



Technische  
Universität  
Braunschweig



# Algorithmen und Datenstrukturen – Übung #2

BFS, DFS und Wachstum von Funktionen

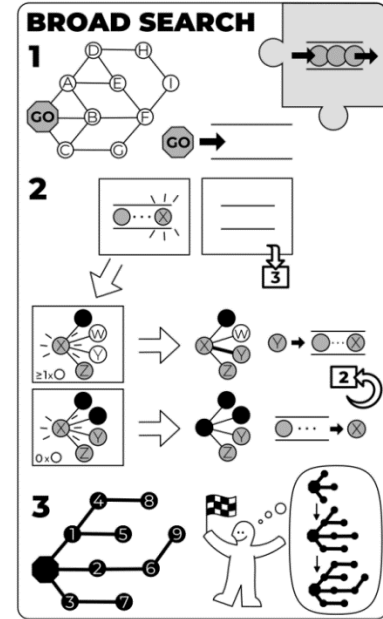
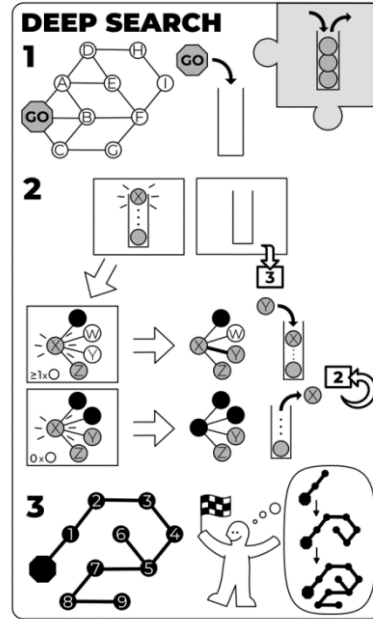
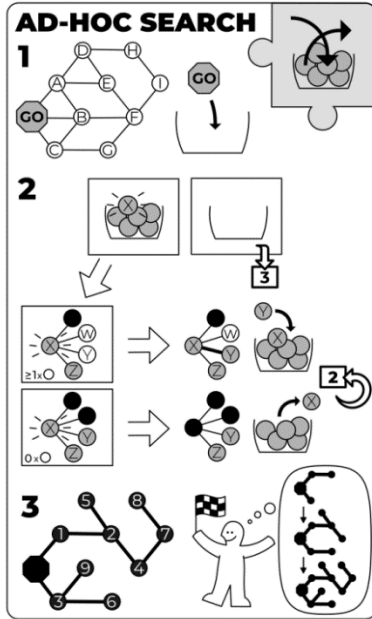
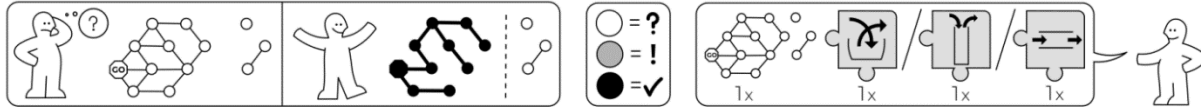
Ramin Kosfeld & Chek-Manh Loi

30.11.2023

# Suche in Graphen

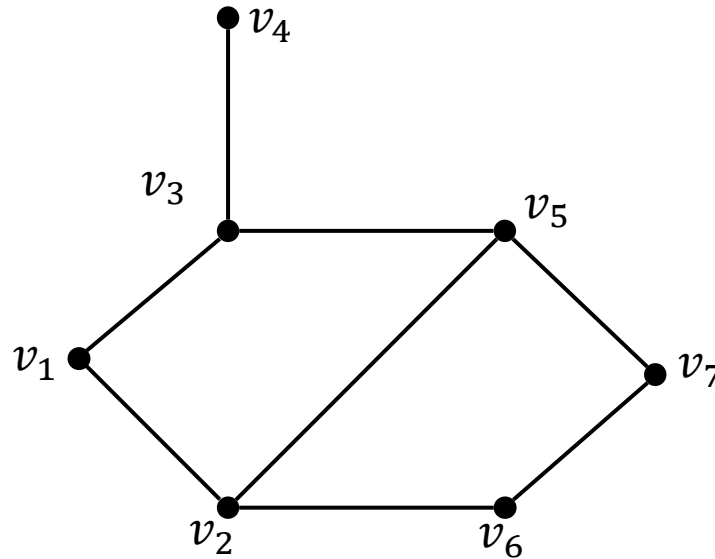
# GRÄPH SKÄN

idea-instructions.com/graph-scan/  
v1.2, CC by-nc-sa 4.0



# Breitensuche

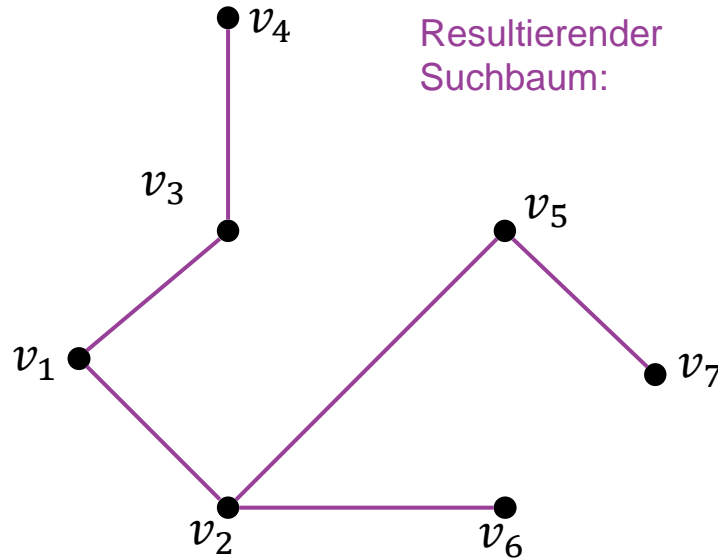
Benutze Warteschlange  
Prinzip: First-in-first-out



$v_1$   
 $v_1, v_2$   
 $v_1, v_2, v_3$   
 $v_2, v_3$   
 $v_2, v_3, v_5$   
 $v_2, v_3, v_5, v_6$   
 $v_3, v_5, v_6$   
 $v_3, v_5, v_6, v_4$   
 $v_5, v_6, v_4$   
 $v_5, v_6, v_4, v_7$   
 $v_6, v_4, v_7$   
 $v_4, v_7$   
 $v_7$   
 $\emptyset$

# Breitensuche

Benutze Warteschlange  
Prinzip: First-in-first-out

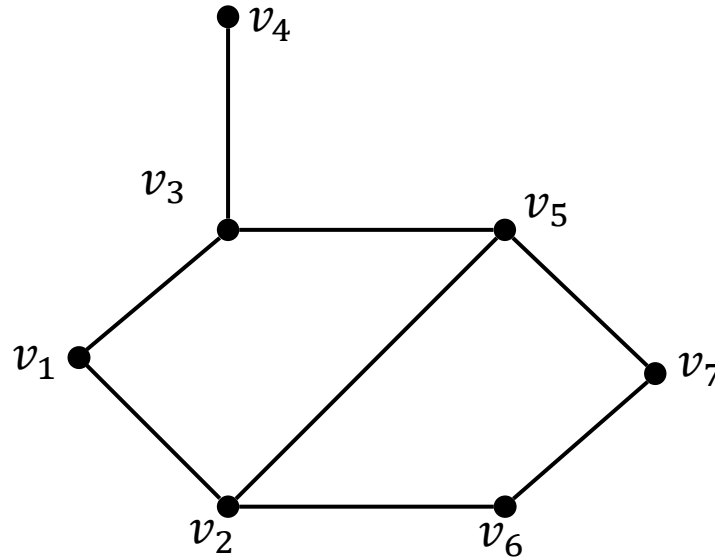


$v_1$   
 $v_1, v_2$   
 $v_1, v_2, v_3$   
 $v_2, v_3$   
 $v_2, v_3, v_5$   
 $v_2, v_3, v_5, v_6$   
 $v_3, v_5, v_6$   
 $v_3, v_5, v_6, v_4$   
 $v_5, v_6, v_4$   
 $v_5, v_6, v_4, v_7$   
 $v_6, v_4, v_7$   
 $v_4, v_7$   
 $v_7$   
 $\emptyset$

# Tiefensuche

Benutze Stapel

Prinzip: Last-in-first-out



$v_1$

$v_1, v_2$

$v_1, v_2, v_5$

$v_1, v_2, v_5, v_3$

$v_1, v_2, v_5, v_3, v_4$

$v_1, v_2, v_5, v_3$

$v_1, v_2, v_5$

$v_1, v_2, v_5, v_7$

$v_1, v_2, v_5, v_7, v_6$

$v_1, v_2, v_5, v_7$

$v_1, v_2, v_5$

$v_1, v_2$

$v_1$

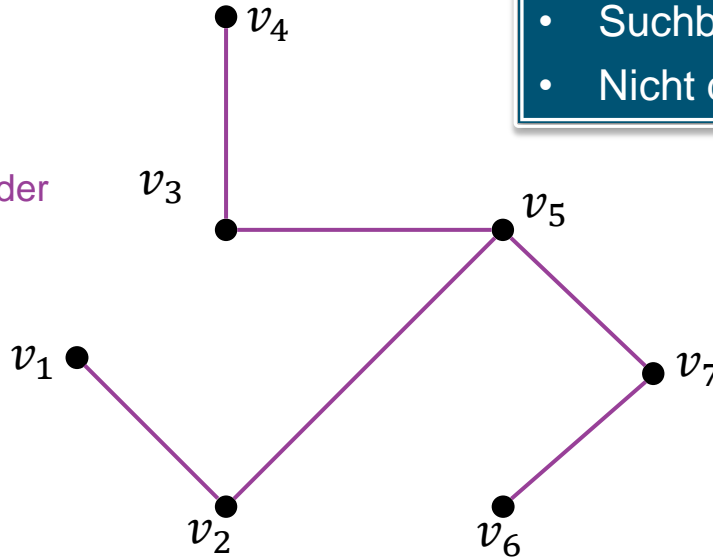
$\emptyset$

# Tiefensuche

Benutze Stapel

Prinzip: Last-in-first-out

Resultierender  
Suchbaum:



## Beliebte Fehler

- Leere Menge am Ende vergessen
- Nicht jede Änderung angegeben
- Suchbaum nicht angegeben
- Nicht den kleinsten Index beachtet

$v_1, v_2, v_5, v_3, v_4$

$v_1, v_2, v_5, v_3$

$v_1, v_2, v_5$

$v_1, v_2, v_5, v_7$

$v_1, v_2, v_5, v_7, v_6$

$v_1, v_2, v_5, v_7$

$v_1, v_2, v_5$

$v_1, v_2$

$v_1$

$\emptyset$

# Six Degrees of Kevin Bacon



Finde kürzesten Weg von einem Schauspieler über Filme zu Kevin Bacon

|  |   |
|--|---|
|  | <b>Wikipedia</b><br>Free online encyclopedia that anyone can edit                                     |
|  | <b>Social networking service</b><br>Online platform that facilitates the building of social relations |
|  | <b>Six degrees of separation</b>  |
|  | <b>Kevin Bacon</b><br>American actor  |

|   |   |
|---|---|
|  | <b>Kevin Bacon</b><br>American actor  |
|  | <b>Virtual International Authority File</b><br>International authority file |
|  | <b>Wikipedia</b><br>Free online encyclopedia that anyone can edit           |

Finde kürzesten Weg von einer Seite über enthaltene Links zu einer anderen



# Six Degrees of Kevin Bacon

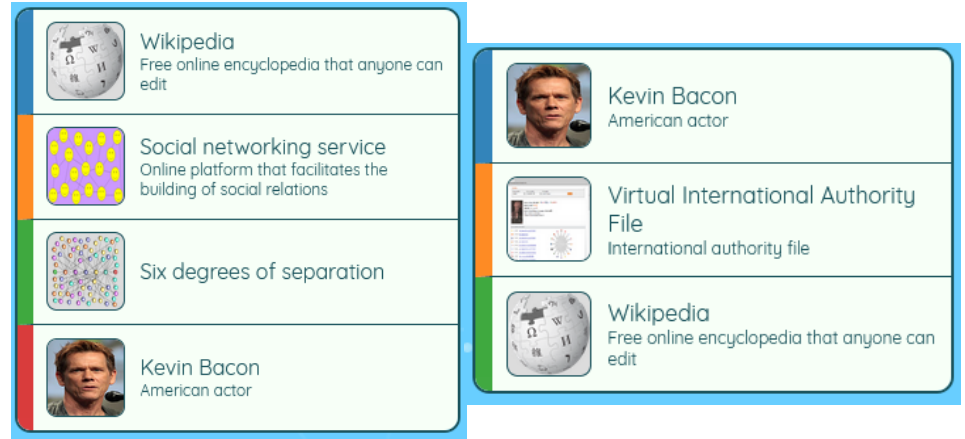


Finde kürzesten Weg von einem Schauspieler über Filme zu Kevin Bacon

## Ungerichteter Graph

Hin- und Rückweg sind gleich lang

# vs. Six Degrees of Wikipedia



Finde kürzesten Weg von einer Seite über enthaltene Links zu einer anderen

## Gerichteter Graph

Hin- und Rückweg können unterschiedlich lang sein

# Labyrinth



Die Reise ins Labyrinth, 1986

<https://fictionmachine.com/2015/07/03/everything-ive-done-ive-done-for-you-labyrinth-1985/>

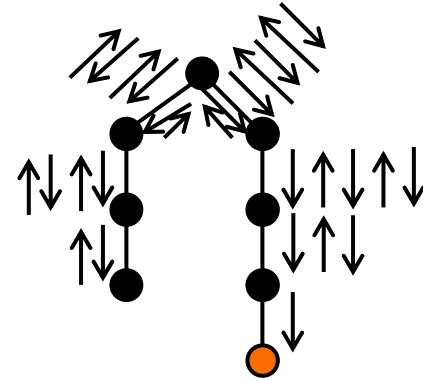
# Zeit für ein Labyrinth

Wie oft muss man durch die Gänge des Labyrinths laufen, wenn man...

1. BFS, oder
2. DFS nutzt?

## BFS - Worst-Case

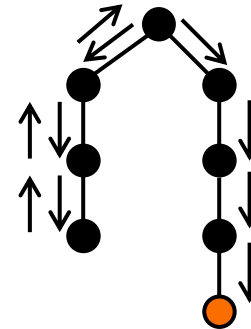
$\Omega(n^2)$  mal über Kanten laufen!



DFS schafft das schneller!

Man kann zeigen:

- Jede Kante wird maximal zwei Mal benutzt.
- Man benötigt maximal  $2n-1$  Schritte
- Es gibt Bäume, bei denen  $2n-1$  Schritte benötigt werden



# Wachstum von Funktionen

# Wir arbeiten mit Pseudocode

- Die *genaue* Laufzeit und Speicherbedarf sind stark von der Implementierung abhängig
  - Gerade, wenn wir nur mit Pseudocode arbeiten, ist es *überhaupt nicht sinnvoll*, eine genaue Funktion anzugeben
- Wir wollen nur eine grobe Abschätzung, eine Reduzierung auf das Wesentliche (eine Abschätzung, die für alle korrekten Implementierungen gilt!)
- In welcher Ordnung wächst die entsprechende Funktion?

# Die Idee

Laufzeit und Speicherbedarf sind Funktionen. Wir wollen also Funktionen untersuchen, ...

Unabhängig von  
**konstanten Faktoren**  
– die kennen wir *eh nicht*

Betrachte die Entwicklung  
**auf lange Sicht**  
– für *immer größere* Eingaben

→ Ordne so das Wachstum der Funktion in eine Klasse ein

# Preliminaries

Betrachte Laufzeit / Speicher etc. als Funktion  $T: \mathbb{N} \rightarrow \mathbb{R}^+$

→ Die Funktion ist abhängig von der *Größe der Eingabe*  $n$ .

Verschiedene Systeme liefern verschiedene Laufzeiten, z.B.  $T_1(n)$  und  $T_2(n)$ .

# $\mathcal{O}$ -Notation

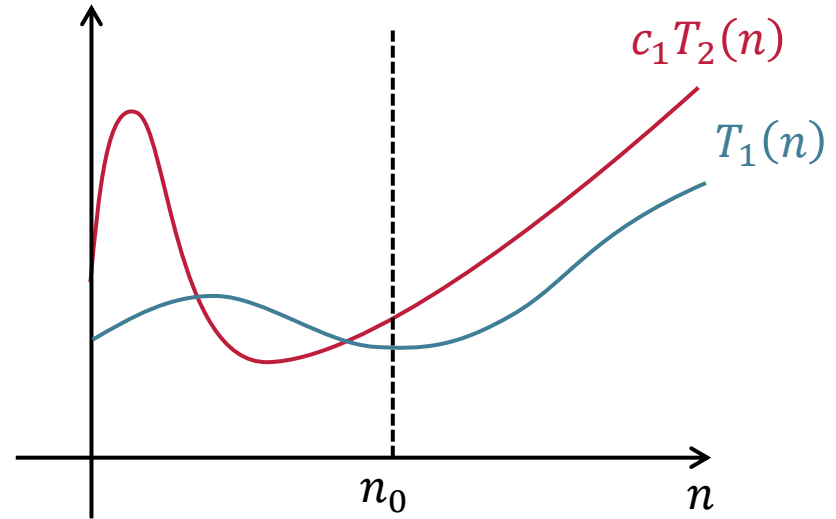
*Definition:*

Es gibt Konstanten  $n_0 \in \mathbb{N}$  und  $c_1 \in \mathbb{R}^+$ ,  
sodass für alle  $n \geq n_0$  gilt:

$$0 \leq T_1(n) \leq c_1 T_2(n)$$



$$T_1(n) \in \mathcal{O}(T_2(n))$$



„ $T_1$  wächst (asymptotisch)  
höchstens so schnell wie  $T_2$ “

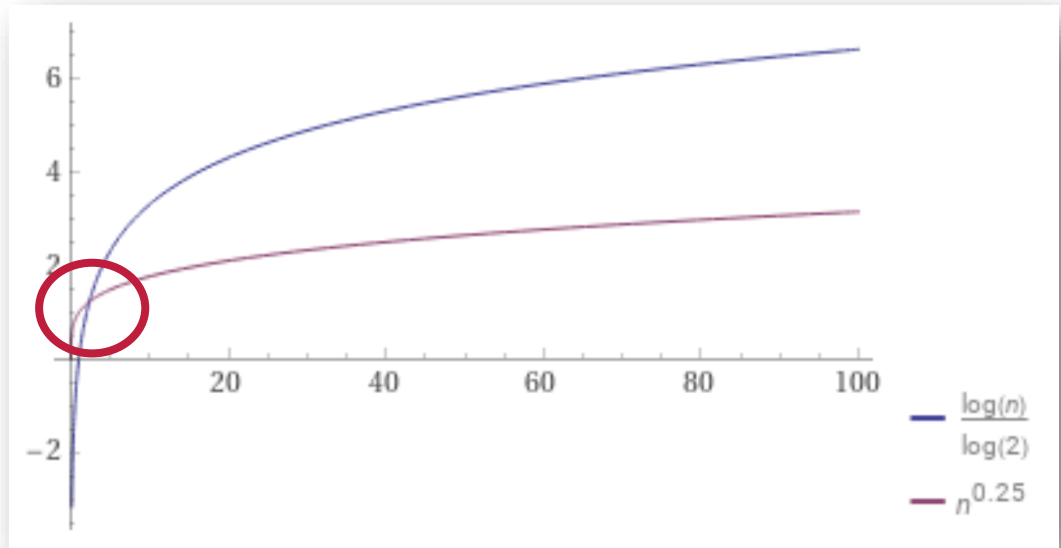


# Wie zeigen wir Zugehörigkeit?

Ist  $n^{0.25} \in O(\log_2 n)$ ?

Ja! Abbildung rechts zeigt das:  
Wähle  $c = 1, n_0 \geq 5$

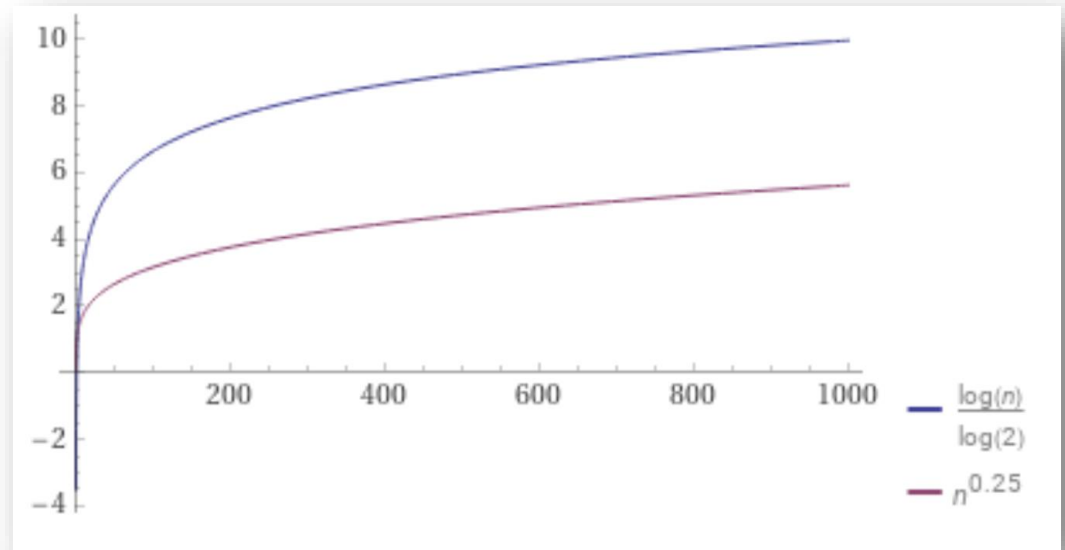
(Sieht man ja)



# Wie zeigen wir Zugehörigkeit?

Ist  $n^{0.25} \in O(\log_2 n)$ ?

Ja! Abbildung rechts zeigt das:  
Wähle  $c = 1, n_0 \geq 5$



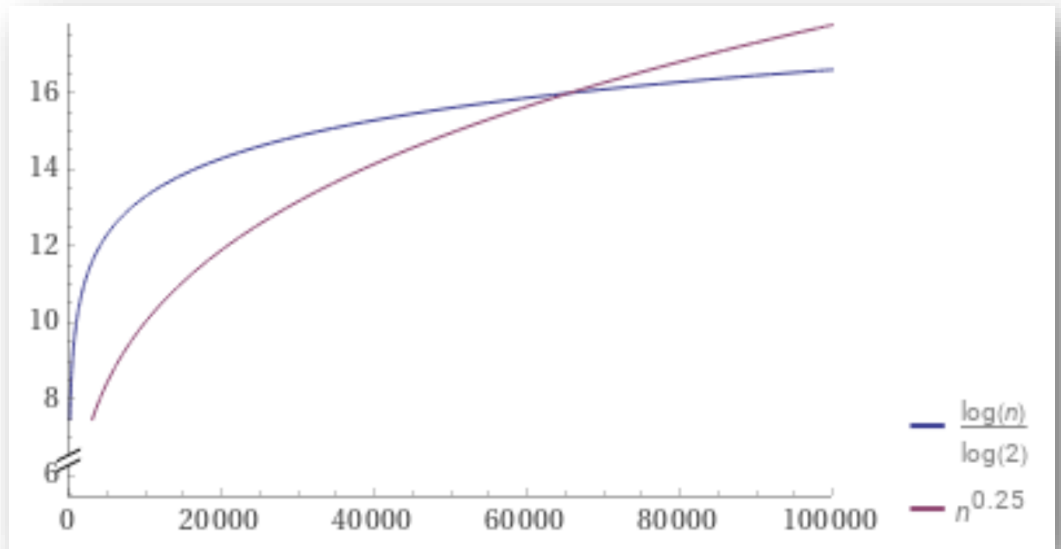
# Wie zeigen wir Zugehörigkeit?

Ist  $n^{0.25} \in O(\log_2 n)$ ?

~~Ja! Abbildung rechts zeigt das:  
Wähle  $c = 1, n_0 \leq 5$~~

Auch wenn es für kleine  $n$  gut aussieht, kann es für große  $n$  anders sein!

Letztendlich muss eine Allaussage bewiesen werden:  
„Für alle  $n \geq n_0$ “



# Eine bessere Möglichkeit:

Zeige  $4n^2 + 12n - 15 \in \mathcal{O}(n^2)$



Bestimme  $n_0$  und  $c_1$ , sodass für alle  $n \geq n_0$  gilt:  $0 \leq 4n^2 + 12n - 15 \leq c_2 \cdot n^2$

## Suche nach $c_1$

$$4n^2 + 12n - 15 \leq$$

Also:  $4n^2 + 12n - 15 \leq 16n^2$  für  $n \geq 1$ .

Die Aussagen oben gelten ab  $n_0 = 1$ .

Also haben wir mit  $c_1 = 16$  und  $n_0 = 1$  Werte für die Konstanten gefunden, für die die Definition gilt.

# Eine bessere Möglichkeit:

Zeige  $4n^2 + 12n - 15 \in \mathcal{O}(n^2)$



Bestimme  $n_0$  und  $c_1$ , sodass für alle  $n \geq n_0$  gilt:  $0 \leq 4n^2 + 12n - 15 \leq c_1 \cdot n^2$

## Suche nach $c_1$

$$4n^2 + 12n - 15 \leq 4n^2 + 12n \leq 4n^2 + 12n^2 = 16n^2 \text{ für } n \geq 1.$$

$$\text{Also: } 4n^2 + 12n - 15 \leq 16n^2 \text{ für } n \geq 1.$$

Die Aussagen oben gelten ab  $n_0 = 1$ .

Also haben wir mit  $c_1 = 16$  und  $n_0 = 1$  Werte für die Konstanten gefunden, für die die Definition gilt.

Damit ist  $4n^2 + 12n - 15 \in \mathcal{O}(n^2)$ .

# $\Omega$ -Notation

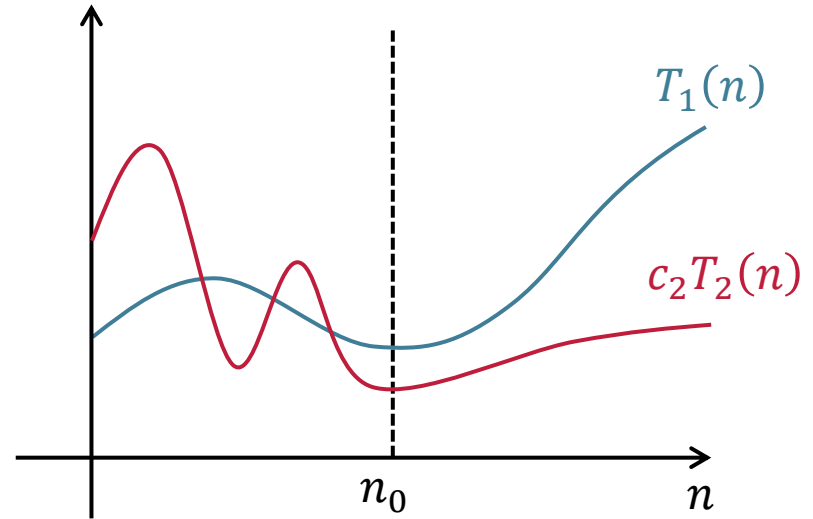
*Definition:*

Es gibt Konstanten  $n_0 \in \mathbb{N}$  und  $c_2 \in \mathbb{R}^+$ ,  
sodass für alle  $n \geq n_0$  gilt:

$$T_1(n) \geq c_2 T_2(n) \geq 0$$



$$T_1(n) \in \Omega(T_2(n))$$



„ $T_1$  wächst (asymptotisch)  
mindestens so schnell wie  $T_2$ “

# $\Theta$ -Notation

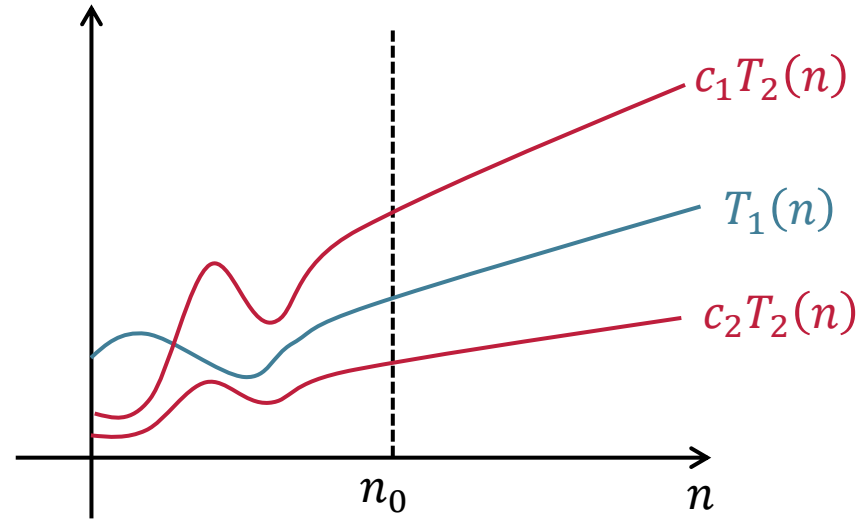
Definition:

Es gibt Konstanten  $n_0 \in \mathbb{N}$  und  $c_1, c_2 \in \mathbb{R}^+$ , sodass für alle  $n \geq n_0$  gilt:

$$0 \leq c_2 T_2(n) \leq T_1(n) \leq c_1 T_2(n)$$



$$T_1(n) \in \Theta(T_2(n))$$



„ $T_1$  wächst (asymptotisch) genau so schnell wie  $T_2$ “

# Zusammengefasst

## Achtung:

$\mathcal{O}$ ,  $\Omega$  und  $\Theta$  sind *Mengen*  
(von Funktionen)!

Wir haben folgende drei Definitionen gesehen:

### $\mathcal{O}$ -Notation:

Es gibt Konstanten  $n_0 \in \mathbb{N}$  und  $c_1 \in \mathbb{R}^+$ , sodass für alle  $n \geq n_0$  gilt:

$$0 \leq T_1(n) \leq c_1 T_2(n) \Leftrightarrow T_1(n) \in \mathcal{O}(T_2(n))$$

### $\Omega$ -Notation:

Es gibt Konstanten  $n_0 \in \mathbb{N}$  und  $c_2 \in \mathbb{R}^+$ , sodass für alle  $n \geq n_0$  gilt:

$$T_1(n) \geq c_2 T_2(n) \geq 0 \Leftrightarrow T_1(n) \in \Omega(T_2(n))$$

### $\Theta$ -Notation:

Es gibt Konstanten  $n_0 \in \mathbb{N}$  und  $c_1, c_2 \in \mathbb{R}^+$ , sodass für alle  $n \geq n_0$  gilt:

$$0 \leq c_2 T_2(n) \leq T_1(n) \leq c_1 T_2(n) \Leftrightarrow T_1(n) \in \Theta(T_2(n))$$

→ „Landau-Symbole“



# Rechenregeln

Ganz grundsätzlich gelten diese Beobachtungen:

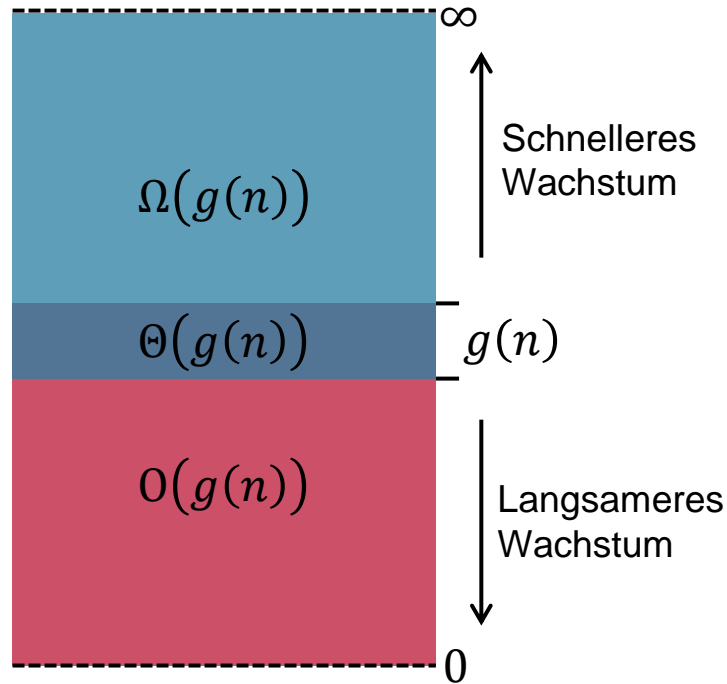
$$f \in \mathcal{O}(g) \Leftrightarrow c \cdot f \in \mathcal{O}(g) \text{ für } c \in \mathbb{R}^+$$

$$f \in \mathcal{O}(g) \Leftrightarrow g \in \Omega(f)$$

$$f \in \Theta(g) \Leftrightarrow f \in \mathcal{O}(g) \wedge f \in \Omega(g)$$

$$f \in \Theta(g) \Leftrightarrow g \in \Theta(f)$$

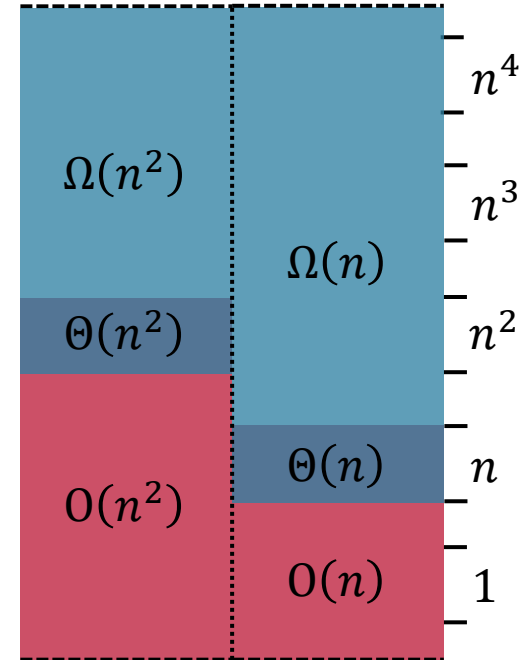
# Relationen zwischen Klassen - Eine grafische Darstellung



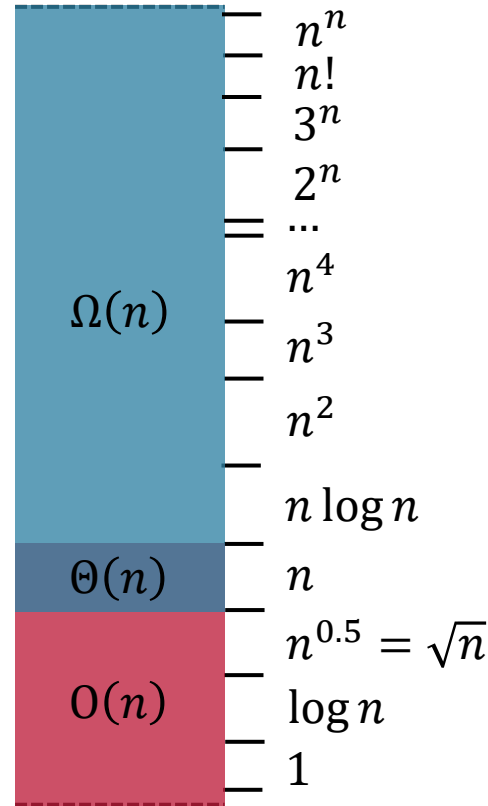
Wir können sogar Klassen miteinander vergleichen.  
Beispiel:

$$\Theta(n^2) \not\subseteq \Omega(n)$$

Zum Merken:  
Wächst die Funktion schneller, wächst der  $\mathcal{O}$ -Bereich und der  $\mathcal{O}$ -Bereich schrumpft.



# Relationen zwischen Klassen - Eine grafische Darstellung



# Relationen zwischen Klassen – Eine Tabelle

| Bedingung   | Klasse         |                 |                 |                 |
|---|----------------|-----------------|-----------------|-----------------|
|   | Klasse         | $O(g(n))$       | $\Theta(g(n))$  | $\Omega(g(n))$  |
| $f(n) \in o(g(n))$<br>$(o(g(n)) := O(g(n)) \setminus \Theta(g(n)))$<br>"Klein-o-Notation"                         | $O(f(n))$      | $\not\subseteq$ | X               | X               |
|   | $\Theta(f(n))$ | $\not\subseteq$ | X               | X               |
|   | $\Omega(f(n))$ | X               | $\not\supseteq$ | $\not\supseteq$ |
| $f(n) \in \Theta(g(n))$   | $O(f(n))$      | =               | $\not\supseteq$ | X               |
|   | $\Theta(f(n))$ | $\subseteq$     | =               | $\not\subseteq$ |
|   | $\Omega(f(n))$ | X               | $\not\supseteq$ | =               |
| $f(n) \in \omega(g(n))$<br>$(\omega(g(n)) := \Omega(g(n)) \setminus \Theta(g(n)))$<br>"Klein- $\omega$ -Notation" | $O(f(n))$      | $\not\supseteq$ | $\not\supseteq$ | X               |
|   | $\Theta(f(n))$ | X               | X               | $\not\subseteq$ |
|   | $\Omega(f(n))$ | X               | X               | $\not\subseteq$ |

# Beispiele

# Beispiel 1

Zeige  $4n^2 + 12n - 15 \in \Theta(n^2)$



Bestimme  $n_0, c_1, c_2$ , sodass für alle  $n \geq n_0$  gilt:  
 $0 \leq c_1 \cdot n^2 \leq 4n^2 + 12n - 15 \leq c_2 \cdot n^2$

**Suche nach  $c_2$**

$$4n^2 + 12n - 15$$

**Suche nach  $c_1$**

$$4n^2 + 12n - 15$$

# Beispiel 1

Zeige  $4n^2 + 12n - 15 \in \Theta(n^2)$

Bestimme  $n_0, c_1, c_2$ , sodass für alle  $n \geq n_0$  gilt:  
 $0 \leq c_1 \cdot n^2 \leq 4n^2 + 12n - 15 \leq c_2 \cdot n^2$

## Suche nach $c_2$

$$4n^2 + 12n - 15 \leq 4n^2 + 12n \leq 4n^2 + 12n^2 = 16n^2 \text{ für } n \geq 1.$$

## Suche nach $c_1$

$$4n^2 + 12n - 15 \geq 4n^2 - 15 \stackrel{n \geq 4}{\geq} 4n^2 - n^2 = 3n^2 \text{ für } n \geq 4.$$

Beide Ungleichungen gelten ab  $n_0 = 4$ . Also  $c_1 = 3$ ,  $c_2 = 16$  und  $n_0 = 4$ .

## Beispiel 2

Zeige oder widerlege:  $2^n \in \Theta(3^n)$

Zunächst:  $2^n \in \mathcal{O}(3^n)$ , denn  $2^n \leq (2 + 1)^n = 3^n$ .

Aber:  $2^n \notin \Omega(3^n)$ !

Ansonsten gäbe es eine Konstante  $c_1$  mit

$$2^n \geq c_1 \cdot 3^n, \text{ also } \frac{2^n}{3^n} \geq c_1$$

Aber:  $\frac{2^n}{3^n} = \left(\frac{2}{3}\right)^n$  und  $\lim_{n \rightarrow \infty} \left(\frac{2}{3}\right)^n = 0$ , d.h. dieses  $c_1$  kann nicht existieren!

$\Rightarrow$  Aussage widerlegt.



## Beispiel 3

$$\log_2 n \in \mathcal{O}(n)$$

Satz (1): Für jedes  $c > 0$  gibt es ein  $n_0$ , sodass für alle  $n \geq n_0$  gilt:  $\log_2 n \leq c \cdot n$

*Beweisidee:* Zeige, dass  $\lim_{n \rightarrow \infty} \frac{\log_2 n}{n} = 0$ , d.h. für wachsendes  $n$  kommen wir beliebig nah an 0 heran. D.h. wir können  $n_0$  so wählen, dass  $\frac{\log_2 n}{n} \leq c$  für alle  $n \geq n_0$  gilt.

# Beispiel 3

Satz (1): Für jedes  $c > 0$  gibt es ein  $n_0$ , sodass für alle  $n \geq n_0$  gilt:  $\log_2 n \leq c \cdot n$

Satz (2): Seien  $a, b \in \mathbb{R}^+$ . Dann gilt  $\log_2^a n \in \mathcal{O}(n^b)$ .

$$\log_2^a n = (\log_2 n)^a$$

Beweis: Wir setzen  $c = 1$ . Wir zeigen, dass ab einem  $n_0$  gilt:

$$\begin{aligned} & \log_2^a n \leq n^b \\ \Leftrightarrow & \log_2 \log_2^a n \leq \log_2 n^b \\ \Leftrightarrow & a \cdot \log_2 \log_2 n \leq b \cdot \log_2 n \\ m := \log_2 n & \\ \Leftrightarrow & \log_2 m \leq \frac{b}{a} m \end{aligned}$$

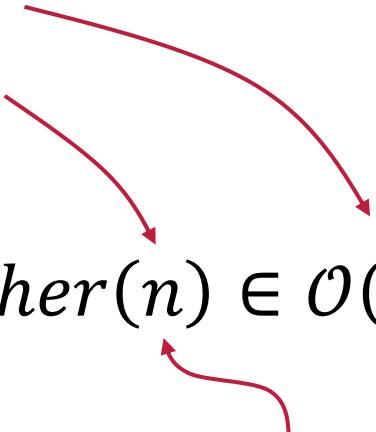
Da  $a, b \in \mathbb{R}^+$  ist auch  $\frac{b}{a} \in \mathbb{R}^+$ . Nach Satz (1) oben ist die letzte Ungleichung ab einem  $n_0$  wahr. Da wir Äquivalenzumformungen benutzt haben, können wir die Lösungskette von unten nach oben gehen.

# Codierungsgrößen

# Codierungsgrößen

Wie viel Speicher brauchen wir, um Dinge im Speicher darzustellen?

- Speicherbedarf einschätzen
- Eingabegröße bestimmen


$$\textit{speicher}(n) \in \mathcal{O}(n^2)$$

Wir geben Laufzeit- / Speicherverhalten immer in Abhängigkeit von der Größe der Eingabe an

# Codierungsgröße - Zahlen

Dezimal

27

## b-adische Notation

- Dezimal (10-adisch)
- Binär (2-adisch)
- Oktal (8-adisch)
- Hexadezimal (16-adisch)

# Codierungsgröße - Zahlen

| Dezimal | Unär | Binär | Oktal | Hexadezimal |
|---------|------|-------|-------|-------------|
| 27      |      | 11011 | 33    | 1B          |

## b-adische Notation

- Dezimal (10-adisch)
- Binär (2-adisch)
- Oktal (8-adisch)
- Hexadezimal (16-adisch)
- Allgemein:  $a_m a_{m-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-\infty}$  wobei  $n = \sum_{i=-\infty}^m a_i b^i$  und  $0 \leq a < b$

# Codierungsgröße - Zahlen

| Dezimal | Unär | Binär | Oktal | Hexadezimal |
|---------|------|-------|-------|-------------|
| 27      |      | 11011 |       |             |

## b-adische

- Dezimal
- Binär
- Oktal
- Hexadezimal
- Allgemein:  $a_m a_{m-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-\infty}$  wobei  $n = \sum_{i=-\infty}^m a_i b^i$  und  $0 \leq a < b$

**TL;DR: Codierungsgrößen**  
Ganze Zahlen belegen *logarithmisch* viel Speicher  
(zur maximalen Zahlengröße)

# Codierungsgröße - Beispiele

- Zeichenketten/Strings  $S$ :  $\approx |S| \cdot \log N$   
für  $N$  mögliche Zeichen.

"(>o.o)>einfach\_orangensaft<(o.o<)"

## Beispiel ASCII-Zeichen

7 bits pro Symbol  $\rightarrow$  128 mögliche Zeichen

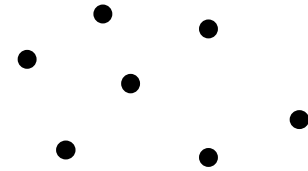
| Code | ...0 | ...1 | ...2 | ...3 | ...4 | ...5 | ...6 | ...7 | ...8 | ...9 | ...A | ...B | ...C | ...D | ...E | ...F |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 0... | NUL  | SOH  | STX  | ETX  | EOT  | ENQ  | ACK  | BEL  | BS   | HT   | LF   | VT   | FF   | CR   | SO   | SI   |
| 1... | DLE  | DC1  | DC2  | DC3  | DC4  | NAK  | SYN  | ETB  | CAN  | EM   | SUB  | ESC  | FS   | GS   | RS   | US   |
| 2... | SP   | !    | "    | #    | \$   | %    | &    | '    | (    | )    | *    | +    | ,    | -    | .    | /    |
| 3... | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | :    | ;    | <    | =    | >    | ?    |
| 4... | @    | A    | B    | C    | D    | E    | F    | G    | H    | I    | J    | K    | L    | M    | N    | O    |
| 5... | P    | Q    | R    | S    | T    | U    | V    | W    | X    | Y    | Z    | [    | \    | ]    | ^    | _    |
| 6... | `    | a    | b    | c    | d    | e    | f    | g    | h    | i    | j    | k    | l    | m    | n    | o    |
| 7... | p    | q    | r    | s    | t    | u    | v    | w    | x    | y    | z    | {    |      | }    | ~    | DEL  |



# Codierungsgröße - Beispiele

- Zeichenketten/Strings  $S$ :  $\approx |S| \cdot \log N$   
für  $N$  mögliche Zeichen.
- Punktmenge  $P$  in  $d$  Dimensionen  
mit  $N$  als die größtmögliche Koordinate:  
 $\approx d \cdot |P| \cdot \log N$ ,
- $n \times m$ -Matrizen  
mit dem größtmöglichen Wert  $N$ :  
 $\approx n \cdot m \cdot \log(N)$ ,

"(>o.o)>einfach\_orangensaft<(o.o<)"



$$\begin{pmatrix} 5 & 12 & 1 \\ 6 & 22 & 5 \\ 0 & 42 & 21 \end{pmatrix}$$

# ... nächstes Mal:

## Beweise – Vollständige Induktion



Technische  
Universität  
Braunschweig

Arne Schmidt | 15.12.2022 | Übung 4 | Seite 10

## ... vollständige Induktion!