Übung Algorithmen und Datenstrukturen

Sommersemester 2016

Patrick Schäfer, 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

Agenda

1. Heute:
    1. Organisatorisches
    2. Die Landau-Notation
    3. Laufzeitanalyse
    4. Besprechung des ersten Übungsblatts

- Eure Vorbereitung zu Hause:
    - Lest das O-Tutorial auf der Webseite der Übung
    - Wiederholt: Grenzwerte, Ableitungen, Potenz- und Logarithmusgesetze

Organisatorisches

- In Goya einschreiben (nicht in Warteliste, Zu Not in eine beliebige Übungsgruppe einschreiben).
- Gruppe bei Goya eintragen (Gruppenname möglichst einfach - den Korrektoren zuliebe)
- E-Mails bei Goya regelmäßig lesen oder E-Mail-Weiterleitung in Goya aktivieren.
- Übungsaufgaben: 6 Blätter / 50 Punkte pro Blatt.
- 50% der Punkte notwendig zum Bestehen.
- Abgabe in 2er Gruppen, müssen nicht in einer Übungsgruppe sein, keine Punkte auf's Übungsblatt bei Einzelabgaben.
- Lösungen getrennt nach Aufgaben abgeben, Blätter einer Aufgabe zusammentackern.
- Namen, Matrikelnummern, Übungsgruppe für Rückgabe (eindeutig!!!) auf jeden Zettel schreiben.
- Abgabe bis 5min vor Vorlesungsbeginn, Briefkasten bei Raum 3.321, Rud25.
- Programmieraufgaben auf gruenau2 testen.
- Bei Feiertagen oder wenn der Besuch der Übung in der man eingetragen ist, nicht möglich ist, andere Übung in derselben Woche besuchen.
- 2 Wochenrhythmus: Üben für ein Blatt vs Lösungen der Aufgaben eines Blattes besprechen.

O-Notation

- Ziel: Abschätzung der Laufzeit eines Algorithmus
  - Wird definiert als Funktion der Eingabe.
  - Gesucht wird (üblicherweise) die Laufzeit im schlechtesten Fall (Worst Case). Das ist die Laufzeit der ungünstigsten Eingabe.
  - Die Abschätzung ist unabhängig von der Hardware und der Implementierung.

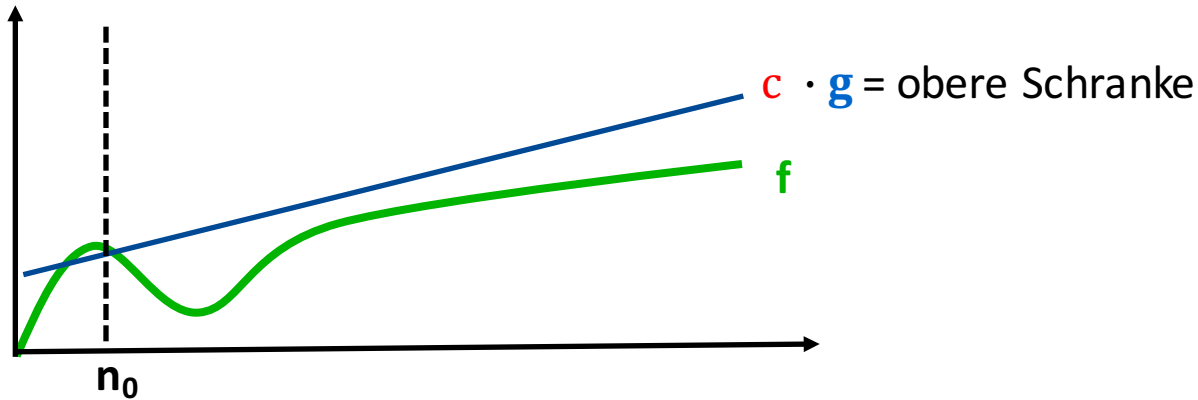
- Grundidee: Welche Faktoren bestimmen die Laufzeit des Algorithmus, wenn die Eingabe groß wird.
  - Betrachten asymptotisches Wachstum der Laufzeit für $n \to \infty$
  - Nur der größte Faktor (mit dem größten Exponent) ist relevant.
    - $f(n) = 4n^4 + 5n^2 + 10 = O(n^4)$
    - D.h., n^4 dominiert für $n \to \infty$

- So erhalten wir obere oder untere Schranken.

O-Notation: Menge von Funktionen

$$\underbrace{O(g)}_{a_1} = \{\underbrace{f: \mathbb{R}_0^+ \to \mathbb{R}_0^+}_{a_2} | \underbrace{\exists c > 0}_{a_3} \underbrace{\exists n_0 > 0}_{a_4} \underbrace{\forall n \geq n_0}_{a_5}: \underbrace{f(n) \leq c \cdot g(n)}_{a_6}\}$$

- a_1: Wir definieren eine Menge $O(g)$,
- a_2: Sie besteht aus der Menge aller Funktionen f,
- a_3 und a_4: Es existieren zwei Konstanten c und n_0,
- a_5: Die nachfolgende Aussage gilt ab einer Konstanten n_0,
- a_6: $c \cdot g(n)$ ist obere Schranke für $f(n)$

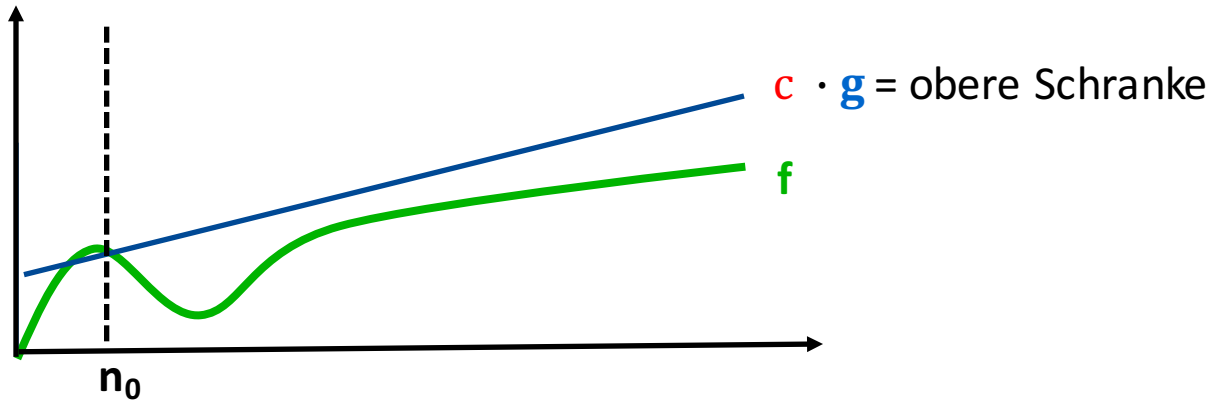


Groß O: Obere Schranke

- **Definition:**

$$O(g) = \{f: \mathbb{R}_0^+ \to \mathbb{R}_0^+ |\ \exists c > 0 \quad \exists n_0 > 0 \quad \forall n \geq n_0 : \quad f(n) \leq c \cdot g(n)\}$$

- Wir schreiben $f(n) \in O(g)$, falls g eine *obere Schranke* von f ist, d.h., f *höchstens so schnell* wie g wächst.



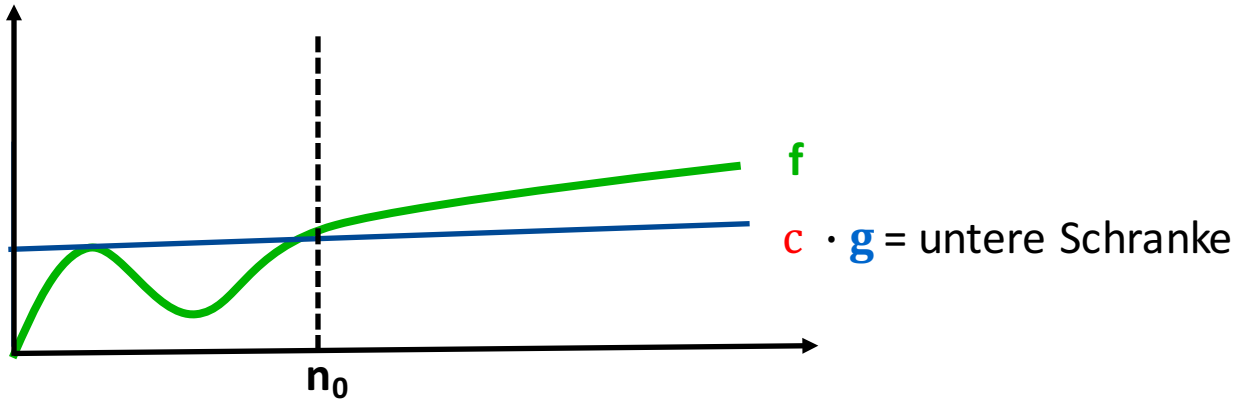
- Beispiele: $10 \cdot n^2 + n \in O(n^2)$, $10 \cdot n^2 \in O(n^3)$, $n^2 \notin O(n)$
- Scharfe obere Schranken angeben: $4n^2 - n \in O(n^2)$ sagt mehr als $4n^2 - n \in O(n^3)$.

Groß Omega: Untere Schranke

- **Definition:**

$$\Omega(g) = \{f: \mathbb{R}_0^+ \to \mathbb{R}_0^+ \mid \exists c > 0 \quad \exists n_0 > 0 \quad \forall n \geq n_0: \ f(n) \geq c \cdot g(n)\}$$

- Wir schreiben $f(n) \in \Omega(g)$, falls g eine untere Schranke von f ist, d.h., f *mindestens so schnell* wächst wie g.



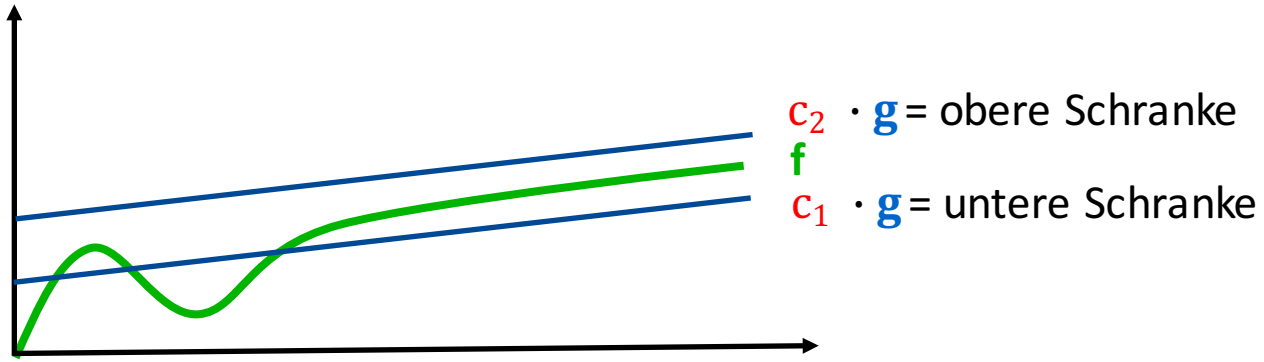
- Beispiele: $n^2 \in \Omega(\sqrt{n})$, $n^2 \in \Omega(n^2)$, $n^2 \notin \Omega(n^3)$
- Scharfe untere Schranken angeben: $4n^2 - n \in \Omega(n^2)$ sagt mehr als $4n^2 - n \in \Omega(n)$.

Theta: Asymptotisch enge Schranke

- **Definition:**

$$\Theta(g) = \{f \mid \exists c_1, c_2 > 0 \quad \exists n_0 > 0 \quad \forall n \geq n_0 \; : \; c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$$

- Wir schreiben $f(n) \in \Theta(g)$, falls g untere und obere Schranke von f ist, d.h., f wächst *ebenso schnell* wie g.



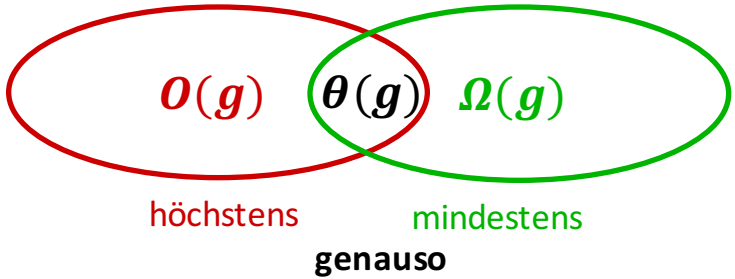
- Beispiel: $n^2 + n \in \Theta(n^2)$

Die Landau-Notation

- $O(g) = \{f \mid \exists c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 : \;\; f(n) \leq c \cdot g(n)\}$
- höchstens so schnell wie…

- $\Omega(g) = \{f \mid \exists c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 : \;\; f(n) \geq c \cdot g(n)\}$
- mindestens so schnell wie…

- $\boldsymbol{\Theta(g)} = \left\{ f \;\middle|\; \begin{array}{c} \exists c_1, c_2 > 0 \;\; \exists n_0 > 0 \\ \forall n \geq n_0 : \;\; c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \end{array} \right\}$
- genauso schnell wie …



- $o(g) = \{f \mid \forall c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 : \;\; f(n) < c \cdot g(n)\}$
- wesentlich langsamer als …

- $\omega(g) = \{f \mid \forall c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 : \;\; f(n) > c \cdot g(n)\}$
- wesentlich schneller als …

Wichtige Komplexitätsklassen

- $O(1)$: konstant (Array Zugriff)
- $O(\log n)$: logarithmisch (Binäre Suche)
- $O(n)$: linear (Sequentielle Suche)
- $O(n \log n)$: linear logarithmisch (Mergesort)
- $O(n^2)$: quadratisch (Bubble Sort)
- $O(n^k)$: polynomial
- $O(2^n)$: exponentiell (Traveling Salesman)

- Komplexität bis zu $O(n^2)$ ist akzeptabel.

Beispiel 1

- Funktionen
  - $f(n) = k$ mit $k > 0$
  - $g(n) = 1$
- Gilt $k \in \Theta(1)$?

- Zu zeigen $k \in O(1) \Leftrightarrow \exists c: \; k \leq c \cdot 1$
  $k \leq c \cdot 1$
  mit c ≥k $\quad \Rightarrow f(n) \in O(g(n))$

- Zu zeigen $k \in \Omega(1) \Leftrightarrow \exists c: \; k \geq c \cdot 1$
  $k \geq c \cdot 1$
  mit c≤k $\quad \Rightarrow f(n) \in \Omega(g(n))$

- Zu zeigen $k \in \Theta(1) \Leftrightarrow \exists c_1, c_2, : c_1 \cdot 1 \leq k \leq c_2 \cdot 1$
  $c_1 \cdot 1 \leq k \leq c_2 \cdot 1$
  mit $c_1 \leq k$, $c_2 \geq k$ $\quad \Rightarrow f(n) \in \Theta(g(n))$

- Allgemein: Konstante Funktionen wachsen asymptotisch gleich schnell.

Beispiel 2

- Funktionen
  - $f(n) = 3n^5 + 4n^3 + 15$
  - $g(n) = n^5$
- Gilt $f(n) \in \Theta(g(n))$?

- Zu zeigen $f(n) \in O(g(n)) \Leftrightarrow \exists c, n_0 > 0 \ \forall n \geq n_0 : f(n) \leq c \cdot g(n)$:

  $3n^5 + 4n^3 + 15 \leq c \cdot n^5$ | mit $3n^5 + 4n^3 + 15 \leq 3n^5 + 4n^5 + 15n^5$

  $\Rightarrow 3n^5 + 4n^5 + 15n^5 \leq c \cdot n^5$

  $\Leftrightarrow (3+4+15)\, n^5 \leq c \cdot n^5$

  mit $c \geq 22$ *und* $n_0 = 1$ $\Rightarrow f(n) = O(g(n))$

- Zu zeigen $f(n) \in \Omega(g(n)) \Leftrightarrow \exists c, n_0 > 0 \ \forall n \geq n_0 : f(n) \geq c \cdot g(n)$:

  $3n^5 + 4n^3 + 15 \geq c \cdot n^5$ | mit $3n^5 + 4n^3 + 15 \geq 3n^5$

  $\Rightarrow 3n^5 \geq c \cdot n^5$

  mit $c \leq 3$ *und* $n_0 = 1$ $\Rightarrow f(n) = \Omega(g(n))$

$\Rightarrow f(n) \in \Theta(g(n))$

- Allgemein gilt: $a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0 = \Theta(n^k)$ für $a_k > 0$

Grenzwert als hinreichendes Kriterium

- Satz: $\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty \Rightarrow f \in O(g) \qquad \Leftrightarrow g \in \Omega(f)$
- D.h. Funktion im Nenner wächst *mindestens so schnell* wie im Zähler

- Satz: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in o(g) \qquad \Leftrightarrow g \in \omega(f)$
- D.h. Funktion im Nenner wächst *schneller* als im Zähler

Beispiel 3 mit hinreichendem Kriterium

- Beispiel:
  - $f(n) = 3n^5 + 4n^3 + 15$
  - $g(n) = n^5$
- Gilt $f(n) \in \Theta(g(n))$?

1. Z.z.: $f(n) \in O(g(n))$?
   - Wir betrachten den Grenzwert:

$$\lim_{n\to\infty} \frac{3n^5+4n^3+15}{n^5} = \lim_{n\to\infty} \left(\frac{3n^5}{n^5} + \frac{4n^3}{n^5} + \frac{15}{n^5}\right) = 3 + 0 + 0 = 3$$

$$\Rightarrow f(n) \in O(g(n))$$

2. Z.z.: $f(n) \in \Omega(g(n))$?
   - Wir betrachten den Grenzwert:

$$\lim_{n\to\infty} \frac{n^5}{3n^5+4n^3+15} = \frac{\infty}{\infty} = ???$$

Beispiel 4 mit hinreichendem Kriterium

- Beispiel:
  - $f(n) = \log n \ (= \log_2 n)$
  - $g(n) = \sqrt{n}$

- Gilt $f(n) \in O(g(n))$?
- Wir betrachten den Grenzwert:

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

Satz von L‘Hôpital

- Satz von L’Hôpital:
  Seien f und g zwei differenzierbare Funktionen, deren Grenzwerte entweder beide gegen 0 oder beide gegen ∞ gehen. Dann gilt :

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \overset{L'H}{=} \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

(falls der Grenzwert existiert).

Beispiel 4 mit L'Hôpital

- Beispiel:
  - $f(n) = \log n \ (= \log_2 n)$
  - $g(n) = \sqrt{n}$
- Gilt $f(n) \in O(g(n))$?
- Wir betrachten den Grenzwert:

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

$$= \lim_{n\to\infty} \frac{\ln n}{\ln 2 \cdot \sqrt{n}}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{(\ln n)'}{(\ln 2 \cdot \sqrt{n})'}$$

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

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

Beispiel 3 mit L'Hôpital

- Beispiel:
  - $f(n) = 3n^5 + 4n^3 + 15$
  - $g(n) = n^5$
- Gilt $f(n) \in \Omega(g(n))$?
- Wir betrachten den Grenzwert:

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{n^5}{3n^5 + 4n^3 + 15}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{(n^5)'}{(3n^5+4n^3+15)'} = \lim_{n\to\infty} \frac{5n^4}{5\cdot 3n^4+3\cdot 4n^2}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{4 \cdot 5n^3}{5 \cdot 3n^4 + 2 \cdot 3 \cdot 4n}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{4 \cdot 5n^3}{4 \cdot 5 \cdot 3n^3 + 2 \cdot 3 \cdot 4 \cdot 1}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{3\cdot 4\cdot 5n^2}{3\cdot 4\cdot 5\cdot 3n^2} = \lim_{n\to\infty} \frac{1}{3} = \frac{1}{3}$$

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

Laufzeitanalyse

Algorithmus $M(n)$

Input: n∈ $\mathbb{N}$
Output: Zahl x

```
(1)  if n= 0 then
(2)     return n;
(3)  else
(4)     return n · M(n − 1);
(5)  end if
```

a) Was berechnet M?
b) Analysiere die Laufzeit.
c) Entwerfe einen effizienten Algorithmus.
d) Induktionsbeweis

Laufzeitanalyse

- Elementare Operationen haben eine konstante Laufzeit:

```
x:=0;
```

O(1)

- Alternativ

```
x=x+i;
```

O(1)

- Bedingte Anweisung haben eine konstante Laufzeit:

```
if (x<0)
    x=x*-1;
```

O(1)
O(1) ==O(1)+O(1)

Laufzeitanalyse

- Schleifen:

```
for i:=1 to 10 do
    x = x + i;
end for
```

```
10
 * O(1)          = 10 * O(1)
                 = O(1)
```

- Oder gegeben Eingabe n

```
for i:=1 to n do
    x = x + i;
end for
```

```
O(n)
 * O(1)          = O(n) * O(1)
```

- Verschachtelte Schleifen

```
for i:=1 to n do
    for j:=1 to n do
        x = x + i*j;
    end for
end for
```

```
O(n)
 * O(n)          = O(n)*O(n)
   * O(1)        = O(n)*O(n)*O(1)=O(n^2)
```

Laufzeitanalyse

Algorithmus $M(n)$

Input: n∈ $\mathbb{N}$
Output: Zahl x

```
(1)  if n= 0 then
(2)      return n;
(3)  else
(4)      return n · M(n − 1);
(5)  end if
```

$$
\begin{aligned}
&M(n)\\
&= M(n-1) \cdot n\\
&= M(n-2) \cdot n \cdot (n-1)\\
&= M(n-3) \cdot n \cdot (n-1) \cdot (n-2)\\
&\ldots\\
&= M(0) \cdot n \cdot (n-1) \cdot (n-2) \ldots \cdot (1)\\
&= 0
\end{aligned}
$$

- $M(n) \in O(n)$

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
- Induktionsschluss: Gültigkeit der Aussage $A(n)$ für alle $n \in \mathbb{N}$ folgt aus IA und IS
- Wichtiges Werkzeug für Korrektheitsbeweise rekursiver Algorithmen

Induktionsbeweis

Algorithmus $M(i)$

Input: n∈ $\mathbb{N}$
Output: Zahl x

```
(1)  if n= 0 then
(2)      return n;
(3)  else
(4)      return n · M(n − 1);
(5)  end if
```

- Behauptung: $\prod_{i=0}^{n} i = M(n) = 0$
- Induktionsanfang: $M(0) = 0$
- Induktionsvoraussetzung: $\prod_{i=0}^{n} i = M(n) = 0$
- Induktionsschritt: $n \to n + 1$

$$\prod_{i=0}^{n+1} i = (n+1) \cdot \prod_{i=0}^{n} i \overset{(I.V.)}{=} (n+1) \cdot M(n) \overset{(def.)}{=} M(n+1) = 0$$

Agenda

1. Organisatorisches
2. Die Landau-Notation
3. Laufzeitanalyse
4. Besprechung des ersten Übungsblatts