

Algorithmen und Datenstrukturen

- Ein Algorithmus ist eine Vorschrift zur Lösung eines Problems, die für eine Realisierung in Form eines Programms auf einem Computer geeignet ist.
- Ein Algorithmus ist eine präzise Handlungsvorschiff, um aus vorgegebenen Eingaben in endlich vielen Schritten eine bestimmte Ausgabe zu ermitteln.
- Die Abfolge der Verarbeitungsschritte muss eindeutig aus einer Algorithmus hervorgehen
- Hauptziel ist korrekte Problem Lösung
→ Terminiertheit → Partielle Korrektheit

Datenstruktur = Menge von Datentypen, zwischen denen Beziehungen bestehen
„Objektmenge (Werte) + Operationen“

- O-Notation: $\lim_{n \rightarrow \infty} \frac{t(n)}{O(t(n))} \leq c \Rightarrow t(n) \in O(c(n))$
- Master-Methode: $T(1) = 1, T(n) = \alpha \sum_{i=1}^k T(\frac{n}{b}) + f(n)$
 - ① $f(n) = O(n^{\log_b a - \epsilon}) \Rightarrow T(n) = \Theta(n^{\log_b a})$
 - ② $f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \log n)$
 - ③ $f(n) = \Omega(n^{\log_b a + \epsilon}) \wedge \exists c_1, t_{n_0}: c_1 f(n/b) \leq c f(n) \Rightarrow T(n) = \Theta(f(n))$
 - ④ $f(n) = \Theta(n^{\log_b a} (\log n)^k) \wedge k \geq 0 \Rightarrow T(n) = \Theta(n^{\log_b a} (\log n)^{k+1})$

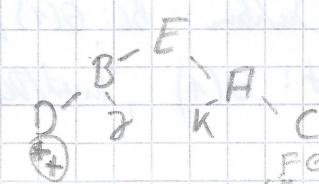
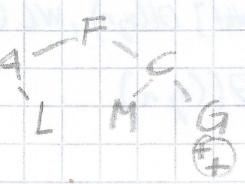
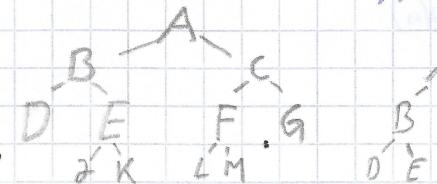
• Baum: Wurzel hat immer Tiefe 0, Blätter haben Höhe 0
Grad = # Nachfolger

• Preorder: ROOT, LEFT, RIGHT, Kinder: LEFT, ROOT, RIGHT

$$\begin{matrix} & 2 & 1 & 3 \\ & \swarrow & \uparrow & \searrow \\ 2 & & 1 & 3 \end{matrix}$$

• AVL-Baum: ∀ Knoten v: $|h(v.\text{left}) - h(v.\text{right})| \leq 1$
 $\lceil \log(n+1) \rceil - 1 \leq h(B) \leq 1,4404 \lceil \log(n+2) \rceil$

AVL-Einfügen: Einfach-Rotation, wenn außer Teilbaum zu hoch
Doppel-Rotation, wenn inner: erst unten andere Richtung



• Rot-Schwarz-Baum

- ① Jeder Knoten ist Rot oder Schwarz
- ② Wurzel ist schwarz
- ③ "Nullptr" ist schwarz (RS-Blatt)
- ④ Jeder Rote Knoten darf nur schwarze Nachfolger haben
- ⑤ Jeder Knoten hat eindeutige Schwarz Höhe! Tiefe Punte zu einem RS-Blatt hat gleiche # zw. Knoten.

• B-Baum Ordnung $t \geq 2$:

- ① Alle Blätter haben die gleiche Tiefe
- ② Knoten mit Nachfolgern und i weiter haben genau $i+1$ Nachfolger
- ③ Alle Knoten (Wurzel) haben mind. $\lceil t/2 \rceil - 1$ und max. $t-1$ Werte
- ④ Suchbaum.

• Hashing

Divisions/Kongruenzmethode: $h(s) = s \bmod p$, $p = 2^k$ size Hashtabelle = p

Multiplikationsmethode: $h(s) = L_p \cdot ((s \cdot x) \bmod 1)$, $x \in [0, 1]$, z.B. $x = (\sqrt{5}-1)/2$

Hashing mit offener Adressierung: $h(s, i)$, $i = \text{Versuch} \# \in \mathbb{N}$

→ Lineares Probiere $h(s, i) = (h(s, 0) + i) \bmod p$

→ Quadrat. Probiere $h(s, i) = (h(s, 0) + c_1 + d c_2 i^2) \bmod p$

→ Doppeltes Hashing $h(s, i) = (h(s, 0) + i h_2(s)) \bmod p$

Sortiermethode	stabil	BC	AC	WC
Bubble Sort	+		$\Theta(n^2)$	
Selection Sort	-		$\Theta(n^2)$	
Insertion Sort	+	$\Theta(n)$	$\Theta(n^2)$	
Merge Sort	+		$\Theta(n \log n)$	
Heap Sort	-		$\Theta(n \log n)$	
Quick Sort	-		$\Theta(n \log n)$	$\Theta(n^2)$
Count Sort	/		$\Theta(nK)$ falls $K = \Theta(n)$	
Map Sort			$\Theta(n)$	falls $G(N, O(n))$

Bäume, Hamwi
Max Teilsumm $n!$

Binär verh. Suchbaum: BC: $\Theta(1)$, AC (wahlf.) $\Theta(\log n)$, WC $\Theta(n)$

AVL Rot: $\Theta(1)$, Inset/Del: $\Theta(\log n)$

allg. Breitensuche = $\Theta(|V| + |E|)$

allg. Tiefensuche = $\Theta(|V| + |E|)$

Topol. Sort in DAG $\Theta(|V|)$

schnittweise Mengen vereinigen $\Theta(\log n)$

Kruskal = minimale Kanten addl. $\Theta(|E| \lg |E|)$
 $= \Theta(|E| \lg |V|)$

Prim = billigste Kante addl. $\Theta(|E| \lg |V|)$
→ mit Heaps
→ "Fib-Heaps Deque" in $\Theta(1)$ $\Rightarrow \Theta(|E| + |V| \lg |V|)$

Dijkstra (VSS & negative Gew.) $\Theta(|V| \lg |V| + |E| \lg_2 |V|)$
→ nur mit Heaps, FibHeaps: $\Theta(|V| \lg |V| + |E|)$

Bellmann-Ford ($n-1$ mal alle Kanten): $\Theta(|V| \cdot |E|)$

FPSF: $\Theta(|V|^4)$, fischer: $\Theta(|V|^3 \lg |V|)$

Floyd-Warshall: $\Theta(|V|^3)$