Universität der Bundeswehr München
Elektrotechnik und Technische Informatik
Oswald / Baumann

Wintertrimester 2021
Praktikum 05
Abgabe: 15.02.-16.02.2021

Grundlagen der Programmierung

Sortieren

Aufgabe 1: Sortieren durch Auswählen

In dieser Aufgabe sollen Sie Zahlen in einem Feld aufsteigend sortieren. Es soll das Sortieren durch Auswählen umgesetzt werden. Die grundlegende Idee des Algorithmus ist dabei wie folgt (siehe Abbildung 1): Das kleinste Element aus dem unsortierten (rechten) Bereich einer Folge herausnehmen und mit dem ersten Element aus dem unsortierten Bereich vertauschen. Danach ist der sortierte (linke) Bereich um eins größer geworden.

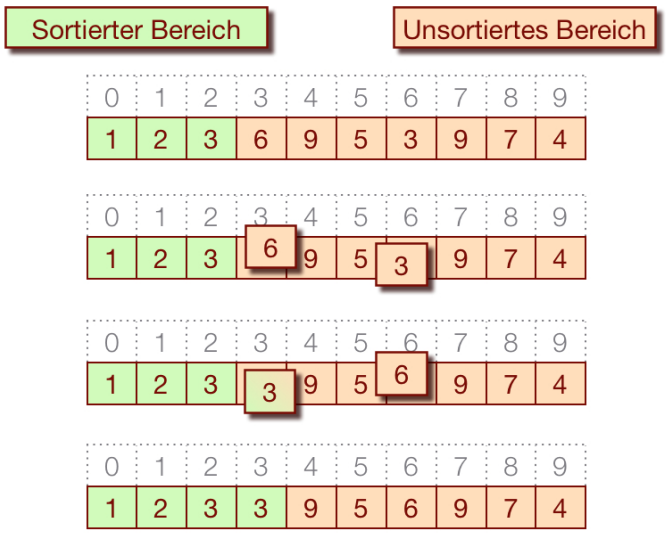

Abbildung 1: Sortieren durch Auswählen

Hinweis: Am Anfang enthält der sortierte Bereich 0 Elemente und der unsortierte Bereich umfasst das gesamte Feld.

a) Erstellen Sie die Klassenmethode
`public static void selectionSort(double[] a)`,
die ein beliebiges Array vom Typ `double[]` nach dem obigen Verfahren aufsteigend sortiert. Sie dürfen für die Umsetzung kein zusätzliches Feld in der Klassenmethode erzeugen, d.h. Ihr Algorithmus muss auf dem übergebenen Feld `a` arbeiten.

b) Ergänzen Sie zwei globale Variablen: die eine zählt die Anzahl der nötigen Vergleiche zwischen im Array vorhandener Zahlen, die andere zählt die Anzahl der Vertauschungen. Geben Sie die beiden Variablen nach der Terminierung des Algorithmus auf der Standardausgabe aus.

Wie viele Vergleiche und Vertauschungen benötigt Ihr Algorithmus, wenn Sie ihn mit den folgenden beiden Aufrufen testen:
`selectionSort(new double[]{1.0, 2.0, 3.0, 4.0, 5.0})` und
`selectionSort(new double[]{5.0, 4.0, 3.0, 2.0, 1.0})`.

Aufgabe 2: Sortieren von Arrays mit Bubblesort

Für diese Aufgabe müssen Sie die Klassen `ArrayGenerator` und `ArrayVisualizer` verwenden. Die Klassen finden Sie in der jar-Datei `ArrayVisualization.jar`. Die jar-Datei und die Dokumentation zu den beiden Klassen finden Sie auf dem Dokumentenserver in den pdf-Dateien `ArrayGenerator.pdf` und `ArrayVisualizer.pdf`. Die Klasse `ArrayGenerator` ermöglicht die Erstellung von Arrays mit unsortierten Zufallszahlen vom Typ `int[]` oder `double[]`. Die Klasse `ArrayVisualizer` ermöglicht es auf einfache Weise beliebige Arrays vom Typ `int[]` auf dem Bildschirm anzuzeigen.

a) Erstellen Sie die Klassenmethode
`public static void bubbleSort(int[] x)`,
die ein beliebiges Array vom Typ `int[]` mit dem Bubblesort-Algorithmus (siehe Vorlesungsunterlagen) sortiert.

   Verwenden Sie die Klasse `ArrayGenerator` um Ihren Algorithmus zu testen. Verwenden Sie die Klasse `ArrayVisualizer` um Ihren Algorithmus zu visualisieren.

b) Ergänzen Sie zwei Variablen: die eine zählt die Anzahl der nötigen Vergleiche zwischen im Array vorhandener Zahlen, die andere zählt die Anzahl der Vertauschungen, die Ihr Algorithmus benötigt. Geben Sie die beiden Variablen nach der Terminierung des Algorithmus auf der Standardausgabe aus.

   Wie viele Vergleiche und Vertauschungen benötigt Ihr Algorithmus, wenn Sie ihn mit den folgenden beiden Aufrufen testen:
   `bubbleSort(ArrayGenerator.generateDescendingArray(10))` und
   `bubbleSort(ArrayGenerator.generateAscendingArray(10))`.
   Im ersten Fall sollte Ihr Algorithmus 45 Vergleiche und 45 Vertauschungen benötigen. Im zweiten Fall sollte Ihr Algorithmus 9 Vergleiche und keine Vertauschung benötigen.

Aufgabe 3: Sortieren eines `double`-Arrays mit Quicksort

Aus der Vorlesung kennen Sie den Quicksort-Algorithmus mit dem Sie ein `int`-Array sortieren können:

```
public static void quickSort(int[] a, int i, int j) {
    if (a == null || j <= i)
        return;

    int pivot = a[i];
    int vonLinks = i;
    int vonRechts = j;

    while (vonLinks <= vonRechts) {
        while (a[vonLinks] < pivot) {
```

```
            vonLinks++;
            if (vonLinks > j) break;
        }
        while (a[vonRechts] >= pivot) {
            vonRechts--;
            if (vonRechts < i) break;
        }
        if (vonLinks < vonRechts) {
            int tmp = a[vonLinks];
            a[vonLinks] = a[vonRechts];
            a[vonRechts] = tmp;
        }
    }

    if (vonRechts < i) {
        // quickSort(a, i, i);
        quickSort(a, i + 1, j);

    } else {
        quickSort(a, i, vonRechts);
        quickSort(a, vonLinks, j);
    }
}
```

In dieser Aufgabe, sollen Sie den obigen Algorithmus so anpassen, dass er auf `double`-Arrays arbeitet und die Sortierreihenfolge genau umgekehrt ist.

a) Adaptieren Sie obige Klassenmethode, so dass an Stelle des `int`-Arrays (erster Parameter) ein `double`-Array übergeben werden kann. Außerdem sollen die Werte in dem Array nach dem Aufruf absteigend sortiert sein.

Der Aufruf:

```
double[] x = {1.0, 3.0, 4.5, 2.3, 7.1, 1.2, 1.5, 3.0};
quickSort(x, 0, x.length - 1);
System.out.println(Arrays.toString(x));
```

soll die Ausgabe:

```
[7.1, 4.5, 3.0, 3.0, 2.3, 1.5, 1.2, 1.0]
```

erzeugen.

Tipp: Sie können die Klasse `ArrayGenerator` verwenden um Ihren Algorithmus zu testen. Die Klasse finden Sie in der jar-Datei `ArrayVisualization.jar`. Die jar-Datei und die Dokumentation zu der Klasse finden Sie auf dem Dokumentenserver in der pdf-Datei `ArrayGenerator.pdf`.