§ 15. LANDAU - SYMBOLE

15.1. Motivation

- Suchen kompakte Notation zur Aufwandsabschätzung von Algorithmen
- Algorithmus soll von Problemgröße abhängen, die durch eine Zahl n beschrieben wird
  (z.B.: n = Zahl der zu sortierenden Objekte in einer Liste)
- Laufzeit ist Abbildung $f: \mathbb{N} \to \mathbb{R}^+$ mit $f(n) = a_n$, d.h. eine Folge
- Meist ist (a_n) mon. wachsend und nicht beschränkt, d.h. $\lim\limits_{n\to\infty} a_n = \infty$
- Interessant ist, wie schnell a_n gegen ∞ strebt. Hierzu vergleicht man (a_n) mit Prototypen von anderen Folgen (b_n).

15.2. Def.: Eine Folge $A = (a_n)$ ist „<u>Groß-O</u>" von $B = (b_n)$, wenn der Quotient $\left(\frac{a_n}{b_n}\right)$ beschränkt ist. Die Folge A ist „<u>Klein-o</u>" von B, wenn $\left(\frac{a_n}{b_n}\right)$ eine Nullfolge ist.

Wir schreiben $A = O(B)$ bzw. $A = o(B)$.

o und O nennt man auch <u>Landau-Symbole</u>.

15.3. Beispiele

a) $2n^2 + 3n + 4 = O(n^2)$, denn

$$\frac{2n^2+3n+4}{n^2} = \frac{n^2\left(2+\frac{3}{n}+\frac{4}{n^2}\right)}{n^2} = 2+\frac{3}{n}+\frac{4}{n^2}$$

Dies ist eine konvergente Folge und somit beschränkt.

b) $2n^2+3n+4$ ist aber auch $o(n^3)$:

$$\frac{2n^2+3n+4}{n^3} = \frac{\frac{2}{n}+\frac{3}{n^2}+\frac{4}{n^3}}{1} \longrightarrow 0 \text{ für } n \to \infty.$$

15.4 Mehrdeutigkeit

Die Angabe $A = O(B)$ ist nicht eindeutig. Für $A = (n+2)$ sind z. B. $A = O(n^2)$, $A = O(n)$, $A = O(\frac{n}{2})$ korrekte Aussagen. Es gilt:

- Konstante Faktoren ändern nicht an der Ordnung:

  $$O(B) = O(cB) \quad \forall\, c \in \mathbb{R}$$

- Terme niedriger Ordnung sind unwichtig:

  z. B. haben (n^2+3n+4) und (n^2+17n) die selbe Ordnung $O(n^2)$.

15.5. Satz (Rechenregeln für Landau-Symbole)

Für reelle Zahlenfolgen A, B und C gelten folgende Regeln:

a) $A = O(A)$

b) $c \cdot O(A) = O(A)$
$c \cdot o(A) = o(A)$
$\left.\begin{array}{l}\\\\\end{array}\right\} \forall\, c \in \mathbb{R}$

c) $O(A) + O(A) = O(A)$
$o(A) + o(A) = o(A)$

d) $O(O(A)) = O(A)$
$o(o(A)) = o(A)$

e) $O(A) \cdot O(B) = O(AB)$
$o(A) \cdot o(B) = o(AB)$

f) $A \cdot O(B) = O(AB)$
$A \cdot o(B) = o(AB)$

g) Transitivität:
$A = O(B),\ B = O(C) \Rightarrow A = O(C)$
$A = o(B),\ B = o(C) \Rightarrow A = o(C)$

15.6. Vergleichbarkeit von Folgen

Nicht alle Folgen sind vergleichbar. Betrachte $A = (a_n)$, $B = (b_n)$ mit

$$a_n = \begin{cases} n^2 & (n \text{ gerade}) \\ n & (n \text{ ungerade}) \end{cases}$$

$$b_n = \begin{cases} n & (n \text{ gerade}) \\ n^2 & (n \text{ ungerade}) \end{cases}.$$

Für gerades n sieht man, dass $A \neq O(B)$.

" ungerades " " " " " $B \neq O(A)$.

15.7 → nächste Seite

15.8. Häufig verwendete Prototypen von Vergleichsfunktionen:

O	Laufzeitverhalten
$O(1)$	konstant
$O(\log_a n)$, $a > 1$	logarithmisch
$O(n)$	linear
$O(n \log_a n)$, $a > 1$	$n \log n$
$O(n^2)$	quadratisch
$O(n^3)$	kubisch
$O(n^k)$	polynomial
$O(a^n)$, $a > 1$	exponentiell

Häufig ist $a = 2$. Die Wahl von a ist jedoch unwichtig, da sich Logarithmen zu unterschiedlichen Basen nur durch eine multiplik. Konstante unterscheiden.

$\log_a n = ld\, n / ld\, a$ mit $ld := \log_2$

15.7. Def.: Wir sagen

$$O(A) = O(B) \quad :\Leftrightarrow \quad A = O(B) \text{ und } B = O(A)$$

$$O(A) < O(B) \quad :\Leftrightarrow \quad A = O(B) \text{ und } B \neq O(A)$$

15.9. Vergleich von Ordnungen

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) <$$

$$< \ldots < O(n^4) < \ldots < O(2^n) < O(3^n) < \ldots$$

n	$\mathrm{ld}\, n$	$n \,\mathrm{ld}\, n$	n^2	n^3	2^n
10	3,32	33,22	100	1000	1024
100	6,64	664,4	10.000	10^6	$1{,}27 \cdot 10^{30}$
1000	9,97	9966	10^6	10^9	10^{301}
10000	13,29	132.877	10^8	10^{12}	10^{3010}

Für große n sollte man daher nicht auf Fortschritte auf dem Rechnersektor hoffen, sondern nach Algorithmen suchen, die eine bessere Ordnung besitzen.