

Theoretische Informatik 2

Sabine Kuske

Linzer Str. 9a, OAS 3005

Tel.: 2335, 8794

kuske@informatik.uni-bremen.de

www.informatik.uni-bremen.de/theorie

7. April 2008



Das große “O”

- ▶ Aufwandsklasse $O(g)$ für $g: \mathbb{N} \rightarrow \mathbb{R}^+$ enthält alle Funktionen $f: \mathbb{N} \rightarrow \mathbb{R}^+$ mit

$$f(n) \leq c \cdot g(n) \text{ für alle } n \geq n_0$$

c, n_0 : konstant und größer als 0

- ▶ $O(g)$ beschreibt alle Probleme, die eine algorithmische Lösung op besitzen mit $T^{op} \in O(g)$
- ▶ Obere Schranke, bei der konstante Faktoren und kleine Eingaben nicht zählen.

Das große "O"

► **Idee:** wähle g einfach und einprägsam, z.B.

$$\log n, n, n \cdot \log n, n^2, n^3, \dots 2^n, \dots$$

- $\log n$: **logarithmisch**
- n : **linear**
- n^2 : **quadratisch**
- n^3 : **kubisch**
- 2^n : **exponentiell**

Eigenschaften

1. $f \in O(f)$.
2. $O(f + c) = O(f)$ (c konstant).
3. $O(c * f) = O(f)$ (c konstant).
4. $f_i \in O(g_i)$ ($i = 1, 2$) $\implies f_1 + f_2 \in O(g_1 + g_2)$ und $f_1 * f_2 \in O(g_1 * g_2)$.
5. $O(f_1 + f_2) = O(\max(f_1, f_2))$.
6. Falls f ein Polynom vom Grad k ist, gilt $f \in O(n^k)$.
7. $f \in O(g)$ und $g \in O(h) \implies f \in O(h)$.

Zusammenhang zwischen verschiedenen Aufwandsklassen

$$O(1) \subset O(\log n) \subset O(n) \subset O(n \log n) \subset O(n^2) \subset O(n^3) \subset \dots \subset O(2^n) \subset O(3^n) \subset \dots \subset O(n!) \subset O(n^n).$$

Linear, quadratisch und exponentiell im Vergleich

n	n^2	2^n
10	100	$1024 \sim 10^3$
20	400	$1048576 \sim 10^6$
30	900	$1073741824 \sim 10^9$
40	1600	$10999511627776 \sim 10^{13}$
50	2500	$1125899906842624 \sim 10^{15}$
60	3600	$1152921504600686976 \sim 10^{18}$
70	4900	$1180591620717411303424 \sim 10^{21}$

ohne Gewähr

Mit Supercomputer gegen Aufwand (?)

- **Szenario:** entwickle ein Programm, das bei Eingabegröße n gerade $n \mid n^2 \mid 2^n$ FLOPs benötigt und miete für 3 Stunden ($\sim 10^4$ Sek.) einen Supercomputer mit 100 Tera-FLOPs pro Sekunde (10^{14} FLOPs/Sek.).
- Welche Eingabegröße lässt sich bearbeiten?

n	n^2	2^n
10^{18}	10^9	60

Mit Supercomputer gegen Aufwand (?)

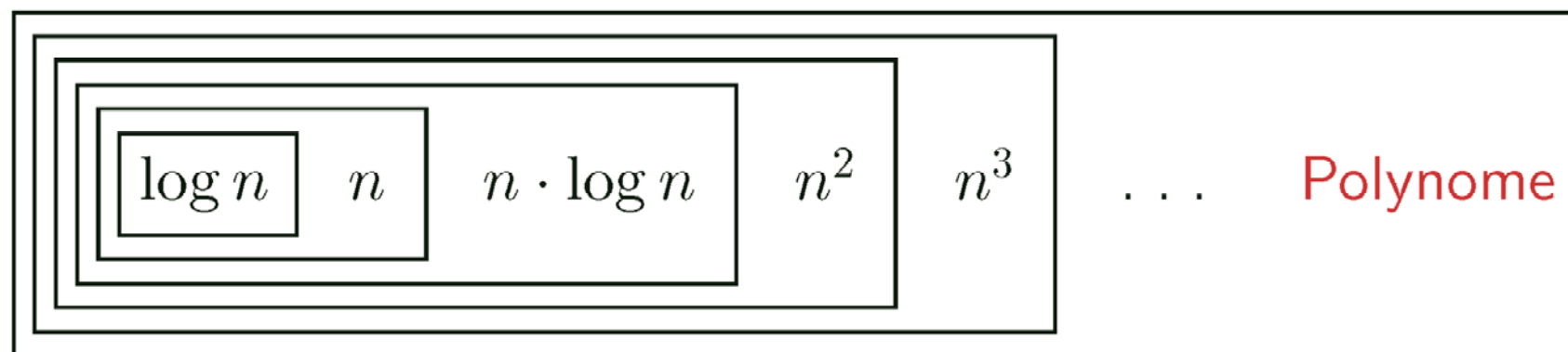
- **Szenario:** entwickle ein Programm, das bei Eingabegröße n gerade $n \mid n^2 \mid 2^n$ FLOPs benötigt und miete für 3 Stunden ($\sim 10^4$ Sek.) einen Supercomputer mit 100 Tera-FLOPs pro Sekunde (10^{14} FLOPs/Sek.).
- Welche Eingabegröße lässt sich bearbeiten?

n	n^2	2^n
10^{18}	10^9	60
10^{21}	$32 \cdot 10^9$	70 (*)

(*): bei Rechenzeitverbesserung um Faktor 1000

Polynomieller Aufwand

- Probleme mit polynomielltem Aufwand können mit herkömmlichen Rechnern in verfügbarer Zeit gelöst werden (wenn der Exponent nicht zu groß ist).



Die Klassen P und NP

P enthält die Probleme, die mit herkömmlichen Computern zeitgerecht lösbar sind

NP enthält eine Vielzahl praktisch relevanter Probleme (wie Tourenplanung, Maschinenbelegung, Stundenplanung, Lagerhaltung, . . .)

Definition von NP und P

$dp \in NP$, falls eine Lösung sol und ein $k \in \mathbb{N}$ existieren mit $T^{sol} \in O(n^k)$

$dp \in P$, falls zusätzlich für alle $w \in A^*$ gilt:

$$dp(w) = T \text{ und } sol(w) \xleftrightarrow{*} t \text{ impl. } t \xleftrightarrow{*} T$$

offensichtlich: $P \subseteq NP$

offen: $NP \subseteq P$

Das $P = NP$ -Problem

Besitzen (Entscheidungs-)Probleme mit einer **N**icht-deterministischen **P**olynomiellen Lösung immer auch eine deterministische **P**olynomielle Lösung?

- ▶ Eines der bekanntesten offenen Probleme der Informatik
- ▶ Nobelpreis-verdächtig

Beispiele

Problem aus P	Aufwand
Suchen in balancierten Bäumen	$\log n$
Suchen in Wörtern, Transponieren, Zählen, Filtern	n
Sortieren durch Mischen	$n \cdot \log n$
Sortieren durch Einsortieren, <i>quicksort</i>	n^2
Matrizenmultiplikation, Wortproblem kontextfreier Sprachen	n^3

Anmerkungen zum Nichtdeterminismus

- ▶ Manches ist nichtdeterministisch

Suchen in Bereichen, unsicheres und unvollständiges Wissen, Expertensysteme, Spiele

- ▶ Gleichwertigkeit ist nichtdeterministisch
- ▶ Manchmal bequem (und schadet nicht)
- ▶ Einschränken, wenn gewünscht und nötig

Beispiel: Little-Solitaire

solitaire

opns: $\text{solitaire}, \text{won} : \{0, 1\}^* \rightarrow \text{BOOL}$

$\text{move} : \{0, 1\}^* \rightarrow \{0, 1\}^*$

vars: $u, v, w \in \{0, 1\}^*$

eqns: $\text{solitaire}(w) = \text{won}(\text{move}(w))$

$\text{won}(w) = \text{count}(1, w) = 1$

$\text{move}(u110v) = \text{move}(u001v)$

$\text{move}(u011v) = \text{move}(u100v)$

$\text{move}(w) = w$

