahlen aus der Eingabe lesen, eine Instanz der Klasse anlegen und die Häufigkeit der entsprechenden Ganzzahlen bestimmen. Diese sollen dann entsprechend der Beispiele ausgegeben werden. Dazu dürfen Sie `Arrays.toString()` aus der Bibliothek `java.util.Arrays` verwenden. Ihr Programm sollte nur $\mathcal{O}(n)$ Rechenschritte benötigen.

```
// Input: -1 0 1 2 3 1 2 3 4 5
(seq -1 3; seq 5) | java AufgabeB5A1
// Output:
[1, 1, 2, 2, 2, 1, 1]
```

Geforderte Klassen und Methoden

```
public class AufgabeB5A1{
  public static void main(String[] args) {...}
  public static int[] readInput() throws NumberFormatException {...}
  public int[] data;
  public AufgabeB5A1(int[] data) {...}
  public int getMin() {...}
  public int getMax() {...}
  public int[] count() {...}
}
```

Kurzaufgabe 5.2: Counting-Sort (8 Punkte)

In dieser Aufgabe erhalten Sie erneut eine Liste von n Ganzzahlen $a_0, a_1, \ldots, a_{n-1}$ via Standard-In und sollen diese in *aufsteigender* Reihenfolge sortieren. Dazu sollen Sie den Algorithmus *Counting-Sort*, der in der Vorlesung behandelt wurde, implementieren. Die Schlüsselmenge entspricht den Ganzzahlen aus dem Intervall $[\min(a_0, a_1, \ldots, a_{n-1}), \max(a_0, a_1, \ldots, a_{n-1})]$. Den Pseudocode finden Sie im Skript.

Legen Sie eine Klasse `AufgabeB5A2` an, in welcher Sie folgende Methoden implementieren:

- Die Methode `public static int[] readInput()` soll wie in Kurzaufgabe 5.1 alle Ganzzahlen aus Standard-In einlesen.
- Der Konstruktor `public AufgabeB5A2(int[] data)` soll eine neue Instanz dieser Klasse erstellen und sich das übergebene `data` in einem Attribut mit dem selben Namen speichern.
- Die Methode `public int[] countingSort()` soll den Algurithmus *Counting-Sort* implementieren. Zum Zählen der Frequenzen dürfen Sie Ihr Ergebnis aus Kuraufgabe 5.1 verwenden (weitere Hilfsarrays sind nicht erlaubt). Überschreiben Sie `this.data` mit der absteigend sortierten Liste von Ganzzahlen, und geben Sie als Rückgabewert das Frequenzarray zurück. Beachten Sie dabei, dass der Algorithmus aus dem Skript angepasst werden muss, um auf Hilfsarrays zu verzichten.
- Die Methode `public static void main(String[] args)` enthält wie immer das ausführbare Programm Ihrer Implementierung. Sie sollen hier die Zahlen aus der Eingabe lesen, eine Instanz der Klasse anlegen und die Liste der Ganzzahlen absteigend sortieren. Die sortierte Liste soll samt den Frequenzarray entsprechend der Beispiele ausgegeben werden. Dazu dürfen Sie `Arrays.toString()` aus der Bibliothek `java.util.Arrays` verwenden. Ihr Programm sollte nur $\mathcal{O}(n + \max(a_0, a_1, \ldots, a_{n-1}) - \min(a_0, a_1, \ldots, a_{n-1}))$ Rechenschritte benötigen.

```
// Input: -1 0 1 2 3 1 2 3 4 5
(seq -1 3; seq 5) | java AufgabeB5A2
// Output:
Before sorting:  [-1, 0, 1, 2, 3, 1, 2, 3, 4, 5]
Frequencies:  [1, 1, 2, 2, 2, 1, 1]
After sorting:  [5, 4, 3, 3, 2, 2, 1, 1, 0, -1]
```

Geforderte Klassen und Methoden

```
public class AufgabeB5A2{
  public static void main(String[] args) {...}
  public static int[] readInput() throws NumberFormatException {...}
  public int[] data;
  public AufgabeB5A2(int[] data) {...}
  public int[] countingSort() {...}
}
```

Kurzaufgabe 5.3: Das Auswahlproblem (4 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 3 das 4-kleinste Element von $[1, 2, 3, 3, 3, 5, 5, 7]$ ist. In dieser Aufgabe erhalten Sie erneut eine Liste von n Ganzzahlen $a_0, a_1, \ldots, a_{n-1}$ via Standard-In und sollen das `k`-kleinste Element in einem Array bestimmen, wobei Duplikate beachtet werden: Das 4-kleinste Element von $[1, 2, 3, 3, 3, 5, 5, 7]$ 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.

Legen Sie eine Klasse `AufgabeB5A3` an, in welcher Sie folgende Methoden implementieren:

- Die Methode `public static int[] readInput()` soll wie in Kurzaufgabe 5.1 und 5.2 alle Ganzzahlen aus Standard-In einlesen.
- Der Konstruktor `public AufgabeB5A3(int[] data)` soll eine neue Instanz dieser Klasse erstellen und sich das übergebene `data` in einem Attribut mit dem selben Namen speichern.
- Die Methode `public int exactSelect(int k)` soll das k-kleinste Element des Arrays zurückgeben. Verwenden Sie dabei den Mechanismus von Counting-Sort, ohne dass Sie `this.data` tatsächlich sortieren. 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).
- Die Methode `public static void main(String[] args)` enthält wie immer das ausführbare Programm Ihrer Implementierung. Um das `k`-kleinste Element zu bestimmen, sollen Sie die Liste aus der Eingabe einlesen, eine Zahl als Argument erhalten und die Instanz-Methode aufrufen. Das `k`-kleinste Element soll entsprechend der Beispiele ausgegeben werden. Ihr Programm sollte nur $\mathcal{O}(n+\max(a_0, a_1, \ldots, a_{n-1})-\min(a_0, a_1, \ldots, a_{n-1}))$ Rechenschritte benötigen. Wie immer sollten Sie bei ungültigen Eingaben passende Fehlermeldungen ausgeben.

```
// Input: 1 3 5 7 3 5 2 3
(seq 1 2 7; seq 3 2 5; seq 2 3) | java AufgabeB5A3 4
// Output:
5
```

Geforderte Klassen und Methoden

```
public class AufgabeB5A3{
  public static void main(String[] args) {...}
  public static int[] readInput() throws NumberFormatException {...}
  public int[] data;
  public AufgabeB5A3(int[] data) {...}
  public int exactSelect(int k) {...}
}
```