## Beispiel-Aufgabe zum Master-Theorem

(a) Betrachten Sie die folgende Methode m in Java, die initial m(r, 0, r.length) für das Array r aufgerufen wird. Geben Sie dazu eine Rekursionsgleichung T(n) an, welche die Anzahl an Rechenschritten von m in Abhängigkeit von der Länge n = r.length berechnet.

```
public static int m(int[] r, int lo, int hi) {
      if (lo < 8 || hi <= 10 || lo >= r.length || hi > r.length) {
       throw new IllegalArgumentException();
     if (hi - lo == 1) {
       return r[lo];
     } else if (hi - lo == 2) {
       return Math.max(r[lo], r[lo + 1]); // 0(1)
     } else {
       int s = (hi - lo) / 3;
11
12
       int x = m(r, lo, lo + s);
       int y = m(r, lo + s, lo + 2 * s);
13
       int z = m(r, lo + 2 * s, hi);
14
15
        return Math.max(Math.max(x, y), 2); // 0(1)
16
   }
17
```

```
Allgemeine Rekursionsgleichung: T(n) = a \cdot T(\frac{n}{b}) + f(n)
a (Anzahl der rekursiven Aufrufe): 3
b (Um welchen Anteil wird das Problem durch den Aufruf verkleinert): um \frac{1}{3} also b = 3
Für den obigen Code: T(n) = 3 \cdot T(\frac{n}{3}) + \mathcal{O}(1)
1. Fall: f(n) \in \mathcal{O}\left(n^{\log_3 3 - \varepsilon}\right) = \mathcal{O}\left(n^{1 - \varepsilon}\right) = \mathcal{O}\left(1\right) für \varepsilon = 1
2. Fall: f(n) \notin \Theta\left(n^{\log_3 3}\right) = \Theta\left(n^1\right)
3. Fall: f(n) \notin \Omega\left(n^{\log_3 3 + \varepsilon}\right) = \Omega\left(n^{1 + \varepsilon}\right)
Also: T(n) \in \Theta\left(n^{\log_3 a}\right)
```