Aufgabe 2

$$\begin{aligned} A \quad & f(n) \\ B \quad & g(n) \\ g(n) = & O(f(n) \log n) \end{aligned}$$

a) Beispiele

i) A schneller B

$$\begin{aligned} f(n) &= n^2 \\ g(n) &= n \log n \\ n \log n &= O(n^2 \log n) \end{aligned}$$

ii) B schneller A

$$\begin{aligned} f(n) &= n \\ g(n) &= n \log n \\ n \log n &= O(n^2 \log n) \end{aligned}$$

b)

$$g(n) = \Omega(f(n) \log n)$$

$\Rightarrow$ **i)** gilt laut Definition.

c) ja, x Eingabe und $T(x)$ Laufzeit

$$\begin{aligned} T(n) &= \max T(x) \\ |x| &= n \quad \text{Laufzeit im schlimmsten Fall} \\ T(n) = \Theta(n^2) \end{aligned}$$

$\Rightarrow$ Bubblesort!
x_i Folge von i sortierten Elementen $\rightarrow T(n) = \Theta(n^2)$
$T(x_i) = O(i)$

$$\begin{aligned} T(n) &= \Omega(n^2) \\ y_n &= \text{Permutation} = \{1, 2, 3, ..., n, n-1, n-2, ..., \underline{1}\} \\ T(y_n) &= \Omega(n^2) \end{aligned}$$

Für alle Funktionen f,g in der Vorlesung gilt entweder $f = O(g)$ oder $g = O(f)$, aber: Es gibt Funktionen, für die wir es nicht machen können.

d) A hat Laufzeit $\Theta(n^2)$, kann es sein, dass $\forall x |x| = n = T(x) = O(n)$?
$\Rightarrow$ Nein. Die Laufzeit im schlimmsten Fall von $\Omega(n^2)$ bedeutet, dass es für jedes n eine Eingabe der Länge n gibt, bei der der Algorithmus n^2 Zeit benötigt.
A hat Laufzeit $O(n^2)$, kann es sein, dass $\forall x |x| = n = T(x) = O(n)$? $\Rightarrow$ ja.
A hat Laufzeit $\Omega(n^2)$, kann es sein, dass $\forall x |x| = n = T(x) = O(n)$? $\Rightarrow$ nein.

e) a) $3^n = O(2^n)$ Wahr oder Falsch? $\Rightarrow$ Falsch!
$4^n = 2^n * 2^n$, wobei $2^n \underbrace{=}_{\text{unmöglich!}} O(2^n)$

b) $\underbrace{\log 3^n}_{n \log 3} = \underbrace{O(\log 2^n)}_{n \log 2} \Rightarrow$ Wahr!

c) $3^n = \Omega(2^n) \Rightarrow 3^n > 2^n$

d) $log3^n = \Omega(log2^n) \Rightarrow \log 3^n > log2^n$

f) $4^n + n^2 \simeq 2^{2n} \geq 2^n \geq 2^{\sqrt{n}} \geq n^2 \log_2(n^2) \simeq n^2 \log n \geq n \log 2^n \simeq 5n^2 + n \log n \simeq n^2 + n - 1 \geq (\log n)^2 \geq \log_3 n \simeq \log_2 n$

Einschub: Laufzeitvergleich

$$5n^2 + nlogn = \Theta(n^2 + n - 1)$$

z.z.:
$$\begin{aligned} O(n^2 + n - 1) & \\ &= \{f | \exists n_0 \in \mathbb{N}, c \in \mathbb{R}^+ \forall n > n_0 \quad f(n) \leq c(n^2) + n - 1\} \\ 5n^2 + n * \log n &< 100(n^2 + n - 1) \\ 5n^2 &< 10n^2 \\ n \log n &< n^2 \quad \forall n \end{aligned}$$

0.1 RAM

Ziel ist es eine RAM zu bauen, die eine RAM bekommt und diese dann um 10 Plätze nach unten verschiebt.
Idee: Von hinten nach vorne Verschieben. Die Programmlänge ist in `R0` gegeben. Es braucht dann zwei freie Register.
Zwei freie Register bekommt man so:

```
R0 = R0 + 12  ; R_0 = n + 12
(R0) = R2     ; R_{n+12} = x_2
R0 = R0 -1    ; R_0 = n+11
(R0) = R1     ; R_{n+11} = x_1
```

Verschieben des Programms (Umschaufeln)

```
R1 = R0 - 11
R2 = R0 -1
loop:
  GZ R1, end
  (R2) = (R1)
  R2 = R2 -1
  R1 = R1-1
  GOTO loop
```

Aufräumen...

```
end:
  R11 = R0
  R0 = R0 + 1
```

```
R12 =  (R0)
R 0 =  R0 - 12
HALT
```

EKM

Einheitskostenmaß, jede Operation kostet 1.
Schritt:

1. 8 bzw. Konstant
2. $4 + \underbrace{8}_{\text{Kosten pro Schleife}} * \underbrace{n}_{\text{Schleifendurchläufe}}$
3. Konstant

$\Rightarrow \Theta(n)$

LKM