

1) Komplexität:

Landau-, O-Notation:

$$O(f) = \{g \mid \exists c \in \mathbb{R} \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}$$

$$f \in O(f)$$

$$f \in O(g) \Leftrightarrow g \in O(f) \text{ genau dann wenn } O(f) = O(g)$$

$$g \in O(f) \rightarrow O(g) \subseteq O(f)$$

Prüfungskriterium:

$$\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = \infty \quad O(f) \subset O(g)$$

$$\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = c > 0 \quad O(f) = O(g)$$

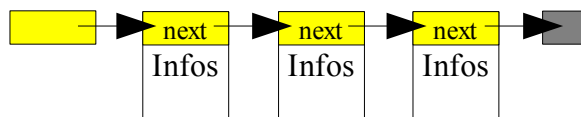
$$\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = 0 \quad O(g) \subset O(f)$$

Reihenfolge der O-Notations-Klassen:

$$O(1) \subset O(\log_2(n)) \subset O(n) \subset O(n \cdot \log_2(n)) \subset O(n^2) \subset O(2^n) \subset O(5^n) \subset O(n!)$$

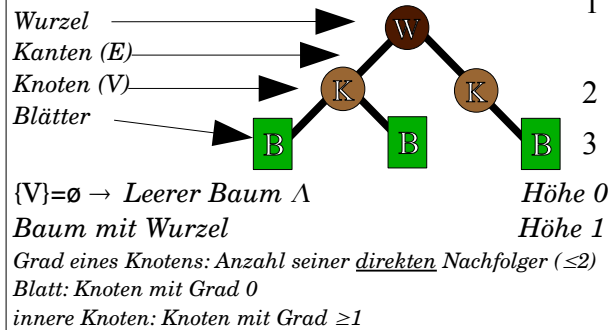
2) Verkettete Strukturen:

Lineare Listen:



einfügen
löschen
blubber

Binäre-Bäume:



Stapel:

LIFO-Prinzip

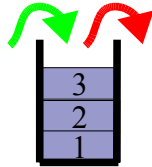
Die Ersten werden die Letzten sein.

Operationen:

TOP: Liefert oberstes Element eines Stapels

PUSH: Push legt Element oben auf den Stapel

POP: Entfernt oberstes Element auf Stapel



Schlangen:

FIFO-Prinzip

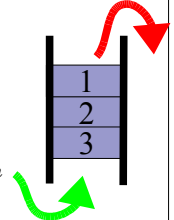
Wer zuerst kommt mahlt zuerst.

Operationen:

TOP: Liefert oberstes Element einer Schlange

PUSH: Push reiht Element hinten in Schlange ein

POP: Entfernt oberstes Element in der Schlange



Preorder:

- 1 Besuch: Bearbeite information, verzweige zum linken Sohn
- 2 Besuch: Verzweige zum rechten Sohn
- 3 Besuch: Entferne K vom Stapel

W - L - R

Inorder:

- 1 Besuch: Verzweige zum linken Sohn
- 2 Besuch: Bearbeite information verzweige zum rechten Sohn
- 3 Besuch: Entferne K vom Stapel

L - W - R

Postorder:

- 1 Besuch: Verzweige zum linkem Sohn
- 2 Besuch: Verzweige zum rechten Sohn
- 3 Besuch: Bearbeite Information in K, Entferne K vom Stapel

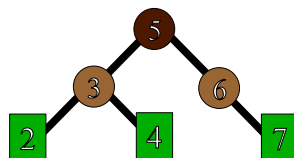
L - R - W

Suchbaum:

Linker Teilbaum ist kleiner als Wurzel.

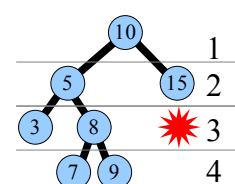
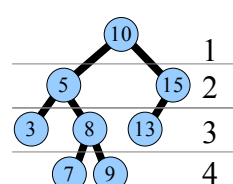
Rechter Teilbaum ist größer als Wurzel.

Linke und Rechte Teilbäume sind Suchbäume



Beim Löschen wähle als ersatz grösstes Element im linken Teilbaum.

AVL-Baum:



Es gilt: $|h(T_L) - h(T_r)| \leq 1$

Balancierte Bäume:

Gewichtsbalanciert: $|T_L| \approx |T_r|$

Höhenbalanciert: $|h(T_L)| \approx |h(T_r)|$

Grundoperationen zur Reorganisationen von binären Suchbäumen:

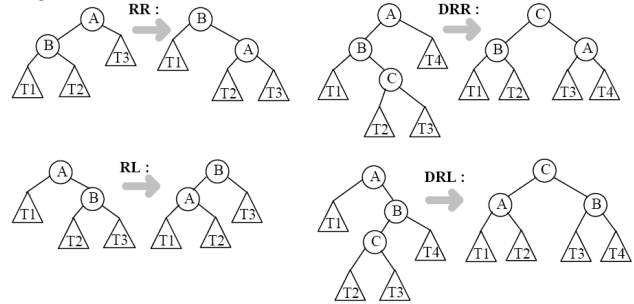
RR: Rotation nach Rechts

DRR: Doppelte Rotation nach Rechts

RL: Rotation nach Links

LRR: Doppelte Rotation nach Links

Diagramme :



3) Rekursionsgleichungen

Teilen und Herschen:

$$T(1) = g(1)$$

$$T(n) = a \cdot T(\lfloor a/b \rfloor) + g(n)$$

a = Größe der zu lösenden Teilprobleme

$\lfloor a/b \rfloor$ = Größe der Teilprobleme

$g(n)$ = Aufwand zum Aufteilen

Hinführung zum Mastertheorem

$$f(n) = \begin{cases} g(1) & n=1 \\ a \cdot f(n/b) + g(n) & n=b^k > 1 \end{cases}$$

$$f(n) = \sum_{i=0}^k a^i \cdot g\left(\frac{n}{b^i}\right) \quad \text{mit } n=b^k \quad f(n) = \sum_{i=0}^k a^i \cdot g(b^{i-k})$$

Mastertheorem

$$f(n) = \begin{cases} c & n=1 \\ a \cdot f(n/b) + c & n=b^k > 1 \end{cases}$$

$$f \in \begin{cases} \Theta(\log n) & a=1 \\ \Theta(n^\gamma) & a>1 \text{ und } \gamma=\log_b(a) \end{cases}$$

$$h(n) = \begin{cases} c & n=1 \\ a \cdot f(n/b) + c \cdot n^p & n=b^k > 1 \end{cases}$$

$$f \in \begin{cases} \Theta(n^p) & a \leq b^p \\ \Theta(n^p \cdot \log(n)) & a = b^p \\ \Theta(n^\gamma) & a > b^p \text{ und } \gamma = \log_b(a) \end{cases}$$