Asymptotische Notation („O-Notation")

Wie schnell ist ein Algorithmus ?

Zwei Arten der Laufzeitoptimierung:

(1) „Der Code läuft jetzt 10% / 20% / ... schneller"

↳ Speedup um Konstante

↳ Programmieren

(2) „Der Algorithmus ist jetzt in einer besseren Komplexitätsklasse" Bsp. $O(n)$ statt $O(n^2)$

↳ AuD!

↳ Wird einen konstanten Speedup (1) immer überholen, <u>sobald „n" groß genug ist</u>

⇝ unabhängig von Sprache / Hardware

In der Praxis:

<u>Erst</u>: Gute Komplexitätsklasse erreichen (2)

<u>Dann</u>: Konstanten verbessern (1)

```
function MINIMUM1(a_1, ..., a_n)
    m ← a_1
    for i ∈ {2, ..., n} do
        if a_i < m then
            m ← a_i
        end if
    end for
    return m
end function
```

$\}\ 10 \in O(1)$

$\}\ 1$

$\}\ 10$

$\} \leq 11(n-1)$

$\}\ 5 \in O(1)$

$\} \ 11(n-1)+15 = 11n+4 =: f_1(n) \in O(n)$

$\leadsto O(n)$ mal $O(1)$

```
function MINIMUM2(a_1, ..., a_n)
    for m ∈ {a_1, ..., a_n} do
        if IS_LOWER_BOUND(m, a_1, ..., a_n) then
            return m
        end if
    end for
end function
```

$\}\ n+5$

$\}\ 1$

$\}\ n(n+5)+1 = n^2+5n+1 =: f_2(n)$

$\leadsto O(n)$ mal $O(n)$ Operationen $\leadsto O(n^2)$

```
function IS_LOWER_BOUND(m, a_1, ..., a_n)
    for i ∈ {1, ..., n} do
        if a_i < m then
            return FALSE
        end if
    end for
    return TRUE
end function
```

$\}\ 1$

$\}\ 5$

$\}\ 5$

$\}\ n+5 \in O(n)$

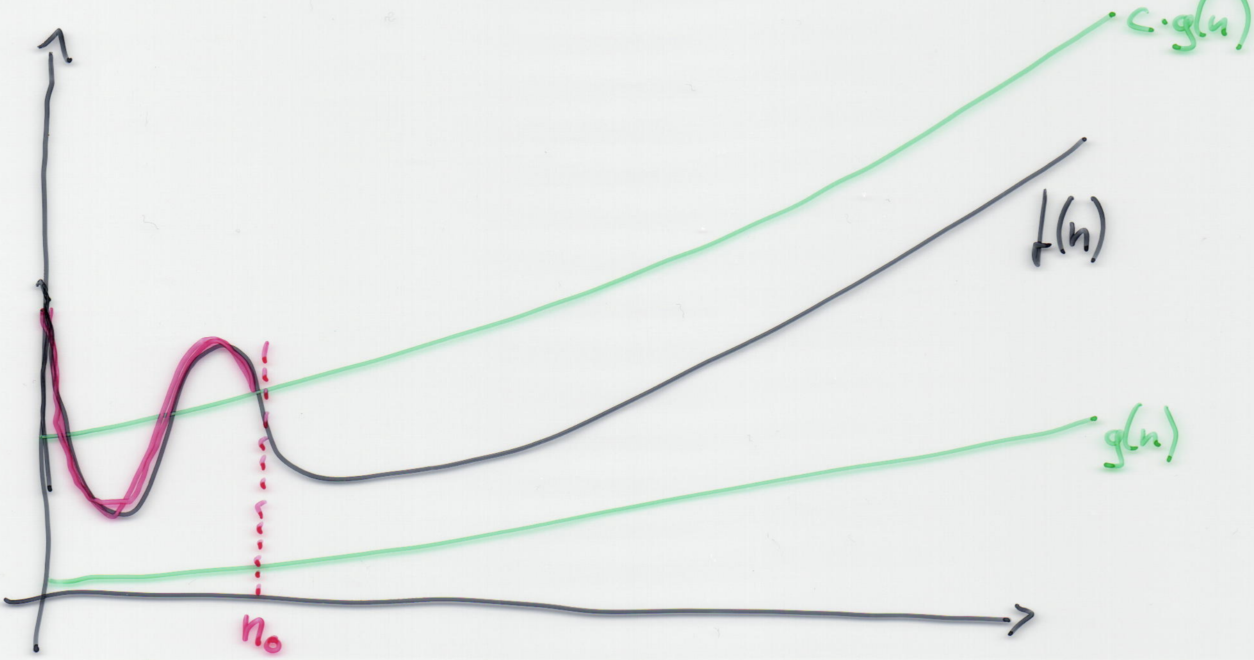
n	1	2	3	4	5	6	7	8	
Alg. 1 $f_1(n)$	15	26	37	48	59	70	81	92	...
Alg. 2 $f_2(n)$	7	15	25	37	51	67	86	105	...

O-Notation „obere Schranke"

$f \in O(g(n)) \iff$ Es gibt $c > 0$, $n_0 \in \mathbb{N}$, so dass

$$0 \leq f(n) \leq c \cdot g(n) \quad \forall n \geq n_0 \text{ gilt.}$$

Was soll das? Man will Aussage über das Wachstum von f treffen. Dabei interessioren „kleine n" nicht!



„f wächst nicht schneller als g"

Beispiel: $f(n) = 70n^3 + 125n^2 + 17$

$g(n) = n^3$

Dann: $f(n) = 70n^3 + 125n^2 + 17 \overset{n \geq 1}{\leq} 70n^3 + 125n^3 + 17n^3$

$= 212n^3$

$= c \cdot g(n)$ für $c = 212$, $g(n) = n^3$

Damit gilt:

Für $c = 212$ und ~~$n_0 = 1$ gilt:~~ $n_0 = 1 \leq n$ gilt

$$f(n) \leq c \cdot g(n)$$

Zudem: $f(n) \geq 0$

Also: $0 \leq f(n) \leq c \cdot g(n) \quad \forall n \geq n_0$

und $f(n) \in O(n^3)$

Anmerkung: c und n_0 sind nicht eindeutig!

$$f(n) = 70n^3 + 125n^2 + 17 \underset{\text{ab } n \geq 125}{\leq} 70n^3 + n^3 + 17$$

$$\underset{n \geq 1}{\leq} 70n^3 + n^3 + 17n^3$$

$$= 88n^3$$

$$= c\,g(n) \quad \text{für } c = 88$$

Also: Für $c = 88$ und $n_0 = 125 \leq n$ gilt:

$$0 \leq f(n) \leq c \cdot g(n)$$

Einschub: Achtung, $f \in \mathcal{O}(g)$ bedeutet nur, dass f durch g asymptotisch nach oben beschränkt ist!

Wenn z.B. $f \in \mathcal{O}(n)$, dann auch $f \in \mathcal{O}(n^2)$, etc.!

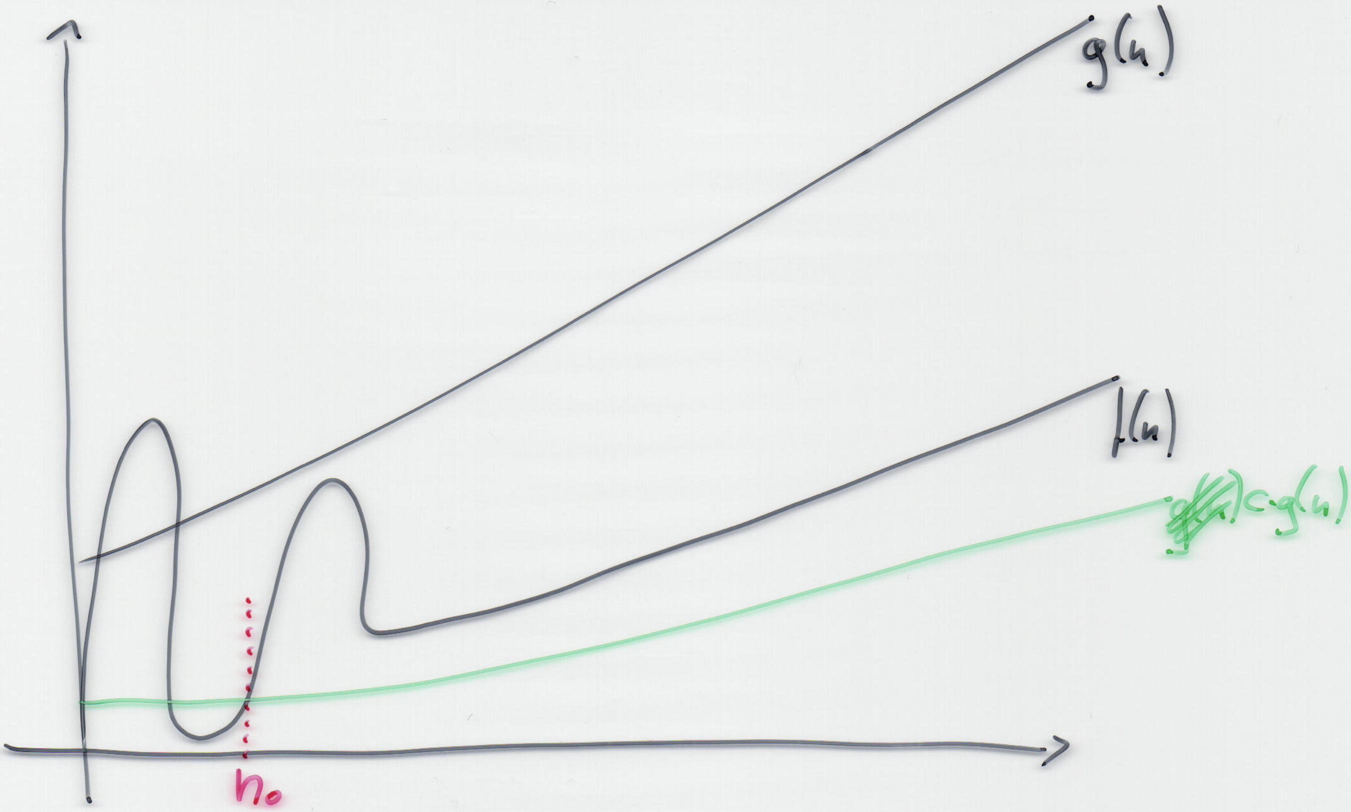
wegen: $n \in \mathcal{O}(n^2)$

	$\Omega(1)$	$\Theta(1)$	$\mathcal{O}(1)$	$\Omega(n)$	$\Theta(n)$	$\mathcal{O}(n)$	$\Omega(n^2)$	$\Theta(n^2)$	$\mathcal{O}(n^2)$
$f(n)=57$	X	X	X			X			X
$f(n)=5n+2$	X			X	X	X			X
$f(n)=n^3+n^2$	X			X			X		
$f(n)=2^n$	X			X			X		
$f(n)=n\log n$	X			X					X

Ω-Notation „untere Schranke"

$f \in \Omega(g(n)) \iff$ Es gibt $c>0$, $n_0 \in \mathbb{N}$, sodass

$$0 \le c \cdot g(n) \le f(n) \quad \forall n \ge n_0 \text{ gilt.}$$



$$f(n) = 70n^3 + 125n^2 + 17$$
$$g(n) = n^3$$

<u>Dann:</u> $$1 \cdot n^3 \underset{n \geq 1}{\leq} 70n^3 \underset{n \geq 1}{\leq} 70n^3 + 125n^2 + 17 = f(n)$$

<u>Damit:</u> Für $c = 1$ und $n_0 = 1$ gilt:

$$0 \leq c\,g(n) \leq f(n) \quad \text{für alle } n \geq n_0$$

Θ-Notation

$f \in \Theta(g) \iff$ Es gibt $c_1, c_2 > 0$, $n_0 \in \mathbb{N}$, so dass

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall n \geq n_0 \text{ gilt.}$$

Ω-Notation

⇝ O-Notation

Übung: $f \in \Theta(g) \iff f \in O(g)$ und $f \in \Omega(g)$

„$\Rightarrow$ <u>und</u> $\Leftarrow$"