Technische Universität Braunschweig

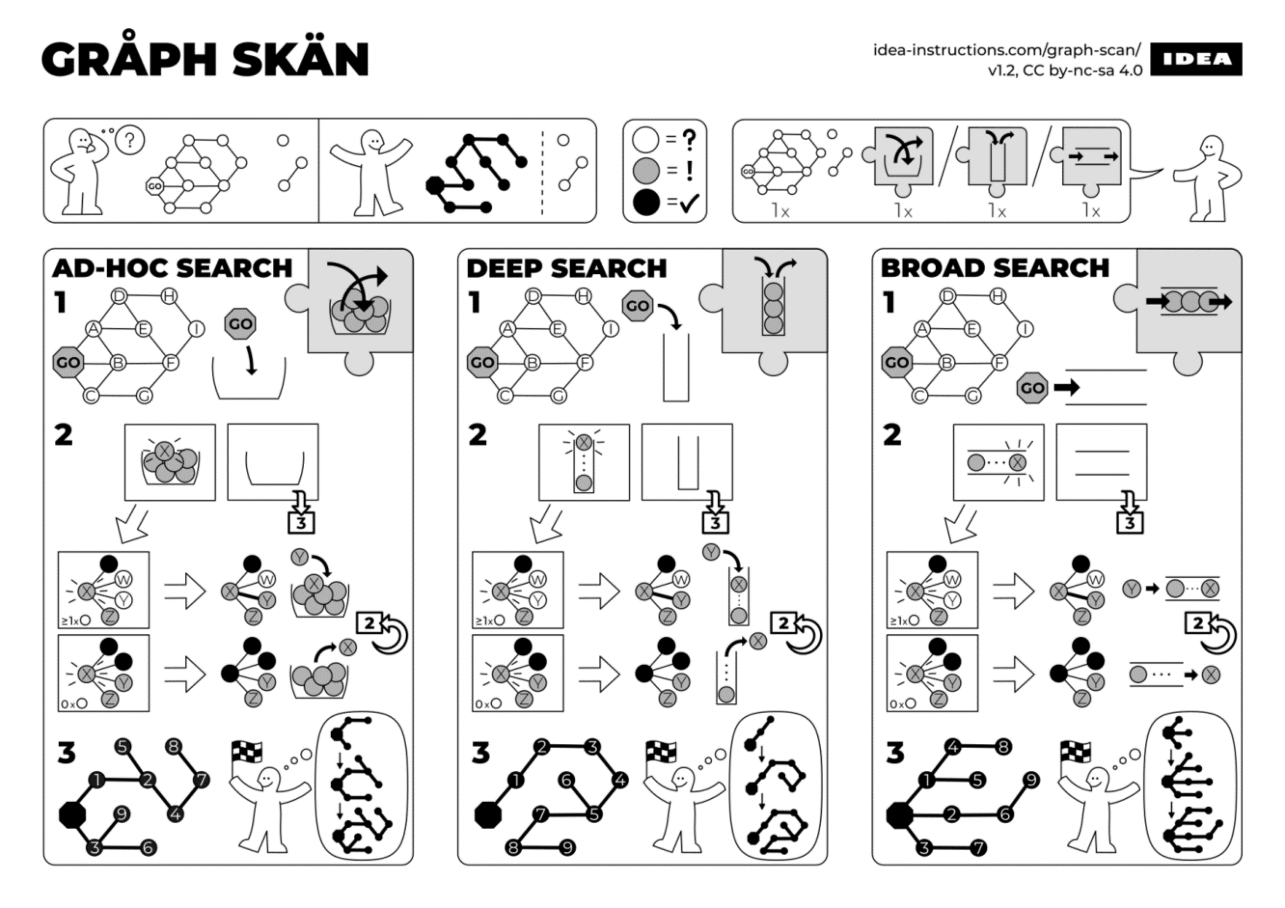
Algorithmen und Datenstrukturen – Übung #3

BFS/DFS, Wachstum von Funktionen

Arne Schmidt

01.12.2022

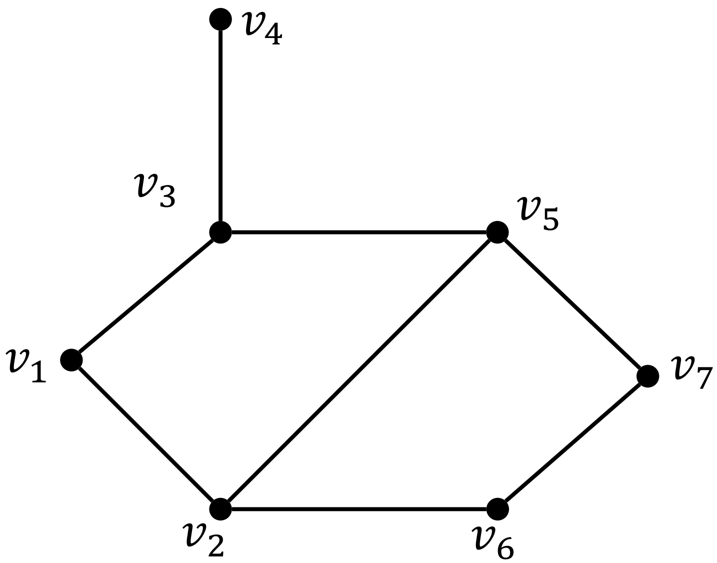
Suche in Graphen



1: https://idea-instructions.com/graph-scan/

Breitensuche

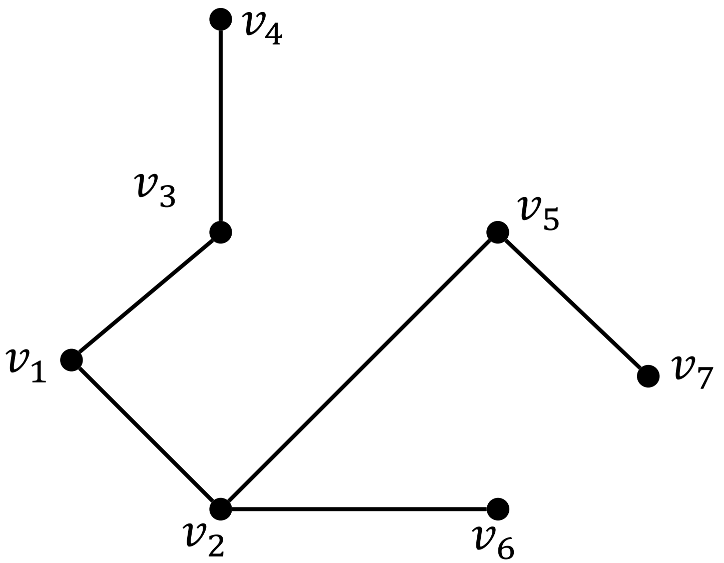
Benutze Warteschlange
Prinzip: First-in-first-out



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

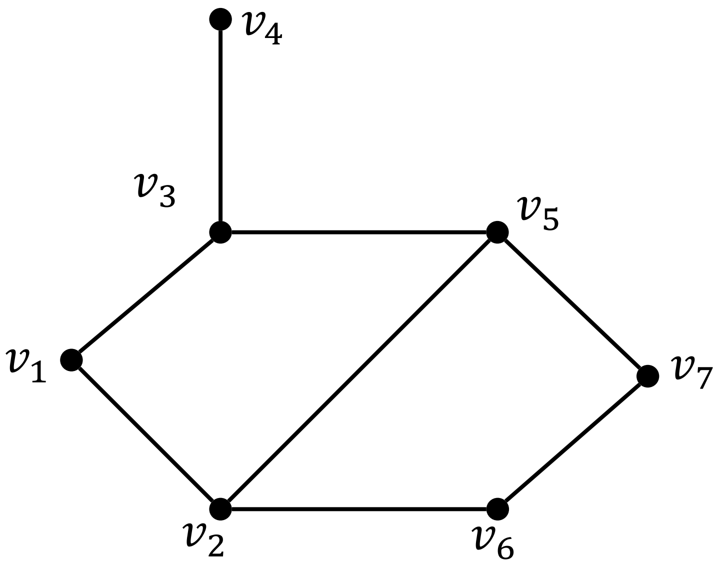
Benutze Warteschlange
Prinzip: First-in-first-out



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

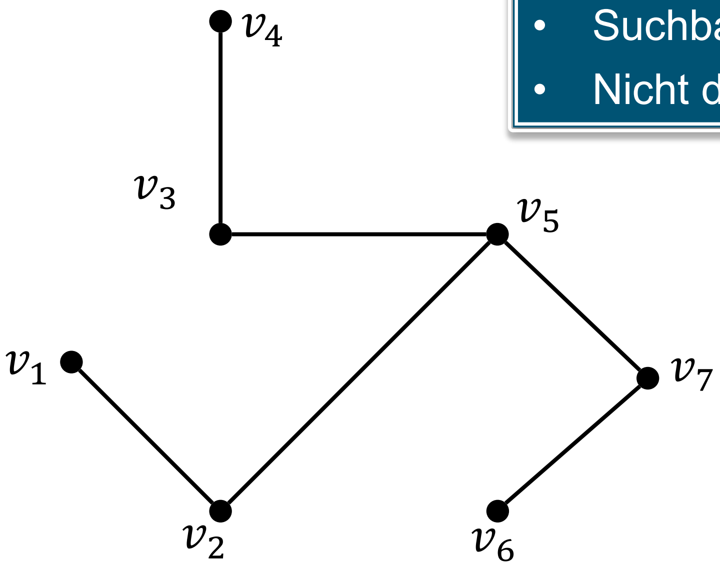
Benutze Stapel
Prinzip: Last-in-first-out



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

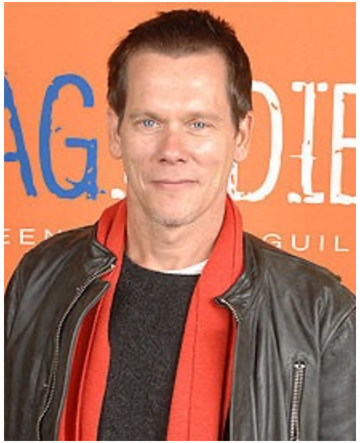


Beliebte Fehler

- 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
$\emptyset$

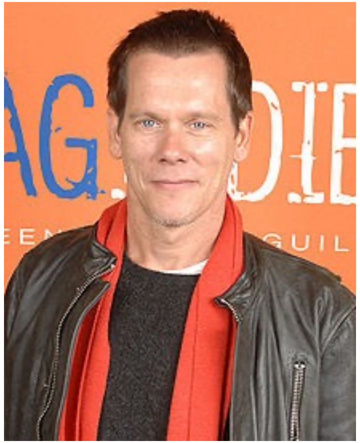
Six Degrees of Kevin Bacon vs. Six Degrees of Wikipedia



Finde kürzesten Weg von einem Schauspieler über Filme zu Kevin Bacon

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

Six Degrees of Kevin Bacon vs. Six Degrees of Wikipedia



Finde kürzesten Weg von einem Schauspieler über Filme zu Kevin Bacon

Ungerichteter Graph

Hin- und Rückweg sind gleich lang

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

Gerichteter Graph

Hin- und Rückweg sind verschieden lang

Labyrinthe

Die Reise ins Labyrinth, 1986

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

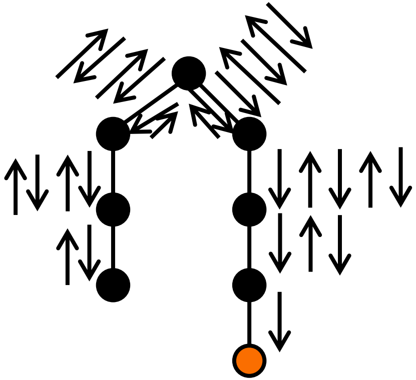
Zeit für ein Labyrinth

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

1. BFS, oder
2. DFS nutzt?

BFS - Worst-Case

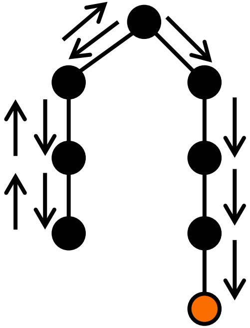
$\Omega(n^2)$ mal über Kanten laufen!



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

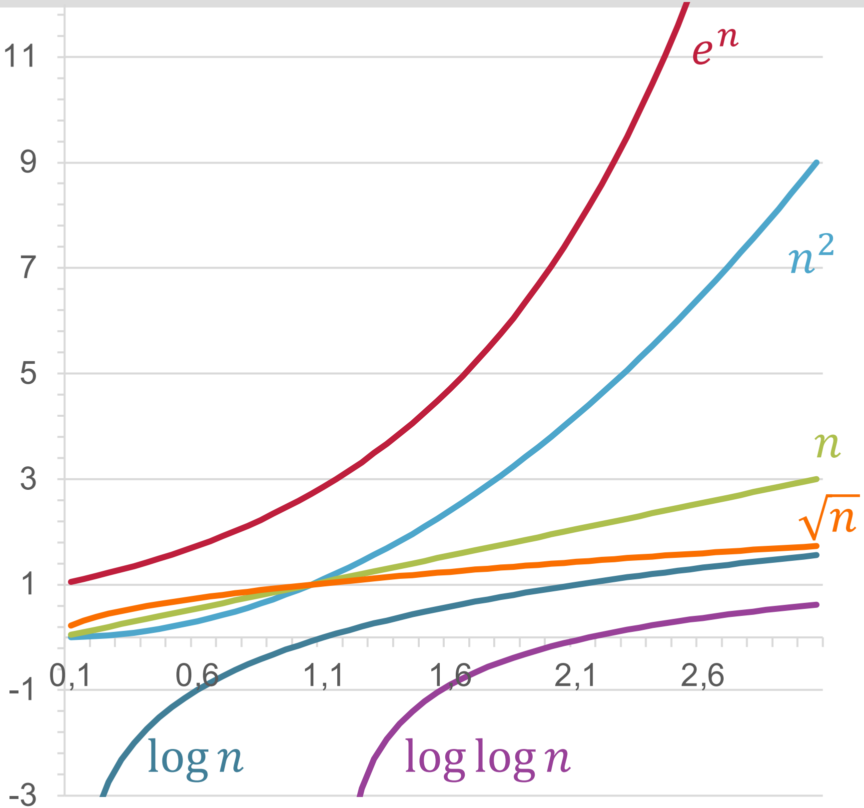


Wachstum von Funktionen

Wachstum von Funktionen / Laufzeiten

Wie lange benötigen Algorithmen?

- Absolute/Genaue Laufzeit ist nicht immer nötig
- Abschätzen: In welcher *Ordnung* wächst die Laufzeit?
- Wovon hängt die Laufzeit ab?
  - Was ist gegeben? (Zahlen, Strings, Graphen,…)
  - Wie viel ist gegeben? (Codierungsgröße)



Codierungsgröße - Zahlen

Dezimal
27

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	~~IIII~~ ~~IIII~~ ~~IIII~~ ~~IIII~~ ~~IIII~~ II	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 $0 \leq a < b$

Zahlen nutzen gewöhnlich eine binäre Darstellung, d.h. eine Zahl belegt $\log_2 n$ bits.

Codierungsgröße - Beispiele

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

(>o.o)>Das_ist_ein_String!!¥(°o°)¥

Beispiel ASCII-Zeichen

7 bits pro Symbol → 128 mögliche Zeichen

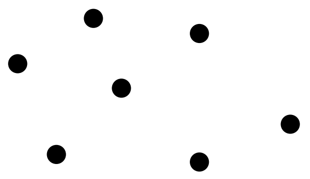
Code	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1…	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2…	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3…	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4…	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5…	P	Q	R	S	T	U	V	W	X	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.

(>o.o)>Das_ist_ein_String!!¥(°o°)¥

- Punktmenge P in d Dimensionen: $\approx d \cdot |P| \cdot \log N$, wobei N die größte Koordinate ist.



- $n{\times}m$-Matrizen: $\approx n \cdot m \cdot \log(N)$, wobei N der größtmögliche Wert ist.

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

Codierungsgröße - Graphen

Wie kann man Graphen speichern?

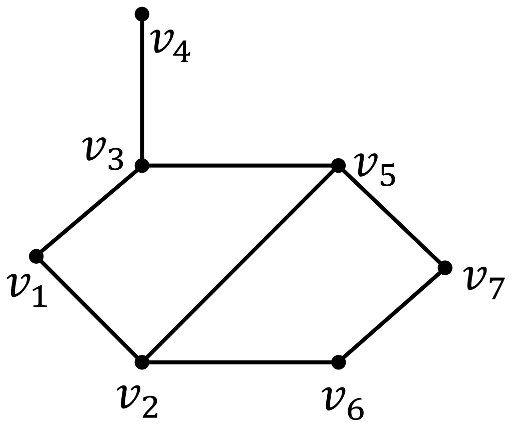
Adjazenzmatrix

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$|V|^2$ bits

Adjazenzliste

$v_1: v_2, v_3$
$v_2: v_1, v_5, v_6$
$v_3: v_1, v_4, v_5$
$v_4: v_3$
$v_5: v_2, v_3, v_7$
$v_6: v_2, v_7$
$v_7: v_5, v_6$



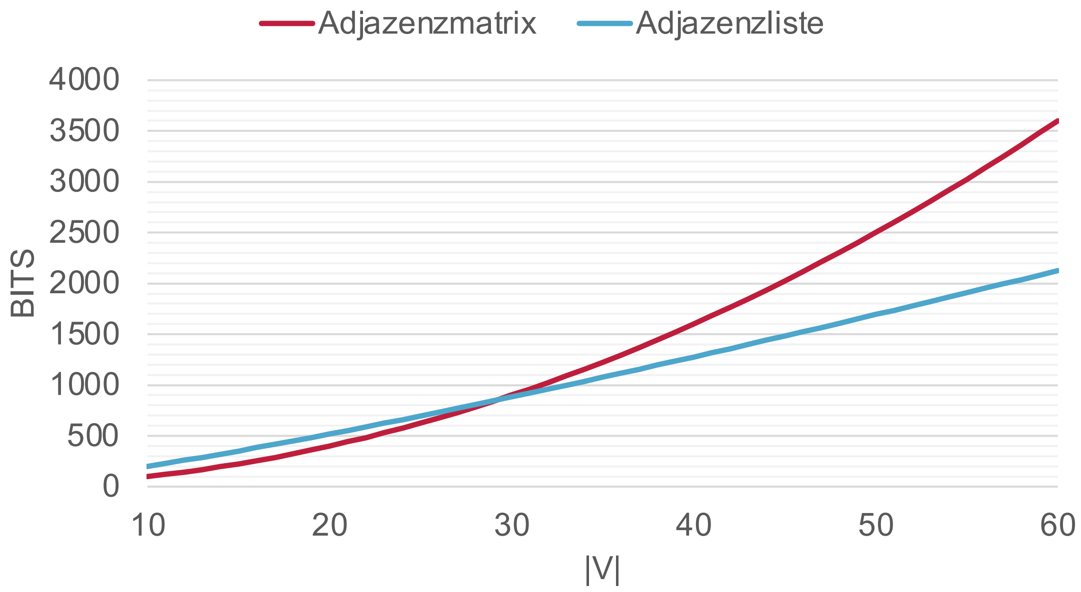
$\approx (|V| + 2|E|) \cdot \log|V|$ bits

Codierungsgröße - Graphen

Adjazenzmatrix: $|V|^2$ bits

Adjazenzliste: $(|V| + 2|E|) \cdot \log|V|$ bits

Annahme: Der Graph besitzt 3-Mal so viele Kanten wie Knoten.

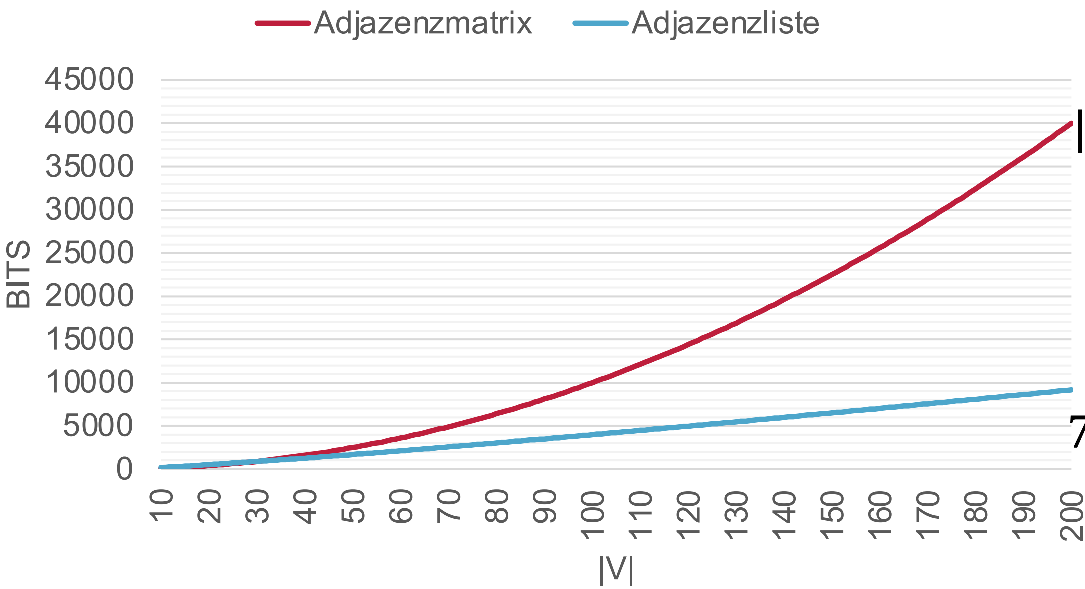


Codierungsgröße - Graphen

Adjazenzmatrix: $|V|^2$ bits

Adjazenzliste: $(|V| + 2|E|) \cdot \log|V|$ bits

Annahme: Der Graph besitzt 3-Mal so viele Kanten wie Knoten.



$|V|^2$ bits

$7|V| \cdot \log|V|$ bits

Wachstum von Funktionen

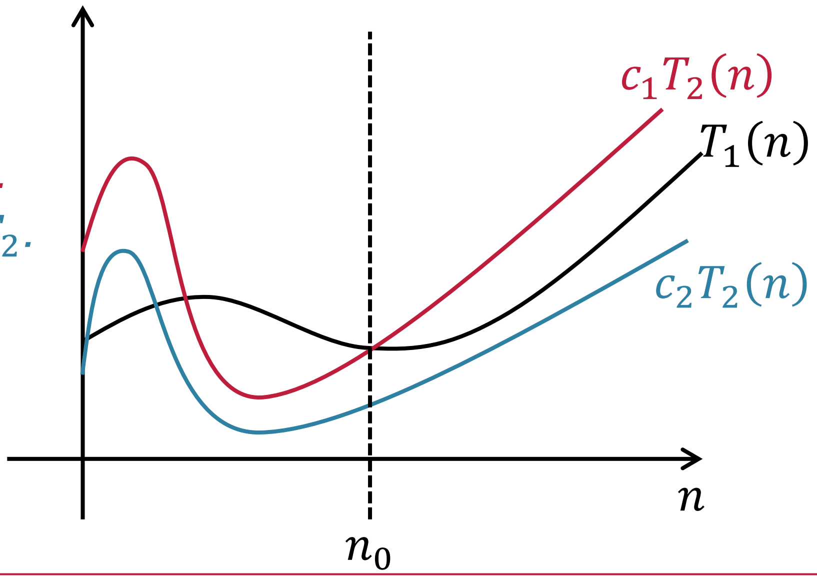
Betrachte Laufzeit als Funktion $T: \mathbb{N} \to \mathbb{R}^+$

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

Aber:
$T_1(n)$ und $T_2(n)$ wachsen “ähnlich schnell”

$T_1(n)$ wächst höchstens c_1 mal so schnell wie T_2.
$T_1(n)$ wächst mindestens c_2 mal so schnell wie T_2.

Das gilt ggf. erst ab einem bestimmten Punkt, nämlich n_0



Landau-Symbole

Achtung:
O, Ω und Θ sind Mengen!

Das motiviert folgende Definitionen:

O-Notation:

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

$$0 \leq T_1(n) \leq c_1 T_2(n) \Leftrightarrow T_1(n) \in O\big(T_2(n)\big)$$

Ω-Notation:

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

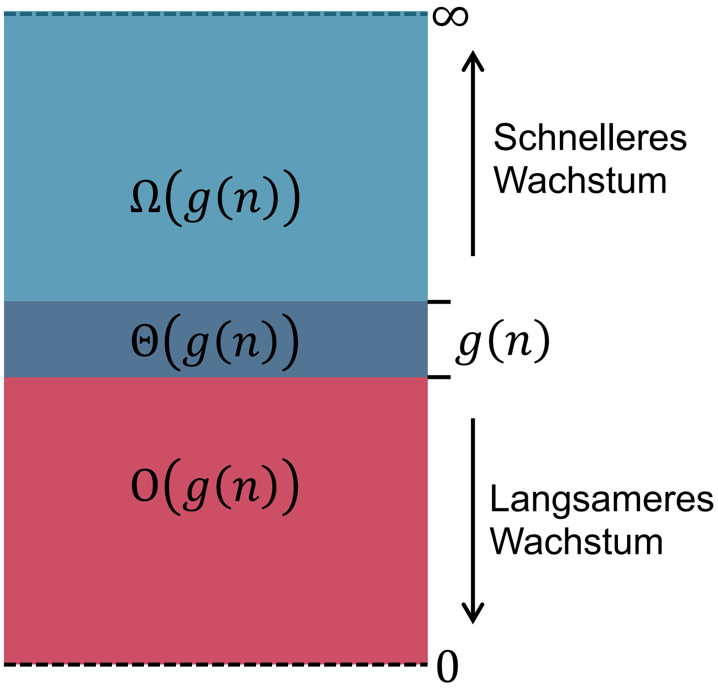
$$T_1(n) \geq c_2 T_2(n) \geq 0 \Leftrightarrow T_1(n) \in \Omega\big(T_2(n)\big)$$

Θ-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:

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

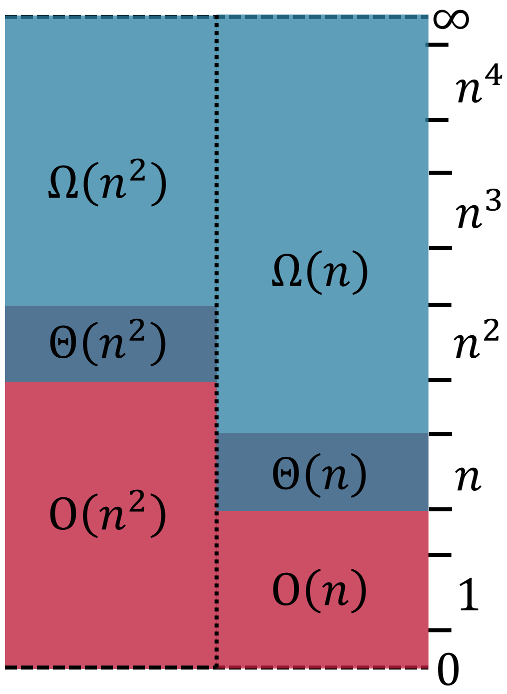
Relationen zwischen Klassen - Eine graphische Darstellung



Wir können sogar Klassen miteinander vergleichen.
Beispiel:

$$\Theta(n^2) \subsetneq \Omega(n)$$

Zum Merken:
Wächst die Funktion schneller, wächst der O-Bereich und der Ω-Bereich schrumpft.



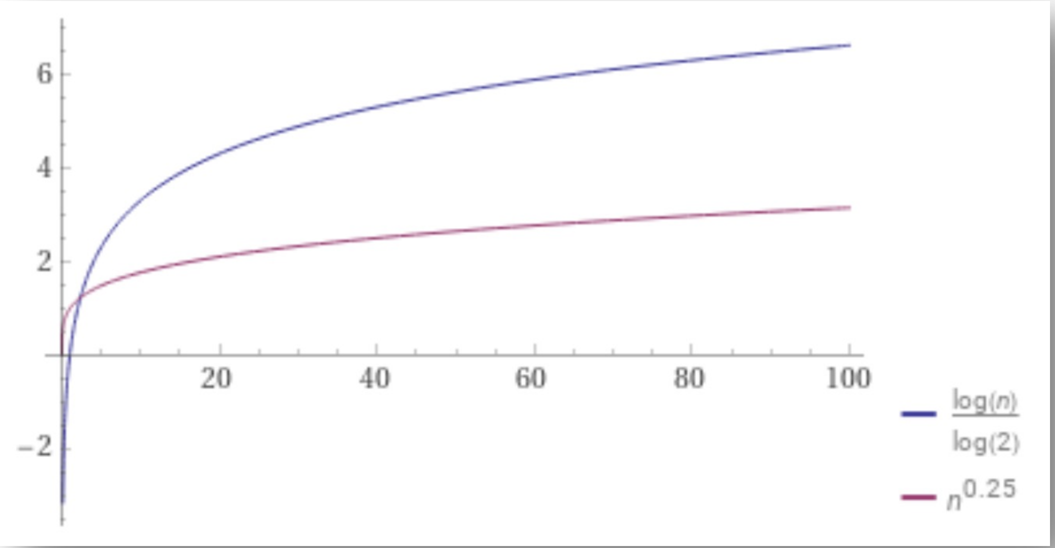
Relationen zwischen Klassen – Eine Tabelle

Bedingung	**Klasse** \ **Klasse**	$O(g(n))$	$\Theta(g(n))$	$\Omega(g(n))$
$f(n) \in o(g(n))$ $\left(o(g(n)) \coloneqq O(g(n)) \setminus \Theta(g(n))\right)$ "Klein-o-Notation"	$O(f(n))$	$\subsetneq$	X	X
	$\Theta(f(n))$	$\subsetneq$	X	X
	$\Omega(f(n))$	X	$\supsetneq$	$\supsetneq$
$f(n) \in \Theta(g(n))$	$O(f(n))$	=	$\supsetneq$	X
	$\Theta(f(n))$	$\subsetneq$	=	$\subsetneq$
	$\Omega(f(n))$	X	$\supsetneq$	=
$f(n) \in \omega(g(n))$ $\left(\omega(g(n)) \coloneqq \Omega(g(n)) \setminus \Theta(g(n))\right)$ "Klein-ω-Notation"	$O(f(n))$	$\supsetneq$	$\supsetneq$	X
	$\Theta(f(n))$	X	X	$\subsetneq$
	$\Omega(f(n))$	X	X	$\subsetneq$

Beispiel Funktionen

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

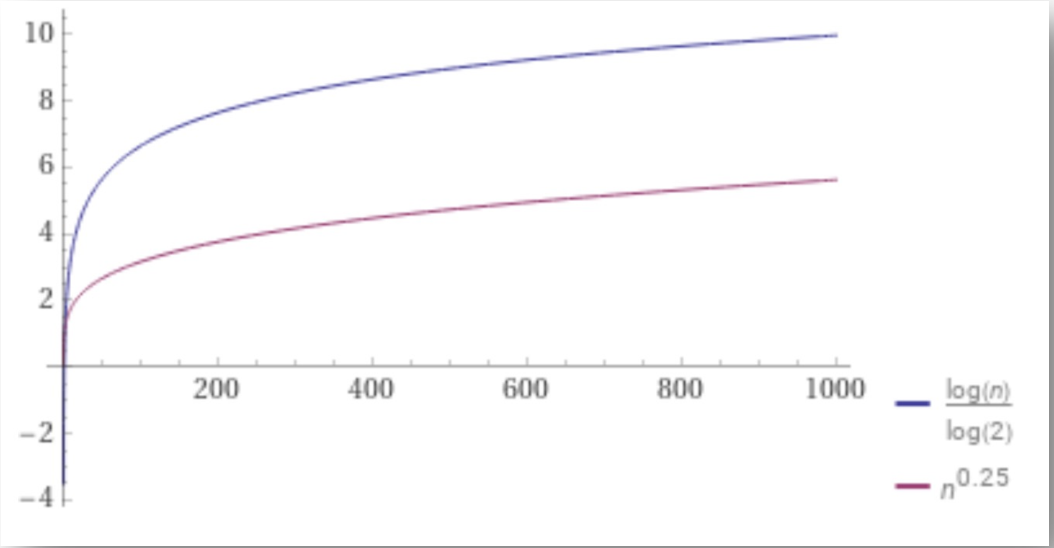
Abbildung rechts zeigt das:
Wähle $c \ = \ 1, n_0 \geq \ 5$



Beispiel Funktionen

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

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



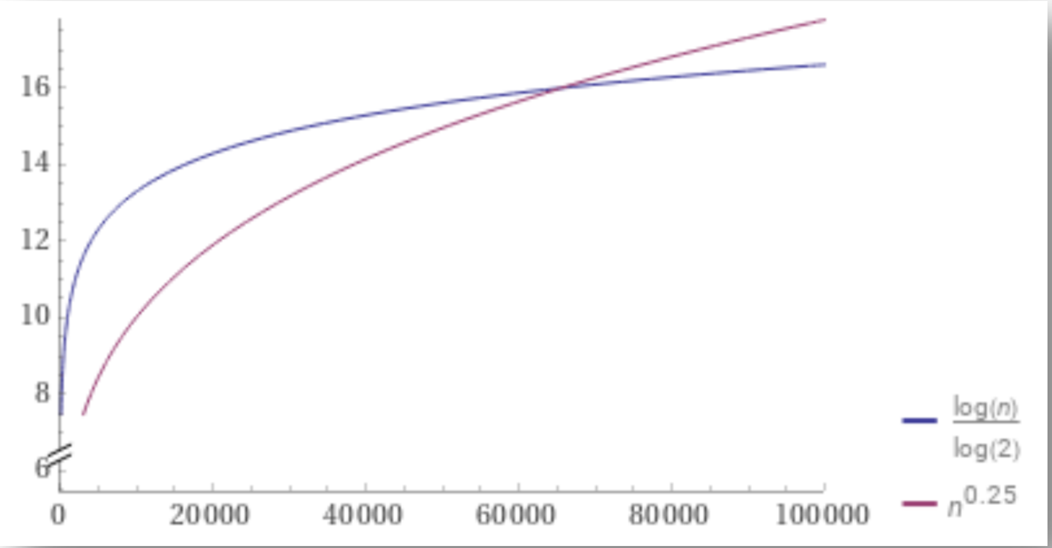
Beispiel Funktionen

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

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

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

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



Beispiele

Beispiel 1

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

$4n^2 + 12n - 15$

Beide Ungleichung 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: $2^n \in O(3^n)$, 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 $\lim_{n\to\infty} \left(\frac{2}{3}\right)^n = 0$, d.h. dieses c_1 kann nicht existieren!

$\Rightarrow$ Aussage widerlegt.

Beispiel 3

Satz 3.1: 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 $\lim\limits_{n \to \infty} \frac{\log_2 n}{n} = 0$, 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 3.1: 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 3.2: Seien $a, b \in \mathbb{R}^+$. Dann gilt $\log_2^a n \in O(n^b)$.

Beweis: Wählen wir die Konstante $c = 1$. Dann soll gelten:

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

Da $a, b \in \mathbb{R}^+$ ist auch $\frac{b}{a} \in \mathbb{R}^+$. Nach Satz 3.1 ist die letzte Ungleichung ab einem n_0 wahr. Da wir Äquivalenzumformungen benutzt haben, können wir die Lösungskette von unten nach oben gehen.