



Übungsblatt 2

Datenstrukturen und Algorithmen (SS 2018)

Abgabe: Mittwoch, 9.05.2018, 23:55 Uhr — Besprechung: ab Montag, 14.05.2018

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 Sortierv Verfahren [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 Sortierv Verfahren 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 11).
- (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 21).
- (c) (4 Punkte) *Quicksort*. Dieses Verfahren verwendet das Teile-und-Herrsche-Prinzip um die Liste in aufsteigender Reihenfolge zu sortieren. Es wird ein beliebiges Pivotelement gewählt (z.B. das

¹https://ilias3.uni-stuttgart.de/goto_Uni_Stuttgart_crs_1432415.html

Element in der Mitte) um die zu sortierende Liste in zwei Teillisten aufzuteilen. Alle Elemente, die kleiner als das Pivotelement sind, werden in die linke Teilliste geschoben; alle Elemente, die größer sind, in die rechte Teilliste. Die Teillisten werden durch Rekursion weiter sortiert. (Eine Umsetzungsvariante findet sich in Foliensatz 3 ab Folie 37).

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 = 1000;
3     for (int i = 2 * n; i > 0; i--) {
4         result++;
5     }
6     for (int j = 0; j < n; j = j + 2) {
7         result++;
8     }
9 }
```

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

```
1 public int alg2(int n) {
2     int result = 10;
3     for (int i = 1; i <= n; i++) {
4         for (int j = n; j > 0; j = j - 2) {
5             for (int k = 0; k < n / 2; k = k + 3) {
6                 result = result + k;
7             }
8         }
9     }
10    return result;
11 }
```

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

```
1 public int alg3(int n) {
2     if (n <= 1) {
3         return n;
4     }
5     return alg3(n - 1) + alg3(n - 1);
6 }
```

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

```
1 public void alg4(int n) {
2     int result = 1;
3     while (result < n) {
4         if (result >= n / 2) {
5             result = n;
6         }
7         else {
8             result = result * 2;
9         }
10    }
11 }
```

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

```
1  public void alg5(int n) {
2      int result = 0;
3      n = n / 2;
4      for (int i = 1; i <= n; i++) {
5          for (int j = 1; j < n; j = j * 2) {
6              result = result + 1;
7          }
8      }
9  }
```

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

```
1  public void alg6 (int n) {
2      int x = 9999;
3      for (int i = n; i >= 1; i--) {
4          int result = i;
5          for (int j = 0; j < x; j++) {
6              result++;
7          }
8      }
9  }
```