Algorithmen (Einführung)

Prüfungsvorbereitung

Aufgabe 1.1

Beschreibe möglichst präzise, was ein Algorithmus ist.

Aufgabe 1.1

Ein Algorithmus ist eine Problemlösungsbeschreibung mit diesen Eigenschaften:

- endlich
- deterministisch (eindeutig bestimmt)
- effektiv (ausführbar)

Aufgabe 1.2

Beschreibt 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 $(n + 1) \to n$
(4) Gehe zu (2)
(5) Ende

Aufgabe 1.2

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

Die Beschreibung ist endlich, deterministisch und effektiv. Daher ist es ein Algorithmus.

Falls die Zahl n kleiner als 100 ist, wird sie so lange um 1 vergrössert, bis $n = 100$ gilt und dann ist man fertig. Ist die Zahl n nicht kleiner als 100, ist man sofort fertig.

Aufgabe 1.3

Beschreibt 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 1.3

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

Die Beschreibung ist endlich und effektiv aber nicht deterministisch, da unklar ist, *wie* die Zahl a gewählt wird. Daher ist es kein Algorithmus

Aufgabe 1.4

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

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

Aufgabe 1.4

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

Die Beschreibung ist endlich und eindeutig aber nicht effektiv, da zu Beginn n durch $m = 0$ dividiert wird. Daher ist es kein Algorithmus.

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 $n/2 \to n$
    (4.2) Sonst setze $(3n + 1) \to n$
(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 1.5

(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 $n/2 \to n$
    (4.2) Sonst setze $(3n + 1) \to n$
(5) Gehe zu (3)
(6) Ende

(a) Input: $n = 10$ Output: 10, 5, 16, 8, 4, 2, 1

(b) Input: $n = 7$ Output: 7, 22, 11, 32, 16, 8, 4, 1

(c) Hier handelt sich um das COLLATZ- oder $(3n + 1)$-Problem: Bisher hat noch niemand eine Zahl n gefunden, für welche die obige Anweisungsfolge nicht terminiert aber es konnte auch noch niemand einen Beweis dafür angeben, dass dies *immer* so ist. Somit wissen wir nicht, ob das obige Verfahren ein Algorithmus gemäss unserer Definition ist.

Aufgabe 2.1

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

Aufgabe 2.1

grösster gemeinsamer Teiler (ggT)

$\Leftrightarrow$ greatest common divisor (gcd)

Aufgabe 2.2

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

Aufgabe 2.2

klassischer euklidscher Algorithmus:

$$\begin{aligned} \mathrm{ggT}(28,16) &= (12,16) \overset{S}{=} (16,12) = (4,12) \overset{S}{=} (12,4) = (8,4) \\ &= (4,4) = (0,4) \overset{S}{=} (4,0) = (4,0) = 4 \end{aligned}$$

Aufgabe 2.3

Zeige am Beispiel von $\mathrm{ggT}(6, -4)$, 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 2.3

klassischer euklidscher Algorithmus:
$\text{ggT}(6, -4) = (10, -4) = (14, -4) = \ldots$

Da der erste Operand bei jedem Schritt grösser wird, terminiert der Algorithmus nicht.

Aufgabe 2.4

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

Aufgabe 2.4

Der zentralen Nachteil besteht darin, dass bei (positiven) Operanden a und b mit grossem Unterschied das klassische Verfahren viel Zeit mit Subtraktionen verbringt, während die moderne Berechnungsmethode dies durch die Berechnung des Rests mit dem Modulo-Operator abkürzt. *Beispiel:*

- *klassisch:* $\mathrm{ggT}(99, 1) = (98, 1) = \cdots = (0, 1) = (1, 0) = 1$
- *modern:* $\mathrm{ggT}(99, 1) = (1, 0) = 1$

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.

```
def ggt_classic(a, b):
    a, b = abs(a), abs(b)
```

Aufgabe 2.5

Python-Funktion, die `ggt(a,b)` mit dem klassischen Algorithmus von Euklid berechnet und zurückgibt.

```
def ggt_classic(a, b):
    a, b = abs(a), abs(b)
    while b != 0:
        if a < b:
            a, b = b, a
        a, b = b, a - b
    return a
```

Aufgabe 2.6

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

Aufgabe 2.6

moderner euklidscher Algorithmus:

$$\mathrm{ggT}(33, 12) = (12, 9) = (9, 3) = (3, 0) = 3$$

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.

d	e	f		g	g	t	_	m	o	d	e	r	n	(	a	,		b	)	:	

Aufgabe 2.7

Python-Funktion, die `ggt(a,b)` mit dem modernen Algorithmus von Euklid berechnet und zurückgibt.

```
def ggt_modern(a, b):
    while b != 0:
        a, b = b, a % b
    return a
```

Aufgabe 2.8

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

Aufgabe 2.8

Beim beim (modernen) Algorithmus von Euklid tritt der Worst Case auf, wenn a und b zwei aufeinanderfolgende Fibonacci-Zahlen sind.

Aufgabe 2.9

Schreibe die ersten 10 Werte der Fibonacci-Folge auf.

Aufgabe 2.9

Die ersten 10 Werte der Fibonacci-Folge:

ältere Schreibweise: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

moderne Schreibweise: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Aufgabe 3.1

Ordne die Algorithmen aufgrund ihrerer 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 3.1

$B < E < C < A < D$

Da es sich um *asymptotische Laufzeiten*, d. h Laufzeiten für *grosse* n handelt, kann es sein, dass die Dominanz einer der Funktionen erst ab einem ausreichend grossen n erkennbar wird.

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 3.2

(a) $O(2^{n+1}) = O(2 \cdot 2^n) = O(2^n)$

(b) $O(\sqrt{4n}) = 2\sqrt{n} = O(\sqrt{n})$

(c) $O((n^2 + n)(n + 2)(n + 3)) = O(n^4 + \dots) = O(n^4)$

(d) $O(3) = O(1)$

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 3.3

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

Zeile	Kosten	Anzahl
2	c	1
3	c	n
4	c	n
5	c	1

Aufgabe 3.3

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

Zeile	Kosten	Anzahl
2	c	1
3	c	n
4	c	n
5	c	1

$$T(n) = c + c \cdot n + c \cdot n + c = 2c + 2cn \in O(n)$$

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 3.4

```
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
```

Zeile	Kosten	Anzahl
2	c	1
3	c	n
4	c	n^2
5	c	n^2
6	c	1

Aufgabe 3.4

```
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
```

Zeile	Kosten	Anzahl
2	c	1
3	c	n
4	c	n^2
5	c	n^2
6	c	1

$$T(n) = 2c + cn + 2cn^2 \in O(n^2)$$

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 3.5

```
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
```

Zeile	Kosten	Anzahl
2	c	1
3	c	1
4	c	$\log_3 n$
5	c	$\log_3 n$
6	c	$\log_3 n$
6	c	1

Aufgabe 3.5

```
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
```

Zeile	Kosten	Anzahl
2	c	1
3	c	1
4	c	$\log_3 n$
5	c	$\log_3 n$
6	c	$\log_3 n$
6	c	1

$$T(n) = 3c + 3c \log_3(n) \in O(\log_3 n)$$

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 3.6

```
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
```

Zeile	Kosten	Anzahl
2	c	1
3	c	n
4	c	n
5	c	n
6	c	n
7	c	1

$$T(n) = 2c + 4cn \in O(n)$$

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 3.7

$$T(n) \in O(n^2)$$

$$T(10\,000) = c \cdot 10\,000^2 \overset{*}{=} 40\,\mathrm{s}$$

$$T(30\,000) = c \cdot 30\,000^2 = c \cdot (3 \cdot 10\,000)^2 = 9 \cdot c \cdot 10\,000^2$$

$$\overset{*}{=} 9 \cdot 40\,\mathrm{s} = 360\,\mathrm{s} = 6\,\mathrm{Min}$$

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 3.8

$$T(n) \in O(\log n)$$

$$T(10^4) = c \cdot \log 10^4 \stackrel{*}{=} 8\,\text{Min.}$$

$$T(10^{12}) = c \cdot \log(10^{12}) = c \cdot \log\left(10^{4 \cdot 3}\right) = c \cdot \log\left(\left(10^4\right)^3\right)$$

$$= 3 \cdot c \cdot \log(10^4) \stackrel{*}{=} 3 \cdot 8\,\text{Min.} = 32\,\text{Min}$$

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 3.9

$$T(n) \in O(n)$$

$$T(6 \cdot 10^7) = c \cdot 6 \cdot 10^7 \overset{*}{=} 4\,\mathrm{s}$$

$$T(9 \cdot 10^7) = c \cdot 9 \cdot 10^7 = 1.5 \cdot c \cdot 6 \cdot 10^7$$
$$\overset{*}{=} 1.5 \cdot 4\,\mathrm{s} = 6\,\mathrm{s}$$

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 3.10

$$T(n) \in O\big(n \log(n)\big)$$

$$T(200) = c \cdot 200 \log_2(200) \overset{*}{=} 3\,\text{ms}$$

$$\begin{aligned} T(40\,000) &= c \cdot 40\,000 \cdot \log_2(40\,000) = c \cdot 200^2 \cdot \log_2(200^2) \\ &= c \cdot 200 \cdot 200 \cdot 2 \cdot \log_2(200) = 400 \cdot c \cdot 200 \log_2(200) \\ &\overset{*}{=} 400 \cdot 3\,\text{ms} = 1200\,\text{ms} = 1.2\,\text{s} \end{aligned}$$

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 3.11

$$T(n) \in O(2^n)$$

$$T(40) = c \cdot 2^{40} \overset{*}{=} 6\,\mathrm{h}$$

$$T(45) = c \cdot 2^{45} = c \cdot 2^{40} \cdot 2^5 = 32 \cdot c \cdot 2^{40}$$

$$\overset{*}{=} 32 \cdot 6\,\mathrm{h} = 8 \cdot 4 \cdot 6\,\mathrm{h} = 8 \cdot 24\,\mathrm{h} = 8\,\mathrm{d}$$

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 3.12

$$T(n) \in O(n!)$$

$$T(7) = c \cdot 7! \overset{*}{=} 10\,\mathrm{s}$$

$$T(9) = c \cdot 9! = c \cdot 9 \cdot 8 \cdot 7! = 72 \cdot c \cdot 7!$$

$$\overset{*}{=} 72 \cdot 10\,\mathrm{s} = 12 \cdot 6 \cdot 10\,\mathrm{s} = 12 \cdot 60\,\mathrm{s} = 12\,\mathrm{Min.}$$

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 3.13

(a) Gnomesort: $O(n^2)$

(b) Suche in unsortierter Liste: $O(n)$

(c) Mergesort: $O(n \log_2 n))$

(d) Lesen eines Elements aus Python-Liste: $O(1)$

Aufgabe 3.14

Welche Probleme werden *nicht handhabbar* genannt?

Aufgabe 3.14

Nicht handhabbare Probleme sind Probleme, für die es keine effizenten Algorithmen gibt, mit denen sie gelöst werden können. Diese liegen, im Allgemeinen in den Komplexitätsklassen $O(k^n)$, $O(n!)$ oder $O(n^n)$.