Kapitel ADS:II

II. Algorithm Engineering

- ❑ Problemklassen und Lösungsstrategien
- ❑ Phasen des Algorithm Engineering
- ❑ Exkurs: Programmiersprachen
- ❑ Pseudocode
- ❑ Rekursion
- ❑ Maschinenmodell
- ❑ Laufzeitanalyse
- ❑ Asymptotische Analyse
- ❑ Algorithmenimplementierung
- ❑ Algorithmenevaluierung

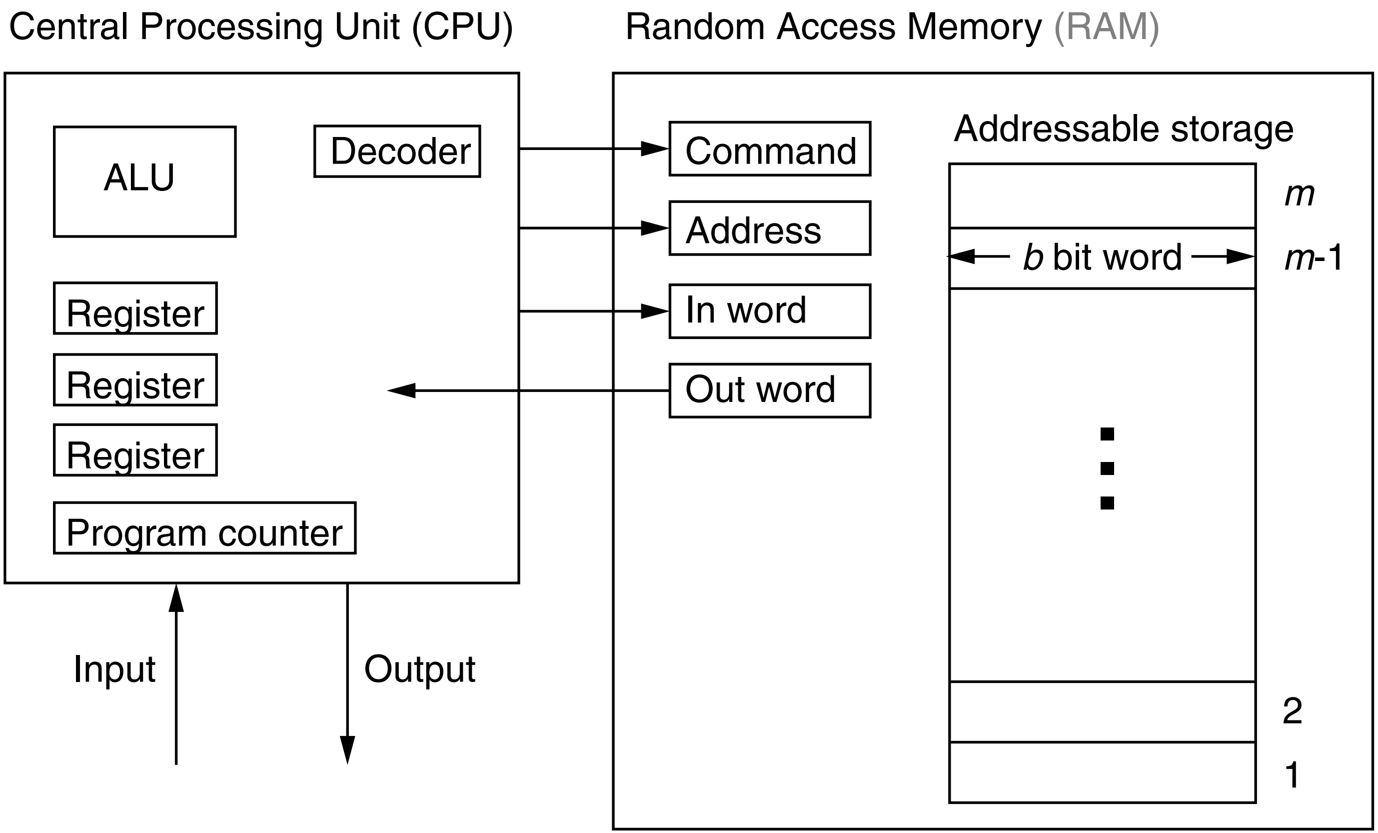
Maschinenmodell

Übersicht über Berechenbarkeitsmodelle

- Abakus
  Ältestes bekanntes mechanisches Hilfsmittel zum Rechnen
- μ-rekursive Funktionen
  Mathematisches Rechnen
- λ-Kalkül
  Funktionale Sprachen, LISP
- Logische Repräsentierbarkeit
  Logikprogrammierung, PROLOG
- Typ-0 Grammatiken / Markov-Algorithmen
  Regelbasierte Sprachen
- Turingmaschine
  Rechnen mit Papier und Bleistift
- Nichtdeterministische Turingmaschine
  Parallelismus / Quantencomputer
- Registermaschine
  Assembler / Maschinenprogrammierung

Maschinenmodell

Random Access Machine (RAM) Modell



Maschinenmodell

Random Access Machine (RAM) Modell

Central Processing Unit (CPU):

- Vereinfachte Menge verfügbarer Instruktionen, analog zu realen Computern.
  - Arithmetische Operationen: Addition, Subtraktion, Multiplikation, Division, Modulo, Abrunden, Aufrunden, k-Bit-Shift links / rechts (Multiplikation / Division mit 2^k).
  - Datenverwaltung: Laden, Speichern, Kopieren.
  - Kontrollfluss: Bedingte Anweisung, Sprunganweisung, Funktionsaufruf, Ergebnisrückgabe.

- Die Ausführung einer Instruktion benötigt eine konstante Zeitspanne.

  Es wird vereinbart, dass die Ausführung einer Instruktion je c Zeiteinheiten benötigt, wobei c eine Konstante ist.

Random Access Memory (RAM):

- Primitive Datentypen

  Ganze Zahlen (*Integer*), Gleitkommazahlen (*Floating Point Numbers*)

- Beschränkte Wortgröße

  Annahme, dass bei n Eingabezahlen jede Zahl mit $b = c \cdot \log_2 n$ Bits kodiert wird, wobei $c \geq 1$ eine Konstante ist. Die Wortgröße kann nicht beliebig wachsen.

Bemerkungen:

- Moderne CPUs bieten weitaus größere Mengen an Instruktionen an als hier vorausgesetzt. Diese betreffen weniger die theoretische Analyse von Algorithmen als ihre Umsetzung in der Praxis: Laufzeitunterschiede können aus Unkenntnis über spezifisch von der Hardware angebotene Funktionen erwachsen.

- CPUs durchlaufen einen Befehlszyklus zur Ausführung von Programmen. Dieser besteht aus 4 Schritten:

  1. Befehl laden: Der nächste auszuführende Befehl wird anhand des Programmzählers bestimmt und aus dem Random-Access-Memory in die Register der CPU geladen.
  2. Befehl dekodieren und Operanden laden: Der Befehlscode wird dekodiert und die Operanden der auszuführenden Instruktion aus dem Random-Access-Memory geladen.
  3. Befehl ausführen: Der Befehl wird unter anderen mit Hilfe der Arithmetisch-logischen Einheit (ALU) ausgeführt.
  4. Ergebnis speichern: Das Ergebnis wird im Hauptspeicher an der gewünschten Adresse abgelegt und der Programmzähler gemäß des Kontrollflusses aktualisiert.

- Verschiedene Instruktion benötigen auch unterschiedlich viele CPU-Befehlszyklen. Diese Details werden im RAM-Modell abstrahiert.

- Die Präzision der Darstellung von Gleitkommazahlen, die für numerische Anwendungen von großer Bedeutung ist, wird im RAM-Modell nicht betrachtet.

- CPUs im Speziellen und Computer im Allgemeinen haben eine Hierarchie von Speichern, die dem schnellen wiederholten Zugriff auf externen Speicher wie Festplatten dienen. Diese Speicherhierarchien werden, mit wenigen Ausnahmen, nicht betrachtet.

Die Laufzeit eines Algorithmus errechnet sich wie folgt:

$$\sum_{\text{Anweisungen}} (\text{Kosten der Anweisung}) \cdot (\text{Anzahl der Ausführungen})$$

Primitive Anweisungen:

- Anweisungen, die als Instruktionen von einer CPU bereitgestellt werden.
- Anweisungen, die unabhängig von den Daten, die zu verarbeiten sind, eine konstante Zahl von CPU-Instruktionen nach sich ziehen.

Komplexe Anweisungen:

- Funktionsaufrufe
  Der Aufruf selbst wird in konstanter Zeit ausgeführt, aber die Laufzeit der Funktion kann mit ihren Parametern variieren.
- Unkonventionelle Anweisungen
  Pseudocode erlaubt sprachliche Anweisungen wie “Sortiere das Array”, was der Ausführung des besten verfügbaren Algorithmus gleichkommt.

Laufzeitanalyse

Abhängige Variablen: Größe der Probleminstanz

Die Laufzeit eines Algorithmus ist von der Größe der Probleminstanz (Eingabegröße) abhängig.

Wir vereinbaren, dass $n \in \mathbf{N}$ die Größe einer Instanz bezeichnet.

Die Bestimmung von n ist abhängig vom Problem. Beispiele:

- Anzahl zu sortierender Zahlen
- Anzahl zu durchsuchender Zahlen
- Summe der Bits zweier zu multiplizierender Zahlen
- Zahl der Knoten und Zahl der Kanten in einem Graphen
- ...

Laufzeitanalyse

Abhängige Variablen: Struktur der Probleminstanz

Die Laufzeit eines Algorithmus ist von der Struktur der Probleminstanz abhängig.

Es werden drei Fälle unterschieden:

- Worst Case (Schlimmstfall)
  Struktur, die die Laufzeit des Algorithmus maximiert.
- Average Case (Durchschnittsfall)
  Struktur, die eine durchschnittliche Laufzeit hervorruft, gemessen an der Häufigkeit des Vorkommens von Instanzen unterschiedlicher Strukturen in der Praxis.
- Best Case (Bestfall)
  Struktur, die die Laufzeit des Algorithmus minimiert.

Die Worst-Case-Laufzeit ist eine

- obere Schranke für die Laufzeit des Algorithmus in der Praxis.
- untere Schranken für die Laufzeit mit der ein Problem gelöst werden kann, sofern der Algorithmus die niedrigste bekannte Worst-Case-Laufzeit erzielt.

Die Average-Case-Laufzeit informiert über zu erwartende Kosten in der Praxis.

Bemerkungen:

- In der Algorithmenanalyse konzentriert man sich häufig auf den Worst Case, da in der Praxis die meisten Systeme robust und hochverfügbar sein sollen und die Worst-Case-Laufzeit eine Garantie für maximal zu erwartende Laufzeit darstellt.

- Probleme werden mitunter anhand der Worst-Case-Laufzeit des besten verfügbaren Algorithmus für das Problem in sogenannte Komplexitätsklassen eingeteilt.

- Die Average-Case-Laufzeit lässt sich oftmals nur schwer abschätzen bzw. kann nur bei Verfügbarkeit von Informationen über den spezifischen Anwendungsfall eines Algorithmus ermittelt werden. Oft ist die durchschnittliche Struktur einer Probleminstanz nicht signifikant verschieden von der schlimmstmöglichen Struktur.

Laufzeitanalyse

Beispiel

InsertionSort(A)

FOR $j = 2$ **TO** n **DO**
    $a_j = A[j]$
    $i = j - 1$
    WHILE $i > 0$ **AND** $A[i] > a_j$ **DO**
        $A[i+1] = A[i]$
        $i = i - 1$
    ENDDO
    $A[i+1] = a_j$
ENDDO

Laufzeitanalyse

Beispiel

InsertionSort(A)

FOR $j = 2$ **TO** n **DO**

  $a_j = A[j]$

  $i = j - 1$

  WHILE $i > 0$ **AND** $A[i] > a_j$ **DO**

    $A[i+1] = A[i]$

    $i = i - 1$

  ENDDO

  $A[i+1] = a_j$

ENDDO

Laufzeitfunktion:

$$T(n) = \; c_1 \; \cdot \; n$$

Laufzeitanalyse

Beispiel

InsertionSort(A)

FOR $j = 2$ **TO** n **DO**
  $a_j = A[j]$
  $i = j - 1$
  WHILE $i > 0$ **AND** $A[i] > a_j$ **DO**
    $A[i+1] = A[i]$
    $i = i - 1$
  ENDDO
  $A[i+1] = a_j$
ENDDO

Laufzeitfunktion:

$$T(n) = c_1 \cdot n + c_2 \cdot (n-1)$$

Laufzeitanalyse

Beispiel

InsertionSort(A)

```
FOR j = 2 TO n DO
    a_j = A[j]
    i = j − 1
    WHILE i > 0 AND A[i] > a_j DO
        A[i + 1] = A[i]
        i = i − 1
    ENDDO
    A[i + 1] = a_j
ENDDO
```

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \;& c_1 \cdot n \\
+\;& c_2 \cdot (n-1) \\
+\;& c_3 \cdot (n-1) \\
+\;& c_8 \cdot (n-1) \\
+\;& c_9 \cdot (n-1)
\end{aligned}
$$

Laufzeitanalyse

Beispiel

InsertionSort(A)

FOR $j = 2$ **TO** n **DO**
  $a_j = A[j]$
  $i = j - 1$
  WHILE $i > 0$ **AND** $A[i] > a_j$ **DO**
    $A[i+1] = A[i]$
    $i = i - 1$
  ENDDO
  $A[i+1] = a_j$
ENDDO

Laufzeitfunktion:

$$\begin{aligned} T(n) = \; & c_1 \cdot n \\ & + c_2 \cdot (n-1) \\ & + c_3 \cdot (n-1) \\ & + c_4 \cdot \textstyle\sum_{j=2}^{n} t_j \\ & + c_8 \cdot (n-1) \\ & + c_9 \cdot (n-1) \end{aligned}$$

Laufzeitanalyse

Beispiel

InsertionSort(A)

FOR $j = 2$ **TO** n **DO**
    $a_j = A[j]$
    $i = j - 1$
    WHILE $i > 0$ **AND** $A[i] > a_j$ **DO**
        $A[i+1] = A[i]$
        $i = i - 1$
    ENDDO
    $A[i+1] = a_j$
ENDDO

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \;& c_1 \cdot n \\
+\;& c_2 \cdot (n-1) \\
+\;& c_3 \cdot (n-1) \\
+\;& c_4 \cdot \textstyle\sum_{j=2}^{n} t_j \\
+\;& c_5 \cdot \textstyle\sum_{j=2}^{n} (t_j - 1) \\
+\;& c_6 \cdot \textstyle\sum_{j=2}^{n} (t_j - 1) \\
+\;& c_7 \cdot \textstyle\sum_{j=2}^{n} (t_j - 1) \\
+\;& c_8 \cdot (n-1) \\
+\;& c_9 \cdot (n-1)
\end{aligned}
$$

Laufzeitanalyse

Beispiel

InsertionSort(A)

FOR $j = 2$ **TO** n **DO**
    $a_j = A[j]$
    $i = j - 1$
    WHILE $i > 0$ **AND** $A[i] > a_j$ **DO**
        $A[i+1] = A[i]$
        $i = i - 1$
    ENDDO
    $A[i+1] = a_j$
ENDDO

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \; & c_1 \cdot n \\
+ \; & c_2 \cdot (n-1) \\
+ \; & c_3 \cdot (n-1) \\
+ \; & c_4 \cdot \textstyle\sum_{j=2}^{n} t_j \\
+ \; & c_5 \cdot \textstyle\sum_{j=2}^{n} (t_j - 1) \\
+ \; & c_6 \cdot \textstyle\sum_{j=2}^{n} (t_j - 1) \\
+ \; & c_7 \cdot \textstyle\sum_{j=2}^{n} (t_j - 1) \\
+ \; & c_8 \cdot (n-1) \\
+ \; & c_9 \cdot (n-1)
\end{aligned}
$$

Best Case:

$$t_j = 1 \quad \rightsquigarrow \quad \sum_{j=2}^{n} 1 \; = \; n - 1$$

Laufzeitanalyse

Beispiel

InsertionSort(A)

FOR $j = 2$ **TO** n **DO**
    $a_j = A[j]$
    $i = j - 1$
    WHILE $i > 0$ **AND** $A[i] > a_j$ **DO**
        $A[i+1] = A[i]$
        $i = i - 1$
    ENDDO
    $A[i+1] = a_j$
ENDDO

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \;& c_1 \cdot n \\
+\;& c_2 \cdot (n-1) \\
+\;& c_3 \cdot (n-1) \\
+\;& c_4 \cdot (n-1) \\
+\;& c_5 \cdot 0 \\
+\;& c_6 \cdot 0 \\
+\;& c_7 \cdot 0 \\
+\;& c_8 \cdot (n-1) \\
+\;& c_9 \cdot (n-1)
\end{aligned}
$$

Best Case:

$$t_j = 1 \quad \rightsquigarrow \quad \sum_{j=2}^{n} 1 \;=\; n - 1$$

Laufzeitanalyse

Beispiel

InsertionSort(A)

```
FOR j = 2 TO n DO
    a_j = A[j]
    i = j − 1
    WHILE i > 0 AND A[i] > a_j DO
        A[i + 1] = A[i]
        i = i − 1
    ENDDO
    A[i + 1] = a_j
ENDDO
```

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \; & c_1 \cdot n \\
+ \; & c_2 \cdot (n-1) \\
+ \; & c_3 \cdot (n-1) \\
+ \; & c_4 \cdot (n-1) \\
+ \; & c_5 \cdot 0 \\
+ \; & c_6 \cdot 0 \\
+ \; & c_7 \cdot 0 \\
+ \; & c_8 \cdot (n-1) \\
+ \; & c_9 \cdot (n-1)
\end{aligned}
$$

Best Case:

$$t_j = 1 \quad \rightsquigarrow \quad \sum_{j=2}^{n} 1 \; = \; n - 1$$

	1	2	3	4	5	6
A	1	2	3	4	5	6

Laufzeitanalyse

Beispiel

InsertionSort(A)

```
FOR j = 2 TO n DO
    a_j = A[j]
    i = j - 1
    WHILE i > 0 AND A[i] > a_j DO
        A[i+1] = A[i]
        i = i - 1
    ENDDO
    A[i+1] = a_j
ENDDO
```

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \; & c_1 \cdot n \\
+ \; & c_2 \cdot (n-1) \\
+ \; & c_3 \cdot (n-1) \\
+ \; & c_4 \cdot (n-1) \\
+ \; & c_5 \cdot 0 \\
+ \; & c_6 \cdot 0 \\
+ \; & c_7 \cdot 0 \\
+ \; & c_8 \cdot (n-1) \\
+ \; & c_9 \cdot (n-1)
\end{aligned}
$$

Best Case:

$$T(n) = \underbrace{(c_1 + c_2 + c_3 + c_4 + c_8 + c_9)}_{a} \cdot n \underbrace{- (c_2 + c_3 + c_4 + c_8 + c_9)}_{b}$$

Laufzeitanalyse

Beispiel

InsertionSort(A)

```
FOR j = 2 TO n DO
    a_j = A[j]
    i = j - 1
    WHILE i > 0 AND A[i] > a_j DO
        A[i+1] = A[i]
        i = i - 1
    ENDDO
    A[i+1] = a_j
ENDDO
```

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \;& c_1 \cdot n \\
&+ c_2 \cdot (n-1) \\
&+ c_3 \cdot (n-1) \\
&+ c_4 \cdot (n-1) \\
&+ c_5 \cdot 0 \\
&+ c_6 \cdot 0 \\
&+ c_7 \cdot 0 \\
&+ c_8 \cdot (n-1) \\
&+ c_9 \cdot (n-1)
\end{aligned}
$$

Best Case:

$$T(n) = a \cdot n + b$$ (lineare Funktion)

Laufzeitanalyse

Beispiel

InsertionSort(A)

```
FOR  j = 2  TO  n  DO
   a_j = A[j]
   i = j − 1
   WHILE  i > 0  AND  A[i] > a_j  DO
      A[i + 1] = A[i]
      i = i − 1
   ENDDO
   A[i + 1] = a_j
ENDDO
```

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \; & c_1 \cdot n \\
+ \; & c_2 \cdot (n-1) \\
+ \; & c_3 \cdot (n-1) \\
+ \; & c_4 \cdot \textstyle\sum_{j=2}^{n} t_j \\
+ \; & c_5 \cdot \textstyle\sum_{j=2}^{n} (t_j - 1) \\
+ \; & c_6 \cdot \textstyle\sum_{j=2}^{n} (t_j - 1) \\
+ \; & c_7 \cdot \textstyle\sum_{j=2}^{n} (t_j - 1) \\
+ \; & c_8 \cdot (n-1) \\
+ \; & c_9 \cdot (n-1)
\end{aligned}
$$

Worst Case:

$$t_j = j \quad \rightsquigarrow \quad \sum_{j=2}^{n} j = \frac{n \cdot (n+1)}{2} - 1$$

	1	2	3	4	5	6
A	6	5	4	3	2	1

Laufzeitanalyse

Beispiel

InsertionSort(A)

```
FOR j = 2 TO n DO
    a_j = A[j]
    i = j - 1
    WHILE i > 0 AND A[i] > a_j DO
        A[i+1] = A[i]
        i = i - 1
    ENDDO
    A[i+1] = a_j
ENDDO
```

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \;& c_1 \cdot n \\
+\;& c_2 \cdot (n-1) \\
+\;& c_3 \cdot (n-1) \\
+\;& c_4 \cdot \left(\tfrac{n\cdot(n+1)}{2} - 1\right) \\
+\;& c_5 \cdot \left(\tfrac{(n-1)\cdot n}{2}\right) \\
+\;& c_6 \cdot \left(\tfrac{(n-1)\cdot n}{2}\right) \\
+\;& c_7 \cdot \left(\tfrac{(n-1)\cdot n}{2}\right) \\
+\;& c_8 \cdot (n-1) \\
+\;& c_9 \cdot (n-1)
\end{aligned}
$$

Worst Case:

$$t_j = j \quad \rightsquigarrow \quad \sum_{j=2}^{n} j = \frac{n\cdot(n+1)}{2} - 1$$

	1	2	3	4	5	6
A	6	5	4	3	2	1

Laufzeitanalyse

Beispiel

InsertionSort(A)

```
1. FOR j = 2 TO n DO
2.     a_j = A[j]
3.     i = j - 1
4.     WHILE i > 0 AND A[i] > a_j DO
5.         A[i+1] = A[i]
6.         i = i - 1
7.     ENDDO
8.     A[i+1] = a_j
9. ENDDO
```

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \; & c_1 \cdot n \\
+ \; & c_2 \cdot (n-1) \\
+ \; & c_3 \cdot (n-1) \\
+ \; & c_4 \cdot \left(\frac{n \cdot (n+1)}{2} - 1\right) \\
+ \; & c_5 \cdot \left(\frac{(n-1) \cdot n}{2}\right) \\
+ \; & c_6 \cdot \left(\frac{(n-1) \cdot n}{2}\right) \\
+ \; & c_7 \cdot \left(\frac{(n-1) \cdot n}{2}\right) \\
+ \; & c_8 \cdot (n-1) \\
+ \; & c_9 \cdot (n-1)
\end{aligned}
$$

Worst Case:

$$
T(n) = \underbrace{\left(\frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)}_{a} \cdot n^2 \underbrace{+ \left(c_1 + c_2 + c_3 + \frac{c_4}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 + c_9\right)}_{b} \cdot n \underbrace{- (c_2 + c_3 + c_4 + c_8 + c_9)}_{c}
$$

Laufzeitanalyse

Beispiel

InsertionSort(A)

```
FOR j = 2 TO n DO
    a_j = A[j]
    i = j - 1
    WHILE i > 0 AND A[i] > a_j DO
        A[i + 1] = A[i]
        i = i - 1
    ENDDO
    A[i + 1] = a_j
ENDDO
```

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \; & c_1 \cdot n \\
+ \; & c_2 \cdot (n-1) \\
+ \; & c_3 \cdot (n-1) \\
+ \; & c_4 \cdot \left(\tfrac{n \cdot (n+1)}{2} - 1\right) \\
+ \; & c_5 \cdot \left(\tfrac{(n-1) \cdot n}{2}\right) \\
+ \; & c_6 \cdot \left(\tfrac{(n-1) \cdot n}{2}\right) \\
+ \; & c_7 \cdot \left(\tfrac{(n-1) \cdot n}{2}\right) \\
+ \; & c_8 \cdot (n-1) \\
+ \; & c_9 \cdot (n-1)
\end{aligned}
$$

Worst Case:

$$T(n) = a \cdot n^2 + b \cdot n + c$$ (quadratische Funktion)

Laufzeitanalyse

Beispiel

InsertionSort(A)

```
FOR j = 2 TO n DO
    a_j = A[j]
    i = j - 1
    WHILE i > 0 AND A[i] > a_j DO
        A[i+1] = A[i]
        i = i - 1
    ENDDO
    A[i+1] = a_j
ENDDO
```

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \; & c_1 \cdot n \\
+ \; & c_2 \cdot (n-1) \\
+ \; & c_3 \cdot (n-1) \\
+ \; & c_4 \cdot \left(\frac{n\cdot(n+1)}{2} - 1\right) \\
+ \; & c_5 \cdot \left(\frac{(n-1)\cdot n}{2}\right) \\
+ \; & c_6 \cdot \left(\frac{(n-1)\cdot n}{2}\right) \\
+ \; & c_7 \cdot \left(\frac{(n-1)\cdot n}{2}\right) \\
+ \; & c_8 \cdot (n-1) \\
+ \; & c_9 \cdot (n-1)
\end{aligned}
$$

- Die Laufzeit von Insertion Sort ist nach unten beschränkt durch eine lineare Funktion und nach oben durch eine quadratische Funktion.

→ Die Laufzeit von Insertion Sort wächst asymptotisch höchstens wie $g(n) = n^2$.

Bemerkungen:

- Wir nehmen an, dass jede Zeile i des Algorithmus eine konstante Zeit c_i benötigt.
- Wenn eine For- oder While-Schleife auf normale Weise wegen Nichterfüllung der Schleifenbedingung endet, wurde die Schleifenbedingung ein Mal häufiger ausgeführt als der Schleifenrumpf.
- Für $j \in [2, n]$ entspricht t_j der Anzahl der Iterationen der While-Schleife.
- Nur Zeilen 4-7 sind abhängig von der Struktur der Probleminstanz. Die übrigen Zeilen hängen nur von der Größe der Probleminstanz n ab.
- Minimal wird t_j genau dann, wenn die While-Schleife für j nicht durchlaufen werden muss. Das geschieht, wenn eine der Schleifenbedingungen nicht erfüllt ist, was nur dann der Fall ist, wenn das aktuell einzusortierende Element $A[j]$ bereits an der richtigen Stelle steht.
- Maximal wird t_j genau dann, wenn die While-Schleife für j das Element $A[j]$ mit allen $j-1$ vorangehenden Elementen vertauschen muss.
- Die arithmetische Reihe: $\sum_{j=1}^{n} j = \frac{n\cdot(n+1)}{2}$. Für $k = j-1$ ist $\sum_{j=2}^{n}(j-1) = \sum_{k=1}^{n-1} k = \frac{(n-1)\cdot n}{2}$.
- Der Average Case entspricht dem Worst Case. Für eine Zufallsfolge von n Zahlen muss für j im Durchschnitt das halbe vorangehende Array durchsucht werden, um zu entscheiden, wo $A[j]$ einsortiert werden muss: Der Erwartungswert $E(t_j) = j/2$. Die Laufzeit ist also im Schnitt die Hälfte der Laufzeit des Worst Cases, also weiterhin eine quadratische Funktion.

Asymptotische Analyse
Wachstumsrate

In der asymptotischen Analyse werden Funktionen bezüglich der wesentlich zu ihrem Wachstum beitragenden Komponenten klassifiziert.

Sei $T(n)$ die Laufzeitfunktion eines Algorithmus:

$$T(n) = a \cdot n^2 + b \cdot n + c$$

Beobachtungen: [plot]

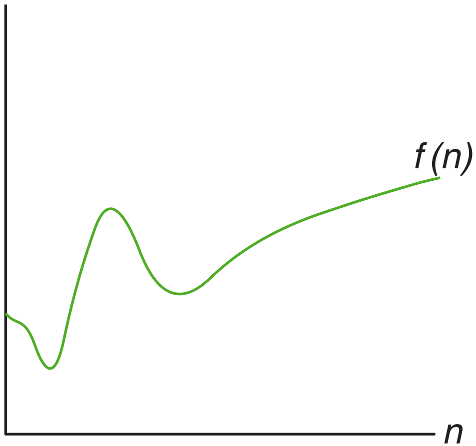
- Für $a > 0$ gibt es ein n_0 für das $a \cdot n^2 > b \cdot n + c$ wird.
- Mit wachsendem n wächst der Anteil von $a \cdot n^2$ an der Summe.
- Für $n \to \infty$ wird der Anteil von $b \cdot n + c$ an der Summe vernachlässigbar.
- Der Faktor a ist konstant und hat keinen Einfluss auf die Wachstumsrate.

Wir bezeichnen die asymptotische Wachstumsrate der Laufzeitfunktion eines Algorithmus als seine Laufzeitkomplexität. Beispiel: Die Laufzeit $T(n)$ wächst asymptotisch wie n^2 für $n \to \infty$: der Algorithmus hat quadratische Laufzeit.

Asymptotische Analyse

Bachmann-Landau-Symbole

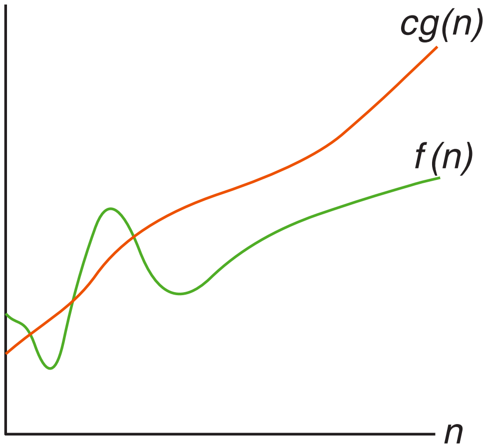
Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



Asymptotische Analyse

Bachmann-Landau-Symbole

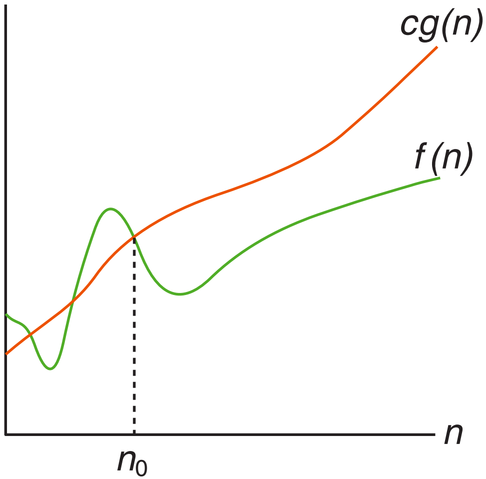
Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



Asymptotische Analyse

Bachmann-Landau-Symbole

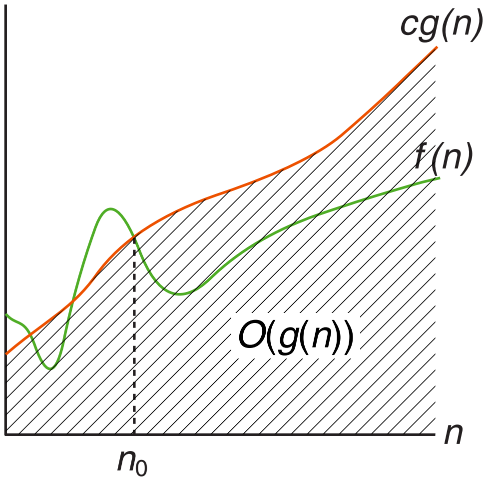
Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



Asymptotische Analyse

Bachmann-Landau-Symbole

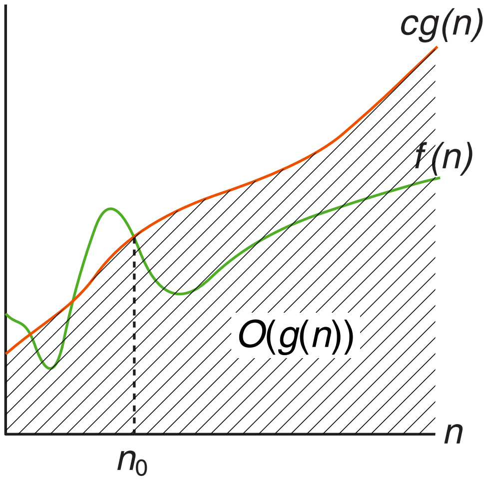
Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.

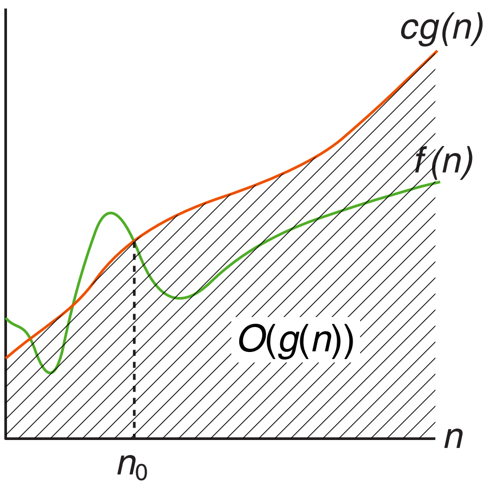


$$O(g(n)) = \{f(n) \mid \text{es existieren Konstanten } c > 0 \text{ und } n_0 > 0\text{, so dass für alle } n \geq n_0 \text{ gilt } 0 \leq f(n) \leq c \cdot g(n)\}$$

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



$$O(g(n)) = \{f(n) \mid \text{es existieren Konstanten } c > 0 \text{ und } n_0 > 0\text{, so dass für alle } n \geq n_0 \text{ gilt } 0 \leq f(n) \leq c \cdot g(n)\}$$

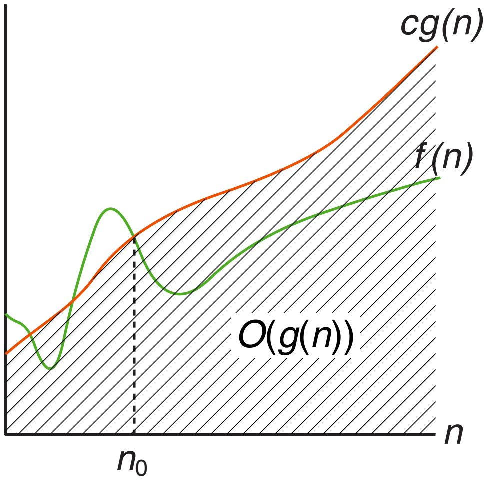
Für alle $f(n) \in O(g(n))$ gilt: $g(n)$ ist eine asymptotisch obere Schranke für $f(n)$.

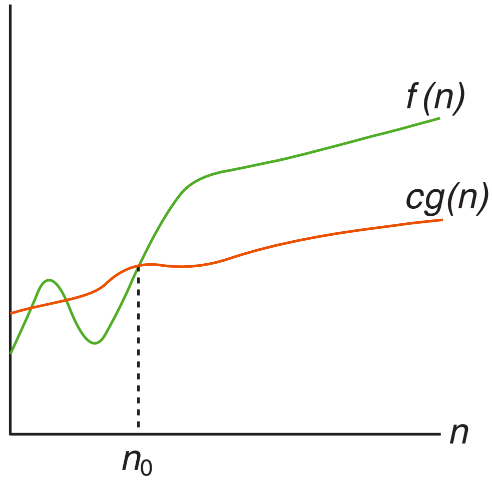
Wenn $f(n) \in O(g(n))$ schreiben wir $f(n) = O(g(n))$.

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \rightarrow \mathbf{R}_0^+$ Funktionen.

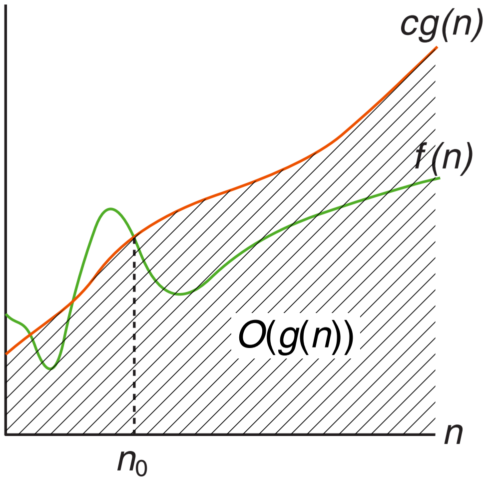


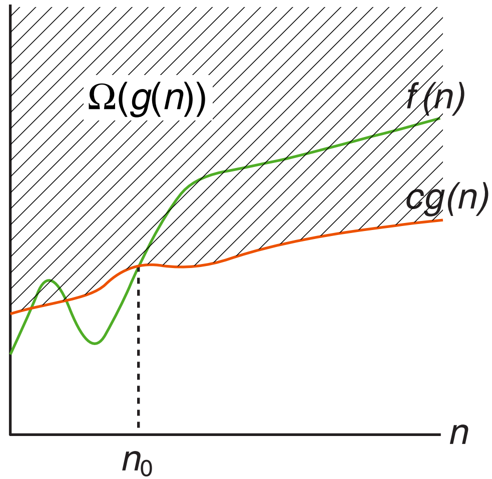


Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \rightarrow \mathbf{R}_0^+$ Funktionen.





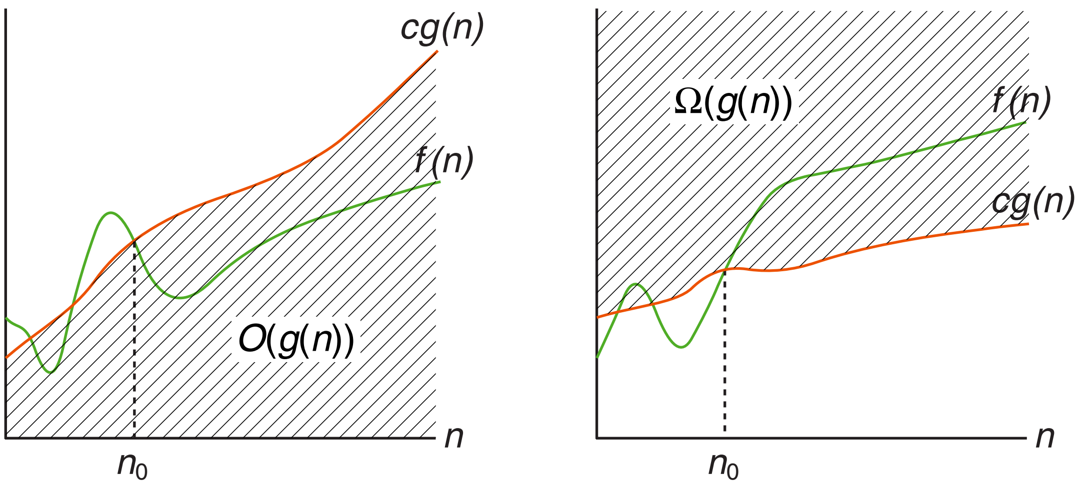
$\Omega(g(n)) = \{f(n) \ \mid$ es existieren Konstanten $c > 0$ und $n_0 > 0$, so dass

für alle $n \geq n_0$ gilt $0 \ \leq \ c \cdot g(n) \ \leq \ f(n)\}$

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



$$\Omega(g(n)) = \{f(n) \mid \text{es existieren Konstanten } c > 0 \text{ und } n_0 > 0\text{, so dass für alle } n \geq n_0 \text{ gilt } 0 \leq c \cdot g(n) \leq f(n)\}$$

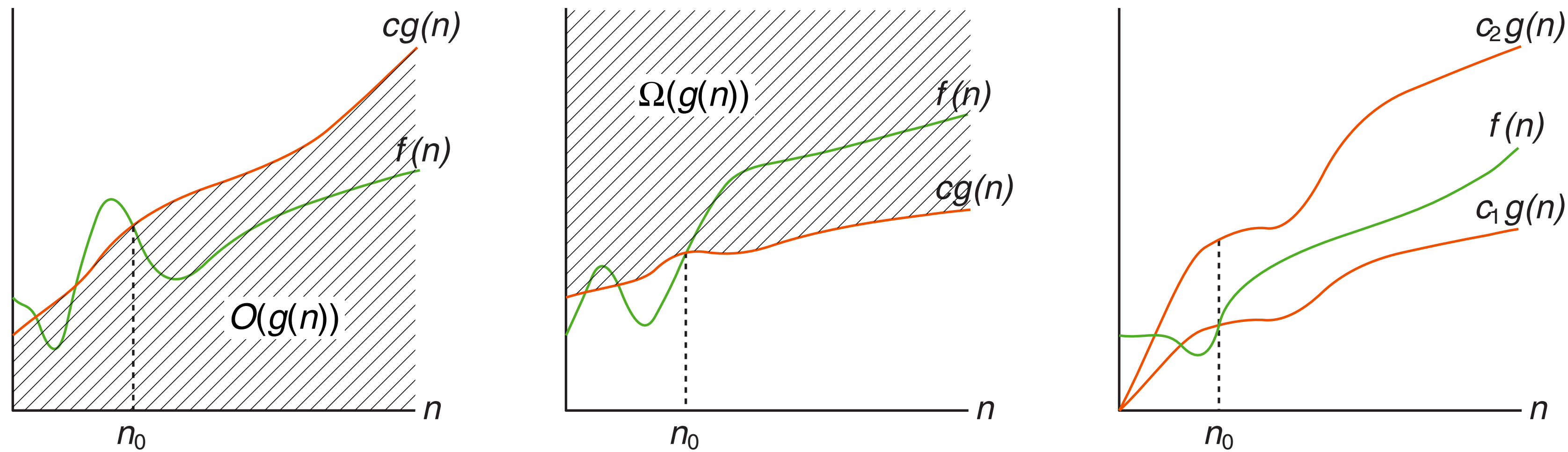
Für alle $f(n) \in \Omega(g(n))$ gilt: $g(n)$ ist eine asymptotisch untere Schranke für $f(n)$.

Wenn $f(n) \in \Omega(g(n))$ schreiben wir $f(n) = \Omega(g(n))$.

Asymptotische Analyse

Bachmann-Landau-Symbole

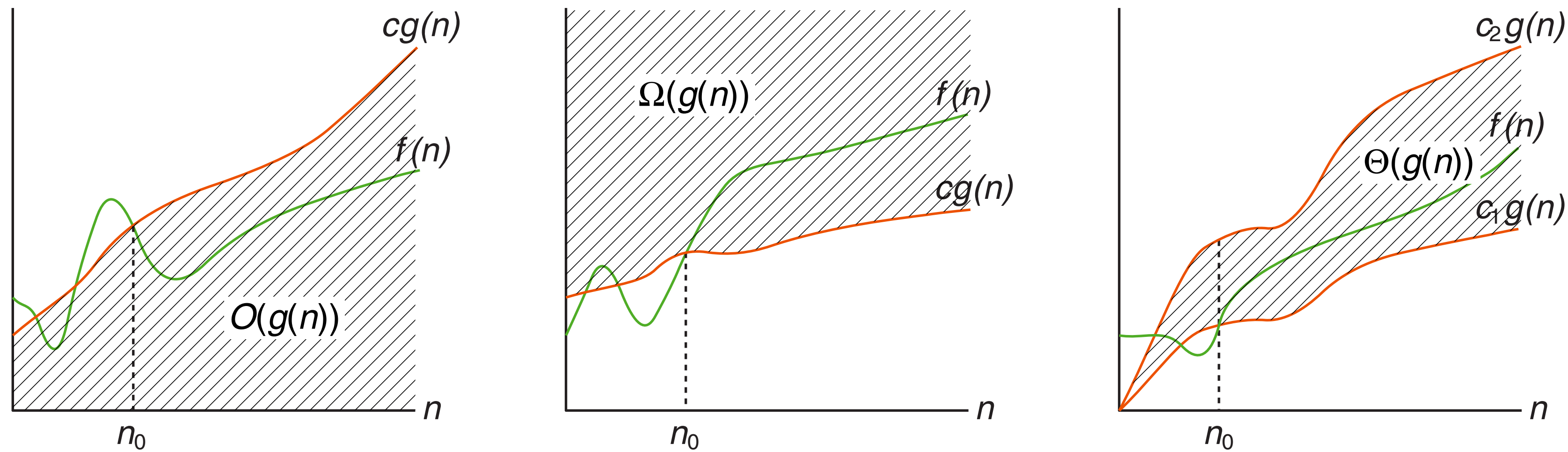
Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.

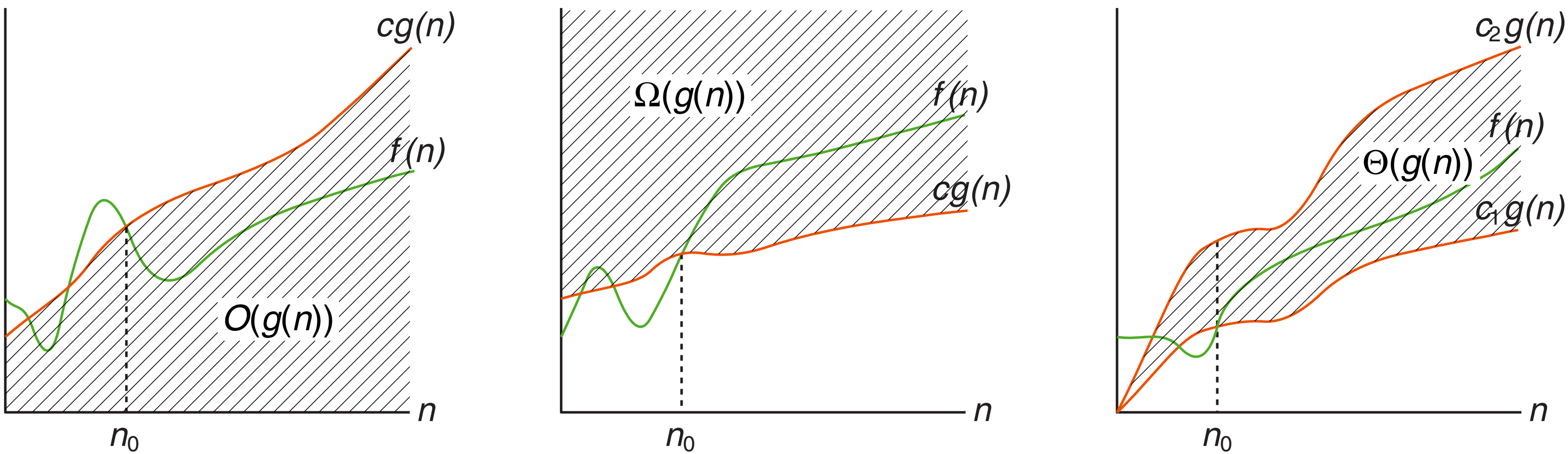


$\Theta(g(n)) = \{f(n) \mid$ es existieren Konstanten $c_1 > 0$, $c_2 > 0$ und $n_0 > 0$, so dass für alle $n \geq n_0$ gilt $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



$$\Theta(g(n)) = \{f(n) \mid \text{es existieren Konstanten } c_1 > 0, c_2 > 0 \text{ und } n_0 > 0, \text{ so dass für alle } n \geq n_0 \text{ gilt } 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$$

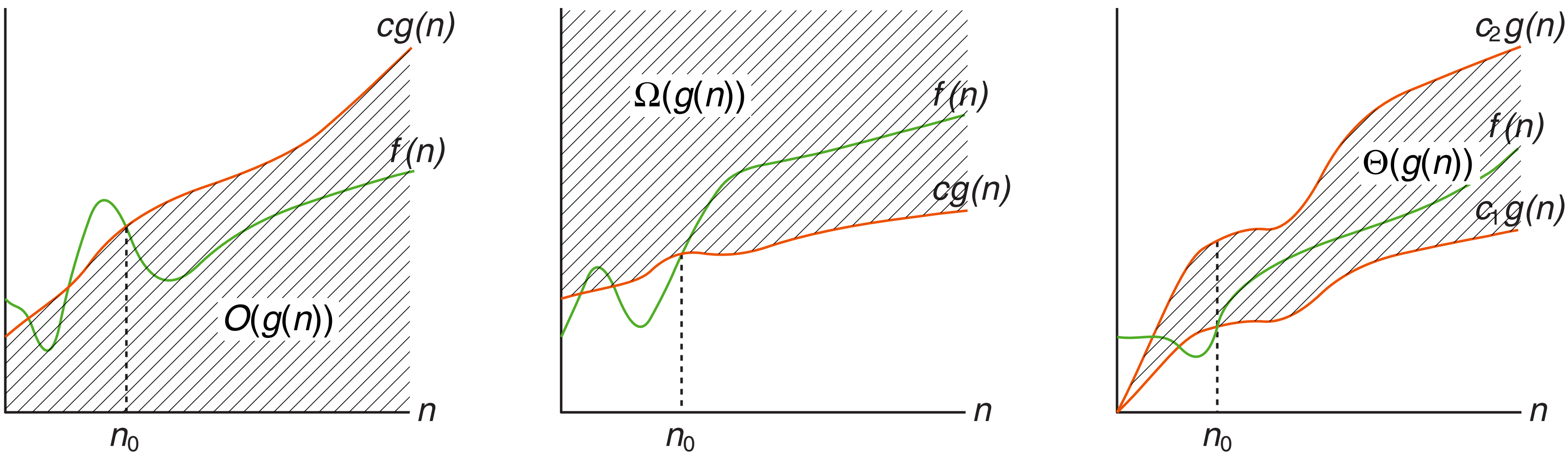
Für alle $f(n) \in \Theta(g(n))$ gilt: $g(n)$ ist eine asymptotisch scharfe Schranke für $f(n)$.

Wenn $f(n) \in \Theta(g(n))$ schreiben wir $f(n) = \Theta(g(n))$.

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \rightarrow \mathbf{R}_0^+$ Funktionen.



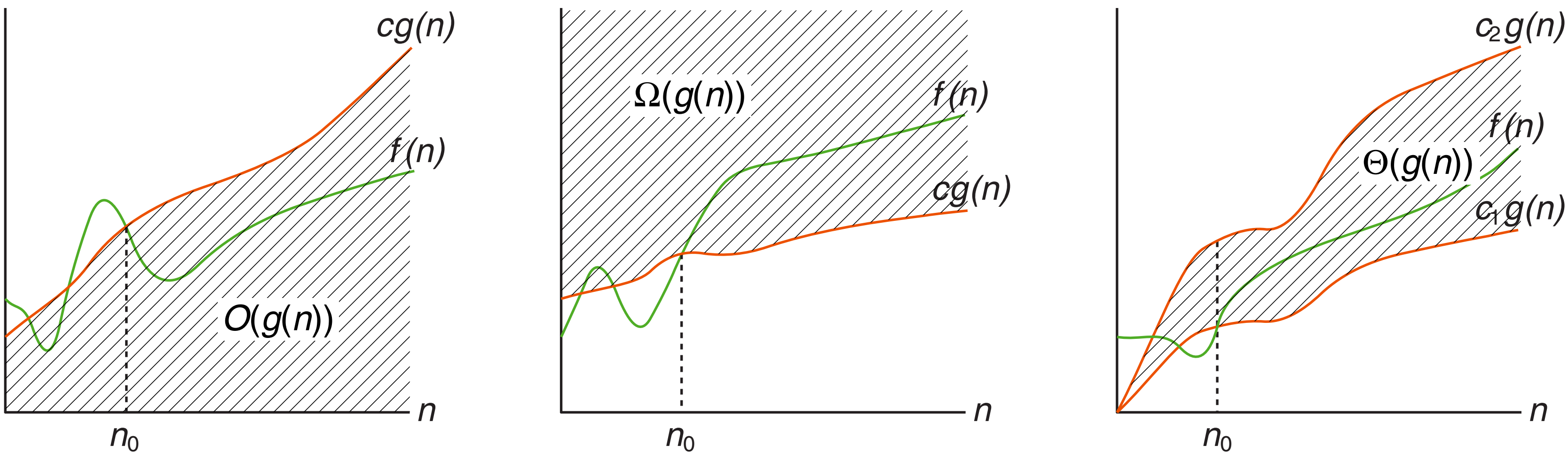
Beispiele:

- $2n^2 = O(n^3)$ mit $c = 1$ und $n_0 = 2$ [plot]
- $\sqrt{n} = \Omega(\lg n)$ mit $c = 1$ und $n_0 = 16$ [plot]
- $n^2/2 - 2n = \Theta(n^2)$ mit $c_1 = 1/4$, $c_2 = 1/2$ und $n_0 = 8$ [plot]

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



Beispiele:

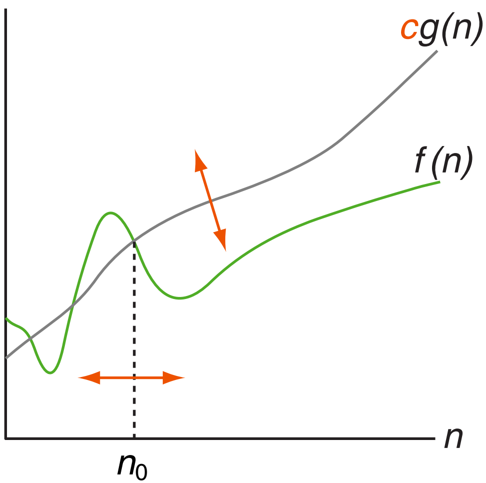
- $O(n^2) = \{n^2,\ \ n^2+n,\ \ 1000n^2+1000n,\ \ n,\ \ n^{1.99999},\ \ n^2/\lg n \lg n \lg n,\ \ \ldots\}$
- $\Omega(n^2) = \{n^2,\ \ n^2+n,\ \ 1000n^2+1000n,\ \ n^3,\ \ n^{2.00001},\ \ n^2 \lg\lg\lg n,\ \ 2^{2^n}, \ldots\}$

➜ $f(n) = \Theta(g(n))$ genau dann, wenn $f(n) = O(g(n))$ und $f(n) = \Omega(g(n))$.

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



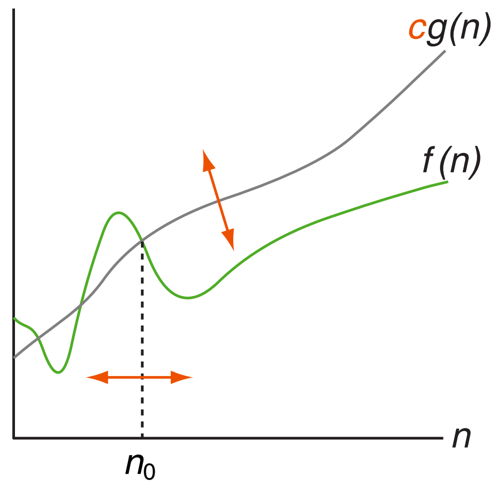
$o(g(n)) = \{f(n) \mid$ für alle Konstanten $c > 0$ existiert eine Konstante $n_0 > 0$, so dass

für alle $n \geq n_0$ gilt $0 \leq f(n) < c \cdot g(n)\}$

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



$o(g(n)) = \{f(n) \mid$ für alle Konstanten $c > 0$ existiert eine Konstante $n_0 > 0$, so dass

für alle $n \geq n_0$ gilt $0 \leq f(n) < c \cdot g(n)\}$

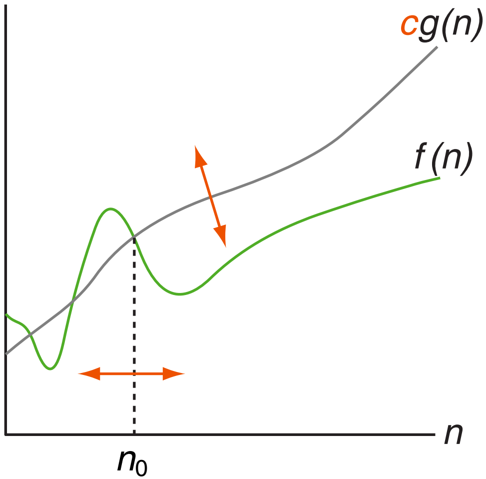
Für alle $f(n) \in o(g(n))$ gilt: $g(n)$ ist asymptotisch strikt größer als $f(n)$.

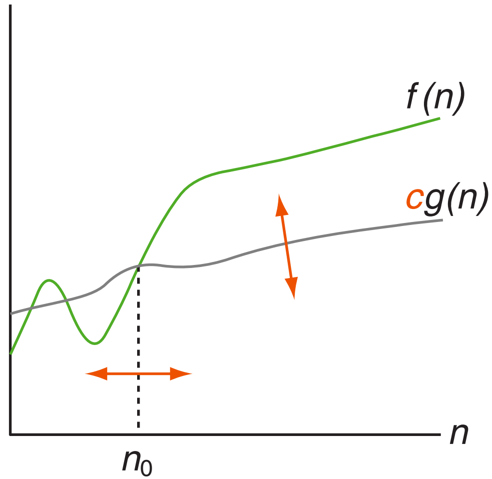
Wenn $f(n) \in o(g(n))$ schreiben wir $f(n) = o(g(n))$.

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



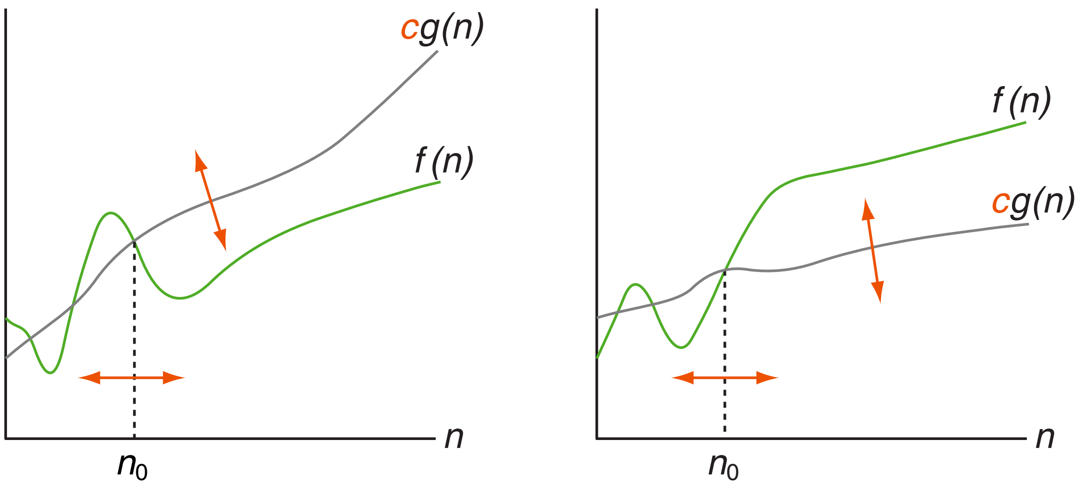


$\omega(g(n)) = \{f(n) \mid$ für alle Konstanten $c > 0$ existiert eine Konstante $n_0 > 0$, so dass für alle $n \geq n_0$ gilt $0 \leq c \cdot g(n) < f(n)\}$

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



$\omega(g(n)) = \{f(n) \mid$ für alle Konstanten $c > 0$ existiert eine Konstante $n_0 > 0$, so dass für alle $n \geq n_0$ gilt $0 \leq c \cdot g(n) < f(n)\}$

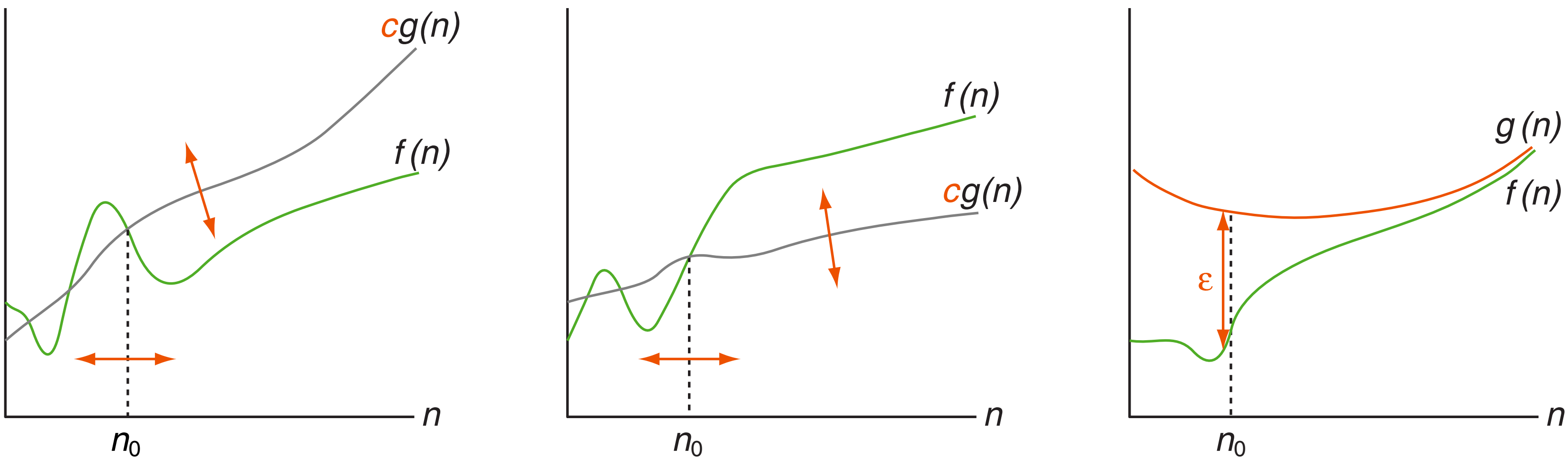
Für alle $f(n) \in \omega(g(n))$ gilt: $g(n)$ ist asymptotisch strikt kleiner als $f(n)$.

Wenn $f(n) \in \omega(g(n))$ schreiben wir $f(n) = \omega(g(n))$.

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \rightarrow \mathbf{R}_0^+$ Funktionen.



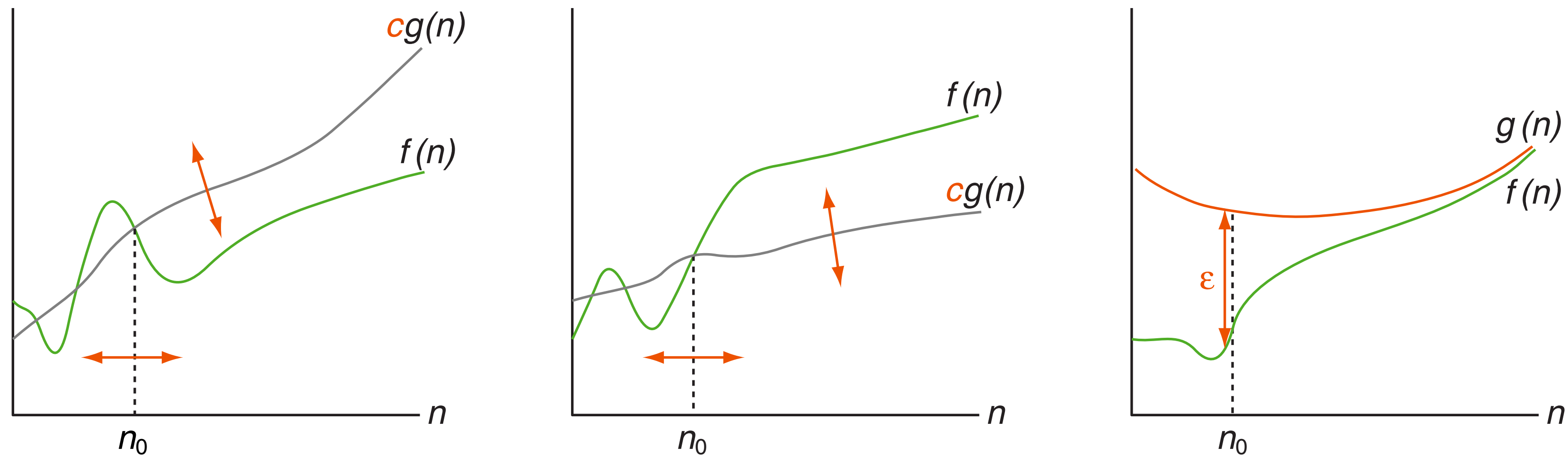
$\vartheta(g(n)) = \{f(n) \mid$ für alle Konstanten $\varepsilon > 0$ existiert eine Konstante $n_0 > 0$, so dass

für alle $n \geq n_0$ gilt $|f(n)/g(n) - 1| \leq \varepsilon\}$

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



$\vartheta(g(n)) = \{f(n) \mid$ für alle Konstanten $\varepsilon > 0$ existiert eine Konstante $n_0 > 0$, so dass für alle $n \geq n_0$ gilt $|f(n)/g(n) - 1| \leq \varepsilon\}$

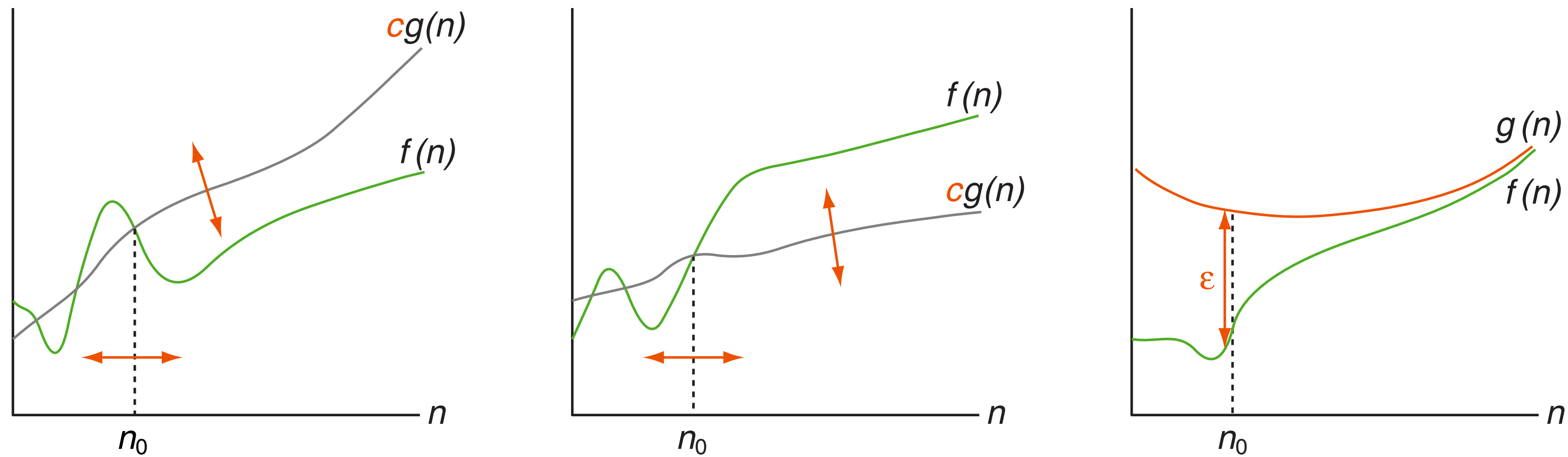
Für alle $f(n) \in \vartheta(g(n))$ gilt: $g(n)$ ist asymptotisch gleich zu $f(n)$.

Wenn $f(n) \in \vartheta(g(n))$ schreiben wir $f(n) \sim g(n)$.

Asymptotische Analyse

Bachmann-Landau-Symbole

Seien $f, g : \mathbf{N} \to \mathbf{R}_0^+$ Funktionen.



Beispiele:

- $n^{1.9999} = o(n^2)$; $n^2/\lg n = o(n^2)$; $n^2 \neq o(n^2)$; $n^2/1000 \neq o(n^2)$
- $n^{2.0001} = \omega(n^2)$; $n^2 \lg n = \omega(n^2)$; $n^2 \neq \omega(n^2)$; $1000n^2 \neq \omega(n^2)$
- $n! \sim \sqrt{2\pi n}(\frac{n}{e})^n$ (Stirling-Formel) [plot]

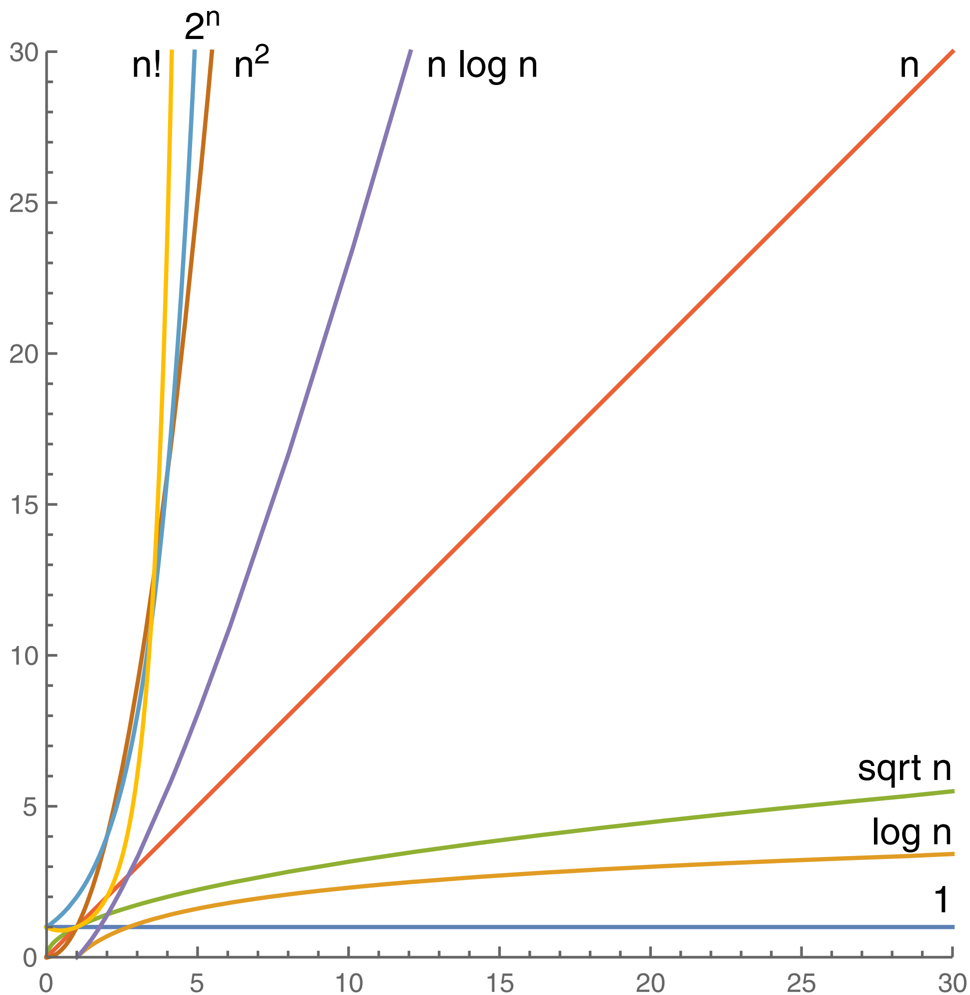
Asymptotische Analyse

Bachmann-Landau-Symbole: Typische Wachstumsraten

Klassifizierung	Wachstum von f	Verhalten
$f(n) = \Theta(n^0) = \Theta(1)$	beschränkt	$n \cdot m \to f(n) \cdot 1$
$f(n) = \Theta(\lg n)$	logarithmisch	$n \cdot 2 \to f(n) \cdot c$
$f(n) = \Theta(\sqrt{n})$	wie Wurzel-n	$n \cdot 4 \to f(n) \cdot 2$
$f(n) = \Theta(n)$	linear	$n \cdot 2 \to f(n) \cdot 2$
$f(n) = \Theta(n \lg n)$	n-log-n	$n \cdot 2 \to f(n) \cdot 2c$
$f(n) = \Theta(n^2)$	quadratisch	$n \cdot 2 \to f(n) \cdot 4$
$f(n) = \Theta(n^c)$	polynomiell	$n \cdot 2 \to f(n) \cdot 2^c$
$f(n) = \Theta(c^n)$	exponentiell	$n + 1 \to f(n) \cdot c$
$f(n) = \Theta(n!)$	faktoriell	$n + 1 \to f(n) \cdot (n + 1)$

Asymptotische Analyse

Bachmann-Landau-Symbole: Typische Wachstumsraten



Asymptotische Analyse

Bachmann-Landau-Symbole: Notation

Funktionsmengen

- $c \cdot n^2 + \Theta(n)$ steht für die Menge aller Funktionen $f(n)$ aus $\Theta(n)$, die mit $c \cdot n^2$ addiert werden können.

Gleichungen

- Der $=$-Operator wird weitläufig stellvertretend für $\subseteq$ verwendet.

  Beispiel: $c \cdot n^2 + \Theta(n) = \Theta(n^2)$ steht für $c \cdot n^2 + \Theta(n) \subseteq \Theta(n^2)$

- Diese Interpretation des Gleichheitszeichens gilt, wenn an mindestens einer Stelle einer Gleichung Bachmann-Landau-Symbole verwendet werden.

Asymptotische Analyse

Bachmann-Landau-Symbole: Eigenschaften

Transitivität

- Aus $f(n) = \Theta(g(n))$ und $g(n) = \Theta(h(n))$ folgt $f(n) = \Theta(h(n))$.
- Analog für O, Ω, o, und ω.

Reflexivität

- $f(n) = \Theta(f(n))$
- Analog für O und Ω.

Symmetrie

- $f(n) = \Theta(g(n))$ genau dann, wenn $g(n) = \Theta(f(n))$.

Austausch-Symmetrie

- $f(n) = O(g(n))$ genau dann, wenn $g(n) = \Omega(f(n))$.
- $f(n) = o(g(n))$ genau dann, wenn $g(n) = \omega(f(n))$.

Asymptotische Analyse

Bachmann-Landau-Symbole: Rechenregeln

Koeffizienten

- Für Konstante $c > 0$ gilt $\Theta(c \cdot g(n)) = \Theta(g(n))$
- Für Konstante $c > 0$ gilt, wenn $f(n) = \Theta(n)$, dann $c \cdot f(n) = \Theta(n)$
- Analog für O, Ω, o, und ω.

Produkt

- Aus $f_1(n) = \Theta(g_1(n))$ und $f_2(n) = \Theta(g_2(n))$ folgt $f_1(n)f_2(n) = \Theta(g_1(n)g_2(n))$.
- Analog für O, Ω, o, und ω.

Addition

- Aus $f_1(n) = \Theta(g_1(n))$ und $f_2(n) = \Theta(g_2(n))$ folgt $f_1(n) + f_2(n) = \Theta(g_1(n) + g_2(n))$.
- Analog für O, Ω, o, und ω.
- Aus $f_1(n) = O(g_1(n))$ und $f_2(n) = O(g_2(n))$ folgt
  $f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\})$.

Asymptotische Analyse

Bachmann-Landau-Symbole: Überblick

$O(g(n)) = \{f(n)$ | es existieren Konstanten $c > 0$ und $n_0 > 0$, so dass
für alle $n \geq n_0$ gilt $0 \leq f(n) \leq c \cdot g(n)\}$

$\Omega(g(n)) = \{f(n)$ | es existieren Konstanten $c > 0$ und $n_0 > 0$, so dass
für alle $n \geq n_0$ gilt $0 \leq c \cdot g(n) \leq f(n)\}$

$\Theta(g(n)) = \{f(n)$ | es existieren Konstanten $c_1 > 0$, $c_2 > 0$ und $n_0 > 0$, so dass
für alle $n \geq n_0$ gilt $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$

$o(g(n)) = \{f(n)$ | für alle Konstanten $c > 0$ existiert eine Konstante $n_0 > 0$, so dass
für alle $n \geq n_0$ gilt $0 \leq f(n) < c \cdot g(n)\}$

$\omega(g(n)) = \{f(n)$ | für alle Konstanten $c > 0$ existiert eine Konstante $n_0 > 0$, so dass
für alle $n \geq n_0$ gilt $0 \leq c \cdot g(n) < f(n)\}$

Bemerkungen:

- Die Menge ϑ (in Zeichen $\sim$) wird nicht zu den Bachmann-Landau-Symbolen gezählt; hier aber zusätzlich dargestellt als passendes Komplement zu Θ. Das Auffinden von Funktionen für andere, aufwändig zu berechnende Funktionen ist ein wichtiges Anwendungsgebiet der asymptotischen Analyse in der Mathematik.

- Asymptotische Analysen werden für Algorithmen sowohl zur Abschätzung ihrer Laufzeit als auch zur Abschätzung ihres Platzverbrauchs in Abhängigkeit der Problemgröße n angewendet.

- In vielen Situation im Algorithmendesign ist es möglich, verringerte Laufzeit gegen erhöhten Platzverbrauch einzutauschen und umgekehrt.

- Äquivalente Definitionen mit Hilfe des Grenzwerts, falls $g(n) > 0$ für hinreichend große n:
  - $f(n) = O(g(n))$ genau dann, wenn $\limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \infty$
  - $f(n) = \Omega(g(n))$ genau dann, wenn $\liminf_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| > 0$
  - $f(n) = \Theta(g(n))$ genau dann, wenn $f(n) = O(g(n))$ und $f(n) = \Omega(g(n))$
  - $f(n) = o(g(n))$ genau dann, wenn $\lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| = 0$
  - $f(n) = \omega(g(n))$ genau dann, wenn $\lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| = \infty$

Bemerkungen: (Fortsetzung)

- In vielen praktischen Fällen ist die Zugehörigkeit einer Funktion $f(n)$ zu einer Klasse für ein $g(n)$ "offensichtlich": Wenn f und g monoton wachsen, genügt es, ein Beispiel für c bzw. c_1, c_2 sowie n_0 anzugeben, ab denen die jeweils vorausgesetzte Ungleichung erfüllt ist.
- Um einen Nachweis der Zugehörigkeit zu erbringen, gibt es eine Reihe möglicher Beweistechniken.
- Beweis der Zugehörigkeit einer Funktion $f(n)$ zu beispielsweise $O(g(n))$ (analog für Ω):
  1. Anwendung der Definition: Finde ein $c > 0$ und ein $n_0 > 0$, so dass $0 \leq f(n) \leq c \cdot g(n)$.
  2. Zeige mittels vollständiger Induktion, dass die Ungleichung für alle $n > n_0$ erfüllt ist.
- Beweis der (Nicht-)Zugehörigkeit zu O, Ω, o, ω: Anwendung der Grenzwertdefinitionen.
- Beweis der Nicht-Zugehörigkeit zu o, ω: Finde ein Gegenbeispiel für $c > 0$ und $n_0 > 0$.

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = O(n)$?

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = O(n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$0 \leq 7n + 8 \leq c \cdot n$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = O(n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$0 \ \leq \ 7n + 8 \ \leq \ c \cdot n$$

Sei $c = 8$ und $n_0 = 8$.

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = O(n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$0 \ \leq \ 7n + 8 \ \leq \ c \cdot n$$

Sei $c = 8$ und $n_0 = 8$.

Beweis für alle $n > n_0$ durch vollständige Induktion:

Induktionsanfang: $7 \cdot 8 + 8 \leq 8 \cdot 8 \ \Leftrightarrow \ 64 \leq 64$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = O(n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$0 \;\leq\; 7n + 8 \;\leq\; c \cdot n$$

Sei $c = 8$ und $n_0 = 8$.

Beweis für alle $n > n_0$ durch vollständige Induktion:

Induktionsanfang: $7 \cdot 8 + 8 \leq 8 \cdot 8 \;\Leftrightarrow\; 64 \leq 64$

Induktionsschritt: Wenn $7 \cdot k + 8 \leq 8k$ für $k \geq 8$, dann gilt

$$7 \cdot (k + 1) + 8 \;\leq\; 8 \cdot (k + 1)$$

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = O(n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$0 \;\leq\; 7n + 8 \;\leq\; c \cdot n$$

Sei $c = 8$ und $n_0 = 8$.

Beweis für alle $n > n_0$ durch vollständige Induktion:

Induktionsanfang: $7 \cdot 8 + 8 \leq 8 \cdot 8 \;\Leftrightarrow\; 64 \leq 64$

Induktionsschritt: Wenn $7 \cdot k + 8 \leq 8k$ für $k \geq 8$, dann gilt

$$\begin{aligned} 7 \cdot (k+1) + 8 \;&\leq\; 8 \cdot (k+1) \\ \Leftrightarrow \qquad 7k + 7 + 8 \;&\leq\; 8k + 8 \end{aligned}$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = O(n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$0 \ \leq \ 7n + 8 \ \leq \ c \cdot n$$

Sei $c = 8$ und $n_0 = 8$.

Beweis für alle $n > n_0$ durch vollständige Induktion:

Induktionsanfang: $7 \cdot 8 + 8 \leq 8 \cdot 8 \ \Leftrightarrow \ 64 \leq 64$

Induktionsschritt: Wenn $7 \cdot k + 8 \leq 8k$ für $k \geq 8$, dann gilt

$$\begin{aligned} & 7 \cdot (k+1) + 8 \ \leq \ 8 \cdot (k+1) \\ \Leftrightarrow \quad & 7k + 7 + 8 \ \leq \ 8k + 8 \\ \Leftrightarrow \quad & 7k + 8 + 7 \ \leq \ 8k + 8 \end{aligned}$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = O(n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$0 \;\leq\; 7n + 8 \;\leq\; c \cdot n$$

Sei $c = 8$ und $n_0 = 8$.

Beweis für alle $n > n_0$ durch vollständige Induktion:

Induktionsanfang: $7 \cdot 8 + 8 \leq 8 \cdot 8 \;\Leftrightarrow\; 64 \leq 64$

Induktionsschritt: Wenn $7 \cdot k + 8 \leq 8k$ für $k \geq 8$, dann gilt

$$\begin{aligned}
 & 7 \cdot (k+1) + 8 \;\leq\; 8 \cdot (k+1) \\
\Leftrightarrow \quad & 7k + 7 + 8 \;\leq\; 8k + 8 \\
\Leftrightarrow \quad & 7k + 8 + 7 \;\leq\; 8k + 8 \\
\Leftarrow \quad & 8k + 7 \;\leq\; 8k + 8
\end{aligned}$$

(Prämisse einsetzen)

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = O(n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$0 \;\leq\; 7n + 8 \;\leq\; c \cdot n$$

Sei $c = 8$ und $n_0 = 8$.

Beweis für alle $n > n_0$ durch vollständige Induktion:

Induktionsanfang: $7 \cdot 8 + 8 \leq 8 \cdot 8 \;\Leftrightarrow\; 64 \leq 64$

Induktionsschritt: Wenn $7 \cdot k + 8 \leq 8k$ für $k \geq 8$, dann gilt

$$\begin{aligned}
& 7 \cdot (k+1) + 8 \;\leq\; 8 \cdot (k+1) \\
\Leftrightarrow \quad & 7k + 7 + 8 \;\leq\; 8k + 8 \\
\Leftrightarrow \quad & 7k + 8 + 7 \;\leq\; 8k + 8 \\
\Leftarrow \quad & 8k + 7 \;\leq\; 8k + 8 \qquad \text{(Prämisse einsetzen)} \\
\Leftrightarrow \quad & 7 \;\leq\; 8 \qquad \square
\end{aligned}$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = o(n)$?

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = o(n)$?

Per Definition muss für alle $c > 0$ ein $n_0 > 0$ existieren, so dass für alle $n \geq n_0$ gilt:

$$0 \leq 7n + 8 < c \cdot n$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = o(n)$?

Per Definition muss für alle $c > 0$ ein $n_0 > 0$ existieren, so dass für alle $n \geq n_0$ gilt:

$$0 \leq 7n + 8 < c \cdot n$$

Sei $c = 1/10$:

$$\begin{aligned} & 0 \leq 7n + 8 < \frac{1}{10} \cdot n \\ \Leftrightarrow\ & 0 \geq -\frac{80}{69} > n \qquad \perp \end{aligned}$$

Für $c = 1/10$ gibt es kein $n_0 > 0$, das die Voraussetzung erfüllt.

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = o(n^2)$?

Per Definition muss für alle $c > 0$ ein $n_0 > 0$ existieren, so dass für alle $n \geq n_0$ gilt:

$$0 \leq \quad 7n + 8 \quad < c \cdot n^2$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = o(n^2)$?

Per Definition muss für alle $c > 0$ ein $n_0 > 0$ existieren, so dass für alle $n \geq n_0$ gilt:

$$0 \leq 7n + 8 < c \cdot n^2$$

$$\Leftrightarrow 0 \leq \frac{7n + 8}{n^2} < c$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = o(n^2)$?

Per Definition muss für alle $c > 0$ ein $n_0 > 0$ existieren, so dass für alle $n \geq n_0$ gilt:

$$\begin{aligned} & 0 \leq 7n + 8 < c \cdot n^2 \\ \Leftrightarrow\ & 0 \leq \frac{7n+8}{n^2} < c \\ \Leftrightarrow\ & \lim_{n\to\infty} \frac{7n+8}{n^2} = 0 \end{aligned}$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = o(n^2)$?

Per Definition muss für alle $c > 0$ ein $n_0 > 0$ existieren, so dass für alle $n \geq n_0$ gilt:

$$
\begin{aligned}
& 0 \leq 7n + 8 < c \cdot n^2 \\
\Leftrightarrow\ & 0 \leq \frac{7n+8}{n^2} < c \\
\Leftrightarrow\ & \lim_{n\to\infty} \frac{7n+8}{n^2} = 0 \\
\Leftrightarrow\ & \lim_{n\to\infty} \frac{7}{2n} = 0 \qquad \text{(Satz von L'Hôpital)} \quad \square
\end{aligned}
$$

Asymptotische Analyse
Bachmann-Landau-Symbole: Beispiele

Ist $7n + 8 = o(n^2)$?

Per Definition muss für alle $c > 0$ ein $n_0 > 0$ existieren, so dass für alle $n \geq n_0$ gilt:

$$
\begin{aligned}
& 0 \leq 7n + 8 < c \cdot n^2 \\
\Leftrightarrow\ & 0 \leq \frac{7n+8}{n^2} < c \\
\Leftrightarrow\ & \lim_{n \to \infty} \frac{7n+8}{n^2} = 0 \\
\Leftrightarrow\ & \lim_{n \to \infty} \frac{7}{2n} = 0 \qquad \text{(Satz von L'Hôpital)} \quad \square
\end{aligned}
$$

Es folgt, dass die Voraussetzung gilt.

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $2^{n+1} = O(2^n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$\begin{aligned} & 0 \leq 2^{n+1} \leq c \cdot 2^n \\ \Leftrightarrow\ & 0 \leq 2 \cdot 2^n \leq c \cdot 2^n \\ \Leftrightarrow\ & 0 \leq 2 \leq c \qquad \square \end{aligned}$$

Mit $c = 2$ und $n_0 = 1$ ist die Voraussetzung erfüllt.

Ist $2^{2n} = O(2^n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$\begin{aligned} & 0 \leq 2^{2n} \leq c \cdot 2^n \\ \Leftrightarrow\ & 0 \leq 2^n \cdot 2^n \leq c \cdot 2^n \\ \Rightarrow\ & 0 \leq 2^n \leq c \qquad \perp \end{aligned}$$

Es gibt kein $c > 0$, das für alle $n > n_0 > 0$ größer als 2^n ist.

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $2^{n+1} = O(2^n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$
\begin{aligned}
& 0 \;\le\; 2^{n+1} \;\le\; c \cdot 2^n \\
\Leftrightarrow\; & 0 \;\le\; 2 \cdot 2^n \;\le\; c \cdot 2^n \\
\Leftrightarrow\; & 0 \;\le\; 2 \le\; c \qquad \square
\end{aligned}
$$

Mit $c = 2$ und $n_0 = 1$ ist die Voraussetzung erfüllt.

Ist $2^{2n} = O(2^n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$
\begin{aligned}
& 0 \;\le\; 2^{2n} \;\le\; c \cdot 2^n \\
\Leftrightarrow\; & 0 \;\le\; 2^n \cdot 2^n \;\le\; c \cdot 2^n \\
\Rightarrow\; & 0 \;\le\; 2^n \le\; c \qquad \bot
\end{aligned}
$$

Es gibt kein $c > 0$, das für alle $n > n_0 > 0$ größer als 2^n ist.

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Ist $2^{n+1} = O(2^n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$\begin{aligned} & 0 \leq 2^{n+1} \leq c \cdot 2^n \\ \Leftrightarrow\ & 0 \leq 2 \cdot 2^n \leq c \cdot 2^n \\ \Leftrightarrow\ & 0 \leq 2 \leq c \qquad \square \end{aligned}$$

Mit $c = 2$ und $n_0 = 1$ ist die Voraussetzung erfüllt.

Ist $2^{2n} = O(2^n)$?

Per Definition muss für ein $c, n_0 > 0$ für alle $n > n_0$ gelten:

$$\begin{aligned} & 0 \leq 2^{2n} \leq c \cdot 2^n \\ \Leftrightarrow\ & 0 \leq 2^n \cdot 2^n \leq c \cdot 2^n \\ \Rightarrow\ & 0 \leq 2^n \leq c \qquad \perp \end{aligned}$$

Es gibt kein $c > 0$, das für alle $n > n_0 > 0$ größer als 2^n ist.

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Zeige, dass $(n+a)^b = \Theta(n^b)$ für $a \in \mathbf{R}$ und $b > 0$.

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Zeige, dass $(n+a)^b = \Theta(n^b)$ für $a \in \mathbf{R}$ und $b > 0$.

Per Definition muss für ein $c_1, c_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \;\leq\; c_1 \cdot n^b \;\leq\; (n+a)^b \;\leq\; c_2 \cdot n^b$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Zeige, dass $(n+a)^b = \Theta(n^b)$ für $a \in \mathbf{R}$ und $b > 0$.

Per Definition muss für ein $c_1, c_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \;\leq\; c_1 \cdot n^b \;\leq\; (n+a)^b \;\leq\; c_2 \cdot n^b$$

Beobachtungen:

$$\begin{aligned} n+a \;&\geq\; n - |a| \\ &\geq\; \tfrac{1}{2}n \qquad \text{sobald } \tfrac{1}{2}n \geq |a| \end{aligned}$$

$$\begin{aligned} n+a \;&\leq\; n + |a| \\ &\leq\; 2n \qquad \text{sobald } n \geq |a| \end{aligned}$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Zeige, dass $(n+a)^b = \Theta(n^b)$ für $a \in \mathbf{R}$ und $b > 0$.

Per Definition muss für ein $c_1, c_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \;\leq\; c_1 \cdot n^b \;\leq\; (n+a)^b \;\leq\; c_2 \cdot n^b$$

Beobachtungen:

$$\begin{aligned} n+a &\;\geq\; n - |a| \\ &\;\geq\; \tfrac{1}{2}n \qquad \text{sobald } \tfrac{1}{2}n \geq |a| \end{aligned}$$

$$\begin{aligned} n+a &\;\leq\; n + |a| \\ &\;\leq\; 2n \qquad \text{sobald } n \geq |a| \end{aligned}$$

Daraus folgt, dass sobald $n > n_0 \geq 2|a|$ ist:

$$0 \;\leq\; \frac{1}{2}n \;\leq\; n+a \;\leq\; 2n$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Zeige, dass $(n+a)^b = \Theta(n^b)$ für $a \in \mathbf{R}$ und $b > 0$.

Per Definition muss für ein $c_1, c_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \ \leq \ c_1 \cdot n^b \ \leq \ (n+a)^b \ \leq \ c_2 \cdot n^b$$

Beobachtungen:

$$\begin{aligned} n+a \ &\geq \ n - |a| \\ &\geq \ \tfrac{1}{2}n \qquad \text{sobald } \tfrac{1}{2}n \geq |a| \end{aligned}$$

$$\begin{aligned} n+a \ &\leq \ n + |a| \\ &\leq \ 2n \qquad \text{sobald } n \geq |a| \end{aligned}$$

Daraus folgt, dass sobald $n > n_0 \geq 2|a|$ ist:

$$0 \ \leq \ \frac{1}{2}n \ \leq \ n+a \ \leq \ 2n$$

Da alle Terme der Ungleichung sowie b positiv für $n > n_0$ sind:

$$\begin{aligned} & 0 \ \leq \ (\tfrac{1}{2}n)^b \ \leq \ (n+a)^b \ \leq \ (2n)^b \\ \Leftrightarrow \quad & 0 \ \leq \ \left(\tfrac{1}{2}\right)^b n^b \ \leq \ (n+a)^b \ \leq \ 2^b n^b \quad \square \end{aligned}$$

Asymptotische Analyse

Bachmann-Landau-Symbole: Beispiele

Zeige, dass $(n+a)^b = \Theta(n^b)$ für $a \in \mathbf{R}$ und $b > 0$.

Per Definition muss für ein $c_1, c_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \; \leq \; c_1 \cdot n^b \; \leq \; (n+a)^b \; \leq \; c_2 \cdot n^b$$

Beobachtungen:

$$\begin{aligned} n+a \; &\geq \; n - |a| \\ &\geq \; \tfrac{1}{2}n \qquad \text{sobald } \tfrac{1}{2}n \geq |a| \end{aligned}$$

$$\begin{aligned} n+a \; &\leq \; n + |a| \\ &\leq \; 2n \qquad \text{sobald } n \geq |a| \end{aligned}$$

Daraus folgt, dass sobald $n > n_0 \geq 2|a|$ ist:

$$0 \; \leq \; \frac{1}{2}n \; \leq \; n+a \; \leq \; 2n$$

Da alle Terme der Ungleichung sowie b positiv für $n > n_0$ sind:

$$\begin{aligned} & 0 \; \leq \; (\tfrac{1}{2}n)^b \; \leq \; (n+a)^b \; \leq \; (2n)^b \\ \Leftrightarrow \quad & 0 \; \leq \; \left(\tfrac{1}{2}\right)^b n^b \; \leq \; (n+a)^b \; \leq \; 2^b n^b \quad \square \end{aligned}$$

Mit $c_1 = (1/2)^b$, $c_2 = 2^b$ und $n_0 = 2|a|$ ist die Voraussetzung erfüllt.

Asymptotische Analyse

Wachstumsrate rekursiver Funktionen

Sei $T(n)$ die Laufzeitfunktion eines rekursiven Algorithmus für Problemgröße n:

$$T(n) = \begin{cases} \Theta(1) & \text{für } n \leq c, \\ a \cdot T(n/b) + f(n) & \text{sonst.} \end{cases}$$

Asymptotische Analyse

Wachstumsrate rekursiver Funktionen

Sei $T(n)$ die Laufzeitfunktion eines rekursiven Algorithmus für Problemgröße n:

$$T(n) = \begin{cases} \Theta(1) & \text{für } n \leq c, \\ a \cdot T(n/b) + f(n) & \text{sonst.} \end{cases}$$

Annahmen:

- $\Theta(1)$ in $T(n)$ ist die Laufzeit, die zur Lösung eines Problems der Größe $n \leq c$ für eine Konstante c benötigt wird.
- Sonst wird das Problem in a gleichartige Teilprobleme der Größe n/b geteilt.
- $T(n/b)$ ist die Laufzeit, die zur Lösung eines Teilproblems benötigt wird.
- $f(n) = D(n) + C(n) + \Theta(1)$ für Divide and Conquer-Algorithmen.
- $D(n)$ ist die Laufzeit, die zur Teilung des Problems benötigt wird.
- $C(n)$ ist die Laufzeit, die zur Kombination der Teillösung benötigt wird.
- $\Theta(1)$ in $f(n)$ ist die Restlaufzeit für sonstige Anweisungen.

Asymptotische Analyse
Wachstumsrate rekursiver Funktionen

Sei $T(n)$ die Laufzeitfunktion eines rekursiven Algorithmus für Problemgröße n:

$$T(n) = \begin{cases} \Theta(1) & \text{für } n \leq c, \\ a \cdot T(n/b) + f(n) & \text{sonst.} \end{cases}$$

Welche asymptotische Laufzeit kann für $T(n)$ ermittelt werden?

Um rekursive Funktionen zu lösen, werden folgende Methoden häufig verwendet:

- Iterative Methode
  Entfalten der Rekursion durch wiederholtes Einsetzen bis ein „Formelmuster“ erkennbar wird.
- Rekursionsbaummethode
  Entfalten der Rekursion als Baum von Kosten. Schrittweise Auswertung der Kostenstruktur im Baum zur „Schätzung“ einer Lösung.
- Substitutionsmethode
  Schätzung einer Lösung. Beweis der Korrektheit durch vollständige Induktion.
- Master-Theorem
  Lösungsvorschrift für bestimmte Arten rekursiver Funktionen.

Asymptotische Analyse

Rekursionsbaummethode: Beispiel

MergeSort(A, p, r)

IF $p < r$ **THEN**
    $q = \lfloor (p + r)/2 \rfloor$
    MergeSort(A, p, q)
    MergeSort$(A, q + 1, r)$
    Merge(A, p, q, r)
ENDIF

Asymptotische Analyse

Rekursionsbaummethode: Beispiel

MergeSort(A, p, r)

IF $p < r$ **THEN**
    $q = \lfloor (p + r)/2 \rfloor$
    MergeSort(A, p, q)
    MergeSort$(A, q + 1, r)$
    Merge(A, p, q, r)
ENDIF

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \; & c_1 \\
+ \; & D(n) \\
+ \; & T(n/2) \\
+ \; & T(n/2) \\
+ \; & C(n) \\
+ \; & c_6
\end{aligned}
$$

Asymptotische Analyse

Rekursionsbaummethode: Beispiel

MergeSort(A, p, r)

```
IF p < r THEN
   q = ⌊(p + r)/2⌋
   MergeSort(A, p, q)
   MergeSort(A, q + 1, r)
   Merge(A, p, q, r)
ENDIF
```

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \; & \Theta(1) \\
+ \; & \Theta(1) \\
+ \; & T(n/2) \\
+ \; & T(n/2) \\
+ \; & C(n) \\
+ \; & \Theta(1)
\end{aligned}
$$

Asymptotische Analyse

Rekursionsbaummethode: Beispiel

MergeSort(A, p, r)

1. **IF** $p < r$ **THEN**
2. $\quad q = \lfloor (p+r)/2 \rfloor$
3. $\quad$ *MergeSort*(A, p, q)
4. $\quad$ *MergeSort*$(A, q+1, r)$
5. $\quad$ *Merge*(A, p, q, r)
6. **ENDIF**

Laufzeitfunktion:

$$
\begin{aligned}
T(n) = \; & \Theta(1) \\
+ \; & \Theta(1) \\
+ \; & T(n/2) \\
+ \; & T(n/2) \\
+ \; & \Theta(n) \\
+ \; & \Theta(1)
\end{aligned}
$$

Asymptotische Analyse

Rekursionsbaummethode: Beispiel

MergeSort(A, p, r)

```
IF p < r THEN
   q = ⌊(p + r)/2⌋
   MergeSort(A, p, q)
   MergeSort(A, q + 1, r)
   Merge(A, p, q, r)
ENDIF
```

Laufzeitfunktion:

$$T(n) = \begin{cases} \Theta(1) & \text{für } n = 1, \\ 2 \cdot T(n/2) + \Theta(n) & \text{für } n > 1. \end{cases}$$

Asymptotische Analyse

Rekursionsbaummethode: Beispiel

MergeSort(A, p, r)

```
IF p < r THEN
   q = ⌊(p + r)/2⌋
   MergeSort(A, p, q)
   MergeSort(A, q + 1, r)
   Merge(A, p, q, r)
ENDIF
```

Laufzeitfunktion (konkretisiert):

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Asymptotische Analyse

Rekursionsbaummethode: Beispiel

MergeSort(A, p, r)

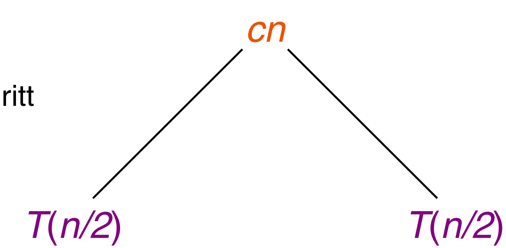
```
IF p < r THEN
   q = ⌊(p + r)/2⌋
   MergeSort(A, p, q)
   MergeSort(A, q + 1, r)
   Merge(A, p, q, r)
ENDIF
```

Laufzeitfunktion (konkretisiert):

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Rekursionsbaum:

1. Rekursionsschritt



Asymptotische Analyse

Rekursionsbaummethode: Beispiel

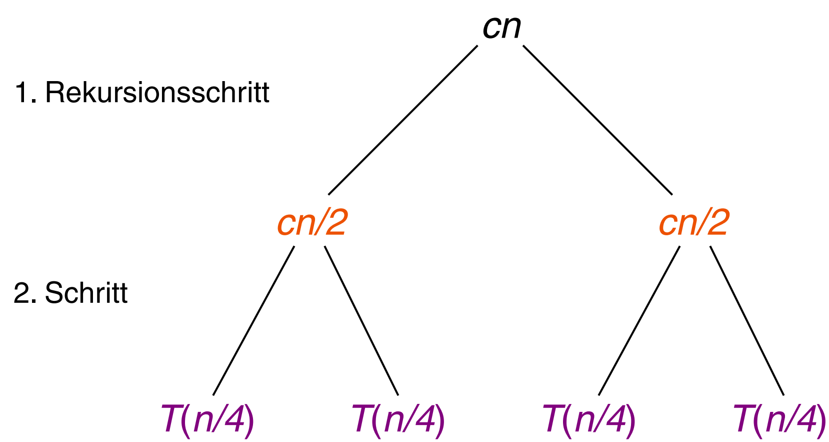
MergeSort(A, p, r)

IF $p < r$ **THEN**
 $q = \lfloor (p + r)/2 \rfloor$
 MergeSort(A, p, q)
 MergeSort$(A, q + 1, r)$
 Merge(A, p, q, r)
ENDIF

Laufzeitfunktion (konkretisiert):

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Rekursionsbaum:



Asymptotische Analyse

Rekursionsbaummethode: Beispiel

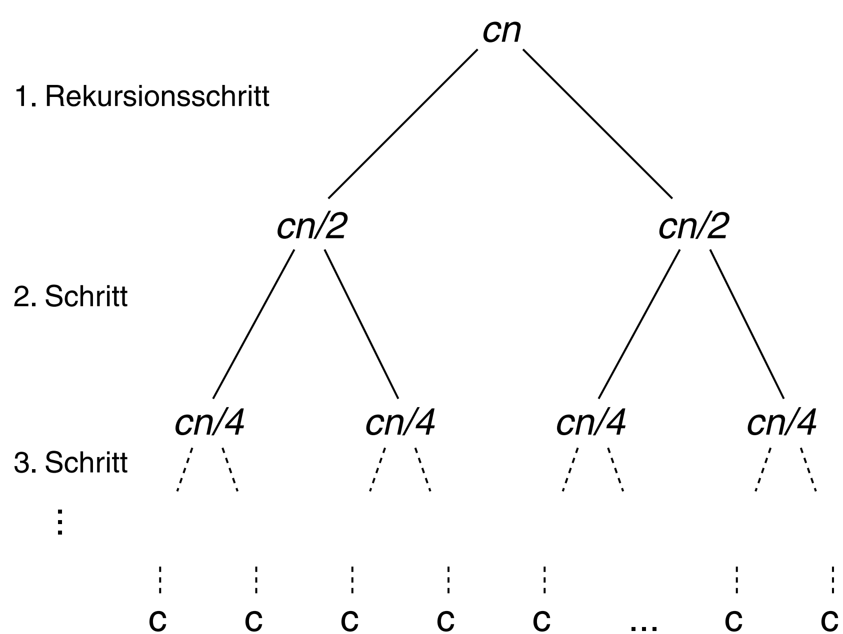
MergeSort(A, p, r)

```
IF p < r THEN
   q = ⌊(p + r)/2⌋
   MergeSort(A, p, q)
   MergeSort(A, q + 1, r)
   Merge(A, p, q, r)
ENDIF
```

Laufzeitfunktion (konkretisiert):

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Rekursionsbaum:



Asymptotische Analyse

Rekursionsbaummethode: Beispiel

MergeSort(A, p, r)

```
IF  p < r  THEN
    q = ⌊(p + r)/2⌋
    MergeSort(A, p, q)
    MergeSort(A, q + 1, r)
    Merge(A, p, q, r)
ENDIF
```

Vereinfachende Annahme:

- n ist 2er-Potenz.

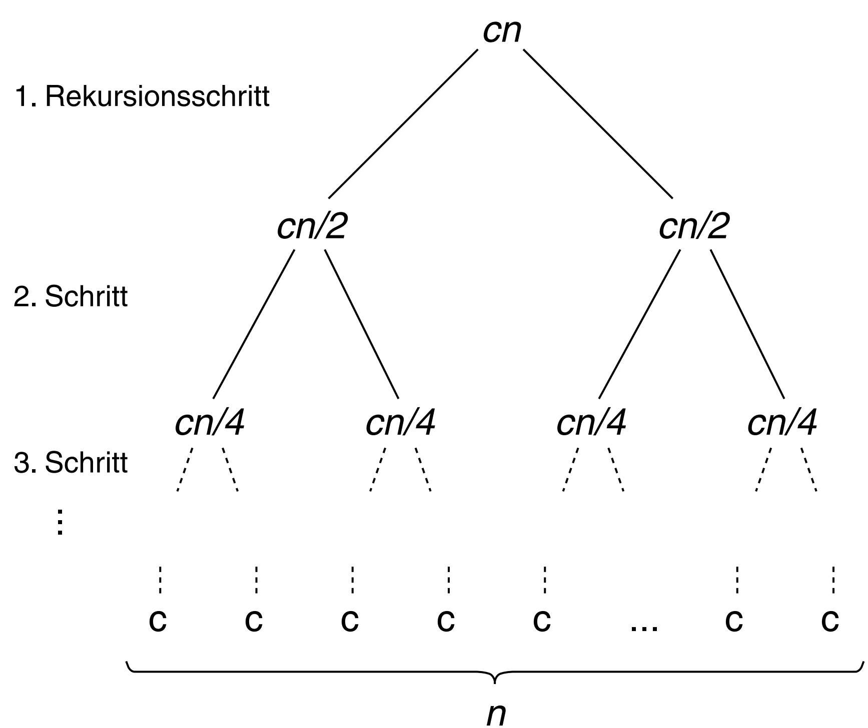
➜ Vernachlässige $\lfloor \cdot \rfloor$ bzw. $\lceil \cdot \rceil$

➜ Unterste Ebene vollständig gefüllt (Worst Case).

Laufzeitfunktion (konkretisiert):

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Rekursionsbaum:



Asymptotische Analyse

Rekursionsbaummethode: Beispiel

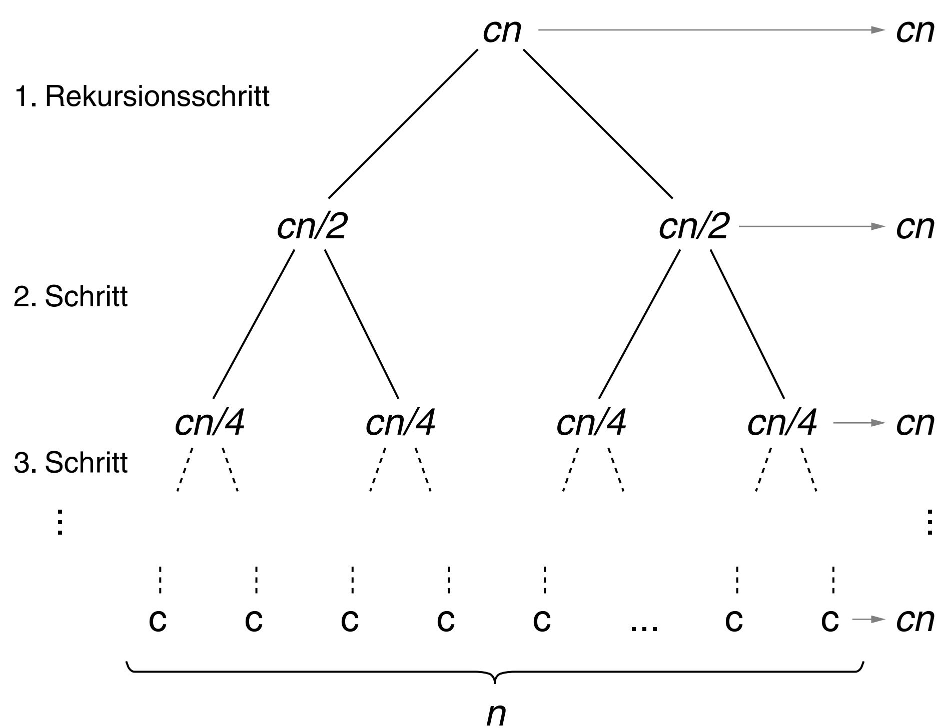
MergeSort(A, p, r)

```
IF p < r THEN
    q = ⌊(p + r)/2⌋
    MergeSort(A, p, q)
    MergeSort(A, q + 1, r)
    Merge(A, p, q, r)
ENDIF
```

Laufzeitfunktion (konkretisiert):

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Rekursionsbaum:



Beobachtungen:

- Kosten pro Ebene: cn

Asymptotische Analyse

Rekursionsbaummethode: Beispiel

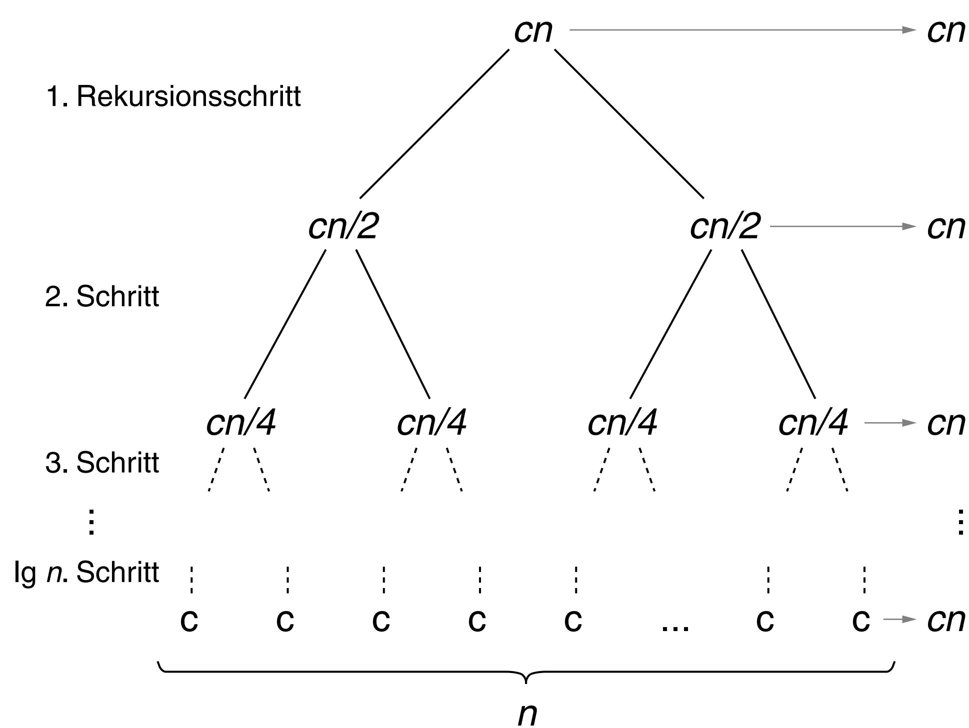
MergeSort(A, p, r)

```
IF p < r THEN
   q = ⌊(p + r)/2⌋
   MergeSort(A, p, q)
   MergeSort(A, q + 1, r)
   Merge(A, p, q, r)
ENDIF
```

Laufzeitfunktion (konkretisiert):

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Rekursionsbaum:



Beobachtungen:

- Kosten pro Ebene: cn
- Anzahl Schritte: $\lg n$
- Anzahl Ebenen: $\lg n + 1$

Asymptotische Analyse

Rekursionsbaummethode: Beispiel

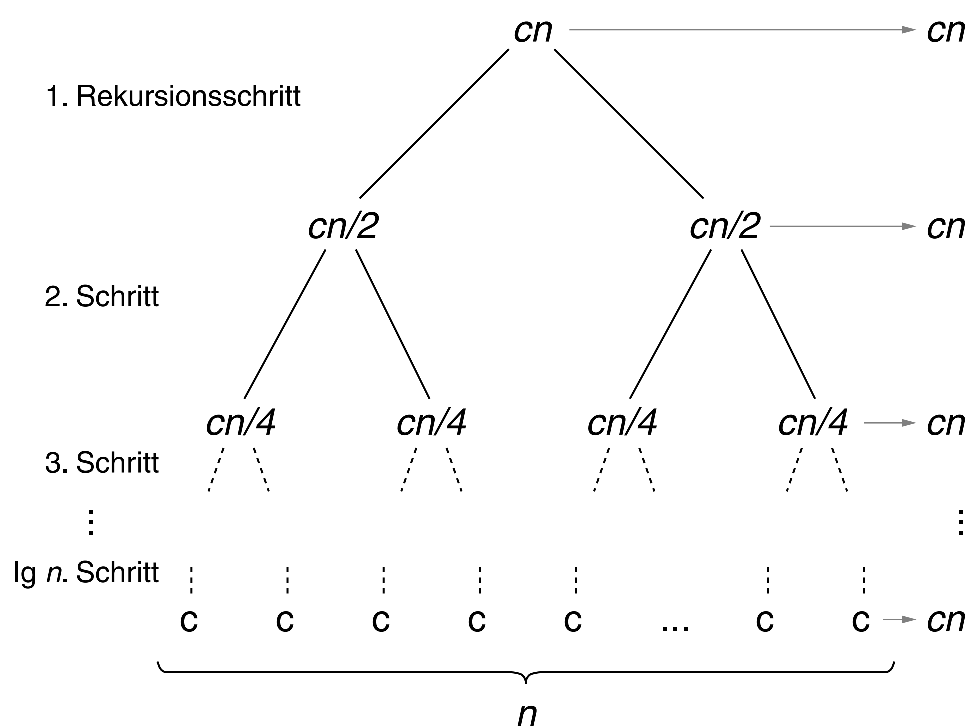
MergeSort(A, p, r)

IF $p < r$ **THEN**
  $q = \lfloor (p + r)/2 \rfloor$
  MergeSort(A, p, q)
  MergeSort$(A, q + 1, r)$
  Merge(A, p, q, r)
ENDIF

Laufzeitfunktion (konkretisiert):

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Rekursionsbaum:



Beobachtungen:

- Kosten pro Ebene: cn
- Anzahl Schritte: $\lg n$
- Anzahl Ebenen: $\lg n + 1$

Hypothese:

- $T(n) = (\lg n + 1) \cdot cn$

Asymptotische Analyse

Rekursionsbaummethode: Beispiel

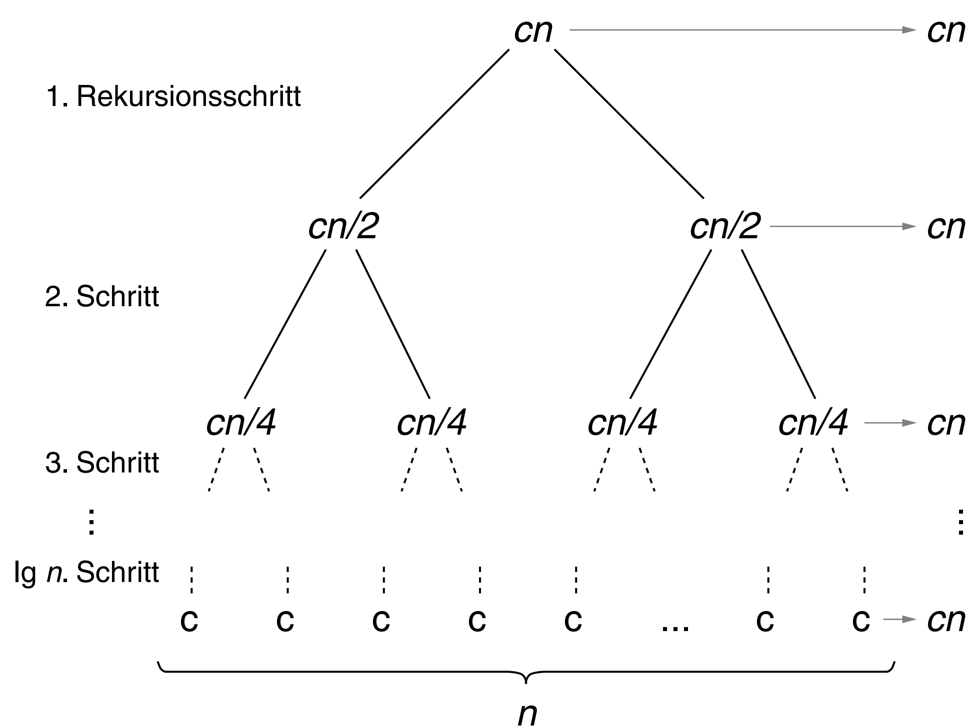
MergeSort(A, p, r)

```
IF p < r THEN
   q = ⌊(p + r)/2⌋
   MergeSort(A, p, q)
   MergeSort(A, q + 1, r)
   Merge(A, p, q, r)
ENDIF
```

Laufzeitfunktion (konkretisiert):

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Rekursionsbaum:



Beobachtungen:

- Kosten pro Ebene: cn
- Anzahl Schritte: $\lg n$
- Anzahl Ebenen: $\lg n + 1$

Hypothese:

- $T(n) = cn \cdot \lg n + cn$

➜ $T(n) = \Theta(n \lg n)$?

Bemerkungen:

- Vereinfachend werden die konstanten Kosten c des Basisfalls mit der Konstante c des zweiten Summanden cn des rekursiven Falls gleichgesetzt.
- Abschätzung der Baumtiefe: Die Größe des Teilproblems auf Ebene i entspricht $n/2^i$. Der Basisfall $n = 1$ ist genau dann erreicht, wenn $n/2^i = 1$, also $i = \log_2 n$.
- Abkürzend schreiben wir $\log_2 n = \lg n$.
- Bei besonderer Sorgfalt kann die Rekursionsbaummethode als Beweis dienen. Aussagekräftiger im Allgemeinen ist jedoch die Substitutionsmethode.

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = cn \cdot \lg n + cn$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = cn \cdot \lg n + cn$$

Beweis durch vollständige Induktion:

Induktionsanfang: Für $n = 1$ ist $c \cdot 1 \cdot \lg 1 + c \cdot 1 = c = T(1)$.

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = cn \cdot \lg n + cn$$

Beweis durch vollständige Induktion:

Induktionsanfang: Für $n = 1$ ist $c \cdot 1 \cdot \lg 1 + c \cdot 1 = c = T(1)$.

Induktionsschritt: Für alle $k < n$ muss $T(k) = ck \cdot \lg k + ck$ gelten, so dass:

$$T(n) = \; 2 \cdot T(\tfrac{n}{2}) + cn \qquad \text{(Voraussetzung)}$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = cn \cdot \lg n + cn$$

Beweis durch vollständige Induktion:

Induktionsanfang: Für $n = 1$ ist $c \cdot 1 \cdot \lg 1 + c \cdot 1 = c = T(1)$.

Induktionsschritt: Für alle $k < n$ muss $T(k) = ck \cdot \lg k + ck$ gelten, so dass:

$$\begin{aligned} T(n) &= 2 \cdot T(\tfrac{n}{2}) + cn && \text{(Voraussetzung)} \\ &= 2 \cdot \left(\tfrac{cn}{2} \lg \tfrac{n}{2} + \tfrac{cn}{2}\right) + cn && \text{(Substitution durch Hypothese)} \end{aligned}$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = cn \cdot \lg n + cn$$

Beweis durch vollständige Induktion:

Induktionsanfang: Für $n = 1$ ist $c \cdot 1 \cdot \lg 1 + c \cdot 1 = c = T(1)$.

Induktionsschritt: Für alle $k < n$ muss $T(k) = ck \cdot \lg k + ck$ gelten, so dass:

$$\begin{aligned}
T(n) &= 2 \cdot T(\tfrac{n}{2}) + cn && \text{(Voraussetzung)} \\
&= 2 \cdot \left(\tfrac{cn}{2} \lg \tfrac{n}{2} + \tfrac{cn}{2}\right) + cn && \text{(Substitution durch Hypothese)} \\
&= cn \cdot \lg \tfrac{n}{2} + cn + cn \\
&= cn \cdot (\lg n - \lg 2) + cn + cn \\
&= cn \cdot \lg n - cn + cn + cn \\
&= cn \cdot \lg n + cn && \square
\end{aligned}$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = \Theta(n \lg n)$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = \Theta(n \lg n)$$

Per Definition muss für ein $d_1, d_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \;\leq\; d_1 \cdot n \lg n \;\leq\; T(n) \;\leq\; d_2 \cdot n \lg n$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = \Theta(n \lg n)$$

Per Definition muss für ein $d_1, d_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \;\leq\; d_1 \cdot n \lg n \;\leq\; T(n) \;\leq\; d_2 \cdot n \lg n$$

Untere Schranke (Ω):

$$T(n) \geq \; 2 \cdot T(\tfrac{n}{2}) + cn$$

Obere Schranke (O):

$$T(n) \leq \; 2 \cdot T(\tfrac{n}{2}) + cn$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = \Theta(n \lg n)$$

Per Definition muss für ein $d_1, d_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \;\leq\; d_1 \cdot n \lg n \;\leq\; T(n) \;\leq\; d_2 \cdot n \lg n$$

Untere Schranke (Ω):

$$\begin{aligned} T(n) &\geq\; 2 \cdot T(\tfrac{n}{2}) + cn \\ &=\; 2 \cdot \left(d_1 \tfrac{n}{2} \lg \tfrac{n}{2}\right) + cn \end{aligned}$$

Obere Schranke (O):

$$\begin{aligned} T(n) &\leq\; 2 \cdot T(\tfrac{n}{2}) + cn \\ &=\; 2 \cdot \left(d_2 \tfrac{n}{2} \lg \tfrac{n}{2}\right) + cn \end{aligned}$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = \Theta(n \lg n)$$

Per Definition muss für ein $d_1, d_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \;\leq\; d_1 \cdot n \lg n \;\leq\; T(n) \;\leq\; d_2 \cdot n \lg n$$

Untere Schranke (Ω):

$$\begin{aligned} T(n) \geq\; & 2 \cdot T(\tfrac{n}{2}) + cn \\ =\; & 2 \cdot \left(d_1 \tfrac{n}{2} \lg \tfrac{n}{2}\right) + cn \\ =\; & d_1 n \cdot \lg \tfrac{n}{2} + cn \\ =\; & d_1 n \cdot (\lg n - \lg 2) + cn \\ =\; & d_1 n \cdot \lg n - d_1 n + cn \end{aligned}$$

Obere Schranke (O):

$$\begin{aligned} T(n) \leq\; & 2 \cdot T(\tfrac{n}{2}) + cn \\ =\; & 2 \cdot \left(d_2 \tfrac{n}{2} \lg \tfrac{n}{2}\right) + cn \\ =\; & d_2 n \cdot \lg \tfrac{n}{2} + cn \\ =\; & d_2 n \cdot (\lg n - \lg 2) + cn \\ =\; & d_2 n \cdot \lg n - d_2 n + cn \end{aligned}$$

Asymptotische Analyse
Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = \Theta(n \lg n)$$

Per Definition muss für ein $d_1, d_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \;\leq\; d_1 \cdot n \lg n \;\leq\; T(n) \;\leq\; d_2 \cdot n \lg n$$

Untere Schranke (Ω):

$$\begin{aligned} T(n) &\geq 2 \cdot T(\tfrac{n}{2}) + cn \\ &= 2 \cdot \left(d_1 \tfrac{n}{2} \lg \tfrac{n}{2}\right) + cn \\ &= d_1 n \cdot \lg \tfrac{n}{2} + cn \\ &= d_1 n \cdot (\lg n - \lg 2) + cn \\ &= d_1 n \cdot \lg n - d_1 n + cn \\ &\geq d_1 n \cdot \lg n \quad \text{für } d_1 \leq c \end{aligned}$$

Obere Schranke (O):

$$\begin{aligned} T(n) &\leq 2 \cdot T(\tfrac{n}{2}) + cn \\ &= 2 \cdot \left(d_2 \tfrac{n}{2} \lg \tfrac{n}{2}\right) + cn \\ &= d_2 n \cdot \lg \tfrac{n}{2} + cn \\ &= d_2 n \cdot (\lg n - \lg 2) + cn \\ &= d_2 n \cdot \lg n - d_2 n + cn \\ &\leq d_2 n \cdot \lg n \quad \text{für } d_2 \geq c \end{aligned}$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = \begin{cases} c & \text{für } n = 1, \\ 2 \cdot T(n/2) + cn & \text{für } n > 1. \end{cases}$$

Hypothese:

$$T(n) = \Theta(n \lg n)$$

Per Definition muss für ein $d_1, d_2, n_0 > 0$ für alle $n \geq n_0$ gelten:

$$0 \;\leq\; d_1 \cdot n \lg n \;\leq\; T(n) \;\leq\; d_2 \cdot n \lg n$$

Untere Schranke (Ω):

$$\begin{aligned} T(n) &\geq 2 \cdot T(\tfrac{n}{2}) + cn \\ &= 2 \cdot \left(d_1 \tfrac{n}{2} \lg \tfrac{n}{2}\right) + cn \\ &= d_1 n \cdot \lg \tfrac{n}{2} + cn \\ &= d_1 n \cdot (\lg n - \lg 2) + cn \\ &= d_1 n \cdot \lg n - d_1 n + cn \\ &\geq d_1 n \cdot \lg n \quad \text{für } d_1 \leq c \end{aligned}$$

Obere Schranke (O):

$$\begin{aligned} T(n) &\leq 2 \cdot T(\tfrac{n}{2}) + cn \\ &= 2 \cdot \left(d_2 \tfrac{n}{2} \lg \tfrac{n}{2}\right) + cn \\ &= d_2 n \cdot \lg \tfrac{n}{2} + cn \\ &= d_2 n \cdot (\lg n - \lg 2) + cn \\ &= d_2 n \cdot \lg n - d_2 n + cn \\ &\leq d_2 n \cdot \lg n \quad \text{für } d_2 \geq c \end{aligned}$$

Daraus folgt $T(n) = \Omega(n \lg n)$ sowie $T(n) = O(n \lg n)$ und damit $T(n) = \Theta(n \lg n)$.

Bemerkungen:

- Es ist im Allgemeinen leichter, obere und untere Schranken getrennt voneinander nachzuweisen.

- In der asymptotischen Analyse einer rekursiven Funktion $T(n)$ wird der Basisfall im Allgemeinen ignoriert, da $T(n)$ für konstantes n konstant ist und da es im Allgemeinen immer möglich ist, einen geeigneten Basisfall zu definieren.

- Es muss die exakte Form der Hypothese hergeleitet werden.

  Beispiel 1: $T(n) = 8T(n/2) + \Theta(n^2)$.

  Hypothese 1: $T(n) \leq dn^3 = O(n^3)$.

$$\begin{aligned} T(n) \leq\ & 8 \cdot T(n/2) + cn^2 \\ =\ & 8d(n/2)^3 + cn^2 \\ =\ & 8d(n^3/8) + cn^2 \\ =\ & dn^3 + cn^2 \\ \not\leq\ & dn^3 \qquad \perp \end{aligned}$$

  Hypothese 2: $T(n) \leq dn^3 - d'n^2 = O(n^3)$.

$$\begin{aligned} T(n) \leq\ & 8 \cdot T(n/2) + cn^2 \\ =\ & 8(d(n/2)^3 - d'(n/2)^2) + cn^2 \\ =\ & 8d(n^3/8) - 8d'(n^2/4) + cn^2 \\ =\ & dn^3 - 2d'n^2 + cn^2 \\ =\ & dn^3 - d'n^2 - d'n^2 + cn^2 \\ \leq\ & dn^3 - d'n^2 \quad \text{für } d' \geq c \qquad \square \end{aligned}$$

  Beispiel 2: $T(n) = 4T(n/4) + n$; Hypothese $T(n) \leq cn = O(n)$.

$$\begin{aligned} T(n) \leq\ & 4(c(n/4)) + n \\ \leq\ & cn + n \\ =\ & O(n) \qquad \perp \end{aligned}$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = T(n/3) + T(2n/3) + cn$$

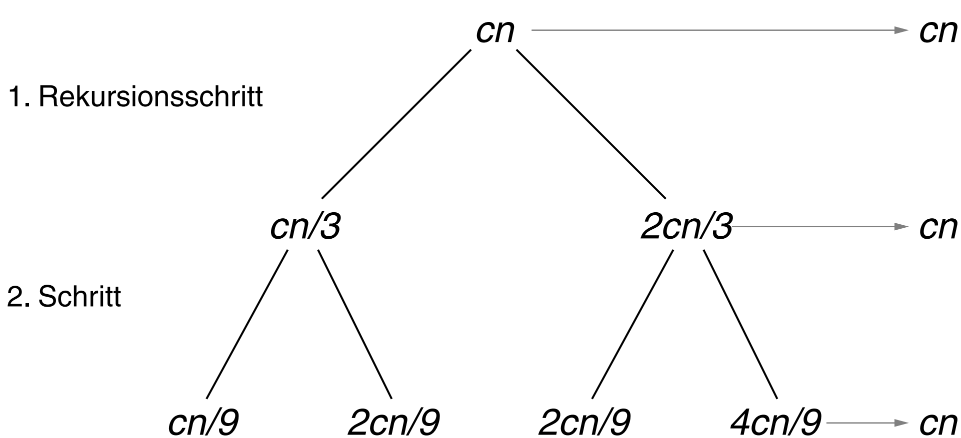
Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = T(n/3) + T(2n/3) + cn$$

Hypothesenbildung nach Rekursionsbaummethode:



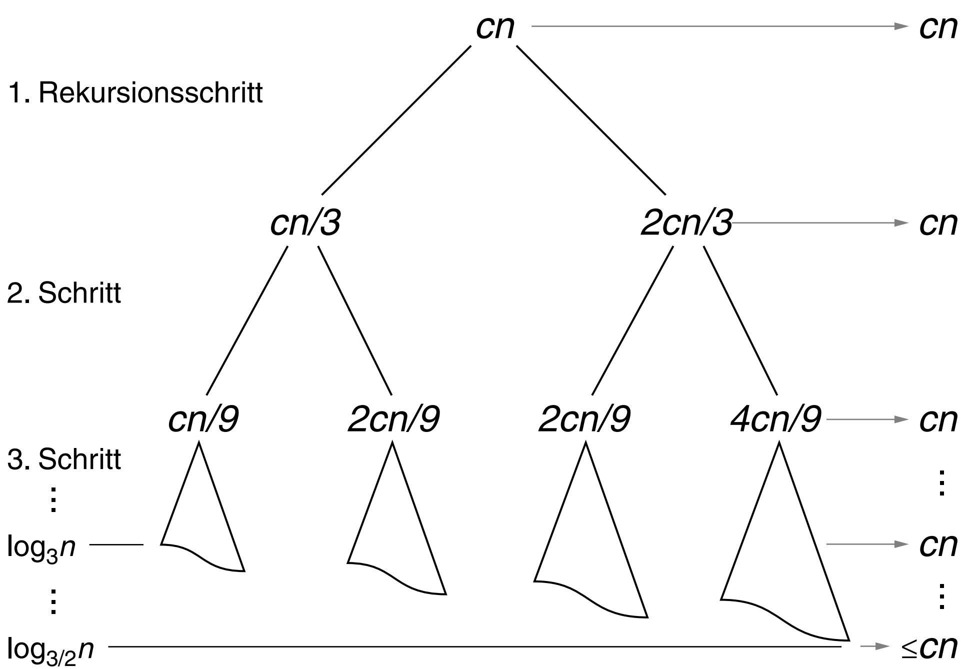
Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = T(n/3) + T(2n/3) + cn$$

Hypothesenbildung nach Rekursionsbaummethode:



Asymptotische Analyse

Substitutionsmethode: Beispiele

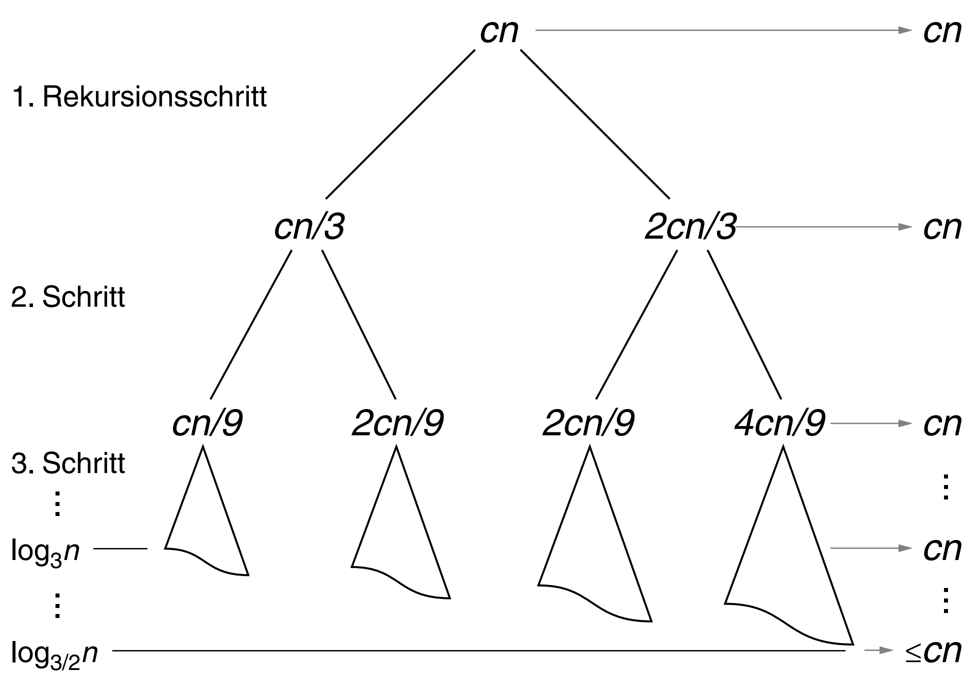
Laufzeitfunktion:

$$T(n) = T(n/3) + T(2n/3) + cn$$

Hypothese:

$$T(n) \geq dn\,\mathsf{log}_3\, n = \Omega(n \lg n)$$
$$T(n) \leq dn\,\mathsf{log}_{3/2}\, n = O(n \lg n)$$

Hypothesenbildung nach Rekursionsbaummethode:



Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = T(n/3) + T(2n/3) + cn$$

Hypothese:

$$T(n) \geq dn\,\mathsf{log}_3\, n = \Omega(n \lg n)$$
$$T(n) \leq dn\,\mathsf{log}_{3/2}\, n = O(n \lg n)$$

Obere Schranke (O):

$$\begin{aligned} T(n) \leq&\ T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + cn \\ \leq&\ d\tfrac{n}{3}\lg\tfrac{n}{3} + d\tfrac{2n}{3}\lg\tfrac{2n}{3} + cn \end{aligned}$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = T(n/3) + T(2n/3) + cn$$

Hypothese:

$$T(n) \geq dn \log_3 n = \Omega(n \lg n)$$
$$T(n) \leq dn \log_{3/2} n = O(n \lg n)$$

Obere Schranke (O):

$$\begin{aligned}
T(n) &\leq T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + cn \\
&\leq d\tfrac{n}{3} \lg \tfrac{n}{3} + d\tfrac{2n}{3} \lg \tfrac{2n}{3} + cn \\
&= d\tfrac{n}{3} \lg n - d\tfrac{n}{3} \lg 3 + d\tfrac{2n}{3} \lg n - d\tfrac{2n}{3} \lg \tfrac{3}{2} + cn \\
&= dn \lg n - d(\tfrac{n}{3} \lg 3 + \tfrac{2n}{3} \lg \tfrac{3}{2}) + cn \\
&= dn \lg n - d(\tfrac{n}{3} \lg 3 + \tfrac{2n}{3} \lg 3 - \tfrac{2n}{3} \lg 2) + cn \\
&= dn \lg n - dn(\lg 3 - \tfrac{2}{3}) + cn
\end{aligned}$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = T(n/3) + T(2n/3) + cn$$

Hypothese:

$$T(n) \geq dn\,\mathsf{log}_3\, n = \Omega(n \lg n)$$
$$T(n) \leq dn\,\mathsf{log}_{3/2}\, n = O(n \lg n)$$

Obere Schranke (O):

$$\begin{aligned}
T(n) &\leq T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + cn \\
&\leq d\tfrac{n}{3} \lg \tfrac{n}{3} + d\tfrac{2n}{3} \lg \tfrac{2n}{3} + cn \\
&= d\tfrac{n}{3} \lg n - d\tfrac{n}{3} \lg 3 + d\tfrac{2n}{3} \lg n - d\tfrac{2n}{3} \lg \tfrac{3}{2} + cn \\
&= dn \lg n - d(\tfrac{n}{3} \lg 3 + \tfrac{2n}{3} \lg \tfrac{3}{2}) + cn \\
&= dn \lg n - d(\tfrac{n}{3} \lg 3 + \tfrac{2n}{3} \lg 3 - \tfrac{2n}{3} \lg 2) + cn \\
&= dn \lg n - dn(\lg 3 - \tfrac{2}{3}) + cn \\
&\leq dn \lg n \quad \text{für } d \geq \tfrac{c}{\lg 3 - \frac{2}{3}} \qquad \square
\end{aligned}$$

Asymptotische Analyse

Substitutionsmethode: Beispiele

Laufzeitfunktion:

$$T(n) = T(n/3) + T(2n/3) + cn$$

Hypothese:

$$T(n) \geq dn\,\mathsf{log}_3\, n = \Omega(n \lg n)$$
$$T(n) \leq dn\,\mathsf{log}_{3/2}\, n = O(n \lg n)$$

Obere Schranke (O):

$$\begin{aligned}
T(n) &\leq T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + cn \\
&\leq d\tfrac{n}{3} \lg \tfrac{n}{3} + d\tfrac{2n}{3} \lg \tfrac{2n}{3} + cn \\
&= d\tfrac{n}{3} \lg n - d\tfrac{n}{3} \lg 3 + d\tfrac{2n}{3} \lg n - d\tfrac{2n}{3} \lg \tfrac{3}{2} + cn \\
&= dn \lg n - d(\tfrac{n}{3} \lg 3 + \tfrac{2n}{3} \lg \tfrac{3}{2}) + cn \\
&= dn \lg n - d(\tfrac{n}{3} \lg 3 + \tfrac{2n}{3} \lg 3 - \tfrac{2n}{3} \lg 2) + cn \\
&= dn \lg n - dn(\lg 3 - \tfrac{2}{3}) + cn \\
&\leq dn \lg n \quad \text{für } d \geq \tfrac{c}{\lg 3 - \frac{2}{3}} \qquad \square
\end{aligned}$$

Untere Schranke (Ω): Analog. Damit ist $T(n) = \Theta(n \lg n)$.

Bemerkungen:

- Baumtiefe: Der Pfad ganz links im Rekursionsbaum erreicht den Basisfall $n = 1$, wenn $(1/3)^i n = 1$ und hat damit die Tiefe $\log_3 n$. Der Pfad ganz rechts im Rekursionsbaum erreicht den Basisfall $n = 1$, wenn $(2/3)^i n = 1$ und hat damit die Tiefe $\log_{3/2} n$.

Asymptotische Analyse

Satz 1 (Master-Theorem)

Seien $a \geq 1$ und $b \geq 1$ Konstanten, $f(n)$ eine Funktion, und $T(n)$ definiert über den natürlichen Zahlen als rekursive Funktion

$$T(n) = a \cdot T(n/b) + f(n),$$

wobei n/b entweder für $\lfloor n/b \rfloor$ oder für $\lceil n/b \rceil$ steht.

Dann hat $T(n)$ folgende asymptotische Schranken:

1. Wenn $f(n) = O(n^{\log_b a - \varepsilon})$ für eine Konstante $\varepsilon > 0$, dann ist $T(n) = \Theta(n^{\log_b a})$.
2. Wenn $f(n) = \Theta(n^{\log_b a})$, dann ist $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. Wenn $f(n) = \Omega(n^{\log_b a + \varepsilon})$ für eine Konstante $\varepsilon > 0$ und wenn $a \cdot f(n/b) \leq c \cdot f(n)$ für eine Konstante $c < 1$ und hinreichend große n, dann ist $T(n) = \Theta(f(n))$

Asymptotische Analyse

Satz 1 (Master-Theorem)

Seien $a \geq 1$ und $b \geq 1$ Konstanten, $f(n)$ eine Funktion, und $T(n)$ definiert über den natürlichen Zahlen als rekursive Funktion

$$T(n) = a \cdot T(n/b) + f(n),$$

wobei n/b entweder für $\lfloor n/b \rfloor$ oder für $\lceil n/b \rceil$ steht.

Dann hat $T(n)$ folgende asymptotische Schranken:

1. Wenn $f(n) = O(n^{\log_b a - \varepsilon})$ für eine Konstante $\varepsilon > 0$, dann ist $T(n) = \Theta(n^{\log_b a})$.

2. Wenn $f(n) = \Theta(n^{\log_b a})$, dann ist $T(n) = \Theta(n^{\log_b a} \lg n)$.

3. Wenn $f(n) = \Omega(n^{\log_b a + \varepsilon})$ für eine Konstante $\varepsilon > 0$ und
wenn $a \cdot f(n/b) \leq c \cdot f(n)$ für eine Konstante $c < 1$ und hinreichend große n,
dann ist $T(n) = \Theta(f(n))$

Asymptotische Analyse

Master-Theorem: Fall 1

Wenn $f(n) = O(n^{\log_b a - \varepsilon})$ für eine Konstante $\varepsilon > 0$, dann ist $T(n) = \Theta(n^{\log_b a})$.

Intuitiv: Die Basisfälle der Rekursion dominieren das Wachstum von $T(n)$.

Die Konstante ε stellt sicher, dass $f(n)$ polynomiell kleiner als $n^{\log_b a}$ ist. Andernfalls gelingt die Fallunterscheidung zu Fall 2 nicht immer.

Beispiele:

- $T(n) = 9T(n/3) + n$
  Vergleiche $f(n) = n$ mit $n^{\log_b a} = n^{\log_3 9} = n^2$.
  Für $\varepsilon = 1$, gilt $f(n) = O(n^{\log_3 9 - 1}) = O(n)$, so dass Fall 1 vorliegt.
  Es folgt $T(n) = \Theta(n^{\log_3 9}) = \Theta(n^2)$.

- $T(n) = 5T(n/2) + \Theta(n^2)$
  Vergleiche $f(n) = n^2$ mit $n^{\log_2 5} = O(n^{2.322})$.
  Es gibt ein $\varepsilon \approx 0.322$, so dass $\log_2 5 - \varepsilon = 2$ und $f(n) = O(n^2)$. Fall 1 liegt vor.
  Es folgt $T(n) = \Theta(n^{\lg 5})$.

Asymptotische Analyse

Master-Theorem: Fall 2

Wenn $f(n) = \Theta(n^{\log_b a})$, dann ist $T(n) = \Theta(n^{\log_b a} \lg n)$.

Intuitiv: Alle Rekursionschritte tragen gleichmäßig zum Wachstum von $T(n)$ bei.

Erweiterung für $f(n) = \Theta(n^{\log_b a} \cdot \lg^k n)$:

2a. Wenn $k > -1$, dann $T(n) = \Theta(n^{\log_b a} \cdot \lg^{k+1} n)$

2b. Wenn $k = -1$, dann $T(n) = \Theta(n^{\log_b a} \cdot \lg \lg n)$

2c. Wenn $k < -1$, dann $T(n) = \Theta(n^{\log_b a})$

Beispiele:

- $T(n) = T(2n/3) + 1$
  Vergleiche $f(n) = 1$ mit $n^{\log_{3/2} 1} = n^0 = 1$.
  Für $k = 0$ gilt $f(n) = \Theta(n^{\log_b a} \cdot \lg^0 n) = \Theta(1)$, so dass Fall 2a vorliegt.
  Es folgt $T(n) = \Theta(n^{\log_b a} \cdot \lg^{k+1} n) = \Theta(\lg n)$.

Asymptotische Analyse
Master-Theorem: Fall 2

Wenn $f(n) = \Theta(n^{\log_b a})$, dann ist $T(n) = \Theta(n^{\log_b a} \lg n)$.

Intuitiv: Alle Rekursionschritte tragen gleichmäßig zum Wachstum von $T(n)$ bei.

Erweiterung für $f(n) = \Theta(n^{\log_b a} \cdot \lg^k n)$:

2a. Wenn $k > -1$, dann $T(n) = \Theta(n^{\log_b a} \cdot \lg^{k+1} n)$

2b. Wenn $k = -1$, dann $T(n) = \Theta(n^{\log_b a} \cdot \lg \lg n)$

2c. Wenn $k < -1$, dann $T(n) = \Theta(n^{\log_b a})$

Beispiele:

- $T(n) = 27T(n/3) + \Theta(n^3 \lg n)$
  Vergleiche $f(n) = n^3 \lg n$ mit $n^{\log_3 27} = n^3$.
  Für $k = 1$ gilt $f(n) = \Theta(n^3 \lg n)$, so dass Fall 2a vorliegt.
  Es folgt $T(n) = \Theta(n^3 \lg^2 n)$.

Asymptotische Analyse

Master-Theorem: Fall 3

Wenn $f(n) = \Omega(n^{\log_b a+\varepsilon})$ für eine Konstante $\varepsilon > 0$ und
wenn $a \cdot f(n/b) \leq c \cdot f(n)$ für eine Konstante $c < 1$ und hinreichend große n,
dann ist $T(n) = \Theta(f(n))$

Intuitiv: Die Wurzel der Rekursion dominiert das Wachstum von $T(n)$.

Die zusätzliche Bedingung in Fall 3 heißt Regularitätsbedingung. Sie stellt sicher, dass das Wachstum auch bei rekursivem Anwenden von $f(n)$ dominiert wird.

Beispiele:

- $T(n) = 3T(n/4) + n \lg n$
  Vergleiche $f(n) = n \lg n$ mit $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$.
  Es gibt ein $\varepsilon \approx 0.207$, so dass $f(n) = \Omega(n^{\log_b a+\varepsilon}) = \Omega(n)$. Da außerdem $a \cdot f(n/b) = 3(n/4)\lg(n/4) \leq (3/4)n \lg n = c \cdot f(n)$ für $c = 3/4$ liegt Fall 3 vor.
  Es folgt $T(n) = \Theta(n \lg n)$.

Asymptotische Analyse

Master-Theorem: Fall 3

Wenn $f(n) = \Omega(n^{\log_b a + \varepsilon})$ für eine Konstante $\varepsilon > 0$ und
wenn $a \cdot f(n/b) \leq c \cdot f(n)$ für eine Konstante $c < 1$ und hinreichend große n,
dann ist $T(n) = \Theta(f(n))$

Intuitiv: Die Wurzel der Rekursion dominiert das Wachstum von $T(n)$.

Die zusätzliche Bedingung in Fall 3 heißt Regularitätsbedingung. Sie stellt sicher, dass das Wachstum auch bei rekursivem Anwenden von $f(n)$ dominiert wird.

Beispiele:

- $T(n) = 5T(n/2) + \Theta(n^3)$
  Vergleiche $f(n) = n^3$ mit $n^{\log_2 5} = O(n^{2.322})$.
  Es gibt ein $\varepsilon \approx 0.678$, so dass $f(n) = \Omega(n^{\log_b a + \varepsilon}) = \Omega(n^3)$. Da außerdem $a \cdot f(n/b) = 5(n/2)^3 \leq (5/8)n^3 = c \cdot f(n)$ für $c = 5/8$ liegt Fall 3 vor.
  Es folgt $T(n) = \Theta(n^3)$.

Das Master-Theorem ist nicht für alle rekursiven Funktionen definiert.

Beispiele:

- $T(n) = 2T(n/2) + n \lg n$
  Vergleiche $f(n) = n \lg n$ mit $n^{\log_b a} = n^{\log_2 2} = n$.
  Es gibt kein $\varepsilon > 0$ und $c > 0$, so dass $f(n) = \Omega(n^{\log_b a + \varepsilon}) = \Omega(n^{1+\varepsilon})$:

$$
\begin{array}{lcll}
 & 0 \leq c \cdot n^{1+\varepsilon} & \leq & n \lg n \quad \text{(Voraussetzung)} \\
\Leftrightarrow & 0 \leq c & \leq & \frac{\lg n}{n^{\varepsilon}} \\
\Leftrightarrow & \lim\limits_{n \to \infty} \frac{\lg n}{n^{\varepsilon}} & > & 0 \\
\Rightarrow & \lim\limits_{n \to \infty} \frac{1}{\varepsilon \cdot \ln(2) \cdot n^{\varepsilon}} & \not> & 0 \quad \text{(Satz von L'Hôpital)} \perp
\end{array}
$$

- $T(n) = 0.5T(n/2) + n$
- $T(n) = 64T(n/8) - n^2 \lg n$
- $T(n) = T(n/2) + n(2 - \cos n)$

Asymptotische Analyse

Master-Theorem: Definitionslücken

Das Master-Theorem ist nicht für alle rekursiven Funktionen definiert.

Beispiele:

- $T(n) = 2T(n/2) + n \lg n$
  Vergleiche $f(n) = n \lg n$ mit $n^{\log_b a} = n^{\log_2 2} = n$.
  Es gibt kein $\varepsilon > 0$ und $c > 0$, so dass $f(n) = \Omega(n^{\log_b a + \varepsilon}) = \Omega(n^{1+\varepsilon})$:

$$\begin{array}{rcccl} & 0 \leq & c \cdot n^{1+\varepsilon} & \leq & n \lg n \quad \text{(Voraussetzung)} \\ \Leftrightarrow & 0 \leq & c & \leq & \frac{\lg n}{n^{\varepsilon}} \\ \Leftrightarrow & & \lim\limits_{n\to\infty} \frac{\lg n}{n^{\varepsilon}} & > & 0 \\ \Rightarrow & & \lim\limits_{n\to\infty} \frac{1}{\varepsilon \cdot \ln(2) \cdot n^{\varepsilon}} & \not> & 0 \quad \text{(Satz von L'Hôpital)} \perp \end{array}$$

- $T(n) = 0.5T(n/2) + n$ (Teilproblemgröße $a < 1$)

- $T(n) = 64T(n/8) - n^2 \lg n$ (Ungültige Laufzeit: $f(n)$ ist negativ)

- $T(n) = T(n/2) + n(2 - \cos n)$ (Verletzung der Regularitätsbedingung von Fall 3)

Bemerkungen:

- Das Theorem wurde erstmals von von Bentley, Haken und Saxe vorgestellt [Bentley 1980]. Der Name „Master-Theorem“ wurde im Buch „Introduction to Algorithms“ von Cormen et al. erstmals verwendet.
- Das Theorem fußt auf dem Vergleich der Funktion $f(n)$ mit $g(n) = n^{\log_b a}$. Der Exponent $\log_b a$ wird auch „kritischer Exponent“ genannt. [plot]
- Für $f(n) = n^k$ und $f(n) = \Omega(n^{\log_b a} + \varepsilon)$, $\varepsilon > 0$, ist die Regularitätsbedingung von Fall 3 erfüllt.
- Vereinfachtes Master-Theorem, falls $f(n) = \Theta(n^k)$:
  1. Wenn $a > b^k$, dann $T(n) = \Theta(n^{\log_b a})$
  2. Wenn $a = b^k$, dann $T(n) = \Theta(n^k \log_b n)$
  3. Wenn $a < b^k$, dann $T(n) = \Theta(n^k)$

Formelsammlung

Monotonie

$f(n)$ steigt monoton, wenn $m \le n \Rightarrow f(m) \le f(n)$.

$f(n)$ fällt monoton, wenn $m \ge n \Rightarrow f(m) \ge f(n)$.

$f(n)$ steigt strikt monoton, wenn $m < n \Rightarrow f(m) < f(n)$.

$f(n)$ fällt strikt monoton, wenn $m > n \Rightarrow f(m) > f(n)$.

Abrundung und Aufrundung (*Floor and Ceiling*)

$\lfloor x \rfloor$ ist die größte ganzzahlige Zahl kleiner oder gleich x.

$\lceil x \rceil$ ist die kleinste ganzzahlige Zahl größer oder gleich x.

$x - 1 < \lfloor x \rfloor \le x \le \lceil x \rceil < x + 1$

Für alle $x \in \mathbf{R} : x \ge 0$ und $a, b \in \mathbf{N} : a, b > 0$:

$\left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil = \left\lceil \frac{x}{ab} \right\rceil$ und $\left\lfloor \frac{\lfloor x/a \rfloor}{b} \right\rfloor = \left\lfloor \frac{x}{ab} \right\rfloor$

$\left\lceil \frac{a}{b} \right\rceil \le \frac{a + (b-1)}{b}$ und $\left\lfloor \frac{a}{b} \right\rfloor \le \frac{a - (b-1)}{b}$

$f(x) = \lceil x \rceil$ und $g(x) = \lfloor x \rfloor$ steigen monoton.

Modulararithmetik

Für alle $a \in \mathbf{Z}$ und $n \in \mathbf{N}$: $a \bmod n = a - n\lfloor a/n \rfloor$

Es folgt $a \le a \bmod n < n$

$(a \bmod n) = (b \bmod n) \Leftrightarrow a \equiv b \pmod n$

Polynome

Polynom $p(n)$ über n vom Grad d: $p(n) = \sum_{i=0}^{d} a_i n^i$, wobei $a_0, \ldots, a_d$ Koeffizienten heißen und $a_d \ne 0$.

$p(n)$ ist asymptotisch positiv genau dann, wenn $a_d > 0$. Es folgt $p(n) = \Theta(n^d)$.

Für $k \ge 0$ steigt n^k monoton; $f(n)$ ist polynomiell beschränkt, wenn $f(n) = O(n^k)$

Exponenten

$a^0 = 1; \qquad a^1 = a; \qquad a^{-1} = 1/a; \qquad a^m a^n = a^{m+n}$

$(a^m)^n = a^{mn} = a^{nm} = (a^n)^m$

Für alle $a, b \in \mathbf{R}$: $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$ für $a > 1$, so dass $n^b = o(a^n)$.

Für $e = 2.71828\ldots$ und $x \in \mathbf{R}$:
$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$

Für alle $x \in \mathbf{R}$: $e^x \ge 1 + x$. Für $x \to 0$ geht $e^x \sim 1 + x$.

Fakultäten

$n! \sim \sqrt{(2\pi n)} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$

Für alle $n \ge 1$: $n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{a_n}$ für $\frac{1}{12n+1} < a_n < \frac{1}{12n}$

$n! = o(n^n); \quad n! = \omega(2^n); \quad \lg(n!) = \Theta(n \lg n)$

Formelsammlung (Fortsetzung)

Logarithmen

$\lg n = \log_2 n$; $\ln n = \log_e n$; $\lg^k n = (\lg n)^k$;

$\lg \lg n = \lg(\lg n)$; $\lg n + k = (\lg n) + k$, nicht $\lg(n+k)$

Für all $a, b, c \in \mathbf{R} : a, b, c > 0$ und $b \neq 1$ und $n \in \mathbf{N}$:

$a = b^{\log_b a}$; $\log_c(ab) = \log_c a + \log_c b$; $\log_b a^n = n \log_b a$;

$\log_b a = \frac{\log_c a}{\log_c b}$; $\log_b(1/a) = -\log_b a$; $\log_b a = \frac{1}{\log_a b}$;

$a^{\log_b c} = c^{\log_b a}$

Für $|x| < 1$: $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \ldots$

Für $x > -1$: $\frac{x}{1+x} \leq \ln(1+x) \leq x$; Gleichheit nur für $x = 0$.

Wenn $f(n) = O(\lg^k n)$ heißt $f(n)$ polylogarithmisch beschränkt.

$\lim\limits_{n \to \infty} \frac{\lg^b n}{n^a} = 0$; es folgt $\lg^b n = o(n^a)$ für $a > 0$.

Iterierte Funktionen

$f^{(i)}(n)$ bezeichnet die i-fache iterative Anwendung von $f(n)$:

$$f^{(i)}(n) = \begin{cases} n & \text{wenn } i = 0 \\ f(f^{(i-1)}(n)) & \text{wenn } i > 0 \end{cases}$$

Beispiel: Wenn $f(n) = 2n$, dann $f^{(i)}(n) = 2^i n$.

Iterierte Logarithmen

$$\lg^*(n) = \begin{cases} 0 & \text{wenn } n \leq 0 \\ 1 + \lg^*(\lg n) & \text{wenn } n > 1 \end{cases}$$

Intervall	$\lg^* n$
$(-\infty, 1]$	0
$(1, 2]$	1
$(2, 4]$	2
$(4, 16]$	3
$(16, 65536]$	4
$(65536, 2^{65536}]$	5

Fibonacci-Zahlen

Seien ϕ (goldener Schnitt) und $\hat{\phi}$ die Lösungen von $x^2 = x + 1$: Dann ist i-te Fibnonacci-Zahl $F_i = \dfrac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$,

wobei $\phi = \dfrac{1+\sqrt{5}}{2} = 1.618\ldots$ und $\hat{\phi} = \dfrac{1-\sqrt{5}}{2} = -0.618\ldots$

Da $|\hat{\phi}| < 1$ gilt $\dfrac{|\hat{\phi}^i|}{\sqrt{5}} < \dfrac{1}{\sqrt{5}} < \dfrac{1}{2}$, so dass $F_i = \left\lfloor \dfrac{\phi^i}{\sqrt{5}} + \dfrac{1}{2} \right\rfloor$.

Damit wachsen die Fibonacci-Zahlen exponentiell.