

Algorithmen und Datenstrukturen (ESE)
Entwurf, Analyse und Umsetzung von
Algorithmen (IEMS)
WS 2012 / 2013

Vorlesung 3, Donnerstag 7. November 2013
(O-Notation, Theta, Omega)

Junior-Prof. Dr. Olaf Ronneberger
Image Analysis Lab
Institut für Informatik
Universität Freiburg

Blick über die Vorlesung heute

■ Organisatorisches

- Ihre Erfahrungen mit dem **Ü2** (Induktion + Laufzeit HeapSort)

■ O-Notation

- Motivation
- Klassische Definition mit C und n_0
- Einfachere Bestimmung über den $\lim_{n \rightarrow \infty}$
- Praktischer Nutzen
- $f = O(g)$ heißt nicht immer f ist besser
- **Übungsblatt 3**: ein paar Beweise / Rechenaufgaben dazu
So was in der Art kommt meistens auch in der Klausur !

Erfahrungen mit dem Ü2

- Zusammenfassung / Auszüge Stand 7. November 08:30
 - Bisher nur 10 Abgaben! Verlängerung bis Dienstag, Übung am Montag nutzen!
 - Aufgaben 1 und 2 waren einfach / bekannt aus Mathe 1
 - „Endlich mal eine Anwendung von Mathe 1“
 - Aufgabe 3 schwierig einen Ansatz zu finden.

O-Notation — Motivation

- Primär interessiert uns oft
 - das "Wachstum" einer Funktion, z.B. einer Laufzeit $T(n)$
 - Die Werte der Konstanten (z.B. $1ns$) sind dabei oft sekundär
 - Und auch, wenn die Schranken erst ab $n \geq \dots$ gelten
 - Zum Beispiel war beim Sortieren interessant, dass
 - die Laufzeit von **MinSort** "wächst wie" n^2
 - aber die Laufzeit von **HeapSort** "wächst wie" $n \cdot \log n$
 - Das wollen wir jetzt formaler machen, damit wir in Zukunft etwas schreiben bzw. sagen können wie:
 - Die Laufzeit des Algorithmus ist $O(n)$ "O von n"
 - Die Laufzeit des Algorithmus ist $\Omega(n)$ "Omega von n"
 - Die Laufzeit des Algorithmus ist $\Theta(n)$ "Theta von n"
 - O, Ω, Θ sind die **Landau-Symbole**

O-Notation — Definition

■ Vorweg

- Wir betrachten Funktionen $f : \mathbb{N} \rightarrow \mathbb{R} : n \mapsto f(n)$
 - \mathbb{N} : die natürlichen Zahlen ... typisch: Eingabegröße
 - \mathbb{R} : die reellen Zahlen ... typisch: Laufzeit
- Zum Beispiel
 - $f(n) = 3 \cdot n$
 - $f(n) = 2 \cdot n \cdot \log n$
 - $f(n) = n^2 / 10$
 - $f(n) = n^2 + 3 \cdot n \cdot \log n - 4 \cdot n$

Definition Groß-O

■ Groß-O, Definition

- Seien f und g zwei Funktionen $\mathbb{N} \rightarrow \mathbb{R}$
- **Intuitiv:** Man sagt f ist Groß-O von g ...
 - ... wenn f "höchstens so stark wächst wie" g
 - es zählt die "Wachstumsrate", nicht die absoluten Werte!
- **Informal:** Man schreibt $f = \mathcal{O}(g)$... Steht für „ist“, nicht „ist gleich“
 - ... wenn ab irgendeinem Wert n_0 für alle $n \geq n_0$
 - $f(n) \leq C \cdot g(n)$ für irgendeine Konstante C
- **Formal:** Korrekterweise müsste man schreiben: $f \in \mathcal{O}(g)$
da $\mathcal{O}(g)$ definiert ist als eine Menge von Funktionen:

$$\mathcal{O}(g) = \left\{ f : \mathbb{N} \rightarrow \mathbb{R} \mid \exists n_0 \in \mathbb{N} \exists C > 0 \forall n > n_0 : f(n) \leq C \cdot g(n) \right\}$$

„Die Menge
aller Funktionen“

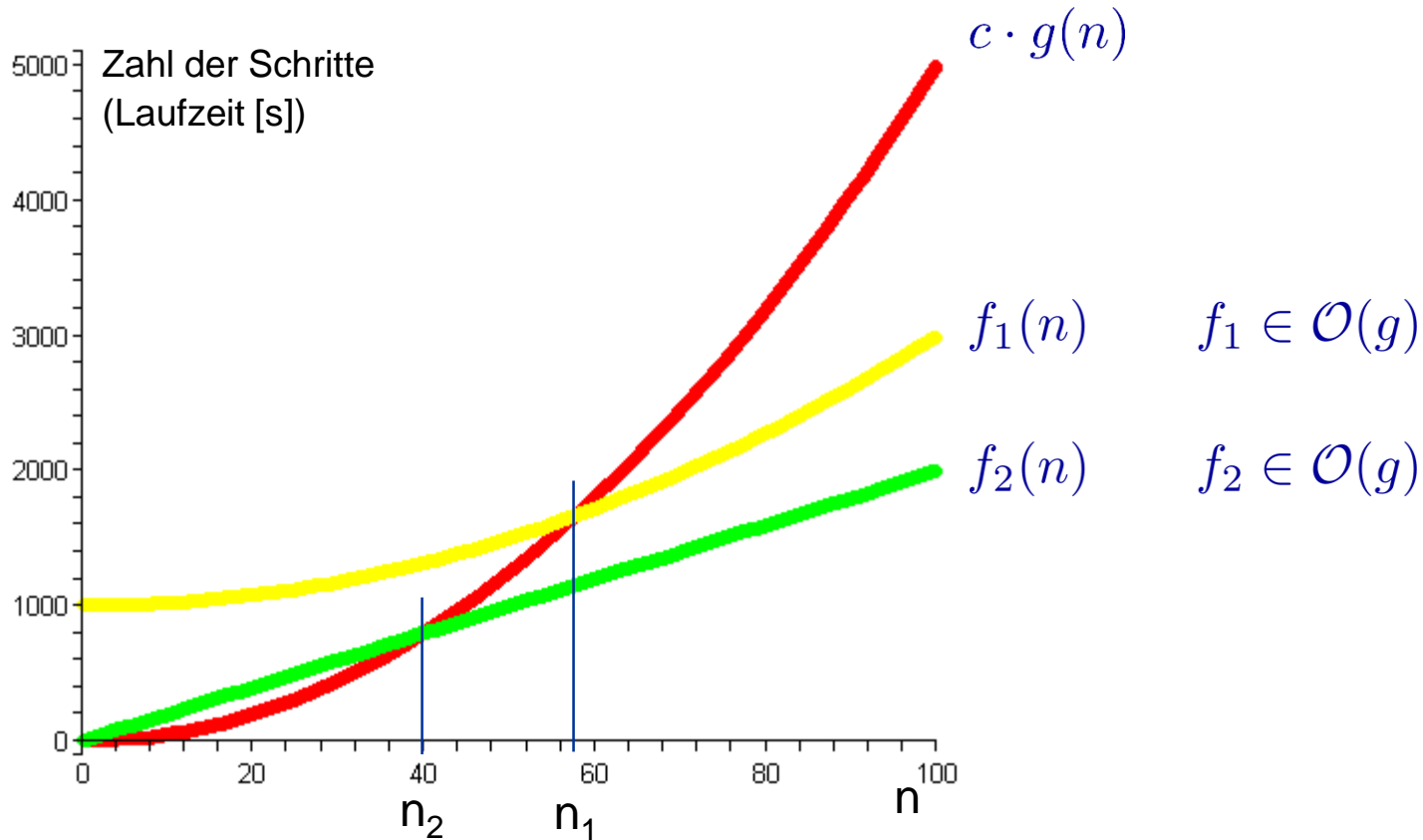
„für die gilt:“

„es existiert“

„für alle“

„so, dass“

Illustration Groß-O



f ist höchstens von der Größenordnung von g .

Beispiele Groß-O

■ Groß-O, Beispiel

- Sei $f(n) = 5 \cdot n + 7$ und $g(n) = n$
- Dann gilt $f = O(g)$ bzw. man (als Informatiker) schreibt $5 \cdot n + 7 = O(n)$
- **Achtung:** Das heißt nicht, dass $f(n) \leq g(n)$, sondern dass $f(n) \leq C \cdot g(n)$ ab einem bestimmten $n > n_0$
- **Intuitiv:** $5 \cdot n + 7$ wächst höchstens "linear"
- Beweis unter Verwendung der Definition von O :
- Es muss gezeigt werden: $\exists n_0, C : 5 \cdot n + 7 \leq C \cdot n \quad \forall n \geq n_0$
- Beweis: $5 \cdot n + 7 \leq 5 \cdot n + n \quad (\text{für } n \geq 7)$
 $\leq 6 \cdot n \quad \square$ \longrightarrow $n_0 = 7$
 $C = 6$
- Alternativer Beweis:

$$5 \cdot n + 7 \leq 5 \cdot n + 7 \cdot n \quad (\text{für } n \geq 1)$$

$$\leq 12 \cdot n \quad \square \quad \longrightarrow \quad n_0 = 1$$

$$C = 12 \quad 8$$

Beispiele Gross-O

- Es interessiert nur der Term höchster Ordnung, der am schnellsten wachsende Summand
- Terme niedriger Ordnung und Konstanten werden ignoriert.
- $f(n)$ muss nur **von oben** durch $c \cdot g(n)$ beschränkt sein.

$$2 \cdot n^2 + 7 \cdot n - 20 \in \mathcal{O}(n^2) \quad (\text{auch} \in \mathcal{O}(n^3), \text{ aber irrelevant})$$

$$\underline{2 \cdot n^2} + 7n \log n - 20 \in \mathcal{O}(n^2)$$

$$7n \log n - 20 \in \mathcal{O}(n \cdot \log n)$$

$$5 \in \mathcal{O}(1)$$

$$2 \cdot n^2 + 7n \log n + n^3 \in \mathcal{O}(n^3)$$

Definition Groß-Omega

■ Groß-Omega, Definition + Beispiel

– **Intuitiv:** Man sagt f ist Groß-Omega von g ...

... wenn f "mindestens so stark wächst wie" g

Also wie Groß-O, nur mit "mindestens" statt "höchstens"

– **Formal:** $\Omega(g)$ ist auch eine Menge von Funktionen:

$$\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R} \mid \exists n_0 \in \mathbb{N} \exists C > 0 \forall n > n_0 : f(n) \geq C \cdot g(n)\}$$

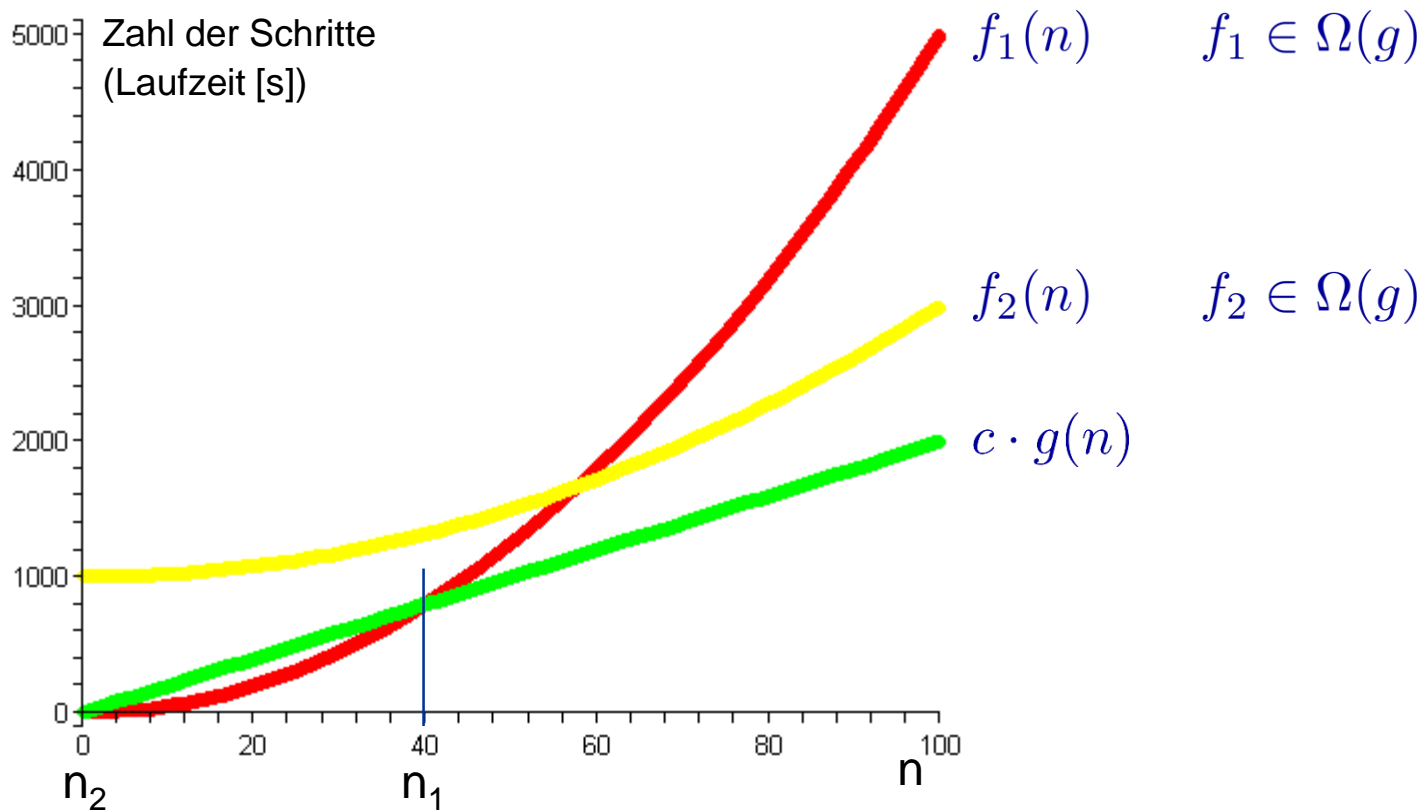
bei $\mathcal{O}(g)$ war das ein " \leq "

– Zum Beispiel $5 \cdot n + 7 = \Omega(n)$

– Beweis unter Verwendung der Definition von Ω :

$$\underbrace{5 \cdot n + 7}_{f(n)} \geq 1 \cdot \underbrace{n}_{g(n)} \quad (\text{für } n \geq 1) \quad \longrightarrow \quad \begin{array}{l} n_0 = 1 \\ C = 1 \end{array}$$

Illustration Groß-Omega



f ist mindestens von der Größenordnung von g .

Beispiele Groß-Omega

- Es interessiert nur der Term höchster Ordnung, der am schnellsten wachsende Summand
- Terme niedriger Ordnung und Konstanten werden ignoriert.
- $f(n)$ muss nur **von unten** durch $c \cdot g(n)$ beschränkt sein.

$$2 \cdot n^2 + 7 \cdot n - 20 \in \Omega(n^2) \quad (\text{auch } \in \Omega(n), \text{ aber irrelevant})$$

$$2 \cdot n^2 + 7n \log n - 20 \in \Omega(n^2)$$

$$7n \log n - 20 \in \Omega(n \cdot \log n)$$

$$5 \in \Omega(1)$$

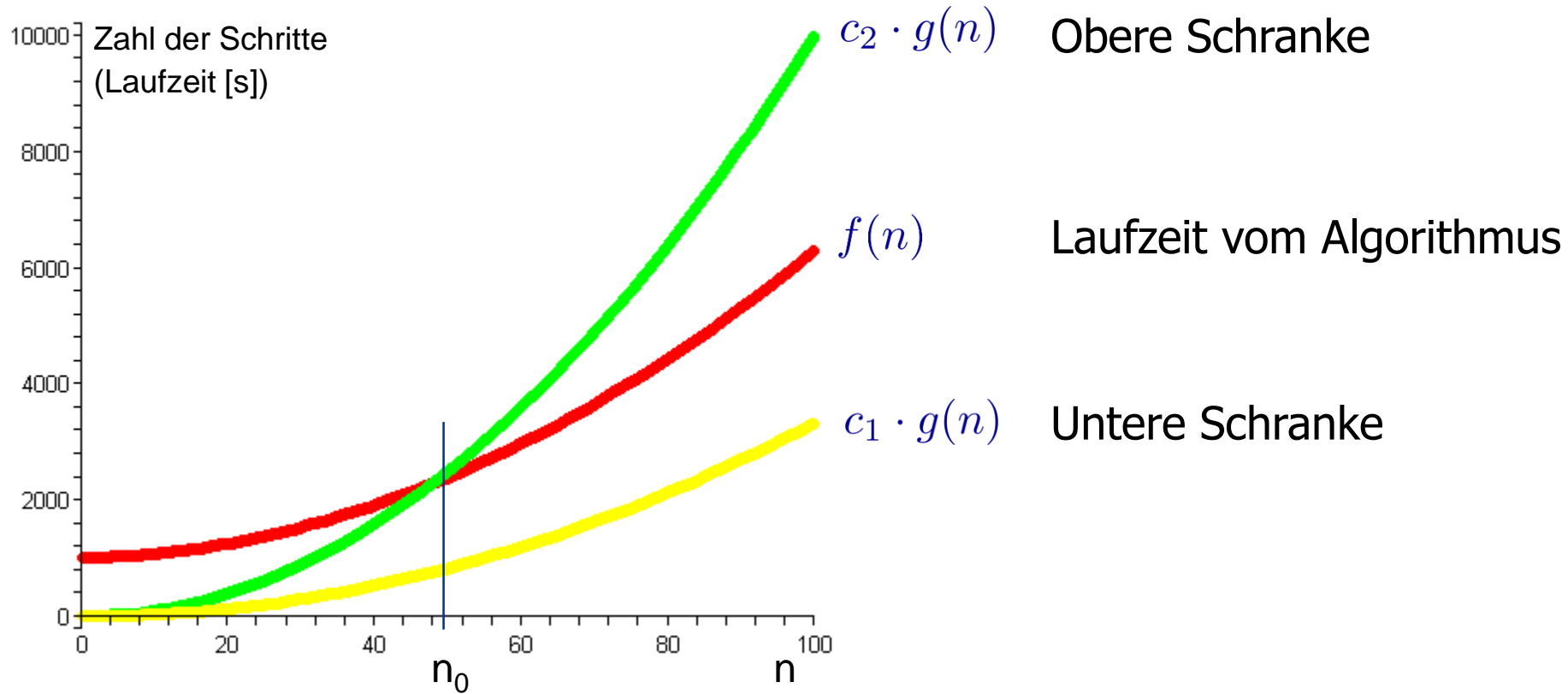
$$2 \cdot n^2 + 7n \log n + n^3 \in \Omega(n^3)$$

Definition Groß-Theta

■ Groß-Theta, Definition + Beispiel

- **Intuitiv:** Man sagt f ist Theta von g ...
... wenn f "genauso so stark wächst wie" g
- **Formal:** Die Menge $\Theta(g)$ ist definiert als
$$\Theta(g) = \mathcal{O}(g) \cap \Omega(g)$$
 $X \cap Y$: die Schnittmenge von X und Y
- Zum Beispiel $5 \cdot n + 7 = \Theta(n)$
- Beweis unter Verwendung der Definition von Θ :
$$f = \mathcal{O}(g) \quad (\text{Beweis siehe vorletzte Folie})$$
$$f = \Omega(g) \quad (\text{Beweis siehe letzte Folie})$$
$$\Rightarrow f = \Theta(g)$$

Illustration Groß-Theta



f und g haben die gleiche Größenordnung

Landau-Symbole Zusammenfassung

- $f = O(g)$: Groß-O
 - f wächst **höchstens** so stark wie g
 - $C \cdot g(n)$ ist die **obere Schranke**

- $f = \Omega(g)$: Groß-Omega
 - f wächst **mindestens** so stark wie g
 - $C \cdot g(n)$ ist die **untere Schranke**

- $f = \Theta(g)$: Groß-Theta
 - f wächst **genau so** stark wie g
 - $C_2 \cdot g(n)$ ist die **obere Schranke** und $C_1 \cdot g(n)$ ist die **untere Schranke**

Typische Funktionenklassen zur Effizienzbeschreibung

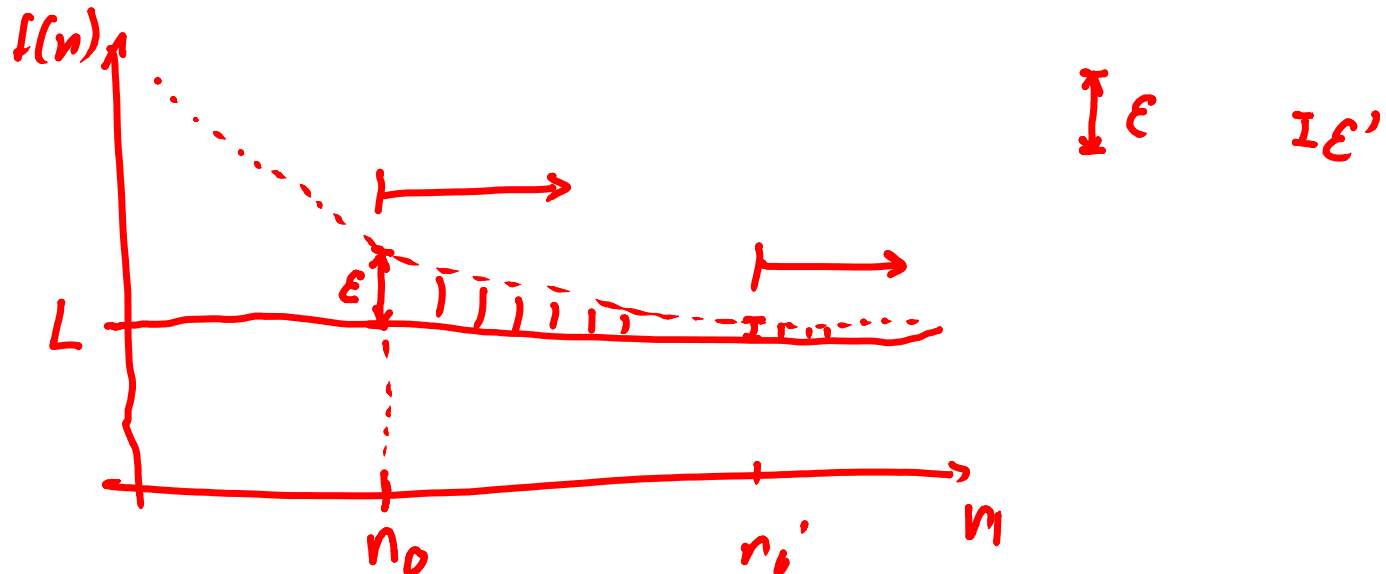


$f \in \Theta(1)$	zeitkonstant, f ist beschränkt
$f \in \Theta(\log n) = \Theta(\log_2 n)$	logarithmisch wachsende Funktionen
$f \in \Theta(n)$	linear wachsende Funktionen
$f \in \Theta(n \log n)$	n-log-n wachsend, superlinear
$f \in \Theta(n^2)$	quadratisch wachsende Funktionen
$f \in \Theta(n^3)$	kubisch wachsende Funktionen
$f \in \Theta(n^k)$	polynomiell wachsende Funktionen
$f \in \Theta(2^n)$	exponentiell wachsende Funktionen

- In den bisherigen Beispielen ...
 - ... haben wir die Zugehörigkeit zu $O(\dots)$ etc. gewissermaßen "zu Fuß" bewiesen, indem wir explizit das n_0 und das C bestimmt haben
 - Die Definitionen erinnern aber sehr an den **Grenzwertbegriff** aus der **Analysis**

■ Grenzwert Definition

- Eine unendliche Folge f_1, f_2, f_3, \dots hat einen Grenzwert L , wenn für alle $\varepsilon > 0$ ein $n_0 \in \mathbb{N}$ existiert so dass für alle $n \geq n_0$ gilt dass $|f_n - L| \leq \varepsilon$
- In Symbolen schreibt man dann $\lim_{n \rightarrow \infty} f_n = L$
- Eine Funktion $f : \mathbb{N} \rightarrow \mathbb{R}$ kann man genauso gut als Folge $f(1), f(2), f(3), \dots$ auffassen und schreibt $\lim_{n \rightarrow \infty} f(n) = L$



- Beispiel für einen Beweis von einem Grenzwert
(sollten Sie eigentlich in [Mathe 1](#) schon mal gesehen haben)

$$\lim_{n \rightarrow \infty} 2 + \frac{1}{n} = 2$$

Ingenieurslösung: ∞ einsetzen: $\frac{1}{\infty} = 0$

Zu zeigen: Für jedes beliebige vorgegebene $\epsilon > 0$
gibt es ein n_0 , so dass für $\forall n \geq n_0$ gilt:

$$\left| 2 + \frac{1}{n} - 2 \right| \leq \epsilon \quad \text{z.B. für } \epsilon = 0.01 \text{ ist } \frac{1}{n} \leq \epsilon \text{ ab } n \geq 100$$

$$\text{allgemein: } n_0 = \left\lceil \frac{1}{\epsilon} \right\rceil$$

damit gilt für $n \geq n_0$:

$$\left| \frac{1}{n} \right| = \frac{1}{n} \leq \frac{1}{n_0} = \frac{1}{\left\lceil \frac{1}{\epsilon} \right\rceil} \leq \frac{1}{\frac{1}{\epsilon}} = \epsilon \quad \square$$

■ Satz

- Seien $f, g : \mathbb{N} \rightarrow \mathbb{R}$ und der Grenzwert $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)}$ existiert (evtl. ist er ∞)
- Dann gelten

$$(1) \quad f = \mathcal{O}(g) \quad \Leftrightarrow \quad \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

$$(2) \quad f = \Omega(g) \quad \Leftrightarrow \quad \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} > 0$$

$$(3) \quad f = \Theta(g) \quad \Leftrightarrow \quad 0 < \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

- Beweis von (1) ... die anderen Beweise gehen analog

$$f = \mathcal{O}(g) \quad \Leftrightarrow \quad \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

Beweis von “ \Rightarrow ”:

$$f = \mathcal{O}(g) \quad \stackrel{\text{def. von } \mathcal{O}}{\Rightarrow} \quad \exists n_0, C \forall n \geq n_0 : f(n) \leq C \cdot g(n)$$

$$\Rightarrow \quad \exists n_0, C \forall n \geq n_0 : \frac{f(n)}{g(n)} \leq C$$

$$\Rightarrow \quad \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \leq C \quad \square$$

O-Notation — Grenzwerte 6/6

$$f = \mathcal{O}(g) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

Beweis von “ \Leftarrow ”:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty \Rightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = C \quad \text{für irgendein } C < \infty$$

$$\stackrel{\text{def. limes}}{\Rightarrow} \exists n_0 \forall n \geq n_0 : \frac{f(n)}{g(n)} \leq C + \epsilon \quad \text{z.B. } \epsilon = 1$$

$$\Rightarrow \exists n_0 \forall n \geq n_0 : f(n) \leq \underline{(C + 1) \cdot g(n)}$$

$$\Rightarrow f = \mathcal{O}(g) \quad \square$$

Die Konst. von
der O-Notation

Rechnen mit Grenzwerten 1/2

■ Variante 1: "zu Fuß"

- Dafür hatten wir gerade das Beispiel $\lim_{n \rightarrow \infty} 2 + \frac{1}{n} = 2$

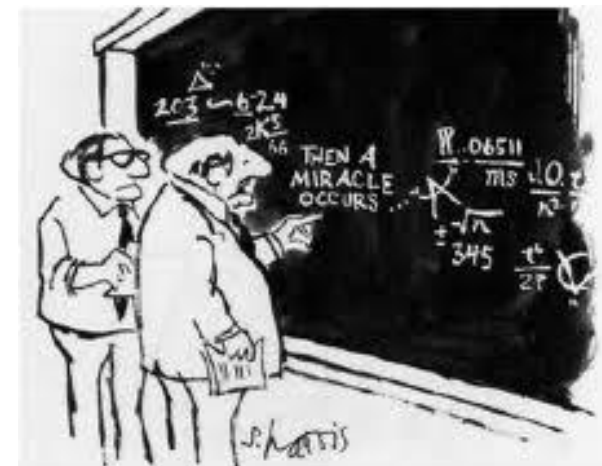
■ Variante 2: Regel von L'Hôpital

- Seien $f, g : \mathbb{N} \rightarrow \mathbb{R}$ wie gehabt
- Es existieren die ersten Ableitungen f' und g' , sowie der Grenzwert $\lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)}$, dann gilt

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

■ Variante 3: göttliche Inspiration

- Erst mit Promotion erlaubt ...



"I think you should be more explicit here in step two."

from *What's so Funny about Science?* by Sidney Harris (1977)

Beispiel L'Hôpital

- Grenzwert lässt sich ingenieurmäßig nicht bestimmen:

$$\lim_{n \rightarrow \infty} \frac{\ln n}{n} = ? \quad \text{einsetzen ergibt: } = \frac{\infty}{\infty}$$

- Mit L'Hôpital: $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)}$

– Zähler: $f(n) := \ln n$, Nenner: $g(n) := n$

– Ableitung vom Zähler existiert: $f'(n) = \frac{1}{n}$

– Ableitung von Nenner existiert: $g'(n) = 1$

– Grenzwert existiert auch: $\lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)} = \lim_{n \rightarrow \infty} \frac{1/n}{1} = \lim_{n \rightarrow \infty} \frac{1}{n} = 0$

– Also ist: $\lim_{n \rightarrow \infty} \frac{\ln n}{n} = \lim_{n \rightarrow \infty} \frac{1}{n} = 0$

- Was darf man ohne Beweis annehmen?
 - Gute Frage!
 - Da gibt es keine klare Regel
 - Im Zweifelsfall immer mehr beweisen als weniger !
 - **Beispiel 1:** $\lim_{n \rightarrow \infty} 1/n = 0$
Brauchen Sie nicht mehr weiter zu beweisen
 - **Beispiel 2:** $\lim_{n \rightarrow \infty} 1/n^2 = 0$
Einfach auf sowas wie Beispiel 1 zurückführen
 - **Beispiel 3:** $\lim_{n \rightarrow \infty} (\log n)/n = 0$
Hier ist ein Argument angebracht, z.B. mit L'Hôpital

- Mit der O-Notation können wir viel einfacher die Laufzeit von Algorithmen abschätzen.
- Dazu zuerst ein paar Rechenregeln und dann praktische Anwendungen

Zusammenfassung Eigenschaften



- Transitivität $f \in \Theta(g) \wedge g \in \Theta(h) \rightarrow f \in \Theta(h)$
 $f \in O(g) \wedge g \in O(h) \rightarrow f \in O(h)$
 $f \in \Omega(g) \wedge g \in \Omega(h) \rightarrow f \in \Omega(h)$
- Reflexivität $f \in \Theta(f)$ $f \in \Omega(f)$ $f \in O(f)$
- Symmetrie $f \in \Theta(g) \leftrightarrow g \in \Theta(f)$
- Austausch-Symmetrie $f \in O(g) \leftrightarrow g \in \Omega(f)$

O-Kalkül

- Einfache Regeln

$$f = O(f)$$

$$kO(f) = O(f)$$

$$O(f + k) = O(f)$$

- Addition

$$O(f) + O(g) = O(\max\{f, g\})$$

- Multiplikation

$$O(f) \cdot O(g) = O(f \cdot g)$$

Laufzeitkomplexität für imperative Sprachkonstrukte



- Problemgröße n in allen Beispielen
- elementare Operationen

<code>i1 := 0;</code>	$O(1)$
-----------------------	--------

- Sequenz elementarer Operationen

<code>i1 := 0;</code>	$O(1)$	$327 \cdot O(1) = O(1)$
<code>i2 := 0;</code>	$O(1)$	
...	...	
<code>i327 := 0;</code>	$O(1)$	

wenn 327 konstant
bzw. unabhängig
von n ist

Laufzeitkomplexität für imperative Sprachkonstrukte



■ Schleife

<pre>for i := 1 to n do begin a[i] := 0; end;</pre>	O(n)	$O(1) \cdot O(n) = O(n)$
	O(1)	

<pre>for i := 1 to n do begin a1[i] := 0; ... a37[i] := 0; end;</pre>	O(n)	O(n)	$O(1) \cdot O(n) = O(n)$
	O(1)	$37 \cdot O(1) = O(1)$	
	...		
	O(1)		

Laufzeitkomplexität für imperative Sprachkonstrukte



■ Schleife

<pre>for i := 1 to n do for j := 1 to n-1 do begin a1[i][j] := j; ... a137[i][j] := i; end;</pre>	$O(n)$	$O(n) \cdot O(n-1) =$	$O(1) \cdot O(n^2) = O(n^2)$
	$O(n-1)$	$O(n) \cdot O(n) = O(n^2)$	
	$O(1)$		
	...	$137 \cdot O(1) = O(1)$	
	$O(1)$		

Laufzeitkomplexität für imperative Sprachkonstrukte



- bedingte Anweisung

<pre>if x<100 then y:=x; else for i:=1 to n if a[i]>y then y:=a[i];</pre>	O(1)	O(1)	O(max{1,n}) = O(n)
	O(1)		
		O(n) · O(1) = O(n)	
	O(n)		
	O(1)		
	O(1)		

Beispiel

Arithmetisches Mittel



- **Eingabe:** Feld x mit n Zahlen
- **Ausgabe:** Feld a mit n Zahlen, wobei $a[i]$ dem arithmetischen Mittel von $x[0]$ bis $x[i]$ entspricht

```
for i:=0 to n-1 do
  begin
    s:=0;
    for j:=0 to i do
      begin
        s:=s+x[j];
      end;
    a[i]:=s/(i+1);
  end;
return a;
```

Laufzeitkomplexität



for i:=0 to n-1 do	O(n)		O(n) · O(i) = O(n ²)
begin		O(n)	
s:=0;	O(1)		
for j:=0 to i do	O(i+1)	O(i)	
s:=s+x[j];	O(1)		
a[i]:=s/(i+1);	O(1)	O(1)	
end;			

i ist von n abhängig, kann nicht als konstant angesehen werden.

- Wie oft werden Anweisungen in der inneren Schleife in Abhängigkeit von der Problemgröße n ausgeführt?

$$1 + 2 + \dots + n = \frac{n(n+1)}{2} = O(n^2)$$

Gaußsche Summenformel

Universität Freiburg - Institut für Informatik - Graphische Datenverarbeitung

■ Sprechweise

- Die O-Notation schaut sich das Verhalten der Funktionen an, wenn $n \rightarrow \infty$ geht (es interessieren nur die $n \geq n_0$)
- Wenn man Laufzeiten, Kosten, etc. als $O(\dots)$ oder $\Omega(\dots)$ oder $\Theta(\dots)$ ausdrückt, spricht man daher von

asymptotischer Analyse

■ Vorsicht

- Asymptotische Analyse sagt nichts über das Laufzeitverhalten bei "kleinen" Eingabegrößen ($n < n_0$) aus
- Für $n < 2$ oder $n < 10$ ist das egal, da wird schon nichts Schlimmes passieren
- Aber das n_0 ist nicht immer so klein ... nächste Folie

■ Beispiel

- Algorithmus A hat Laufzeit $f(n) = 80 \cdot n$
- Algorithmus B hat Laufzeit $g(n) = 2 \cdot n \cdot \log_2 n$
- Dann ist $f = O(g)$ aber **nicht** $f = \Theta(g)$
- Das heißt, A ist asymptotisch schneller als B
 - d.h. ab irgendeinem n_0 ist für alle $n \geq n_0$ $f(n) \leq g(n)$
- Berechnung von n_0 : bei n_0 schneiden sich die beiden Funktionen:

$$f(n_0) = g(n_0)$$

$$80 \cdot n_0 = 2 \cdot n_0 \cdot \log_2 n_0$$

$$40 = \log_2 n_0$$

$$n_0 = 2^{40} = (2^{10})^4 = (1024)^4 \approx (10^3)^4 = 10^{12} = 1 \text{ Billionen}$$

- D.h., A ist erst ab 1 Billionen Elemente schneller als B

Literatur / Links

■ O-Notation / Ω -Notation / Θ -Notation

– In Mehlhorn/Sanders:

2.1 Asymptotic Notation

– In Cormen/Leiserson/Rivest

2.1 Asymptotic Notation

– In Wikipedia

http://en.wikipedia.org/wiki/Big_O_notation

<http://de.wikipedia.org/wiki/Landau-Symbole>

Übungsblatt 3

- Nochmal Beweise.
- Solche Aufgaben kommen mit Sicherheit in der Klausur vor!