Materialien zur Vorlesung Algorithmen und Datenstrukturen

(Prof. Der)

Komplexität von Algorithmen - Mastertheorem und Mastermethode

Die Mastermethode dient der Lösung von Rekursionsgleichungen der Form

$$T(n) = aT\left(\frac{n}{b}\right) + g(n) \qquad a \geq 1, b > 1$$

$g(n)$ ist eine Funktion über den natürlichen Zahlen. Wir setzen

$$u = \log_b a$$

wobei $\log_b$ der Logarithmus zur Basis b ist. Dann ergeben sich für $T(n)$ die folgenden asymptotischen Grenzen.

1. Falls $g(n) = O\left(n^{u-\varepsilon}\right)$ für ein $\varepsilon > 0$ dann ist

$$T(n) = \Theta\left(n^u\right)$$

2. Falls $g(n) = \Theta\left(n^u\right)$ dann ist

$$T(n) = \Theta\left(n^u \log_2 n\right)$$

3. Falls $g(n) = \Omega\left(n^{u+\varepsilon}\right)$ für ein $\varepsilon > 0$ und

$$ag\left(\frac{n}{b}\right) \leq cg(n)$$

für ein $c < 1$ und genügend große n dann ist

$$T(n) = \Theta(g(n))$$

Beispiel 1: Beim Verfahren des Heapsort ist die Komplexität für die Herstellung eines Heaps durch die Rekursiongleichung

$$T(n) \leq T(2n/3) + \Theta(1)$$

gegeben. Es ist also $a = 1$ und $b = 3/2$. Damit ist der Fall 2 des Mastertheorems gegeben, denn es ist

$$g(n) = \Theta(1) = \Theta\left(n^{\log_b a}\right)$$

weil $u = \log_b a = 0$ wenn $a = 1$ (bei beliebigem b).

Somit

$$T(n) = \Theta\left(n^0 \log_2 n\right) = \Theta(\log_2 n)$$

+++++++++++

Für den Umgang mit den Logarithmen sind die folgenden Relationen wichtig

$$\begin{aligned} \log(ab) &= \log a + \log b \\ \log a^m &= m \log a \end{aligned}$$

Das gilt bei beliebiger Basis. Außerdem:

$$\log_b b = 1$$

$$\log_b a = \frac{\ln g}{\ln b} \log_g a$$

wobei ln der natürliche Logarithmus ist. Der Übergang von einer Basis zur anderen ist also durch einen konstanten Faktor vermittelt.