Logik

Aussagenlogik

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

Theoreme sind wahre Aussagen.

Junktoren

Negation $\neg\mathcal{A}$ „Nicht" (`!`, `~`,)

Konjunkt. $\mathcal{A} \wedge \mathcal{B}$ „und" (`&&`,)

Disjunkt. $\mathcal{A} \vee \mathcal{B}$ „oder" (`||`,)

Implikat. $\mathcal{A} \Rightarrow \mathcal{B}$ „Wenn, dann" / „$\mathcal{B}$" ($\rightarrow$, `if`)

$\mathcal{A} \Rightarrow \mathcal{B}$ „$\mathcal{A}$ hinreichend"

$\mathcal{B} \Rightarrow \mathcal{A}$ „$\mathcal{A}$ notwendig"

Äquiv. $\mathcal{A} \Leftrightarrow \mathcal{B}$ „Genau dann, wenn" ($\leftrightarrow$, $\equiv$, `==`,)

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

$\mathcal{A}$	$\mathcal{B}$	$\neg\mathcal{A}$	$\mathcal{A}\wedge\mathcal{B}$	$\mathcal{A}\vee\mathcal{B}$	$\mathcal{A}\Rightarrow\mathcal{B}$	$\mathcal{A}\Leftrightarrow\mathcal{B}$
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
$A \wedge B$	$B \wedge A$	Kommutativ
$A \vee B$	$B \vee A$	
$A \wedge (B \wedge C)$	$(A \wedge B) \wedge C$	Assoziativ
$A \vee (B \vee C)$	$(A \vee B) \vee C$	
$A \wedge (B \vee C)$	$(A \wedge B) \vee (A \wedge C)$	Distributiv
$A \vee (B \wedge C)$	$(A \vee B) \wedge (A \vee C)$	
$A \wedge A$	A	Idempotenz
$A \vee A$	A	
$\neg\neg A$	A	Involution
$\neg(A \wedge B)$	$\neg A \vee \neg B$	De-Morgan
$\neg(A \vee B)$	$\neg A \wedge \neg B$	
$A \wedge (A \vee B)$	A	Absorption
$A \vee (A \wedge B)$	A	
$A \Rightarrow B$	$\neg A \vee B$	
$\neg(A \Rightarrow B)$	$A \wedge \neg B$	Elimination
$A \Leftrightarrow B$	$(A \Rightarrow B) \wedge (B \Rightarrow A)$	

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"

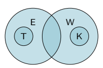
Quantitative Aussagen

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

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

- $U = \emptyset^{\complement}$ 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)

$$A \Leftrightarrow B \Leftrightarrow C \Leftrightarrow \cdots$$
$$\equiv A \Rightarrow B \Rightarrow C \Rightarrow \cdots \Rightarrow A$$

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 $F(k)$ $\forall n_0 \leq k \leq n \in \mathbb{N}$.

Häufige Fehler

- Nicht voraussetzen, was zu beweisen ist
- Äquival. 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

$$|M| \begin{cases} = n & \text{endlich} \\ & M \text{ injekt.} \Leftrightarrow M \text{ surj.} \\ \geq \infty & \text{unendlich} \end{cases}$$

$= |N| \Leftrightarrow \exists f_{\text{bijekt.}} : M \to N$

Kardinalität ÄK. für Gleichmächtigkeit

$$|M| \leq |N| \Leftarrow \exists f_{\text{injekt.}} : M \to N$$

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

Abzählbar $|M| \leq |\mathbb{N}|$

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

$\div$	1	2	3	4	$\cdots$
1	$\frac{1}{1}$	$\frac{2}{1}$	$\frac{3}{1}$	$\frac{4}{1}$	$\cdots$
2	$\frac{1}{2}$	$\frac{2}{2}$	$\frac{3}{2}$	$\frac{4}{2}$	$\cdots$
3	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{3}{3}$	$\frac{4}{3}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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

(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 $M^{\complement}$ $\{x \mid x \notin M\}$

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 $\forall i \in I$.

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

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

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

❧ ❧ ❧

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

Kontinuumshypothese

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

Auswahlaxiom (AC)

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

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

Nutzung kennzeichnen!

- unabh. vom ZFC

Relationen

Kartesisches Produkt

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

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

$\equiv$ **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 $\forall m \in M \backslash \{x\} : m \not\preceq x$

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

$$f : X \to Y$$

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

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

Umkehrfunktion $f^{-1} : Y \to X$ 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

$$\left.\begin{array}{l} f : M \to N \\ g : N \to M \end{array}\right\} \text{injekt.}$$

$$\Rightarrow \exists B_{\text{bijekt.}} : M \to N$$

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

- $f(X) \subseteq f(Y)$ (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 $f : \mathcal{P}(M) \to \mathcal{P}(M)$ einen Fixpunkt

Verkettung $f \circ g : A \to C$

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

(der Reihenfolge nach)

$$\begin{tikzcd} & B \arrow[dr, "g"] & \\ A \arrow[ur, "f"] \arrow[rr, "f \circ g"'] & & C \end{tikzcd}$$

Verbände

Sei $(M, \preceq)$ teilweise geordnet

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

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

- $\exists^{\min}/_{\max} M = {}^{\sup}/_{\inf}\emptyset$

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
$a + 0 = a$ $0 \in \mathbb{R}$

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

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

Assoziativität $a*(b*c) = (a*b)*c$

Kommutativität $a * b = b * a$

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

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

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

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

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 $[y] := \max\{k \in \mathbb{Z} \mid k \le y\} = \lfloor y \rfloor$

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

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

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

Die Umkehrung gilt mit Lemma:

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

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

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

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

Intervalle

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

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

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

Kleinstes/Größtes Element

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

Maximum $\max(A) := a_0$
$\Leftrightarrow \forall a \in A: a \le a_0$

$(\nexists^{\min}/_{\max}(a;b))$

Beschränktheit A heißt

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

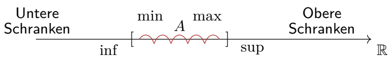
Unten beschränkt $\exists s \in \mathbb{R} \forall a \in A: s \le a$

Vollständigkeit

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

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

Vollständigkeitsaxiom $\exists \sup(A)$.



Folgen

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

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

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

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

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 $n! = \prod^n i \ (0! = 1)$

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

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

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. $n \in \mathbb{N}_0, x \ge -1$

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

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

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

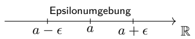
Dreiecksungleichung $|x+y| \le |x| + |y|$

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

Konvergenz

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

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



- $a_n \xrightarrow{n\to\infty} a \Leftrightarrow \lim_{n\to\infty} a_n = a$

Beschränkt + monoton $\Rightarrow$ konvergent:

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

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

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

Folgen gegen 1

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

Bestimmt Divergent

$$a_n \xrightarrow{n\to\infty} \infty \Leftrightarrow$$
$$\forall R > 0 \exists n \ge n_0 \in \mathbb{N}: a_n \ge R$$
$$a_n \xrightarrow{n\to\infty} -\infty \Leftrightarrow$$
$$\forall R < 0 \exists n \ge n_0 \in \mathbb{N}: a_n \le R$$

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

Monotonie

Monoton fallend
$a_n \underset{\text{(streng)}}{\ge} a_{n+1} \quad \forall n \in \mathbb{N}$

Monoton steigend
$a_n \underset{\text{(streng)}}{\le} a_{n+1} \quad \forall n \in \mathbb{N}$

Beschränktheit

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

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

Grenzwertsätze

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

- $a_n \xrightarrow{n\to\infty} a \wedge a_n \xrightarrow{n\to\infty} b$ $\Rightarrow a = b$ (Max. einen Grenzw.)
- $a = 0 \wedge (b_n)_{beschr.}$ $\Leftrightarrow \lim_{n\to\infty} a_n b_n = 0$
- $a_n \leq b_n \Leftrightarrow a \leq b$ (nicht $<$)
- $\lim_{n\to\infty} \begin{cases} a_n \pm b_n = a \pm b \\ a_n \ast b_n = a \ast b \\ a_n \ast c = a \ast c \\ \sqrt[k]{a_n} = \sqrt[k]{a} \\ |a_n| = |a| \end{cases}$

Einschachtelungssatz

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

Spezielle Folgen

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

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

(da n_k mnt. steigend)

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

(nicht streng!)

Häufungspunkt h mit einer Teilfolge

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

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

Bolzano-Weierstraß

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

(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 $\mathbb{R}$

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

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

Stetigkeit

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

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

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

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

(Grenzwertsätze gelten analog)

Stetig an Stelle f stetig bei a

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

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

Einseitiger Grenzwert $x_0 {}^{<}/_{>} a \in D$

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

Grenzwert gegen ∞ D unbeschränkt

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

Grenzwert $= \infty$

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

Eigenschaften stetiger Funktionen

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

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

$$f(a) < c < f(b)$$
$$\Rightarrow \exists \xi \in (a;b) : f(\xi) = c$$

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

Satz

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

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

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

Reihen

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

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

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

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

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

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

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

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

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

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

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)

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

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

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

$$\lim_{n\to\infty}\frac{a_n}{b_n} > 0 \Rightarrow$$

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

Exponentialfunktion

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

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

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

Cauchy-Produkt

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

Korollar

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

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

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

Logarithmen

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

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

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

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

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, $0^0 := 1$)

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

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

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

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

Differenzierbarkeit

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

Differenzierbar an der Stelle a, falls

$$\lim_{x\to a}\frac{f(x)-f(a)}{x-a} =: f'(x)$$

$$= \lim_{h\to 0}\frac{f(a+h)-f(a)}{h}$$

- 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 $f, g : [a,b] \to \mathbb{R}$ 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-mal ableitbar $\exists f', f'', \ldots, f^{(n)}$

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 (Touring)
- 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_B
- Average-case
- Worst-case C_W

Asymptotische Zeit-/Speicherkomplexität

Groß-O-Notation Kosten $C_f(n)$ mit $g : \mathbb{N} \to \mathbb{R} \exists c > 0 \exists n_0 > 0 \forall n \geq n_0$

Untere Schranke $\Omega(f)$
$C_f(n) \geq c * g(n)$

Obere Schranke $O(f)$
$C_f(n) \leq c * g(n)$

Exakte Schranke $\Theta(f)$
$C_f(n) \in \Omega(f) \cap O(f)$
Polynom $k$ten Grades $\in \Theta(n^k)$

(Beweis: g und c finden)

Groß-O	Wachstum	Klasse	
$O(1)$	Konstant		
$O(\log n)$	Logarithmisch		
$O(n)$	Linear		lösbar
$O(n \log n)$	Nlogn		
$O(n^2)$	Quadratisch	Polynomiell $O(n^k)$	
$O(n^3)$	Kubisch		
$O(2^n)$	Exponentiell	Exponentiell $O(\alpha^n)$	
$O(n!)$	Fakultät		hart
$O(n^n)$			

Rechenregeln

Elementare Operationen, Kontrollstr. $\in \mathbf{O(1)}$

Schleifen $\in i$ Wiederholungen $*$ $O(f)$ teuerste Operation

Abfolge $O(g)$ nach $O(f)$ $\in$ $O(\max(f; g))$

Rekursion $\in k$ Aufrufe $*$ $O(f)$ teuerste Operation

Mastertheorem $a \geq 1, b > 1, \Theta \geq 0$

$$T(n) = a * T(\frac{n}{b}) + \Theta(n^k)$$

$$\Rightarrow \begin{cases} \Theta(n^k) & a < b^k \\ \Theta(n^k \log n) & a = b^k \\ \Theta(n^{\log_b a}) & a > b^k \end{cases}$$

Floor/Ceiling Runden

Floor $\lfloor x \rfloor$ nach unten

Ceiling $\lceil x \rceil$ nach oben

Suchverfahren

Lineare Liste endlich, geordnete (nicht sortierte) Folge n Elemente $L := [a_0, \ldots, a_n]$ gleichen Typs.

Array Sequenzielle Abfolge im Speicher, statisch, Index $O(1)$, schnelle Suchverfahren $\boxed{L[0] \mid \cdots \mid L[n-1]}$

Sequenziell $C_A(n) = \frac{1}{n} * \sum^n i = \frac{n+1}{2} \in O(n)$

Algorithm: Sequential Search

Input: Liste L, Predikat x
Output: Index i von x
for $i \leftarrow 0$ **to** $L.len - 1$ **do**
  if $x = L[i]$ **then**
    return i
  end
end
return -1

Auswahlproblem Finde i-kleinstes Element in unsortierter Liste $\in \Theta(n)$

Algorithm: i-Smallest Element

Input: Unsortierte Liste L, Level i
Output: Kleinstes Element x
$p \leftarrow L[L.\text{len} - 1]$
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
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 $C_W(n) = \lfloor \log_2 n \rfloor + 1$, $C_A(n) \overset{n\to\infty}{\approx} \log_2 n \in O(\log n)$

Algorithm: Binary Search

Input: Sortierte Liste L, Predikat x
Output: Index i von x
if $L.len = 0$ **then**
  return -1
else
  $m \leftarrow \lfloor \frac{L.\text{len}}{2} \rfloor$
if $x = L[m]$ **then**
  return m
if $x < L[m]$ **then**
  return Binary Search $[L[0], \ldots, L[m-1]]$
if $x > L[m]$ **then**
  return $m + 1 +$ Binary Search $[L[m+1], \ldots, L[L.len - 1]]$
end

Sprung Kosten Vergleich a, Sprung b mit optimaler Sprungweite:

$$m = \lfloor \sqrt{(\frac{a}{b}) * n} \rfloor$$

$$C_A(n) = \frac{1}{2}(\lceil \frac{n}{m} \rceil * a + mb) \in O(\sqrt{n})$$

Algorithm: Jump Search

Input: Sortierte Liste L, Predikat x
Output: Index i von x
$m \leftarrow \lfloor \sqrt{n} \rfloor$
while $i < L.len$ **do**
  $i \leftarrow i + m$
  if $x < L[i]$ **then**
    return Search $[L[i-m], \ldots, L[i-1]]$
  end
end
return -1

- k-Ebenen Sprungsuche $\in O(\sqrt[k]{n})$
- Partitionierung in Blöcke m möglich

Exponentiell $\in O(\log x)$

Algorithm: Exponential Search

Input: Sortierte Liste L, Predikat x
Output: Index i von x
while $x > L[i]$ **do**
  $i \leftarrow 2 * i$
end
return Search $[L\lfloor i/2 \rfloor, \ldots, L[i-1]]$

- Unbekanntes n möglich

Interpolation $C_A(n) = 1 + \log_2 \log_2 n$, $C_W(n) \in O(n)$

Algorithm: Searchposition

Input: Listengrenzen $[u, v]$
Output: Suchposition p
return $\lfloor u + \frac{x - L[u]}{L[v] - L[u]}(v - u) \rfloor$

Algorithm: Interpolation Search

Input: Sortierte Liste $[L[u], \ldots, L[v]]$, Predikat x
Output: Index i von x
if $x < L[u] \vee x > L[v]$ **then**
  return -1
$p \leftarrow$ 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_i: $C_A(n) = \sum_{i=0}^{n} i p_i$

Frequency-count Zugriffszähler pro Element

Transpose Tausch mit Vorgänger

Move-to-front

Verkettete Listen

Container Jedes Element p ist in der Form $p \to \boxed{\text{(key)} \mid \text{value} \mid \text{next}}$. Index ist seq. Suche $\in O(n)$

Löschen $\in O(1)$

Algorithm: Delete

Input: Zeiger p auf *Vorgänger* des löschendes Elements
if $p \neq \emptyset \wedge p \to next \neq \emptyset$ **then**
  $p \to$ next $\leftarrow (p \to$ next$) \to$ next
end

- desh. sehr dynamisch

Suchen $C_A(n) = \frac{n+1}{2} \in O(n)$

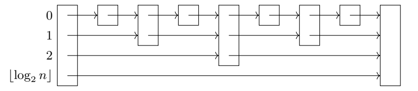
Algorithm: Search Linked List

Input: Verkettete Liste L, Predikat x
Output: Zeiger p auf x
$p \leftarrow L.$head **while** $p \to value \neq x$ **do**
  $p \leftarrow p \to$ next
end
return p

Doppelt Verkettet Zeiger auf Vorgänger $\boxed{\text{(key)} \mid \text{value} \mid \text{prev} \mid \text{next}}$

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

Skip



- Zeiger auf Ebene i zeigt zu nächstem 2^i Element
- Suchen $\in O(\log n)$

(Perfekt) Einfügen, Löschen $\in \mathbf{O(n)}$ (Vollst. Reorga.)

Randomisiert Höhe zufällig (keine vollst. Reorga.)
$P(h) = \frac{1}{2^{h+1}}$: Einfügen, Löschen $\in \mathbf{O(\log n)}$

Spezielle Listen

ADT „Abstrakte Datentypen“

Stack $S = |\texttt{TOP}, \cdots$ Operationen nur auf letztem Element $\in O(1)$

Queue $Q = ||\texttt{HEAD}, \cdots, \texttt{TAIL}$ Vorne Löschen, hinten einfügen $\in O(1)$

Priority Queue $P = \begin{bmatrix} p_0 & p_1 & \cdots & p_n \\ a_0 & a_1 & \cdots & a_n \end{bmatrix}$
Jedes Element a hat Priorität p; Entfernen von Element mit höchster (`MIN`) Priorität

Sortierverfahren

Sortierproblem

Gegeben (endliche) Folge von Schlüsseln (von Daten) $(K_i)_{i \in I}$

Gesucht Bijektive Abbildung $\pi : I \to I$ (Permutation), sodass $K_{\pi(i)} \leq K_{\pi(i+1)} \quad \forall i \in I$

mit Optimierung nach geringen

- Schlüsselvergleichen C
- Satzbewegungen M

Eigenschaften

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

Relation *Stabil* vs. *instabil*: Vorherig 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. $x \leq y \land y \leq x \Rightarrow x = y$

Transitiv $x \leq y \land y \leq z \Rightarrow x = z$

Total (Vollständig) $x \leq y \lor y \leq x$

(ohne Total: „*Halbordnung*")

Grad der Sortierung

Anzahl der Inversionen Anzahl kleinerer Nachfolger für jedes Element:

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

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

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

Längster Run Anzahl der Elemente der längsten sortierten Teilliste:

$$\text{las}(L) := \max\{r.\text{len} \mid r \text{ ist Run in } L\}$$

$$\text{rem}(L) := L.\text{len} - \text{las}(L)$$

Einfache Sortierverfahren $\mathbf{O(n^2)}$

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[i] < 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 − −
        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
    i − −
end
```

Verbesserte Sortierverfahren $\mathbf{O(n \log n)}$

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 + +
    end
    while L[j] > piv do
        j − −
    end
    if i ≤ j then
        Swap L[i], L[j]
        i + +
        j − −
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
       ∀j > i erfüllen MHE
Output: Liste L mit MHE ∀j ≥ 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
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 − −
    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 ... m − 1] und L[m ... r]
       sortiert, Indices l, m, r
Output: Liste L mit L[l ... 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 ... 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 $\mathbf{O(n)}$

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

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

$$v = (v_1, \ldots, v_p) \leq w = (w_1, \ldots, w_q)$$
$$\Leftrightarrow \forall 1 \leq i \leq p : v_i = w_i \quad p \leq q$$
$$\lor \forall 1 \leq j \leq i : v_j = w_j \quad v_i < w_i$$

Fachverteilen Sortieren von n k-Tupeln in k Schritten: Sortieren nach letztem Element, vorletzem 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.

❧ ❧ ❧

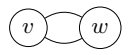
Algo.	Stabil	Mem.	Schlüsselvergleiche C_B	C_A	C_W	Satzbewegungen M_B	M_A	M_W	
Selection	✗	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$3(n-1)$	$3(n-1)$	$3(n-1)$	$O(n^2)$
Insertion	✓	1	$n-1$	$\overset{n\to\infty}{\approx} \frac{n(n-1)}{4} + n - \ln n$	$\frac{n(n-1)}{2}$	$2(n-1)$	$\frac{n^2+3n-4}{4} + n - 1$	$\frac{n^2+3n-4}{2}$	
Bubble	✓	1	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	0	$\frac{3n(n-1)}{4}$	$\frac{3n(n-1)}{2}$	
			Best-case		**Average-case**		**Worst-case**		
Shell	✗	1	-		-		-		$O(n \log n)$
Quick	✗	$\log n$	$n \log n$		$n \log n$		n^2		
Turnier	✗	$2n-1$	$n \log n$		$n \log n$		$n \log n$		
Heap	✗	1	$n \log n$		$n \log n$		$n \log n$		
Merge	✓	n	$n \log n$		$n \log n$		$n \log n$		
Untere Schranke $\Omega(n \log n)$ für allgemeine Sortierverfahren									
Distribution	✓	n	n		n		$n \log n,\ n^2$		$O(n)$

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 

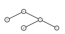
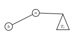
Zusammenhängend Für jede zwei Knoten gibt es genau eine Folge von Kanten die sie verbindet

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

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

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 $\text{Links}(\mathbf{u})$
- $BF(u) = +2, BF(v) \in \{0, -1\}$: Einfachrotation $\text{Rechts}(\mathbf{u})$
- $BF(u) = -2, BF(v) = +1$: Doppelrotation $\text{Rechts}(\mathbf{v}) + \text{Links}(\mathbf{u})$
- $BF(u) = +2, BF(v) = -1$: Doppelrotation $\text{Links}(\mathbf{v}) + \text{Rechts}(\mathbf{u})$

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, $B_h = \texttt{BUILD}(B_{h-1}, x, B_{h-2})$ für $h \geq 2$

(Maximal unbalancierter AVL-Baum der Höhe h)

Gewichtsbalancierte Binärbäume

Wurzelbalance $\rho(B) = \frac{n_l+1}{n+1}$ 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: $\boxed{P_0|K_1|P_1|\ldots|K_b|P_b}$
- 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 Schlüsselteillänge 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 Schlüsseln S werden Adressen/Indices A direkt berechnet,

$$h : S \to A$$

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 $h(K_i) := K_i \mod q$

- 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 Zufallsgenerators

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 Hashfunktionen h_i nach i-ter Kollision

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

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

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

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

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

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

Zufällig Deterministischer Zufallsgenerator generiert Schrittfolge z_i

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

Double-Hash Zweite Hashfunktion h'

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

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 Sondierungsfolgen, häufig spearater Überlaufsbereich mit dynamischer Zuordnung der Buckets

Dynamische Hashstrukturen

Nachteile der Hashtabelle

- Statische Allokationen speicherineffizient
- Re-hashing bei Speichererweiterung

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 Zeiger

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 Missmatch, kein Zurückgehen

Next-Tabelle

- Wie lang sind Präfix und Suffix gleich im Pattern vor jedem Buchstabe?
- next$[0] = -1$

Algorithm: Next-Tabelle

Input: Muster pattern$[0 \ldots m-1]$
Output: Tabelle next$[0 \ldots m]$

```
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 Missmatch oder kompletten Match verschieben des Präfix auf den Suffix (oder bei 0 komplett durch)

Algorithm: Knuth-Morris-Pratt-Suche

Input: Pattern$[0..m-1]$, String$[0 \ldots n-1]$, 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[i]
    end
    j ← j + 1
    i ← i + 1
    if j = m then
        Print i − m
        j ← next[i]
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$[0 \ldots |\Sigma| - 1]$

```
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 Missmatch verschieben des letzten Pattern-Buchstaben zu String-Buchstaben
- Wenn Patter-Buchstabe nicht vorhanden, dann komplett verschieben
- $C_A(n, m) \in O(n/m)$
- $C_W(n, m) \in O(n * m)$

Algorithm: Boyer-Moore-Suche

Input: Pattern$[0 \ldots m-1]$, String$[0 \ldots n-1]$, 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
    return −1
end
```

Statische Textsuche

- Index im Anhang von Büchern
- Signatur-Dateien

Approximative Suche

Hamming-Distanz Anzahl der Missmatches zwischen s_1 und s_2

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

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

Elektrischer Strom

Elektrisches Feld

Elektrische Ladung

$$Q = N * e_0 = [C] = [As]$$

- $1C = (6,242 * 10^{18}) * e_0$
- $e_0 = 1,602 * 10^{-19} C$

Culombsches Gesetz

$$\vec{F} = \frac{1}{4\pi\epsilon} * \frac{Q_1 * Q_2}{r^2} * (\vec{r_0}) = [N]$$

- $\epsilon_0 = 8,854 * 10^{-12} \frac{C^2}{Nm^2}$
- Ungleiche Ladungen (Q) ziehen sich an, gleich stoßen sich ab

Elektrisches Feldstärke

$$\vec{E} = \frac{\vec{F}}{q} = \left[\frac{V}{m}\right] = \left[\frac{N}{C}\right]$$

- Kraft, die Probeladung q erfährt
- Feldlinien von kleineren Ladung zur größeren Ladung (Positiv zu Negativ); gleich der wirkenden Kraftrichtung

Elektrisches Potential

$$\varphi(r) = \frac{Q}{4\pi\epsilon r} = (-\int_{\infty}^{r} \frac{Q}{4\pi\epsilon r^2} dr)$$

- Punktladung Q erzeugt Potential um sich
- Potential ist Steigung des E-Feld $E = -\frac{d\varphi}{dr}$

Elektrische Spannung

$$U = \frac{W}{q} = [V] = [\frac{Nm}{C}]$$
$$U_{r_1 \to r_2} = \varphi(r_1) - \varphi(r_2)$$

- Arbeit um q von r_1 nach r_2 zu bewegen $W_{r_1 \to r_2} = \int_{r_1}^{r_2} \vec{F} dr$

Elektrischer Strom

$$I = Q/t = [A] = \left[\frac{C}{s}\right]$$

- Gleichmäßig gerichteter Fluss von Elektronen von Minus nach Plus
- $1A = \frac{1}{1,602} * 10^{19}$ Elektronen pro Sekunde
- $\Rightarrow Q = \int_0^t i(t)dt$

Elektrische Arbeit

$$W = I * t * U = [Ws] = [J]$$

- Ladungstransport über Zeit mit Spannung
- Am Widerstand freigesetzte Energie $W = \frac{U^2}{R} * t$

Elektrische Leistung

$$P = \frac{W}{t} = U * I = [W] = [VA]$$

- Arbeit pro Zeit
- Am Widerstand $P = U^2/R$

Elektrisches Netz

Strom fließt per Definition von Plus $(+)$ nach Minus $(-)$

Generator G gibt Energie frei $W < 0$

Verbraucher R verbraucht E. $W > 0$

Verbindungsleitungen nach Kirchhoff:

Knoten K Verzweigung der Verbindungsleitung

$$\sum_{i \in K} I_i = 0A$$

- Stromrichtung einmalig willkürlich festlegen
- Eingehende Ströme addieren, ausgehende subtrahieren

Masche M Geschlossener Pfad ohne Knotenwiederholung

$$\sum_{k \in M} U_k = 0V$$

- Pfad startet im Knoten
- Vorher Spannungsrichtung (= Stromrichtung) einzeichnen
- Spannungsrichtung in Maschenrichtung addieren, entgegen Maschenrichtung (Quellen) subtrahieren

Lösen Linearer Gleichungssysteme

Kirchhoff'sche Sätze schaffen Lineares Gleichungssystem der Form

$$Ax = b$$

- x ist der gesuchte Vektor der Ströme $I_k = x_k$
- A ist die Matrix der Koeffizienten (Widerstände)
- b sind vom Strom unabhängige Größen (Spannungen, $0A$ im Knoten)

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

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

(Zeile $\times$ Spalte)

Determinante

$$\det A = \sum_{i=1}^{n} (-1)^{i+j} * a_{ij} * \det A_{ij}$$
$$= \sum_{j=1}^{n} (-1)^{i+j} * a_{ij} * \det A_{ij}$$

- Für Matrix $A = (n \times n)$
- „Entwickeln" nach i-ter Zeile oder j-ter Spalte
- A_{ij} = Matrix A ohne i-te Zeile und j-te Spalte
- Zeile/Spalte wählen mit viel $a_{ij} = 0$, damit $\det A_{ij}$ nicht berechnet werden muss

(2×2) **Matrix**

$$\det A = \begin{vmatrix} a & c \\ b & d \end{vmatrix} = ad - bc$$

(3×3) **Matrix (Regel von Sarrus)**

$$\begin{array}{ccc:cc} + & + & + & & \\ a_{11} & a_{12} & a_{13} & a_{11} & a_{12} \\ a_{21} & a_{22} & a_{23} & a_{21} & a_{22} \\ a_{31} & a_{32} & a_{33} & a_{31} & a_{32} \\ - & - & - & & \end{array}$$

Cramer'sche Regel

$$x_k = (I_k) = \frac{\det A_k}{\det A} \quad \det A_k \neq 0$$
$$A_k = (a_1 \mid \cdots \mid a_{k-1} \mid b \mid a_{k+1} \mid a_m)$$

- A_k ist Matrix A mit Vektor b statt k-ter Spalte

Wechselspannung

Elektromagnetisches Feld

Elektrische Bauteile

Elektrischer Leiter

Elektrische Flussdichte

$$D = \frac{Q}{A} = \left[\frac{C}{m^2}\right]$$

- Frei bewegliche Ladungsträger verteilen sich gleichmäßig auf der Oberfläche
- $\Rightarrow Q = A * \iint_A Dd$
- $\vec{D} = \epsilon_0 * \epsilon_r * \vec{E}$ (r raumfüllendes Material)

Metallischer Leiter

$$R = \rho \frac{l}{A}$$

- Linearer Widerstand, abhängig vom Material ρ
- $\rho = [\Omega \frac{mm^2}{m}] \propto$ Länge, kleinere Oberfläche

Ohmsch: Lineare Widerstände

$$U = R * I$$

- Kurz „Uri"
- Strom $\propto$ Spannung, kleinerer Widerstand

$$R = [\Omega] = \left[\frac{V}{A}\right]$$

Leitwert $G = 1/R = [S] = [\frac{A}{V}]$

Schaltung

Reihe $R_G = \sum R_k$

- $I_k = I \Rightarrow U_k = I * R_k$

Parallel $R_G = 1/\sum \frac{1}{R_k}$

- $U_k = U \Rightarrow I_k = U/R_k$

Kennlinie Graph $I(U)$

- Je flacher desto stärker der Widerstand
- Für nicht-lineare Graphen gilt das Ohmsche Gesetz nicht!

Kapazitiv: Kondensator

$$Q = C * U$$

(„Kuh gleich Kuh“)

$$E = \frac{U}{d} = \frac{D}{\epsilon}$$

Kapazität

$$C = \frac{\epsilon * A}{d} = [F] = \left[\frac{C}{V}\right]$$

- Kondensator speichert elektrische Ladung
- $\propto$ Große Oberfläche, große Permittivität

Schaltung

Reihe $C_G = 1/\sum \frac{1}{C_k}$

Parallel $C_G = \sum C_k$

Influenz: Faraday'scher Käfig Das Innere eines metallischen Hohlraums ist feldfrei.

Induktiv: Spule

Quellen

Messgeräte

Voltmeter

- Schaltung in Parallel

Amperemeter

- Schaltung in Reihe

Spezielle Kombinationen

Spannungsteiler

$$U_A = \frac{U_0}{\frac{R_1}{R_2} + 1}$$

Potentiometer $R_1 = R - R_2$

$$\Rightarrow U_A = U_0 * \frac{R_2}{R}$$

Potentiometer unter Last R_L $R_1 = R - (R_2 \parallel R_L)$

$$\Rightarrow U_A = U_0 * \frac{R_2}{R} * \frac{R_L}{R_L + R_2}$$

Transformator

Schwingkreis