Universität Stuttgart

Institut für Visualisierung
und Interaktive Systeme

Übungsblatt 2

Datenstrukturen und Algorithmen (SS 2017)

Abgabe: Mittwoch, 10.05.2017, 23:59 Uhr — Besprechung: ab Montag, 15.05.2017

Bitte lösen Sie die Übungsaufgabe in **Gruppen von 3 Studenten** und wählen EINEN Studenten aus, welcher die Lösung im ILIAS als **PDF** als **Gruppenabgabe** (unter Angabe aller Gruppenmitglieder) einstellt. Bitte erstellen Sie dazu ein **Titelblatt**, welches die Namen der Studenten, die Matrikelnummern, und die E-Mail-Adressen enthält.

Die Aufgaben mit Implementierung sind mit Impl gekennzeichnet. Das entsprechende Eclipse-Projekt kann im ILIAS heruntergeladen werden. Bitte beachten Sie die Hinweise zu den Implementierungsaufgaben, die im ILIAS verfügbar sind.[1]

Dieses Übungsblatt beinhaltet 3 Aufgaben mit einer Gesamtzahl von 30 Punkten.

Aufgabe 1 Verständnis *[Punkte: 6]*

(a) *(3 Punkte)* Zeichnen Sie folgenden Funktionen in ein gemeinsames Koordinatensystem ein:
$f_1(n) = n$
$f_2(n) = n^2 log(n)$
$f_3(n) = n!$
$f_4(n) = log(n)$
$f_5(n) = 2^n$
$f_6(n) = n^3$

(b) *(3 Punkte)* Geben Sie zu den Funktionen aus Teilaufgabe a) die *asymptotischen Komplexitäten* in O-Notation an und ordnen Sie diese *der Größe nach, beginnend mit der größten Komplexität.* Nennen Sie zu jeder Funktion zudem jeweils die entsprechende *Komplexitätsklasse.*

Aufgabe 2 Impl Sortierverfahren *[Punkte: 12]*

Gegeben im Eclipse-Projekt zu dieser Aufgabe ist das Interface `ISimpleList` für eine Liste, deren Elemente das Interface `Comparable` implementieren. Gegeben ist außerdem die Klasse `SimpleList`, die das Interface `ISimpleList` implementiert und im Rahmen dieser Aufgabe **nicht** zu verändern ist. Implementieren Sie die folgenden drei Sortierverfahren jeweils als statische Methode der Klasse `Sorter`, die jeweils eine Liste `ISimpleList` als Eingabeparameter erwarten, welche durch die Methode sortiert wird.

(a) *(4 Punkte) Selectionsort.* Dieses Verfahren sortiert die Liste in aufsteigender Reihenfolge, indem es das jeweils *kleinste* Element im unsortierten Teil der Liste sucht und es mit dem Anfang des unsortierten Teils der Liste vertauscht. (Ein ähnliches Beispiel findet sich in Foliensatz 3 auf Folie 13).

(b) *(4 Punkte) Bubblesort.* Dieses Verfahren sortiert die Liste in aufsteigender Reihenfolge, indem es beim Durchlaufen der unsortierten Liste jeweils das aktuelle Element mit seinem rechten Nachbarn vergleicht und diese vertauscht, falls das aktuelle Element größer als sein rechter Nachbar ist. Die Liste wird so oft durchlaufen, bis es keine Änderungen mehr gibt. (Ein Beispiel findet sich in Foliensatz 3 auf Folie 25).

(c) *(4 Punkte) Shakersort.* Dieses Verfahren ist eine Erweiterung des Bubblesort-Verfahrens. Statt die Liste in einer Richtung zu durchlaufen, läuft Shakersort in die entgegengesetzte Richtung,

[1] `https://ilias3.uni-stuttgart.de/goto_Uni_Stuttgart_fold_1214489.html`

sobald es den Anfang oder das Ende des unsortierten Teils der Liste erreicht. Der erste Durchlauf des Verfahrens läuft vom Anfang der Liste zu deren Ende, um das größte Element zu suchen und dieses ans Ende der Liste zu schieben. Im zweiten Durchlauf wird die Liste vom Ende zum Anfang durchlaufen, um das kleinste Element zu suchen und dieses an den Anfang der Liste zu verschieben. Die weiteren Durchläufe werden jeweils weiter im Wechsel durchgeführt.

Aufgabe 3 Asymptotische Komplexität *[Punkte: 12]*

Bestimmen Sie die *asymptotische Komplexität* der folgenden Algorithmen in *Abhängigkeit von n*. Dabei sei n jeweils eine *positive natürliche Zahl*. Begründen Sie Ihre Antwort kurz in maximal fünf Hauptsätzen. **Eine Antwort ohne Begründung wird mit null Punkten bewertet.**

(a) *(2 Punkte)* Gegeben ist der folgende Algorithmus `alg1`.

```
1 public void alg1(int n) {
2         int result = 0;
3         for (int i = 0; i < n; i++) {
4                 result--;
5         }
6         for (int j = n - 1; j >= 0; j--) {
7                 result++;
8         }
9 }
```

(b) *(2 Punkte)* Gegeben ist der folgende Algorithmus `alg2`.

```
1  public void alg2(int n) {
2          int result = 1;
3          while (result < n) {
4                  if (result >= n / 2) {
5                          result = n;
6                  } else {
7                          result = result * 2;
8                  }
9          }
10 }
```

(c) *(2 Punkte)* Gegeben ist der folgende Algorithmus `alg3`.

```
1 public int alg3(int n) {
2         int summe = 0;
3         for (int i = n; i >= 1; i = i / 2) {
4                 for (int j = 0; j < 100; j++) {
5                         summe++;
6                 }
7         }
8         return summe;
9 }
```

(d) *(2 Punkte)* Gegeben ist der folgende Algorithmus `alg4`.

```
1 public int alg4(int n) {
2         int summe = 0;
3         if (n > 1) {
4                 summe = alg4(n - 1) + 1;
5         } else {
6                 summe = 1;
7         }
8         return summe;
9 }
```

(e) *(2 Punkte)* Gegeben ist der folgende Algorithmus `alg5`.

```
1  public int alg5(int n) {
2          if (n == 0) {
3                  return 0;
4          }
5
6          int i = 1;
7          int j = n;
8          while (i < j) {
9                  i = i + 1;
10                 j = j - 1;
11         }
12
13         return alg5(i - 1) + 1;
14
15 }
```

(f) *(2 Punkte)* Gegeben ist der folgende Algorithmus `alg6`.

```
1  public int alg6(int n) {
2          int sum = 0;
3          for (int i = 0; i < n; i++) {
4                  for (int j = n; j > 0; j--) {
5                          for (int k = 0; k < n / 2; k++) {
6                                  sum = sum + 1;
7                          }
8                  }
9          }
10         for (int l = 0; l < n; l++) {
11                 sum = sum - 1;
12         }
13         return sum;
14 }
```