O-Notation

Programmieren und Software-Engineering
Homomorphismen, Formale Sprachen und Syntax-Analyse

22. Februar 2023

Motivation

- Bestimmte Programmteile verursachen je nach Datenmenge unterschiedliche *Laufzeiten* bei der Ausführung.
- Bei geringen Datenmengen bleibt dies oft unbemerkt und spielt keine Rolle.
- Reale Programme arbeiten jedoch mit großen Datenmengen, umfangreichen Datenbanken, führen komplexe Berechnungen durch, etc.
- Durch die theoretische Laufzeitanalyse kann man etwaige Flaschenhälse und Performanceprobleme erkennen.
- Relevant ist hierbei weniger die konkrete Ausführungsgeschwindigkeit, sondern das Wachstum der Laufzeit in Abhängigkeit von der Anzahl der Eingabedaten.
- In der praktischen Arbeit werden *Profiler* verwendet, um zu analysieren wieviel Zeit in welchen Programmteilen verbracht wird.
- Für theoretische Analysen wird die *O*-Notation verwendet.

Komplexität von Algorithmen

Die theoretische Analyse von Algorithmen untersucht:

- (Lauf-)Zeit-Komplexität
- (Speicher-)Platz-Komplexität
- Evtl. weitere Parameter, z.B.:
  - Anzahl Vergleiche
  - Anzahl Bewegungen der Datensätze

Weitere Unterscheidung:

- **Worst-Case**: der ungünstigste Fall
- Average-Case: der zu erwartende, durchschnittliche Fall
- Best-Case

Komplexität von Algorithmen

- Untersucht werden die Eingenschaften von Algorithmen in Abhängigkeit von den Eingabedaten.
- Wesentliche Kenngröße ist die Anzahl der Daten, oft mit n oder m bezeichnet.
  - *Beispiel:* Sortieren von n Zahlen
  - *Beispiel:* Berechnen eines Euler-Zyklus von einem Graphen mit n Knoten und m Kanten
  - *Beispiel:* Tiefensuche in Graphen mit n Knoten und m Kanten
  - *Beispiel:* Suchen eines Elementes in n vorhandenen Einträgen
- Zentrale Frage: wie hängt die Laufzeit von n (oder m) ab?
- Von Interesse ist lediglich die **Größenordnung**!
- Diese Größenordnung wird mit der Bachmann-Landau-Notation, oder kurz O-**Notation** dargestellt.

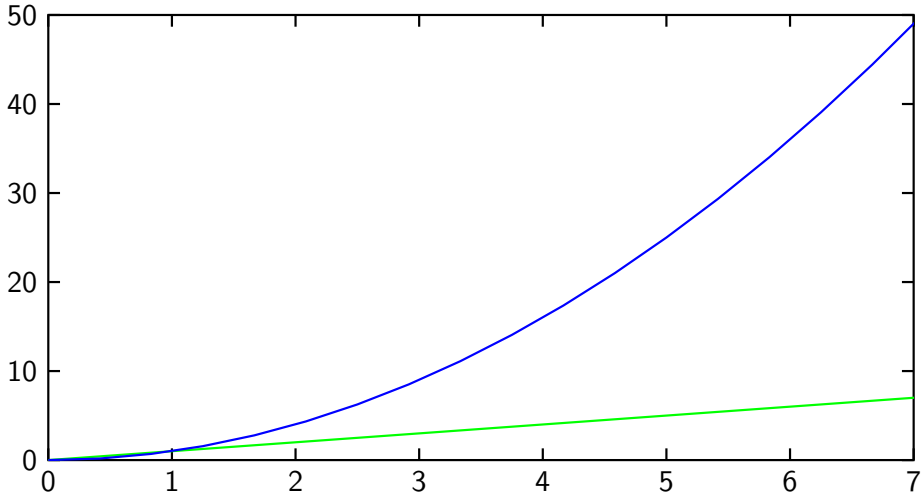
Komplexität von Algorithmen

- Ein sehr günstiges Laufzeitverhalten ist, wenn bei n Datensätzen **ca.** n Schritte ausgeführt werden.
  - Man nennt dies: **lineares** Laufzeitverhalten
  - Der Algorithmus durchläuft $O(n)$ Schritte
- Werden ungefähr n^2 viele Schritte ausgeführt:
  - **Quadratisches** Laufzeitverhalten
  - Der Algorithmus durchläuft $O(n^2)$ Schritte
- Höhere Laufzeiten sind oftmals in der Praxis problematisch, selbst bei quadratischer Laufzeit ist oft schon Vorsicht angebracht

Definition ((informell) Laufzeit eines Algorithmus anhand der O-Notation)

Ein Algorithmus A hat bei n Eingabedaten eine Laufzeit von $O(f(n))$, wenn bei seiner Ausführung *in etwa* ("in der Größenordnung von") $f(n)$ Schritte ausgeführt werden.

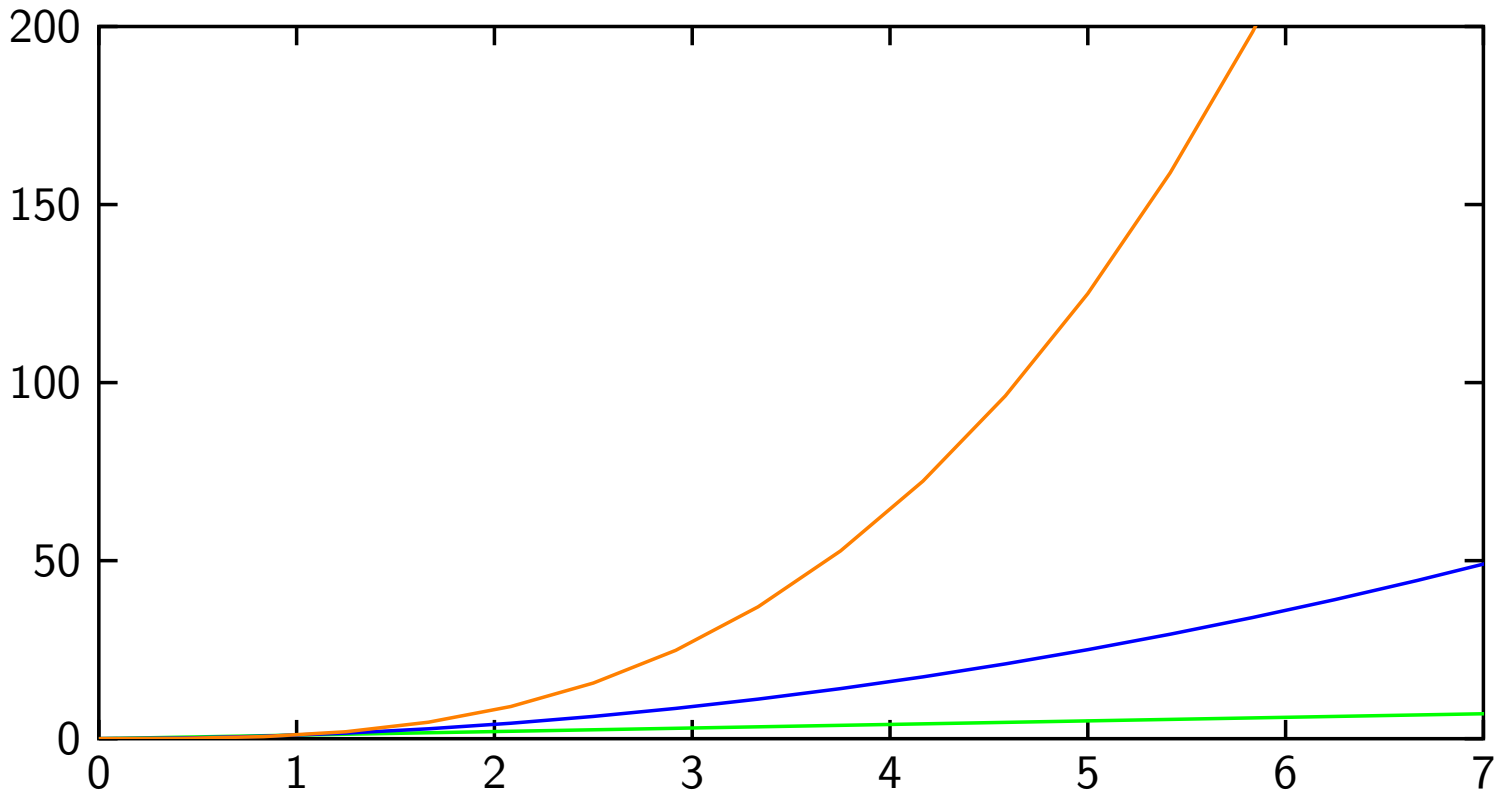
Wachstum von $f(n)$



$f(n) = n$, $f(n) = n^2$

Vergleich von linearem zu quadratischem Laufzeitverhalten

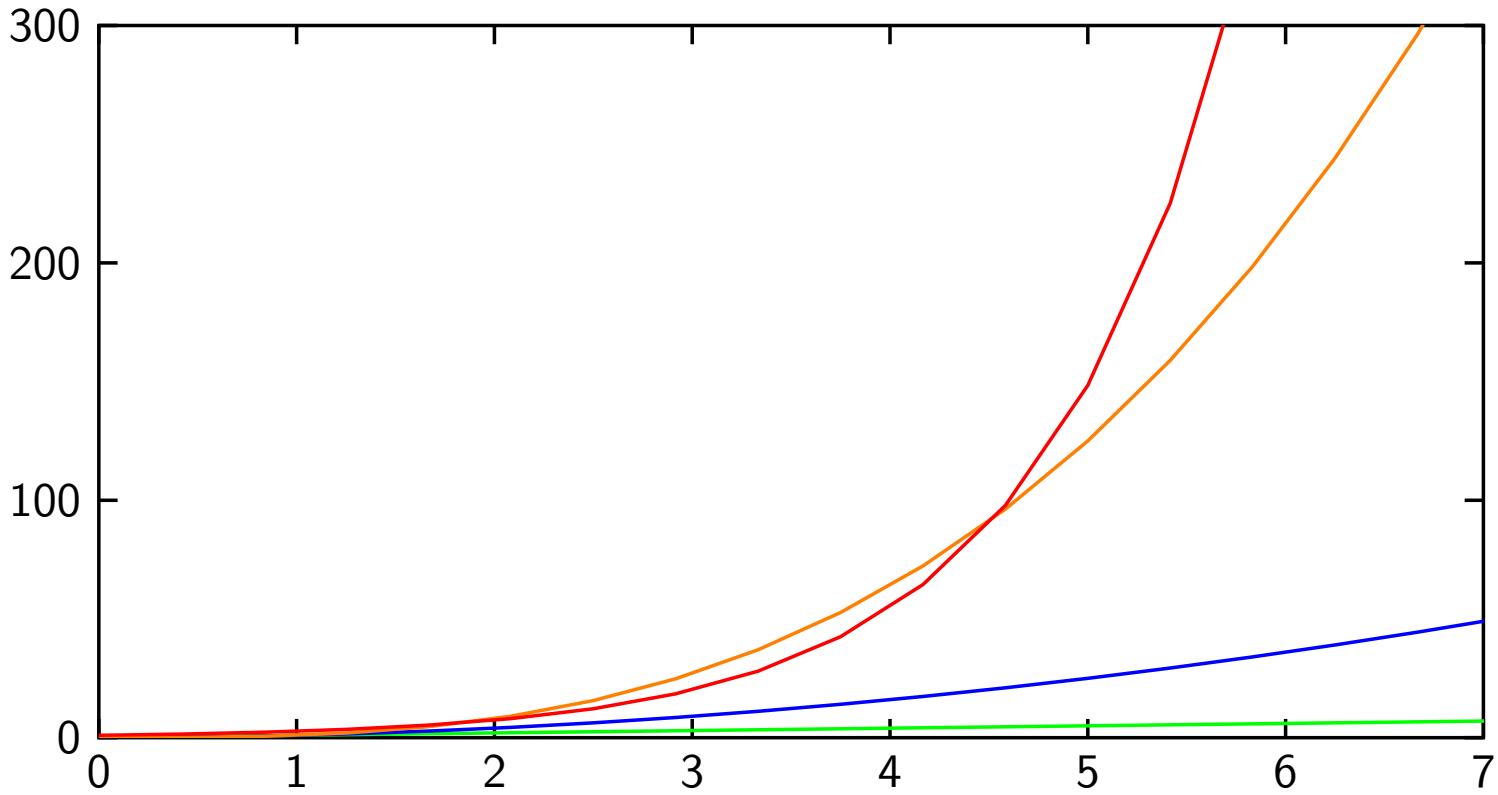
Wachstum von $f(n)$



$f(n) = n$, $f(n) = n^2$, $f(n) = n^3$

Vergleich von n, n^2 und n^3

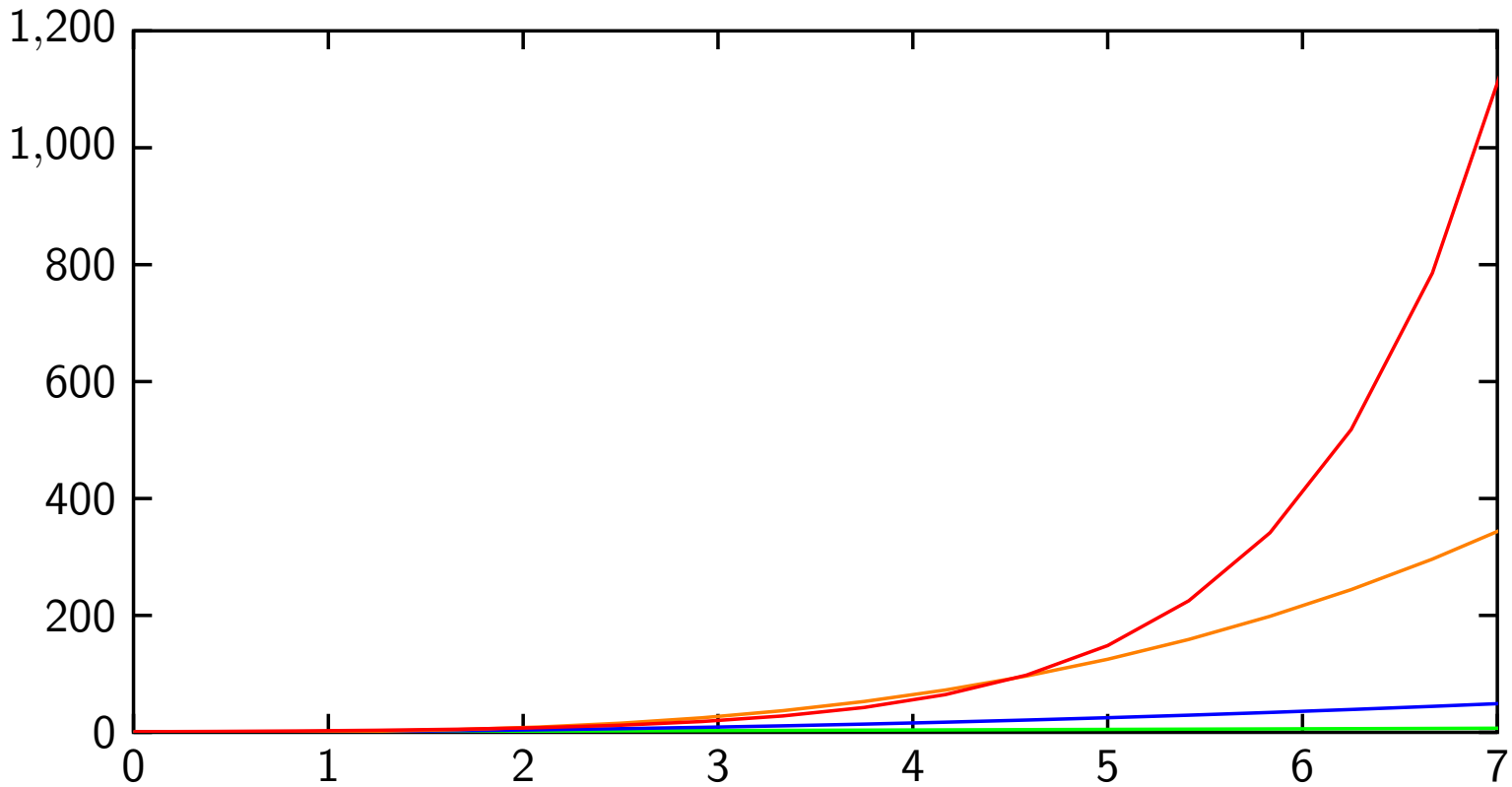
Wachstum von $f(n)$



$f(n) = n$, $f(n) = n^2$, $f(n) = n^3$, $f(n) = e^n$

Zusätzliche exponentielle Kurve, zunächst mit $y_{max} = 300$

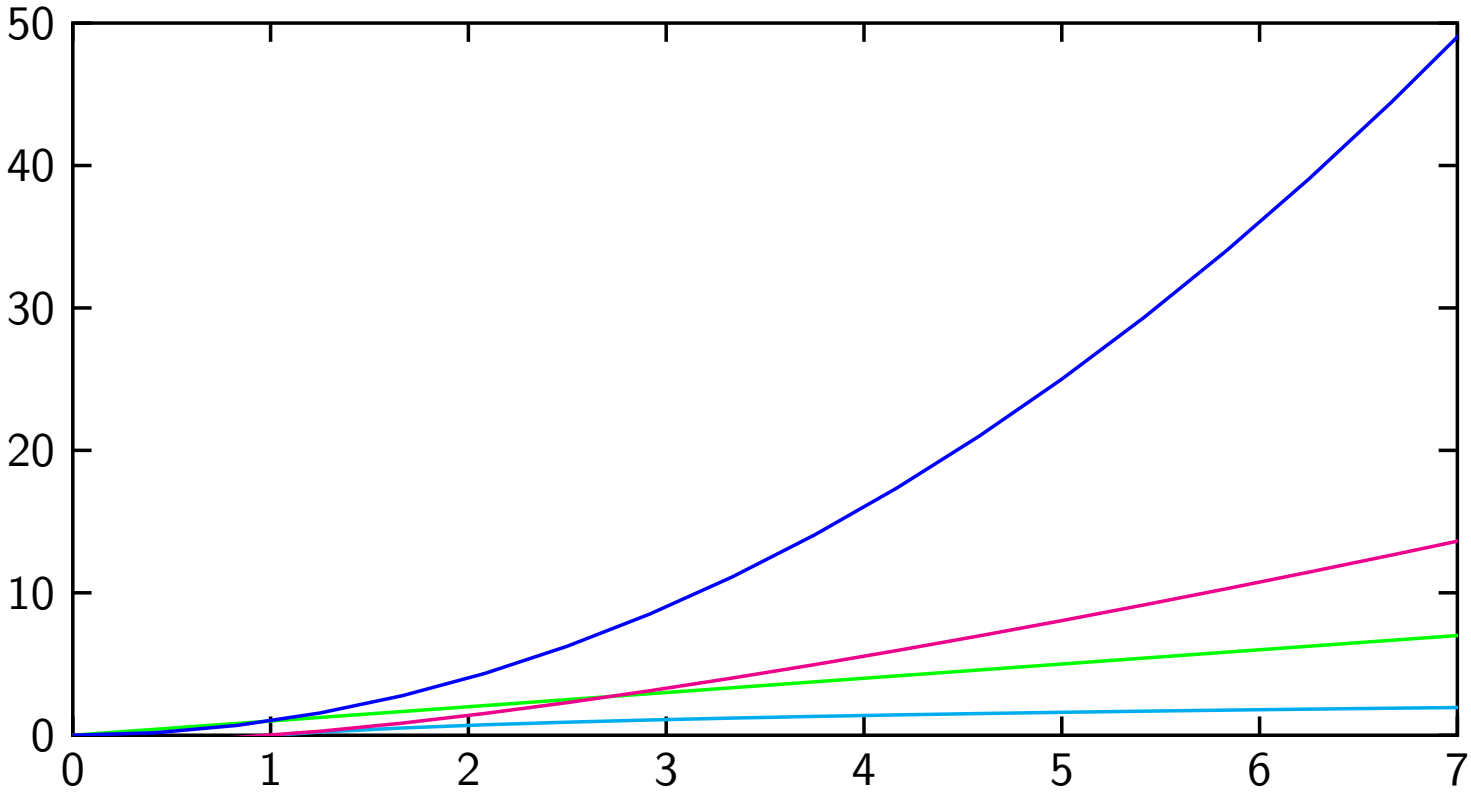
Wachstum von $f(n)$



$f(n) = n$, $f(n) = n^2$, $f(n) = n^3$, $f(n) = e^n$

Zusätzliche exponentielle Kurve, jetzt mit $y_{max} = 1200$

Wachstum von $f(n)$



$f(n) = n$, $f(n) = n^2$, $f(n) = \ln n$, $f(n) = n \ln n$

Weiteres Beispiel: die Logarithmus-Funktion. Diese strebt zwar gegen Unendlich, aber extrem langsam! Wichtige Konsequenz: $n \ln n$ verhält sich "fast" wie n

Bachmann-Landau Notation

Eine Funktion $f(n)$ liegt in $O(g(n))$ wenn ab einem Wert n_0 und einer Konstante c die Funktionswerte von $f(n)$ kleiner sind als jene von $c \cdot g(n)$. Man schreibt hierfür auch $f(n) \in O(g(n))$.

Definition (Bachmann-Landau O-Notation)

$$O(g(n)) = \{f(n) \mid (\exists c, n_0 > 0), (\forall n \geq n_0) : 0 \leq f(n) \leq cg(n)\}$$

Somit ist $c \cdot g(n)$ ab n_0 eine **obere Schranke** für $f(n)$! Es müssen immer nur die Terme der höchsten Ordnung betrachtet werden, und diese ohne etwaige konstante Faktoren.

Bachmann-Landau Notation

Beispiele:

- $3n^2 \in O(n^2)$
- $\frac{3}{4}n^3 + 5n^2 \in O(n^3)$
- $42n + \ln n + 5150 \in O(n)$
- $n \log n \in O(n^2)$
- $n \log n \in O(n \log n)$ (!) die kleinste obere Schranke!
- $n + m^2 \in O(n + m^2)$