

Aufgabe 1

$f(n)$	$n = 2$	$n = 4$	$n = 8$	$n = 16$
1	1	1	1	1
$\log_2 n$	1	2	3	4
$\sqrt{n}$	1.4	2	2.8	4
n	2	4	8	16
$n \log_2 n$	2	8	24	64
n^2	4	16	64	256
n^3	8	64	512	4096
2^n	4	16	256	65536
$n!$	2	24	40320	$2.1 \cdot 10^{13}$

Aufgabe 2

- (a) $T(n) = 4n + 5n^2 - 2 \in \mathcal{O}(n^2)$
- (b) $T(n) = 2^{n+3} = 2^n \cdot 2^3 \in \mathcal{O}(2^n)$
- (c) $T(n) = 4 \in \mathcal{O}(1)$
- (d) $T(n) = \sqrt{7.6n} = \sqrt{7.6} \cdot \sqrt{n} \in \mathcal{O}(\sqrt{n})$
- (e) $T(n) = \log_2(234n) = \log_2 234 + \log_2 n \in \mathcal{O}(\log_2 n)$
- (f) $T(n) = (4n + 3)(5n - 4)(7n - 6) \in \mathcal{O}(n^3)$

Aufgabe 3

$$(T_1(n) + T_2(n)) \in \mathcal{O}(\max(n^2, n^3)) = \mathcal{O}(n^3)$$

Aufgabe 4

$$T_1(n) \cdot T_2(n) \in \mathcal{O}(n^2 \cdot n^3) = \mathcal{O}(n^5)$$

Aufgabe 5

$$T(n) = C \cdot 100^2 = 20 \mu s \quad (*) \quad [C \text{ ist ein Proportionalitätsfaktor}]$$

$$T(2n) = C \cdot 200^2 = C \cdot (2 \cdot 100)^2 = 2^2 \cdot C \cdot 100^2 \stackrel{(*)}{=} 4 \cdot 20 \mu s = 80 \mu s$$

Allgemein: In $\mathcal{O}(n^2)$ bewirkt das Verdoppeln der Problemgröße eine Vervierfachung der Laufzeit.

Man hätte auch die erste Gleichung nach C auflösen und diesen Wert in die zweite Gleichung einsetzen können. Meist lässt sich die Rechnung jedoch in der oben beschriebenen Weise „kurzschliessen“.

Aufgabe 6

$$T(200) = C \cdot \sqrt{200} = 10 \text{ ms } (*)$$

$$\begin{aligned} T(20\,000) &= C \cdot \sqrt{100 \cdot 200} \\ &= C \cdot \sqrt{100} \cdot \sqrt{200} = 10 \cdot C \cdot \sqrt{200} \\ &\stackrel{(*)}{=} 10 \cdot 10 \text{ ms} = 100 \text{ ms} \end{aligned}$$

Allgemein: In $O(\sqrt{n})$ bewirkt eine Vergrößerung der Problemgröße mit dem Faktor 100 eine Vergrößerung der Laufzeit mit dem Faktor $\sqrt{100} = 10$.

Aufgabe 7

$$T(1000) = C \cdot \log_2(1000) = 5 \text{ s } (*)$$

$$\begin{aligned} T(8000) &= C \cdot \log_2(8 \cdot 1000) = C [\log_2(8) + \log_2(1000)] \\ &= C \log_2(8) + C \log_2(1000) \\ &\stackrel{(*)}{=} C \cdot 3 + 5 \text{ s} = \dots \end{aligned}$$

Hier benötigen wir den konkreten Wert der Konstanten C .

$$C \cdot \log_2(1000) = 5 \text{ s} \quad \text{verwende } 10^3 \approx 2^{10}$$

$$C \cdot \log_2(2^{10}) \approx 5 \text{ s}$$

$$C \cdot 10 \approx 5 \text{ s}$$

$$C \approx 0.5 \text{ s}$$

$$\text{Damit: } \dots = 0.5 \text{ s} \cdot 3 + 5 \text{ s} = 6.5 \text{ s}$$

Allgemein: Multipliziert man die Problemgröße eines logarithmisch wachsenden Algorithmus mit dem Faktor k , so erhöht sich die Laufzeit um den Summanden $C \log(k)$.

Aufgabe 8

$$T(19) = C \cdot 19! = 50 \text{ ms } (*)$$

$$T(20) = C \cdot 20! = C \cdot 20 \cdot 19! = 20 \cdot C \cdot 19!$$

$$\stackrel{(*)}{=} 20 \cdot 50 \text{ ms} = 1000 \text{ ms} = 1 \text{ s}$$

Allgemein: Vergrößert man ein exponentiell wachsendes Problem der Größe n um eine weitere Eingabe, so erhöht sich die Laufzeit um den Faktor $n + 1$.

Aufgabe 9

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

Zeile	Kosten	Anzahl
1	c_1	1
2	c_2	n
3	c_3	n

$$T(n) = c_1 \cdot 1 + (c_2 + c_3) \cdot n \in \mathcal{O}(n) \text{ wobei } n = \text{len}(A)$$

Aufgabe 10

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

Zeile	Kosten	Anzahl
1	c_1	1
2	c_2	$n - 1$
3	c_3	$(n - 1)(n - 1)$
4	c_4	$(n - 1)(n - 1)$

$$T(n) = c_1 \cdot 1 + c_2(n - 1) + (c_3 + c_4)(n - 1)(n - 1) \in \mathcal{O}(n^2)$$

Aufgabe 11

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

Zeile	Kosten	Anzahl
1	c_1	1
2	c_2	1
3	c_3	1
4	c_4	1

$$T(n) = (c_1 + c_2 + c_3 + c_4) \cdot 1 \in \mathcal{O}(1)$$

Aufgabe 12

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

Zeile	Kosten	Anzahl
1	c_1	1
2	c_2	1
3	c_3	$\log_2 n$
4	c_4	$\log_2 n$
5	c_5	$\log_2 n$

$$T(n) = (c_1 + c_2) \cdot 1 + (c_3 + c_4 + c_5) \cdot \log_2 n \in \mathcal{O}(\log_2 n)$$

Aufgabe 13

- (a) Nach einem Element in einer sortierten Liste suchen.

$$\mathcal{O}(\log_2 n)$$

- (b) Zwei Matrizen multiplizieren.

$$\mathcal{O}(n^3)$$

- (c) Eine Liste von Zahlen mit Bubblesort sortieren.

$$\mathcal{O}(n^2)$$

- (d) Die Brute-Force-Lösung des Travelling Salesman-Problems.

$$\mathcal{O}(n!)$$

- (e) Eine Liste von Zufallszahlen mit Quicksort sortieren.

$$\mathcal{O}(n \log_2 n)$$

Aufgabe 14

