$\forall G(V,E): \sum_v \deg(v) = |E| \cdot 2$ | Durchmesser: Maximum der Abstände

Baum: kein Kreis $\Rightarrow |E| = |V| - 1$ BAUM

$\pi \hat{=}$ Vorgängerknoten | $O(|V|+|E|)$ | Back-K. grau → grau

DFS: $d \hat{=}$ Abstand von Wurzel | Forward-K. grau → schwarz

DFS: Tree-Kante: grau → weiß | Cross-K. alles andere

DAG $\Leftrightarrow$ DFS: keine B-Kanten BFS/DFS

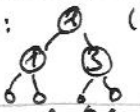
Prim: r aufzeigen $\Rightarrow$ immer die Kante mit dem geringsten Gewicht dazu (zusammenhängend) $O(|E| \log |V|)$ MST

Kruskal: $\emptyset$ anfangen $\Rightarrow$ immer die leichteste Kante dazu, die keinen Kreis erzeugt - auch nicht zusammenhängend

O dann Θ gleich Ω unter o stark oder ω stark unter Θ

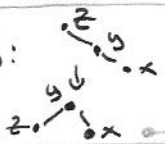
Stirling: $\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \leq n! \leq \sqrt{2\pi n}\left(\frac{n}{e}\right)^{n+1/12n}$

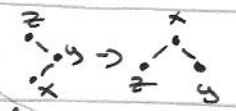
$a,b > 0$: $(\log n)^a \in O(n^b)$ | $n^a \in O(b^n)$

Such-Baum:  (Inorder → geordnet) AVL

AVL-Baum: Höhe $O(n \log n)$ $(n(h) \geq 1 + n(h-1) + n(h-2)$

$\geq 2n(h-2) \geq \dots \geq 2^{\frac{h-1}{2}} n(1)$

Einfache Rot: 

Doppelte: 

Reihenfolge der Teilbäume bleibt

Suchen/Einfügen/Löschen: $O(\log n)$

MST(Prim) + Π-Zeiger auf das anbindende Element

Kruskal: Größe der Komponente: Anzahl d. Knoten

Iteration: in Liste einfügen $\Theta(n^2)$ SORTIERUNGEN

Selektion: kleinstes Elem. einfügen ~~an Stelle des 1. Elem.~~ $\Theta(n^2)$

Quick: teilen an def. Element und rekursiv sortieren $\Theta(n \log n)$

Merge: halbieren und später rekombinieren $\Theta(n \log n)$ $O(n^2)$

Counting: Elemente $\in K \Rightarrow$ Duplikate und zählen, neu Durchlauf und ausgeben (n-mal) $O(n+K)$

Radix: Ziffern von hinten nach vorn (für d Ziffern) $\Theta(dn)$

Heap: Heap erstellen, Wurzel entnehmen $O(n \log n)$

Binär: max. n stellige Zahlen: $O(\log n)$ SORTIERUNGEN

vergleichsbasierte Sortierung: worst case $\Omega(n \ln n)$

public class A extends B | final → keine Vererbung BAUM

public abstract class $\hat{=}$ interface

Wurzel: Tiefe 0 | Blatt: Höhe 0 BAUM

Heap: binärer Baum (sortiert mit Knoten $\leq$ Kinder und linksfüllend) | Kinder bei Array: $i \Rightarrow 2i+1 / 2i+2$ HEAP

Höhe eines Heaps mit n Zahlen: $h = \lceil \log(n+1) \rceil$

Bottom-Up: $n/2$ Heaps kombinieren und neu einsinken

Kollisionen: $\lceil \frac{n}{N} \rceil$ Divisionsmethode: $h(x) = x \bmod N$ HASH

MAD: $h(x) = (ax+b) \bmod N$ (a teilerfremd zu N)

Sondieren: $h(x) \to h(x)+1$ ($h(x)+2 \dots$ ggf. n^2

fair: $i,j \in \{0,\dots,N-1\}$: $\left| |h^{-1}(i)| - |h^{-1}(j)| \right| \leq 1$

$\Rightarrow O(1+\frac{n}{N})$ universell: $|\{h \in H \mid h(x) = h(y)\}| = \frac{|H|}{N}$

Java: Attribute des jeweiligen Objekts; Methoden der unteren Klasse; Methoden greifen auf Attribute des eigenen Objekts zu