Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Algorithmen & Komplexität

Institut für Informatik

Algorithmen und Datenstrukturen

Wintersemester 2020/21

3. Vorlesung

Laufzeitanalyse

Prof. Dr. Alexander Wolff *Lehrstuhl für Informatik I*

Recap: Diskutieren Sie mit ~~Ihrer Nachbarin!~~ *sich!*

1. Was sind die zwei (oder drei?) entscheidensten Fragen, die wir uns über einen Algorithmus stellen?

2. Warum eigentlich interessieren wir uns fürs Sortieren?

3. Welche Entwurfstechniken für Algorithmen kennen wir schon?

4. Wie beweisen wir die Korrektheit
   a) einer Schleife?
   b) eines inkrementellen Algorithmus?
   c) eines rekursiven Algorithmus?

Heute schon implementiert?
Zum Beispiel
– InsertionSort,
– MergeSort, ...
Probieren Sie's selbst!

Laufzeit analysieren: InsertionSort

```
InsertionSort(int[] A)
    for j = 2 to A.length do
        key = A[j]
        i = j − 1
        while i > 0 and A[i] > key do
            A[i + 1] = A[i]
            i = i − 1
        A[i + 1] = key
```

Zwei Konventionen:

1) $n :=$ Größe der Eingabe

$=_{\text{hier}} A.length$

2) Wir zählen nur Vergleiche!

(zwischen Elementen der Eingabe)

Gesucht: Maß für die Laufzeit, das nur von n abhängt.

Problem: Tatsächliche Laufzeit hängt stark von Eingabe ab.

Lösung(?): *Betrachte Extremfälle!*

Bester Fall: A aufsteigend sortiert $\Rightarrow n - 1$ Vergleiche

Schlechtester Fall: A absteigend sortiert

$\Rightarrow 1 + 2 + \cdots + (n-1) = \frac{n^2 - n}{2}$ Vgl.

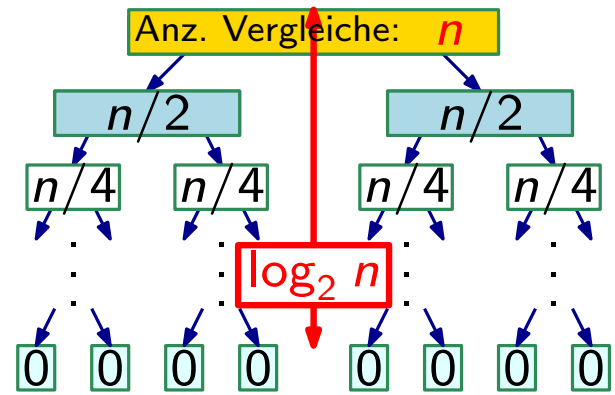
Laufzeit von MergeSort

MergeSort(int[] A, int $\ell = 1$, int $r = A.length$)

if $\ell < r$ **then**

- $m = \lfloor(\ell + r)/2\rfloor$ } teile
- MergeSort(A, ℓ, m)
- MergeSort$(A, m+1, r)$ } herrsche
- Merge(A, ℓ, m, r) } kombiniere

Rekursionsbaum von MergeSort



Effizient? ✓ Sei $V_{MS}(n)$ die maximale Anzahl von Vergleichen, die MergeSort zum Sortieren von n Zahlen benötigt.

Dann gilt

$$V_{MS}(n) = \left\{ \begin{array}{ll} 0 & \text{falls } n = 1, \\ 2V_{MS}(n/2) + n & \text{sonst.} \end{array} \right\} = n \cdot \log_2 n$$

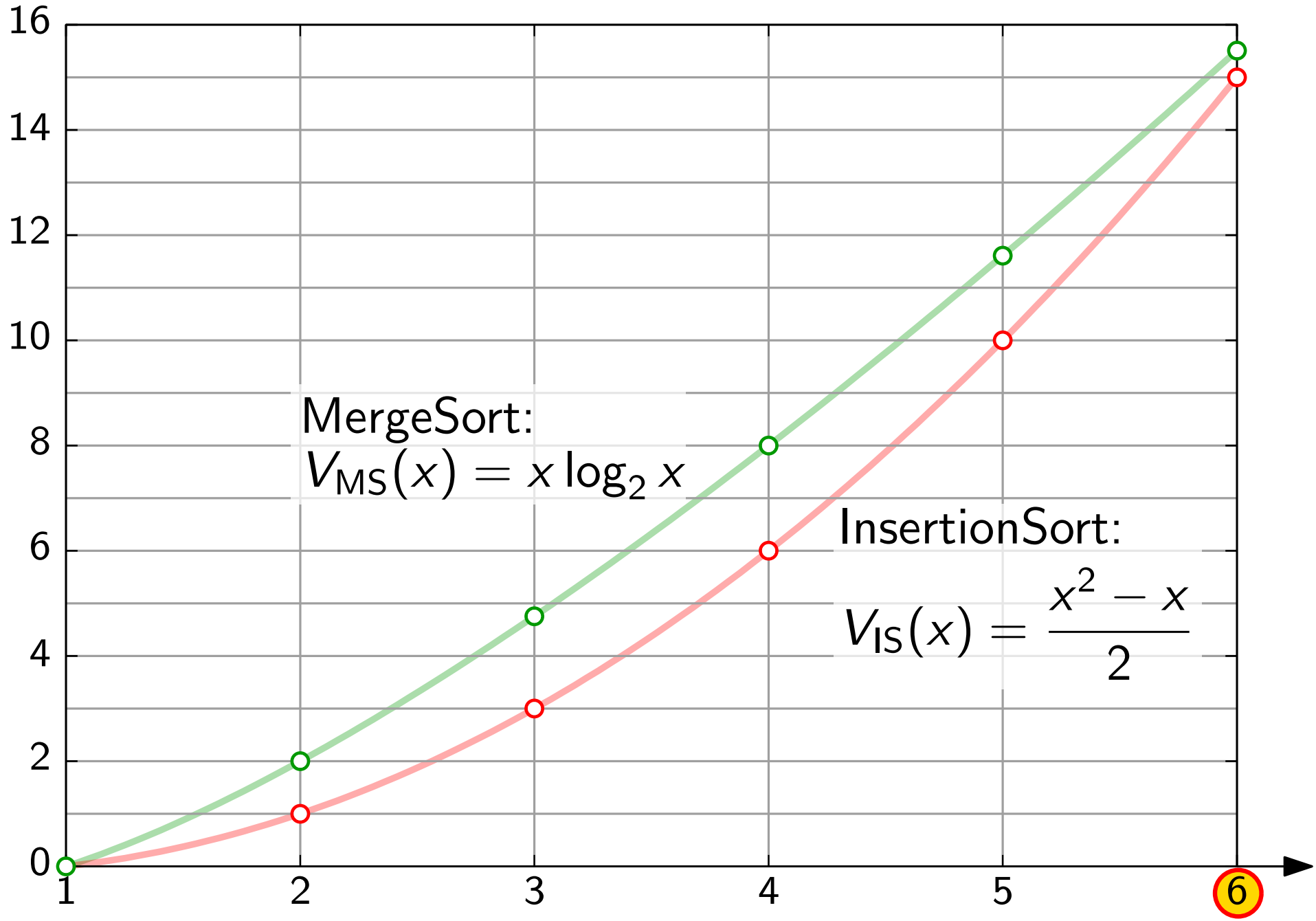
falls n Zweierpotenz – und sonst?

Beweis: per Induktion über n.

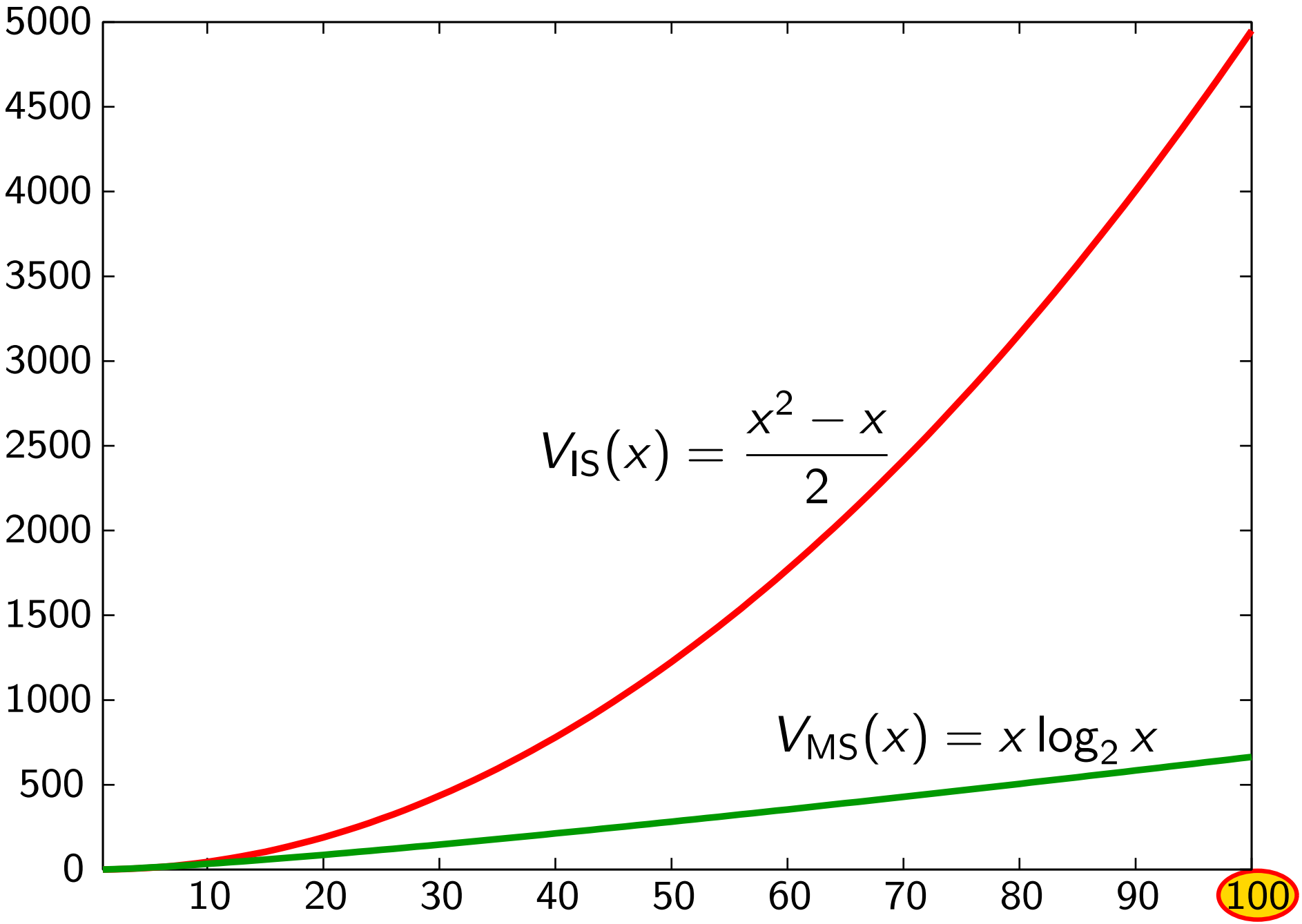
Induktionsanfang: $V_{MS}(1) = 1 \cdot \log_2 1 = 0$ ✓

Induktionsschritt: $V_{MS}(n) \underset{n>1}{=} 2 \cdot \frac{n}{2} \log_2 \frac{n}{2} + n = n \log_2 n - n \log_2 2 + n$
$= n \log_2 n$ ✓

Vergleich InsertionSort vs. MergeSort



Vergleich InsertionSort vs. MergeSort



Ein Klassifikationsschema für Funktionen (I)

Definition.

Sei $g : \mathbb{N} \to \mathbb{R}$ eine Funktion. Dann ist „Groß-Oh von g“

$$O(g) = \left\{ f : \mathbb{N} \to \mathbb{R} \;\middle|\; \begin{array}{c} \text{es gibt positive Konstanten } c \text{ und } n_0, \\ \text{so dass für alle } n \geq n_0 \text{ gilt:} \\ f(n) \leq c \cdot g(n) \end{array} \right\}$$

die Klasse der Fkt., die *höchstens* so schnell wachsen wie g.

Beispiel. $f(n) = 2n^2 + 4n - 20$

Behaupt.: $f \in O(n^2)$; m.a.W. f wächst höchstens quadratisch.

Beweis. *Wähle* positive c und n_0,
so dass für alle $n \geq n_0$ gilt: $f(n) \leq c \cdot n^2$.

$f(n) = 2n^2 + 4n - 20 \leq 6n^2 \quad \Rightarrow \quad$ wähle $c = 6$.

da $4n \leq 4n^2$

Welches n_0? Aussage gilt für jedes $n \geq 0$. Nimm z.B. $n_0 = 1$.

□

Ein Klassifikationsschema für Funktionen (I)

Definition.

Sei $g : \mathbb{N} \to \mathbb{R}$ eine Funktion. Dann ist „Groß-Oh von g“

$$O(g) = \left\{ f : \mathbb{N} \to \mathbb{R} \;\middle|\; \begin{array}{c} \text{es gibt positive Konstanten } c \text{ und } n_0, \\ \text{so dass für alle } n \geq n_0 \text{ gilt:} \\ f(n) \leq c \cdot g(n) \end{array} \right\}$$

die Klasse der Fkt., die *höchstens* so schnell wachsen wie g.

Beispiel. $f(n) = 2n^2 + 4n - 20$ *negiere!* ($\neg$)

Behaupt.: $f \notin O(n)$; m.a.W. f wächst schneller als linear.

Beweis. Zeige: für alle pos. Konst. c und n_0 gibt es ein $n \geq n_0$, so dass $f(n) > c \cdot n$.

Ein Klassifikationsschema für Funktionen (I)

Definition.

Sei $g : \mathbb{N} \to \mathbb{R}$ eine Funktion. Dann ist „Groß-Oh von g“

$$O(g) = \left\{ f : \mathbb{N} \to \mathbb{R} \;\middle|\; \begin{array}{c} \text{es gibt positive Konstanten } c \text{ und } n_0, \\ \text{so dass für alle } n \geq n_0 \text{ gilt:} \\ f(n) \leq c \cdot g(n) \end{array} \right\}$$

die Klasse der Fkt., die *höchstens* so schnell wachsen wie g.

Beispiel. $f(n) = 2n^2 + 4n - 20$

Behaupt.: $f \notin O(n)$; m.a.W. f wächst schneller als linear.

Beweis. Zeige: für alle pos. Konst. c und n_0 gibt es ein $n \geq n_0$, so dass $f(n) > c \cdot n$.

Also: bestimme n in Abh. von c und n_0, so dass

$n \geq n_0$ und $f(n) = 2n^2 + 4n - 20 > c \cdot n$.

Fortsetzung des Beweises $f \notin O(n)$

Bestimme n in Abh. von c und n_0, so dass $n \geq n_0$ und

$$f(n) = 2n^2 + 4n - 20 > c \cdot n.$$

Problem: Die „-20“ stört.

Aber wenn $n \geq 5$, dann gilt $4n - 20 \geq 0$.

D.h. wenn $n \geq 5$ und $2n^2 > cn$, dann $f(n) > cn$.

$\Updownarrow$

$n > c/2$

$\Uparrow$

Wie wär's mit $n = c$?

Gut, aber wir müssen sicherstellen, dass auch $n \geq 5$ und $n \geq n_0$ gilt.

Also nehmen wir $n = \lceil \max(c, 5, n_0) \rceil$.

Für dieses n gilt $n \geq n_0$ und $f(n) > cn$. Also gilt $f \notin O(n)$. $\square$

Ein Klassifikationsschema für Funktionen (II)

Definition.

Sei $g : \mathbb{N} \to \mathbb{R}$ eine Funktion. Dann ist „Groß-Omega von g“

$$\Omega(g) = \left\{ f : \mathbb{N} \to \mathbb{R} \;\middle|\; \begin{array}{c} \text{es gibt positive Konstanten } c \text{ und } n_0, \\ \text{so dass für alle } n \geq n_0 \text{ gilt:} \\ f(n) \geq c \cdot g(n) \end{array} \right\}$$

die Klasse der Fkt., die *mindestens* so schnell wachsen wie g.

Beispiel. $f(n) = 2n^2 + 4n - 20$

Bewiesen: $f \notin O(n)$, $f \in O(n^2)$, $f \in O(n^3)$

Entsprechend: $f \in \Omega(n)$, $f \in \Omega(n^2)$, $f \notin \Omega(n^3)$

Zusammen: $f \in \Theta(n^2)$

d.h. es gibt pos. Konst. c_1, c_2, n_0, so dass für alle $n \geq n_0$ gilt:

$$c_1 \cdot n^2 \leq f(n) \leq c_2 \cdot n^2.$$

Das Klassifikationsschema – *intuitiv*

$f \in O(n^2)$	bedeutet	f wächst	*höchstens* quadratisch.
$f \in \Omega(n^2)$			*mindestens*
$f \in \Theta(n^2)$			*„genau“*
$f \in o(n^2)$			*echt langsamer als*
$f \in \omega(n^2)$			*echt schneller als*

Genaue Definition für „klein-Oh“ und „klein-Omega“ s. Kap. 3, [CLRS].

Übung.

Gegeben folgende Funktionen $\mathbb{N} \to \mathbb{R}$ mit $n \mapsto \ldots$:

$$n^2,\ \log_2 n,\ \sqrt{n \log_2 n},\ 1{,}01^n,\ n^{\log_3 4},\ \log_2(n \cdot 2^n),\ 4^{\log_3 n}.$$

Sortieren Sie nach Geschwindigkeit des *asymptotischen Wachstums*, also so, dass danach gilt: $O(\ldots) \subseteq O(\ldots) \subseteq \cdots \subseteq O(\ldots)$.