Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Einführung in die Informatik
Sommersemester 2016

Prof. Dr. Wolfram Burgard
Alexander Schiotka
Andreas Kuhner

Übungsblatt 9

Abgabe bis Montag, 27.06.2016, 23:59 Uhr

Hinweis:

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

Aufgabe 9.1

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

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

$O(2^{n+1})$

$c >= 2^5$

$4^n = 2^{2^n} = 2^{2n} = 2^{n+n}$

Aufgabe 9.2

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;
}
```

$3^{2^n} = 3^{2n}$

Führen Sie eine Aufwandsabschätzung in best- und worst case für diese Methode in Abhängigkeit von der Länge n der ArrayList `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.

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

$\sqrt{n} \cdot n + \log(n) + n^3 \leq 3n^3$ } f_1

$\sqrt{n} \cdot n \uparrow n^3$ $\quad \log(n) + n^3 \uparrow n^3$

$\forall n > 10 : 3n^3 \leq c \cdot 2^n$ für $c = 3$

IS: von n nach $n+1$: $3(n+1)^3 = 3(n^3 + 3n^2 + 3n + 1)$

$\leq 3(8n^3)$ und $c \cdot 2^{n+1}$

IV $= c \cdot 2^n \cdot 2$

$\Rightarrow 3(8n^3) \leq 3(8 \cdot 2^n) \leq c \cdot 2^n \cdot 2$ für $c = 12$

$\Rightarrow 3 \cdot 8 \cdot n^3 \leq 2 \cdot 12 \cdot 2^n$

$\Rightarrow n^3 \in O(2^n)$

f_2: $n + n^{\frac{1}{2}} \leq 2n$

$1 + n^{\frac{-1}{2}} \leq 2$

$\searrow 0$

f_3: $2^{5+n} = 2^5 \cdot 2^n$

$\Rightarrow c >= 2^5$

f_4: $2^{n+n} = 2^{2n} = 2^{2^n} = 4^n$

f_5: $f_5(n) = 9^n$ $\quad 9^n > 8^n$

$O(2^n)$ $O(4^n)$

$8^n = 2^{3^n} = 2^{n+n+n} = (2^n)^3$

$8^n = (2 \cdot 4)^n = 2^n \cdot 4^n$

$\Rightarrow 9^n$

Aufgabe 9.3

Im Folgenden soll ein Sortieralgorithmus (BubbleSort) auf einer ArrayList implementiert werden.

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 aufeinanderfolgende 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.