Karlsruher Institut für Technologie

Tutorium 17, #11

Max Göckel– *uzkns@student.kit.edu*

Institut für Theoretische Informatik - Grundbegriffe der Informatik

Algorithmen: O-Kalkül

Um Algorithmen zu vergleichen und einzuordnen gibt es verschiedene Schranken:

- Obere Schranke O (Groß-O): In $O(f(n))$ sind alle Funktionen die langsamer oder gleich schnell wie f wachsen.
  - $\{g(n) | \exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : g(n) \leq c \cdot f(n)\}$
- Untere Schranke Ω (Omega): In $\Omega(f(n))$ sind alle Funktionen die schneller oder gleich schnell wie f wachsen.
  - $\{g(n) | \exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}$
- Mittlere Schranke Θ (Theta): In $\Theta(f(n))$ sind alle Funktionen die genau gleich schnell wie f wachsen.
  - $\{g(n) | \exists c_1, c_2 \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : c_1 f(n) \leq g(n) \leq c_2 f(n)\}$
  - $O(f(n)) \cap \Omega(f(n))$

O-Kalkül: Beispiel

Sei $f(n) = \Pi \cdot n^{10}$ und $g(n) = \frac{1}{e^9} \cdot n^{10}$.

Finde passende c, n_0 um das Laufzeitverhalten von f und g zu vergleichen.

O-Kalkül: Beispiel

Sei $f(n) = \Pi \cdot n^{10}$ und $g(n) = \frac{1}{e^9} \cdot n^{10}$.

- $f(n) = \Pi n^{10} \leq (e^9 \Pi) \cdot \frac{1}{e^9} n^{10} = e^9 \Pi \cdot g(n)$
  - $f(n) \in O(g(n))$ mit $n_0 = 1$ und $c = e^9 \Pi$
- $g(n) = \frac{1}{e^9} \cdot n^{10} \leq n^{10} \leq \Pi n^{10} = 1 \cdot f(n)$
  - $g(n) \in O(f(n))$ mit $n_0 = 1$ und $c = 1$

Also ist $g(n) \in \Theta(f(n))$.

O-Kalkül: Aufgaben

- $f(n) = 5n^2 + 3$ und $g(n) = \frac{1}{2}n^2$
- $f(n) = 3^n$ und $g(n) = 5^n$
- $f(n) = 3n^7 + 4n^6 - n^3 + n$ und $g(n) = 2n^7 - n^5 + 3n^2$

Finde passende c, n_0 um das Laufzeitverhalten von f und g zu vergleichen.

Mastertheorem

Das Mastertheorem ermöglicht Laufzeitabschätzungen für rekursive Algorithmen.
Die Funktionen der Algorithmen müssen folgende Form besitzen, damit das Mastertheorem angewendet werden kann:
$T(n) = a \cdot T(\frac{n}{b}) + f(n)$.

Fakultät für Informatik
Institut für Theoretische Informatik

Mastertheorem

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

- a: Anzahl der Teilprobleme
- $\frac{n}{b}$: Größe der Teilprobleme
- $f(n)$: Von $T(n)$ unabhängige Funktion zur Kombination der Teilprobleme

Bsp.: $T_1(n) = a \cdot T_1(\frac{n}{10}) + 3n^2$

Mastertheorem: Fälle

Bei der Laufzeit der Form $T(n) = a \cdot T(\frac{n}{b}) + f(n)$ gibt es 3 Fälle:

1. Ist $f(n) \in O(n^{log_b a - \epsilon})$ für ein $\epsilon > 0 \Rightarrow T(n) \in \Theta(n^{log_b a})$
2. Ist $f(n) \in \Theta(n^{log_b a}) \Rightarrow T(n) \in \Theta(n^{log_b a} \cdot log(n))$
3. Ist $f(n) \in \Omega(n^{log_b a + \epsilon})$ für ein $\epsilon > 0$ und $\exists d \in (0, 1)$, sodass für ein großes n gilt: $a \cdot f(\frac{n}{b}) \leq d \cdot f(n) \Rightarrow T(n) \in \Theta(f(n))$

Mastertheorem: Aufgaben

1. $T(n) = 2T(\frac{n}{2}) + 10n$
2. $20n^2 + T(\frac{n}{2}) \cdot 8$
3. $4T(\frac{n}{4}) + n^2$

Mastertheorem: Lösung

1. $a = 2, b = 2, f(n) = 10n; log_b a = 1;$
   $f(n) = 10n \in \Theta(n^1) \Rightarrow T(n) \in \Theta(n \cdot logn)$
2. $a = 8, b = 2, f(n) = 20n^2; log_b a = 3;$
   $f(n) = 20n^2 \in O(n^{log_b a - \epsilon}) = O(n^{3-\epsilon})$ für $\epsilon = \frac{1}{2} \Rightarrow T(n) \in \Theta(n^3)$

Fakultät für Informatik
Institut für Theoretische Informatik

Mastertheorem: Lösung der 3.

$a = 4, b = 4, f(n) = n^2,$
$log_b a = 1,$
$f(n) = n^2 \in \Omega(n^{log_b a + \epsilon}) = \Omega(n^{1+\epsilon}),$
Für $\epsilon = \frac{1}{2}$. Es soll gelten $af(\frac{n}{b}) \leq df(n), d \in (0, 1)$.

$af(\frac{n}{b}) = 4f(\frac{n}{4}) = \frac{n^2}{4} \leq dn^2 = df(n)$ mit $d = \frac{1}{4}$.
$\Rightarrow T(n) \in \Theta(n^2)$

Klausuraufgaben

Um der Vorlesung nicht zu weit "voraus" zu sein, machen wir jetzt ein paar Klausuraufgaben zu Themen die wir schon hatten.