

Übungsblatt 9

Abgabe bis Mittwoch, 01.07.09, 18 Uhr

Aufgabe 9.1

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 9.2

1. Schreiben Sie ein Java-Programm, welches alle Zahlen, die in einem Vektor gespeichert sind, normalisiert. Gehen Sie davon aus, dass die Zahlen vom Typ `Double` sind.
2. Führen Sie eine Aufwandsabschätzung für die Laufzeit in Abhängigkeit von der Anzahl n der Elemente im Vektor durch.

Hinweis: Normalisieren bedeutet, dass jeder Wert durch die Summe aller Werte geteilt wird. Nach der Normalisierung summieren sich alle Werte zu 1 auf.

Aufgabe 9.3

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);
}
```

Aufgabe 9.4

Jede symmetrische positiv definite Matrix $A \in \mathbb{R}^{n \times n}$ kann eindeutig in der Form

$$A = LL^T$$

geschrieben werden. Dabei ist L eine untere Dreiecksmatrix und kann mittels Cholesky-Zerlegung bestimmt werden¹. Die Elemente der Dreiecksmatrix L lassen sich wie folgt bestimmen:

$$L_{ii} = \left(a_{ii} - \sum_{k=0}^{i-1} L_{ik}^2 \right)^{1/2}$$

$$L_{ji} = \frac{1}{L_{ii}} \left(a_{ij} - \sum_{k=0}^{i-1} L_{ik} L_{jk} \right) \quad j = i + 1, i + 2, \dots, n - 1$$

Implementieren Sie diese Zerlegung. Erweitern Sie dazu die Klasse `Matrix` (siehe Homepage) um die Methode `public Matrix chol()`. Führen Sie anschließend eine Aufwandsabschätzung ihrer Implementierung in Abhängigkeit von n durch.

¹siehe auch „Taschenbuch der Mathematik“, Bronstein, Semendjajew, Musiol und Mühlig