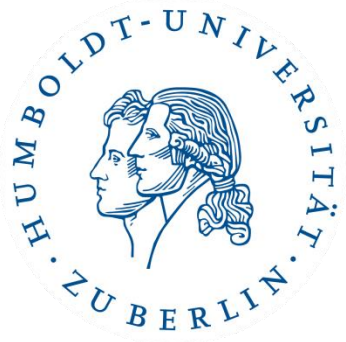
Übung Algorithmen und Datenstrukturen



Sommersemester 2017

Marc Bux, Humboldt-Universität zu Berlin

Organisatorisches

Vorlesung:	Montag	11 – 13 Uhr	Ulf Leser	RUD 26, 0‘115	
	Mittwoch	11 – 13 Uhr	Ulf Leser	RUD 26, 0‘115	
Übung:	Montag	13 – 15 Uhr	Berit Grußien	RUD 26, 1'303	Gruppe 1
	Montag	13 – 15 Uhr	Marc Bux	RUD 26, 1'305	Gruppe 2
	Dienstag	13 – 15 Uhr	Marc Bux	RUD 26, 0'313	Gruppe 3
	Dienstag	13 – 15 Uhr	Berit Grußien	RUD 26, 1'303	Gruppe 4
	Mittwoch	13 – 15 Uhr	Patrick Schäfer	RUD 26, 0'313	Gruppe 5
	Mittwoch	15 – 17 Uhr	Patrick Schäfer	RUD 26, 0'313	Gruppe 6
	Donnerstag	09 – 11 Uhr	Lucas Heimberg	RUD 26, 0'313	Gruppe 7
	Freitag	09 – 11 Uhr	Lucas Heimberg	RUD 26, 1'303	Gruppe 8

Klausur: 7. August und 29. September 2017, jeweils 11 – 15 Uhr

Websites:	Übung:	http://hu.berlin/algodat17
	Vorlesung:	http://hu.berlin/vl_algodat17
	Moodle:	http://moodle.hu-berlin.de

Inhalt dieser Veranstaltung

- Heute (KW 17):
  - Organisatorisches
  - Vorbereitung Aufgabenblatt 1, Aufgaben 1 & 2
- Nächste Woche (KW 18):
  - Vorbereitung Aufgabenblatt 1, Aufgaben 3 & 4
- KW 19, 21, 23, 25, 27:
  - Vorbereitung Aufgabenblatt n in KW 15 + 2n
  - Abgabe von Aufgabenblatt n in KW 17 + 2n
- KW 20, 22, 24, 26, 28:
  - Vorrechnen von Aufgabenblatt n in KW 18 + 2n
  - jeder kommt mal dran
  - Freiwillige vor, ansonsten wird gelost (seid vorbereitet!)
- KW 29: Vorrechnen von Aufgabenblatt 6
- Außerdem:
  - Rückgabe korrigierter Abgaben

Übungsaufgaben

- 6 Blätter mit je 50 Punkten
  - Veröffentlichung alle zwei Wochen in Moodle und auf der Website
  - jedes Aufgabenblatt muss bearbeitet werden
  - Abgabe des ersten Aufgabenblatts: bis 8. Mai
- Lösungen schriftlicher Aufgaben separat auf Papier abgeben
  - Blätter einer Aufgabe zusammentackern
  - Abgabe vor der Vorlesung bis 11:10 Uhr
  - oder bis 10:45 Uhr im Briefkasten bei Raum RUD25, 3.321
  - vorher kopieren oder abfotografieren (zur Vorstellung in der Übung)
- Programmieraufgaben (Java 8) in Moodle abgeben
  - vorher auf gruenau2 testen
  - gleicher Termin wie schriftliche Aufgaben
- auf jedes Blatt / jede Java-Datei jeder Abgabe:
  - Namen
  - CMS-Benutzernamen
  - Moodle-Arbeitsgruppe
  - gewünschten Rückgabetermin (eindeutig: Wochentag, Uhrzeit, Dozent)

Übungsschein

- 50% der Punkte (150) = Übungsschein = Prüfungszulassung
- Anmeldung in Agnes
  - Keine Zulassungsgarantie zum Wunschtermin
- Anmeldung in Moodle
  - Einschreibeschlüssel: Montag MoB.y9tknp / Dienstag DiB.3rvh2v
  - Bildung von Abgabegruppen bestehend aus zwei (in Ausnahmefällen drei) Studenten
  - Abgabegruppen können sich über mehrere Termine erstrecken
  - Wichtige Nachrichten werden in den Moodle-Foren angekündigt oder über Moodle versendet (ggf. Weiterleitung einrichten)
- Abgaben ohne Anmeldung in Moodle: 0 Punkte
- Abgaben mit invalider Gruppengröße: 0 Punkte
- bei vermutetem Abschreiben: 0 Punkte
- nicht ausführbare Programmieraufgaben: 0 Punkte

Aufgaben für Euch

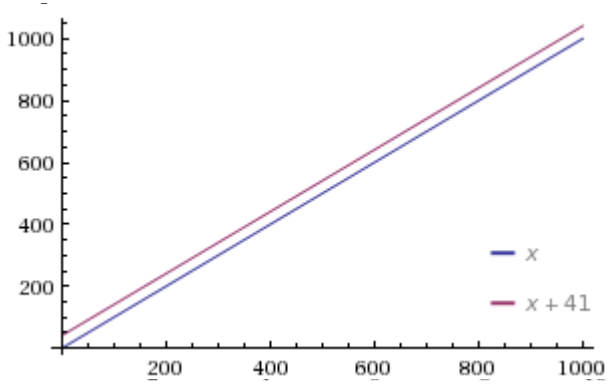
- jetzt:
  - wer hat den Übungsschein bereits?
  - wer hat noch keinen Übungspartner?
- zu nächster Woche:
  - noch kein bestätigter Termin in Agnes: Für eine der Übungsgruppen 6-8 entscheiden und Berit Grußien anschreiben
  - noch kein Übungspartner: Übungspartner finden (z.B. über den dafür eingerichteten Post im Moodle-Forum)
  - in Moodle anmelden und Abgabegruppe bilden
  - dafür sorgen, dass über Agnes und Moodle versandte Nachrichten in einem regelmäßig aberufenem Postfach ankommen
  - O-Tutorial auf der Übungswebsite durchlesen
  - Grenzwerte, Ableitungen, Potenz- und Logarithmusgesetze wiederholen
  - mit der Bearbeitung von Aufgabenblatt 1 beginnen

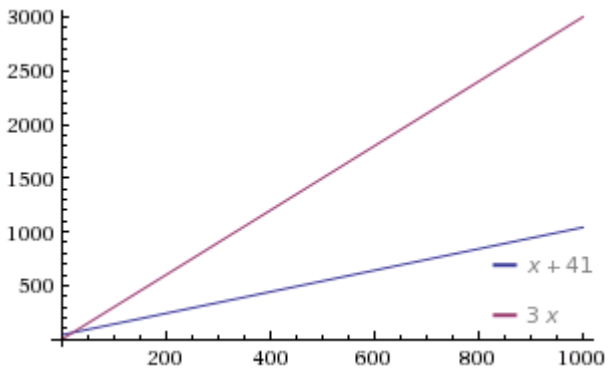
Agenda

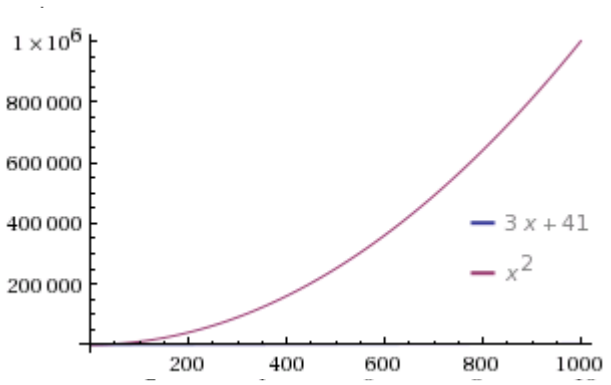
1. Organisatorisches
2. Die Landau-Notation
3. Vorbereitung Aufgabenblatt 1, Aufgabe 1
4. Vorbereitung Aufgabenblatt 1, Aufgabe 2

Zeitkomplexität

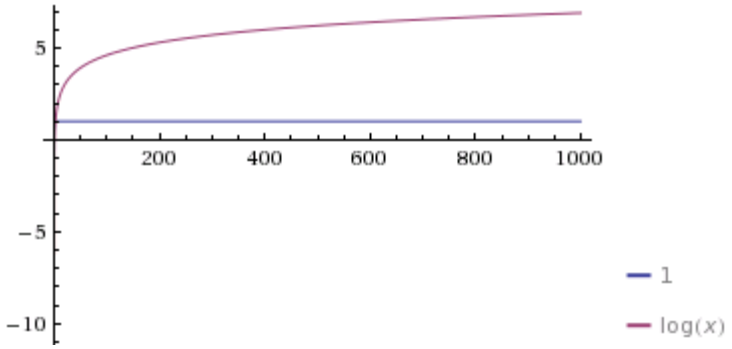
- Ziel: Abschätzung der Anzahl notwendiger Operationen (und damit der Laufzeit) eines Algorithmus
  - wird definiert als Funktion der Eingabe
  - üblicherweise im schlechtesten Fall (Worst Case), d.h. im Falle der ungünstigsten Eingabe
  - unabhängig von Hardware und Implementierung
- Grundidee: Welche Terme bestimmen die Laufzeit des Algorithmus, wenn die Eingabe sehr groß wird
  - nur der Term mit dem größten asymptotischen Wachstum (für $n \to \infty$) ist interessant
  - konstante Summanden und Faktoren sind unerheblich

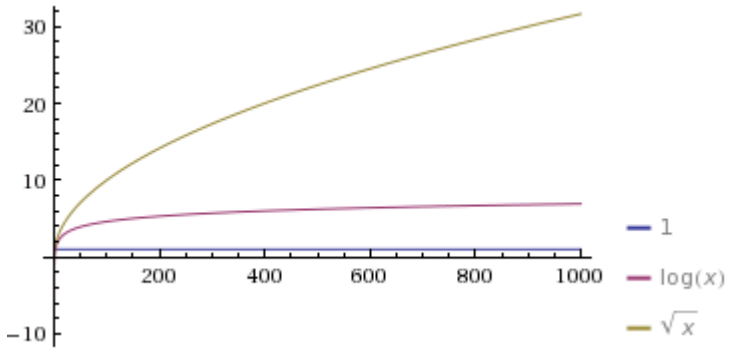


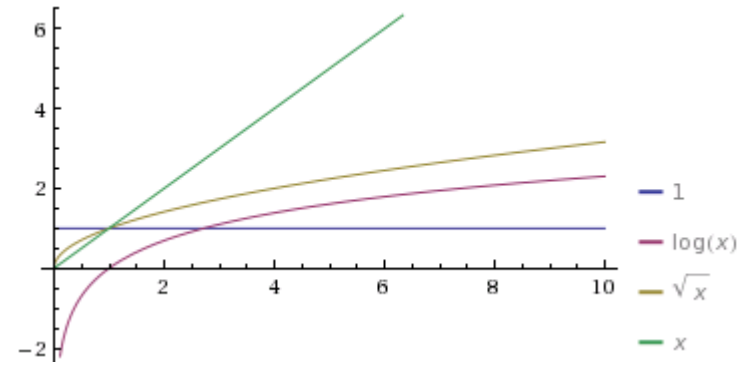


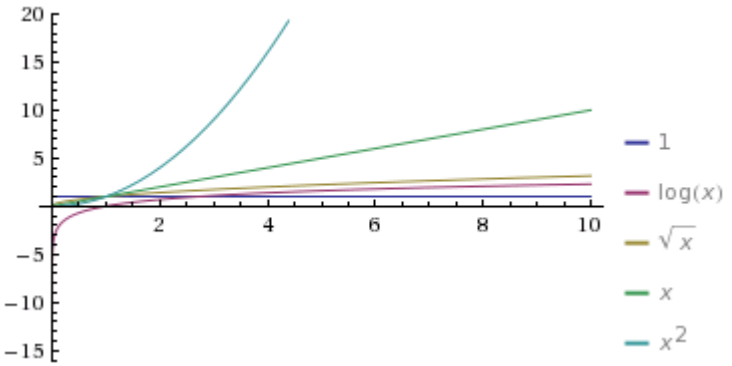


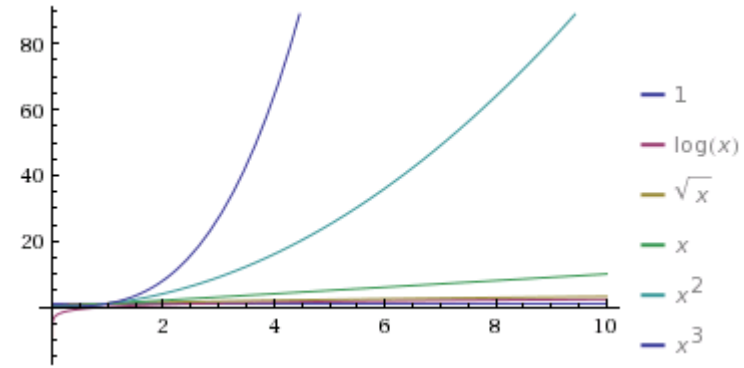
Einige gängige Funktionen

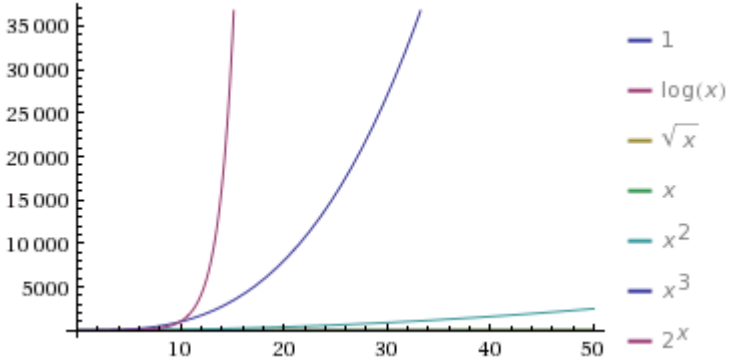












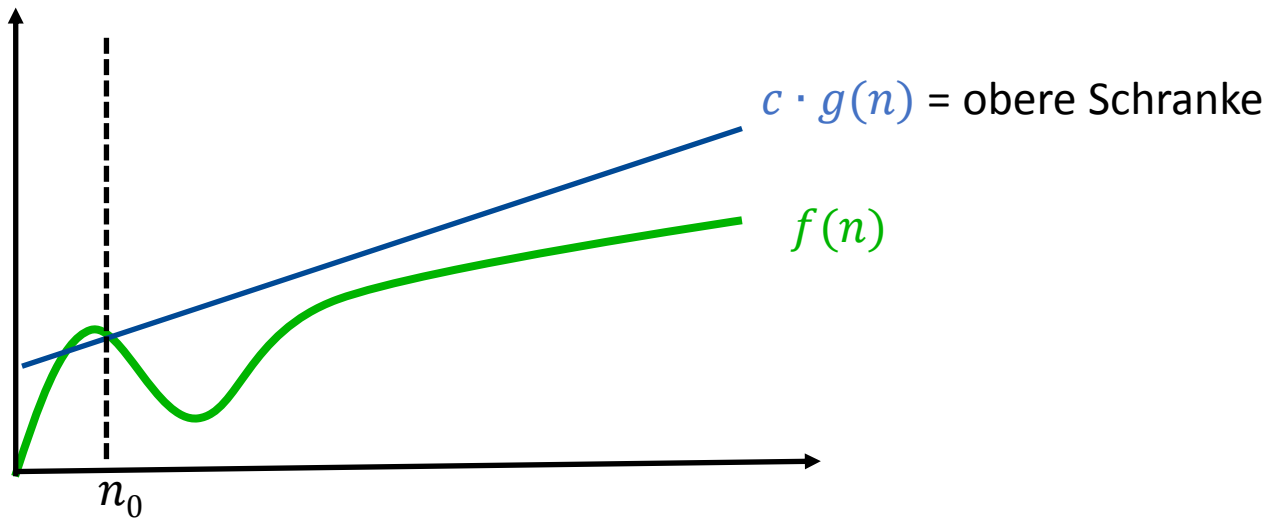
Die Landau-Notation (O-Notation)

- seien f und g Funktionen von $\mathbb{N}$ nach $\mathbb{R}_{\geq 0}$
- wir schreiben $f \in O(g)$, falls es Zahlen $c \in \mathbb{R}^+$, $n_0 \in \mathbb{N}$ gibt, so dass für alle $n \geq n_0$ gilt: $f(n) \leq c \cdot g(n)$
- Bedeutung:
  - „g wächst mindestens so schnell wie f“
  - „g ist obere Schranke für f“
- Beispiele: $4n^2 \in O(n^3)$, $4n^2 \in O(n^2)$, $n^3 \notin O(4n^2)$
- Scharfe obere Schranken angeben: $4n^2 \in O(n^2)$ sagt mehr als $4n^2 \in O(n^3)$
- $O(g)$ ist die Menge aller Funktionen, welche die obige Bedingungen erfüllen
  - $O(g) = \{f: \mathbb{N} \to \mathbb{R}_{\geq 0} \mid \exists c \in \mathbb{R}^+ \ \exists n_0 \in \mathbb{N} \ \forall n \geq n_0: \ f(n) \leq c \cdot g(n)\}$

O-Notation: Menge von Funktionen

$$\underbrace{\mathrm{O}(g)}_{a_1} = \{\underbrace{f: \mathbb{N} \to \mathbb{R}_{\geq 0}}_{a_2} | \underbrace{\exists c \in \mathbb{R}^+}_{a_3} \underbrace{\exists n_0 \in \mathbb{N}}_{a_4} \underbrace{\forall n \geq n_0:}_{a_5} \underbrace{f(n) \leq c \cdot g(n)}_{a_6}\}$$

- a_1: Wir definieren $\mathrm{O}(g)$
- a_2: als Menge aller Funktionen f, für die gilt:
- a_3 und a_4: es existieren zwei Konstanten c und n_0,
- a_5: sodass für alle Werte n ab n_0 gilt, dass
- a_6: $c \cdot g(n)$ obere Schranke für $f(n)$ ist.



Wichtige Komplexitätsklassen

- $\mathrm{O}(1)$: konstant (Array-Zugriff)
- $\mathrm{O}(\log n)$: logarithmisch (Binäre Suche)
- $\mathrm{O}(n)$: linear (Sequentielle Suche)
- $\mathrm{O}(n \log n)$: linear-logarithmisch (MergeSort)
- $\mathrm{O}(n^2)$: quadratisch (BubbleSort)
- $\mathrm{O}(n^k)$: polynomiell
- $\mathrm{O}(2^n)$: exponentiell (Traveling Salesman)

Weitere Landau-Terme

- Definitionen:
  - $O(g) = \{f: \mathbb{N} \to \mathbb{R}_{\geq 0} \mid \exists c \in \mathbb{R}^+ \ \exists n_0 \in \mathbb{N} \ \forall n \geq n_0: \ f(n) \leq c \cdot g(n)\}$
  - $\Omega(g) = \{f: \mathbb{N} \to \mathbb{R}_{\geq 0} \mid \exists c \in \mathbb{R}^+ \ \exists n_0 \in \mathbb{N} \ \forall n \geq n_0: \ f(n) \geq c \cdot g(n)\}$
  - $\Theta(g) = \{f: \mathbb{N} \to \mathbb{R}_{\geq 0} \mid f \in O(g) \wedge f \in \Omega(g)\}$
  - $o(g) = \{f: \mathbb{N} \to \mathbb{R}_{\geq 0} \mid \forall c \in \mathbb{R}^+ \ \exists n_0 \in \mathbb{N} \ \forall n \geq n_0: \ f(n) < c \cdot g(n)\}$
  - $\omega(g) = \{f: \mathbb{N} \to \mathbb{R}_{\geq 0} \mid \forall c \in \mathbb{R}^+ \ \exists n_0 \in \mathbb{N} \ \forall n \geq n_0: \ f(n) > c \cdot g(n)\}$

- Bedeutung: „g wächst ...
  - mindestens so schnell wie f (ist obere Schranke für f)“
  - höchstens so schnell wie f (ist untere Schranke für f)“
  - ungefähr genauso schnell wie f“
  - schneller als f“
  - langsamer als f“

Beispiel 1

- Definitionen (kurz):
  - $f \in O(g) \Leftrightarrow \exists c \exists n_0 \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
  - $f \in \Omega(g) \Leftrightarrow \exists c \exists n_0 \forall n \geq n_0 : f(n) \geq c \cdot g(n)$
  - $f \in \Theta(g) \Leftrightarrow f \in O(g) \wedge f \in \Omega(g)$
- Funktionen
  - $f(n) = k$ mit $k > 0$
  - $g(n) = 1$
- wähle $c = k, n_0 = 0$

  $\forall n \geq n_0 : f(n) \leq c \cdot g(n)$ | da $k \leq k \cdot 1$

  $\Rightarrow f \in O(g) = O(1)$

  $\forall n \geq n_0 : f(n) \geq c \cdot g(n)$ | da $k \geq k \cdot 1$

  $\Rightarrow f \in \Omega(g) = \Omega(1)$

  $\Rightarrow f \in \Theta(g) = \Theta(1)$
- konstante Funktionen wachsen alle asymptotisch gleich schnell (nämlich gar nicht)

Beispiel 2

- Definitionen (kurz):
  - $f \in O(g) \Leftrightarrow \exists c \exists n_0 \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
  - $f \in \Omega(g) \Leftrightarrow \exists c \exists n_0 \forall n \geq n_0 : f(n) \geq c \cdot g(n)$
  - $f \in \Theta(g) \Leftrightarrow f \in O(g) \wedge f \in \Omega(g)$
- Funktionen
  - $f(n) = 3n^5 + 4n^3 + 15$
  - $g(n) = n^5$
- wähle $c = 3 + 4 + 15 = 22, n_0 = 1$

  $\forall n \geq n_0 : f(n) \leq c \cdot g(n)$ | da $3n^5 + 4n^3 + 15 \leq 22 \cdot n^5$ für $n \geq 1$

  $\Rightarrow f \in O(g) = O(n^5)$
- wähle $c = 3, n_0 = 1$

  $\forall n \geq n_0 : f(n) \geq c \cdot g(n)$ | da $3n^5 + 4n^3 + 15 \geq 3 \cdot n^5$ für $n \geq 1$

  $\Rightarrow f \in \Omega(g) = \Omega(n^5)$

  $\Rightarrow f \in \Theta(g) = \Theta(n^5)$

Beispiel 2 (2)

- Definitionen (kurz):
  - $f \in O(g) \Leftrightarrow \exists c \exists n_0 \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
  - $f \in \Omega(g) \Leftrightarrow \exists c \exists n_0 \forall n \geq n_0 : f(n) \geq c \cdot g(n)$
  - $f \in \Theta(g) \Leftrightarrow f \in O(g) \wedge f \in \Omega(g)$
  - $f \in o(g) \Leftrightarrow \forall c \exists n_0 \forall n \geq n_0 : f(n) < c \cdot g(n)$
  - $f \in \omega(g) \Leftrightarrow \forall c \exists n_0 \forall n \geq n_0 : f(n) > c \cdot g(n)$
- Funktionen
  - $f(n) = 3n^5 + 4n^3 + 15$
  - $g(n) = n^5$
- wähle $c = 3$

  $\neg(\forall c : f(n) < c \cdot g(n))$ | da $3n^5 + 4n^3 + 15 > 3 \cdot n^5$ für $n \geq 1$

  $\Rightarrow f \notin o(g) = o(n^5)$
- wähle $c = 3 + 4 + 15 = 22$

  $\neg(\forall c : f(n) > c \cdot g(n))$ | da $3n^5 + 4n^3 + 15 < 22 \cdot n^5$ für $n \geq 1$

  $\Rightarrow f \notin \omega(g) = \omega(n^5)$

Zusammenhänge zwischen O, Ω, Θ, o und ω

- Definitionen (kurz):
  - $f \in O(g) \Leftrightarrow \exists c \exists n_0 \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
  - $f \in \Omega(g) \Leftrightarrow \exists c \exists n_0 \forall n \geq n_0 : f(n) \geq c \cdot g(n)$
  - $f \in o(g) \Leftrightarrow \forall c \exists n_0 \forall n \geq n_0 : f(n) < c \cdot g(n)$
  - $f \in \omega(g) \Leftrightarrow \forall c \exists n_0 \forall n \geq n_0 : f(n) > c \cdot g(n)$
- Satz: $f \in O(g) \Leftrightarrow g \in \Omega(f)$

  Beweis: $f \in O(g)$

  $\Leftrightarrow \exists c\ \exists n_0\ \forall n \geq n_0 : f(n) \leq c \cdot g(n)$

  $\Leftrightarrow$ für $c' = \frac{1}{c}$ und das gleiche n_0 gilt: $\forall n \geq n_0 : g(n) \geq c' \cdot f(n)$

  $\Leftrightarrow g \in \Omega(f)$ ■
- Satz: $f \in o(g) \Leftrightarrow g \in \omega(f)$
- Satz: $f \in o(g) \Rightarrow f \notin \Omega(g)$

  Beweis: $f \in o(g)$

  $\Rightarrow \forall c\ \exists n_0\ \forall n \geq n_0 : f(n) < c \cdot g(n)$

  $\Rightarrow$ es existiert kein c', so dass für ein n_0 gilt: $\forall n \geq n_0 : f(n) \geq c' \cdot g(n)$

  $\Rightarrow f \notin \Omega(g)$ ■
- Satz: $f \in \omega(g) \Rightarrow f \notin O(g)$

vgl. Blatt 1, Aufgabe 2

Grenzwert als hinreichendes Kriterium

- Satz: $\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty \Rightarrow f \in O(g)$
  - „g wächst mindestens so schnell wie f"
- Satz: $\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0 \Rightarrow f \in \Omega(g)$
  - „g wächst höchstens so schnell wie f"
- Satz: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Leftrightarrow f \in o(g)$
  - „g wächst schneller als f"
- Satz: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Leftrightarrow f \in \omega(g)$
  - „g wächst langsamer als f"
- Beispiel:

  $f(n) = 23$

  $g(n) = \log\log n$

  $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{23}{\log\log n} = 0$

  $\Rightarrow f \in o(g)$

  $\Rightarrow f \notin \Omega(g)$ und $g \in \omega(f)$

Logarithmen und Satz von L'Hôpital

- Logarithmengesetz für Produkte: $\log_b x^r = r \cdot \log_b x$
- $\log_a n = \frac{\log_a n \cdot \log_b a}{\log_b a} = \frac{\log_b a^{\log_a n}}{\log_b a} = \frac{\log_b n}{\log_b a} = \frac{1}{\log_b a} \cdot \log_b n$
  - Logarithmen zu verschiedenen Basen können mit einem konstanten Faktor ineinander umgerechnet werden
  - $\log_a n = \Theta(\log_b n)$
- Satz von L'Hôpital: Seien f und g zwei differenzierbare Funktionen, deren Grenzwerte entweder beide gegen 0 oder beide gegen ∞ gehen. Dann gilt $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$ (falls der Grenzwert existiert).
- Beispiel:
  - $f(n) = \log n \ (= \log_2 n)$
  - $g(n) = \sqrt{n}$

    $\lim_{n\to\infty} \log n = \lim_{n\to\infty} \sqrt{n} = \infty$

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

    $\Rightarrow f \in o(g), f \notin \Omega(g), g \in \omega(f)$

vgl. Blatt 1, Aufgabe 1

Agenda

1. Organisatorisches
2. Die Landau-Notation
3. Vorbereitung Aufgabenblatt 1, Aufgabe 1
4. Vorbereitung Aufgabenblatt 1, Aufgabe 2

Vorbereitung Aufgabenblatt 1, Aufgabe 1

Beweisen oder widerlegen Sie für die Teilaufgaben (a) bis (d) die folgenden Aussagen: $f \in \mathrm{O}(g)$, $f \in \Omega(g)$.

	$f(n)$	$g(n)$
(a)	$\sqrt{n}$	$n^{\frac{3}{4}}$
(b)	$\sqrt{n} + \log n$	$2\sqrt{n}$
(c)	2^n	2^{n+1}
(d)	2^n	$n!$

Agenda

1. Organisatorisches
2. Die Landau-Notation
3. Vorbereitung Aufgabenblatt 1, Aufgabe 1
4. Vorbereitung Aufgabenblatt 1, Aufgabe 2

Vorbereitung Aufgabenblatt 1, Aufgabe 2

Seien f und g Funktionen, die von $\mathbb{N}$ nach $\mathbb{R}^+$ abbilden.
Beweisen Sie die folgenden Aussagen:

1. Wenn $f \in O(g)$ und $g \in O(h)$, dann gilt $f \in O(h)$.
2. Seien $a, b \in \mathbb{R}^+$ Konstanten, dann gilt $a^{n+b} \in \Theta(a^n)$.

Ausblick

- zu nächster Woche:
  - noch kein bestätigter Termin in Agnes: Für eine der Übungsgruppen 6-8 entscheiden und Berit Grußien anschreiben
  - noch kein Übungspartner: Übungspartner finden (z.B. über den dafür eingerichteten Post im Moodle-Forum)
  - in Moodle anmelden und Abgabegruppe bilden
  - dafür sorgen, dass über Agnes und Moodle versandte Nachrichten in einem regelmäßig aberufenem Postfach ankommen
  - O-Tutorial auf der Übungswebsite durchlesen
  - Grenzwerte, Ableitungen, Potenz- und Logarithmusgesetze wiederholen
  - mit der Bearbeitung von Aufgabenblatt 1 beginnen
- nächste Woche:
  - Wiederholung und Vertiefung der O-Notation
  - Vorbereitung Aufgabenblatt 1, Aufgaben 3 & 4 (Pseudocode-Analyse, Algorithmenentwurf)
  - Klärung von Fragen zu Aufgabenblatt 1