Bäume, binäre Bäume, binäre Suchbäume und AVL-Bäume

Was ist ein Baum?

Nicht-lineare Datenstruktur

Baumstruktur: *Eine hierarchische Struktur, die aus Knoten und Kanten besteht*

Quelle: Informatik 2 (Schöningh) Seite 184 Infokasten

Wurzel:

Einziger Knoten ohne Vorgänger

Pfad:

- Der Weg über Kanten den man gehen muss, um von einem Knoten zu einem anderen zu gelangen
- unmöglich im Kreis zu laufen

Tiefe:

Anzahl der Kanten auf dem Pfad von der Wurzel bis zu diesem Knoten.

Grad:

Der (Verzweigungs-) Grad (oder die Ordnung) eines Baumes bezeichnet die maximale Anzahl der Nachfolger, die ein Knoten dieses Baumes haben kann.

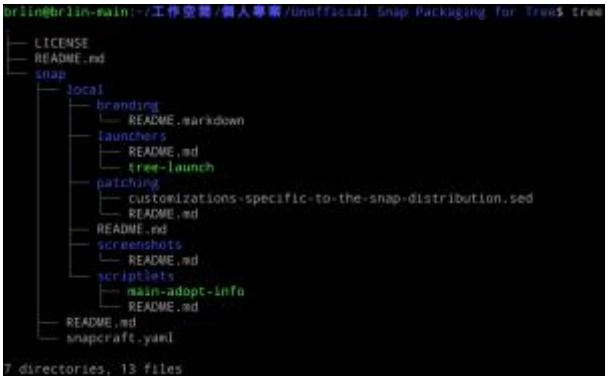
Ebenen/Stufen:

- gehen von der Wurzel aus
- *Alle Knoten mit dem gleichen Abstand zur Wurzel liegen auch in derselben Ebene*
- graphische Darstellungen von Baumstrukturen sollen dies zum Ausdruck bringen

Teilbaum

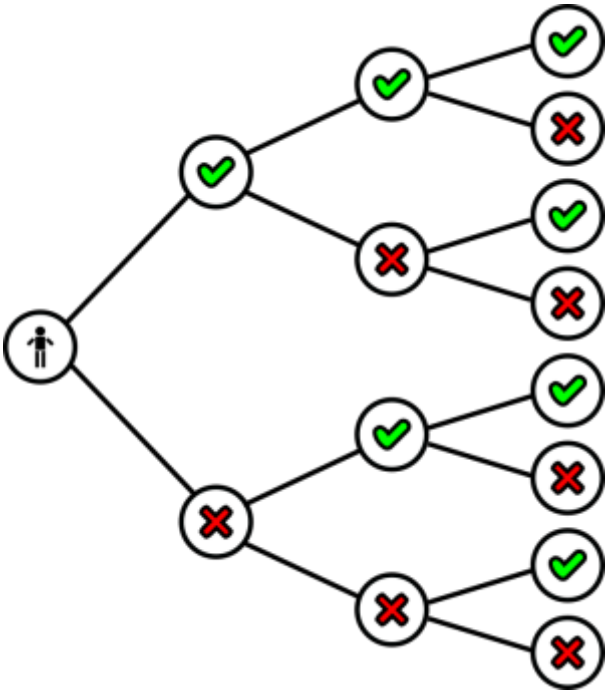
- sind rekursiv (selbstähnlich) aufgebaut
- Jeder zusammenhängede Teil eines Baumes ist wieder ein Baum

Quelle: Informatik 2 (Schöningh) Seite 184 -185



1 - Ein Baum zum Darstellen einer Datenstruktur

Binärer Baum



2 - Ein binärer Baum

Quelle: https://openclipart.org/detail/15876/event-tree

Kriterien eines binären Baums:

- es gibt nur genau eine Wurzel.
- jeder Knoten außer der Wurzel hat genau einen Vaterknoten.
- Die Wurzel ist der Vorfahre jedes Knotens.
- Jeder Knoten hat höchstens zwei Kinderknoten, ein linkes und ein rechtes.

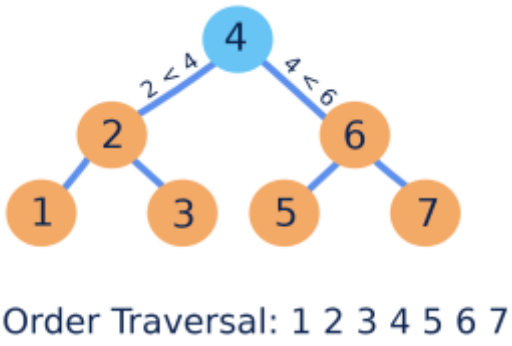
Quelle : http://www.inf.fu-berlin.de/lehre/WS11/infa/bintree.pdf

Methoden eines binären Baumes:

- isEmpty() - gibt "true" zurück wenn der Baum leer ist
- setContent(ContentType pContent) - ermöglicht das einfügen eines Objektes in den Baum
- getContent() - gibt das Objekt an der aktuellen Stelle zurück.
- setLeftTree(BinaryTree<ContentType> pTree) - übergibt den BinaryTree pTree als linken Teilbaum des Knotens.
- setRightTree(BinaryTree<ContentType> pTree) - übergibt den BinaryTree pTree als rechten Teilbaum des Knotens.

Traversierung eines binären Baumes:

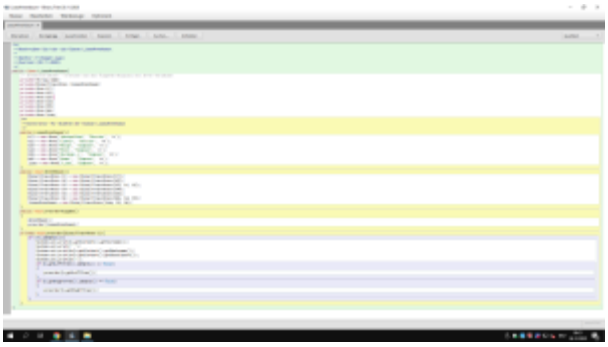
- Pre-Order Traversierung: Wurzel - linker Teilbaum - rechter Teilbaum
- In-Order Traversierung: linker Teilbaum - Wurzel - rechter Teilbaum
- Post-Order Traversierung: linker Teilbaum - rechter Teilbaum - Wurzel



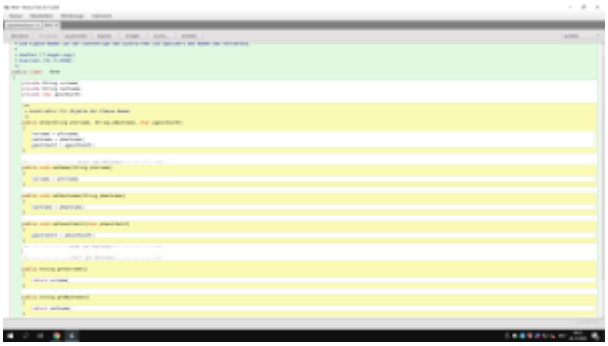
3 - In Order Traversierung eines binären Baumes.

Beispiel der Implementation eines binaren Baumes

4 - Implementationsdiagramm eines binären Baumes



5 - Implementation der Klasse LisasAhnenbaum



6 - Implementation des ContentType Ahne

Binäre Suchbäume

Was ist ein Binärer Suchbaum/ Wofür wird er genutzt?

- Sortierte Datenstruktur-> Schnelleres finden von Dateien bzw. Daten
- Durch schrittweises einfügen einfach zu erstellen

Kriterien für einen binären Suchbaum:

- Alle Kriterien die auf einen binären Baum zutreffen müssen
- Alphabetische bzw. Zahlen Wert orientierte Hierachie (Links -> Kleiner, Rechts->Größer)

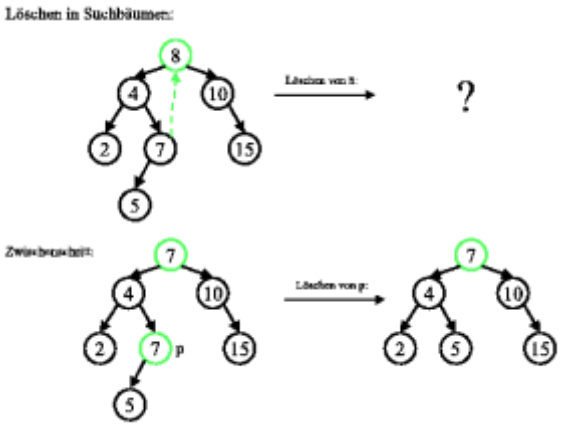
Unterschiede zu einem einfachen binären Baum:

- Zusätzliche Methoden zum Vergleich von Elementen -> Interface(isGreater(), isEqual(),isLess())
- Beim Einfügen (insert()) werden Elemente jeweils mit der Wurzel verglichen. Hierzu führen sie z.B. isGreater() aus. Diese Methode gibt true zurück, falls das aktuelle element größer als die Wurzel ist. Die Methode nutzt hierbei die methode compareTo() um Buchstaben so wie Zahlen einem Zahlenwert zu zu ordnen. Ein negativer Rückgabewert heißt dass die verglichene Zahl oder der Buchstabe weiter vorne im Alphabet oder in der Zahlenreihe vorhanden ist.
- Weiteres Beispiel: Innerhalb des Baumes soll überprüft werden, ob zwei Werte die gleichen sind. Dafür wird isEqual() benutzt. Sie gibt bei gleichen Zahlen true als Wahrheitswert zurück.
- Einfügen von "Oben nach unten"

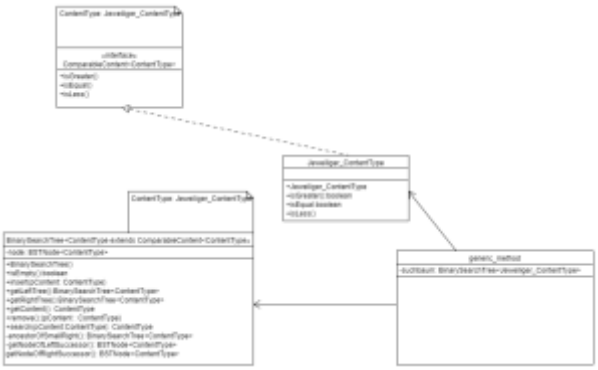
Löschen von Daten aus einem Suchbaum:

- Die Methode remove() ist zuständig
- Falls das zu löschende objekt das letzte eines Zweiges ist, kann es einfach entfernt werden
- Bei einem Objekt welches Nachfolger hat: Das kleinste Objekt des rechten Teilbaums bzw. das größte des linken wird als neue Wurzel verwendet, der restliche Strang wird wieder angehangen

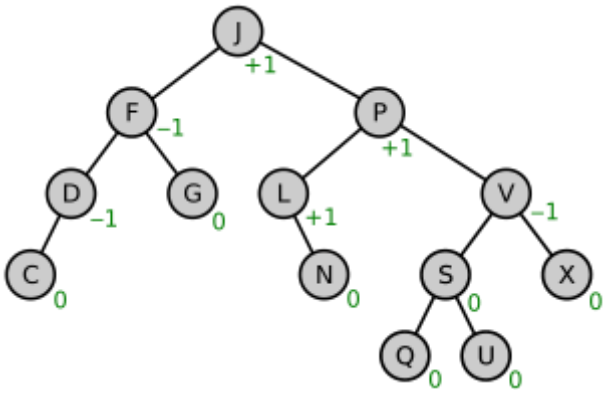
7 - Beispielimplementation einer Benutzerverwaltung



8 - Quelle : http://www.hotti.ch/doc/avlbaeume/node4.html



AVL-Bäume



9 - Quelle: *https://de.wikipedia.org/wiki/AVL-Baum#/media/Datei:AVL-tree-wBalance_K.svg*

Merkmal:

Selbstausbalancierender binärer Suchbaum

Denfinition:

Ein Knoten eines binären Baumes heißt ausgeglichen oder balanciert, wenn sich die Höhen seiner beiden Söhne um höchstens 1 unterscheiden.

Quelle: http://www-lehre.informatik.uni-osnabrueck.de/~ainf/2000/skript/node60.html

Bedingung:

Höhe der beiden Nachkommen dürfen sich um höchstens 1 unterscheiden

- d.h. die Höhe des Baums wächst nur logarithmisch
- Balancefaktor BF = Höhe (rechter Teilbaum) - Höhe (linker Teilbaum)
- Höhe = Anzahl der Kinder/Blätter
- BF wird von den Blättern zur Wurzel bestimmt
- Max. Höhenunterschied: 1 bzw. -1

Einfügen/Löschen:

Beim Einfügen oder Löschen wird der Baum durch Rotation(en) rebalanciert.

- Der Balancefaktor wird wieder neu berechnet
- Dort wo der Balancefaktor die Bedingungen nicht erfüllt wird eine Rotation durchgeführt (siehe Tabelle)

Quelle: https://studyflix.de/informatik/avl-baum-1434

Rotation - Hält den Baum balanciert

Rotation	BF Oberer Knoten	BF Unterer Knoten
Rechts-Rotation	-2	-1
Links-Rotation	+2	+1
Rechts-Links-Rotation	+2	-1
Links-Rechts-Rotation	-2	+1

10 - Quelle: https://studyflix.de/informatik/avl-baum-1434

11 - Quelle: https://ccsearch-dev.creativecommons.org/photos/4e011d51-82ec-4728-b00f-21093ed5906e

Laufzeit

Die Laufzeit bzw. Komplexität von AVL-Bäumen ist abhängig von den verschiedenen Operationen:

	Average-Case	Worst-Case
Suchen	$O(logn)$	$O(logn)$
Einfügen	$O(1)$	$O(logn)$
Löschen	$O(1)$	$O(logn)$

12 - Quelle: https://studyflix.de/informatik/avl-baum-1434

Weitere Resourcen:

Bäume:

- Buch Schöningh Informatik 2 Seite 183 - 185 Erklärung
- Buch Schöningh Informatik 2 Seite 186 - 188 Aufgaben

binäre Bäume:

- Buch Schöningh Informatik 2 Seite 189 - 192f. Erklärung
- Buch Schöningh Informatik 2 Seite 193 - 194 Aufgaben

binäre Suchbäume:

- Buch Schöningh Informatik 2 Seite 195 - 201f. Erklärung
- Buch Schöningh Informatik 2 Seite 202 - 206 Aufgaben

AVL-Bäume:

- https://studyflix.de/informatik/avl-baum-1434

- http://www-lehre.informatik.uni-osnabrueck.de/~ainf/2000/skript/node60.html
- https://de.wikipedia.org/wiki/AVL-Baum