

Übungsblatt 8

Abgabe bis Freitag, 29.06.2012, 12:00 Uhr

Hinweis:

Aufgaben immer per E-Mail (eine E-Mail pro Blatt und Gruppe) an den zuständigen Tutor schicken (Bei Programmieraufgaben Java Quellcode und eventuell benötigte Datendateien).

Aufgabe 8.1

1. Schreiben Sie eine Klasse, die eine `ArrayList` für `Double`-Objekte als Instanzvariable enthält und implementieren Sie den entsprechenden Konstruktor sowie `set`- und `get`-Methoden.
2. Implementieren Sie eine Methode `bool swap(int idx1, int idx2)`, die die Elemente an den übergebenen Indizes vertauscht und `true` zurückgibt, wenn die Elemente erfolgreich vertauscht wurden. Werden Indizes übergeben, die nicht innerhalb der `ArrayList` liegen, soll `false` zurückgegeben werden.
3. Implementieren Sie eine Methode `void sort()`, die die Liste aufsteigend sortiert. Gehen Sie dabei wie folgt vor:
 - Iterieren Sie über die Liste und vergleichen Sie nacheinander jeweils zwei aufeinander folgende Elemente. Vertauschen Sie diese, falls sie nicht in der richtigen Ordnung vorliegen.
 - Wiederholen Sie diesen Vorgang maximal n mal, wobei n der Anzahl der Elemente in der Liste entspricht. Sie können den Algorithmus auch früher abbrechen, falls in einem kompletten Durchlauf keine Elemente vertauscht wurden.
4. Begründen Sie, warum die Liste nach aufruf der `sort`-Methode in sortierter Form vorliegt.
5. Führen Sie eine Aufwandsabschätzung in best- und worst case für die `sort`-Methode in Abhängigkeit der Länge der Liste durch.

Aufgabe 8.2

Betrachten Sie folgende Tabelle. In welche Komplexitätsklassen gehören die Funktionen $f_i(n)$ jeweils? Beachten Sie, dass eine Funktion auch mehreren Klassen angehören kann.

	$O(n)$	$O(n^4)$	$O(\log(n))$	$O(2^n)$	$O(4^n)$
$f_1(n) = \sqrt{n} \cdot n + \log(n) + n^3$					
$f_2(n) = n + \sqrt{n}$					
$f_3(n) = 2^{5+n}$					
$f_4(n) = 2^{n+n} + 1000$					
$f_5(n) = 9^n$					
$f_6(n) = n^4 + 7777 \cdot n^3$					
$f_7(n) = \frac{1}{n}$					

Aufgabe 8.3

1. Geben Sie die Definition der O -Notation für die asymptotische Komplexität an.
2. Betrachten Sie die folgende Methode:

```
static int myMethod(ArrayList<Double> a) {
    int n = a.size();
    int value = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i+1; j < n; j++) {
            if (a.get(i) == a.get(j)) {
                value += doSomething(a);
            }
        }
    }
    return value;
}
```

Führen Sie eine Aufwandsabschätzung in best- und worst case für diese Methode in Abhängigkeit von der Länge n des Arrays a durch. Die Methode `doSomething` hat eine Komplexität von $O(n)$. Gehen Sie davon aus, dass die Methode `get` der Klasse `ArrayList` eine konstante Laufzeit $O(1)$ hat.