9. Analyse von Algorithmen

Ziel: Entwicklung von effizienten Algorithmen

Voraussetzungen über das Rechnermodell:

- sequentieller Programmablauf
- RAM

Analyse:
- feststellen, wieviele Grundoperationen der Algorithmus ausführt
- in Abhängigkeit von der Größe der Eingabedaten
- Größenordnung dieser Anzahl

Analysearten

A priori - Analyse:

- rechnerunabhängig
- drückt die Komplexität des Alg. in Abhängigkeit von der Größe der Eingabedaten aus

Arten:

- worst case - Komplexität: obere Schranke für die Komplexität
- average case (mittlere Komplexität): obere Schranke für die mittlere (erwartete) Laufzeit
- untere Komplexitätsschranke

A posteriori - Analyse

Testen von Laufzeit (und evtl Speicherbedarf) einer Implementation des Algorithmus auf einem Rechner.

Asymptotische Notation

1. obere Schranken

Definition. Sei $g: \mathbb{N} \to \mathbb{N}$.

$$O(g) := \left\{ f: \mathbb{N} \to \mathbb{N} \;\middle|\; \exists\, c > 0,\ n_0 \in \mathbb{N}\ \forall n \geq n_0 : f(n) \leq c \cdot g(n) \right\}$$

Für $f \in O(g)$ schreibt man auch

$f = O(g)$ oder $f(n) \in O(g(n))$

Satz 9.1. Ist $f: \mathbb{N} \to \mathbb{N}$ ein Polynom vom Grad m, dann gilt $f(n) = O(n^m)$.

Beweis: Ist $f(n) = a_m n^m + \ldots + a_1 n + a_0$ (mit $a_i \in \mathbb{R}$, $a_m \neq 0$), dann gilt:

$$\begin{aligned} f(n) = |f(n)| &= |a_m n^m + \ldots + a_1 n + a_0| \\ &\leq |a_m| n^m + \ldots + |a_1| n + |a_0| \\ &= \left(|a_m| + \frac{|a_{m-1}|}{n} + \ldots + \frac{|a_1|}{n^{m-1}} + \frac{|a_0|}{n^m}\right) \cdot n^m \\ &\leq \underbrace{(|a_m| + |a_{m-1}| + \ldots + |a_1| + |a_0|)}_{=: c} \cdot \underbrace{n^m}_{g(n)} \\ &\leq c \cdot g(n) \end{aligned}$$

für alle $n \geq 1 =: n_0$.

Also: $f = O(g) = O(n^m)$. □

Definition. Sei $g: \mathbb{N} \to \mathbb{N}$.

$$o(g) := \{ f: \mathbb{N} \to \mathbb{N} \mid \forall c > 0 \ \exists n_0 \in \mathbb{N} \ \forall n \geq n_0 : f(n) < c \cdot g(n) \}$$

analog: $f = o(g)$, $f(n) \in o(g(n))$ ...

Lemma 9.1. Für $f, g: \mathbb{N} \to \mathbb{N}$ gilt:

$$f = o(g) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 .$$

Beweis:

„$\Rightarrow$": Sei $f = o(g)$. Sei $\varepsilon > 0$ beliebig. Dann gilt $f(n) < \varepsilon \cdot g(n)$ für alle $n > n_0$ mit einem $n_0 \in \mathbb{N}$, also $\frac{f(n)}{g(n)} < \varepsilon$, d.h. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

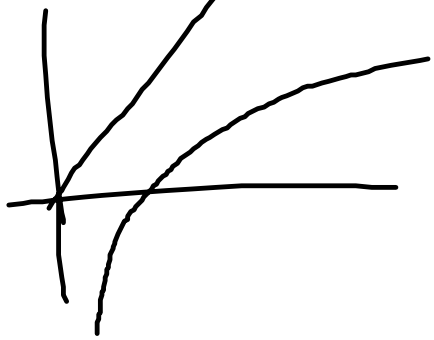
„$\Leftarrow$": Sei $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Sei $c > 0$. Dann ex. $n_0 \in \mathbb{N}$, so daß $\frac{f(n)}{g(n)} < c$ für alle $n \geq n_0$, d.h. $f(n) < c \cdot g(n)$. Also: $f \in o(g)$. □

Regel von l'Hospital.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{d}{dn} f(n)}{\frac{d}{dn} g(n)}$$

Beweis: s. Analysis

Beispiele:



i. $\log(n) = o(n)$

denn: $\log n = \frac{\ln n}{\ln 2}$

$$\lim_{n\to\infty} \frac{\log n}{n} = \lim_{n\to\infty} \frac{\frac{1}{\ln 2}\cdot\frac{1}{n}}{1} = \lim_{n\to\infty} \frac{1}{\ln 2}\cdot\frac{1}{n} = 0$$

ii. $n^k = o(2^n)$ für jedes $k \in \mathbb{N}$

denn:

$$\lim_{n\to\infty} \frac{n^k}{2^n} = \lim_{n\to\infty} \frac{k n^{k-1}}{\ln 2 \cdot 2^n} = \ldots = \lim_{n\to\infty} \frac{k!}{(\ln 2)^k \cdot 2^n} = 0$$

$2^n = e^{n \ln 2}$

$\left(e^{n\ln 2}\right)' = \ln 2 \cdot e^{n \ln 2} = \ln 2 \cdot 2^n$

Vergleich:

$O(1)$

$< O(\log\log n)$

$< O(\log n)$ (logarithmisch)

soll heißen: $\log n \in o(n)$ →

$< O(n)$ (linear)

$< O(n \cdot \log n)$ (→ sortieren!)

$< O(n^2)$ (quadratisch)

$< O(n^3)$ (kubisch)

$< O(n^k)$ (für $k \geq 4$) (polynomial)

$< O(2^n)$ (exponentiell)

2. Untere Schranken

Definition. Sei $g: \mathbb{N} \to \mathbb{N}$.

$$\Omega(g) := \{ f: \mathbb{N} \to \mathbb{N} \mid \exists c > 0, n_0 \in \mathbb{N} \; \forall n \geq n_0 : f(n) \geq c\, g(n) \}$$

(Bem.: $f \in O(g) \iff g \in \Omega(f)$)

$$\Theta(g) := O(g) \cap \Omega(g)$$

$$= \{ f: \mathbb{N} \to \mathbb{N} \mid \exists c_1, c_2 > 0, n_0 \in \mathbb{N} \; \forall n \geq n_0 : c_1\, g(n) \leq f(n) \leq c_2\, g(n) \}$$

Beispiel:

- sequentielle Suche

  $f(n)$ sei die Anzahl der Vergleiche bei der sequentiellen Suche in einem unsortierten Array der Größe n

  i. $f(n) = O(n)$
  ii. $f(n) = \Omega(n)$

  $\Rightarrow f(n) = \Theta(n)$

- Matrixmultiplikation