InfTech2 - Hausaufgabe 3 - T02G01

Aufgabe 1 – Komplexität

```
1       void sort(int[] a) {
2           int n = a.length;
3               for (int i = 0; i < n; i++) {
4                   for (int j = 0; j < n−1; j++) {
5                       if(a[j] > a[j+1]) {
6                           int tmp = a[j];
7                           a[j] = a[j+1];
8                           a[j+1] = tmp;
9                       }
10                  }
11              }
12 }
```

1) In den Zeilen 6-8 finden sich ***drei*** Zuweisungen. Wenn der Methode also ein Array der Länge n übergeben wird, durchläuft diese n-mal die erste for-Schleife, (n-1)-mal die zweite for-Schleife und, wenn die if-Bedingung greift (wovon wir ausgehen), gibt es drei Zuweisungen.
Daraus folgt:

T(n) = n*(n-1)*3 (das entspricht $T(3n^2)$)

[Eigentlich: T(n) = 3 + 3* n*(n-1), wenn man die Zuweisungen von n, i und j mitzählt.]

2) Für den Worst-Case muss das Array genau anders herum sortiert sein, also vom höchsten Wert (am Index 0) absteigend.

3) Damit der Best-Case eintritt, muss das Array bereits richtig (also aufsteigend) sortiert sein.
In diesem Fall werden die drei Zuweisungen in den Zeilen 6-8 nie ausgeführt.
Da wir uns in diesem Beispiel auf die Zeilen 6-8 beschränken und keine anderen Zuweisungen betrachten, gibt es somit keinen Zeitaufwand im Best-Case. (T(n) = 0)

4) Zählt man den Vergleich in Zeile 5 mit, so gibt es jede „Runde“ einen Vorgang.
Wenn wir also wieder die beiden for-Schleifen mitzählen, erhalten wir:

T(n) = n*(n-1)*1

5) Man könnte die zweite for-Schleife (Zeile 4) ändern in:

```
for (int j=0; j < n-i-1; j++) {
```

So wird immer noch dafür gesorgt, dass das Array sortiert werden kann, da der höchste Wert immer ganz nach hinten „geschoben“ wird.