Master-Theorem

Der Hauptsatz der Laufzeitfunktionen – oder oft auch aus dem Englischen als Master-Theorem entlehnt – bietet eine *schnelle Lösung* für die Frage, *in welcher Laufzeitklasse* eine gegebene *rekursiv definierte Funktion* liegt. Mit dem Master-Theorem kann allerdings *nicht jede rekursiv definierte Funktion* gelöst werden. Lässt sich keiner der *drei möglichen Fälle* des Master-Theorems auf die Funktion T anwenden, so muss man die Komplexitätsklasse der Funktion anderweitig berechnen.

schnelle Lösung
in welcher Laufzeitklasse
rekursiv definierte Funktion
nicht jede rekursiv definierte Funktion
drei möglichen Fälle

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

$a =$ Anzahl der Unterprobleme in der Rekursion $a \geq 1$

$\frac{1}{b} =$ Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird $b > 1$

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

[1] [2]

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

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

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

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

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

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

Literatur

[1] *Qualifizierungsmaßnahme Informatik: Algorithmen und Datenstrukturen 2. Sortieren, Suchen, Komplexität*. `https://www.studon.fau.de/file2566441_download.html`.

[2] *Wikipedia-Artikel „Master-Theorem“*. `https://de.wikipedia.org/wiki/Master-Theorem`.

[1] *Wikipedia-Artikel „Master-Theorem“*.
[2] *Qualifizierungsmaßnahme Informatik: Algorithmen und Datenstrukturen 2*, Seite 19-35.