

Task 1

MISZCZUK-STREIT

① 1. Verankerung

$$\forall (n \in \mathbb{N}) \left(\sum_{i=0}^n i = \frac{n(n+1)}{2} \right)$$

$$n=0:$$

$$\sum_{i=0}^0 i = 0 = \frac{0(0+1)}{2} = 0$$

$$\frac{0(1)}{2} = \frac{0}{2} = 0 \quad \checkmark (\checkmark)$$

E

I.H.:
GILT FÜR EIN
BELEBIGES
 $n \in \mathbb{N}$.

1. Ht. 2. Annahme / HYPOTHESE

$n=2:$?!

$$\frac{2(2+1)}{2} = \frac{0+1+2}{3}$$

$$2+1 = 3 \quad \checkmark$$

$$3. \text{ Vererbung} \quad \sum_{i=0}^{n+1} i = \sum_{i=0}^n i + n+1 =$$

$$\frac{n(n+1)}{2} + n+1 = \frac{(n+1)(n+1+1)}{2} \quad ?!$$

$$\frac{n(n+1) + 2n+2}{2} = \frac{(n+1)(n+2)}{2}$$

$$\frac{n^2+n+2n+2}{n^2+3n+2} = \frac{(n+1)(n+2)}{n^2+3n+2} \quad \checkmark$$

(2) 1. Verankerung

$$\forall (n \in \mathbb{N}) \forall (x \in \mathbb{R} \setminus \{1\}) \left(\sum_{i=0}^n x^i = \frac{x^{n+1} - 1}{x - 1} \right)$$

$$n=0 \quad \checkmark$$

$$\frac{x^{0+1} - 1}{x - 1} = 1 = x^0 \quad \checkmark$$

2. Annahme

1/1/2 ?!

$$\frac{x^{2+1} - 1}{x - 1} = \cancel{x^0 + x^1 + x^2} = x^2 + x + 1$$

$$\frac{x^3 - 1}{x - 1} = \frac{(x-1) \cdot (x^2 + x + 1)}{(x-1)} = x^2 + x + 1 \quad \checkmark$$

$$3. \text{ Vererbung} \quad \sum_{i=0}^{n+1} x^i = \sum_{i=0}^n x^i + x^{n+1} = \text{I.H.}$$

$$\frac{x^{n+1} - 1}{x - 1} + x^{n+1} = \frac{x^{(n+1)+1} - 1}{x - 1}$$

$$\frac{(x^{n+1} - 1) + (x^{n+1} \cdot (x-1))}{x - 1} = \frac{(x^{n+2} - 1) + (x^{n+2} - x^{n+1})}{x - 1} =$$

$$= \frac{x^{n+2} - 1}{x - 1} \quad \checkmark \checkmark$$

③ 1. Verankerung

$$\forall (n \in \mathbb{N}) \forall (x \in \mathbb{R}) \left(\sum_{i=0}^n \binom{n}{i} x^i = (1+x)^n \right)$$

$\because n=0:$ ✓

$$(x+1)^0 = 1 \quad = \dots = \sum_{i=0}^0 \binom{0}{i} x^i$$

2. Annahme

/// ??

$$(x+1)^2 = x^2 + 2x + 1 \quad \checkmark$$

3. Vererbung

$$(x+1)^n + \binom{n}{n} x^{n+1} = (x+1)^{n+1}$$

$$(x+1)^n + \binom{n}{n+1} x^{n+1} = (x+1)^n \cdot (x+1)$$

$$(x+1)^n + x^{n+1} = (x+1)^n \cdot (x+1)$$

$$\sum_{i=0}^n \binom{i}{i} x^i + \sum_{i=0}^{n+1} \binom{i}{i} x^i \stackrel{?}{=} \sum_{i=0}^{n+1} \binom{n+1}{i} x^i \quad \square$$

$$= 1 + \underbrace{\sum_{i=1}^n \binom{i}{i} x^i}_{\text{1}} + \underbrace{\sum_{i=1}^{n+1} \binom{i-1}{i-1} x^i}_{\text{1}} + x^{n+1} = 1 + \left(\sum_{i=1}^n \left(\binom{i}{i} + \binom{n}{i-1} \right) x^i \right) + x^{n+1}$$

$$= 1 + \sum_{i=1}^n \binom{n+1}{i} x^i + x^{n+1}$$

$$= \sum_{i=0}^{n+1} \binom{n+1}{i} x^i \quad \checkmark \quad \square$$

BITTE SORGEFALT BEI DEN UNTEREN TEILS ANSTREGEN!

- ① Prove that each of our little notions of asymptotic complexity defines a relation that is not reflexive but transitive

proof for little-o (\circ)

Reflexivität: Per Definition bedeutet \circ , dass die Funktion $f(n)$ langsamer, aber nie gleich wie die Funktion $g(n)$ wächst.

Somit wird die Anforderung für Reflexivität nicht erfüllt
INVOLVIERT NUR EIN FUNKTIONS-SYMBOL!

$$f(n^2) \circ g(n^3) \rightarrow n^2 < c \cdot (n^3) \quad c > 0$$

↳ n^2 wächst immer schneller aber nie gleich, egal welchen Wert man bei c einsetzt.

EMPFEHLUNG:
 $f(n) = 0$ WAHLN

Transitivität:

$$f(n) < c_1 \cdot g(n) \quad \checkmark$$

$$g(n) < c_2 \cdot h(n) \quad \checkmark$$

1) MIT c_1 MULTIPLIZIEREN
2) TRANSITIVITÄT VON $<$ ANWENDEN

$$f(n) < c_1 \cdot c_2 \cdot h(n)$$

WARUM? ↗ BEGRUNDUNG UND ZWISCHENSCHRITTE,
BITTE!

$$f(n) < c \cdot h(n)$$

↳ Somit wird bewiesen, dass $f(n) < c_1 \cdot g(n) < c_2 \cdot h(n)$ und somit $f(n) = o(h(n))$ ist.

proof for little-omega / omicron (ω)

Reflexivität: Per Definition wächst bei ω $f(n)$ immer schneller, aber nie gleichschnell wie $g(n)$.

Somit ist die Anforderung für die Reflexivität nicht erfüllt

$$f(n^2) \circ g(n^2) \rightarrow n^3 > c \cdot (n^2) \quad c > 0$$

↳ n^3 wächst immer schneller wachsen, egal welchen Wert man bei c einsetzt.

Transitivität:

$$f(n) > c_1 \cdot g(n)$$

$$g(n) > c_2 \cdot h(n) \quad (\checkmark) \text{ DITO}$$

$$f(n) > c_1 \cdot c_2 \cdot h(n)$$

$$f(n) > c \cdot h(n)$$

$$\underbrace{f(n) > c_1 \cdot g(n) > c_2 \cdot h(n)}_{\rightarrow f(n) > c_1 \cdot c_2 \cdot h(n)} \rightarrow f(n) = \omega(h(n))$$

② prove that each of our big notions of asymptotic complexity defines a relation that is reflexive and transitive (a so-called preoder)

proof for big-O

Reflexivität: Per definition bedeutet O, dass eine Funktion f schlussendlich langsamer oder gleich schnell wie eine Funktion g wächst.

also gilt $f(n) \in O(f(n))$. O ist reflexiv. ✓
 BITTE ETWAS MEHR FORMALES DETAIL.

Transitivität: O ist transitiv, wenn folgende Aussage bewiesen werden kann:

$$(f(n) = O(g(n)) \wedge g(n) = O(h(n))) \Rightarrow f(n) = O(h(n)) \quad \checkmark$$

(wenn $f(n) = O(g(n))$ ist und $g(n) = O(h(n))$ ist, dann ist $f(n)$ auch $O(h(n))$). ✓

Per Definition von O gilt, dass es positive Konstanten c, n_0 gibt, sodass gilt:

$$f(n) \leq c \cdot g(n) \text{ für } n \geq n_0 \quad \checkmark$$

Daraus kann man für die gegebene Aussage notieren:

$$f(n) \leq c_1 \cdot g(n)$$

$$g(n) \leq c_2 \cdot h(n)$$

$$\therefore f(n) \leq c_1 \cdot c_2 \cdot h(n)$$

{ Diese zwei Zeilen werden kombiniert

Mit c_1 multiplizieren (1)

(2) via Transitivität von \leq

{ $c_1 \cdot c_2$ sind Konstanten und werden zu c.

WICHTIGSTER SCHRITT!

$$\therefore f(n) \leq c \cdot h(n)$$

somit gilt, dass $f(n) = O(h(n))$ ist. ✓

proof for Omega

Reflexivität: Per definition bedeutet Ω , dass eine Funktion f schlussendlich schneller oder gleich schnell wie eine Funktion g wächst.

also gilt $f(n) \in \Omega(f(n))$. Ω ist reflexiv. ✓ DITO

Transitivität: Ω ist transitiv, wenn folgende Aussage bewiesen werden kann:

$$(f(n) = \Omega(g(n)) \wedge g(n) = \Omega(h(n))) \Rightarrow f(n) = \Omega(h(n))$$

(wenn $f(n) = \Omega(g(n))$ ist und $g(n) = \Omega(h(n))$ ist, dann ist $f(n)$ auch $\Omega(h(n))$). ✓

Per Definition von Ω gilt, dass es positive Konstanten c, n_0 gibt, sodass gilt:

$$f(n) \geq c \cdot g(n) \text{ für } n \geq n_0$$

Daraus kann man für die gegebene Aussage notieren:

$$f(n) \geq c_1 \cdot g(n)$$

{ Diese zwei Zeilen werden kombiniert

$$g(n) \geq c_2 \cdot h(n)$$

DITO $\Rightarrow f(n) \geq c_1 \cdot c_2 \cdot h(n)$ { $c_1 \cdot c_2$ sind Konstanten und werden zu c.

$$\Rightarrow f(n) \geq c \cdot h(n)$$

somit gilt, dass $f(n) = \Omega(h(n))$ ist. ✓

proof for big-theta

Reflexivität: Per Definition bedeutet Θ , dass eine Funktion schlussendlich gleich schnell wächst wie eine Funktion g .

y also gilt $f(n) = \Theta(f(n))$ **FORMALEN BITTE!**

Transitivität: Θ ist transitiv, wenn folgende Aussage bewiesen werden kann:

$$(f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n))) \Rightarrow f(n) = \Theta(h(n))$$

(Wenn $f(n) \Theta(g(n))$ ist und $g(n) \Theta(h(n))$ ist, dann ist $f(n)$ auch $\Theta(h(n))$.)

Per Definition von Θ gilt, dass es positive Konstanten c_1, n_0 gibt, sodass gilt:
 $f(n) = c_1 \cdot g(n)$ für $n \geq n_0$

Daraus kann man für die gegebene Aussage notieren:

$$f(n) = c_1 \cdot g(n) \quad |$$

$g(n) = c_2 \cdot h(n) \quad |$ diese zwei Zeilen werden kombiniert.

DITO $\rightarrow f(n) = c_1 \cdot c_2 \cdot h(n) \quad |$ c_1, c_2 sind Konstanten und werden zu c . ①

$$\rightarrow f(n) = c \cdot h(n)$$

Somit gilt, dass $f(n) = \Theta(h(n))$ ist.

④ prove that Θ defines a symmetric (and thus an equivalence) relation.

Definition Symmetrie: $\forall x, y \in M : (x R y \Rightarrow y R x)$ ✓

"Die Symmetrie einer zweistelligen Relation R auf einer Menge ist gegeben, wenn aus $x R y$ stets $y R x$ folgt" - Wikipedia

Zu beweisen ist also, dass gilt: $f(n) = \Theta(g(n)) \Rightarrow g(n) = \Theta(f(n))$ ✓

Per Definition von Theta gibt es drei positive Konstanten c_1, c_2, n_0 , sodass gilt:

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \quad |$$
 für alle $n \geq n_0$ **(HYPOTHESE)**

Bei dieser Gleichung können nun c_1 und c_2 dividiert werden:

$$g(n) \leq \frac{1}{c_1} \cdot f(n) \wedge g(n) \geq \frac{1}{c_2} \cdot f(n) \quad | :-)$$

Anderer sortiert also:

$$\left(\frac{1}{c_2}\right) \cdot f(n) \leq g(n) \leq \left(\frac{1}{c_1}\right) \cdot f(n) \quad |$$

c_1 und c_2 sind positive Konstanten, deshalb sind $\frac{1}{c_1}$ und $\frac{1}{c_2}$ ebenfalls Konstanten und können gemäß der Big-O-/Omega-Notation vernachlässigt werden. ✓

Somit gilt $g(n) = \Theta(f(n))$ ✓ **(KONKLUSION)**

Jetzt könnte noch bewiesen werden, dass auch gilt: $g(n) = \Theta(f(n)) \Rightarrow f(n) = \Theta(g(n))$
 Dieser Beweis würde analog zum ersten Beweis durchgeführt, ausser dass "f" und "g" getauscht werden können. Da der erste Beweis stimmt, wird auch der zweite stimmen.

Somit gilt auch $f(n) = \Theta(g(n))$ ✓

**NICHT NÖTIG ZU
ZEIGEN WEGEN ✓**

NUN SO: ANPRÜFUNGEN HABEN SIE KEINE ZEIT
 ① Definitionen: ZEIT ZUM SORGFÄLTIGEN ENTWICKELN NUTZEN! $c, d \in \mathbb{N}_{\geq 0}$

$$\begin{aligned} T_1 & \left\{ \begin{array}{l} c_3 \\ T_1(\lfloor n/2 \rfloor) + i \cdot c + d \end{array} \right. \begin{array}{l} \text{otherwise} \\ \text{if } n=1 \end{array} & \text{STRUKTUR,} \\ & \downarrow T_1 \left\{ \begin{array}{l} c_3 \\ T_1(\lfloor n/2 \rfloor) + 1 \cdot c + d \end{array} \right. \begin{array}{l} \text{otherwise} \\ \text{if } n=1 \end{array} & \text{BEGRÜNDUNGEN} \\ & \downarrow T_2 \left\{ \begin{array}{l} c_3 \\ T_2(\lfloor n/2 \rfloor) + 2c + d \end{array} \right. \begin{array}{l} \text{otherwise} \\ \text{if } n=1 \end{array} & \end{aligned}$$

$$\begin{aligned} T_q & \left\{ \begin{array}{l} c_3 \\ T_q(\lfloor n/2 \rfloor) + c + d \end{array} \right. \begin{array}{l} \text{if } n=1 \\ \text{if } n \text{ is even} \end{array} & \\ & \quad \left. \begin{array}{l} T_q(\lfloor n/2 \rfloor) + 2c + d \\ \text{otherwise } (n \text{ is odd}) \end{array} \right. & \end{aligned}$$

Annahme:

$$T_1(n) \leq T_q(n) \leq T_2(n)$$

Trivialer Fall: (Verankerung)

$$\underline{n=1} \checkmark$$

$$T_1(n) = c_3$$

$$\underline{n=1} \checkmark \quad T_2(n) = c_3$$

$$T_q(n) = c_3$$

$$\text{darauf folgt: } \underline{T_1 = T_q = T_2}$$

somit ist die Annahme wahr.

$$\underline{n=2}$$

$$T_1(n) = T(\lfloor 2/2 \rfloor) + 1 \cdot c + d$$

$$\underline{T_1 = T_q \leq T_2}$$

$$T_2(n) = T(\lfloor 2/2 \rfloor) + 2 \cdot c + d$$

$$T_q(n) = T(\lfloor 2/2 \rfloor) + 1 \cdot c + d$$

Case 2

$$\underline{n=3}$$

$$T_1(n) = T(\lfloor 3/2 \rfloor) + 2 \cdot c + d$$

$$\underline{T_1 < T_q = T_2}$$

$$\therefore T_2(n) = T(\lfloor 3/2 \rfloor) + 2 \cdot c + d$$

$$T_q(n) = T(\lfloor 3/2 \rfloor) + 2 \cdot c + d$$

Vererbung

(jeweils mit $n+2$, damit n jeweils gerade/
ungerade bleibt)

DIE INDIZES FEHLEN :-c BITTE VORSICHTIG SEIN!

$$T_1(n+2) = T_1(\lfloor (n+2)/2 \rfloor) + c + d$$

?!

$$= T_1(\lfloor (n/2) + 1 \rfloor) + 1c + d$$

$$T_2(n+2) = T_2(\lfloor (n/2) + 1 \rfloor) + 2c + d$$

$$T_4(n+2) = T_4(\lfloor (n/2) + 1 \rfloor) + 1 \cdot c + d$$

$$\underline{\underline{T_1 = T_2 < T_4}}$$

$$T_1(n+2) = T_1(\lfloor (n+2)/2 \rfloor) + c + d$$

$$T_2(n+2) = T_2(\lfloor (n+2)/2 \rfloor) + 2c + d$$

$$T_4(n+2) = T_4(\lfloor (n+2)/2 \rfloor) + 2c + d$$

$$\underline{\underline{T_1 < T_4 = T_2}}$$

Alle Fälle sind wahr und entsprechen der Aussage!

Fortsetzung:

$$\underline{n > 1}$$

$$\cancel{n+1} = 2k \quad (\cancel{n+1} \text{ ist gerade})$$

$$k \in \mathbb{N}_{\geq 1}$$

$$T_1(n) = (\lfloor n/2 \rfloor) + 1 \cdot c + d$$

$$T_2(n) = (\lfloor n/2 \rfloor) + 2 \cdot c + d \quad \Rightarrow$$

$$T_q(n) = (\lfloor n/2 \rfloor) + c + d$$

$$\text{daraus folgt: } T_1 = T_q < T_2$$

somit ist auch da die Annahme wahr

$$\checkmark \quad \cancel{n+1} = 2k+1 \quad (\cancel{n+1} \text{ ist ungerade})$$

$$T_1 = (\lfloor n/2 \rfloor) + 1 \cdot c + d$$

$$T_2 = (\lfloor n/2 \rfloor) + 2 \cdot c + d$$

$$T_q = (\lfloor n/2 \rfloor) + 2c + d \quad \Rightarrow$$

$$\text{daraus folgt: } T_1 < T_q = T_2$$

auch in diesem Fall ist die Annahme wahr.

Da in allen Fällen die Annahme wahr ist,-
ist $T_1 \leq T_q \leq T_2$ bewiesen.

② prove that $T_i(n)$ is $\Theta(\log n)$ with the Master Theorem

Master theorem:

$$T_h(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ \Theta(T_h(n/b)) + f(n) & \text{otherwise} \end{cases}$$

a ist die Anzahl Subprobleme

n/b ist die Größe jedes Subproblems

c ist die maximale Größe für Subprobleme

$f(n)$ ist $\Theta(n^e (\log(n))^k)$ für Parameter $e, k \in \mathbb{N}_{\geq 0}$

\hookrightarrow ist der Preis, den es kostet, das Problem in Subprobleme zu zerlegen und wieder zusammenzubringen.

$$T_h(n) = \begin{cases} \Theta(n \log_b(a)) & \text{if } e < \log_b(a) \\ \Theta(f(n) \cdot \log(n)) & \text{if } e = \log_b(a) \\ \Theta(f(n)) & \text{if } e > \log_b(a) \end{cases} \quad \left. \begin{array}{l} \text{häufiger Fall} \\ \text{wenn } c=1 \end{array} \right\}$$

Zu untersuchende Funktion g :

(noch Umformung zu $T_i(n)$)

$$T_i(n) = \begin{cases} c s & \text{if } n=1 \\ T_i(n/b) + i \cdot c + d & \text{otherwise} \end{cases}$$

$\downarrow a=1$ $\downarrow b=2$ $\downarrow = f(n)$
konstant ✓

$$\rightarrow \log_b(a) = \log_2(1) = 0 \quad \checkmark$$

Man nehme obige Definition: $f(n)$ ist $\Theta(n^e (\log(n))^k)$. Da $f(n)$ hier $i \cdot c + d$ entspricht und somit konstant ist, muss auch Θ konstant sein. Wählt man für die Parameter $e, k \in \mathbb{N}_{\geq 0}$ den Wert 0. Ist dies der Fall?

Somit trifft der 2. Fall des Master Theorems zu: $e = \log_2 a \rightarrow 0 = 0$ ✓
Also ist $T_i(n) = \Theta(f(n) \cdot \log(n))$. $f(n)$ ist aber eine Konstante und kann somit gemäß Big-Omega-Notation vernachlässigt werden. ✓

→ Es gilt $T_i(n) = \Theta(\log(n))$. ✓

(3) conclude that $T_0(n)$ is $\Theta(\log(n))$ with a justification

gemäss Task 1 und 2 bewiesen gilt:

$$T_1(n) \leq T_0(n) \leq T_2(n) \quad \checkmark \quad \text{und}$$

$T_1(n)$ ist $\Theta(\log(n))$ ✓ und somit (neu)

→ $T_1(n)$ ist $\Theta(\log(n))$ und $T_2(n)$ ist $\Theta(\log(n))$ ✓

fügt man die erste und die dritte Erkenntnis zusammen:

$$\Theta(\log(n)) \leq T_0(n) \leq \Theta(\log(n)) \quad \checkmark$$

muss gemäss der Definition von Theta auch $T_0(n)$ $\Theta(\log(n))$ sein. ✓

Algorithmen und Datenstrukturen – Graded homework 2

1.1) prove that $\log_b(\cdot)$ is $\Theta(\log(\cdot))$

$\Theta(\text{theta})$: Die Funktion $f: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ ist asymptotisch ab einem gewissen Zeitpunkt gleichwertig zur Funktion $g: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ (sprich " $f(n)$ ist $\Theta(g(n))$ "), falls, und nur falls: $f(n) \in \Theta(g(n)) \cap \Omega(g(n))$ ist.

→ a und b sind Konstanten die vernachlässigt werden können, da sie zur Komplexität des Algorithmus nichts beitragen. x wiederum ist eine Variable und darf nicht vernachlässigt werden.

$$\log_b(x) = \frac{\log_b(x)}{\log_b(a)} = \frac{1}{\log_b(a)} \cdot \log_b(x)$$

[Ein Basiswechsel an $\log_a(x)$ wird vorgenommen.]

[durch das Umfassen des Terms wird $\frac{1}{\log_b(a)}$ konstant und kann vernachlässigt werden.]

$$\rightarrow \log_b(x) = \Theta(\log_b(x))$$

FÜR PRÜFUNGEN ETWAS MEHR MATHEMATISCHEM DETAIL, BITTE.

1.2) prove that if $c_1 n^c$ is $\Theta(c_3 n^{c_4})$ then $c_2 = c_4$

→ c_1 und c_3 sind Konstanten die gemäß der big-O/omega Notation vernachlässigt werden können. Schrift:

$$n^{c_2} \text{ ist } \Theta(n^{c_4}) \quad \checkmark$$

$$\rightarrow \text{Da } \Theta(n^2) \neq \Theta(n^3), \text{ muss hier } c_2 = c_4 \text{ sein.} \quad \checkmark$$

} ETWAS KURZ

2: . verify the increasing ordering of the following ACs:

$\Theta(1)$ is constant asymptotic complexity (AC)

$\Theta(\log(n))$ is logarithmic AC

$\Theta(\sqrt{n})$ is sub-linear AC

$\Theta(n)$ is linear AC

$\Theta(n\log(n))$ is sub-quadratic AC

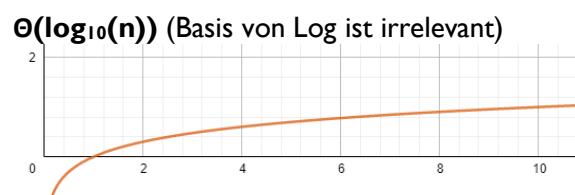
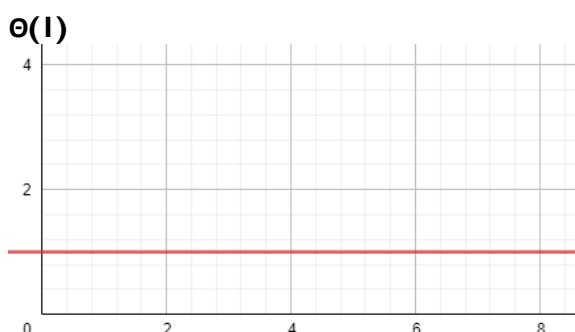
$\Theta(n^2)$ is quadratic AC

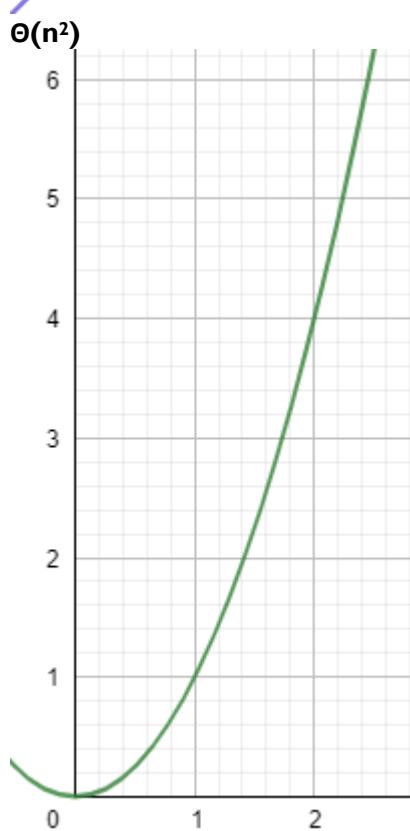
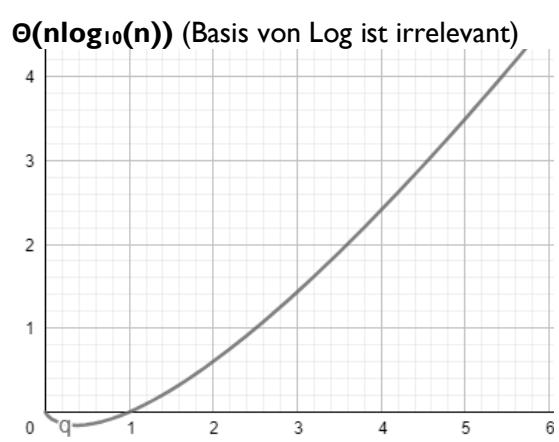
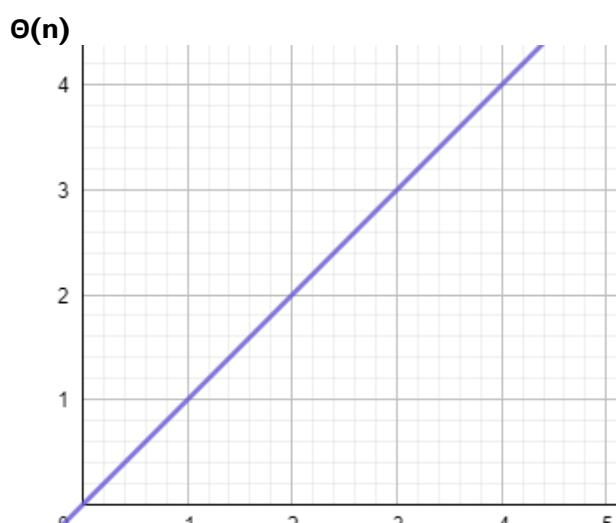
$\Theta(n^3)$ is cubic AC

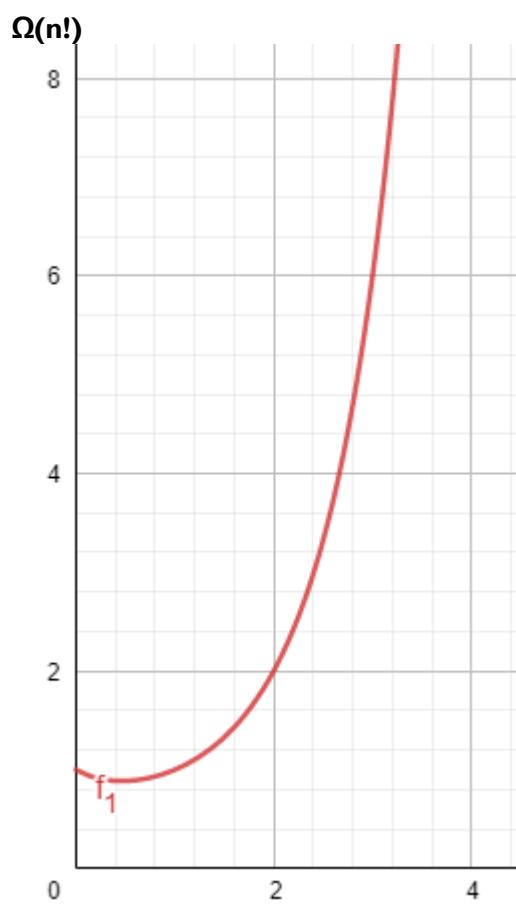
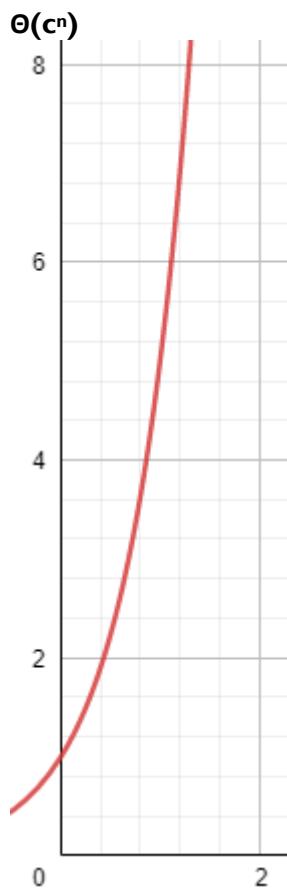
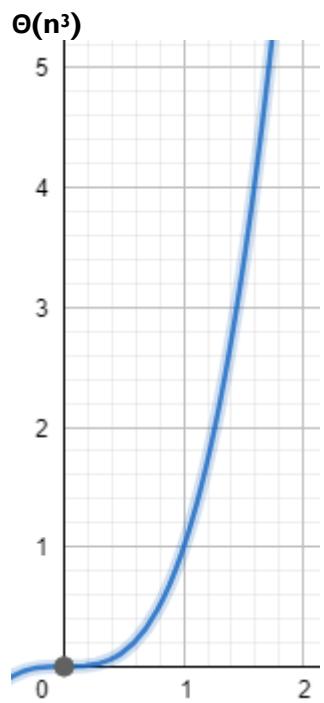
$\Theta(c^n)$ is exponential AC

$\Omega(n!)$ is super-exponential AC (ex.: $\Theta(n^n)$)

Grafisch dargestellt sehen die verschiedenen Terme folgendermassen aus: \checkmark



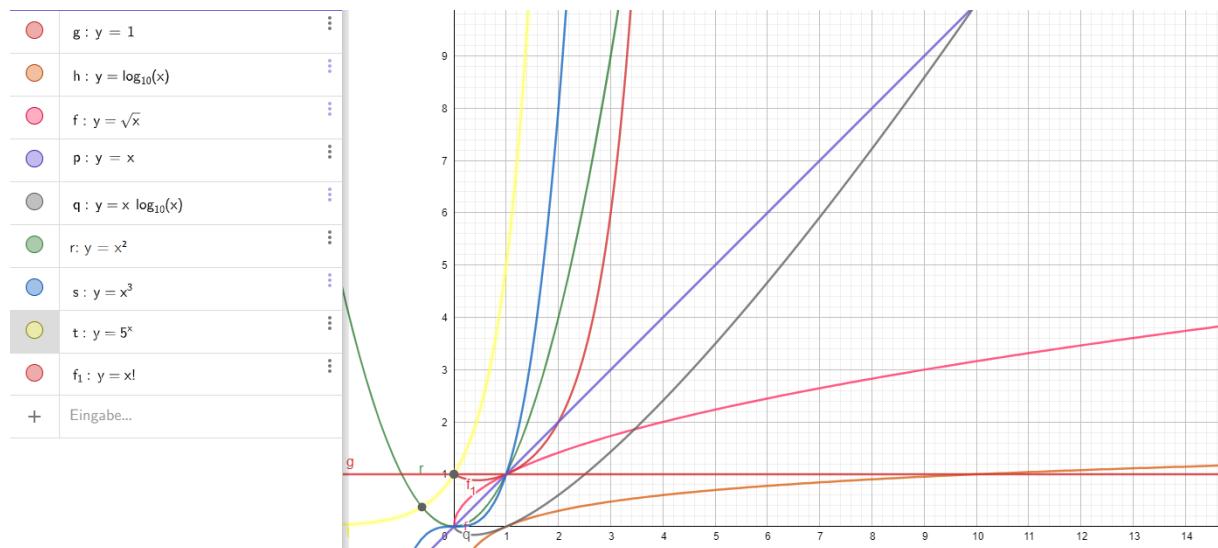




STREIT, MISZCZUK

Die obigen Abbildungen sind der vorgegebenen Reihenfolge nach aufgeführt. Es ist zu erkennen, dass die Steigung immer wie steiler wird. Dabei ist eine flachere Steigung insofern «besser» bei einem Algorithmus als eine steile, als bei einer flacheren Steigung weniger Rechenpower benötigt wird. ✓

Folgende Grafik stellt alle Terme im gleichen Koordinatensystem dar. Dabei wurden die Farben angepasst, um die verschiedenen Graphen gemäss Legende unterscheiden zu können. Sämtliche Graphen wurden mithilfe von Geogebra.org geplotted. ✓



Aus Gründen des Zooms ist auf der Grafik nicht mehr zu erkennen, dass $\Theta(n \log_{10}(n))$ $\Theta(n)$ schlussendlich kreuzt und somit steiler wird. $\Omega(n!)$ kreuzt nach und nach $\Theta(n^2)$, $\Theta(n^3)$ und $\Theta(c^n)$. Somit ist die vorgegebene Reihenfolge schlussendlich korrekt. ✓

**LOGARITHMISCHE SKALEN
VERWENDEN ; -)**

3) What are the ACs of the summation laws? *

1) arithmetic sum

$$\forall (n \in \mathbb{N}) \left(\sum_{i=0}^n i = \frac{n(n+1)}{2} \right)$$

$$\frac{n(n+1)}{2} = \frac{n^2 + n}{2} = n^2 - \frac{1}{2}$$

$\frac{1}{2}$ ist konstant und fällt somit auch weg ✓
 $n^2 > n$, n kann also vernachlässigt werden ✓

→ übrig bleibt n^2 , die AC davon ist also die quadratische AC. ✓

2) geometric sum

$$\forall (n \in \mathbb{N}) \forall (x \in \mathbb{R} \setminus \{-1\}) \left(\sum_{i=0}^n x^i = \frac{x^{n+1} - 1}{x - 1} \right)$$

$$\frac{x^{n+1} - 1}{x - 1} = x^n \cdot \frac{x - 1}{x - 1} = x^n \cdot 1$$

$x^n > x$, also kann x vernachlässigt werden
 x konstantenfallen weg ✓

GINFACHTER ;=)

x^n VARIABLE
 $\frac{1}{x-1}$ KONSTANT RELATIV ZU
 $x-1$ KONSTANTE (ABSOLUT)

→ x^n bleibt übrig, dies entspricht dem exponentiellen AC ✓

* bei der Aufgabe 3 werden die Gleichungen betrachtet, nicht die Summenterme

3) monomial law

$$\forall (n \in \mathbb{N}) \forall (x \in \mathbb{R}) \left(\sum_{i=0}^n (n)_i x^i = (1+x)^n \right)$$

$$(1+x)^n = x^n$$

konstante fällt weg ✓

→ übrig bleibt wieder x^n , die AC ist also exponentiell ✓

BEGRÜNDUNGEN, ORDNUNG, SAWBERKET :)

STREIT, MISZCZUK

GRADED HOMEWORK 5

$$\begin{aligned}
 h(m) &= \log_2(g(m)) = \log_2(f(2^m)) = \log_2\left(2^m \cdot f\left(\frac{2^m}{2}\right)\right) \quad | \text{Log-Gesetz für Multiplikationen} \\
 &= \log_2(2^m) + \log_2\left(f\left(\frac{2^m}{2}\right)\right) \quad | \text{Log-Gesetz für Exponenten} \\
 &= m \log_2(2) + 2 \log_2\left(f\left(\frac{2^m}{2}\right)\right) \quad | \log_2(2) = 1 \\
 &= m + 2 \log_2\left(f\left(\frac{2^m}{2}\right)\right) \quad | \left(\frac{2^m}{2}\right) = 2^{m-1}, f() "auspacken" \\
 &= m + 2 \log_2\left(2^{m-1} \cdot f\left(\frac{2^{m-1}}{2}\right)\right) \quad | \text{Log-Gesetz für Exponenten, Log-Gesetz für Multiplikationen} \\
 &\xrightarrow{\text{BITTE WEITER ENTWICKELN}} = m + 2(m-1) \log_2(2) + 2 \log_2\left(f\left(\frac{2^{m-2}}{2}\right)\right) \quad | \log_2(2) = 1 \\
 &= m + 2(m-1) + 4 \log_2\left(f\left(\frac{2^{m-2}}{2}\right)\right) \quad | \text{VIEL ANSEHT}
 \end{aligned}$$

→ Es wird ersichtlich, dass diese Gleichung so in das Equation Schema der inhomogeneous linear recurrence passt:

$$2^0(m-c) + 2^1(m-1) + 2^2(m-2) + \dots + 2^c(m-c)$$

2) 1. $h(m) = \sum_{i=0}^{c=m} 2^i(m-i)$

**BITTE INSTRUKTIONEN BEFOLGEN
(NICHT WEGEN AUTORITÄT SONDERN GELINGEN!)**

2. Die einzige Konstante, deren Wert bestimmt werden kann, ist $2^c(m-i)$ wenn $i=c=m$ ist: $2^i \cdot 0 = 0$. Dies ist somit auch der base case.

3. gemäß Definition: $h(m) = \log_2(g(m))$

$$= \sum_{i=0}^{c=m} 2^i(m-i) = g(m)$$

END RESULTAT:

$$f(n) = \frac{2^n}{4n^3}$$

NICHT n !

4. gemäß Definition: $g(m) = f(2^m)$

$$= \sum_{i=0}^{c=m} 2^i(m-i) = f(m)$$



Graded Homework 6

Task 1

Task 1

prove this matrix equality by induction on n

Useful Definitions: :-)

$F(0) = 0$
 $F(1) = 1$
 $F(n+2) = F(n+1) + F(n+0)$

1. base case
2. base case
recursive case

verankerung mit $n=0$:

$$\begin{bmatrix} F(0+0) & F(0+1) \\ F(0+1) & F(0+2) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & F(0+1)+F(0+0) \end{bmatrix}^{0+1} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^1 \quad \checkmark$$

I.V. Induktionsvoraussetzung (gem. Folie S. 199):

$$\forall n \in \mathbb{N} \quad \left(\begin{bmatrix} F(n+0) & F(n+1) \\ F(n+1) & F(n+2) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n+1} \right) \quad \text{DA'S IST ZU BEWEISEN, KANN ALSO KEINE VORAUSSETZUNG SEIN!}$$

Iteration

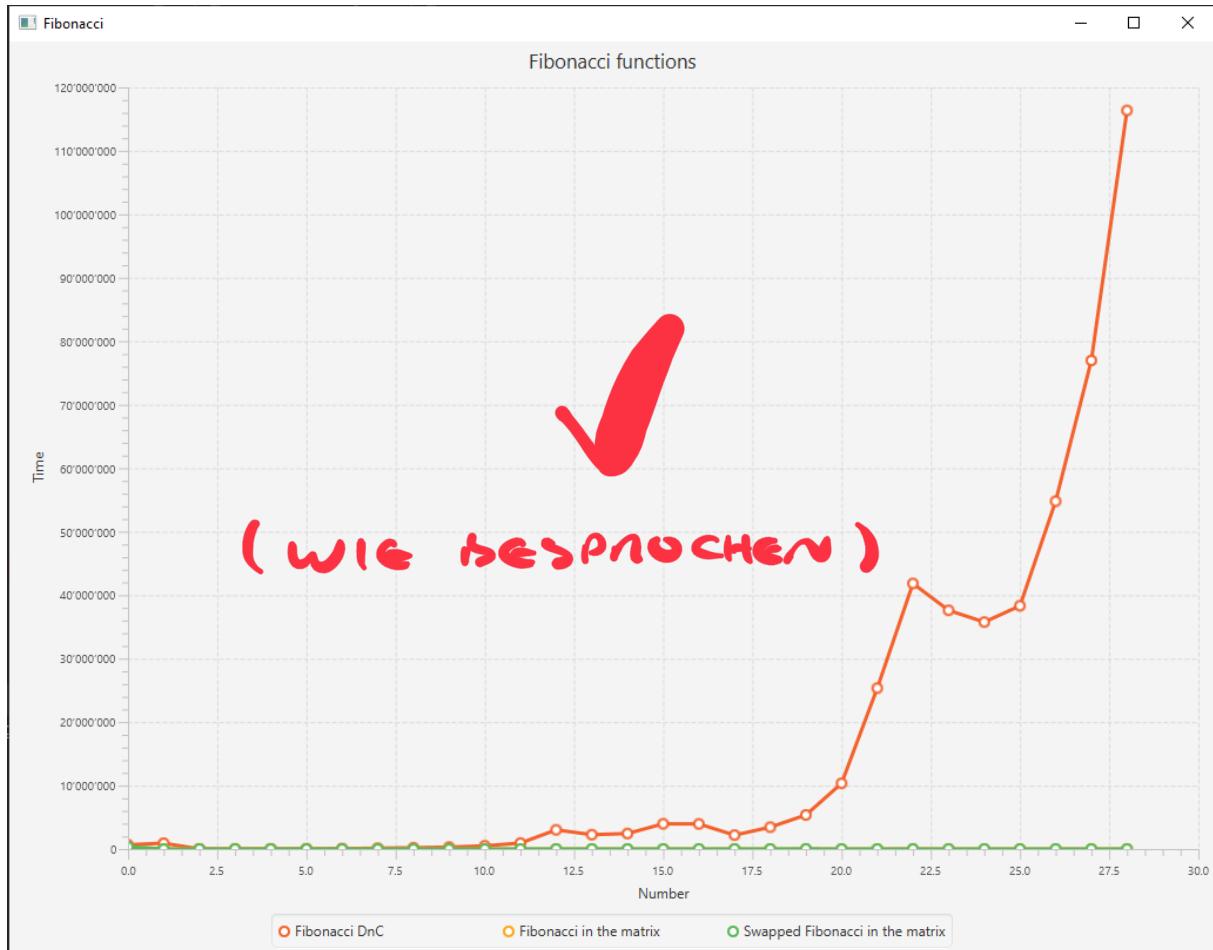
Vererbung mit $n+1$:

$$\begin{bmatrix} 0 & 1 \end{bmatrix}^{(n+1)+1} = \underbrace{\begin{bmatrix} 0 & 1 \end{bmatrix}^{n+1}}_{\text{Iteration gem. Induktionsvoraussetzung}} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} F(n+0) & F(n+1) \\ F(n+1) & F(n+2) \end{bmatrix}}_{\text{DEFINITION DES POTENZIERENS}} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \cdot F(n+0) + 1 \cdot F(n+1) & 0 \cdot F(n+1) + 1 \cdot F(n+2) \\ 1 \cdot F(n+0) + 1 \cdot F(n+1) & 1 \cdot F(n+1) + 1 \cdot F(n+2) \end{bmatrix} = \begin{bmatrix} F(n+1)+0 & F(n+1)+1 \\ F(n+1)+1 & F(n+1)+2 \end{bmatrix}$$

? entspricht der linken Seite der Induktionsvoraussetzung
 \rightarrow Induktion erfolgreich

Task 3



In der obigen Abbildung sind die Algorithmen «Fibonacci DnC», «Fibonacci in the matrix» und «Swapped Fibonacci in the matrix» zu sehen. Swapped bezieht sich hierbei darauf, dass die Arrayrepräsentation der Matrix um Reihen und Spalten vertauscht wurde. Es sind nur zwei Linien zu sehen, da sich «Fibonacci in the Matrix» hinter «Swapped Fibonacci in the matrix» versteckt. Würde man beim Beginn der Linie hineinzoomen, würde man auch noch die gelbe Farbe erkennen. Der kleine Hügel am Anfang kann damit begründet werden, dass die jeweilige Klasse zum ersten Mal initialisiert wird.

Die beiden Matrix-Algorithmen haben eine Zeitkomplexität von $O(n)$, wie anhand der Grafik zu erkennen ist. Die ausschlaggebenden Zeitkomplexitätsfunktionen ($O(\log n)$, $O(1)$, $O(n)$, ...) wurden weggelassen, um das Diagramm nicht zu überfüllen. $O(n)$ wäre eine Diagonale von der unteren linken zur oberen rechten Ecke.