Das Master-Theorem

Bei rekursiven Algorithmen kann die Abschätzung der Laufzeit in den meisten Fällen zunächst nur mit einer rekursiven Gleichung angegeben werden.

So hatten wir zum Beispiel bei Mergesort die Rechenzeit

$$T(n) = 2 \cdot T(n/2) + \Theta(n)$$

Wie kann man hieraus eine geschlossene Formel erhalten?

Die Antwort gibt uns das sogenannte *Master-Theorem*, das wir nun formulieren und beweisen wollen.

Beachte: Das hier verwendete und bewiesene Master-Theorem wird im Allgemeinen als *Master-Theorem I* bezeichnet.

Es gibt auch noch ein *Master-Theorem II*, das wir nur angeben, aber nicht beweisen werden.

Die Aussage

Satz:

Für $a, b \in \mathbb{N}$, $b > 1$ und eine Funktion $g : \mathbb{N} \to \mathbb{N}$ mit $g \in \Theta(n^c)$ gelte

1) $t(1) = g(1)$
2) $t(n) = a \cdot t(n/b) + g(n)$

Dann gilt:

(i)	$t(n) \in \Theta(n^c)$	falls $a < b^c$
(ii)	$t(n) \in \Theta(n^c \log n)$	falls $a = b^c$
(iii)	$t(n) \in \Theta(n^{(\log a)/(\log b)})$	falls $a > b^c$

Anwendungen

Bevor wir das Master-Theorem beweisen, wollen wir es auf die Fälle *Mergesort* und *Multiplikation großer Zahlen* anwenden.

Mergesort:

Wir haben $g(n) = n$ und $T(n) = 2 \cdot T(n/2) + n$,
also $a = b = 2$ und $c = 1$.
Damit sind wir im Fall (ii) mit $c = 1$ und erhalten
als Laufzeit $T(n) = \Theta(n \log n)$.

Multiplikation großer Zahlen:

Jetzt ist $T(n) = 3 \cdot T(n/2) + n$, also
$a = 3$, $b = 2$, $c = 1$, damit Fall (iii) und
$T(n) \in \Theta(n^d)$ mit $d = \log_2 3 \approx 1.59$.

Beweis

Wir zeigen zunächst, dass unter den Voraussetzungen des Satzes mit $k = \log_b n$ gilt:

$$t(n) = \sum_{i=0}^{k} a^i \cdot g(\tfrac{n}{b^i})$$

Wir führen eine Induktion über k und beachten $n = b^k$:

Induktionsanfang $k = 0$:

Es gilt $n = 1$ und wir müssen zeigen, dass gilt:

$$t(1) = \sum_{i=0}^{0} a^i \cdot g(\tfrac{1}{b^i}),$$

aber das folgt aus $\sum_{i=0}^{0} a^i \cdot g(\frac{1}{b^i}) = a^0 \cdot g(\frac{1}{b^0}) = g(1)$

und der Voraussetzung $t(1) = g(1)$.

Beweis: Induktionsschritt

Nun sei $k > 0$. Nach Induktionsvoraussetzung gilt

$$t(b^{k-1}) = \sum_{i=0}^{k-1} a^i \cdot g\left(\frac{b^{k-1}}{b^i}\right)$$

Das setzen wir ein in $t(n) = a \cdot t(n/b) + g(n)$ und erhalten:

$$\begin{aligned}
t(n) &= a \cdot t\left(\tfrac{n}{b}\right) + g(n) \qquad \text{Beachte: } \tfrac{n}{b} = \tfrac{b^k}{b} = b^{k-1} \\
&= a \cdot \left(\sum_{i=0}^{k-1} a^i \cdot g\left(\frac{b^{k-1}}{b^i}\right) \right) + g(n) \\
&= \left(\sum_{i=1}^{k} a^i \cdot g\left(\tfrac{n}{b^i}\right) \right) + a^0 \cdot g\left(\tfrac{n}{b^0}\right) \\
&= \sum_{i=0}^{k} a^i \cdot g\left(\tfrac{n}{b^i}\right)
\end{aligned}$$

q.e.d.

Letzter Schritt

Wir benutzen die gezeigte Gleichung $t(n) = \sum_{i=0}^{k} a^i \cdot g(\frac{n}{b^i})$,
und verwenden statt $g(n) \in \Theta(n^c)$ einfacher $g(n) = n^c$.

Man kann leicht prüfen, dass Konstanten hier allgemein kein Problem darstellen.

Es folgt $t(n) = \sum_{i=0}^{k} a^i \cdot (\frac{n}{b^i})^c = n^c \cdot \sum_{i=0}^{k} (\frac{a}{b^c})^i \in O(n^c)$ für $a < b^c$.

Mit $g(n)$ ist auch $t(n)$ in $\Omega(n^c)$, also folgt $t(n) \in \Theta(n^c)$ für $a < b^c$, und damit Behauptung (i) des Satzes.

Für $a = b^c$ erhält man aus obigem $t(n) = n^c \cdot (k+1) \in \Theta(n^c \log n)$. Das ergibt Fall (ii). Fall (iii) folgt so:

$$t(n) = n^c \cdot \sum_{i=0}^{k} (\frac{a}{b^c})^i = n^c \cdot \frac{(\frac{a}{b^c})^{k+1} - 1}{\frac{a}{b^c} - 1}.$$

Also gilt hier $t(n) \in \Theta(n^c \cdot (\frac{a}{b^c})^{\log_b n}) = \Theta(a^{\log_b n}) = \Theta(n^{\frac{\log a}{\log b}})$.

Die Gleichungen dieser Zeile sind einfach zu zeigen – der Leser möge sie selbst erarbeiten.