

**Aufgabe 1**

Warum ist ein Kochrezept strenggenommen kein Algorithmus.

**Aufgabe 2**

Zeige schrittweise, wie der klassische Algorithmus von Euklid den  $\text{ggT}(17, 5)$  berechnet, indem du die Kette *aller* Zwischenresultate notierst.

**Aufgabe 3**

Zeige, warum die klassische Version des euklidischen Algorithmus nicht funktioniert, wenn eine der beiden Zahlen negativ ist, indem du  $\text{ggT}(8, -6)$  berechnest.

**Aufgabe 4**

Zeige schrittweise, wie der moderne Algorithmus von Euklid den  $\text{ggT}(17, 5)$  berechnet, indem du die Kette *aller* Zwischenresultate notierst.

**Aufgabe 5**

Zeige schrittweise, wie der moderne Algorithmus von Euklid den  $\text{ggT}(72, 116)$  berechnet, indem du die Kette *aller* Zwischenresultate notierst.

**Aufgabe 6**

Implementiere den klassischen Algorithmus von Euklid für den grössten gemeinsamen Teiler zweier ganzer Zahlen `a` und `b` als Python-Funktion `gcd_classic(a, b)`. Die Funktion soll den  $\text{ggT}$  als Wert zurückgeben.

## Aufgabe 7

Implementiere den modernen Algorithmus von Euklid zur Berechnung des grössten gemeinsamen Teilers (ggT) zweier ganzer Zahlen  $a$  und  $b$  als Python-Funktion `gcd_classic(a, b)`. Die Funktion soll den ggT als Wert zurückgeben.

## Aufgabe 8

Bestimme für jede Funktion  $f(n)$  und jede Problemgrösse  $n$  die Dauer  $t$ , wenn der Algorithmus  $f(n)$  Sekunden zur Lösung des Problems benötigt.

| $f(n)$       | $n = 2$ | $n = 4$ | $n = 8$ | $n = 16$            |
|--------------|---------|---------|---------|---------------------|
| 1            |         |         |         |                     |
| $\log_2 n$   |         |         |         |                     |
| $\sqrt{n}$   |         |         |         |                     |
| $n$          |         |         |         |                     |
| $n \log_2 n$ |         |         |         |                     |
| $n^2$        |         |         |         |                     |
| $n^3$        |         |         |         | 4096                |
| $2^n$        |         |         |         | 65536               |
| $n!$         |         |         | 40320   | $2.1 \cdot 10^{13}$ |

## Aufgabe 9

Zu welcher Komplexitätsklasse gehören die Algorithmen mit der folgenden Laufzeitfunktion  $T(n)$  [in ms].

- (a)  $T(n) = 4n + 5n^2 - 2$
- (b)  $T(n) = 2^{n+3}$
- (c)  $T(n) = 4$
- (d)  $T(n) = \sqrt{7.6n}$
- (e)  $T(n) = \log_2(234n)$
- (f)  $T(n) = (4n + 3)(5n - 4)(7n - 6)$

## Aufgabe 10

In welcher Komplexitätsklasse befindet sich  $T_1(n) + T_2(n)$ , wenn  $T_1(n) \in \mathcal{O}(n^2)$  und  $T_2(n) \in \mathcal{O}(n^3)$  gilt?

## Aufgabe 11

In welcher Komplexitätsklasse befindet sich  $T_1(n) \cdot T_2(n)$ , wenn  $T_1(n) \in \mathcal{O}(n^2)$  und  $T_2(n) \in \mathcal{O}(n^3)$  gilt?

### Aufgabe 12

Eine Implementation eines Algorithmus' hat eine Laufzeitkomplexität von  $\mathcal{O}(n^2)$  und benötigt etwa  $20\mu\text{s}$  für das Lösen eines Problems der Grösse  $n = 100$ . Bestimme die ungefähre Laufzeit für ein Problem der Grösse  $n = 200$ .

### Aufgabe 13

Eine Implementation eines Algorithmus' hat eine Laufzeitkomplexität von  $\mathcal{O}(\sqrt{n})$  und benötigt etwa  $10\text{ms}$  für das Lösen eines Problems der Grösse  $n = 200$ . Bestimme die ungefähre Laufzeit für ein Problem der Grösse  $n = 20000$ .

### Aufgabe 14

Eine Implementation eines Algorithmus' hat eine Laufzeitkomplexität von  $\mathcal{O}(\log_2 n)$  und benötigt etwa  $5\text{s}$  für das Lösen eines Problems der Grösse  $n = 1000$ . Bestimme die ungefähre Laufzeit für ein Problem der Grösse  $n = 8000$ .

### Aufgabe 15

Eine Implementation eines Algorithmus' hat eine Laufzeitkomplexität von  $\mathcal{O}(n!)$  und benötigt etwa  $50\text{ms}$  für das Lösen eines Problems der Grösse  $n = 19$ . Bestimme die ungefähre Laufzeit für ein Problem der Grösse  $n = 20$ .

### Aufgabe 16

Bestimme die Komplexitätsklasse des Python Code-Fragments:

```
1 s = 0
2 for i in range(0, len(A)):
3     s += A[i]
```

### Aufgabe 17

Bestimme die Komplexitätsklasse des Python Code-Fragments:

```
1 s = 1
2 for i in range(1, n):
3     for j in range(1, n):
4         s = s + i*j
```

### Aufgabe 18

Bestimme die Komplexitätsklasse des Python Code-Fragments.

```
1 a = 4
2 b = a**2
3 c = -b
4 d = (a+b)*c
```

### Aufgabe 19

Bestimme die Komplexitätsklasse des folgenden Code-Fragments:

```
1 i = n
2 s = 0
3 while i > 0:
4     s = s + 1
5     i = i // 2
```

### Aufgabe 20

Zu welcher Komplexitätsklasse gehören die folgenden Algorithmen?

- (a) Ein Element in einer unsortierten Liste suchen.
- (b) Zwei Matrizen multiplizieren.
- (c) Eine Liste mit Bubblesort sortieren.
- (d) Die Brute-Force-Lösung des Travelling Salesman-Problems.
- (e) Eine Liste mit zufällig angeordneten Elementen mit Quicksort sortieren.