

Staatsexamen 66115 / 2018 / Frühjahr / Thema Nr. 2 / Aufgabe Nr. 6

## Aufgabe 6 [Methode „m()“]

Der Hauptsatz der Laufzeitfunktionen ist bekanntlich folgendermaßen definiert:

### Exkurs: Master-Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$a$  = Anzahl der Unterprobleme in der Rekursion, der rekursiven Aufrufe ( $a \geq 1$ ).

$\frac{1}{b}$  = Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird, Anteil an der Verkleinerung des Problems ( $b > 1$ ).

$f(n)$  = Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen. Eine von  $T(n)$  unabhängige und nicht negative Funktion.

Dann gilt:

1. Fall:  $T(n) \in \Theta(n^{\log_b a})$

falls  $f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon})$  für  $\varepsilon > 0$

2. Fall:  $T(n) \in \Theta(n^{\log_b a} \cdot \log n)$

falls  $f(n) \in \Theta(n^{\log_b a})$

3. Fall:  $T(n) \in \Theta(f(n))$

falls  $f(n) \in \Omega(n^{\log_b a + \varepsilon})$  für  $\varepsilon > 0$  und ebenfalls für ein  $c$  mit  $0 < c < 1$  und alle hinreichend großen  $n$  gilt:  $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

- (a) Betrachten Sie die folgende Methode `m` in Java, die initial mit `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.

```

5  public static int m(int[] r, int lo, int hi) {
6      if (lo < 8 || hi <= 10 || lo >= r.length || hi > r.length) {
7          throw new IllegalArgumentException();
8      }
9
10     if (hi - lo == 1) {
11         return r[lo];
12     } else if (hi - lo == 2) {
13         return Math.max(r[lo], r[lo + 1]); // O(1)
14     } else {
15         int s = (hi - lo) / 3;
16         int x = m(r, lo, lo + s);
17         int y = m(r, lo + s, lo + 2 * s);
18         int z = m(r, lo + 2 * s, hi);
19         return Math.max(Math.max(x, y), z); // O(1)
20     }
21 }
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2018/fruehjahr/MasterTheorem.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2018/fruehjahr/MasterTheorem.java)

### Allgemeine Rekursionsgleichung:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

### Anzahl der rekursiven Aufrufe ( $a$ ):

3

### Anteil Verkleinerung des Problems ( $b$ ):

um  $\frac{1}{3}$  also  $b = 3$

### Laufzeit der rekursiven Funktion ( $f(n)$ ):

$$\mathcal{O}(1)$$

**Ergibt folgende Rekursionsgleichung:**

$$T(n) = 3 \cdot T\left(\frac{n}{3}\right) + \mathcal{O}(1)$$

- (b) Ordnen Sie die rekursive Funktion  $T(n)$  aus (a) einem der drei Fälle des Mastertheorems zu und geben Sie die resultierende Zeitkomplexität an. Zeigen Sie dabei, dass die Voraussetzung des Falles erfüllt ist.

**1. Fall:**  $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$ :

$$f(n) \in \mathcal{O}\left(n^{\log_3 3 - \varepsilon}\right) = \mathcal{O}\left(n^{1 - \varepsilon}\right) = \mathcal{O}(1) \text{ für } \varepsilon = 1$$

**2. Fall:**  $f(n) \in \Theta\left(n^{\log_b a}\right)$ :

$$f(n) \notin \Theta\left(n^{\log_3 3}\right) = \Theta\left(n^1\right)$$

**3. Fall:**  $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$ :

$$f(n) \notin \Omega\left(n^{\log_3 3 + \varepsilon}\right) = \Omega\left(n^{1 + \varepsilon}\right)$$

$$\text{Also: } T(n) \in \Theta\left(n^{\log_b a}\right)$$

Github: Staatsexamen/66115/2018/03/Thema-2/Aufgabe-6.tex