

## Übungsblatt 8

Abgabe bis Montag, 30.06.08, 16 Uhr

### Hinweis:

Programmieraufgaben immer per Email (eine Email pro Blatt und Gruppe) an den zuständigen Tutor schicken (Java Quellcode und eventuell benötigte Datendateien). Bitte werfen Sie Ihre schriftlichen Lösungen in die Briefkästen in Geb. 051, Erdgeschoss ein. Für den Erhalt von Bonuspunkten müssen Sie in wenigstens 9 Übungen anwesend sein und müssen wenigstens 9 Übungszettel bearbeitet haben.

### Aufgabe 8.1

1. Schreiben Sie eine rekursive Java Methode `fakultaet`, die zu einer Eingabe  $n \in \mathbb{N}$  die Fakultät ( $n! = 1 \cdot 2 \cdot \dots \cdot n$ ) berechnet.
2. Schreiben Sie ein Java Programm, das die Fakultät für alle Zahlen zwischen 1 und 10 berechnet und auf dem Bildschirm ausgibt.

### Aufgabe 8.2

Betrachten Sie die Definition der O-Notation für asymptotische Komplexität. Welche der folgenden Aussagen trifft zu? Korrigieren Sie die falschen Aussagen.

1. Das Symbol  $f$  in  $O(f)$  steht für eine reelle Zahl.
2. Die Variable  $n$  in der Definition steht für die Anzahl der Programmzeilen, falls ein Java Programm untersucht werden soll.
3.  $O(f)$  beschreibt eine Menge von Funktionen. Eine typische Fragestellung in der Informatik ist, ob eine Funktion  $g(n)$ , die den Aufwand eines Verfahrens in Abhängigkeit von der Problemgröße  $n$  beschreibt, in dieser Menge liegt:  $g \in O(f)$ .
4. Aufgrund der Konstanten  $c$  in der Definition liegen die Funktionen  $n^{15}$  und  $15 \cdot n^{15}$  in derselben Komplexitätsklasse, die Funktionen  $n^{16}$  und  $(16 \cdot n)^{16}$  dagegen nicht.
5. Es gilt  $O(f) = O(u)$  für  $f(x) = \pi + \pi \cdot x^2$  und  $u(n) = (n + \pi) \cdot (n - \pi)$ .
6. Es gilt  $p \in O(n^2)$  für  $p(n) = (0^n + 1^n) \cdot (n \cdot \pi) \cdot (n \cdot \pi)$ .
7. Das Sortieren eines Arrays hat stets denselben Aufwand in der O-Notation, egal welches Sortierverfahren verwendet wird.

8. Der Best-Case Aufwand eines Verfahrens liegt immer in  $O(1)$ .

### Aufgabe 8.3

Betrachten Sie die folgenden beiden Programmstücke und führen Sie jeweils eine Aufwandsabschätzung in Abhängigkeit von der Anzahl der Elemente in dem entsprechenden Vektor  $v$  durch.

```
for (i=1; i<v.size()-1; i+=2) {
    double sum = 0;
    for (j=i-1; j<i+2; j++)
        sum += ((Double) v.elementAt(j)).doubleValue();
    if (sum > 0)
        v.setElementAt(new Double(((Double)
            v.elementAt(i)).doubleValue()/sum), i);
}
```

```
for (i=0; i<v.size(); i++) {
    double sum = 0;
    for (j=i+1; j<v.size(); j++)
        sum += ((Double) v.elementAt(j)).doubleValue();
    if (sum > 0)
        v.setElementAt(new Double(((Double)
            v.elementAt(i)).doubleValue()/sum), i);
}
```