Landau-Notation

Erinnere dich an die Definition (mit $f : \mathbb{N} \to \mathbb{N}$):
$\mathcal{O}(f) = \{g : \mathbb{N} \to \mathbb{N} | \exists m \exists c \forall n : g(n) \leq m \cdot f(n) + c\}$

Aufgabe 1

Beweise folgende Aussagen:

$$f \in \mathcal{O}(f) \tag{1}$$
$$h \in \mathcal{O}(g) \wedge g \in \mathcal{O}(f) \Rightarrow h \in \mathcal{O}(f) \tag{2}$$
$$k \in \mathbb{N} \Rightarrow \mathcal{O}(f) = \mathcal{O}(k + f) \tag{3}$$
$$k \in \mathbb{N} \Rightarrow \mathcal{O}(f) = \mathcal{O}(k \cdot f) \tag{4}$$
$$\mathcal{O}(f + g) = \mathcal{O}(\max(f, g)) \tag{5}$$
$$f(n) = n^2 \wedge g(n) = n \Rightarrow f \notin \mathcal{O}(g) \tag{6}$$
$$a, b \in \mathbb{R}_{\geq 0} \wedge f(n) = log_a(n) \wedge g(n) = log_b(n) \Rightarrow \mathcal{O}(f) = \mathcal{O}(g) \tag{7}$$

Dabei ist max punktweise zu verstehen:
$\max(f, g)(n) = \max(f(n), g(n))$

BubbleSort

```
Input:L
n ← |L|
for i = 1, ..., n do
    for j = 0, ..., n − i do
        if L_j > L_{j+1} then
            x ← L_j
            L_j ← L_{j+1}
            L_{j+1} ← x
        end
    end
end
return L
```

Algorithm 1: BubbleSort

Aufgabe 2

Führe BubbleSort am Beispiel der Liste $L = [3, 1, 4, 2, 5]$ aus.

Aufgabe 3

Welche worst-case-Komplexität hat BubbleSort? Welche best-case-Komplexität hat er? Überlege dir dazu jeweils möglichst ungünstige bzw. günstige Sortierungen des Inputs.

Aufgabe 4

Wie kann man BubbleSort modifizieren? Wird dadurch eine der beiden Komplexitäten wesentlich besser?