Datenstrukturen und Algorithmen

Vorlesung 2: Asymptotische Effizienz (K3)

Prof. Dr. Erika Ábrahám

Theorie Hybrider Systeme
Informatik 2

http://ths.rwth-aachen.de/teaching/ss-14/datenstrukturen-und-algorithmen/

Diese Präsentation verwendet in Teilen Folien von Joost-Pieter Katoen.

17. April 2014

RWTH AACHEN UNIVERSITY

Übersicht

Übersicht

Laufzeit von Algorithmen

Betrachte

Die Laufzeit eines Algorithmus ist keine Zahl, sondern eine Funktion.

Sie gibt die Laufzeit des Algorithmus für jede Eingabelänge an.

Laufzeiten

Die Best-Case/Average-Case/Worst-Case Laufzeit $B(n)/A(n)/W(n)$ für Eingabelänge n ist die kürzeste/mittlere/längste Laufzeit aus allen Eingaben mit Länge n.

Asymptotische Betrachtung

Die exakte Bestimmung der Funktionen $B(n)$, $A(n)$ und $W(n)$ ist üblicherweise sehr schwierig. Außerdem:

- ist sie von zweifelhaftem Nutzen für Vergleiche:
  Ist etwa $W(n) = 1021n$ besser als $W(n) = \frac{1}{2}n^2$?
- wollen wir eingabenunabhängige Konstanten (z.B. Rechnergeschwindigkeit, Güte des Compiler, usw.) ausklammern.

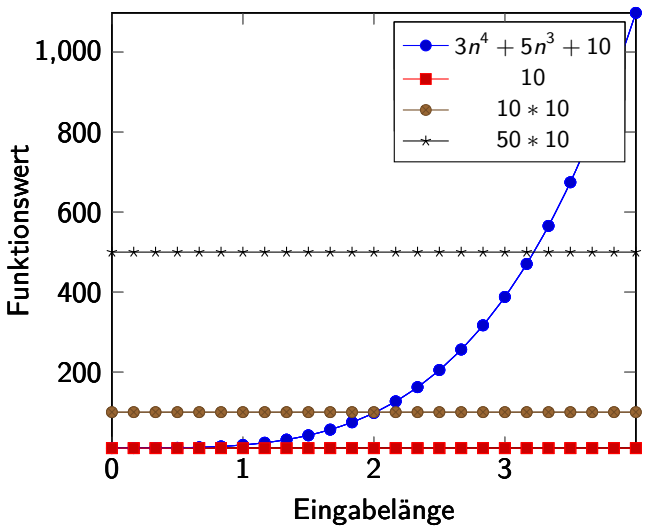
Daher: Normalerweise keine exakte sondern asymptotische Betrachtung.

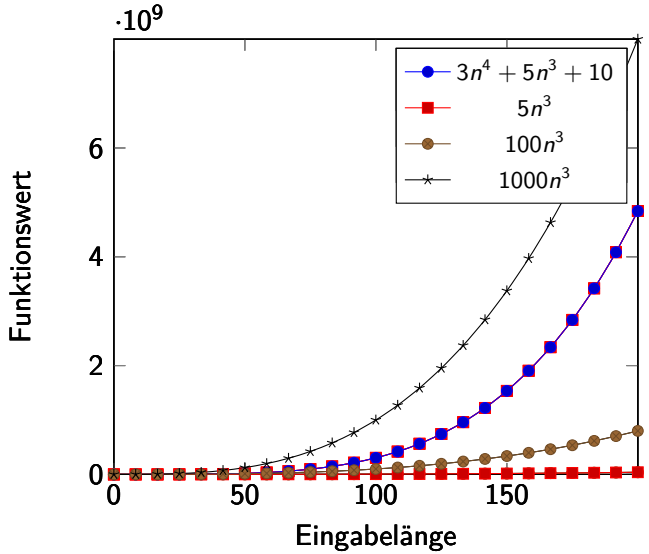
- Betrachte Wachstum der Laufzeit für $n \to \infty$.
- Kurze Eingaben und konstante Faktoren werden vernachlässigt.
- Anschaulich: Wir lassen Glieder niedriger Ordnung weg, z.B.:

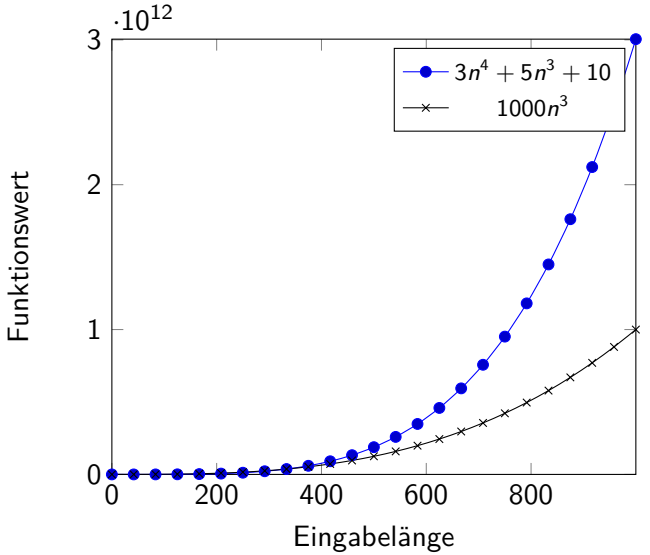
$$W(n) = 3n^4 + 5n^3 + 10 \in \mathcal{O}(n^4)$$

(d.h. n^4 ist dominierender Faktor für $n \to \infty$)

- So erhalten wir untere/obere Schranken für $A(n)$, $B(n)$ und $W(n)$!
- Mathematische Zutat: Asymptotische Ordnung von Funktionen.







Wie vergleichen wir das Wachstum von Funktionen?

Seien $f, g : \mathbb{N} \to \mathbb{R}_{\geqslant 0}$, so dass $\lim_{n\to\infty} \frac{g(n)}{f(n)}$ existiert.

$$\begin{array}{rlcl} & \lim_{n\to\infty} \frac{g(n)}{f(n)} & = \infty & : \quad g \text{ wächst schneller als } f \\ 0 < & \lim_{n\to\infty} \frac{g(n)}{f(n)} & < \infty & : \quad g \text{ wächst gleich schnell wie } f \\ 0 = & \lim_{n\to\infty} \frac{g(n)}{f(n)} & & : \quad g \text{ wächst langsamer als } f \end{array}$$

Beispiel

- $\lim_{n\to\infty} \frac{n^2}{n} = \lim_{n\to\infty} n = \infty$
- $\lim_{n\to\infty} \frac{3n^4+5n^2+10}{n^4} = \lim_{n\to\infty}(3 + \frac{5}{n^2} + \frac{10}{n^4}) = 3$
- $\lim_{n\to\infty} \frac{100n^3}{n^4} = \lim_{n\to\infty} \frac{100}{n} = 0$

Manchmal ist die Berechnung nicht so einfach...

Regel von L'Hôpital

Seien $f, g : \mathbb{N} \to \mathbb{R}_{\geqslant 0}$ differenzierbar und beide bestimmt divergierend oder beide zu 0 konvergierend. Wenn $\lim_{n\to\infty} \frac{g'(n)}{f'(n)}$ existiert dann gilt:

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{g'(n)}{f'(n)}.$$

Beispiel

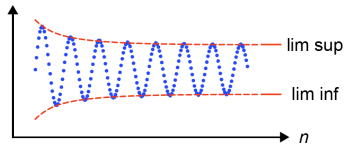
$$\lim_{n\to\infty} \frac{e^n}{n} = \lim_{n\to\infty} \frac{(e^n)'}{(n)'} = \lim_{n\to\infty} \frac{e^n}{1} = \infty$$

Wenn der Limes nicht existiert...

Limes inferior und Limes superior

Sei $h : \mathbb{N} \to \mathbb{R}_{\geqslant 0}$. Dann:

1. $\liminf_{n\to\infty} h(n) = \lim_{n\to\infty} \left(\inf_{m\geqslant n} h(m) \right)$
2. $\limsup_{n\to\infty} h(n) = \lim_{n\to\infty} \left(\sup_{m\geqslant n} h(m) \right)$



Einige Fakten

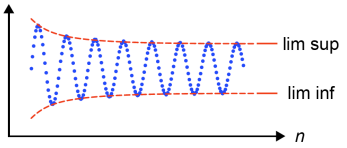
1. Existieren $\liminf_{n\to\infty} h(n)$ und $\limsup_{n\to\infty} h(n)$: $\liminf_{n\to\infty} h(n) \leqslant \limsup_{n\to\infty} h(n)$.
2. Existiert $\lim_{n\to\infty} h(n)$ dann: $\liminf_{n\to\infty} h(n) = \limsup_{n\to\infty} h(n) = \lim_{n\to\infty} h(n)$.

Wenn der Limes nicht existiert...

Limes inferior und Limes superior

Sei $h : \mathbb{N} \to \mathbb{R}_{\geqslant 0}$. Dann:

1. $\liminf_{n\to\infty} h(n) = \lim_{n\to\infty} \left(\inf_{m\geqslant n} h(m) \right)$
2. $\limsup_{n\to\infty} h(n) = \lim_{n\to\infty} \left(\sup_{m\geqslant n} h(m) \right)$

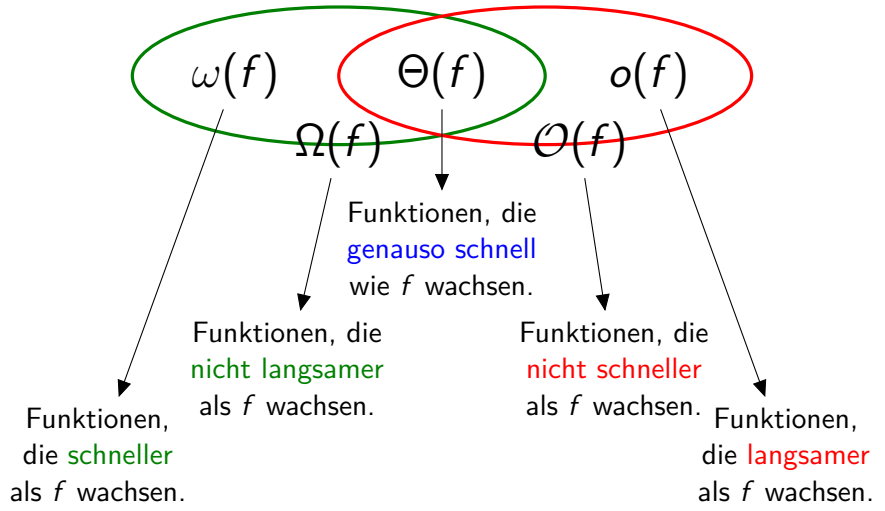


Seien $f, g : \mathbb{N} \to \mathbb{R}_{\geqslant 0}$.

$$\begin{array}{rlcl} 0 < & \liminf_{n\to\infty} \frac{g(n)}{f(n)} & & : \quad g \text{ wächst nicht langsamer als } f \\ 0 < & \liminf_{n\to\infty} \frac{g(n)}{f(n)} & < \infty & : \quad g \text{ wächst gleich schnell wie } f \\ & \limsup_{n\to\infty} \frac{g(n)}{f(n)} & < \infty & : \quad g \text{ wächst nicht schneller als } f \end{array}$$

Übersicht

Die Klassen ω, Ω, Θ, $\mathcal{O}$ und o



Die Klasse Klein-O

$o(f)$ ist die *Menge* von Funktionen, die langsamer als f wachsen.

Definition

$g \in o(f)$ gdw. $\forall c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ g(n) < c \cdot f(n)$.

Lemma

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

Beweis.

$$\begin{aligned} & \lim_{n\to\infty} \tfrac{g(n)}{f(n)} = 0 \\ \Leftrightarrow\ & \forall c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ \tfrac{g(n)}{f(n)} < c \\ \Leftrightarrow\ & \forall c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ g(n) < c \cdot f(n)\,. \end{aligned}$$

□

Die Klasse Klein-O

$$\begin{aligned} & g \in o(f) \\ \text{gdw.}\ & \forall c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ g(n) < c \cdot f(n) \\ \text{gdw.}\ & \lim_{n\to\infty} \tfrac{g(n)}{f(n)} = 0. \end{aligned}$$

Beispiel

$1 \in o(\log(n))$: $n_0 > a^{1/c}$, $\lim_{n\to\infty} \frac{1}{\log(n)} = 0$

$n \in o(n^2)$: $n_0 > \frac{1}{c}$, $\lim_{n\to\infty} \frac{n}{n^2} = \lim_{n\to\infty} \frac{1}{n} = 0$

Die Klasse Klein-Omega

$\omega(f)$ ist die *Menge* von Funktionen, die schneller als f wachsen.

Definition

$g \in \omega(f)$ gdw. $\forall c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ c \cdot f(n) < g(n)$.

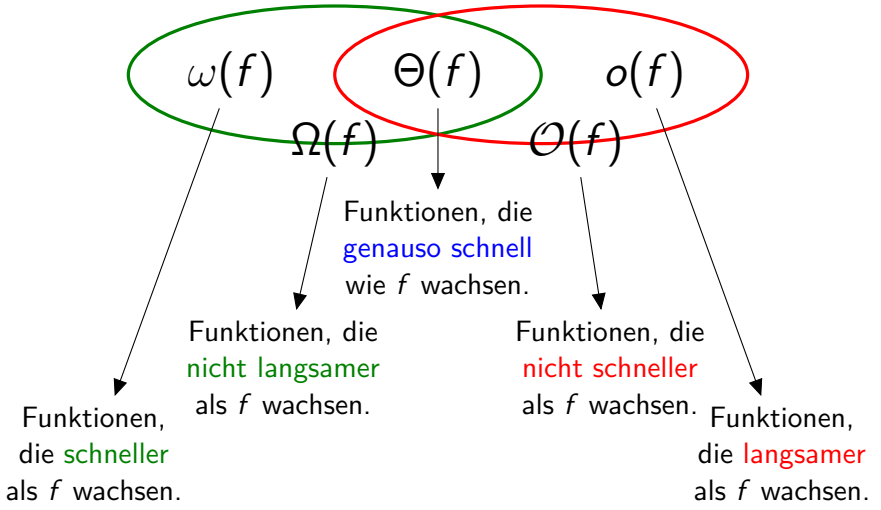
Lemma

$g \in \omega(f)$ *gdw.* $\lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$ *gdw.* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

Beziehung zwischen o und ω

- $f \in o(g)$ gdw. $g \in \omega(f)$.

Die Klassen ω, Ω, Θ, $\mathcal{O}$ und o



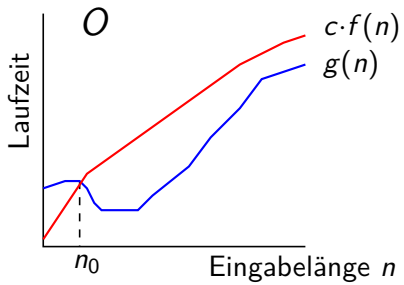
Die Klasse Groß-O

Seien f und g Funktionen von $\mathbb{N}$ (Eingabelänge) nach $\mathbb{R}_{\geqslant 0}$ (Laufzeit).

$\mathcal{O}(f)$ ist die *Menge* von Funktionen, die nicht schneller als f wachsen.

- $g \in \mathcal{O}(f)$ heißt: $c \cdot f(n)$ ist obere Schranke für $g(n)$ mit einem geeigneten $c > 0$.

Diese Eigenschaft gilt ab einer Konstanten n_0; Werte unter n_0 werden vernachlässigt.



Die Klasse Groß-O liefert eine obere Schranke für die Komplexität einer Funktion.

- Kleinste obere Schranken sind von größtem Nutzen! $g \in \mathcal{O}(n^2)$ sagt mehr als $g \in \mathcal{O}(n^3)$.

Definition

$g \in \mathcal{O}(f)$ gdw. $\exists c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ g(n) \leqslant c \cdot f(n)$.

Die Klasse Groß-O

Definition

$g \in \mathcal{O}(f)$ gdw. $g \in \mathcal{O}(f)$ gdw. $\exists c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ g(n) \leqslant c \cdot f(n)$.

Lemma

$g \in \mathcal{O}(f)$ *gdw.* $\limsup_{n\to\infty} \frac{g(n)}{f(n)} < \infty$.

Beweis.

„$\Longrightarrow$“: Sei $\limsup_{n\to\infty} \frac{g(n)}{f(n)} = c < \infty$. Für $\varepsilon \geqslant 0$ es folgt $c + \varepsilon \geqslant \frac{g(n)}{f(n)}$ und $f(n) \neq 0$ bis auf endlich viele Ausnahmen. Ab einem $n_0 \in \mathbb{N}$ gilt für alle $n \geqslant n_0$ also: $c + \varepsilon \geqslant \frac{g(n)}{f(n)}$; und damit: $g(n) \leqslant (c + \varepsilon) \cdot f(n)$.

„$\Longleftarrow$“: Seien n_0', $c > 0$ so dass $\forall n \geqslant n_0'.\ g(n) \leqslant c \cdot f(n)$.
Da Limes existiert, ab einem $n_0 \geqslant n_0'$ gilt $f(n) \neq 0$ für alle $n \geqslant n_0$.
Damit ist $\forall n \geqslant n_0.\ \frac{g(n)}{f(n)} \leqslant c$.

Die Folge $a_n = \frac{g(n)}{f(n)}$ ist in $[0, c]$, also beschränkt und abgeschlossen.
Dann existiert $\limsup_{n\to\infty} a_n < \infty$. □

Die Klasse Groß-O

Definition

$g \in \mathcal{O}(f)$ gdw. $\exists c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ g(n) \leqslant c \cdot f(n)$.

Lemma

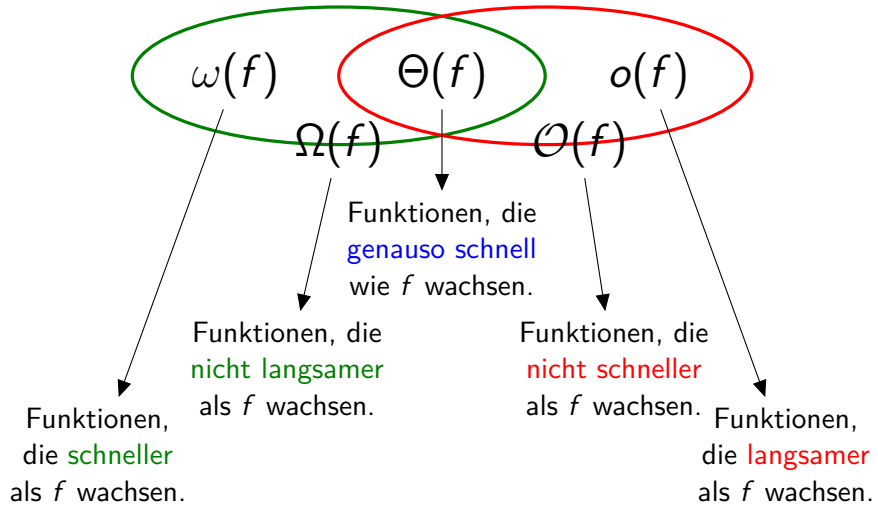
$g \in \mathcal{O}(f)$ *gdw.* $\limsup_{n\to\infty} \frac{g(n)}{f(n)} < \infty$.

Beispiel

Betrachte $g(n) = 3n^2 + 10n + 6$. Dann ist:

- $g \notin \mathcal{O}(n)$, da $\limsup_{n\to\infty} g(n)/n = \infty$.
- $g \in \mathcal{O}(n^2)$, da $g(n) \leqslant 20n^2$ für $n \geqslant 1$.
- $g \in \mathcal{O}(n^3)$, da $g(n) \leqslant \frac{1}{10}n^3$ für n hinreichend groß.

Die Klassen ω, Ω, Θ, $\mathcal{O}$ und o

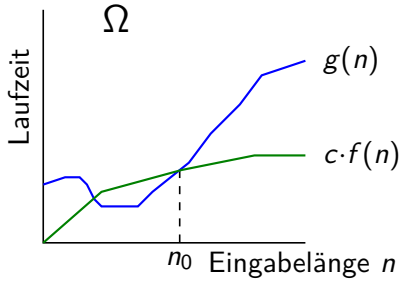


Die Klasse Groß-Omega

$\Omega(f)$ ist die *Menge* von Funktionen, die nicht langsamer als f wachsen.

- $g \in \Omega(f)$ heißt: $c \cdot f(n)$ ist untere Schranke für $g(n)$.

Diese Eigenschaft gilt ab einer Konstanten n_0; Werte unter n_0 werden vernachlässigt.



Die Klasse Groß-Omega liefert eine untere Schranke für die Komplexität einer Funktion.

- Größte untere Schranken sind von größtem Nutzen! $g \in \Omega(n^2)$ sagt mehr als $g \in \Omega(n)$.

Die Klasse Groß-Omega

Definition

$g \in \Omega(f)$ gdw. $\exists c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ c \cdot f(n) \leqslant g(n)$.

Lemma

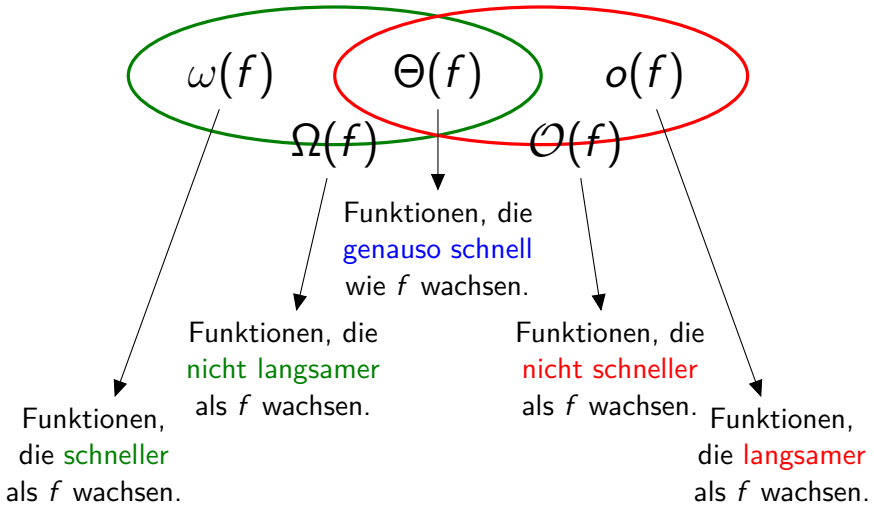
$g \in \Omega(f)$ *gdw.* $\liminf_{n\to\infty} \frac{g(n)}{f(n)} > 0$.

Beispiel

Betrachte $g(n) = 3n^2 + 10n + 6$. Dann ist:

- $g \in \Omega(n)$, da $\liminf_{n\to\infty} g(n)/n = \infty > 0$.
- $g \in \Omega(n^2)$, da $\liminf_{n\to\infty} g(n)/n^2 = 3 > 0$.
- $g \notin \Omega(n^3)$, da $g(n) \leqslant 5n^3$ für $n \geqslant 2$.

Die Klassen ω, Ω, Θ, $\mathcal{O}$ und o

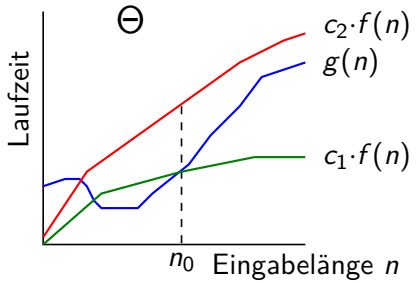


Die Klasse Groß-Theta

$\Theta(f)$ ist die *Menge* von Funktionen, die genauso schnell wie f wachsen.

- $g \in \Theta(f)$ heißt:
  $c_2 \cdot f(n)$ ist obere Schranke **und**
  $c_1 \cdot f(n)$ ist untere Schranke
  für $g(n)$.

Diese Eigenschaft gilt ab einer Konstanten n_0; Werte unter n_0 werden vernachlässigt.



Die Klasse Groß-Theta liefert eine obere und untere Schranke für die Komplexität einer Funktion.

Definition

$g \in \Theta(f)$ gdw. $\exists c_1, c_2 > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ c_1 \cdot f(n) \leqslant g(n) \leqslant c_2 \cdot f(n)$.

$g \in \Theta(f)$ *gdw.* $0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$.

Die Klasse Groß-Theta

Definition

$g \in \Theta(f)$ gdw. $\exists c_1, c_2 > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ c_1 \cdot f(n) \leqslant g(n) \leqslant c_2 \cdot f(n)$.

Lemma

$g \in \Theta(f)$ *gdw.* $0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$.

Beispiel

Betrachte $g(n) = 3n^2 + 10n + 6$. Dann ist:

- $g \notin \Theta(n)$, da zwar $g \in \Omega(n)$, aber $g \notin \mathcal{O}(n)$.
- $g \in \Theta(n^2)$, da $\lim_{n\to\infty} g(n)/n^2 = 3$.
- $g \notin \Theta(n^3)$, da zwar $g \in \mathcal{O}(n^3)$, aber $g \notin \Omega(n^3)$.

Beispiel: Fibonacci-Zahlen

Wachstum einer Kaninchenpopulation

- Zu Beginn gibt es ein Paar neugeborene Kaninchen.
- Jedes Paar wirft ab dem zweiten Monat jeden Monat ein weiteres Paar.
- Sie sterben nie und hören niemals auf.

$$Fib(0) = 1 \quad \text{und} \quad Fib(1) = 1$$
$$Fib(n+2) = Fib(n+1) + Fib(n) \quad \text{für } n \geqslant 0.$$

n	0	1	2	3	4	5	6	7	8	9	...
$Fib(n)$	1	1	2	3	5	8	13	21	34	...	

$Fib(n) \in \mathcal{O}(2^n)$ und $Fib(n) \in \Omega\left(2^{\frac{n}{2}}\right)$

Einige elementare Eigenschaften

Reflexivität

- $f \in \mathcal{O}(f)$, $f \in \Omega(f)$, $f \in \Theta(f)$.

Transitivität

- Aus $f \in \mathcal{O}(g)$ und $g \in \mathcal{O}(h)$ folgt $f \in \mathcal{O}(h)$.
- Aus $f \in \Omega(g)$ und $g \in \Omega(h)$ folgt $f \in \Omega(h)$.
- Aus $f \in \Theta(g)$ und $g \in \Theta(h)$ folgt $f \in \Theta(h)$.

Symmetrie von Θ

- $f \in \Theta(g)$ gdw. $g \in \Theta(f)$. Θ ist damit eine Äquivalenzrelation!

Beziehung zwischen $\mathcal{O}$ und Ω

- $f \in \mathcal{O}(g)$ gdw. $g \in \Omega(f)$.

Beispiel

Die folgenden 3 Aussagen sind alle gültig:

(a) $\log_a n \in \Theta(\log_b n)$, (b) $\log_b n \in \Theta(\log_a n)$, (c) $\mathcal{O}(\log_a n) = \mathcal{O}(\log_b n)$.

Beweis.

Wir beweisen (c). Zu zeigen: $\exists c_1, c_2 > 0$, so dass

$$\underbrace{\log_a n \leqslant c_1 \cdot \log_b n}_{\log_a n \in \mathcal{O}(\log_b n)} \quad \text{und} \quad \underbrace{\log_b n \leqslant c_2 \cdot \log_a n}_{\log_b n \in \mathcal{O}(\log_a n)}.$$

$$\text{Dann: } log_a n \leqslant c_1 \cdot \log_b n \;\Leftrightarrow\; \log_a n \leqslant c_1 \cdot \frac{\log_a n}{\log_a b} \;\Leftrightarrow\; \log_a b \leqslant c_1$$

Wähle $c_1 \geqslant \lceil \log_a b \rceil$.

Analog erhalten wir $\log_b a \leqslant c_2$; dann wähle $c_2 \geqslant \lceil \log_b a \rceil$.

Die Aussagen (a) und (b) folgen auf ähnliche Weise. □

Übersicht

$g \in o(f)$	$\forall c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ g(n) < c \cdot f(n)$ $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$
$g \in \mathcal{O}(f)$	$\exists c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ g(n) \leqslant c \cdot f(n)$ $\limsup_{n\to\infty} \frac{g(n)}{f(n)} < \infty$
$g \in \Theta(f)$	$\exists c_0, c_1 > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ c_0 \cdot f(n) \leqslant g(n) \leqslant c_1 \cdot f(n)$ $0 < \liminf_{n\to\infty} \frac{g(n)}{f(n)} \leqslant \limsup_{n\to\infty} \frac{g(n)}{f(n)} < \infty$
$g \in \Omega(f)$	$\exists c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ c \cdot f(n) \leqslant g(n)$ $0 < \liminf_{n\to\infty} \frac{g(n)}{f(n)}$
$g \in \omega(f)$	$\forall c > 0.\ \exists n_0.\ \forall n \geqslant n_0.\ c \cdot f(n) < g(n)$ $\lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$

Effizienz von Algorithmen – Typische Laufzeiten

Anzahl der elementaren Operationen bildet die Basis zur Bestimmung der Wachstumsrate der Zeitkomplexität bei immer längeren/größeren Eingaben.

Beispiel

Typische Effizienzklassen für Eingabelänge n:

$\mathcal{O}(1)$	konstant
$\mathcal{O}(\log n)$	logarithmisch
$\mathcal{O}(n)$	linear
$\mathcal{O}(n \cdot \log n)$	n-log-n
$\mathcal{O}(n^2)$	quadratisch
$\mathcal{O}(n^3)$	kubisch
$\mathcal{O}(n^a)$	polynomiell
$\mathcal{O}(2^n)$	exponentiell

Übersicht

Platzkomplexität

Platzkomplexität

Unter der Platzkomplexität eines Problems versteht man den (minimalen) Bedarf an Speicherplatz eines Algorithmus zur Lösung dieses Problems, in Abhängigkeit von der Länge der Eingabe.

Platzkomplexität

- Nicht nur die Zeitkomplexität, sondern auch der Speicherbedarf ist wichtig!
- Dilemma: Eine Reduktion der Zeitkomplexität führt oft zur Erhöhung der Platzkomplexität, und vice versa.
- Dies werden wir in später in der DSAL Vorlesung öfters feststellen.

Platz ist manchmal mächtiger als Zeit

Beispiel

Algorithmus zur Überprüfung ob ein Wort (Liste) von der Form $a^n b^n$ ist.

- Lösung 1:
  1. Falls ein a nach einem b auftritt, *reject*.
  2. Wenn das Wort leer ist, *accept*.
  3. Falls sowohl ein a als auch ein b vorkommt dann lösche beide, sonst *reject*.
  4. Gehe zu (2).

  Zeitkomplexität: $\mathcal{O}(n^2)$, Platzkomplexität: $\mathcal{O}(1)$
- Lösung 2:
  1. Bestimme die Anzahl n_a der aufeinanderfolgenden as am Anfang des Wortes.
  2. Bestimme die Anzahl n_b der aufeinanderfolgenden bs danach.
  3. Falls $n > n_a + n_b$ oder $n_a \neq n_b$ dann *reject*, sonst *accept*.

  Zeitkomplexität: $\mathcal{O}(n)$, Platzkomplexität: $\mathcal{O}(1)$

Beispiel: Platzkomplexität von Liedern [Knuth, 1984]

Beispiel

Betrachte eine Lied mit n Wörter, d.h., die Eingabelänge ist n.

Was ist die benötigte Platzkomplexität $S(n)$ um ein Lied der Länge n zu singen?

Obere und untere Schranken

1. $S(n) \in \mathcal{O}(n)$, da höchstens n verschiedene Wörter gespeichert werden müssen.
2. $S(n) \in \Omega(1)$, da wir mindestens eine Sache über das Lied wissen müssen, um es singen zu können.

Kann man die Platzkomplexität durch Refrains (= Kehrverse) reduzieren?

Refrains

Refrain

Die Wiederkehr von textlich/musikalisch (wenigstens überwiegend) identischen Zeilen am Schluss einer Strophe oder zwischen den Strophen.

Beispiel

Soo.. Bye, bye miss American Pie
Drove me Chevy to the levee but the levee was dry
Them good old boys were drinking whiskey and rye?
Singing this will be the day that I die
this will be the day that I die *[Don McLean]*

Platzkomplexität

Speichere den Refrain einmal und singe ihn $\mathcal{O}(n)$ Mal.

Dann: $S(n) \in \mathcal{O}(n)$, da die Anzahl der Wörter immer noch $\mathcal{O}(n)$ ist; z.B. bei Strophelänge = Refrainlänge halbiert sich der Speicherbedarf.

Die k Weihnachtstage

Reduktion der Platzkomplexität

Reduziere $S(n)$ durch eine bestimmte, sich ändernde Liedstruktur, etwa:

On the kth day of Xmas, my true love gave to me gift$_k$, gift$_{k-1}$, ..., gift$_1$
On the $(k-1)$st day of Xmas, my true love gave to me gift$_{k-1}$, ..., gift$_1$
. . . . . .
On the first day of Xmas, my true love gave to me a bottle of wine

Bekanntere Variante: „Old MacDonald had a farm“.

Platzkomplexität

Der benötigte Zeit um das Lied zu singen ist (betrachte keine Refrains):

$$n = \sum_{i=1}^{k} i = k \cdot \left(\frac{k+1}{2}\right) \in \Theta(k^2)$$

Da $n \in \Theta(k^2)$ folgt $k \in \mathcal{O}(\sqrt{n})$, also $S(n) \in \mathcal{O}(\sqrt{n})$.

100 Bierflaschen

Eine weitere Vereinfachung

Ein (sehr) langweiliges Lied für lange Autofahrten:

n bottles of beer on the wall, n bottles of beer
You take one down and pass it around
$n-1$ bottles of beer on the ball
. . . . . *[Andy Kaufman]*

Platzkomplexität

$S(n) \in \mathcal{O}(\log n)$, da nur der Wert von n von Bedeutung ist. Dafür reichen $\log n$ Bits aus.

Es geht jedoch noch etwas einfacher, nämlich indem man auf das Zählen verzichtet.

Untere Schranke?

Ein Lied mit $S(n) \in \Theta(1)$

That's the way, uh-huh, uh-huh
I like it, uh-huh, huh
. . . . . *[KC & the Sunshine Band, 1977]*