Aufgabe 1

Das Modell geht davon aus, dass ein Algorithmus für seine Ausführung eine von der Problemgrösse n abhängige Anzahl elementarer Verarbeitungsschritt benötigt.

Nehmen wir vereinfachend an, dass jeder solche Verarbeitungsschritt gleich viel Zeit benötigt, so ist die Laufzeit $T(n)$ des Algorithmus proportional zu der Gesamtzahl dieser Schritte.

$T(n) \in O(f(n))$, wenn es eine Funktion $f(n)$ gibt, so dass

$$\lim_{n\to\infty} \frac{T(n)}{f(n)} < \infty$$

in Worten: $T(n)$ wächst für grosse n nicht schneller als $f(n)$.

Aufgabe 2

(a) $T(n) = 4n^2 + 2n + 5n^3 + 1 \in O(n^3)$

Kontrolle: $\lim_{n\to\infty} \frac{5n^3 + 4n^2 + 2n + 1}{n^3} = \lim_{n\to\infty} \left(5 + \frac{4}{n} + \frac{2}{n^2} + \frac{1}{n^3}\right) = 5 < \infty$

(b) $T(n) = 5 \cdot 3^{n-1} = 5 \cdot 3^n \cdot 3^{-1} = \frac{5}{3} \cdot 3^n \in O(3^n)$

Kontrolle: $\lim_{n\to\infty} \frac{5 \cdot 3^{n-1}}{3^n} = \lim_{n\to\infty} \left(5 \cdot 3^{-1}\right) = \frac{5}{3} < \infty$

(c) $T(n) = 27 \in O(1)$

Kontrolle: $\lim_{n\to\infty} \frac{27}{1} = 27 < \infty$

(d) $T(n) = (4n^2 + 3)(5n - 4)(7n^3 - 6) \in O(n^6)$

(e) Logarithmengesetze:

$T(n) = \log_2(4n^2) = \log_2(4) + 2\log_2(n) \in O(\log_2(n))$

Aufgabe 3

Basiswechsel: $\log_a(x) = \frac{\log_b(x)}{\log_b(a)}$ für alle Basen a, $b > 0$ und $a, b \neq 1$

$$T(n) = \log_2(n) = \frac{\log_{10}(n)}{\log_{10}(2)} = \underbrace{\frac{1}{\log_{10}(2)}}_{\text{const.}} \cdot \log_{10}(n) \in O(\log_{10}(n))$$

Aufgabe 4

$T_1(n) \cdot T_2(n) \cdot T_3(n) \in \mathcal{O}\big(n^3 \cdot n^4 \cdot n\big) = \mathcal{O}(n^8)$

Aufgabe 5

$T_1(n) + T_2(n) + T_3(n) \in \mathcal{O}\big(\max\{n^3, n^2, n^4\}\big) = \mathcal{O}(n^4)$

Aufgabe 6

(a) $T_1(n) = 100 \cdot n$, $T_2(n) = 0.1 \cdot n^2$

$$
\begin{aligned}
T_1(n) &= T_2(n) \\
100 \cdot n &= 0.1 \cdot n^2 \\
0 &= 0.1 \cdot n^2 - 100n \quad || \cdot 10 \\
0 &= n^2 - 1000n \\
0 &= n(n - 1000) \\
n_1 &= 0 \quad \text{keine sinnvolle Lösung} \\
n_2 &= 1000
\end{aligned}
$$

Moral: ein Velo ist grundsätzlich langsamer als ein Auto, auch wenn es im unteren Geschwindigkeitsbereich das Auto überholt.

(b) $T_1(n) = 1000 \cdot 2^n$, $T_2(n) = 4^n$

$$
\begin{aligned}
T_1(n) &= T_2(n) \\
1000 \cdot 2^n &= 4^n \\
1000 &= \frac{4^n}{2^n} \\
1000 &= \left(\frac{4}{2}\right)^n = 2^n \\
n &= 10 \quad (\text{denn } 2^{10} = 1024)
\end{aligned}
$$

Aufgabe 7

$T(100) = C \cdot 100^3 = 20\,\mu\text{s}$ $(*)$ [C ist ein Proportionalitätsfaktor]

$T(200) = C \cdot 200^3 = C \cdot (2 \cdot 100)^3 = 2^3 \cdot C \cdot 100^3 \overset{(*)}{=} 8 \cdot 20\,\mu\text{s} = 160\,\mu\text{s}$

Allgemein: In $O(n^3)$ bewirkt das Verdoppeln der Problemgrösse eine Verachtfachung der Laufzeit.

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

Aufgabe 8

$T(200) = C \cdot 2^{200} = 10\,\text{s}$ $(*)$

$T(205) = C \cdot 2^{205} = C \cdot 2^{200} \cdot 2^5$

$\overset{(*)}{=} 10\,\text{s} \cdot 32 = 320\,\text{s}$

Allgemein: In $O(2^n)$ bewirkt eine Vergrösserung der Problemgrösse um k eine Vergrösserung der Laufzeit mit dem Faktor 2^k. Daher sagt man bei Algorithmen mit exponentiellen (und höheren) Laufzeitkomplexitäten, dass das Problem (mit diesem Algorithmus) *nicht handhabbar* sei.

Aufgabe 9

$T(10^3) = C \log_2(10^3) = 5\,\text{s}$ $(*)$

$T(10^6) = C \log_2(10^6) = C \log_2\left((10^3)^2\right)$

$= 2 \cdot C \log_2(10^3)$ Logarithmengesetze

$\overset{(*)}{=} 2 \cdot 5\,\text{s} = 10\,\text{s}$

Allgemein: quadriert man die Problemgrösse eines logarithmisch wachsenden Algorithmus, so verdoppelt sich die Laufzeit.

Aufgabe 10

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

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

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

Allgemein: Vergrössert man ein faktoriell wachsendes Problem der Grösse n um eine weitere Eingabe, so vergrössert sich die Laufzeit mit dem Faktor $n + 1$.

Aufgabe 11

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

Zeile	Kosten	Anzahl
1	c	1
2	c	n
3	$2c$	n

$T(n) = c + 3c \cdot n \in \mathcal{O}(n)$

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
2	c	1
3	$2c$	$\log_2(n)$
4	$2c$	$\log_2(n)$
5	$2c$	$\log_2(n)$

$T(n) = 2c + 6c\log_2(n) \in \mathcal{O}(\log(n))$

Aufgabe 13

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

Zeile	Kosten	Anzahl
1	c	1
2	c	n
3	c	n^2

Zeile	Kosten	Anzahl
4	c	$3n^2$
6	c	$2n$
7	c	$4n$

$T(n) = c + 7cn + 4cn^2 \in \mathcal{O}(n^2)$

Aufgabe 14

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

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

$T(n) = 4c \in \mathcal{O}(1)$

Aufgabe 15

(a) Ein Element in einer unsortierten Liste der Länge n suchen.
$\mathcal{O}(n)$

(b) Zwei Matrizen multiplizieren.
$\mathcal{O}(n^3)$

(c) Eine n-elementige Liste von Zahlen mit Gnomesort sortieren.
$\mathcal{O}(n^2)$

(d) Eien quadratische Gleichung lösen.
$\mathcal{O}(1)$

(e) Eine n-elementige Zufallsliste mit Quicksort sortieren.
$\mathcal{O}(n \log_2 n)$