Westfälische Wilhelms-Universität Münster

Übungen zur Vorlesung „Datenstrukturen und Algorithmen“ im SoSe 2017

Prof. Dr. Klaus Hinrichs
Aaron Scherzinger

Blatt 6

Abgabe im Learnweb bis zum 15.06.2017 um 10 Uhr

Aufgabe 19: (24+6=30 Punkte) Verwenden Sie den im Learnweb für diese Aufgabe zur Verfügung gestellten Programmrahmen, um die folgenden Aufgaben zu bearbeiten:

(a) Implementieren Sie eine animierte Version der folgenden Sortierverfahren:

- *selection sort*
- *insertion sort*
- *merge sort*

Machen Sie sich hierzu zunächst mit der Verwendung der Methoden `setData`, `setLegend`, `setLabel`, `setHighlight`, `setColor`, `sleep` und `repaint` der mitgelieferten Klasse `Visualizer` vertraut. Das zu sortierende Array `int[] a`, das Zahlen zwischen 1 und 100 enthält, ist im Koordinatensystem so dargestellt, dass die Punkte $(i, a[i])$ markiert werden. Ein Beispiel für die Animation des *Bubble-Sort*-Verfahrens (siehe Aufgabe 20), welche die entsprechende Funktionalität der Klasse `Visualizer` verwendet, ist in der Methode `visualBubbleSort` der Klasse `Aufgabe19` als Beispiel implementiert.

Implementieren Sie nun die drei oben angegebenen Verfahren in `JAVA` in den vorgegebenen Methoden `visualSelectionSort`, `visualInsertionSort` und `visualMergeSort` in der Klasse `Aufgabe19`. Ihre Animation soll jeden Vergleich und jede Vertauschung anzeigen. Fügen Sie der `main`-Methode die entsprechenden Aufrufe der Sortieralgorithmen hinzu. Beachten Sie hierbei, dass Sie (wie in der `main`-Methode bereits beispielhaft angegeben) den Zustand zurücksetzen müssen.

(b) Testen Sie Ihre Lösung mit den im Learnweb bereitgestellten Datensätzen. Realisieren Sie hierzu eine Methode, die den Inhalt einer `ASCII`-Datei (jeweils ein `int`-Wert pro Zeile) in das Array `int[] a` einliest. Implementieren Sie die Möglichkeit, über Kommandozeilenparameter beim Aufruf des Programms auszuwählen, welche Datei eingeladen werden soll. Wenn kein Parameter angegeben wird, soll der ursprünglich im Programm vorhandene Zufallsgenerator (d.h. die Methode `fillArray`) verwendet werden.
Hinweis: Machen Sie sich für die Bearbeitung dieser Aufgabe selbstständig mit der notwendigen Funktionalität der `JAVA` API vertraut.

Aufgabe 20: (2+3+3=8 Punkte) Betrachten Sie den folgenden *Bubble-Sort*-Algorithmus.

```
public void bubblesort(int[] a) {
    int hilf = 0;
    boolean vertauscht = false;
    int run = 0;
    do {
        run += 1;
        vertauscht = false;
        for (int i=0; i<a.length-run; i++) {
            if (a[i] > a[i+1]) {
                hilf = a[i];
                a[i] = a[i+1];
                a[i+1] = hilf;
                vertauscht = true;
            }
        }
    } while (vertauscht);
}
```

Bearbeiten Sie nun die folgenden Aufgaben:

(a) Beschreiben Sie umgangssprachlich (aber exakt) die Vorgehensweise dieses Algorithmus.

(b) Begründen Sie, warum der Algorithmus terminiert.

(c) Bestimmen Sie die asymptotische Laufzeitkomplexität des Algorithmus.

Aufgabe 21: (6+8+6=20 Punkte) Bearbeiten Sie die folgenden Aufgaben zum Quicksort-Algorithmus. Verwenden Sie dazu eine der in der Vorlesung besprochenen Techniken zur Bestimmung des Pivot-Elements.

(a) Integrieren Sie den oben genannten Quicksort-Algorithmus in Ihre Lösung zu Aufgabe 19.

(b) Modifizieren Sie den in der Vorlesung besprochenen *Quicksort*-Algorithmus, sodass ab einer (vorher festgelegten) Teilproblemgröße von m Elementen nicht mehr rekursiv verfahren wird, sondern *insertion sort* aufgerufen wird. Bestimmen Sie experimentell einen (für Ihre Implementierung) geeigneten Wert von m und dokumentieren Sie Ihre Vorgehensweise hierbei.

(c) Integrieren sie diese modifizierte Version ebenfalls in Ihre Lösung zu Aufgabe 19.