Einführung in die Informatik I (autip)

Dr. Stefan Lewandowski

Fakultät 5: Informatik, Elektrotechnik und Informationstechnik
Abteilung Formale Konzepte
Universität Stuttgart

11.01.2006 / Version 12. Januar 2006

dies ist nur eine Kurzübersicht, damit Sie die Definitionen nochmal anschauen können und an den Übungsaufgaben versuchen können diese nachzuvollziehen – ich werde die ausführliche Variante baldmöglichst in den zweiten Teil des Skripts integrieren

4 Aufwandsabschätzungen – die O-Notation

- bisher: Angabe für den Aufwand meist in der Form „proportional zu $f(n)$“ – bei Beispielen Minimumsuche ok, bei gerechter Erbschaft bedingt ok, bei Intervallschachtelung (binärer Suche) weniger geeignet → formale Lösung: O-Notation.
- Hauptsächliche Motivation zur O-Notation: Bewertung und Vergleich der Effizienz von Algorithmen

Die Aufwandsabschätzungen werden in Abhängigkeit der Problemgröße angegeben – dies ist i.A. die Anzahl der zu bearbeitenden Elemente oder z.B. die Länge der eingegebenen Zeichenfolge. Wir konzentrieren uns dabei auf große Problemgrößen – es interessiert also nur das asymptotische Laufzeitverhalten.

Die O-Notation definiert die Umschreibung „höchstens um einen konstanten Faktor größer und nur endliche viele Ausnahmen“ formal.

Es gilt für Funktionen $f, g : \mathbb{N}_0 \to \mathbb{R}_0^+$:

- $f \in O(g) \;:\Leftrightarrow\; \exists c \in \mathbb{R}^+,\ n_0 \in \mathbb{N} :\ f(n) \leq c \cdot g(n),\ n \geq n_0$,
  d. h., f wächst asymptotisch nicht schneller als g
- $f \in o(g) \;:\Leftrightarrow\; \forall c \in \mathbb{R}^+,\ \exists n_0 \in \mathbb{N} :\ f(n) \leq c \cdot g(n),\ n \geq n_0$,
  d. h., f wächst asymptotisch echt langsamer als g
- $f \in \Omega(g) \;:\Leftrightarrow\; g \in O(f)$,
  d. h., f wächst asymptotisch nicht langsamer als g
- $f \in \omega(g) \;:\Leftrightarrow\; g \in o(f)$,
  d. h., f wächst asymptotisch echt schneller als g

- $f \in \Theta(g) \;:\Leftrightarrow\; f \in O(g) \wedge g \in O(f)$,
  d. h., f wächst asymptotisch gleich schnell wie g

Einige Beispiele zur Illustration der Definition $O(\cdot)$ ... es sind die Konstanten c und n_0 zu finden, so dass die Ungleichung in der Definition erfüllt ist.

In der Praxis: für fast alle Fälle kann man über $\lim_{n\to\infty}(f(n)/g(n))$ bestimmen, in welchem Verhältnis bzgl. O-Notation die beiden Funktionen f und g zueinanderstehen. Divergiert der Quotient ($\lim \to \infty$), so wächst f echt schneller (also $f \in \omega(g)$ bzw. $g \in o(f)$), bei $\lim \to 0$ wächst f echt langsamer (also $f \in o(f)$ bzw. $f \in \omega(f)$), gilt $lim \to c$ für eine Konstante ungleich 0, so unterscheiden sich die beiden Funktionen ab einem n_0 nur noch maximal um einen konstanten Faktor, somit $f \in O(g)$ und $g \in O(f)$ und somit auch $f \in \Theta(g)$ und $g \in \Theta(f)$). Existiert der Grenzwert nicht, dann sind die Funktionen entweder bzgl. O nicht vergleichbar (in der Praxis selten) oder der Quotient bewegt sich in einem beschränkten Bereich, ist aber nicht konvergent (ebenfalls selten), Beispiele selbst überlegen.

Programmkonstrukte und deren Laufzeiten:

- Folge von Anweisungen: Summe der Einzelaufwände
- for-Schleifen (bedingt auch while-Schleifen): Anzahl der Durchläufe mal Aufwand für den Schleifenrumpf
- Fallunterscheidung: Summe der Einzelaufwände (!)
- Rekursion: führt auf Gleichungen für den Aufwand – hier hilft nur Übung um zu erkennen, welcher Aufwand letztlich nötig ist. Hat man eine Idee, was rauskommen könnte, so kann man versuchen, über die Gleichungen und mit vollständiger Induktion die Vermutung zu beweisen.