

Mart Hagedoorn Guangping Li Sommersemester 2024 10. Mai

DAP2 Praktikum – Blatt 6

Abgabe: KW 21

Studienleistung

- Zum Bestehen des Praktikums muss jeder Teilnehmer*innen 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!

Frequenzen Zählen. Schreiben Sie ein Programm, das eine Ganzzahl n und 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 $\max - \min + 1$, sodass C[i] die Anzahl der Vorkommen von $i + \min$ in der Liste A ist.

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

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

Counting-Sort. In der Vorlesung haben Sie CountingSort kennengelernt.

• Für dieses Problem erhalten sie erneut eine Ganzzahl n und eine Liste von Ganzzahlen $a_0, a_1, \ldots, a_{n-1}$ via Standard-In. Implementieren Sie den Algorithmus Counting-Sort aus der Vorlesung. 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 $\max - \min + 1$ verwenden (und weitere Hilfsarrays sind nicht erlaubt). Dabei sind \max und \min der maximale bzw. minimale Wert in der gegebenen List von Ganzzahlen.

Ausgabe der Methode ist eine absteigend sortierten Liste B der Länge n von Ganzzahlen von data.

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

• Schreiben Sie eine geeignete main-Methode, welche das Frequenzarray und die absteigend sortierte Liste ausgibt. Wie immer können Sie davon ausgehen, dass die Eingabe korrekt ist und müssen den Code nicht um diese Eingabe Robust gestalten.

Beispielausgaben des Programms:

```
java B6A1.java <Enter>
6 1 2 2 3 4 4
Input Array: [1, 2, 2, 3, 4, 4]
The minimum value: 1
The maximum value: 4
Frequencies: [1, 2, 1, 2]
After sorting: [4, 4, 3, 2, 2, 1]
```

Langaufgabe 6.2: Radixsort

(4 Punkte)

In der Vorlesung haben Sie *Radixsort* kennengelernt. Radixsort wird oft damit motiviert, dass eine Liste von mehrdimensionalen Schlüsseln sortiert werden soll. Tatsächlich ist Radixsort aber auch gut dazu geeignet, eine Liste von Ganzzahlen zu sortieren. Dabei wird jeder int-Wert als ein Schlüssel mit vier Komponenten interpretiert, wobei jede Komponente eines der vier Bytes ist, welche den int-Wert bilden. Das höchstwertigste Byte (Most-Significant-Digit, MSD) ist die erste Komponente, und das niederwertigste Byte (Least-Significant-Digit, LSD) ist die letzte Komponente. Beispiel:

Wert in Basis-10-Darstellung:

1661914940

Wert in 32-Bit-Binärdarstellung:

 $01100011\ 00001110\ 11001111\ 00111100$

Wert in Byte-Komponenten:

Jedes Byte entspricht dabei einer Stelle in der Basis-256-Darstellung des Wertes:

$$1661914940 = 99 \cdot 256^3 + 14 \cdot 256^2 + 207 \cdot 256^1 + 60 \cdot 256^0.$$

Sortieren nach einem Byte. Implementieren Sie die Methode:

public static void sortByByte(int[] input, int 1, int r, int b).

Diese bekommt ein Array input übergeben, und soll das Arrayintervall input [1, r] absteigend und stabil nach dem b-niederwertigsten Byte sortieren (für b=0 nach dem niederwertigsten Byte, für b=3 nach dem höchstwertigsten). Dabei können Sie den Mechanismus von Counting-Sort verwenden oder Arrays.sort() nutzen, und ein Frequenzarray mit 256 Einträgen benutzen. Sie dürfen ein Hilfsarray der Länge r-1+1 verwenden, und davon ausgehen, dass nur positive Ganzzahlen eingegeben werden. Überschreiben Sie input mit der absteigend sortierten Liste von Ganzzahlen.

Tipp: Wie Sie oben gesehen haben, ist das b-niederwertigste Byte eines Integers a der Wert der b-niederwertigsten Stelle in Basis 256. Mathematisch entspricht dies (a/256^b) mod 256. Praktisch berechnen Sie die Division durch einen Bitshift, und den Modulo durch die Verundung mit einer Maske: (a * (8 * b)) & 0xFF.

Radixsort. Implementieren Sie die Methode:

```
public void radix(int[] data),
```

welche die Ganzzahlen in data absteigend sortiert. Diese Methode sortiert die Zahlen zuerst nach dem niederwertigsten Byte mittels sortByByte (Aufgabe 6.1). Dann sortiert der Prozess die Zahlen nach dem zweit-niederwertisten Byte, dem dritt-niederwertigsten Byte, und schließlich nach dem höchstwertigsten (viert-niederwertigsten) Byte.

Abschließend sollen Sie eine geeignete main-Methode schreiben, sodass Ihr Porgramm eine Ganzzahl n und eine Liste von Ganzzahlen $a_0, a_1, \ldots, a_{n-1}$ via Standard-In erhält, und dann die Liste **absteigend** sortiert ausgibt. Sie dürfen davon ausgehen, dass nur positive Ganzzahlen eingegeben werden.

Beispielausgaben des Programms:

java B6A2.java <Enter>

7 692092 122 22223 2777 322434 47676 4999 <Enter>

Input Array: [692092, 122, 22223, 2777, 322434, 47676, 4999] After sorting: [692092, 322434, 47676, 22223, 4999, 2777, 122]