

Aufgabe 1.1

Beschreibe möglichst präzise, was ein Algorithmus ist.

Aufgabe 1.2

Beschreibe die unten stehende Folge von Anweisungen einen Algorithmus? Begründe.

- (1) Input: Zahl $n \in \mathbb{N}$
- (2) Gilt $n < 100$?
 Falls ja: gehe zu (3)
 Sonst: gehe zu (5)
- (3) Setze $(n + 1) \rightarrow n$
- (4) Gehe zu (2)
- (5) Ende

Aufgabe 1.3

Beschreibe die unten stehende Folge von Anweisungen einen Algorithmus? Begründe.

- (1) Input: Liste L mit 9 natürlichen Zahlen
- (2) Wähle ein Zahl a aus der Liste L
- (3) Output(a)
- (4) Ende

Aufgabe 1.4

Beschreibe die unten stehende Folge von Anweisungen einen Algorithmus? Begründe.

- (1) Input: Zahl $n \in \mathbb{N}$
- (2) $0 \rightarrow m$
- (3) Ist $m < n$?
 Falls ja: gehe zu (4)
 Sonst: gehe zu (7)
- (4) Output(n/m)
- (5) $(m + 1) \rightarrow m$
- (6) Gehe zu (2)
- (7) Ende

Aufgabe 1.5

Gegeben ist eine Folge von Anweisungen.

- (1) Input: Zahl $n \in \mathbb{N}$
- (2) Output(n) (d. h. schreibe n auf)
- (3) Gilt $n = 1$?
Falls ja: gehe zu (6)
Sonst: gehe zu (4)
- (4) Ist die Zahl n gerade?
(4.1) Falls ja, setze $n/2 \rightarrow n$
(4.2) Sonst setze $(3n + 1) \rightarrow n$
- (5) Gehe zu (3)
- (6) Ende

- (a) Führe diese Anweisungen mit dem Input $n = 10$ aus. Gelangt man zum Ende?
- (b) Führe diese Anweisungen mit dem Input $n = 7$ aus. Gelangt man zum Ende?
- (c) *Bonusfrage*: Beschreibt die Anweisungsfolge einen Algorithmus? Begründe.

Aufgabe 2.1

Gib die korrekte englische Übersetzung von „grösster gemeinsamer Teiler“ sowie die zugehörige (englische) Abkürzung an.

Aufgabe 2.2

Zeige schrittweise, wie der im Unterricht besprochene klassische Algorithmus von Euklid den grössten gemeinsamen Teiler $\text{ggT}(28, 16)$ berechnet.

Aufgabe 2.3

Zeige am Beispiel von $\text{ggT}(6, -4)$, warum der im Unterricht besprochene klassische Algorithmus von Euklid zur Berechnung des grössten gemeinsamen Teilers nicht funktioniert, wenn einer der beiden Argumente negativ ist?

Aufgabe 2.4

Beschreibe den zentralen Nachteil des klassischen Algorithmus von Euklid gegenüber der modernen Version anhand eines gut gewählten Beispiels.

Aufgabe 2.5

Vervollständige die Python-Funktion, die mit dem klassischen Algorithmus von Euklid den ggT der beiden ganzen Zahlen a und b berechnet und als Wert zurückgibt.

[illegible]

Aufgabe 2.6

Zeige schrittweise, wie der im Unterricht besprochene moderne Algorithmus von Euklid den grössten gemeinsamen Teiler $\text{ggT}(33, 12)$ berechnet.

Aufgabe 2.7

Vervollständige die Python-Funktion, die mit dem modernen Algorithmus von Euklid den grössten gemeinsamen Teiler der beiden ganzen Zahlen a und b berechnet und als Wert zurückgibt.

[illegible]

Aufgabe 2.8

Beschreibe, für welche Eingaben beim (modernen) Algorithmus von Euklid der Worst Case auftritt.

Aufgabe 2.9

Schreibe die ersten 10 Werte der Fibonacci-Folge auf.

Aufgabe 3.1

Ordne die Algorithmen aufgrund ihrer Laufzeitkomplexität mit dem Symbol $<$ in aufsteigender Reihenfolge.

- Algorithmus A : $O(2^n)$
- Algorithmus B : $O(\log_2 n)$
- Algorithmus C : $O(n^2)$
- Algorithmus D : $O(n!)$
- Algorithmus E : $O(\sqrt{n})$

Aufgabe 3.2

Vereinfache die Laufzeitkomplexitäten.

- (a) $O(2^{n+1})$
- (b) $O(\sqrt{4n})$
- (c) $O((n^2 + n)(n + 2)(n + 3))$
- (d) $O(3)$

Aufgabe 3.3

Bestimme die Komplexitätsklasse $O(?)$ der folgenden Python-Funktion in Abhängigkeit der Inputgrösse $n \in \mathbb{N}$.

Annahme: Alle elementaren Operationen haben gleiche konstante Kosten.

```
1 def funktion(n):  
2     s = 0  
3     for i in range(0, n):  
4         s = s + i  
5     return s
```

Aufgabe 3.4

Bestimme die Komplexitätsklasse $O(?)$ der folgenden Python-Funktion in Abhängigkeit der Inputgrösse $n \in \mathbb{N}$.

Annahme: Alle elementaren Operationen haben gleiche konstante Kosten.

```
1 def funktion(n):  
2     s = 0  
3     for i in range(0, n):  
4         for j in range(0, i):  
5             s = s + j  
6     return s
```

Aufgabe 3.5

Bestimme die Komplexitätsklasse $O(?)$ der folgenden Python-Funktion in Abhängigkeit der Inputgrösse $n \in \mathbb{N}$.

Annahme: Alle elementaren Operationen haben gleiche konstante Kosten.

```
1 def funktion(n):  
2     s = 0  
3     i = 1  
4     while i < n:  
5         s = s + i;  
6         i = 3*i  
7     return n
```

Aufgabe 3.6

Bestimme die Komplexitätsklasse $O(?)$ der folgenden Python-Funktion in Abhängigkeit der Inputgrösse $n \in \mathbb{N}$.

Annahme: Alle elementaren Operationen haben gleiche konstante Kosten.

```
1 def funktion(n):  
2     s = 0  
3     for i in range(0, n):  
4         s = s + i  
5     for j in range(0, n):  
6         s = s * j  
7     return s
```

Aufgabe 3.7

Ein Programm, das einen Algorithmus mit der Laufzeitkomplexität $O(n^2)$ implementiert, benötigt im schlimmsten Fall 40 Sekunden für die Verarbeitung von 10 000 Objekten vergleichbarer Grösse.

Welche maximale Laufzeit benötigt das Programm auf der gleichen Hardware für die Verarbeitung von 30 000 Objekten vergleichbarer Grösse?

Aufgabe 3.8

Ein Programm, das einen Algorithmus mit der Laufzeitkomplexität $O(\log n)$ implementiert, benötigt im Worst Case 8 Minuten für die Verarbeitung von 10^4 Objekten vergleichbarer Grösse.

Das Programm soll 10^{12} Objekte von vergleichbarer Grösse auf der gleichen Hardware verarbeiten. Schätze die maximale Laufzeit ab.

Aufgabe 3.9

Ein Computerprogramm in der Komplexitätsklasse $O(n)$ benötigt für die Verarbeitung von $6 \cdot 10^7$ Objekten im schlimmsten Fall etwa 4 Sekunden.

Welche maximale Laufzeit benötigt das Programm auf der gleichen Hardware für die Verarbeitung von $9 \cdot 10^7$ Objekten vergleichbarer Grösse?

Aufgabe 3.10

Ein Programm implementiert einen Algorithmus der Komplexitätsklasse $O(n \log n)$. Damit lässt sich ein Problem der Grösse $n = 200$ Worst Case in 3 Millisekunden verarbeiten. Welche ungefähre maximale Laufzeit benötigt dasselbe Programm auf derselben Hardware für ein Problem der Grösse $n = 40\,000$?

Aufgabe 3.11

Ein Programm implementiert einen Algorithmus der Komplexitätsklasse $O(2^n)$. Damit lässt sich ein Problem der Grösse $n = 40$ in etwa 6 Stunden verarbeiten. Welche ungefähre maximale Laufzeit benötigt dasselbe Programm auf derselben Hardware für ein Problem der Grösse $n = 45$?

Aufgabe 3.12

Ein Programm implementiert einen Algorithmus der Komplexitätsklasse $O(n!)$. Damit lässt sich ein Problem der Grösse $n = 7$ im schlimmsten Fall in 10 Sekunden verarbeiten. Welche ungefähre maximale Laufzeit benötigt dasselbe Programm auf derselben Hardware für ein Problem der Grösse $n = 9$?

Aufgabe 3.13

In welche Laufzeitklasse gehören die folgenden Algorithmen?

- (a) Gnomesort
- (b) Suchen in einer unsortierten Liste
- (c) Mergesort
- (d) Lesen eines bestimmten Elements aus einer Python-Liste

Aufgabe 3.14

Welche Probleme werden *nicht handhabbar* genannt?