Westfälische Hochschule

Prof. Dr. Manfred Meyer
Fachbereich 6 / Bocholt
Professur BWL / Wirtschaftsinformatik

Algorithmen und Datenstrukturen
(Studiengang Wirtschaftsinformatik 3. Semester)

Aufgabenstellung für das Praktikum am 23.11.2023

Aufgabe 17:

Programmieren Sie für das Sortieren durch Auswahl (SelectionSort) eine rekursive Implementierung mit folgender Signatur:

```
static void selectionSort_rekursiv(int[] zahlen)
```

Die Implementierung weiterer statischer Operationen ist ausdrücklich zugelassen.

Testen Sie Ihre Funktion durch ein entsprechendes Rahmenprogramm zum Einlesen, Sortieren und Ausgeben des sortierten Feldes.

Aufgabe 18:

Ändern Sie den QuickSort-Algorithmus aus der Vorlesung so, dass er als Vergleichselemente den Median von drei aus dem zu sortierenden Bereich stammenden Zahlenbenutzt. Definieren Sie dazu eine Operation mit der Signatur

```
static void quickSort3M(int[] zahlen),
```

wobei die Implementierung weiterer statischer Operationen ausdrücklich zugelassen ist.

Testen Sie Ihre Funktion durch ein entsprechendes Rahmenprogramm zum Einlesen, Sortieren und Ausgeben des sortierten Feldes.

Aufgabe 19:

Modifizieren Sie die Implementierung des Sortierverfahrens QuickSort so, dass dieser keine rekursiven Aufrufe mehr enthält. Verwenden Sie stattdessen eine eigenen Datentyp Auftrag sowie eine Instanz der Klasse `Stack<Auftrag>`, in der Sie den Laufzeitstapel der rekursiven Verarbeitung nachbilden. Definieren Sie dazu eine Operation mit der Signatur

```
static void quickSortIt(int[] zahlen),
```

wobei keine weiteren statischen Operationen benötigt und auch nicht zugelassen sind.

Testen Sie Ihre Funktion durch das Rahmenprogramm aus Aufgabe 18.

Aufgabe 20:

Modifizieren Sie die Implementierung des Sortierverfahrens HeapSort so, dass das Feld statt wie bisher aufsteigend, nunmehr absteigend sortiert wird. Definieren Sie dazu eine Operation mit der Signatur

```
static void heapSortDescending(int[] zahlen),
```

wobei die Implementierung einer weiteren statischen Operation analog zu `downheap` ausdrücklich zugelassen ist.

Testen Sie Ihre Funktion durch das Rahmenprogramm aus Aufgabe 18.

Challenge:

Ermitteln Sie die Anzahl der notwendigen Vergleiche und Zuweisungen, die von den jeweiligen Verfahren

- InsertionSort,
- SelectionSort,
- BubbleSort (optimierte Version mit vorzeitigem Abbruch),
- QuickSort (ganz klassisch, mit Pivot-Element aus der Mitte des Feldes),
- quickSort3M (aus Aufgabe 16, also mittels Dreier-Median),
- quickSort42 (ein QuickSort-Verfahren mit konstantem Pivot-Element 42) sowie für den
- HeapSort (bzw. heapSortDescending aus Aufgabe 18, wenn Sie diese bearbeitet haben)

die Anzahl der notwendigen Vergleiche und Zuweisungen bei Anwendung für die Sortierung der folgenden Zahlenfolge benötigt werden:

37, 14, 22, 41, 23, 12, 42, 19, 11, 43, 13, 20, 15, 21, 16, 24, 17,
27, 25, 18, 26, 30, 28, 33, 29, 31, 40, 32, 34, 39, 35, 38, 36, 10.

Als Vergleiche zählen alle arithmetischen Vergleiche von Feldinhalten (Schlüsseln), als Zuweisung jede Zuweisung eines Inhalts (Zahl) in eine Komponente des Feldes oder aus dem Feld in eine lokale Variable.

Diese Aufgabe können Sie gerne auch in Kleingruppen arbeitsteilig bearbeiten.

Viel Spaß und Erfolg bei der Bearbeitung der Aufgaben!