

# Algorithmen und Datenstrukturen 1

## Serie 1

1.  $g(n)$  ist in der Tabelle mit X markiert wenn  $g(n)$  eine Abschätzung von  $n$  ist:

$f(n)$	$g(n)$	$O(g(n))$	$\Omega(g(n))$	$\Theta(g(n))$
$\sqrt[4]{\frac{1}{2}} + \sqrt[3]{4n}$	$\sqrt[3]{4n} + 8\sqrt[2]{n}$	X	-	-
$\log_8 n - \log_{10} n$	$\log_6 n - \log_8 n$	X	X	X
$n^3 \log n$	$n^3 \log n^2$	X	X	X
$(n+1)(n+2)$	$(n-1)(n-2)$	X	X	X
$n + \frac{n}{2} + \frac{n}{3}$	$n \log n$	X	-	-
$n \log n + n^{1.8}$	$n\sqrt{n}$	-	X	-

2. Der zu implementierende Javamethode:

```
public static Comparable getThirdLargestElement(Comparable[] array)
{
    Comparable big1= array[0];
    Comparable big2= null;
    Comparable big3= null;

    for (int i= 1; i < array.length; i++) {
        if ((array[i].compareTo(big3) => 0) || (big3 == null)) {
            if ((array[i].compareTo(big2) => 0) || (big2 == null)) {
                if (array[i].compareTo(big1) => 0) {
                    big3= big2;
                    big2= big1;
                    big1= array[i];
                } else {
                    big3= big2;
                    big2= array[i]
                }
            } else {
                big3= array[i];
            }
        }
    }

    return big3;
}
```

Die Zeitkomplexität ist  $O(6n)$ .

3. a) Die Laufzeit der inneren for-Schleife ist  $O(n^2)$ , da  $g$  von dieser Grössenordnung ist. Die äussere Schleife wird immer  $n$ -mal durchlaufen, deshalb ist die Komplexität hier  $O(n)$ . Die Verzweigung im innersten hat  $O(1)$ + Kosten der längsten Alternative, somit  $O(const \cdot n + 1)$ .

Wir haben hier also je nach der Grössenordnung von  $f$  eine Komplexität von:

$$O(a) = n \cdot n^2 \cdot (const \cdot n + 1) = const \cdot n^4 + n^3$$

$$\Omega(a) = n \cdot n^2 \cdot (const \cdot n + 1) = const \cdot n^4 + n^3$$

$$O(a) = \Omega(a) = \Theta(a)$$

- b) Die Laufzeit der inneren for-Schleife ist  $O(n)$ , da  $g$  von dieser Grössenordnung ist. Die äussere Schleife wird immer  $n$ -mal durchlaufen, deshalb ist die Komplexität hier  $O(n)$ . Die Verzweigung im innersten hat  $O(1)$ + Kosten der längsten Alternative.

Wir haben hier also je nach der Grössenordnung und der Komplexität von  $f$ :

$$O(b) = n \cdot n \cdot (n^2 \cdot n \log n + 1) = n^2(n^3 \log n + 1)$$

$$\Omega(b) = n \cdot n \cdot (n \cdot n \log n + 1) = n^2(n^2 \log n + 1)$$

4. a) Ja,  $O(n^2)$  ist eine Abschätzung nach oben.  
 b)  $O(n\sqrt{n})$   
 c)  $\Omega(n\sqrt{\log n})$