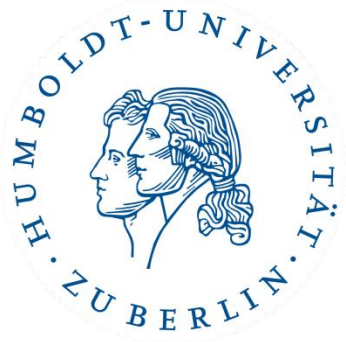
# Übung Algorithmen und Datenstrukturen




Sommersemester 2016

Marc Bux, Humboldt-Universität zu Berlin

# Organisation

| | | | | |
|---|---|---|---|---|
| Vorlesung: | Montag | 11 – 13 Uhr | Marius Kloft | RUD 26, 0‘115 |
| | Mittwoch | 11 – 13 Uhr | Marius Kloft | RUD 26, 0‘115 |
| Übung: | Montag | 09 – 11 Uhr | Marc Bux | RUD 26, 1'303 |
| | Montag | 13 – 15 Uhr | Marc Bux | RUD 26, 1'305 |
| | Montag | 13 – 15 Uhr | Florian Tschorsch | RUD 26, 1'303 |
| | Dienstag | 09 – 11 Uhr | Patrick Schäfer | RUD 26, 1'303 |
| | Dienstag | 13 – 15 Uhr | Kim Völlinger | RUD 26, 0'313 |
| | Mittwoch | 09 – 11 Uhr | Berit Grußien | RUD 26, 1'306 |
| | Donnerstag | 13 – 15 Uhr | Kim Völlinger | RUD 26, 1'305 |
| | Donnerstag | 13 – 15 Uhr | Patrick Schäfer | RUD 26, 0'313 |
| | Freitag | 09 – 11 Uhr | Berit Grußien | RUD 26, 1'305 |
| | Freitag | 11 – 13 Uhr | Florian Tschorsch | RUD 26, 1'305 |
| Tutorium: | Montag | 17 – 19 Uhr | Michael R. Jung | RUD 26, 1'303 |
| | Mittwoch | 17 – 19 Uhr | Michael R. Jung | RUD 26, 1'303 |
| | Donnerstag | 15 – 17 Uhr | Michael R. Jung | RUD 26, 1'306 |
| | Freitag | 11 – 13 Uhr | Michael R. Jung | RUD 25, 3.101 |

Klausur: 1. August und 4. Oktober 2016, jeweils 9 – 12 Uhr

# Inhalt dieser Veranstaltung

- Heute (KW 17): Organisatorisches, Vorbereitung und Besprechung 1. Übungsblatt
- In KW 18, 20, 22, 24, 26:
  - Vorbereitung und Besprechung von Übungsblatt $n$ in KW $14 + 2n$
  - Abgabe von Übungsblatt $n$ in KW $16 + 2n$
  - Rückgabe der Lösungen von Übungsblatt $n$ in KW $18 + 2n$
- In KW 19, 21, 23, 25, 27, 29:
  - Besprechung / Vorrechnen der Lösungen von Übungsblatt $n$ in KW $17 + 2n$
- Wenn anderer Termin besser passt oder Übung ausfällt: zu anderer Übung gehen (so lange es genügend freie Plätze gibt)
- Websites:
  - Übung: https://hu.berlin/algodat16
  - Vorlesung: https://hu.berlin/vl_algodat16
  - Tutorium: https://hu.berlin/tutAD16

# Übungsaufgaben

- 6 Blätter mit je 50 Punkten
- Ausgabe zwei Wochen vor Abgabetermin in Goya und auf der Website
  - Erstes Übungsblatt: Seit 20. April
- Abgabe des ersten Übungsblatts: Bis 4. Mai
  - Danach immer Montags alle zwei Wochen
- Lösungen schriftlicher Aufgaben sind auf Papier mit nach Aufgaben getrennten Blättern abzugeben
  - Blätter einer Aufgabe zusammentackern
- Abgabe schriftlicher Aufgaben vor der Vorlesung bis 10:55 Uhr
  - Oder vorher im Briefkasten in Raum RUD25, 3.321
- Programmieraufgaben (Java 1.7) sind auf gruenau2 zu testen und in Goya abzugeben (gleicher Termin wie schriftliche Aufgaben)
- Namen, Matrikelnummer, Übungsgruppe, gewünschten Rückgabetermin (eindeutig: Wochentag, Uhrzeit, Dozent) auf jedem Blatt / jeder Java-Klasse jeder Abgabe angeben

# Übungsschein und Goya

- Übungsschein = Zulassung zur Prüfung
- Insgesamt müssen 50% der Punkte erreicht werden (150 Punkte)
- Anmeldung in Goya
  - nicht auf Warteliste (Wartelisten werden am 2.4. geschlossen)
  - zur Not zu einem beliebigen Termin eintragen und in Goya Wechsel beantragen
- Gruppenbildung in Goya
  - Gruppen können sich über mehere Termine erstrecken
  - Möglichst einfache / kurze Gruppennamen verwenden
- Wichtige Informationen werden über Goya verschickt
  - Ggf. Weiterleitung aktivieren
- Abgaben ohne Anmeldung in Goya: 0 Punkte
- Abgaben mit invalider Gruppengröße: 0 Punkte
- Bei vermutetem Abschreiben: 0 Punkte
- Nicht ausführbare Programmieraufgaben: 0 Punkte

# Übungsgruppen

- Übungsblätter sind in Gruppen zu bearbeiten
  - Zusammensetzung: zwei (in Ausnahmefällen drei) Studenten
- Ziel: Zusammenarbeit fördern und voneinander lernen
- Als Absicherung und um die eigene Abgabe in der Übung parat zu haben: Abgabe kopieren
- Wenn Übungspartner sich nicht fair beteiligt: früh Bescheid geben
- Wer hat den Übungsschein bereits?
- Wer steht noch auf einer Warte- oder Vormerkliste?
- Wer hat noch keinen Übungspartner?

# Agenda

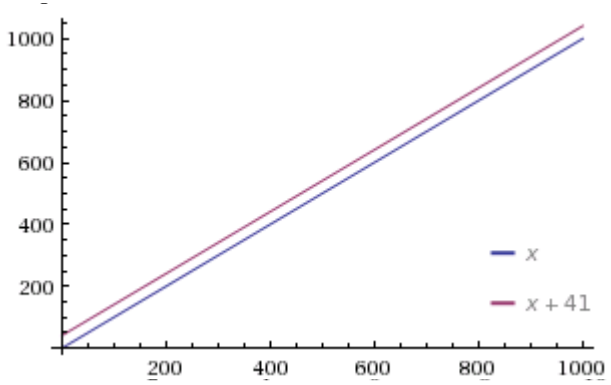
1. Organisatorisches
2. Vorstellung des ersten Übungsblatts
3. Die Landau-Notation
4. Pseudocode-Analyse und Algorithmenentwurf

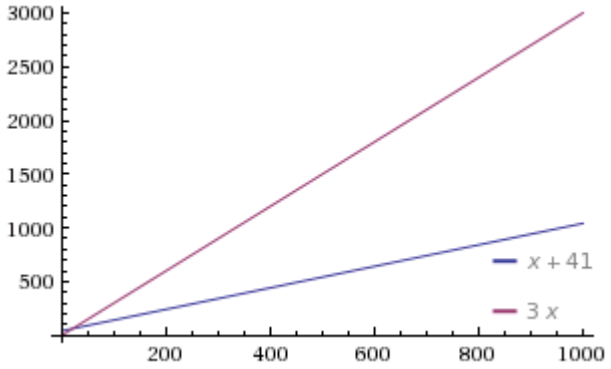
# Agenda

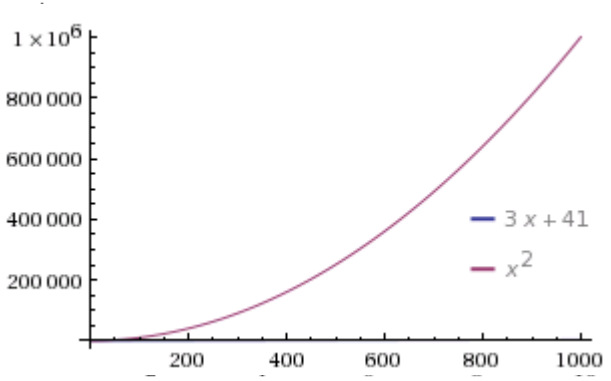
1. Organisatorisches
2. Vorstellung des ersten Übungsblatts
3. Die Landau-Notation
4. Pseudocode-Analyse und Algorithmenentwurf

# Zeitkomplexität

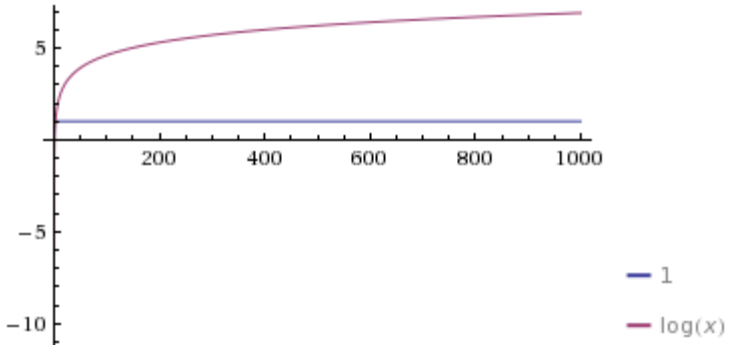
- Ziel: Abschätzung der Anzahl notwendiger Operationen eines Algorithmus
  - als Funktion der Eingabe
  - (üblicherweise) im schlechtesten Fall
  - unabhängig von Hardware und Implementierung
- Grundidee: Welche Terme bestimmen die Laufzeit des Algorithmus, wenn die Eingabe sehr groß wird
  - Nur der Term mit dem größten Wachstum ist interessant
  - Konstante Summanden und Faktoren sind unerheblich

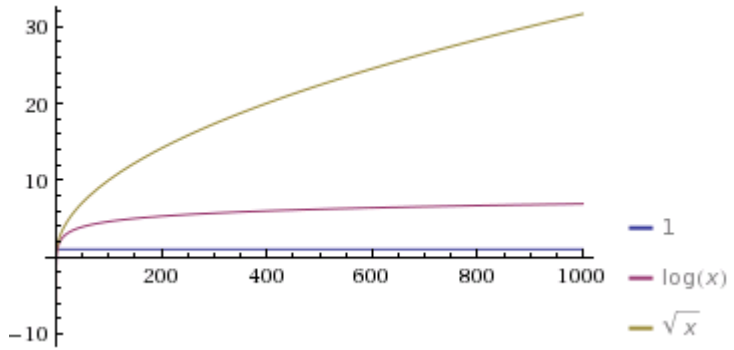


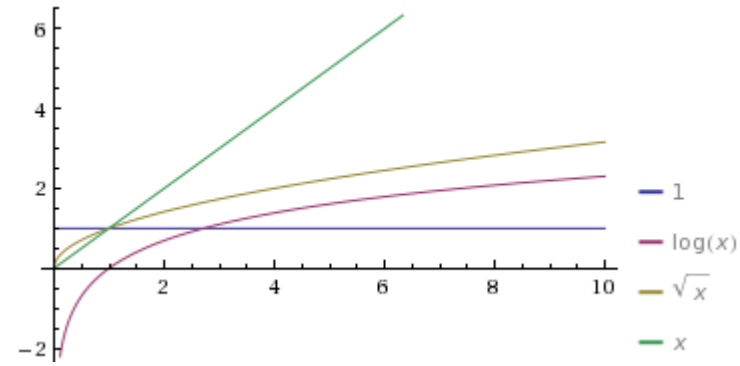



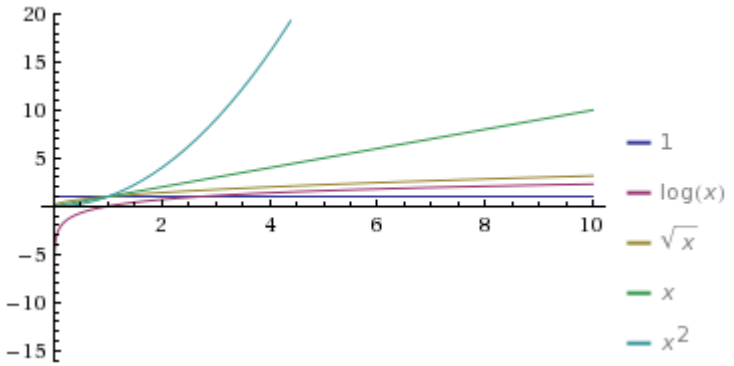



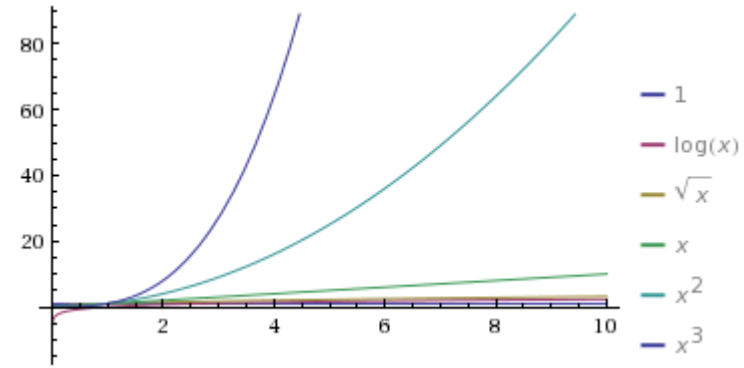
# Einige gängige Funktionen

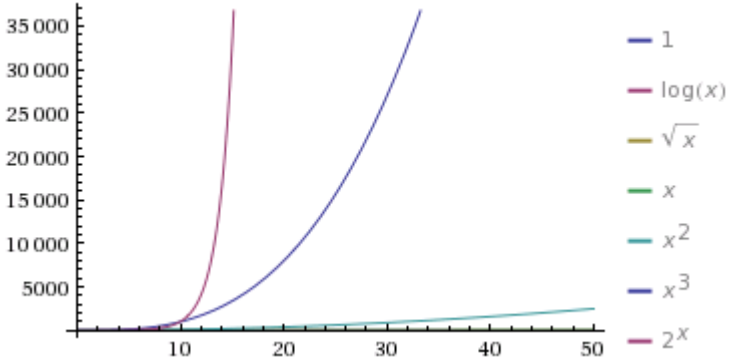


# Die Landau-Notation

- Seien $f$ und $g$ Funktionen von $\mathbb{R}_{\geq 0}$ nach $\mathbb{R}_{\geq 0}$
- Wir schreiben $f(n) = \mathrm{O}(g(n))$, falls es Zahlen $n_0 \geq 0$ und $c > 0$ gibt, so dass für alle $n \geq n_0$ gilt: $f(n) \leq c \cdot g(n)$
- Bedeutung: „$f$ wächst nicht wesentlich schneller als $g$.“
- Formal bezeichnet $\mathrm{O}\big(g(n)\big)$ die Klasse aller Funktionen $f$, die obige Bedingungen erfüllen
  - $\mathrm{O}(g) = \left\{ f: \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \middle| \begin{matrix} \exists c \in \mathbb{R} > 0 \quad \exists n_0 \in \mathbb{R} \geq 0 \\ \forall n \geq n_0 : \quad f(n) \leq c \cdot g(n) \end{matrix} \right\}$
- Gleichung $f(n) = \mathrm{O}\big(g(n)\big)$ drückt in Wahrheit eine Element-Beziehung $f \in \mathrm{O}(g(n))$ aus
- O-Terme können auch auf der linken Seite vorkommen; in diesem Fall wird eine Inklusionsbeziehung ausgedrückt
  - $n^2 + \mathrm{O}(n) = \mathrm{O}(n^2)$ steht für die Aussage $\{n^2 + f | f \in \mathrm{O}(n)\} \subseteq \mathrm{O}(n^2)$

# Die Landau-Notation (2)

- Definitionen:
  - $O(g) = \left\{ f: \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \;\middle|\; \begin{array}{l} \exists c \in \mathbb{R} > 0 \quad \exists n_0 \in \mathbb{R} \geq 0 \\ \forall n \geq n_0 \colon \; f(n) \leq c \cdot g(n) \end{array} \right\}$
  - $\Omega(g) = \left\{ f: \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \;\middle|\; \begin{array}{l} \exists c \in \mathbb{R} > 0 \quad \exists n_0 \in \mathbb{R} \geq 0 \\ \forall n \geq n_0 \colon \; f(n) \geq c \cdot g(n) \end{array} \right\}$
  - $\Theta(g) = \left\{ f: \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \;\middle|\; \begin{array}{l} \exists c_1, c_2 \in \mathbb{R} > 0 \quad \exists n_0 \in \mathbb{R} > 0 \\ \forall n \geq n_0 \colon \; c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \end{array} \right\}$
  - $o(g) = \left\{ f: \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \;\middle|\; \begin{array}{l} \forall c \in \mathbb{R} > 0 \quad \exists n_0 \in \mathbb{R} > 0 \\ \forall n \geq n_0 \colon \; f(n) < c \cdot g(n) \end{array} \right\}$
  - $\omega(g) = \left\{ f: \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \;\middle|\; \begin{array}{l} \forall c \in \mathbb{R} > 0 \quad \exists n_0 \in \mathbb{R} > 0 \\ \forall n \geq n_0 \colon \; f(n) > c \cdot g(n) \end{array} \right\}$
- Bedeutung: „$f$ wächst ...
  - nicht wesentlich schneller als $g$“
  - nicht wesentlich langsamer als $g$“
  - ungefähr genauso schnell wie $g$“
  - langsamer als $g$“
  - schneller als $g$“

# Beispiel 1

- Definitionen (kurz):
  - $f(n) = O(g(n)) \Leftrightarrow \exists c, n_0 \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
  - $f(n) = \Omega(g(n)) \Leftrightarrow \exists c, n_0 \forall n \geq n_0 : f(n) \geq c \cdot g(n)$
  - $f(n) = \Theta(g(n)) \Leftrightarrow \exists c_1, c_2, n_0 \forall n \geq n_0 : c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
- 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 $k \leq k \cdot 1$

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

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

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

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

# Beispiel 2

- Definitionen (kurz):
  - $f(n) = O(g(n)) \Leftrightarrow \exists c, n_0 \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
  - $f(n) = \Omega(g(n)) \Leftrightarrow \exists c, n_0 \forall n \geq n_0 : f(n) \geq c \cdot g(n)$
  - $f(n) = \Theta(g(n)) \Leftrightarrow \exists c_1, c_2, n_0 \forall n \geq n_0 : c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
- 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 $n \geq 1$

  $\Rightarrow f(n) = O(g(n)) = O(n^5)$
- 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 $n \geq 1$

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

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

# Beispiel 2 (2)

- Definitionen (kurz):
  - $f(n) = O(g(n)) \Leftrightarrow \exists c, n_0 \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
  - $f(n) = \Omega(g(n)) \Leftrightarrow \exists c, n_0 \forall n \geq n_0 : f(n) \geq c \cdot g(n)$
  - $f(n) = \Theta(g(n)) \Leftrightarrow \exists c_1, c_2, n_0 \forall n \geq n_0 : c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
  - $f(n) = o(g(n)) \Leftrightarrow \forall c \exists n_0 \forall n \geq n_0 : f(n) < c \cdot g(n)$
  - $f(n) = \omega(g(n)) \Leftrightarrow \forall c \exists n_0 \forall n \geq n_0 : f(n) > c \cdot g(n)$
- 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 $n \geq 1$

  $\Rightarrow f(n) \notin o(g(n)) = o(n^5)$
- 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 $n \geq 1$

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

# Zusammenhänge zwischen $O$, $\Omega$, $\Theta$, $o$ und $\omega$

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

- Definitionen (kurz):
  - $f(n) = O(g(n)) \Leftrightarrow \exists c, n_0 \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
  - $f(n) = o(g(n)) \Leftrightarrow \forall c \exists n_0 \forall n \geq n_0 : f(n) < c \cdot g(n)$
- Satz: $\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty \Rightarrow f \in O(g)$
  - Funktion im Nenner wächst mindestens so schnell wie im Zähler
- Satz: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in o(g)$
  - Funktion im Nenner wächst schneller als im Zähler
- 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 $\infty$ 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. Vorstellung des ersten Übungsblatts
3. Die Landau-Notation
4. Pseudocode-Analyse und Algorithmenentwurf

# Vollständige Induktion

- Beweismethode für den Beweis einer Aussage $A(n)$ über natürliche Zahlen, d.h. $n \in \mathbb{N}$
- Induktionsanfang (IA): Beweise, dass $A(0)$ wahr ist
- Induktionsschritt (IS): Beweise, dass wenn $A(k)$ wahr ist (IV), auch $A(k+1)$ (IB) wahr sein muss.
  - Induktionsvoraussetzung (IV): Die Aussage $A(k)$ ist wahr für ein bestimmtes $k \in \mathbb{N}$
  - Induktionsbehauptung (IB): Die Aussage $A(k+1)$ ist wahr (unter Verwendung der IV)
- Induktionsschluss: Gültigkeit der Aussage $A(n)$ für alle $n \in \mathbb{N}$ folgt aus IA und IS
- vgl. Dominoeffekt
- Wichtiges Werkzeug für Korrektheitsbeweise rekursiver Algorithmen

# Übungsaufgabe

Gegeben Sei der folgende Algorithmus:

**Algorithmus** $M(i)$

**Input:** $i \in \mathbb{N}$
**Output:** Zahl $x$

(1) **if** $i = 0$ **then**
(2) return $i$;
(3) **else**
(4) return $i \cdot M(i-1)$;
(5) **end if**

vgl. Blatt 1, Aufgabe 3-4

1. Was berechnet der Algorithmus $M(i)$? Geben Sie hierzu eine Formel an.
2. Analysieren Sie die Laufzeit des Algorithmus.
3. Beweisen Sie die Korrektheit des Algorithmus.

# Korrektheitsbeweis

Behauptung: $M(i) = 0$

Beweis durch vollständige Induktion über i:

IA (i $= 0$): $M(i) \overset{(2)}{=} i = 0$

IS ($i \to i + 1$):

IV: $M(i) = 0$

IB: $M(i + 1) = 0$

$$M(i+1) \overset{(4)}{=} (i+1) \cdot M\big((i+1) - 1\big)$$
$$= (i+1) \cdot M(i) \overset{\text{IV}}{=} (i+1) \cdot 0 = 0$$ ■

# Ausblick: Nächste Woche

- Klärung von Fragen zum ersten Übungsblatt
- Fortsetzung der Vertiefung der Landau-Notation
- Vorstellung des zweiten Übungsblatts (Stacks & Queues)
- Vorbereitende Aufgaben für das zweite Übungsblatt
- Fragen?