

WS 2016/17

# Diskrete Strukturen

## Kapitel 2: Grundlagen (Wachstum)

Hans-Joachim Bungartz

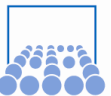
Lehrstuhl für wissenschaftliches Rechnen

Fakultät für Informatik

Technische Universität München

[http://www5.in.tum.de/wiki/index.php/Diskrete\\_Strukturen\\_-\\_Winter\\_16](http://www5.in.tum.de/wiki/index.php/Diskrete_Strukturen_-_Winter_16)

- Mathematische und notationelle Grundlagen
  - Mengen
  - Relationen und Abbildungen
  - Aussagen- und Prädikatenlogik
  - Beweismethoden
  - Wachstum von Funktionen



# Kapitel 2: Grundlagen (Wachstum) TUM

- Programme brauchen mehr Ressourcen (Zeit, Speicher, ...), je größer ihre Eingabe.
- Die Laufzeit (Speicherverbrauch) wird durch eine Funktion  $f(x)$  dargestellt:  
 $f(x)$  = maximale Laufzeit über alle Eingaben der Größe  $x$ .
- Wir sind an einer **Abschätzung des Wachstums** von  $f(x)$  interessiert:
  - Wenn  $f(x)$  schneller wächst als  $g(x)$ , dann wird  $f(x)$  immer irgendwann größer als  $g(x)$  werden (für **ausreichend große** Werte von  $x$ ).



- Beispiel:

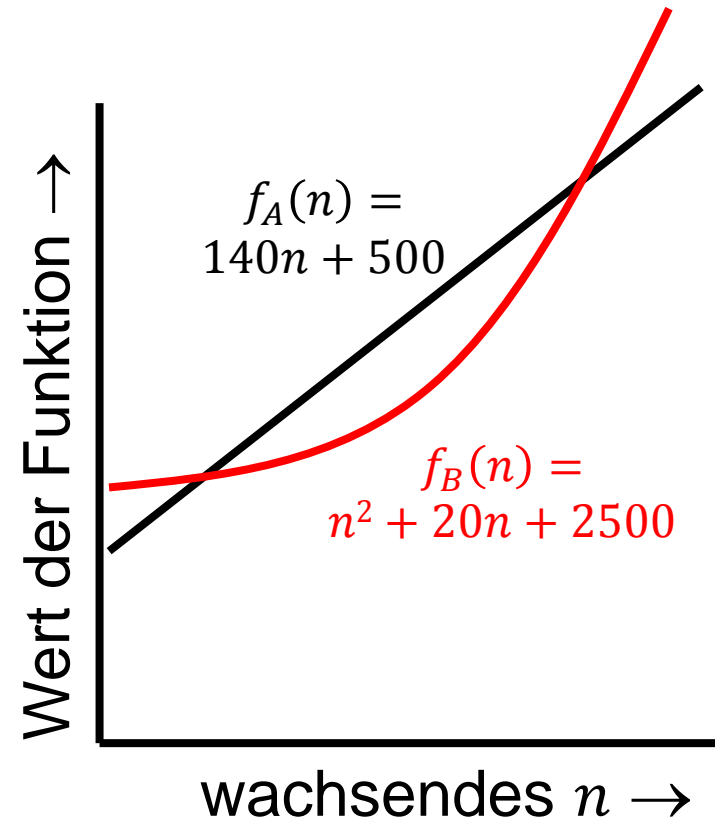
- Eine Web-Seite soll entwickelt werden, die Benutzerdaten bearbeitet.
- Programm  $A$  benötigt  $f_A(n) = 140n + 500$  Mikrosekunden, um  $n$  Einträge zu bearbeiten.
- Programm  $B$  benötigt  $f_B(n) = n^2 + 20n + 2500$  Mikrosekunden für  $n$  Einträge.

Welches Programm soll verwendet werden?



- Eine kleine Berechnung zeigt:

Für  $20 \leq n \leq 100$  ist  
 $n^2 + 20n + 2500 \leq 140n + 500$



- Die multiplikativen Konstanten 140, 1, 20 und die additiven Konstanten 500, 2500 sind meistens schwer zu bestimmen (und abhängig von Einzelheiten der Implementierung).
- Oft ist nur bekannt:
  - Programm  $A$  braucht  $an + b$  Mikrosekunden,
  - Programm  $B$  braucht  $cn^2 + dn + e$  Mikrosekundenmit nicht allzu großen Konstanten  $a, b, c, d, e$ .
- Wichtig ist nun: Unabhängig von den Werten von  $a, \dots, e$  gibt es **immer** eine Zahl  $n_0$ , sodass ab  $n_0$  Daten das Programm  $A$  schneller ist als  $B$ .



# Kapitel 2: Grundlagen (Wachstum) TUM

- Wir sagen, dass  $f_A(n)$  langsamer als  $f_B(n)$  wächst, falls ab einem gewissen Punkt  $f_A(n)$  stets unter  $f_B(n)$  liegt.
- Wir führen die **Groß- $O$ -Notation** ein, um diesen Sachverhalt zu formalisieren. Diese wurde von **Paul Bachmann** (1837–1920) entwickelt, von **Edmund Landau** (1877–1938) verbreitet, und von **D. E. Knuth** in der Algorithmenanalyse eingeführt.
- Das „ $O$ “ wird auch Landau Symbol genannt.
- Informell ist  $O(f)$  die Menge der Funktionen, die langsamer oder so schnell wie  $f$  wachsen.



- Definition (Groß- $O$ -Notation):  
Es sei  $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x > 0\}$ .

$f(n) \in O(g(n))$  gilt genau dann, wenn  $\exists c \in \mathbb{R}^+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 :$   
 $|f(n)| \leq c \cdot g(n).$

„ $f$  wächst (bis auf einen konstanten Faktor)  
höchstens so schnell wie  $g$ “





# Kapitel 2: Grundlagen (Wachstum) $\Sigma\Pi$

- Warum  $|f(n)| \leq c \cdot g(n)$  statt  $|f(n)| \leq g(n)$ ?
- Nehmen wir an, dass
  - $f(n) = an + b$  und
  - $g(n) = cn + d$für Konstanten  $a, b, c, d$ .
- Wir wollen  $f$  und  $g$  als Funktionen betrachten, die genau so schnell wachsen.
- Mit der Definition auf der vorigen Folie gilt  $f \in O(g)$  und  $g \in O(f)$ .

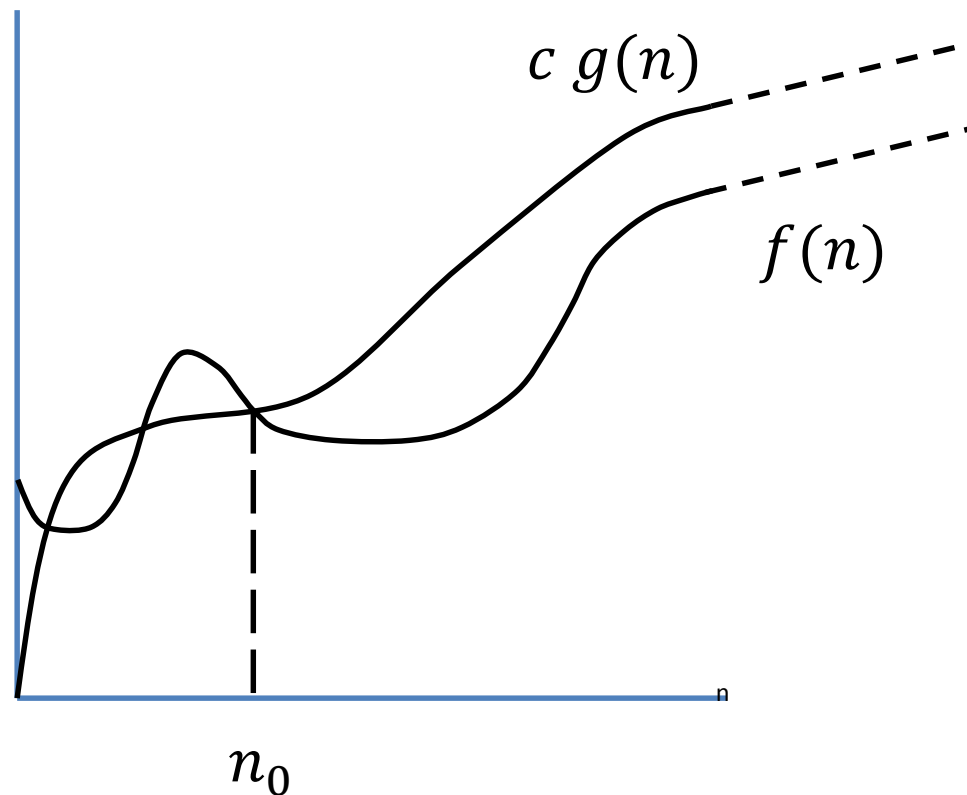


- Beachte:
  - $O(g)$  ist eine Menge von Funktionen.
  - Statt  $f \in O(g)$  sagt man auch: „ $f$  ist höchstens von der Ordnung  $g$ “ oder „ $f$  ist  $O(g)$ “ oder (unsauber) „ $f = O(g)$ “.
  - Meistens werden nur Funktionen  $\mathbb{N}_0 \rightarrow \mathbb{R}^+$  betrachtet (oder  $\mathbb{N} \rightarrow \mathbb{N}_0$ ), dann ist der Absolutbetrag überflüssig.
  - Manchmal werden auch Funktionen von  $\mathbb{R} \rightarrow \mathbb{R}$  betrachtet.



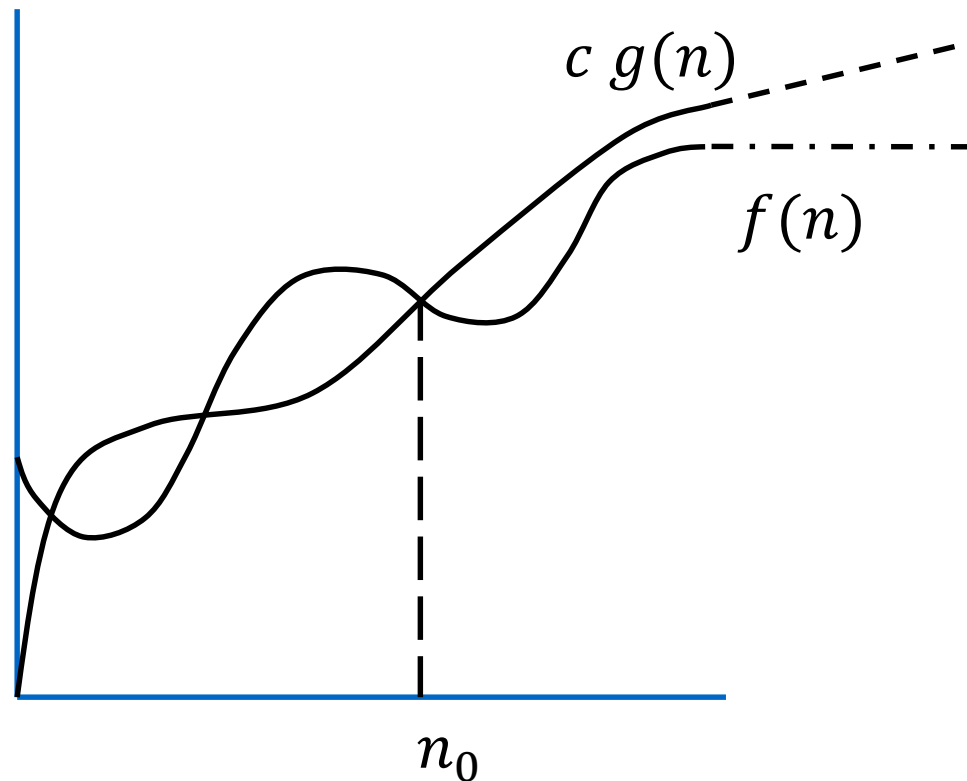
# Kapitel 2: Grundlagen (Wachstum) $\tau\mu\pi$

- Veranschaulichung der Groß- $O$ -Notation:



# Kapitel 2: Grundlagen (Wachstum) $\tau\mu\pi$

- Veranschaulichung der Groß- $O$ -Notation:



- Definition (Klein- $o$ -Notation):

$f(n) \in o(g(n))$  genau dann, wenn  $\forall c \in \mathbb{R}^+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 :$   
 $|f(n)| < c \cdot g(n).$

„ $f$  wächst echt langsamer als  $g$ .“



- Definition (Groß-Omega-Notation):

$f(n) \in \Omega(g(n))$  genau dann, wenn  $\exists c \in \mathbb{R}^+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 :$   
 $|f(n)| \geq c \cdot g(n) \geq 0.$

„ $g$  wächst (bis auf einen konstanten Faktor)  
höchstens so schnell wie  $f$ .“



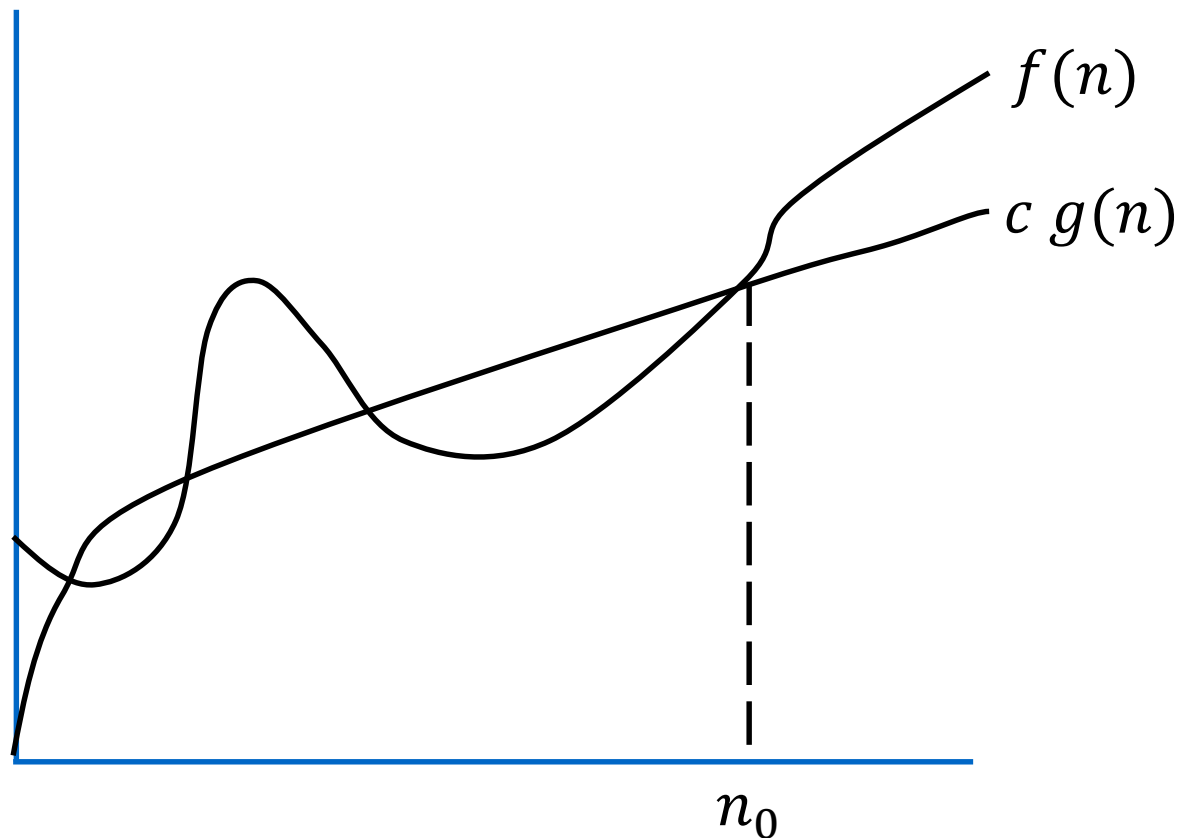
- Definition (Klein-Omega-Notation):

$f(n) \in \omega(g(n))$  genau dann, wenn  $\forall c \in \mathbb{R}^+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 :$   
 $|f(n)| > c \cdot g(n) \geq 0.$

„ $f$  wächst echt schneller als  $g$ .“



- Veranschaulichung der Klein-Omega-Notation:





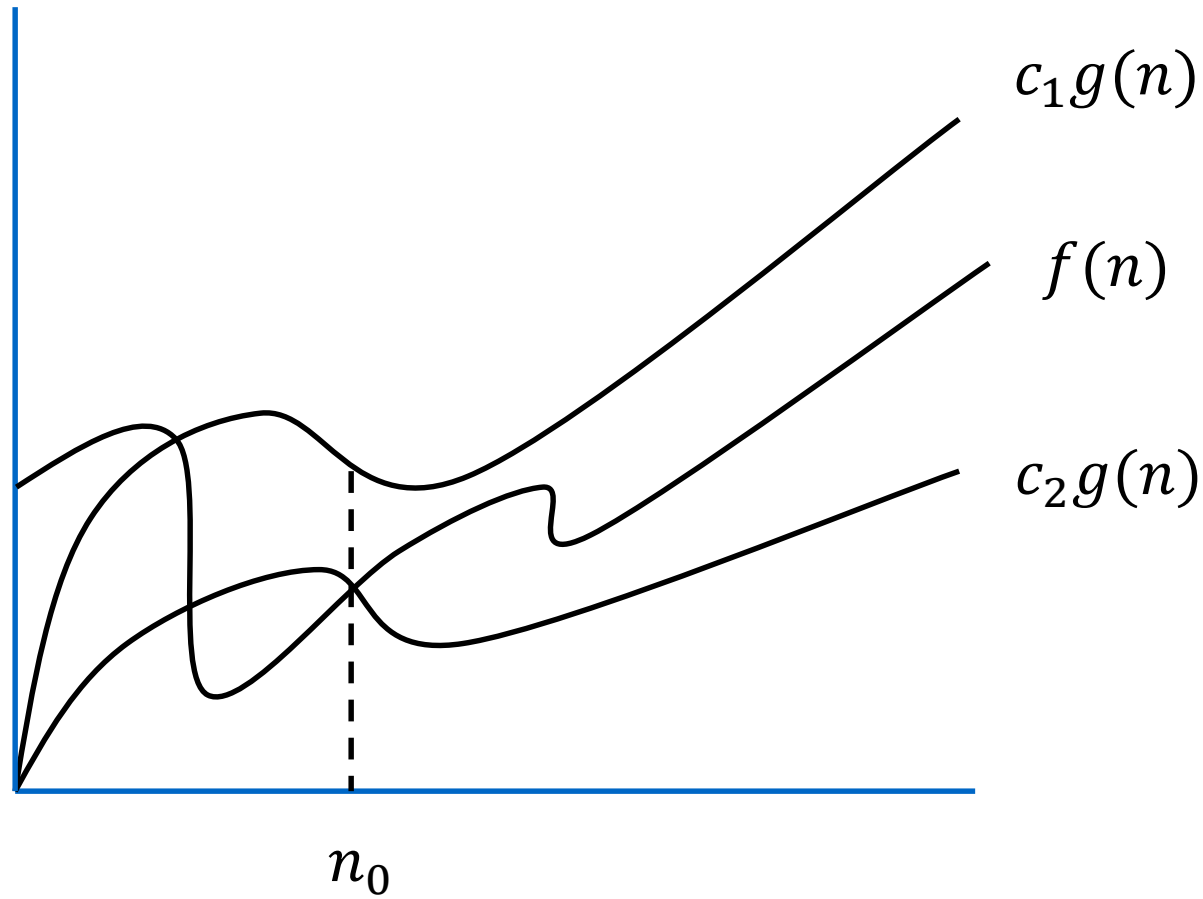
- Definition (Groß-Theta-Notation):

$f(n) \in \Theta(g(n))$  genau dann, wenn  
 $f(n) \in O(g(n))$  und  $g(n) \in O(|f(n)|)$ .

„ $f$  wächst bis auf konstante Faktoren genau so schnell wie  $g$ .“



- Veranschaulichung der Groß- $\Theta$ -Notation:



# Kapitel 2: Grundlagen (Wachstum) $\Sigma\Pi$

- Beispiel:

Zeige, dass  $f(n) = 30n + 8 \in O(n)$ .

Wir müssen also zeigen:

$$\exists c \in \mathbb{R}^+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : 30n + 8 \leq cn$$

Nehme:

$$c = 31, \quad n_0 = 8.$$

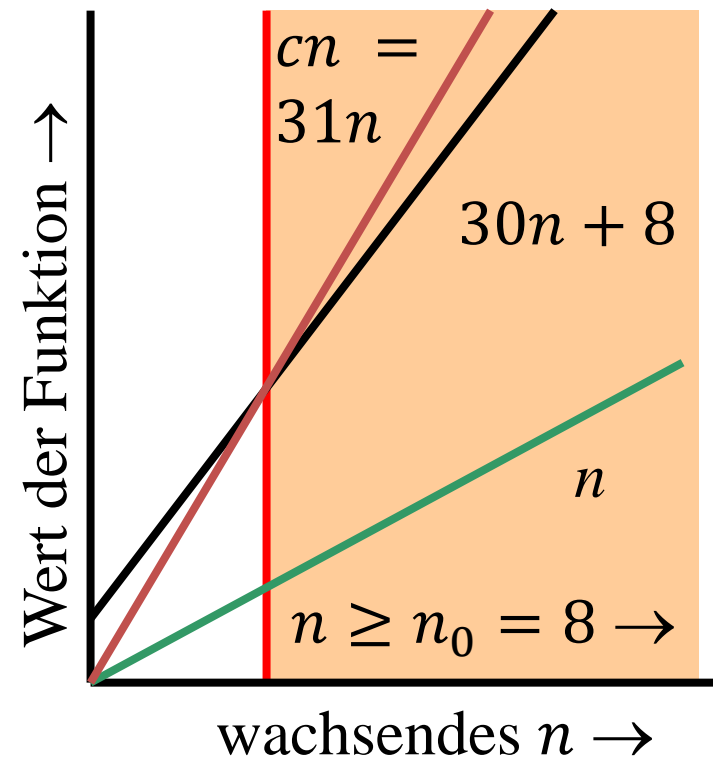
Daraus folgt für alle  $n \geq n_0$ :

$$cn = 31n = 30n + n \geq 30n + 8.$$



# Kapitel 2: Grundlagen (Wachstum) TUM

- Beachte, dass  $30n + 8$  nicht **überall** kleiner als  $31n$  ist.
- Aber es ist kleiner als  $31n$  für  $n \geq 8$ .



# Kapitel 2: Grundlagen (Wachstum) $\Sigma\Pi$

- Beispiel:

Zeige, dass  $f(n) = n^2 + 1 \in O(n^2)$ .

Wir müssen zeigen:

$$\exists c \in \mathbb{R}^+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : n^2 + 1 \leq cn^2.$$

Nehme:

$$c = 2, \quad n_0 = 1.$$

Daraus folgt für alle  $n \geq n_0$ :

$$cn^2 = 2n^2 = n^2 + n^2 \geq n^2 + 1.$$



Theorem:

Für Polynome  $f(x) = \sum_{i=0}^n a_i x^i$  gilt  
 $f(x) \in O(x^n)$ .

Beweis:

$$\begin{aligned}\forall x \geq 1: |f(x)| &= \left| \sum_{i=0}^n a_i x^i \right| \leq \sum_{i=0}^n |a_i| x^i \\ &= x^n \left( \sum_{i=0}^n |a_i| x^{i-n} \right) \leq x^n \sum_{i=0}^n |a_i| =: x^n c.\end{aligned}$$



- Eine Funktion ist  $O(g)$ , wenn Konstanten  $c$  und  $n_0$  existieren, die die Bedingung erfüllen.
- Aber, die speziellen Werte von  $c$  und  $n_0$ , die die Bedingung erfüllen, sind **nicht eindeutig**: Jeder Wert größer als  $c$  und/oder  $n_0$  erfüllt die Bedingung auch.
- In der Regel sind wir jedoch **nicht daran interessiert**, die **kleinsten** Werte von  $c$  und  $n_0$  zu finden.



- Bemerkungen zur O-Notation:
  - $O(\cdot)$ , als Relation gesehen, ist transitiv:  
 $f \in O(g) \wedge g \in O(h) \rightarrow f \in O(h)$ .
  - Wenn  $g \in O(f)$  und  $h \in O(f)$ , dann ist  
 $g + h \in O(f)$ .
  - Für alle  $c > 0$  und  $\lim_{n \rightarrow \infty} f(n) = \infty$ :  
 $O(cf) = O(f + c) = O(f - c) = O(f)$ .
  - Wenn  $f_1 \in O(g_1)$  und  $f_2 \in O(g_2)$ , dann  
 $f_1 \cdot f_2 \in O(g_1 \cdot g_2)$  und  
 $f_1 + f_2 \in O(g_1 + g_2) = O(\max(g_1, g_2))$   
 $= O(g_1)$ , wenn  $g_2 \in O(g_1)$ .





- Behauptung:

$$n! \in O(n^n).$$

- Beweis:

$$\begin{aligned}\forall n \in \mathbb{N}: n! &= n \cdot (n - 1) \cdot \dots \cdot 2 \cdot 1 \\ &\leq n \cdot n \cdot \dots \cdot n \\ &= n^n. \quad \square\end{aligned}$$



# Kapitel 2: Grundlagen (Wachstum) $\Sigma\Pi$

- Behauptung:

$$2^n \in o(2^{2n}).$$

- Beweis:

$$\text{Z. z: } \forall c \in \mathbb{R}^+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : 2^n < c \cdot 2^{2n}.$$

Sei  $c > 0$  beliebig. Wähle  $n_0 = \lceil 1 + \log\left(\frac{1}{c}\right) \rceil$ .

Sei nun  $n \geq n_0$  beliebig:

$$c 2^{2n} = c 2^n 2^n \geq c 2^{n_0} 2^n \geq c 2 \left(\frac{1}{c}\right) 2^n > 2^n.$$

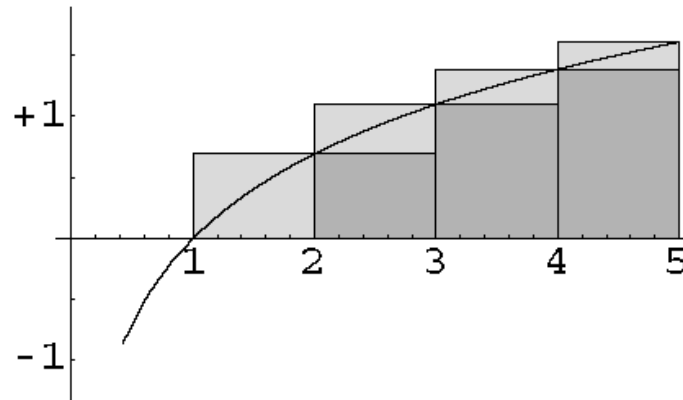


# Kapitel 2: Grundlagen (Wachstum) $\Sigma\Pi$

- Theorem:  $n! \in \Omega\left(\left(\frac{n}{e}\right)^n\right)$  und  $n! \in O\left(n \cdot \left(\frac{n}{e}\right)^n\right)$ .

Beweis:

$$\forall n > 0: \sum_{k=1}^{n-1} \ln k < \int_1^n \ln x \, dx < \sum_{k=2}^n \ln k < \int_1^{n+1} \ln x \, dx .$$

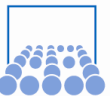


Beweis (Fortsetzung):

Es gilt:

$$\int_1^n \ln x \, dx = (x \ln x - x) \Big|_1^n = n \ln n - n + 1;$$

$$\int_1^{n+1} \ln x \, dx = (n + 1) \cdot \ln(n + 1) - n.$$



Beweis (Fortsetzung) :

Also:

$$\forall n \in \mathbb{N}: n \ln n - n + 1 < \ln n! < (n + 1) \ln(n + 1) - n$$

und damit

$$\frac{n^n}{e^{n-1}} \leq n! \leq \frac{(n+1)^{n+1}}{e^n}$$

oder

$$e \cdot \left(\frac{n}{e}\right)^n \leq n! \leq (n+1) \cdot \left(1 + \frac{1}{n}\right)^n \cdot \left(\frac{n}{e}\right)^n \leq (n+1) \cdot e \cdot \left(\frac{n}{e}\right)^n .$$



- Auch außerhalb der Analyse des Wachstumsverhaltens von Funktionen wird die  $O$ -Notation verwendet:
  - Z.B. zur Abschätzung von Termen, die nicht besonders wichtig sind.
  - Wenn „ $O(g)$ “ als Term in einem arithmetischen Ausdruck verwendet wird, bedeutet dies: „eine Funktion  $f$  aus der Menge  $O(g)$ “.
  - $f + O(g) := \{ f' \mid \exists f'' \in O(g): f' = f + f'' \}$ .



- Beispiel (Formel zur Annäherung für die Fakultät):

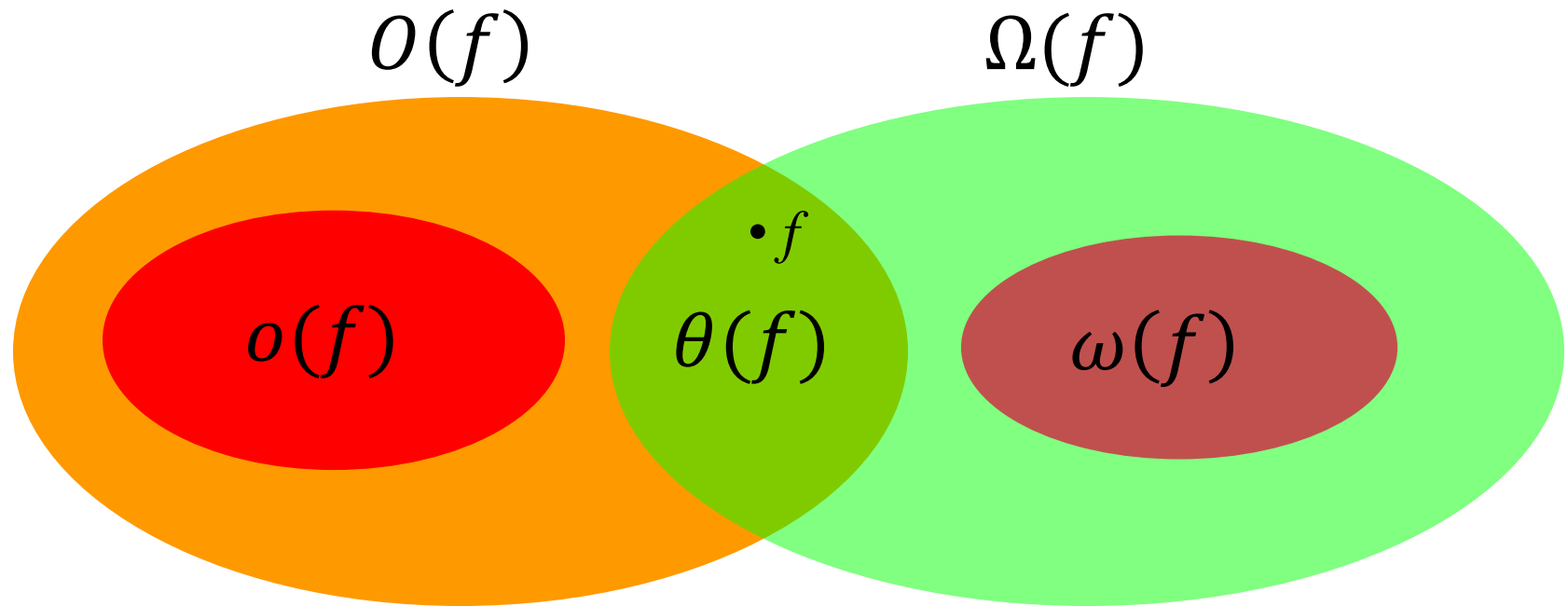
$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \frac{1}{12n} + o\left(\frac{1}{n^2}\right)\right).$$

- Der letzte Term ist für alle  $n > n_0$  kleiner oder gleich  $c \cdot 1/n^2$ .
- Da  $1/n^2$  für große  $n$  gegen 0 geht, wird der Fehler der Annäherung kleiner, je größer  $n$  wird.



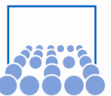
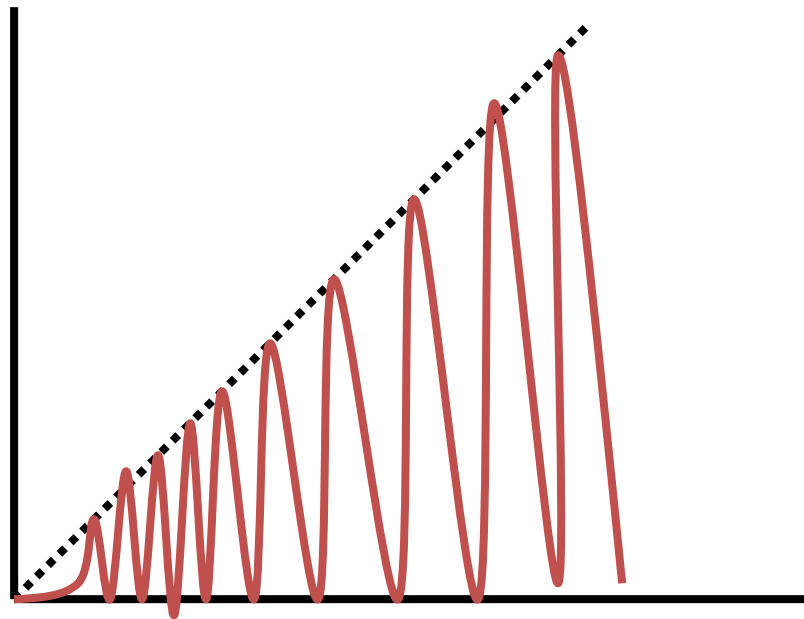
# Kapitel 2: Grundlagen (Wachstum) $\Sigma\Pi$

- Relationen zwischen Wachstumsrelationen:





- Eine Funktion, die  $O(n)$ , aber weder  $o(n)$  noch  $\Theta(n)$  ist:



- Beziehung zwischen Wachstum und Grenzwerten:

Wenn die Grenzwerte existieren, dann:

$$f \in O(g) \leftrightarrow \lim_{n \rightarrow \infty} \frac{|f(n)|}{g(n)} < \infty,$$

$$f \in o(g) \leftrightarrow \lim_{n \rightarrow \infty} \frac{|f(n)|}{g(n)} = 0,$$

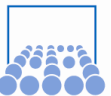
$$f \in \Omega(g) \leftrightarrow \lim_{n \rightarrow \infty} \frac{g(n)}{|f(n)|} < \infty,$$

$$f \in \omega(g) \leftrightarrow \lim_{n \rightarrow \infty} \frac{g(n)}{|f(n)|} = 0.$$



- **Strikte Ordnung** von Funktionen:
  - Häufig schreibt man  $f < g$  für  $f \in o(|g|)$ .
  - Für alle  $k > 1$  gilt:

$$\begin{aligned} 1 < \log_2 \log_2 n < \log_2 n < \log_2^k n < \\ < n^{\frac{1}{k}} < n < n \log_2 n < \\ < n^k < k^n < n! < n^n < \dots \end{aligned}$$



- Beispiele Wachstumsverhalten:

**TABLE 2** The Computer Time Used by Algorithms.

<i>Problem Size</i>	<i>Bit Operations Used</i>					
	$\log n$	$n$	$n \log n$	$n^2$	$2^n$	$n!$
10	$3 \times 10^{-9}$ s	$10^{-8}$ s	$3 \times 10^{-8}$ s	$10^{-7}$ s	$10^{-6}$ s	$3 \times 10^{-3}$ s
$10^2$	$7 \times 10^{-9}$ s	$10^{-7}$ s	$7 \times 10^{-7}$ s	$10^{-5}$ s	$4 \times 10^{13}$ yr	*
$10^3$	$1.0 \times 10^{-8}$ s	$10^{-6}$ s	$1 \times 10^{-5}$ s	$10^{-3}$ s	*	*
$10^4$	$1.3 \times 10^{-8}$ s	$10^{-5}$ s	$1 \times 10^{-4}$ s	$10^{-1}$ s	*	*
$10^5$	$1.7 \times 10^{-8}$ s	$10^{-4}$ s	$2 \times 10^{-3}$ s	10 s	*	*
$10^6$	$2 \times 10^{-8}$ s	$10^{-3}$ s	$2 \times 10^{-2}$ s	17 min	*	*

Annahme: eine Operation dauert  $10^{-9}$  Sekunden,  $\log n = \log_2 n$ .



- Hierarchie von Größenordnungen:

---

Größenordnung	Name
$O(1)$	konstante Funktionen
$O(\log n)$	logarithmische Funktionen
$O(\log^k n)$	poly-logarithmische Funktionen
$O(n)$	lineare Funktionen
$O(n \log n)$	$n \log n$ -wachsende Funktionen
$O(n^2)$	quadratische Funktionen
$O(n^3)$	kubische Funktionen
$\bigcup_{k \geq 1} O(n^k)$	polynomielle Funktionen

---



Praktische Anwendungen in der Informatik:

- Abschätzung der benötigten Laufzeit (etwa bei Echtzeitanwendungen) oder des benötigten Speicheraufwands
- Maß für die Effizienz eines Algorithmus
- Klassifizierung von Algorithmen in Komplexitätsklassen (z.B. beim Sortieren)

