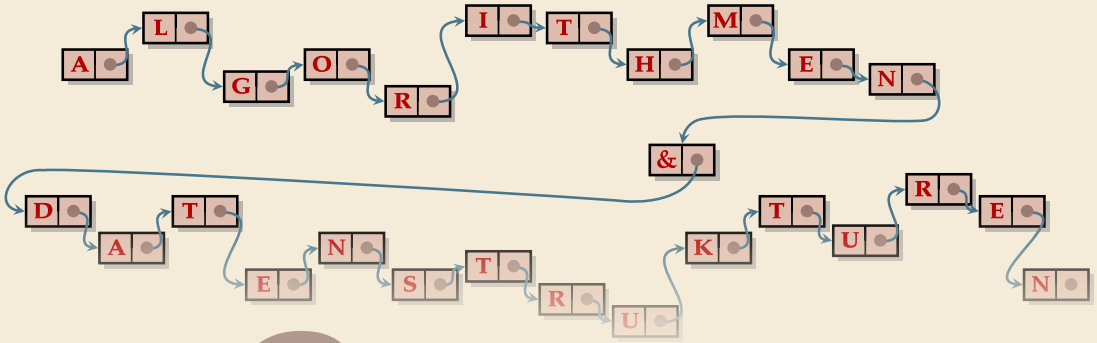


3 Mathematische Grundlagen

Algorithmen & Datenstrukturen · Sommersemester 2026

Prof. Dr. Sebastian Wild

CS210 (Sommersemester 2026)
Philipps-Universität Marburg
version `2026-04-29 17:18 H`

Outline

3 Mathematische Grundlagen

Recap: Skalierbarkeit

Analogie: Architektur

- 1m Brücke: Holzplanke genügt!
- 10x ⇝ Konstruktion nötig
- 1000x ⇝ Meisterleistung der Bauingenieurskunst

Informatik

- Eingabegrößen in Informatiksystemen erreichen oft Faktor 1 000 000 000!

  1 Byte → 1 GB

⇝ *Können uns keine Holzbretter leisten.*

Eine 1km Brücke ist nicht einfach ein 1000-fach größeres Brett.

2m	~10x →	12m	~10x →	170m	~10x →	1280m
€ 50 000	~10x →	€ 350 000	~50x →	€ 22 000 000	~200x →	€ 4 500 000 000

(Kosten sind Schätzungen der heutigen Kosten)

Wie quantifizieren wir “Skalierbarkeit”?

- ⇝ *Für Algorithmen (in erster Näherung)* ***Wachstum*** *der Kosten in Eingabegröße entscheidend.*
- Tricks in der Implementierung und bessere Hardware retten uns nicht
- tragen aber natürlich zu den Gesamtkosten bei
- ⇝ für Performance-Bottlenecks durch Experten noch etwas rauszuholen
- ⇝ den deutlich größeren Impact hat aber die Wahl der richtigen Algorithmen und Datenstrukturen

Wie können wir also Algorithmen bezüglich des ***Wachstums*** *der Kosten* ***vergleichen****?*

- Sprache für Performancecharakteristika: **Asymptotische Approximationen**
  O-Notation (plus Varianten und Verfeinerungen)
- ⇝ Damit müssen Sie souverän umgehen lernen.

Dazu legen wir hier die Grundlagen und geben die formalen Definitionen.

3.1 Beispiel: 3SUM

3SUM

3SUM-Problem

- **Gegeben:** $A[0..n)$ with $A[i] \in \mathbb{Z}$ for $0 \leq i < n$.
- **Ziel:** Anzahl c von Tripeln aus A mit Summe 0

$$c \quad = \quad \left|\left\{(i,j,k) \;:\; 0 \leq i < j < k < n \;\wedge\; A[i] + A[j] + A[k] = 0\right\}\right|$$

Parameter:

- Eingabegröße: #Zahlen n

Anwendungen: 3 Punkte auf einer Geraden

- kollineare Tripel von Punkten finden
- Tripel von Geraden finden, die sich in einem Punkt schneiden
- Dreiecke aus Punkten formen mit kleinem Flächeninhalt
- ...

Einschub: Template für Entwurf von Algorithmen

1. **Algorithmische Idee**
   Kurze abstrakte Idee, die dem Algorithmus zu Grunde liegt (Freitext)
   (ein Experte könnte den Rest von hier ergänzen)

2. **Pseudocode**
   struktierte Beschreibung der Methode inklusive Rand- und Basisfälle
   eindeutig, präzise; nah an echtem Code

3. **Korrektheitsbeweis**
   Argument, warum der Algorithmus das korrekte Ergebnis liefert
   oft per Induktion unter Verwendung von Invarianten

4. **Analyse**
   Analyse der Kosten des Algorithmus
   i. d. R. Θ-Klasse für den Worst Case;
   wo interessant auch Speicherbedarf

Brute-force 3SUM

- allgemeiner Ratschlag: erstmal die denkbar einfachste Lösung versuchen

1. **Algorithmische Idee**
Alle Tripel durchprobieren

2. **Pseudocode**

```
1  c = 0
2  for i = 0, ..., n − 3
3      for j = i + 1, ..., n − 2
4          for k = j + 1, ..., n − 1
5              t = A[i] + A[j] + A[k]
6              if t == 0 then c := c + 1
7          end for
8      end for
9  end for
10 return c
```

3. **Korrektheitsbeweis**
direkt nach Definition von c

Analysen später nochmal im Detail

4. **Analyse**

Ausführungen von Zeilen 5–6

$$= \sum_{i=0}^{n-3} \sum_{j=i+1}^{n-2} \sum_{k=j+1}^{n-1} 1 = \sum_{i=0}^{n-3} \sum_{j=i+1}^{n-2} (n-(j+1))$$

$$= \sum_{i=0}^{n-3} \sum_{j=1}^{n-i-2} j = \sum_{i=0}^{n-3} \frac{(n-i-2)(n-i-1)}{2}$$

$$= \frac{1}{2} \sum_{i=1}^{n-2} i(i+1) = \frac{1}{2} \sum_{i=1}^{n-2} i^2 + \frac{1}{2} \sum_{i=1}^{n-2} i$$

$$= \frac{(n-2)(n-1)(2(n-2)+1)}{12} + \frac{(n-2)(n-1)}{4}$$

$$= \frac{(n-2)(n-1)n}{6}$$

Ist das jetzt gut??

Einschub: Überschlagsrechnungen (*back-of-the-envelope math*)

Ob $T(n) = \frac{(n-2)(n-1)n}{6}$ Operationen machbar sind, hängt von n ab!

Ganz grobe Überschlagsrechnungen sind hier hilfreich!

n	$T(n)$	**Zeit bei 1GHz / 1ns pro Operation**
10	120	instant
100	161 700	0.1 ms
1 000	166 167 000	0.16 s
10 000	166 616 670 000	2:40 min
100 000	166 661 666 700 000	46 h (!)

⇝ Arrays mit 10 000 Zahlen grade noch machbar, 100 000 eher nicht.

Übrigens: Man kann 3SUM deutlich schneller lösen . . .
werden wir noch einmal wiedersehen 😉

3.2 Folgen & Grenzwerte

Definitionen zur Erinnerung

Ein bisschen Notation

- $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots, \}$ natürliche Zahlen mit 0
- $\mathbb{N}_{\geq 1} = \{1, 2, 3, \ldots, \}$ natürliche Zahlen ab 1
- ~~$\mathbb{N}$~~: (Uneindeutigkeit vermeiden)
- $\mathbb{R}$: reelle Zahlen
- $\lg = \log_2$; $\ln = \log_e$

Folgen

- (Reelle) *Zahlenfolge* $(a_n)_{n \in \mathbb{N}_{\geq 1}}$
  formal eine Funktion $a : \mathbb{N}_{\geq 1} \to \mathbb{R}$
  schreiben auch "$a_1, a_2, a_3, \ldots$"
- *Grenzwert* einer Folge
  $\lim\limits_{n \to \infty} a_n = a^*$ genau dann, wenn $\boxed{\forall \varepsilon > 0 \; \exists n_0 \in \mathbb{N}_{\geq 1} \; \forall n \geq n_0 : |a_n - a^*| < \varepsilon}$
- Nicht jede Folge hat einen Grenzwert, aber wenn er existiert, dann ist er eindeutig.

Folgen – Beispiel

Es sei die Folge $(a_n)_{n \in \mathbb{N}_{\geq 1}}$ mit $a_n = \frac{1}{n}$ gegeben.

Es gibt: $\lim_{n \to \infty} a_n = 0$

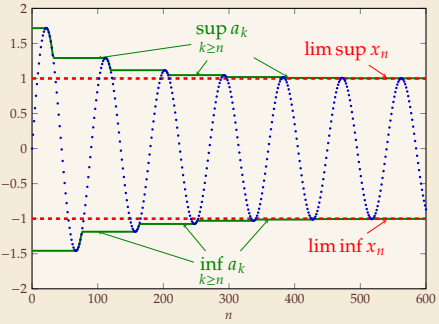
Grenzwerte von Funktionen

$$\lim_{x \to \xi} f(x) = f^* \text{ gdw } \forall \varepsilon > 0 \; \exists \delta > 0 \; \forall x \in \mathbb{R} : |x - \xi| < \delta \implies |f(x) - f^*| < \varepsilon$$

"ε-δ-Kriterium"

Limes superior, Limes inferior

- Supremum ist die kleinste obere Schranke
  Für $M \subseteq \mathbb{R}$ ist $\sup M = \min\{b \in \mathbb{R} \,:\, \forall x \in M : x \leq b\}$
- Analog ist Infimum die größte untere Schranke
- Beide müssen nicht existieren ($+\infty$ bzw. $-\infty$); wenn sie existieren, sind sie eindeutig.



- Für Folge $(a_n)_{n \in \mathbb{N}_{\geq 1}}$ ist der *Limes superior*:

$$\limsup_{n \to \infty} a_n \;=\; \inf\Big\{\sup\big\{a_k : k \geq n\big\} : n \in \mathbb{N}_{\geq 1}\Big\}$$

  Äquivalent: größter Häufungspunkt von (a_n);
  beste obere Schranke für a_n
  für genügend große n

- Analog Limes inferior: $\liminf_{n \to \infty} a_n \;=\; \sup\Big\{\inf\big\{a_k : k \geq n\big\} : n \in \mathbb{N}_{\geq 1}\Big\}$

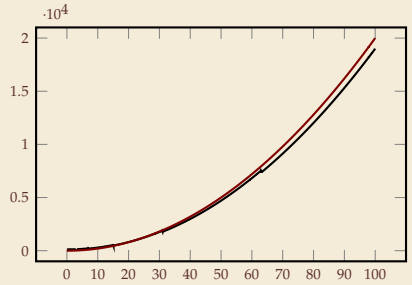
3.3 Asymptotische Notation

Asymptotische Vergleiche

Asymptotik = Approximation um ∞

Beispiel: Betrachte Funktion $f(n)$ mit

$$2n^2 - 3n\lfloor\log_2(n+1)\rfloor + 7n - 3\lfloor\log_2(n+1)\rfloor + 120 \quad \sim \quad \mathbf{2n^2}$$



Asymptotische Approximationen in der Analyse von Algorithmen

- abstrahiert von unwichtigen Details
- vereinfacht die Analyse selbst
- ermöglicht oft erst konklusiven Vergleich

Asymptotische Notation – Formale Definition

- **"Tilde Notation":** $f(n) \sim g(n)$ gdw $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$

  genau dann, wenn (*iff*)

  „f und g sind *asymptotisch äquivalent*"

- **"Big-Oh Notation":** $f(n) \in O\big(g(n)\big)$ gdw $\left|\frac{f(n)}{g(n)}\right|$ beschränkt für $n \geq n_0$

  schreiben auch '=' stattdessen

  gdw $\limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \infty$

  brauchen Supremum da Limes evtl. nicht existiert!

  Varianten:

  "Big-Omega"

  - $f(n) \in \Omega\big(g(n)\big)$ gdw $g(n) \in O\big(f(n)\big)$
  - $f(n) \in \Theta\big(g(n)\big)$ gdw $f(n) \in O\big(g(n)\big)$ **und** $f(n) \in \Omega\big(g(n)\big)$

  "Big-Theta"

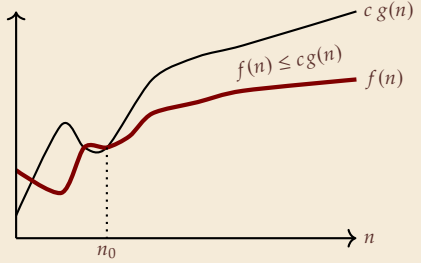
- **"Little-Oh Notation":** $f(n) \in o\big(g(n)\big)$ gdw $\lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| = 0$

  analog: $f(n) \in \omega\big(g(n)\big)$ falls $|f(n)/g(n)| \to \infty$

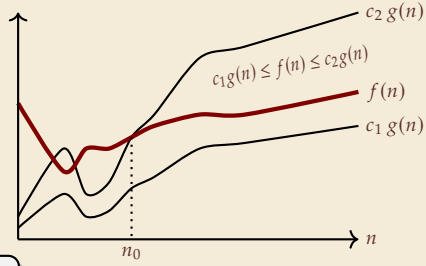
(Vorteil dieser Definition: Funktioniert genauso für $f, g : \mathbb{R} \to \mathbb{R}$ und für andere Grenzprozesse als $n \to \infty$)

Asymptotische Notation – Intuition

- $f(n) = O(g(n))$: $f(n)$ ist **höchstens** $g(n)$
  bis auf konstante Faktoren
  und
  für hinreichend große n



- $f(n) = \Theta(g(n))$: $f(n)$ ist **gleich** $g(n)$
  bis auf konstante Faktoren
  und
  für hinreichend große n



Plots können irreführend sein!

Beispiel

O-Notation – Beispiel

$f(n) = 3n^3 + 2n^2, \quad g(n) = n^3$

3.4 Rechenregeln für Asymptotiken

O-Notation – Beispiel 2

$n \ln n = o(n^2)$

Regel von de l'Hospital

Bei Grenzwerten der Form $\frac{\infty}{\infty}$ bzw. $\frac{0}{0}$ kann man oft folgende Regel anwenden.

Theorem 3.1 (de l'Hospital)

Gegeben Funktionen $f, g : \mathbb{R}_{>n_0} \to \mathbb{R}$, die

1. *differenzierbar* auf $\mathbb{R}_{>n_0}$ sind,
2. $\forall x \in \mathbb{R}_{>n_0} : g'(x) \neq 0$,
3. $\lim_{x\to\infty} |f(x)| = \lim_{x\to\infty} |g(x)| \in \{0, \infty\}$, und
4. $\lim_{x\to\infty} \frac{f'(x)}{g'(x)}$ existiert

Dann ist $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$.

Rechenregeln

Lemma 3.2 (Rechenregeln für asymptotische Notationen)

Es gilt:

(a) $f = O(f)$, $f = \Theta(f)$, $f = \Omega(f)$ *(Reflexivität)*.

(b) $c \cdot O(f) = O(f)$ für c konstant.

(c) $O(f) + O(g) = O(f + g) = O(\max(f, g))$.

(d) $O(O(f)) = O(f)$.

(e) $O(f) \cdot O(g) = O(f \cdot g)$.

(f) Falls $f \sim g$, dann gilt $f = g \cdot (1 \pm o(1))$.

(g) $f = O(g) \leftrightarrow g = \Omega(f)$ *(Schiefsymmetrie)*,
$f = \Theta(g) \leftrightarrow g = \Theta(f)$ *(Symmetrie)*.

(h) $\Omega(f) + \Omega(g) = \Omega(\min(f, g))$.

(i) $f = \circ(g) \wedge g = \circ(h) \rightarrow f = \circ(h)$ für $\circ \in \{O, o, \Omega, \omega, \Theta\}$ *(Transitivität)*.

O-Notation – Quantoren-Definition

Für alle asymptotischen Notationen $f(n) = O(g(n))$: Grenzwert des Quotienten $\dfrac{f(n)}{g(n)}$

Alternative Definition (EAA Buch) (äquivalent für $f : \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq 0}$)

Definition 1.16 (Asymptotische Notation [Landau-Klassen]):
Sei $\mathcal{F} := \mathsf{Abb}(\mathbb{N}_0, \mathbb{R}_0^+)$ *die Menge der Abbildungen von der Menge der natürlichen Zahlen inklusive der Null in die Menge der nicht-negativen reellen Zahlen. Sei ferner* $g \in \mathcal{F}$. *Wir definieren die folgenden Mengen von Funktionen:*

1.) $\mathcal{O}(g) := \big\{ f \in \mathcal{F} \,\big|\, (\exists c > 0)(\exists n_0 \in \mathbb{N}_0)(\forall n \geq n_0)\ (f(n) \leq c \cdot g(n)) \big\}$,

2.) $\Omega(g) := \big\{ f \in \mathcal{F} \,\big|\, (\exists c > 0)(\exists n_0 \in \mathbb{N}_0)(\forall n \geq n_0)\ (f(n) \geq c \cdot g(n)) \big\}$,

3.) $\Theta(g) := \mathcal{O}(g) \cap \Omega(g)$,

4.) $o(g) := \big\{ f \in \mathcal{F} \,\big|\, (\forall c > 0)(\exists n_0 \in \mathbb{N}_0)(\forall n \geq n_0)\ (f(n) \leq c \cdot g(n)) \big\}$,

5.) $\omega(g) := \big\{ f \in \mathcal{F} \,\big|\, (\forall c > 0)(\exists n_0 \in \mathbb{N}_0)(\forall n \geq n_0)\ (f(n) \geq c \cdot g(n)) \big\}$,

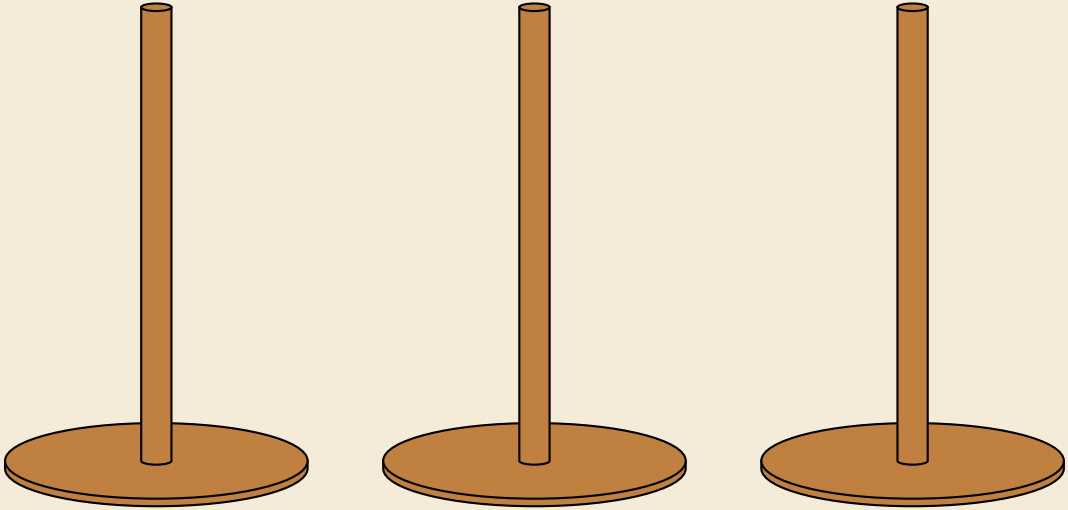
6.) $\sim(g) := \big\{ f \in \mathcal{F} \,\big|\, (\forall \varepsilon > 0)(\exists n_0 \in \mathbb{N}_0)(\forall n \geq n_0)\ \big(\big|\frac{f(n)}{g(n)} - 1\big| \leq \varepsilon\big) \big\}$.

Hier nehmen wir an, dass $g(n) = 0$ *höchstens für endlich viele* n *gilt.*

(Beachte die unterschiedlichen Quantoren bei c *bzw.* ε*!)* ◀

3.5 Rekursiv definierte Folgen

Die “Türme von Hanoi”



Rekursive Lösung

```
1 procedure towersOfHanoi(s, t, b, n)
2     // Move topmost n pieces from s to t
3     // Assumes the topmost n disks are on pile s
4     if n == 0 return
5     towersOfHanoi(s, b, t, n − 1)
6     move disk n from s to t
7     towersOfHanoi(b, t, s, n − 1)
```

- einfacher Code
- landet gezielt auf korrektem Stab!

https://upload.wikimedia.org/wikipedia/commons/2/20/Tower_of_Hanoi_recursion_SMIL.svg

s: Quellstab (source)
t: Zielstab (target)
b: Hilfsstab (buffer)

- aber wie lange braucht der Code denn?

Anzahl Bewegungen

- Nur Zeile 6 bewegt eine Scheibe! ... aber natürlich auch rekursiv aufgerufen.
  - Für $n = 0$ keine weitere Bewegung; sonst eine Bewegung und 2 mal die Bewegungen für $n - 1$

$\rightsquigarrow$ *Rekursionsgleichung / Rekurrenz*: $T : \mathbb{N}_{\geq 0} \to \mathbb{N}_{\geq 0}$ $T(n) =$ # Bewegungen für n Scheiben

$$T(0) = 0, \qquad T(n) = T(n-1) + 1 + T(n-1) = 2T(n-1) + 1 \qquad (n \geq 1)$$

Rekursion und Rekurrenzen

$$T(0) = 0, \quad T(n) = 2T(n-1)+1 \quad (n \geq 1)$$

1. Werte für kleine n

n	$T(n)$
0	0
1	$2 \cdot 0 + 1 = 1$
2	$2 \cdot 1 + 1 = 3$
3	$2 \cdot 3 + 1 = 7$
4	$2 \cdot 7 + 1 = 15$
5	$2 \cdot 15 + 1 = 31$
6	$2 \cdot 31 + 1 = 63$

(Informatiker erkennen Zweierpotenzen 😉)

2. Lösung raten

(manchmal schwierig!!)

Oft hilfreich: einsetzen!

$$\begin{aligned} T(n) &= 2T(n-1)+1 \\ &= 2\big(2T(n-2)+1\big)+1 \\ &= 2^2 T(n-2)+2+1 \\ &= 2^3 T(n-3)+2^2+2+1 \\ &= 2^n T(n-n)+ \\ &\quad 2^{n-1}+\cdots+2+1 \\ &= \sum_{i=0}^{n-1} 2^i \;=\; 2^n-1 \end{aligned}$$

$\rightsquigarrow$ *"educated guess"*:

$$\boxed{T(n) = 2^n - 1}$$

3. Per Induktion beweisen

Lemma 3.3 (Türme von Hanoi)

$T(n) = 2^n - 1$.

Beweis:

Per Induktion über n.

- Induktionsanfang (IA): $n = 0$

  $T(0) = 0 = 2^0 - 1$ ✓
- Induktionsvoraussetzung (IV):

  $\forall n' \leq n \;:\; T(n') = 2^{n'} - 1$
- Induktionsschritt (IS): $n \to n+1$

$$\begin{aligned} T(n+1) &= 2T(n)+1 \\ &\overset{\text{IV}}{=} 2\big(2^n-1\big)+1 \\ &= 2^{n+1}-2+1 \\ &= 2^{n+1}-1 \end{aligned}$$

■

3.6 Das Master-Theorem

Divide-and-Conquer Rekursionen

Wir werden eine Reihe von Algorithmen sehen die nach dem
Divide-&-Conquer-Prinzip (*„Teile und herrsche“*) arbeiten.

Klassisches Beispiel: *Mergesort* (⇝ Unit 7)

0. Falls Eingabegröße $n \leq 1$, **fertig**.
1. *Teile* die Eingabe der Größe n in **2** Teile der Größe $\approx \frac{n}{\mathbf{2}}$ auf.
2. Löse **rekursiv** die **2** Teilprobleme der Größe $\frac{n}{\mathbf{2}}$.
3. **Kombiniere** die Lösungen der Teilprobleme zu einer Gesamtlösung.

(Wir kümmern uns hier (noch) nicht darum, wie/was das algorithmisch tut!)

Aber wir wollen Verfahren kennenlernen, wie man solche Algorithmen analysieren kann.

Laufzeit von Mergesort

$$T(n) = \begin{cases} \mathbf{0} & n \leq 1 \\ \mathbf{2} \cdot T(n/\mathbf{2}) + \boldsymbol{n} & n \geq 2 \end{cases}$$

Für $n = 2^k$, $k \in \mathbb{N}_0$

Rekursionsbaum-Methode für Mergesort

Die Master-Methode

- Allgemeines D&C-Verfahren
  - a rekursive Aufrufe (für Konstante $a > 0$)
  - Teilprobleme der Größe n/b (für Konstante $b > 1$)
  - nicht-rekursiver "conquer" Aufwand $f(n)$ (für Funktion $f : \mathbb{R} \to \mathbb{R}$)
  - Basisfall-Aufwand d (some constant $d > 0$)

$\rightsquigarrow$ Laufzeit $T(n)$ erfüllt Rekursion

$$T(n) = \begin{cases} a \cdot T\left(\frac{n}{b}\right) + f(n) & n > 1 \\ d & n \leq 1 \end{cases} \qquad (n = b^k, k \in \mathbb{N}_0)$$

Theorem 3.4 (Master Theorem)
Mit $c := \log_b(a)$, gilt für die obige Rekursion:

(a) $T(n) = \Theta(n^c)$ falls $f(n) = O(n^{c-\varepsilon})$ für konstantes $\varepsilon > 0$.

(b) $T(n) = \Theta(n^c \log n)$ falls $f(n) = \Theta(n^c)$.

(c) $T(n) = \Theta(f(n))$ falls $f(n) = \Omega(n^{c+\varepsilon})$ für konstantes $\varepsilon > 0$ **und** f erfüllt die *Regularitätsbedingung* $\exists n_0, \alpha < 1 \; \forall n \geq n_0 \; : \; a \cdot f\left(\frac{n}{b}\right) \leq \alpha f(n)$.

Regularitätsbedingung

Fall (c) braucht leider die technische Regularitätsbedingung.

Aber, für einfache Funktionen f gilt diese immer.

Lemma 3.5 (Regularitätsbedingung)

Für $f(n) = n^\beta \lg^\gamma(n)$ mit Konstanten $\beta > \log_b(a)$ und γ gilt die Regularitätsbedingung. ◀

Master Theorem – Recursion Tree

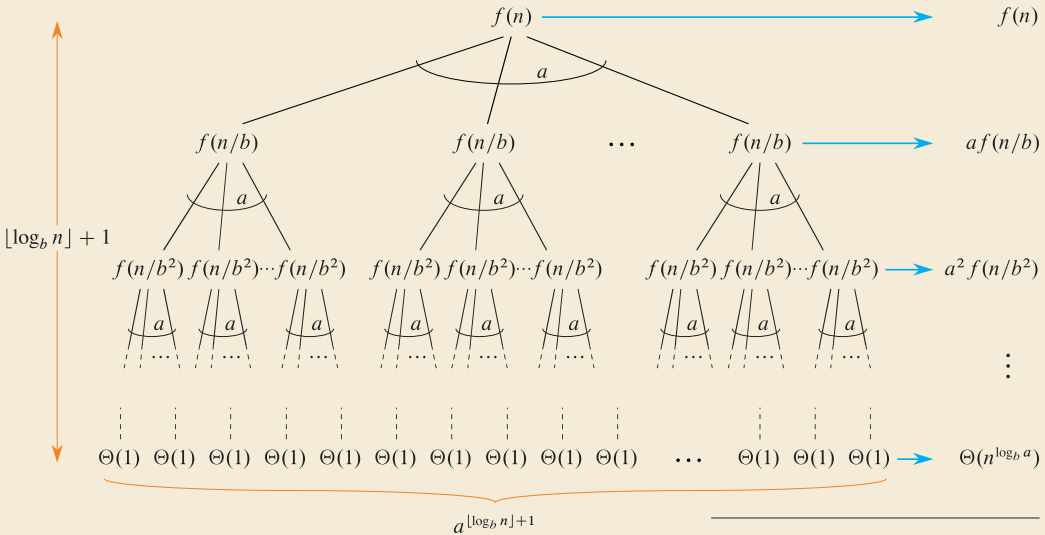


Figure 4.3 of Cormen et al. *Introduction to Algorithms* 4th ed.

Was ist mit Runden?

Für allgemeines n sind die Teilproblem $\lfloor \frac{n}{b} \rfloor$ bzw. $\lceil \frac{n}{b} \rceil$. *Macht das was?*

Im Allgemeinen Ja, aber:

Lemma 3.6 (Polynomial-growth master method)

Falls f die *polynomial-growth condition* erfüllt, ist die Θ-Klasse der Lösung der D&C-Rekursion mit oder ohne Runden gleich. ◀

- $f : \mathbb{R}_{>0} \to \mathbb{R}$ erfüllt die *polynomial-growth condition*, falls

  $$\exists n_0\ \forall C \geq 1\ \exists D > 1 \quad \forall n \geq n_0\ \forall c \in [1, C] \quad : \quad \tfrac{1}{D} f(n) \leq f(cn) \leq D f(n)$$

- Intuition n um (bis zu) konstanten Faktor C größer machen
  verändert Funktionswert auch höchstens um konstanten Faktor $D = D(C)$
  (für hinreichend große n)

- Insbesondere: $f(n) = \Theta(n^\beta \log^\gamma(n) \log\log^\delta(n))$ für Konstanten β, γ, δ
  erfüllt *polynomial-growth condition*

  Null erlaubt!

A Rigorous and Stronger Meta Theorem

Theorem 3.7 (Roura's Discrete Master Theorem)

Let $T(n)$ be recursively defined as

$$T(n) \;=\; \begin{cases} b_n & 0 \le n < n_0, \\ f(n) + \displaystyle\sum_{d=1}^{D} a_d \cdot T\Big(\frac{n}{b_d} + r_{n,d}\Big) & n \ge n_0, \end{cases}$$

where $D \in \mathbb{N}$, $a_d > 0$, $b_d > 1$, for $d = 1, \ldots, D$ are constants, functions $r_{n,d}$ satisfy $|r_{n,d}| = O(1)$ as $n \to \infty$, and function $f(n)$ satisfies $f(n) \sim B \cdot n^{\alpha} (\ln n)^{\gamma}$ for constants $B > 0$, α, γ.
Set $H = 1 - \sum_{d=1}^{D} a_d \, (1/b_d)^{\alpha}$; then we have:

(a) If $H < 0$, then $T(n) = O(n^{\tilde{\alpha}})$, for $\tilde{\alpha}$ the unique value of α that would make $H = 0$.

(b) If $H = 0$ and $\gamma > -1$, then $T(n) \sim f(n) \ln(n)/\tilde{H}$ with constant $\tilde{H} = (\gamma + 1) \sum_{d=1}^{D} a_d \, b_d^{-\alpha} \ln(b_d)$.

(c) If $H = 0$ and $\gamma = -1$, then $T(n) \sim f(n) \ln(n) \ln(\ln(n))/\hat{H}$ with constant $\hat{H} = \sum_{d=1}^{D} a_d \, b_d^{-\alpha} \ln(b_d)$.

(d) If $H = 0$ and $\gamma < -1$, then $T(n) = O(n^{\alpha})$.

(e) If $H > 0$, then $T(n) \sim f(n)/H$.

3.7 Grundlagen Stochastik

Grundbegriffe der Stochastik

- Betrachten *Zufallsexperimente*, deren Ergebnisse von zufällig wirkenden Faktoren abhängen (z. B. Würfeln)
- Ein *Elementarereignis* ω ist ein unmittelbar mögliches Ergebnis des betrachteten Zufallsexperiments
  Menge möglicher Elementarereignisse ist der *Ereignisraum* Ω
- (für abzählbar unendliche oder endliche Ω)

  $A \subseteq \Omega$ ein *Ereignis* (z. B. Würfel zeigt gerade Zahl)
- Wir ordnen A die *Wahrscheinlichkeit* $\mathbb{P}[A]$ zu
  1. $\mathbb{P}[A] \geq 0$ für alle $A \subset \Omega$.
  2. $\mathbb{P}[\Omega] = 1$.
  3. Für $A_1, A_2, \ldots$ **disjunkte** Ereignisse in Ω, ist

     $$\mathbb{P}[A_1 \cup A_2 \cup \cdots] = \sum_{i=1}^{\infty} \mathbb{P}[A_i]$$

     (*Ver-oder-ung* der die Ereignisse definierenden Bedingungen).

Grundbegriffe der Stochastik [2]

- Die *bedingte Wahrscheinlichkeit* für Ereignis A, unter der Voraussetzung, dass B bereits eingetreten ist, ist definiert als

$$\mathbb{P}[A \mid B] \;:=\; \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]}, \qquad \mathbb{P}[B] > 0.$$

- A und B sind *(stochastisch) unabhängig*, falls $\mathbb{P}[A \mid B] \;=\; \mathbb{P}[A]$
- Damit gilt für **unabhängige** Ereignisse $\mathbb{P}[A \cap B] \;=\; \mathbb{P}[A] \cdot \mathbb{P}[B]$
  (anschaulich: *Ver-und-ung* der Ereignisse entspricht)

Zufallsvariablen

- Eine *Zufallsvariable* ist eine Funktion $X : \Omega \to \mathbb{R}$
  Schreiben X statt $X(\omega)$.
- Ist der Wertebereich von X gleich $\{a_i : i \in \mathcal{I}\}$, dann ist

  $$\mathbb{P}[X = a_i] \;=\; \sum_{\substack{\omega \in \Omega \\ X(\omega) = a_i}} \mathbb{P}[\omega], \qquad i \in \mathcal{I},$$

  die zugehörige *Wahrscheinlichkeitsverteilung* von X.
- Für $\mathcal{I}$ endlich bzw. abzählbar unendlich heißt X *diskrete* Zufallsvariable.

- **Beispiele**
  - **Uniforme Verteilung:** für $|\mathcal{I}| = n < \infty$
    $\mathbb{P}[X = a_i] = \frac{1}{n}$
  - **Bernoulli Verteilung:**
    $\mathcal{I} = \{0, 1\}$
    $\mathbb{P}[X = 1] = p$, $\mathbb{P}[X = 0] = 1 - p$,

Beispiel Zufallsvariablen

Gegeben ein roter und ein grüner fairer Würfel.

$\rightsquigarrow$ Ereignisraum $\Omega = \{1,2,3,4,5,6\} \times \{1,2,3,4,5,6\}$

$= \{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),$
$(2,1),(2,2),(2,3),(2,4),(2,5),(2,6),$
$(3,1),(3,2),(3,3),(3,4),(3,5),(3,6),$
$(4,1),(4,2),(4,3),(4,4),(4,5),(4,6),$
$(5,1),(5,2),(5,3),(5,4),(5,5),(5,6),$
$(6,1),(6,2),(6,3),(6,4),(6,5),(6,6)\}$

- Zufallsvariable X = Anzahl Augen des roten Würfels
  formal $X(3,4) = 3$ etc.
- analog Y = Anzahl Augen des grünen Würfels
  $Y(3,4) = 5$
- können mit $X + Y$ die Summe der gewürfelten Zahlen bilden
  $(X+Y)(3,4) = 8$

$\rightsquigarrow$ Durch Zufallsvariablen können wir mit Ergebnissen von Zufallsexperimenten *rechnen*!

- $\mathbb{P}[X+Y=8] = \dfrac{\left|\{(2,6),(3,5),(4,4),(5,3),(6,2)\}\right|}{36} = \dfrac{5}{36}$

Erwartungswert

Definition 3.8

Ist X eine diskrete Zufallsvariable mit Werten in $\mathcal{X} \subset \mathbb{R}$ dann ist der *Erwartungswert von* X definiert als $\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x \cdot \mathbb{P}[X = x]$. ◀

Lemma 3.9 (Linearität des Erwartungswerts)

Für reellen Zahlen a, b und c und Zufallsvariablen X und Y ist

$$\mathbb{E}[a \cdot X + b \cdot Y + c] = a \cdot \mathbb{E}[X] + b \cdot \mathbb{E}[Y] + c. \quad ◀$$

Linearität gilt für beliebige X und Y, auch stochastisch abhängige.