

Datenstrukturen und Algorithmen

Vorlesung 2: Asymptotische Effizienz (K3)

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

<http://www-i2.rwth-aachen.de/i2/dsa110/>

20. April 2010



Übersicht

1 Asymptotische Betrachtung

- Begründung
- Grenzwerte

2 Asymptotische Komplexitätsklassen

- Die Klasse Groß-O
- Die Klasse Groß-Omega
- Die Klasse Groß-Theta

3 Platzkomplexität

Übersicht

1 Asymptotische Betrachtung

- Begründung
- Grenzwerte

2 Asymptotische Komplexitätsklassen

- Die Klasse Groß-O
- Die Klasse Groß-Omega
- Die Klasse Groß-Theta

3 Platzkomplexität

Laufzeit von Algorithmen

Betrachte

Die Laufzeit eines Algorithmus ist keine Zahl, sondern eine **Funktion**.
Sie gibt die Laufzeit des Algorithmus für jede Eingabelänge an.

Worst-Case Laufzeit

Die **Worst-Case** Laufzeit $W(n)$ für Eingabelänge n ist die **längste** Laufzeit aus allen Eingaben mit Länge n .

Best-Case Laufzeit

Die **Best-Case** Laufzeit $B(n)$ für Eingabelänge n ist die **kürzeste** Laufzeit aus allen Eingaben mit Länge n .

Asymptotische Betrachtung

Die exakte Bestimmung der Funktionen $A(n)$, $B(n)$ 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 maschinenabhängige Konstanten (z. B. Rechengeschwindigkeit), Initialisierungsaufwand, usw. ausklammern.

Daher: Normalerweise keine exakte sondern **asymptotische** Betrachtung.

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

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

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

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

Übersicht

1 Asymptotische Betrachtung

- Begründung
- Grenzwerte

2 Asymptotische Komplexitätsklassen

- Die Klasse Groß-O
- Die Klasse Groß-Omega
- Die Klasse Groß-Theta

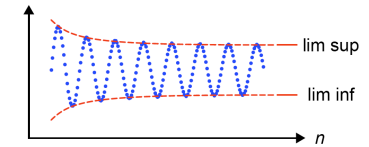
3 Platzkomplexität

Grenzwerte

Limes inferior und Limes superior

Sei $(x_i)_{i \in \mathbb{N}}$ die Folge $x_1, x_2, \dots$. Dann:

- $\liminf_{n \rightarrow \infty} x_n = \lim_{n \rightarrow \infty} \left(\inf_{m \geq n} x_m \right)$
- $\limsup_{n \rightarrow \infty} x_n = \lim_{n \rightarrow \infty} \left(\sup_{m \geq n} x_m \right)$



Einige Fakten

- Existieren $\liminf_{n \rightarrow \infty} x_n$ und $\limsup_{n \rightarrow \infty} x_n$: $\liminf_{n \rightarrow \infty} x_n \leq \limsup_{n \rightarrow \infty} x_n$.
- Existiert $\lim_{n \rightarrow \infty} x_n$ dann: $\liminf_{n \rightarrow \infty} x_n = \limsup_{n \rightarrow \infty} x_n = \lim_{n \rightarrow \infty} x_n$.

Sind f, g differenzierbar, dann gilt $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = \lim_{n \rightarrow \infty} \frac{g'(n)}{f'(n)}$. L'Hôpital

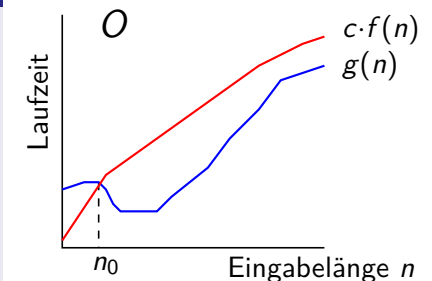
Die Klasse Groß-O (I)

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

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

- ▶ $g \in O(f)$ heißt: $c \cdot f(n)$ ist **obere** Schranke für $g(n)$.

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!
 $g \in O(n^2)$ sagt mehr als $g \in O(n^3)$.

Die Klasse Groß-O (II)

Definition

$g \in O(f)$ gdw. $\exists c > 0, n_0$ mit $\forall n \geq n_0 : 0 \leq g(n) \leq c \cdot f(n)$.

Definition (alternativ)

$g \in O(f)$ gdw. $\limsup_{n \rightarrow \infty} \frac{g(n)}{f(n)} = c \geq 0$ mit $c \neq \infty$.

Theorem

Es seien $f, g : \mathbb{N} \rightarrow \mathbb{R}$ zwei Funktionen. Es sei nur endlich oft $f(n) = 0$. Dann:

$\limsup_{n \rightarrow \infty} \frac{g(n)}{f(n)}$ existiert gdw. $\exists c > 0, n_0$ mit $\forall n \geq n_0 : g(n) \leq c \cdot f(n)$.

Die Klasse Groß-O (IV)

Definition

$g \in O(f)$ gdw. $\exists c > 0, n_0$ mit $\forall n \geq n_0 : 0 \leq g(n) \leq c \cdot f(n)$.

Definition (alternativ)

$g \in O(f)$ gdw. $\limsup_{n \rightarrow \infty} \frac{g(n)}{f(n)} = c \geq 0$ mit $c \neq \infty$.

Beispiel

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

- ▶ $g \notin O(n)$, da $\limsup_{n \rightarrow \infty} g(n)/n = \infty$.
- ▶ $g \in O(n^2)$, da $g(n) \leq 20n^2$ für $n \geq 1$.
- ▶ $g \in O(n^3)$, da $g(n) \leq \frac{1}{10}n^3$ für n hinreichend groß.

Die Klasse Groß-O (III)

Theorem

Es seien $f, g : \mathbb{N} \rightarrow \mathbb{R}$ zwei Funktionen. Es sei nur endlich oft $f(n) = 0$. Dann existiert $\limsup_{n \rightarrow \infty} \frac{g(n)}{f(n)}$ gdw. $\exists c > 0, n_0, \forall n \geq n_0 : g(n) \leq c \cdot f(n)$.

Beweis.

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

„ $\Leftarrow$ “: Gegeben seien nun $n'_0, c > 0$ so dass $\forall n \geq n'_0 : g(n) \leq c \cdot f(n)$. Ab einem $n_0 \geq n'_0$ gilt (wie oben) außerdem $f(n) \neq 0$ für alle $n \geq n_0$. Damit ist $\forall n \geq n_0 : 0 \leq \frac{g(n)}{f(n)} \leq c$.

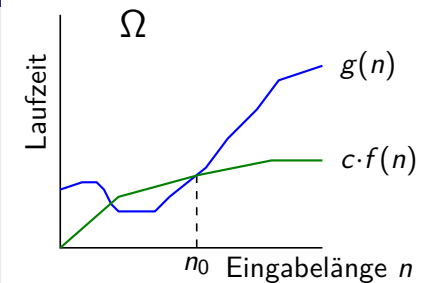
Die Folge $a_n = \frac{g(n)}{f(n)}$ ist in $[0, c]$, also beschränkt und abgeschlossen. Dann existiert $\limsup_{n \rightarrow \infty} a_n < \infty$. □

Die Klasse Groß-Omega (I)

$\Omega(f)$ 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 (II)

Definition

$g \in \Omega(f)$ gdw. $\exists c > 0, n_0$ mit $\forall n \geq n_0 : c \cdot f(n) \leq g(n)$.

Definition (alternativ)

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

Beispiel

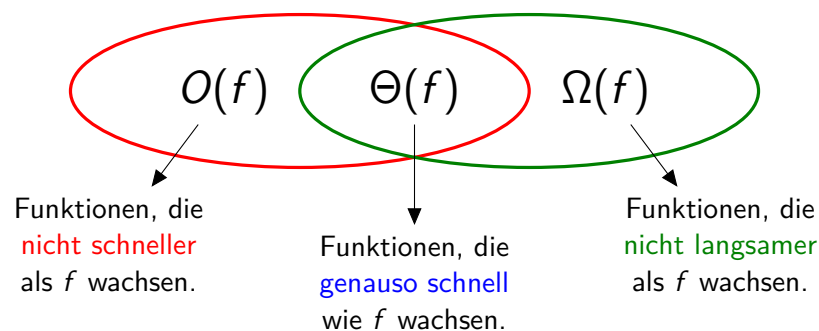
Betrachte $g(n) = 3n^2 + 10n + 6$. Dann ist:

- ▶ $g \in \Omega(n)$, da $\liminf_{n \rightarrow \infty} g(n)/n = \infty > 0$.
- ▶ $g \in \Omega(n^2)$, da $\liminf_{n \rightarrow \infty} g(n)/n^2 = 3 > 0$.
- ▶ $g \notin \Omega(n^3)$, da $g(n) \leq 5n^3$ für $n \geq 2$.

Die Klassen O , Ω und Θ

Beziehung zwischen O , Ω und Θ

$g \in \Theta(f)$ gdw. $g \in O(f)$ und $g \in \Omega(f)$.



Lemma

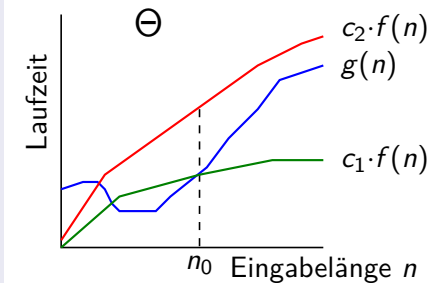
$g \in \Theta(f)$ wenn $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = c$ für ein $0 < c < \infty$.

Die Klasse Groß-Theta (I)

$\Theta(f)$ 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. $\exists c_1, c_2 > 0, n_0$ mit $\forall n \geq n_0 : c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)$.

Die Klasse Groß-Theta (II)

Definition

$g \in \Theta(f)$ gdw. $\exists c_1, c_2 > 0, n_0$ mit $\forall n \geq n_0 : c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)$.

Lemma

$g \in \Theta(f)$ wenn $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = c$ für ein $0 < c < \infty$.

Beispiel

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

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

Beispiel: Fibonacci-Zahlen

Wachstum einer Kaninchenpopulation

- ▶ Zu Beginn gibt es ein Paar geschlechtsreifer Kaninchen.
- ▶ Jedes neugeborene Paar wird im zweiten Lebensmonat geschlechtsreif.
- ▶ Jedes geschlechtsreife Paar wirft pro Monat ein weiteres Paar.
- ▶ Sie sterben nie und hören niemals auf.

$$\text{Fib}(0) = 0 \quad \text{und} \quad \text{Fib}(1) = 1$$

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

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

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

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) $O(\log_a n) = O(\log_b n)$.

Beweis.

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

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

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

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

Analog erhalten wir $\log_b a \leq c_2$; dann wähle $c_2 \geq \lceil \log_b a \rceil$.

Die Aussagen (a) und (b) folgen auf ähnliche Weise. $\square$

Einige elementare Eigenschaften

Reflexivität

$$\triangleright f \in O(f), f \in \Omega(f), f \in \Theta(f).$$

Transitivität

- ▶ Aus $f \in O(g)$ und $g \in O(h)$ folgt $f \in O(h)$.
- ▶ 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

$$\triangleright f \in \Theta(g) \text{ gdw. } g \in \Theta(f).$$

Beziehung zwischen O und Ω

$$\triangleright f \in O(g) \text{ gdw. } g \in \Omega(f).$$

Die Klassen Klein-O, Klein-Omega

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

Definition

$$g \in o(f) \text{ gdw. } \forall c > 0, \exists n_0 \text{ mit } \forall n \geq n_0 : 0 \leq g(n) < c \cdot f(n).$$

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

Definition

$$g \in \omega(f) \text{ gdw. } \forall c > 0, \exists n_0 \text{ mit } \forall n \geq n_0 : c \cdot f(n) < g(n).$$

Beziehung zwischen o und ω

$$\triangleright f \in o(g) \text{ gdw. } g \in \omega(f).$$

Übersicht

1 Asymptotische Betrachtung

- Begründung
- Grenzwerte

2 Asymptotische Komplexitätsklassen

- Die Klasse Groß-O
- Die Klasse Groß-Omega
- Die Klasse Groß-Theta

3 Platzkomplexität

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. $S(n) \in O(n)$, 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?

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.

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 $O(n)$ Mal.

Dann: $S(n) \in O(n)$, da die Anzahl der Wörter immer noch $O(n)$ 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 k th 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

Die 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 $k \in O(\sqrt{n})$, also $S(n) \in O(\sqrt{n})$.

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]

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 wall

.....

[Andy Kaufman]

Platzkomplexität

$S(n) \in O(\log n)$, 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.