

# Logik

## Aussagenlogik

**Aussage** Satz/Formel entweder wahr oder falsch; „-form“ bei zu wenig Infos.

**Theoreme** sind wahre Aussagen.

## Junktoren

**Negation**  $\neg A$  „Nicht“ (!, ~,  $\neg$ )

**Konjunkt.**  $A \wedge B$  „und“ (&,  $\cap$ )

**Disjunkt.**  $A \vee B$  „oder“ (||,  $\cup$ )

**Implikat.**  $A \Rightarrow B$  „Wenn, dann“ / „B“ ( $\rightarrow$ ,  $\Rightarrow$ )

$A \Rightarrow B$  „A hinreichend“

$B \Rightarrow A$  „A notwendig“

**Äquiv.**  $A \Leftrightarrow B$  „Genau dann, wenn“ ( $\leftrightarrow$ ,  $\equiv$ ,  $\Leftrightarrow$ )

**Wahrheitswertetabelle** mit  $2^n$  Zeilen für  $n$  Atome. Konstruktionssystematik: Frequenz pro Atom verdoppeln.

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Äquivalente Formeln $\Leftrightarrow$	Bezeichnung
$A \wedge B$ $A \vee B$ $A \wedge (B \wedge C)$ $A \vee (B \vee C)$ $A \wedge (B \vee C)$ $A \vee (B \wedge C)$ $A \wedge A$ $A \vee A$ $\neg \neg A$ $\neg(A \wedge B)$ $\neg(A \vee B)$ $A \wedge (A \vee B)$ $A \vee (A \wedge B)$ $\neg(A \Rightarrow B)$ $A \Rightarrow B$	$B \wedge A$ $B \vee A$ $(A \wedge B) \wedge C$ $(A \vee B) \vee C$ $(A \wedge B) \vee (A \wedge C)$ $(A \vee B) \wedge (A \vee C)$ $A$ $A$ $A$ $\neg A \vee \neg B$ $\neg A \wedge \neg B$ $A$ $A$ $\neg A \vee B$ $A \wedge \neg B$ $(A \Rightarrow B) \wedge (B \Rightarrow A)$
	Kommutativ
	Assoziativ
	Distributiv
	Idempotenz
	Involution
	DE-MORGAN
	Absorption
	Elimination

## Axiomatik

**Axiome** als wahr angenommene Aussagen; an Nützlichkeit gemessen.

Anspruch, aber nach GÖDELS Unvollständigkeitssatz nicht möglich:

- Unabhängig
- Vollständig
- Widerspruchsfrei

## Prädikatenlogik

**Quantoren** Innerhalb eines Universums:

**Existenzq.**  $\exists$  „Mind. eines“

**Individuum**  $\exists!$  „Genau eines“

**Allq.**  $\forall$  „Für alle“

**Quantitative Aussagen**

**Erfüllbar**  $\exists x F(x)$

**Widerlegbar**  $\exists x \neg F(x)$

**Tautologie**  $\top = \forall x F(x)$  (alle Schlussregeln)

**Kontradiktion**  $\perp = \forall x \neg F(x)$

Klassische Tautologien	Bezeichnung
$A \vee \neg A$ $A \wedge (A \Rightarrow B) \Rightarrow B$ $(A \wedge B) \Rightarrow A$ $A \Rightarrow (A \vee B)$	Ausgeschlossenes Drittes Modus ponens Abschwächung

**Negation** (DE-MORGAN)

$$\neg \exists x F(x) \Leftrightarrow \forall x \neg F(x)$$

$$\neg \forall x F(x) \Leftrightarrow \exists x \neg F(x)$$

## Häufige Fehler

- $U = \emptyset^c$  nicht notwendig
- $\exists x(P(x) \Rightarrow Q(x)) \not\Leftrightarrow \exists x P(x)$
- $\neg \exists x \exists y P(x, y) \Leftrightarrow \forall x \neg \exists y P(x, y)$

## Beweistechniken

**Achtung:** Aus falschen Aussagen können wahre **und** falsche Aussagen folgen.

**Direkt**  $A \Rightarrow B$  Angenommen  $A$ , zeige  $B$ . Oder: Angenommen  $\neg B$ , zeige  $\neg A$  (**Kontraposition**).

$$(A \Rightarrow B) \Leftrightarrow (\neg B \Rightarrow \neg A)$$

**Fallunters.** Aufteilen, lösen, zusammenführen. O.B.d.A. = „Ohne Beschränkung der Allgemeinheit“

**Widerspruch**  $(\neg A \Rightarrow \perp) \Rightarrow A$   
Angenommen  $A \wedge \neg B$ , zeige Kontradiktion. (Reductio ad absurdum)

**Ring** (Transitivität der Implikation)

$$A \Leftrightarrow B \Leftrightarrow C \Leftrightarrow \dots$$
$$\Leftrightarrow A \Rightarrow B \Rightarrow C \Rightarrow \dots \Rightarrow A$$

**Induktion**  $F(n) \quad \forall n \geq n_0 \in \mathbb{N}$

1. **Anfang:** Zeige  $F(n_0)$ .
2. **Schritt:** Angenommen  $F(n)$  (Hypothese), zeige  $F(n+1)$  (Behauptung).

**Starke Induktion:**  
Angenommen  $F(k) \quad \forall n_0 \leq k \leq n \in \mathbb{N}$ .

## Häufige Fehler

- Nicht voraussetzen, was zu beweisen ist
- Äquiv. von Implikat. unterscheiden (Zweifelsfall immer Implikat.)

## Naive Mengenlehre

**Mengen** Zusammenfassung versch. Objekte „Elemente“.

**Element**  $x \in M$  „enthält“

**Leere M.**  $\emptyset = \{\}$

**Universum**  $U$

**Einschränkung**  $\{x \mid F(x)\}$

## Relationen

**Teilmenge**  $N \subseteq M$   
 $\Leftrightarrow \forall n \in N : n \in M$

**Gleichheit**  $M = N$   
 $\Leftrightarrow M \subseteq N \wedge N \subseteq M$

## Mächtigkeit

$$|M| \begin{cases} = n & \text{endlich} \\ & M \text{ injekt.} \Leftrightarrow M \text{ surj.} \\ \geq \infty & \text{unendlich} \end{cases}$$
$$= |N| \Leftrightarrow \exists f_{\text{bijekt.}} : M \rightarrow N$$

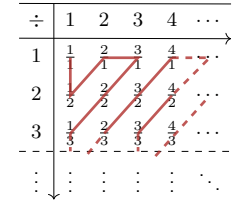
**Kardinalität** ÄK. für Gleichmächtigkeit

$$|M| \leq |N| \Leftrightarrow \exists f_{\text{injekt.}} : M \rightarrow N$$

- $M \subseteq N \Rightarrow |M| \leq |N|$
- $|M| \leq |N| \Leftrightarrow \exists f_{\text{surj.}} : N \rightarrow M$  (AC)

**Abzählbar**  $|M| \leq |\mathbb{N}|$

- Endliche Mengen,  $\emptyset, \mathbb{N}, \mathbb{Z}, \mathbb{Q}$
- $M_{\text{abz.}} \wedge N_{\text{abz.}} \Rightarrow (M \cup N)_{\text{abz.}}$   
( $= \{m_1, n_1, m_2, n_2, \dots\}$ )
- $M_{\text{abz.}} \wedge N \subseteq M \Rightarrow N_{\text{abz.}}$



$$f(1) = 0, r_{11} r_{12} r_{13} r_{14} \dots$$
$$f(2) = 0, r_{21} r_{22} r_{23} r_{24} \dots$$
$$f(3) = 0, r_{31} r_{32} r_{33} r_{34} \dots$$
$$f(4) = 0, r_{41} r_{42} r_{43} r_{44} \dots$$

(CANTORS Diagonalargumente)

## Operationen

**Vereinig.**  $M \cup N$   
 $\Leftrightarrow \{x \mid x \in M \vee x \in N\}$

**Schnitt**  $M \cap N \Leftrightarrow \{x \mid x \in M \wedge x \in N\}$  ( $= \emptyset$  „disjunkt“)

**Diff.**  $M \setminus N \Leftrightarrow \{x \mid x \in M \wedge x \notin N\}$

**Komplement**  $M^c \{x \mid x \notin M\}$

Alle logischen Äquivalenzen gelten auch für die Mengenoperationen.

## Häufige Fehler

- $\forall M : \emptyset \subseteq M$ , nicht  $\forall M : \emptyset \in M$

## Quantitative Relationen

Sei Indexmenge  $I$  und Mengen  $M_i \quad \forall i \in I$ .

$$\bigcup_{i \in I} M_i := \{x \mid \exists i \in I : x \in M_i\}$$

$$\bigcap_{i \in I} M_i := \{x \mid \forall i \in I : x \in M_i\}$$

## Neutrale Elemente

- $\bigcup_{i \in \emptyset} M_i = \emptyset$  („hinzufügen“)
- $\bigcap_{i \in \emptyset} M_i = U$  („wegnehmen“)

## Potenzmenge

$$\mathcal{P}(M) := \{N \mid N \subseteq M\}$$

**Satz von Cantor**  $|M| < |\mathcal{P}(M)|$

$$|\mathcal{P}(M)| = 2^{|M|} \quad (\in / \notin \text{ binär})$$

- Menge der Kardinalitäten  $\mathcal{K}$  ist unendlich

**Satz von Hartogs (AC)**  $(\mathcal{K}, \preceq)$  ist total geordnet

$$\aleph_0 \preceq \aleph_1 \preceq \aleph_2$$

$$|(0, 1)| = |\mathbb{R}| = |\mathcal{P}(\mathbb{N})|$$

## Kontinuumshypothese

$$\nexists M : |\mathbb{N}| < |M| < |\mathcal{P}(\mathbb{N})| = |\mathbb{R}|$$

## Auswahlaxiom (AC)

Für Menge  $\mathcal{X}$  nicht-leerer Mengen:

$$\exists c : \mathcal{X} \rightarrow \bigcup \mathcal{X} \\ \forall X \in \mathcal{X} : c(X) \in X$$

Nutzung kennzeichnen!

- unabh. vom ZFC

## Relationen

### Kartesisches Produkt

$$X_1 \times \cdots \times X_n := \{(x_1, \dots, x_n) \mid x_1 \in X_1, \dots, x_n \in X_n\}$$

**Relation**  $\sim$  von/auf  $M$  nach  $N$  ist Teilmenge  $R \subseteq M \times N$ . ( $R' \subseteq N \times P$ )

$$m \sim n \Leftrightarrow (m, n) \in R$$

$\equiv$  **Reflexiv**  $\forall x \in M : (x, x) \in R$   
 $\Leftrightarrow \text{id}_M \subseteq R$

**Irreflexiv**  $\forall x \in M : (x, x) \notin R$   
 $\Leftrightarrow \text{id}_M \cap R = \emptyset$

$\equiv$  **Sym.**  $\forall (x, y) \in R : (y, x) \in R$   
 $\Leftrightarrow R \subseteq R^{-1}$

$\preceq$  **Antis.**  $\forall x, y : ((x, y) \in R \wedge (y, x) \in R) \Rightarrow x = y$   
 $\Leftrightarrow R \cap R^{-1} \subseteq \text{id}_M$

$\equiv$  **Transitiv**  $\forall x, y, z : ((x, y) \in R \wedge (y, z) \in R) \Rightarrow (x, z) \in R$   
 $\Leftrightarrow R; R \subseteq R$

**Vollst.**  $\forall x, y \in M : (x, y) \in R \vee (y, x) \in R$   
 $\Leftrightarrow R \cup R^{-1} = M \times M$

## Spezielle Relationen

**Inverse Relation**  $R^{-1}$  mit  $R \in M \times N :=$   
 $\{(n, m) \in N \times M \mid (m, n) \in R\}$

**Komposition**  $R; R'$  mit  $R' \in N \times P :=$   
 $\{(m, p) \in M \times P \mid \exists n \in N : (m, n) \in R \wedge (n, p) \in R'\}$

**Leere Relation**  $\emptyset$

**Identität**  $\text{id}_M := \{(m, m) \mid m \in M\}$   
 $(=)$

**Allrelation**  $M \times M$

**Äquivalenzrelation**  $\equiv$  reflexiv, symmetrisch und transitiv. (Gleichheit\*\*\*)

**Äquivalenzklasse**  $[m]_{\equiv}$  auf  $M$ , Vertreter  $m \in M$ .

$$[m]_{\equiv} := \{x \in M \mid m \equiv x\} \\ \Leftrightarrow [m]_{\equiv} = [x]_{\equiv}$$

**Zerlegung**  $\mathcal{N} \subseteq \mathcal{P}(M)$  von  $M$ .

- $\emptyset \notin \mathcal{N}$
- $M = \bigcup \mathcal{N}$

- $N \cap N' = \emptyset$   
 $(N, N' \in \mathcal{N} : N \neq N')$
- (Korrespondiert zur ÄR.)

**Quotient**  $(M / \equiv)$  Sei  $\equiv$  ÄR. auf  $M$ .  
 (ist Zerlegung)

$$(M / \equiv) := \{[m]_{\equiv} \mid m \in M\}$$

(Korrespondiert zur ÄK.)

**Ordnungsrelation**  $\preceq$  reflexiv, antisymmetrisch, transitiv

**Minimale**  $x \forall m \in M \setminus \{x\} : m \not\preceq x$

**Untere Schranken**  $m \in \downarrow X$   
 $\forall x \in X : m \preceq x$

- $\downarrow / \uparrow \emptyset = M$

**Kleinstes**  $\min_{\preceq} X \in X$

**Infimum**  $\max \downarrow X$

- $\inf \{x, y\} = x \wedge y$
- $\sup \{x, y\} = x \vee y$

**Totale Ordnung** + vollständig (Trichotomie)

## Abbildungen

**Abbildung**  $f$  von  $X$  (Definitions b.) nach  $Y$  (Werteb.) ordnet jedem  $x \in X$  eindeutig ein  $y \in Y$  zu.

**Totalität**  $\forall x \in X \exists y \in Y : f(x) = y$

**Eindeutigkeit**  $\forall x \in X \forall a, b \in Y : f(x) = a \wedge f(x) = b \Rightarrow a = b$

$$f : X \rightarrow Y$$

**Bilder**  $f(X') = \{f(x) \mid x \in X'\}$   $X' \subseteq X$

**Urbilder**  $f^{-1}(Y') = \{x \in X \mid f(x) \in Y'\}$   $Y' \subseteq Y$

**Graph**  $\text{gr}(f) := \{(x, f(x)) \mid x \in X\}$

**Identität**

$$\text{id}_A : A \rightarrow A$$

$$\text{id}_A(a) := a \quad \forall a \in A$$

**Umkehrfunktion**  $f^{-1} : Y \rightarrow X$  wenn  $f$  bijektiv und  $(f \circ f^{-1})(y) = y$  bzw.  $f; f^{-1} = \text{id}_X \wedge f^{-1}; f = \text{id}_X$   
 Für die Relation  $f^{-1}$  gilt:

- $x \in f^{-1}(\{f(x)\})$
- $f(f^{-1}(\{y\})) = \{y\}$  falls  $f$  surjektiv

## Eigenschaften

**Injektiv**  $\forall x_1, x_2 \in X : x_1 \neq x_2 \Leftrightarrow f(x_1) \neq f(x_2)$

**Surjektiv**  $\forall y \in Y \exists x \in X : y = f(x)$

**Bijektiv/Invertierbar** wenn injektiv und surjektiv

**Cantor-Schröder-Bernstein**

$$\left. \begin{array}{l} f : M \rightarrow N \\ g : N \rightarrow M \end{array} \right\} \text{injekt.} \\ \Rightarrow \exists B_{\text{bijekt.}} : M \rightarrow N$$

**Fixpunkt**  $f(m) = m$   
 Sei  $X \subseteq Y \subseteq M, f : M \rightarrow N$

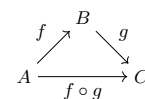
- $f(X) \subseteq f(Y)$  (Monotonie)
- $M \setminus Y \subseteq M \setminus X$
- $M \setminus (M \setminus X) = X$

**Knaster-Tarski-Lemma** Sei  $X \subseteq Y \subseteq M \Rightarrow f(X) \subseteq f(Y)$  (monoton), dann hat  $f : \mathcal{P}(M) \rightarrow \mathcal{P}(M)$  einen Fixpunkt

**Verkettung**  $f \circ g : A \rightarrow C$

$$(f \circ g)(a) = f(g(a))$$

(der Reihenfolge nach)



## Verbände

Sei  $(M, \preceq)$  teilweise geordnet

$$\forall m, n \in M \exists^{\text{inf}} / \sup \{m, n\}$$

**Vollständig**  $\forall X \subseteq M : \exists^{\text{inf}} / \sup X$

$$\bullet \exists^{\text{min}} / \max M = \sup / \inf \emptyset$$

## Distributivität

$$\forall x, y, z \in M : \\ x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) \\ x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

- Jede total geordnete Menge ist distributiv

# Analysis

## Reelle Zahlen $\mathbb{R}$

## Angeordnete Körper

(Gilt auch für  $\mathbb{Z}$  und  $\mathbb{Q}$ )

**Körperaxiome**  $(\mathbb{R}, +, *)$   $a, b, c \in \mathbb{R}$

**Addition**  $(\mathbb{R}, +)$

**Assoziativität**  
 $a + (b + c) = (a + b) + c$

**Kommutativität**  
 $a + b = b + a$

**Neutrales Element Null**  
 $a + 0 = a \quad 0 \in \mathbb{R}$

**Inverses „Negativ“**  
 $a + (-a) = 0 \quad (-a) \in \mathbb{R}$

**Multiplikation**  $(\mathbb{R}, *)$

**Assoziativität**  $a * (b * c) = (a * b) * c$

**Kommutativität**  $a * b = b * a$

**Neutrales Element Eins**  
 $a * 1 = a \quad 1 \in \mathbb{R} \setminus \{0\}$

**Inverses „Kehrwert“**  
 $a * (a^{-1}) = 1$   
 $a \neq 0, (a^{-1}) \in \mathbb{R}$

## Distributivität

$$a * (b + c) = a * b + a * c$$

Totale Ordnung

Transitivität  
 $a < b \wedge b < c \Rightarrow a < c$

Trichotomie Entweder  
 $a < b$  oder  $a = b$  oder  $b < a$   
 $\Rightarrow$  **Irreflexivität** ( $a < b \Rightarrow a \neq b$ )

Addition  
 $a < b \Rightarrow a + c < b + c$

Multiplikation  
 $a < b \Rightarrow a * c < b * c \quad 0 < c$

Bei Additiver oder Multiplikativer Inversion dreht sich die Ungleichung.

Archimedes Axiom

$$\forall x \in \mathbb{R} \exists n \in \mathbb{N} : n > x$$
$$n > \frac{1}{x}$$

Teilbarkeit

$$a|b \Leftrightarrow \exists n \in \mathbb{Z} : b = a * n$$

( $\Rightarrow \sqrt{2} \notin \mathbb{Q}$ , da mit  $\frac{a}{b} = \sqrt{2}$  nicht teilerfremd)

Häufige Fehler

- Nicht durch Null teilen/kürzen
- Nicht  $-x < 0$  annehmen
- Multiplikation mit negativen Zahlen kehrt Ungleichungen

Operationen

Brüche

- $\frac{a}{b} * \frac{c}{d} = \frac{ac}{bd}$
- $\frac{a}{b} \stackrel{*d}{=} \frac{ad}{bd}$
- $\frac{a}{c} + \frac{b}{c} = \frac{a+b}{c}$
- $\frac{a}{b} + \frac{c}{d} = \frac{ad+cb}{bd}$

Wurzeln  $b^n = a \Leftrightarrow b = \sqrt[n]{a}$

- $\sqrt[n]{a * b} = \sqrt[n]{a} * \sqrt[n]{b}$
- $\sqrt[n]{\sqrt[n]{a}} = \sqrt[n * n]{a}$
- $\sqrt[n]{a} < \sqrt[n]{b} \quad 0 \leq a < b$
- $\sqrt[n+1]{a} < \sqrt[n]{a} \quad 1 < a$
- $\sqrt[n]{a} < \sqrt[n+1]{b} \quad 0 < a < 1$

$$\sqrt[n]{a^n} = |a| \quad a \in \mathbb{R}$$

Potenzen  $a^{\frac{x}{y}} = \sqrt[y]{a^x}$

- $a^x * b^x = (a * b)^x$
- $a^x * a^y = a^{x+y}$
- $(a^x)^y = a^{x*y}$

Dezimaldarstellung

Gauss-Klammer  $[y] := \max\{k \in \mathbb{Z} \mid k \leq y\} = \lfloor y \rfloor$

$$[y] = k \Leftrightarrow k \leq y < k + 1$$

Existenz  $\forall x \geq 0 \exists! (a_n)_{n \in \mathbb{N}}$  mit

- $a_n \in \{0, \dots, 9\} \quad \forall n \in \mathbb{N}$
- $\sum_{i=0}^n \frac{a_i}{10^i} \leq x < \sum_{i=0}^{n+1} \frac{a_i}{10^i} + \frac{1}{10^{n+1}} \quad \forall n \in \mathbb{N}_0$

Die Umkehrung gilt mit Lemma:

$$x = \sum_{n=0}^{\infty} \frac{a_n}{10^n}$$

Lemma  $x \geq 0, (a_n)_{n \in \mathbb{N}}$  Dezi. von  $x$

$$\neg(\exists N \in \mathbb{N} \forall n \geq N : a_n = 9)$$

$$x \in \mathbb{Q} \Leftrightarrow (a_n)_{n \in \mathbb{N}} \text{ periodisch}$$

Intervalle

Sei  $A \subseteq \mathbb{R}, A \neq \emptyset, a_0 \in A$ .

Geschlossen  $[a; b] := \{x \in \mathbb{R} \mid a \leq x \leq b\}$   
(„Ecken sind mit enthalten“)

Offen  $(a; b) := \{x \in \mathbb{R} \mid a < x < b\}$   
(Bei  $\infty$  immer offen, da  $\infty \notin \mathbb{R}$ )

Kleinstes/Größtes Element

Minimum  $\min(A) := a_0$   
 $\Leftrightarrow \forall a \in A : a_0 \leq a$

Maximum  $\max(A) := a_0$   
 $\Leftrightarrow \forall a \in A : a \leq a_0$   
( $\frac{1}{2} \min / \max(a; b)$ )

Beschränktheit  $A$  heißt

Oben beschränkt  $\exists s \in \mathbb{R} \forall a \in A : a \leq s$

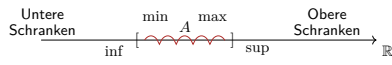
Unten beschränkt  $\exists s \in \mathbb{R} \forall a \in A : s \leq a$

Vollständigkeit

Infimum (klein)  $\inf(A)$   
 $:= \max\{s \in \mathbb{R} \mid \forall a \in A : s \leq a\}$

Supremum (groß)  $\sup(A)$   
 $:= \min\{s \in \mathbb{R} \mid \forall a \in A : a \leq s\}$

Vollständigkeitsaxiom  $\exists \sup(A)$ .



Folgen

Folge  $(a_n)_{n \in \mathbb{N}}$  in  $A$  ist eine Abb.  $f : \mathbb{N} \rightarrow A$  mit  $a_n = f(n)$ .

Arithmetische Folge  $a_{n+1} = a_n + d$   
 $a_n = a + (n - 1) * d \quad d, a \in \mathbb{R}$

Geometrische Folge  $a_{n+1} = a_n * q$   
 $a_n = q^n \quad q \in \mathbb{R}$

Rekursion  $a_n$  ist auf  $a_{n-1}$  definiert.

$$a_{n+1} = F(n, a_n) \quad \forall n \in \mathbb{N}$$
$$F : A \times \mathbb{N} \rightarrow A$$

Primfaktorzerlegung  $n \in \mathbb{N}, n \geq 2$

$$\exists p_1, \dots, p_n \in \mathbb{P} : n = p_1 * \dots * p_n$$

Summen und Produkte

Summe  $\sum_{i=1}^n i = 1 + 2 + \dots + n$

Produkt  $\prod_{i=1}^n i = 1 * 2 * 3 * \dots * n$

Fakultät  $n! = \prod_{i=1}^n i$  (**0! = 1**)

Gaussche Summe  $n \in \mathbb{N}$

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

Geom. Summe  $q \in \mathbb{R} \setminus \{0\}, n \in \mathbb{N}_0$

$$\sum_{i=0}^n q^i = \frac{1 - q^{n+1}}{1 - q}$$

Bernoulli Unglei.  $n \in \mathbb{N}_0, x \geq -1$

$$(1 + x)^n \geq 1 + nx$$

Binom. Koeff.  $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

- Rechnen:  $\frac{n > k}{0 < (n-k)}$
- $\binom{n}{0} = \binom{n}{n} = 1$
- $\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$

Binomischer Satz  $n \in \mathbb{N}$

$$(a + b)^n = \sum_{k=0}^n \binom{n}{k} * a^{n-k} b^k$$

Grenzwerte

Betrag  $|x| := \begin{cases} x & 0 \leq x \\ -x & x < 0 \end{cases}$

Lemma  $|x * y| = |x| * |y|$

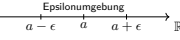
Dreiecksungleichung  $|x + y| \leq |x| + |y|$

Umgekehrte Dreiecksungleichung  
 $||x| - |y|| \leq |x - y|$

Konvergenz

Sei  $(a_n)_{n \in \mathbb{N}} \subseteq \mathbb{R}, a \in \mathbb{R}$ .

$$a_n \xrightarrow{n \rightarrow \infty} a \Leftrightarrow \forall \epsilon > 0 \exists n_0 \in \mathbb{N} \forall n \in \mathbb{N} n \geq n_0 : |a_n - a| \leq \epsilon$$
$$(a - \epsilon \leq a_n \leq a + \epsilon)$$



$$\bullet a_n \xrightarrow{n \rightarrow \infty} a \Leftrightarrow \lim_{n \rightarrow \infty} a_n = a$$

Beschränkt + monoton  $\Rightarrow$  konvergent:

$$\lim_{n \rightarrow \infty} a_n = \begin{cases} \inf\{a_n \mid n \in \mathbb{N}\} & (a_n)_{\text{fall.}} \\ \sup\{a_n \mid n \in \mathbb{N}\} & (a_n)_{\text{steig.}} \end{cases}$$

Nullfolgen  $\lim_{n \rightarrow \infty} a_n = 0$

- $\lim_{n \rightarrow \infty} \frac{1}{n^k} = 0 \quad k \in \mathbb{N}$
- $\lim_{n \rightarrow \infty} nq^n = 0$

Folgen gegen 1

- $\lim_{n \rightarrow \infty} \sqrt[n]{a} = 1 \quad a > 0$
- $\lim_{n \rightarrow \infty} \sqrt[n]{n} = 1$

Bestimmt Divergent

$$a_n \xrightarrow{n \rightarrow \infty} \infty \Leftrightarrow \forall R > 0 \exists n \geq n_0 \in \mathbb{N} : a_n \geq R$$
$$a_n \xrightarrow{n \rightarrow \infty} -\infty \Leftrightarrow \forall R < 0 \exists n \geq n_0 \in \mathbb{N} : a_n \leq R$$

$$\lim_{n \rightarrow \infty} q^n \begin{cases} = 0 & (-1; 1) \\ = 1 & = 1 \\ \geq \infty & > 1 \\ \text{div.} & \leq -1 \end{cases}$$

Monotonie

Monoton fallend

$$a_n \geq a_{n+1} \quad \forall n \in \mathbb{N}$$

(streng)

Monoton steigend

$$a_n \leq a_{n+1} \quad \forall n \in \mathbb{N}$$

(streng)

## Beschränktheit

$$\exists k > 0 \forall n \in \mathbb{N} : |a_n| \leq k$$

- Konvergent  $\Rightarrow$  beschränkt
- Unbeschränkt  $\Rightarrow$  divergent

## Grenzwertsätze

$$\lim_{n \rightarrow \infty} a_n = a, \lim_{n \rightarrow \infty} b_n = b$$

$$a_n \xrightarrow{n \rightarrow \infty} a \wedge a_n \xrightarrow{n \rightarrow \infty} b \Rightarrow a = b \text{ (Max. einen Grenzw.)}$$

$$a = 0 \wedge (b_n)_{\text{beschr.}} \Leftrightarrow \lim_{n \rightarrow \infty} a_n b_n = 0$$

$$a_n \leq b_n \Leftrightarrow a \leq b \text{ (nicht <)}$$

$$\lim_{n \rightarrow \infty} \begin{cases} a_n \pm b_n = a \pm b \\ a_n * b_n = a * b \\ a_n * c = a * c \\ \sqrt[k]{a_n} = \sqrt[k]{a} \\ |a_n| = |a| \end{cases}$$

## Einschachtelungssatz

$$\lim_{n \rightarrow \infty} a_n = \lim_{n \rightarrow \infty} b_n = a$$
$$\forall n \geq N \in \mathbb{N} : a_n \leq c_n \leq b_n$$
$$(\exists) \lim_{n \rightarrow \infty} c_n = a$$

## Spezielle Folgen

**Teilfolge** *streng mnt.* Folge  $(b_k)_{k \in \mathbb{N}}$  mit  $(n_k)_{k \in \mathbb{N}}$ , sodass  $b_k = a_{n_k} \quad \forall k \in \mathbb{N}$ .

$$\lim_{n \rightarrow \infty} a_n = a \Rightarrow \lim_{n \rightarrow \infty} a_{n_k} = a$$

(da  $n_k$  mnt. steigend)

$$\forall (a_n)_{n \in \mathbb{N}} \exists (a_{n_k})_{k \in \mathbb{N}} \text{ mnt.}$$

(nicht streng!)

**Häufungspunkt**  $h$  mit einer Teilfolge

$$\lim_{n \rightarrow \infty} a_{n_k} = h$$

$$\lim_{n \rightarrow \infty} a_n = a \Leftrightarrow \exists ! : h = a$$

## Bolzano-Weierstraß

$$(a_n)_{n \in \mathbb{N}}_{\text{beschr.}} \Rightarrow \exists h_{\text{Häuf.}}$$

(Beschränkte Teilfolgen besitzen mind. einen Häufungspunkt)

## Cauchy-Folge

$$\forall \epsilon > 0 \exists n_0 \in \mathbb{N} \forall n, m \geq n_0 : |a_n - a_m| \leq \epsilon$$

(Konv. ohne bekannten Grenzwert)

## Vollständigkeit von $\mathbb{R}$

$$(a_n)_{n \in \mathbb{N}}_{\text{CAUCHY}} \Leftrightarrow \exists \lim_{n \rightarrow \infty} a_n$$

$$(\exists \lim_{n \rightarrow \infty} a_n \Rightarrow (a_n)_{n \in \mathbb{N}}_{\text{CAUCHY}})$$
$$\Rightarrow (a_n)_{n \in \mathbb{N}}_{\text{beschr.}}$$
$$\Rightarrow \exists h \text{ (BW)}$$
$$\Rightarrow \lim_{n \rightarrow \infty} a_n = h$$

## Stetigkeit

**Berührungspunkt**  $D \subseteq \mathbb{R}, a \in \mathbb{R}$

$a$  BP. von  $D$

$$\Leftrightarrow \exists (x_n)_{n \in \mathbb{N}} \text{ in } D : x_n \xrightarrow{n \rightarrow \infty} a$$
$$\Leftrightarrow \forall \delta > 0 \exists x \in D : |x - a| \leq \delta$$

**Grenzwert gegen Stelle**  $f : D \rightarrow \mathbb{R}, y \in \mathbb{R}, a$  BP. von  $D$

$$\lim_{x \rightarrow a} f(x) = y$$
$$\Leftrightarrow \forall (x_n)_{n \in \mathbb{N}} \text{ in } D :$$
$$x_n \xrightarrow{n \rightarrow \infty} a \Rightarrow f(x_n) \xrightarrow{n \rightarrow \infty} y$$
$$\Leftrightarrow \forall \epsilon > 0 \exists \delta > 0 \forall x \in D :$$
$$|x - a| \leq \delta \Rightarrow |f(x) - y| \leq \epsilon$$

(Grenzwertsätze gelten analog)

## Stetig an Stelle $f$ stetig bei $a$

$$\lim_{x \rightarrow a} f(x) = f(a)$$
$$\Leftrightarrow \forall (x_n)_{n \in \mathbb{N}} \text{ in } D :$$
$$x_n \xrightarrow{n \rightarrow \infty} a \Rightarrow f(x_n) \xrightarrow{n \rightarrow \infty} f(a)$$
$$\Leftrightarrow \forall \epsilon > 0 \exists \delta > 0 \forall x \in D :$$
$$|x - a| \leq \delta \Rightarrow |f(x) - f(a)| \leq \epsilon$$

(U.A. stetig: Summen, Produkte, Quotienten, Verkettungen stetiger Fkt. und Polynome)

**Einseitiger Grenzwert**  $x_0 < / > a \in D$

$$\lim_{x \nearrow / \searrow a} f(x) = y$$
$$\Leftrightarrow \forall (x_n)_{n \in \mathbb{N}} \text{ in } D :$$
$$(x_n \xrightarrow{n \rightarrow \infty} a \wedge \forall n : x_n < / > a)$$
$$\Rightarrow f(x_n) \xrightarrow{n \rightarrow \infty} y$$
$$\Leftrightarrow \lim_{x \rightarrow a} f(x) = y \wedge x_0 < / > a \in D$$

**Grenzwert gegen  $\infty$**   $D$  unbeschränkt

$$\lim_{x \rightarrow \infty} f(x) = y$$
$$\Leftrightarrow \forall (x_n)_{n \in \mathbb{N}} \text{ in } D :$$
$$x_n \xrightarrow{n \rightarrow \infty} \infty \Rightarrow f(x_n) \xrightarrow{n \rightarrow \infty} y$$
$$\Leftrightarrow \forall \epsilon > 0 \exists x_0 \in \mathbb{R} \forall x \in D :$$
$$x \geq x_0 \Rightarrow |f(x) - y| \leq \epsilon$$

**Grenzwert**  $= \infty$

$$\lim_{x \rightarrow a} f(x) = \infty$$
$$\Leftrightarrow \forall (x_n)_{n \in \mathbb{N}} \text{ in } :$$
$$x_n \xrightarrow{n \rightarrow \infty} a \Rightarrow f(x_n) \xrightarrow{n \rightarrow \infty} \infty$$
$$\Leftrightarrow \forall R > 0 \exists \delta > 0 \forall x \in D :$$
$$|x - a| \leq \delta \Rightarrow f(x) \geq R$$

## Eigenschaften stetiger Funktionen

**Lemma**  $f(a) > \eta \Rightarrow \forall x \exists \delta > 0 \in D \cap [a - \delta, a + \delta] : f(x) > \eta$

**Zwischenwert**  $[a; b] \subseteq \mathbb{R}, f : [a; b] \rightarrow \mathbb{R}$  stetig,  $f(a) \neq f(b)$

## Konvergenzkriterien

### Cauchy

$$\Leftrightarrow (\sum_{k=1}^n a_k)_{n \in \mathbb{N}}_{\text{CAUCHY}}$$
$$(\sum_{k=1}^{\infty} a_k)_{\text{konv.}}$$
$$\Leftrightarrow \forall \epsilon > 0 \exists n_0 \in \mathbb{N} \forall n > m > n_0 :$$
$$|\sum_{k=m+1}^n a_k| \leq \epsilon$$

### Notwendig

$$(\sum_{n=1}^{\infty} a_n)_{\text{konv.}} \Rightarrow \lim_{n \rightarrow \infty} a_n = 0$$
$$\lim_{n \rightarrow \infty} a_n \neq 0 \Rightarrow (\sum_{n=1}^{\infty} a_n)_{\text{div.}}$$

**Beschränkt**  $a_n \geq 0 (\Rightarrow \text{mnt.}) \quad \forall n \in \mathbb{N}$

$$(\sum_{n=1}^{\infty} a_n)_{\text{beschr.}} \Leftrightarrow (\sum_{n=1}^{\infty} a_n)_{\text{konv.}}$$

**Majorante**  $0 \leq a_n \leq b_k \quad \forall n \in \mathbb{N}$

$$(\sum_{n=1}^{\infty} b_n)_{\text{konv.}} \Leftrightarrow (\sum_{n=1}^{\infty} a_n)_{\text{konv.}}$$

**Quotient**  $a_n \geq 0 \quad \forall n \in \mathbb{N}$

$$\lim_{n \rightarrow \infty} \frac{a_{n+1}}{a_n} \begin{cases} < 1 \rightarrow (\sum_{n=1}^{\infty} a_n)_{\text{konv.}} \\ > 1 \rightarrow (\sum_{n=1}^{\infty} a_n)_{\text{div.}} \end{cases}$$

**Wurzel**  $a_n \geq 0 \quad \forall n \in \mathbb{N}$

$$\lim_{n \rightarrow \infty} \sqrt[n]{a_n} \begin{cases} < 1 \rightarrow (\sum_{n=1}^{\infty} a_n)_{\text{konv.}} \\ > 1 \rightarrow (\sum_{n=1}^{\infty} a_n)_{\text{div.}} \end{cases}$$

### Absolut

$$(\sum_{n=1}^{\infty} |a_n|)_{\text{konv.}} \Rightarrow (\sum_{n=1}^{\infty} a_n)_{\text{konv.}}$$

$$|\sum_{n=1}^{\infty} a_n| \leq \sum_{n=1}^{\infty} |a_n|$$

(Dreiecksungleichung)

**Korollar**  $f(a) * f(b) < 0 \Rightarrow \exists \xi \in (a; b) : f(\xi) = 0$  (versch. Vorzeichen)

### Satz

$$f : [a; b] \rightarrow \mathbb{R} \text{ stetig}$$
$$\Rightarrow f \text{ beschränkt}$$
$$\Rightarrow \exists^{\min / \max} \{f(x) \mid x \in [a; b]\}$$

**Satz** Sei  $I$  Intervall,  $I, J \subseteq \mathbb{R}, f : I \rightarrow J$  stetig, strg. mnt ( $\Rightarrow$  injektiv), surjektiv

$$\Rightarrow J \text{ Intervall}$$
$$\Rightarrow f \text{ bijektiv}$$
$$\Rightarrow f^{-1} : J \rightarrow I \text{ stetig}$$

## Reihen

**Reihe**  $(s_n)_{n \in \mathbb{N}} = \sum_{k=1}^{\infty} a_k$  mit den Gliedern  $(a_k)_{k \in \mathbb{N}}$ .

**nte Partialsumme**  $s_n = \sum_{k=1}^n a_k$

**Grenzwert** ebenfalls  $\sum_{k=1}^{\infty} a_k$ , falls  $s_n$  konvergiert

### Spezielle Reihen

**Geom.**  $\sum_{k=0}^{\infty} q^k = \frac{1}{1-q} \quad q \in (-1; 1)$

**Harmon.**  $\sum_{k=1}^{\infty} \frac{1}{k}$  divergent

**Allg. Harmon.**  $\sum_{k=1}^{\infty} \frac{1}{k^\alpha}$  konvergiert  $\forall \alpha > 1$

### Lemma

- $\sum_{k=1}^{\infty} a_k, \sum_{k=1}^{\infty} b_k$  konvergent
- $-\sum_{k=1}^{\infty} a_k + \sum_{k=1}^{\infty} b_k = \sum_{k=1}^{\infty} (a_k + b_k)$
- $-c * \sum_{k=1}^{\infty} a_k = \sum_{k=1}^{\infty} c * a_k$
- $\exists N \in \mathbb{N} : (\sum_{k=N}^{\infty} a_k)_{\text{konv.}} \Rightarrow (\sum_{k=1}^{\infty} a_k)_{\text{konv.}}$  (Es reicht spätere Glieder zu betrachten)
- $(\sum_{k=1}^{\infty} a_k)_{\text{konv.}} \Rightarrow \forall N \in \mathbb{N} : (\sum_{k=N}^{\infty} a_k)_{\text{konv.}} \Rightarrow \lim_{N \rightarrow \infty} \sum_{k=N}^{\infty} a_k = 0$

Leibniz  $(a_n)_{n \in \mathbb{N}}$  mnt. Nullfolge

$$\left(\sum_{n=1}^{\infty} (-1)^n * a_n\right)_{\text{konv.}}$$

Grenzwert  $a_n, b_n \geq 0 \quad \forall n \in \mathbb{N}$

$$\lim_{n \rightarrow \infty} \frac{a_n}{b_n} > 0 \Rightarrow \left(\sum_{n=1}^{\infty} a_n\right)_{\text{konv.}} \Leftrightarrow \left(\sum_{n=1}^{\infty} b_n\right)_{\text{konv.}}$$

Exponentialfunktion

$$\exp(x) := \sum_{n=0}^{\infty} \frac{x^n}{n!} = e^x$$

- $\exp(0) = 1$
- $\exp(1) = e \approx 2,71828 \notin \mathbb{Q}$   
 $e = \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n$

$$\exp(x) * \exp(y) = \exp(x + y)$$

Cauchy-Produkt

$$\left(\sum_{n=0}^{\infty} a_n\right) \left(\sum_{n=0}^{\infty} b_n\right) = \sum_{n=0}^{\infty} \sum_{k=0}^n a_k b_{n-k}$$

Korollar

- $\exp(x) > 0$
- $\frac{1}{\exp(x)} = \exp(-x)$
- $x < y \Rightarrow \exp(x) < \exp(y)$
- $\exp(r * x) = (\exp(x))^r$
- $\exp(r) = e^r$

$$\exp_a(x) := \exp(x * \log a) = a^x$$

- $a > 1 \Rightarrow \text{strng. mnt. steigend}$
- $0 < a < 1 \Rightarrow \text{strng. mnt. fallend}$
- $0 < a \neq 1 \Rightarrow \exp_a : \mathbb{R} \rightarrow \mathbb{R}^+$  bijektiv

Logarithmen

$$\log = \exp^{-1} : \mathbb{R}^+ \rightarrow \mathbb{R}$$

- $\log 1/x = -\log x$
- $\log x/y = \log x - \log y$
- $\log x^r = r * \log x$

$$\log(x * y) = \log x + \log y$$

$$\log_a x = \frac{\log x}{\log a} = \exp_a^{-1}$$

Trigonometrische Funktionen

$$\sin x := \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)!}$$

$$\cos x := \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!}$$

(beide absolut konvergent,  $0^0 := 1$ )

- $|\sin / \cos x| \leq 1$
- $\sin -x = -\sin x$
- $\cos -x = \cos x$
- $\sin(x + y) = \sin(x) \cos(y) + \cos(x) \sin(y)$
- $\cos(x + y) = \cos(x) \cos(y) - \sin(x) \sin(y)$
- $\sin 2x = 2 \sin(x) \cos(x)$
- $\cos 2x = \cos^2 x - \sin^2 x$
- $\sin^2 x + \cos^2 x = 1$
- $\sin x - \sin y = 2 \cos\left(\frac{x+y}{2}\right) \sin\left(\frac{x-y}{2}\right)$
- $\cos x - \cos y = 2 \sin\left(\frac{x+y}{2}\right) \sin\left(\frac{y-x}{2}\right)$

$$\pi : \cos \frac{\pi}{2} = 0$$

- $\sin / \cos(x + 2\pi) = \sin / \cos x$
- $\sin / \cos(x + \pi) = -\sin / \cos x$

- $\sin / \cos\left(x + \frac{\pi}{2}\right) = \cos / \sin x$
- $\sin x = 0 \quad \forall k \in \mathbb{Z} : x = k\pi$
- $\cos x = 0 \quad \forall k \in \mathbb{Z} : x = (2k + 1) * \frac{\pi}{2}$

$$\tan x := \frac{\sin x}{\cos x}$$

Differenzierbarkeit

$D \subseteq \mathbb{R}, f : D \rightarrow \mathbb{R}, a \in D$  BP von  $D \setminus \{a\}$

Differenzierbar an der Stelle  $a$ , falls

$$\lim_{x \rightarrow a} \frac{f(x) - f(a)}{x - a} =: f'(x) = \lim_{h \rightarrow 0} \frac{f(a + h) - f(a)}{h}$$

- Differenzierbar bei  $a \Rightarrow$  stetig bei  $a$

Summenregel  $(f + g)'(a) = f'(a) + g'(a)$

Faktorregel  $(c * f)'(a) = c * f'(a)$

Produktregel  $(f * g)'(a) = f'(a) * g(a) + f(a) * g'(a)$

= Reziprokregel  $(1/f)'(a) = -\frac{g'(a)}{g^2(a)}$

= Quotientenregel  $(f/g)'(a) = \frac{f'(a) * g(a) - f(a) * g'(a)}{g^2(a)}$

Kettenregel  $(f \circ g)'(a) = f'(g(a)) * g'(a)$

Umkehrfunktion  $(f^{-1})'(b) = 1/f'(f^{-1}(b))$

$f'$	$f$	$F$
0	$a$	$ax + c$
1	$x$	$\frac{1}{2}x^2 + c$
$-1/x^2$	$1/x$	$\ln(x) + c$
$\frac{1}{2\sqrt{x}}$	$\sqrt{x}$	$\frac{2}{3}x\sqrt{x} + c$
$ax^a - 1$	$x^a$	$\frac{1}{a+1}x^{a+1} + c$
$\cos x$	$\sin x$	$-\cos(x) + c$
$-\sin x$	$\cos x$	$\sin(x) + c$
$e^x$	$e^x$	$e^x$
$a^x \ln a$	$a^x$	
$\frac{1}{x \ln a}$	$\log_a x$	

Sei  $f, g : [a, b] \rightarrow \mathbb{R}$  diffbar und stetig:

Satz von Rolle

$$f(a) = f(b) \Rightarrow \exists \xi \in (a, b) : f'(\xi) = 0$$

Mittelwertsatz

$$\exists \xi \in (a, b) : f'(\xi) = \frac{f(b) - f(a)}{b - a}$$

$$\exists \xi \in (a, b) :$$

$$f'(\xi)(g(b) - g(a)) = g'(\xi)(f(b) - f(a))$$

Monotonie

- $(\forall x \in D : f(x) \leq 0) \Rightarrow f$  mnt. fallend
- $(\forall x \in D : f(x) < 0) \Rightarrow f$  strng. mnt. fallend
- $f$  (nicht streng) mnt. fallend  $\Rightarrow \forall x \in D : f'(x) \leq 0$

Höhere Ableitungen

$$n\text{-mal ableitbar } \exists f', f'', \dots, f^{(n)}$$

Stetig ableitbar Ableitung stetig

Extrema

Lokales Extrema

$$\exists \epsilon > 0 \forall x \in D \cap (x_0 - \epsilon, x_0 + \epsilon) : f(x_0) \leq / \geq f(x)$$

Ist  $D$  Intervall und  $x_0$  innerer Punkt und lokales Extremum:

$$\Rightarrow f'(x_0) = 0$$

(Achtung: Umkehrung nicht notwendig!)

Sei zusätzlich  $f(x_0) = 0$  und  $f$  2 mal ableitbar:

- $f''(x_0) < 0 \Rightarrow x_0$  lokales Maximum
- $f''(x_0) > 0 \Rightarrow x_0$  lokales Minimum

Algorithmen auf Datenstrukturen

Algorithmus Handlungsvorschrift aus endlich vielen Einzelschritten zur Problemlösung.

- Korrektheit (Test-based dev.)
- Terminierung (TURING)
- Effizienz (Komplexität)

Formen (High to low) Menschl. Sprache, Pseudocode, Mathematische Ausdrücke, Quellcode, Binärcode

Divide & Conquer

Divide Zerlegen in kleinere Teilprobleme

Conquer Lösen der Teilprobleme mit gleicher Methode (rekursiv)

Merge Zusammenführen der Teillösungen



Effizienz

Raum/Zeit-Tradeoff: Zwischenspeichern vs. Neuberechnen

Programmlaufzeit/-allokationen	Komplexität
Einfluss äußerer Faktoren	Unabh.
Konkrete Größe	Asymptotische Schätzung

Inputgröße n Jeweils

- Best-case C<sub>B</sub>
- Average-case
- Worst-case C<sub>W</sub>

Asymptotische /Speicherkomplexität

Groß-O-Notation Kosten C<sub>f</sub>(n) mit g : N → R ∃ c > 0 ∃ n<sub>0</sub> > 0 ∀ n ≥ n<sub>0</sub>

Untere Schranke Ω(f)  
C<sub>f</sub>(n) ≥ c \* g(n)

Obere Schranke O(f)  
C<sub>f</sub>(n) ≤ c \* g(n)

Exakte Schranke Θ(f)  
C<sub>f</sub>(n) ∈ Ω(f) ∩ O(f)  
Polynom kten Grades ∈ Θ(n<sup>k</sup>)

(Beweis: g und c finden)

Groß-O	Wachstum	Klasse	
O(1)	Konstant		lösbar
O(log n)	Logarithmisch		
O(n)	Linear		
O(n log n)	Nlogn		
O(n <sup>2</sup> )	Quadratisch	Polynomiell O(n <sup>k</sup> )	
O(n <sup>3</sup> )	Kubisch		hart
O(2 <sup>n</sup> )	Exponentiell	Exponentiell O(α <sup>n</sup> )	
O(n!)	Fakultät		
O(n <sup>n</sup> )			

Rechenregeln

Elementare Operationen, Kontrollstr. ∈ O(1)

Schleifen ∈ i Wiederholungen \* O(f) teuerste Operation

Abfolge O(g) nach O(f) ∈ O(max(f; g))

Rekursion ∈ k Aufrufe \* O(f) teuerste Operation

Mastertheorem a ≥ 1, b > 1, Θ ≥ 0

T(n) = a \* T(n/b) + Θ(n<sup>k</sup>)  
⇒ { Θ(n<sup>k</sup>) a < b<sup>k</sup>  
Θ(n<sup>k</sup> log n) a = b<sup>k</sup>  
Θ(n<sup>log<sub>b</sub> a</sup>) a > b<sup>k</sup>

Floor/Ceiling Runden

Floor [x] nach unten

Ceiling [x] nach oben

Suchverfahren

Lineare Liste endlich, geordnete (nicht sortierte) Folge n Elemente L := [a<sub>0</sub>, ..., a<sub>n</sub>] gleichen Typs.

Array Sequenzielle Abfolge im Speicher, statisch, Index O(1), schnelle Suchverfahren [L[0] | ... | L[n - 1]]

Sequenziell C<sub>A</sub>(n) = 1/n \* Σ<sup>n</sup> i = (n+1)/2 ∈ O(n)

Algorithm: Sequential Search  
Input: Liste L, Predikat x  
Output: Index i von x  
for i ← 0 to L.len - 1 do  
  if x = L[i] then  
    return i  
end  
return -1

Auswahlproblem Finde i-kleinstes Element in unsortierter Liste ∈ Θ(n)

Algorithm: i-Smallest Element  
Input: Unsortierte Liste L, Level i  
Output: Kleinstes Element x  
p ← L[L.len - 1]  
for k = 0 to L.len - 1 do  
  if L[k] < p then  
    Push(L <, L[k])  
  if L[k] > p then  
    Push(L >, L[k])  
end  
if L <.len = i - 1 then  
  return p  
if L <.len > i - 1 then  
  return i-Smallest Element L <  
if L <.len < i - 1 then  
  return i-Smallest Element (L >,  
  i - 1 - L <.len)  
end

Sortierte Listen

Binär C<sub>W</sub>(n) = [log<sub>2</sub> n] + 1, C<sub>A</sub>(n) <sup>n → ∞</sup> ≈ log<sub>2</sub> n ∈ O(log n)

Algorithm: Binary Search  
Input: Sortierte Liste L, Predikat x  
Output: Index i von x  
if L.len = 0 then  
  return -1  
else  
  m ← [L.len / 2]  
  if x = L[m] then  
    return m  
  if x < L[m] then  
    return Binary Search [L[0], ..., L[m - 1]]  
  if x > L[m] then  
    return m + 1 + Binary Search [L[m + 1], ..., L[L.len - 1]]  
end

Sprung Kosten Vergleich a, Sprung b mit optimaler Sprungweite:

m = [√(a/b) \* n]

C<sub>A</sub>(n) = 1/2 \* ([n/m] \* a + mb) ∈ O(√n)

Algorithm: Jump Search  
Input: Sortierte Liste L, Predikat x  
Output: Index i von x  
m ← [√n]  
while i < L.len do  
  i ← i + m  
  if x < L[i] then  
    return Search [L[i - m], ..., L[i - 1]]  
  end  
end  
return -1

- k-Ebenen Sprungsuche ∈ O(√n)
- Partitionierung in Blöcke m möglich

Exponentiell ∈ O(log x)

Algorithm: Exponential Search  
Input: Sortierte Liste L, Predikat x  
Output: Index i von x  
while x > L[i] do  
  i ← 2 \* i  
end  
return Search [L[i/2], ..., L[i - 1]]

- Unbekanntes n möglich

Interpolation C<sub>A</sub>(n) = 1 + log<sub>2</sub> log<sub>2</sub> n, C<sub>W</sub>(n) ∈ O(n)

Algorithm: Searchposition  
Input: Listengrenzen [u, v]  
Output: Suchposition p  
return [u + (x - L[u]) / (L[v] - L[u]) \* (v - u)]

Algorithm: Interpolation Search  
Input: Sortierte Liste [L[u], ..., L[v]], Predikat x  
Output: Index i von x  
if x < L[u] ∨ x > L[v] then  
  return -1  
p ← Searchposition(u, v)  
if x = L[p] then  
  return p  
if x > L[p] then  
  return Interpolation Search(p + 1, v, x)  
else  
  return Interpolation Search(u, p - 1, x)  
end

Häufigkeitsordnungen mit Zugriffswahrscheinlichkeit p<sub>i</sub>: C<sub>A</sub>(n) = Σ<sub>i=0</sub><sup>n</sup> i p<sub>i</sub>

Frequency-count Zugriffszähler pro Element

Transpose Tausch mit Vorgänger

Move-to-front

Verkettete Listen

Container Jedes Element p ist in der Form p → (key | value | next). Index ist seq. Suche ∈ O(n)

Löschen ∈ O(1)

Algorithm: Delete  
Input: Zeiger p auf Vorgänger des löschenden Elements  
if p ≠ ∅ ∧ p → next ≠ ∅ then  
  p → next ← (p → next) → next  
end

- desh. sehr dynamisch

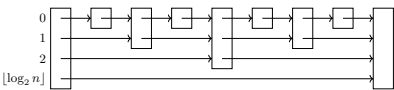
Suchen C<sub>A</sub>(n) = (n+1)/2 ∈ O(n)

Algorithm: Search Linked List  
Input: Verkettete Liste L, Predikat x  
Output: Zeiger p auf x  
p ← L.head while p → value ≠ x do  
  p ← p → next  
end  
return p

Doppelt Verkettet Zeiger auf Vorgänger (key | value | prev | next)

- Bestimmung des Vorgängers (bei Einfügen, Löschen) ∈ O(1) statt O(n)
- Höherer Speicheraufwand

Skip



- Zeiger auf Ebene i zeigt zu nächstem 2<sup>i</sup> Element

- Suchen ∈ O(log n)

(Perfekt) Einfügen, Löschen ∈ O(n) (Vollst. Reorga.)

Randomisiert Höhe zufällig (keine vollst. Reorga.)  
P(h) = 1/2<sup>h+1</sup>: Einfügen, Löschen ∈ O(log n)

Spezielle Listen

ADT „Abstrakte Datentypen“

Stack S = |TOP, ... Operationen nur auf letztem Element ∈ O(1)

Queue Q = |HEAD, ... , TAIL Vorne Löschen, hinten einfügen ∈ O(1)

Priority Queue P = [p<sub>0</sub> p<sub>1</sub> ... p<sub>n</sub> / a<sub>0</sub> a<sub>1</sub> ... a<sub>n</sub>]  
Jedes Element a hat Priorität p; Entfernen von Element mit höchster (MIN) Priorität

Sortierverfahren

Sortierproblem

Gegeben (endliche) Folge von Schlüsseln (von Daten) (K<sub>i</sub>)<sub>i ∈ I</sub>

Gesucht Bijektive Abbildung π : I → I (Permutation), sodass K<sub>π(i)</sub> ≤ K<sub>π(i+1)</sub> ∀ i ∈ I

mit Optimierung nach geringen

- Schlüsselvergleichen C
- Satzbewegungen M

## Eigenschaften

**Ordnung** *Allgemein* vs. *speziell*: Ordnung wird nur über Schlüsselvergleiche hergestellt

**Relation** *Stabil* vs. *instabil*: Vorherige relative Reihenfolge bleibt erhalten

**Speicher** *In situ* vs. *ex situ*: Zusätzlicher Speicher notwendig

**Lokal** *Intern* vs. *extern*: Alles im RAM oder Mischung vorsortierter externer Teilfolgen

**Ordnung**  $\forall x, y \in X$

**Reflexiv**  $x \leq x$

**Antisym.**  $x \leq y \wedge y \leq x \Rightarrow x = y$

**Transitiv**  $x \leq y \wedge y \leq z \Rightarrow x \leq z$

**Total (Vollständig)**  $x \leq y \vee y \leq x$

(ohne Total: „*Halbordnung*“)

## Grad der Sortierung

**Anzahl der Inversionen** Anzahl kleinerer Nachfolger für jedes Element:

$$\begin{aligned} \text{inv}(L) &:= |\{(i, j) \mid \\ 0 \leq i < j \leq n-1, \\ L[i] \geq L[j]\}| \end{aligned}$$

**Anzahl der Runs** Ein *Run* ist eine sortierte Teilliste, die nicht nach links oder rechts verlängert werden kann. Die Anzahl der Runs ist:

$$\begin{aligned} \text{runs}(L) &:= |\{i \mid \\ 0 \leq i < n-1, \\ L[i+1] < L[i]\}| + 1 \end{aligned}$$

**Längster Run** Anzahl der Elemente der längsten sortierten Teilliste:

$$\begin{aligned} \text{las}(L) &:= \max\{r.\text{len} \mid \\ r \text{ ist Run in } L\} \\ \text{rem}(L) &:= L.\text{len} - \text{las}(L) \end{aligned}$$

## Einfache Sortierverfahren $O(n^2)$

**Selection** Entferne kleinstes Element in unsortierter Liste und füge es sortierter Liste an.

```
Algorithm: Selectionsort
Input: Liste L
Output: Sortierte Liste L
for i ← 0 to L.len - 2 do
  min ← i
  for j ← i + 1 to L.len - 1 do
    if L[j] < L[min] then
      min ← j
  end
  if min ≠ i then
    Swap L[min], L[i]
end
if L.len = 0 then
  return -1
```

**Insertion** Verschiebe erstes Element aus unsortierter Liste von hinten durch sortierte Liste, bis das vorgehende Element kleiner ist.

```
Algorithm: Insertionsort
Input: Liste L
Output: Sortierte Liste L
for i ← 1 to L.len - 1 do
  if L[i] < L[i - 1] then
    temp ← L[i]
    j ← i
    while temp < L[j - 1] ∧ j > 0 do
      L[j] ← L[j - 1]
      j ← j - 1
    end
    L[j] ← temp
end
```

**Bubble** Vertausche benachbarte Elemente, durchlaufe bis nichts vertauscht werden muss. *Achtung*: Die hinteren Elemente können im Durchlauf ignoriert werden!

```
Algorithm: Bubblesort
Input: Liste L
Output: Sortierte Liste L
i ← L.len
swapped ← 1
while swapped do
  swapped ← 0
  for j ← 0 to i - 2 do
    if L[j] > L[j + 1] then
      Swap L[j], L[j + 1]
      swapped ← 1
    end
  end
  i ← i - 1
end
```

## Verbesserte Sortierverfahren $O(n \log n)$

**Shell** Insertionsort, nur werden Elemente nicht mit Nachbarn getauscht, sondern in  $t$  Sprüngen  $h_i$ , die kleiner werden (Kamm). Im letzten Schritt dann Insertionsort ( $h_t = 1$ ); somit Sortierung von grob bis fein, also Reduzierung der Tauschvorgänge.

```
Algorithm: Shellsort
Input: Liste L, Absteigende Liste von Sprunggrößen H
Output: Sortierte Liste L
foreach h in H do
  for i ← h to L.len - 1 do
    temp ← L[i]
    for j ← i; temp < L[j - h] ∧ j ≥ h;
      j ← j - h do
      L[j] ← L[j - h]
    end
    L[j] ← temp
  end
end
```

**Quick** Rekursiv: Pivot-Element in der Mitte, Teillisten  $L_{<}$ ,  $L_{>}$ , sodass  $\forall l_{<} \in L_{<} \forall l_{>} \in L_{>} : l_{<} < x < l_{>}$ . Zerlegung: Durchlauf von Links bis  $L[i] \geq x$  und von Rechts bis  $L[j] \leq x$ , dann tauschen.

```
Algorithm: Quicksort
Input: Liste L, Indices l, r
Output: L, sortiert zwischen l und r
if l ≥ r then
  return
i ← l
j ← r
piv ← L[(l+r)/2]
do
  while L[i] < piv do
    i ← i + 1
  end
  while L[j] > piv do
    j ← j - 1
  end
  if i ≤ j then
    Swap L[i], L[j]
    i ← i + 1
    j ← j - 1
  end
while i ≤ j;
Quicksort(L, l, j)
Quicksort(L, i, r)
```

**Turnier** Liste also Binärbaum, bestimme  $\min(L)$  durch Austragen des Turniers, entferne Sieger und wiederhole von Siegerpfad aus.

**Heap** Stelle Max-Heap (größtes Element in der Wurzel) her, gib Wurzel aus und ersetze mit Element ganz rechts in unterster Ebene.

```
Algorithm: Max-Heapify
Input: Liste L, Index i der MHE widerspricht und
       $\forall j > i$  erfüllen MHE
Output: Liste L mit MHE  $\forall j \geq i$ 
l ← 2i + 1
r ← 2i + 2
if l < L.len ∧ L[l] > L[i] then
  largest ← l
else
  largest ← i
end
if r < L.len ∧ L[r] > L[largest] then
  largest ← r
end
if largest ≠ i then
  Swap L[i], L[largest]
  Max-Heapify(L, largest)
end
```

```
Algorithm: Build-Max-Heap
Input: Liste L
Output: Liste L mit MHE
for i ← ⌊L.len/2⌋ - 1 to 0 do
  Max-Heapify(L, i)
end
```

```
Algorithm: Heapsort
Input: Liste L
Output: Sortierte Liste L
Build-Max-Heap L
for i ← L.len - 1 to 1 do
  Swap L[0], L[i]
  L.len ← L.len - 1
  Max-Heapify L, 0
end
```

**Merge** Zerlege Liste in  $k$  Teile, sortiere diese (mit Mergesort) und verschmelze die sortierten Teillisten (merge).

```
Algorithm: 2-Merge
Input: Liste L mit  $L[l \dots m - 1]$  und  $L[m \dots r]$ 
      sortiert, Indices l, m, r
Output: Liste L mit  $L[l \dots r]$  sortiert
j ← l
k ← m
for i ← 0 to r - l do
  if k > r ∨ (j < m ∧ L[j] ≤ L[k]) then
    B[i] ← L[j]
    j ← j + 1
  else
    B[i] ← L[k]
    k ← k + 1
  end
end
for i ← 0 to r - l do
  L[l + i] ← B[i]
end
```

```
Algorithm: Rekursives 2-Mergesort
Input: Liste L, Indices l, r
Output: Liste L mit  $L[l \dots r]$  sortiert
if l ≥ r then
  return
else
  m ← ⌊(l+r+1)/2⌋
  Mergesort(L, l, m - 1)
  Mergesort(L, m, r)
  Merge(L, l, m, r)
end
```

## Iteratives 2-Mergesort

```
Algorithm: Iteratives 2-Mergesort
Input: Liste L
Output: Sortierte Liste L
for k ← 2; k < n; k ← k * 2 do
  for i ← 0; i + k ≤ n; i ← i + k do
    Merge(L, i, min(i + k - 1, n - 1),
          i + k/2)
  end
end
Merge(L, 0, n - 1, k/2)
```

**Natürliches Mergesort** Verschmelzen von benachbarten Runs (Ausnutzen der Vorsortierung)

## Untere Schranke allgemeiner Sortierverfahren

Jedes allgemeine Sortierverfahren benötigt im Worst- und Average-case Schlüsselvergleiche von mindestens:

$$\Omega(n \log n)$$

(Siehe Pfadlänge auf Entscheidungsbaum)

## Spezielle Sortierverfahren $O(n)$

**Distribution** Abspeichern der Frequenz jedes Elementes  $k$  auf  $F[k]$ ; Ausgeben jedes Index  $F[k]$  mal.

**Lexikographische Ordnung**  $\leq$  Sei  $A = \{a_1, \dots, a_n\}$  ein Alphabet, dass sich mit gegebener Ordnung  $a_1 < \dots < a_n$  wie folgt auf dem Lexikon  $A^* = \bigcup_{n \in \mathbb{N}_0} A^n$  fortsetzt:

$$\begin{aligned} v &= (v_1, \dots, v_p) \leq w = (w_1, \dots, w_q) \\ \Leftrightarrow \forall 1 \leq i \leq p: v_i &= w_i \quad p \leq q \\ \vee \forall 1 \leq j \leq i: v_j &= w_j \quad v_i < w_i \end{aligned}$$

**Fachverteilen** Sortieren von  $n$   $k$ -Tupeln in  $k$  Schritten: Sortieren nach letztem Element, vorletztem usw.

## Große Datensätze sortieren

**Indirekt** Liste von Zeigern  $Z[i] = i$  auf die eigentlichen Listenelemente. Schlüsselvergleiche mit  $L[Z[i]]$ , Satzbewegungen nur als Zeigertausch in  $Z$ . Anschließend linear kopieren.

**Extern** Zerlegen in  $m$  Blöcke, sortieren im Hauptspeicher (Run) der mind.  $m+1$  Blöcke groß ist, verschmelzen der Runs ( $m$ -Wege-Merge).

## Ausgeglichenes 2-Wege-Mergesort

Daten auf Band  $n$ , sortieren von Block  $r_1 < n$  auf zweites Band und  $r_2$  auf drittes Band, löschen des ersten Bandes und Merge  $2r$  abwechselnd auf erstes (neues  $2r_1$ ) und viertes Band (neues  $2r_2$ ) und wiederholen.

**Replacement Selectionsort** Lese  $r < n$  Elemente auf Priority-Queue  $Q$ . Falls  $x = \min(Q) \geq$  letztem Element auf zweiten Band, schreibe  $x$  aus, sonst schreibe  $Q$  auf Band. Wiederhole auf dritten Band und dann merge.

Algo.	Stabil	Mem.	Schlüsselvergleiche				Satzbewegungen			
			$C_L$	$C_R$	$M_L$	$M_R$	$C_L$	$C_R$	$M_L$	$M_R$
Selection	✓	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Insertion	✓	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Bubble	✓	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Shell	✓	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Quick	✓	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Insertion	✓	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Heap	✓	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Merge	✓	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Untere Schranke ( $\Omega(n \log n)$ ) für allgemeine Sortierverfahren										
Distribution	✓	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$



## Bäume

- Verallg. von Listen: Element/Knoten kann mehrere Nachfolger haben

- Darstellung von Hierarchien

**Ungerichteter Graph**  $(V, E)$  mit einer Menge Knoten  $V$  und Kanten  $E \subseteq V \times V$

**Baum** Ungerichteter Graph mit

**Einfach** keine Schleife  oder Doppelkanten 


**Zusammenhängend** Für jede zwei Knoten gibt es genau eine Folge von Kanten die sie verbindet

**Azyklisch** kein Zyklus (Cycle) 

**Wurzelbaum** Baum mit genau einem Knoten der Wurzel heißt

**Orientierter Wurzelbaum** Alle Knoten sind Wurzel ihrer disjunkten Unterbäume und haben verschiedene Werte gleichen Typs. (Im Nachfolgenden einfach nur „Baum“)

**Darstellungsarten**

**Graph** 

**Array**  $[a, b, c, \emptyset, d, e]$

**Menge**  $\{\{a, b, c, d, e\}, \{b\}, \{c, d, e\}, \{d\}, \{\emptyset\}\}$

**Klammer**  $(a, (b), (c, (d), (e)))$

**Größen**

**Ordnung** Max. Anzahl von Kindern jedes Knoten eines Baums

**Tiefe** Anzahl Kanten zwischen einem Knoten und Wurzel

**Stufe** Alle Knoten gleicher Tiefe

**Höhe** Max. Tiefe +1

**Eigenschaften**

**Geordnet** Kinder erfüllen Ordnung von links nach rechts

**Vollständig** Alle Blätter auf gleicher Stufe, jede Stufe hat max. Anzahl von Kindern

**Binärbäume**

Geordneter, orientierter Wurzelbaum der Ordnung 2.

**Strikt** Jeder Knoten hat 0 oder 2 Kinder (Kein Knoten hat genau 1 Kind).

**Vollständig** Jeder Knoten außer der letzten Stufe hat genau 2 Kinder.

**Fast Vollständig** Vollständig, außer Blätter können rechts fehlen.

**Ausgeglichen** Vollständig, aber Blätter auf letzten 2 Stufen

2 Binärbäume heißen

**Ähnlich** selbe Struktur

**Äquivalent** Ähnlich und selbe Knoten

**Größen**

- Für  $i$  Stufen max.  $2^i$  Knoten
- Für  $n$  Knoten genau  $n - 1$  Kanten
- Vollständiger B. mit  $n$  Knoten hat Höhe von  $\log_2 n + 1$

**Speicherung**

**Verkettet**

Zeiger Links   Knoten   Zeiger Rechts
---------------------------------------

**Feldbaum** Sequenz von

Knoten   Index Links   Index Rechts
-------------------------------------

**Sequenziell** Lesen vollst. Baum links nach rechts, oben nach unten, leere Elemente für fehlende Knoten (ineffizient für degenerierte Bäume)

**Traversierung**

- $W$  Verarbeite Wurzel
- $L$  Durchlaufe linken Unterbaum
- $R$  Durchlaufe rechten Unterbaum

Konvention erst links, dann rechts:

- $WLR$  Preorder
- $LWR$  Inorder
- $LRW$  Postorder

Implementation rekursiv oder linear mit eigenem Stack (effizienter)

**Gefädelte Binärbäume**

Zeiger „Faden“ in Knoten zeigt auf nächsten Knoten nach Durchlaufordnung.

Nachteil: Zusätzlicher Speicheraufwand teilweise redundant; Lösung: Nur Null-Zeiger (Blätter) sind Fäden

**rFaden** zeigt auf Nachfolgerknoten

**lFaden** zeigt auf Vorgängerknoten

**Binäre Suchbäume**

**Natürliche binäre Suchbäume**

$$B_l < B_x < B_r$$

**Suchen** rekursiv oder mit Durchlaufalg.  $\in O(\ln n)$

**Einfügen** dort wo Suche terminiert

**Löschen** mit zwei nicht-leeren Unterbäumen: Hochziehen des größten Wertes im linken oder kleinsten Wert im rechten Unterbaum (Alt: Als gelöscht markieren)

**Balancierte Binärbäume**

Grundoperationen auf ausgeglichene Binärbäume kosten am wenigsten. Herstellung der Ausgeglichenheit in  $O(n)$

**Balancefaktor** von Knoten  $x$  ist  $BF(x) := h(B_l(x)) - h(B_r(x))$

**k-Balanciert**  $\forall x \in B : |BF(x)| \leq k$

**AVL-Baum** 1-balancierter Binärer Suchbaum

Herstellung der Ausgeglichenheit durch Rotationen

•  $BF(u) = -2, BF(v) \in \{0, -1\}$ : Einfachrotation **Links(u)**

•  $BF(u) = +2, BF(v) \in \{0, -1\}$ : Einfachrotation **Rechts(u)**

•  $BF(u) = -2, BF(v) = +1$ : Doppelrotation **Rechts(v) + Links(u)**

•  $BF(u) = +2, BF(v) = -1$ : Doppelrotation **Links(v) + Rechts(u)**

Für jeden AVL-Baum  $T$  der Höhe  $h$  gilt:

•  $|T| \geq F_h$  (Fibonacci)

•  $h \leq \frac{\log_2(n\sqrt{5}+1)}{\log_2(\frac{1+\sqrt{5}}{2})}$

**Fibonacci-Bäume**  $B_0$  ist leerer Baum,  $B_1$  ist einzelner Knoten,  $B_h = \text{BUILD}(B_{h-1}, x, B_{h-2})$  für  $h \geq 2$  (Maximal unbalancierter AVL-Baum der Höhe  $h$ )

**Gewichtsbalancierte Binärbäume**

**Wurzelbalance**  $\rho(B) = \frac{n_l+1}{n_r+1}$  mit  $n$  Knoten und  $n_l$  Knoten im linken Unterbaum

**Gewichtsbalanciert (BB)**  $\forall$  Unterbaum  $B' : \alpha \leq \rho(B') \leq 1 - \alpha$

- $\alpha = 1/2$ : Vollst. Binärbaum
- $\alpha < 1/2$ : Zunehmend weniger ausgeglichen
- $\alpha = 0$ : Keine Einschränkung

**Mehrwegbäume**

Breiter Baum als Indexstruktur für große externe Daten („Seiten“)

**m-Wege-Suchbäume**

- $m$ -ter Ordnung (max.  $m$  Kinder)
- Knoten mit max.  $b \leq m - 1$  sortierten Einträgen: 

$P_0   K_1   P_1   \dots   K_b   P_b$
---------------------------------------
- Werte im Unterbaum:  $K_i < B_{P_i} < K_{i+1}$

**B-Bäume** der Klasse  $t$  ist (fast-ausgeglichener)  $2t$ -Wege-Suchbaum

- Blätter der Wurzel gleich weit entfernt
- Alle Knoten außer Wurzel min.  $t - 1$ , max.  $2t - 1$  Werte und min.  $t$ , max.  $2t$  Kinder (außer Blätter)
- Wurzel min. 1, max.  $2t - 1$  Werte (oder B. leer) und min. 2, max.  $2t$  Kinder (oder Blatt)

Für  $n$  Knoten ist Höhe  $h \leq 1 + \log_t \frac{n+1}{2}$

**Suchen** Finde größten Index im Knoten  $x \leq K_i$ , suche in  $P_i$

**Einfügen** Teilen voller  $(2t - 1)$  Knoten bei Suche, einfügen im Blatt

**Teilen** (Elternknoten ist nicht voll, da vorher geteilt) Mittlerer Wert in Elternknoten, Werte links davon in linken Unterbaum

**Löschen** Verschieben o. Verschmelzen zu kleiner  $(t - 1)$  Knoten bei Suche, dann entfernen

**Verschieben** Kleinster Wert (ganz vorne) im rechten Unterbaum in Knoten ziehen, Knoten in linken Unterbaum rechts anfügen (und umgekehrt, je nach dem welcher Baum größer ist)

**Verschmelzen** Beide Bäume zu klein, also  $t - 1$  zu einem Unterbaum zusammenfügen ( $2t - 2$ )

**B\*-Bäume** B-Baum Variante mit Daten in den Blättern, Blätter sequenziell verkettet; Standard in DBS



**Binäre B-Bäume** Alternative zu AVL-Bäumen

Digitale Suchbäume

Blattschlüssel = Zeichenkette/Wort des Pfads von Wurzel zu Blatt  
Für max. Schlüssellänge  $l$  und Schlüsselteilänge  $k$  ist Höhe  $= l/k + 1$

**m-äre Tries** Knoten enthalten (Null-)Zeiger für jeden Teilschlüssel der Länge  $k$  in  $m = |\Sigma|^k$ ; Schlechte Speichernutzung, desh. Kompression des Knoten

PATRICIA-Tree

Präfix-/Radix-Baum

Hashing

Aus Schlüssel  $S$  werden Adressen/Indices  $A$  direkt berechnet,

$$h : S \rightarrow A$$

**Kollision**  $|A| \ll |S| \Rightarrow \neg(h \text{ injekt.})$

**Synonyme**  $h(K_i) = h(K_j)$

**Kollisionsklasse**  $[A]_h = \{K \in S \mid h(K) = A\}$

Hashfunktionen

**Divisionsrest**  $h(K_i) := K_i \bmod q$

- $q$  prim  $\Rightarrow$  keinen Teiler mit  $K$
- Optimal bei äquidistanter Schlüsselverteilung

**Falten** Teilsequenzen des Schlüssels werden addiert (Quersumme) oder XOR-verknüpft (Binär)

**Rand-Falten** Rechte Teilsequenzen werden gespiegelt

**Shift-Falten** Teilsequenzen in Reihenfolge

**Mid-Square-Hash**  $h(K) := K^2[K.\text{len} - t/2, K.\text{len} + t/2]$

**Zufalls-Hash**  $K_i$  ist Saat des Zufalls-generators

**Ziffernanalyse-Hash** Teilsequenz von  $K_i$

Hashtabelle

**Kapazität**  $m$

**Belegte Adressen**  $n_a$

**Belegungsfaktor**  $\beta = n_a/m$  sollte  $< .85$  und somit  $m > n_a$

**Erfolgreiche Suche** in  $S(\beta)$  Schritten

**Erfolglose Suche** in  $U(\beta)$  Schritten

Kollisionsbehandlung

Beim Auftritt einer Kollision  $h(K_q) = h(K_p)$  eines gespeicherten  $K_q$ , welches die Adresse für  $K_p$  besetzt:

**Sondieren** Zusätzliche Klasse Hash-funktionen  $h_i$  nach  $i$ -ter Kollision

**Linear**  $h_i(K_p) = (h_0(K_p) + f(i, h(K_p))) \bmod m$

- $S(\beta) \approx \frac{1}{2}(1 + \frac{1}{1-\beta})$
- $U(\beta) \approx \frac{1}{2}(1 + \frac{1}{(1-\beta)^2})$

**Quadratisch**  $h_i(K_p) = (h_0(K_p) + ai + bi^2) \bmod m$

$$h_i(K_p) = (h_0(K_p) - \lceil i/2 \rceil^2 (-1)^i) \bmod m$$

(Sucht in quadratisch wachsenden Abstand in beide Richtungen zur ursprünglichen Adresse)

- Sondierungsfolge versch. Schlüssel korrelieren nicht (Uniform)
- $S(\beta) \approx -\frac{1}{\beta} \ln(1 - \beta)$
- $U(\beta) \approx \frac{1}{1-\beta}$

**Zufällig** Deterministischer Zufalls-generator generiert Schrittfolge  $z_i$

$$h_i(K_p) = (h_0(K_p) + z_i) \bmod m$$

**Double-Hash** Zweite Hashfunktion  $h'$

$$h_i(K_p) = (h_0(K_p) + ih'(K_p)) \bmod m$$

Platzhalter für gelöschte Schlüssel zur Signalisierung sondierter Adressen

**Verkettung** Synonyme werde in dynamischer externen Struktur (Sekundärbereich) in Einfügereihenfolge linear verkettet

- $S(\beta) \approx 1 + \frac{\beta}{2}$
- $U(\beta) \approx \beta - e^{-\beta}$

Hashing auf Externspeicher

- Adresse bezeichnet Bucket der mehrere Daten in Einfügereihenfolge fässt
- Überlaufsmethode beliebig, aber Vermeidung langer Sondierungs-folgen, häufig separater Überlaufsbereich mit dynamischer Zuordnung der Buckets

Dynamische Hashstrukturen

Nachteile der Hashtabelle

- Statische Allokationen speicherin-effizient
- Re-hashing bei Speichererweite-rung

**Erweiterbares Hashing** Digitalbaumk; Bits des Schlüssels oder Hashs steuern Pfad

**HAMT: Hashed Array Mapped Tries** Viele Nullzeiger werden durch Bitmap-Kompression vermieden: Knoten mit  $n$  Feldern hat  $n$  lange Bitmap: 0 zeigt Nullzeiger an, 1 zeigt belegt durch Zei-ger

Signaturen

Möglichst eindeutiges Merkmal eines Datensatzes

**Rolling-Hash** Signaturhash der mit Hilfe des vorgehenden Fensters (Teilzeichenkette) in konstanter statt linearer Zeit berechnet werden kann

Textsuche

Finden aller Positionen (erste Indice) eines Patterns der Länge  $m$  in einem String der Länge  $n$  durch Vergleich mit allen Fenstern

**Naiv**  $\in O(n * m)$

**Statisch effiziente** Index-Strukturen (z.B Suffix-Baum, Signaturen)  $\in O(m)$

**Patternanalyse** Vorverarbeitung des Patterns  $\in O(n + m)$

**Patternanalyse**  $\in O(n + m)$

Knuth-Morris-Pratt

Nutzung bereits gelesener Informationen bei Mismatch, kein Zurückgehen

**Next-Tabelle**

- Wie lang sind Präfix und Suffix gleich im Pattern vor jedem Buch-stabe?
- $\text{next}[0] = -1$

```
Algorithm: Next-Tabelle
Input: Muster pattern[0 . . . m - 1]
Output: Tabelle next[0 . . . m]
i ← 0
j ← -1
next[i] ← j
while j < m do
  while j ≥ 0 ∧ patter[j] ≠ pattern[i] do
    j ← next[j]
  end
  i ← i + 1
  j ← j + 1
  next[i] ← j
end
```

**Suche**  $\in O(n + m)$  Bei Mismatch oder kompletten Match verschieben des Präfix auf den Suffix (oder bei 0 komplett dahinter)

```
Algorithm: Knuth-Morris-Pratt-Suche
Input: Pattern[0 . . m - 1], String[0 . . . n - 1],
Next-Tabelle
Output: Alle Positionen wo das Pattern im String liegt
i ← 0
j ← 0
while j < n do
  while j ≥ 0 ∧ string[i] ≠ pattern[j] do
    j ← next[j]
  end
  j ← j + 1
  i ← i + 1
  if j = m then
    Print i - m
    j ← next[j]
  end
end
```

Boyer-Moore

**Last-Tabelle**

- Letztes Vorkommen im Pattern für jeden Buchstaben des Alphabets
- 1 falls nicht vorkommen

```
Algorithm: Last-Tabelle
Input: Alphabet Σ
Output: Tabelle next[0 . . . |Σ| - 1]
foreach a ∈ Σ do
  last[a] ← -1
end
for j to m - 1 do
  a ← pattern[j]
  last[a] ← j
end
```

**Suche**

- Vergleiche Patter von Rechts nach Links
- Bei Mismatch verschieben des letzten Pattern-Buchstaben zu String-Buchstaben
- Wenn Patter-Buchstabe nicht vor-handen, dann komplett verschie-ben
- $C_A(n, m) \in O(n/m)$
- $C_W(n, m) \in O(n * m)$

```
Algorithm: Boyer-Moore-Suche
Input: Pattern[0 . . . m - 1], String[0 . . . n - 1],
Last-Tabelle
Output: Position des ersten Vorkommens oder -1
i ← 0
while i ≤ n - m do
  j ← m - 1
  while j ≥ 0 ∧ pattern[j] = string[i + j] do
    j ← j - 1
  end
  if j < 0 then
    return i
  else
    if last[string[i + j]] > j then
      i ← i + 1
    else
      i ← i + j - last[string[i + j]]
    end
  end
end
return -1
```

## Statische Textsuche

- Index im Anhang von Büchern
- Signatur-Dateien

## Approximative Suche

**Hamming-Distanz** Anzahl der Miss-matches zwischen  $s_1$  und  $s_2$

**Editierdistanz** Kosten  $s_1$  zu  $s_2$  editieren (Cut, Paste, Replace)

**$k$ -Mismatch-Suchproblem** Alle Vorkommen eines Muster in einem Text mit einer HAMMING-Distanz  $\leq k$

## Exkurs Lineare Algebra

**Matrixmul.**  $(m \times n)(n \times p) = (m \times p)$

$$(AB)_{ij} = \sum_{k=1}^m a_{ik} b_{kj}$$

(Reihe  $\times$  Spalte)