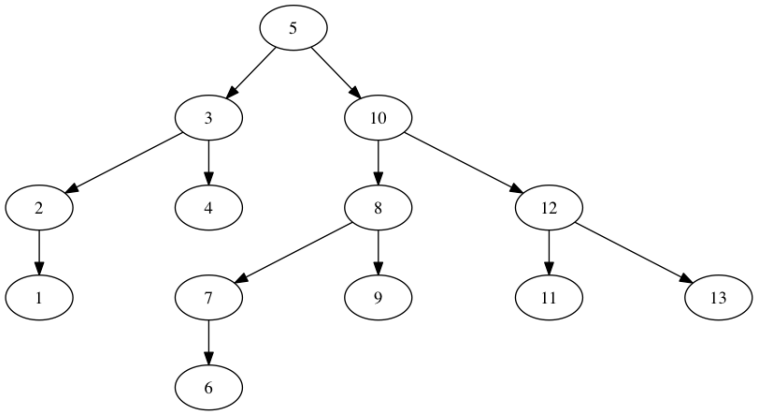
Übungsblatt 4

Ausgabe: 4.5.2018
Abgabe: 16.5.2018 (23:59) - Aufgaben 1 und 2 - / 30.5.2018 (23:59 Uhr) - Aufgabe 3 -
Testat: 31.5.2018 (13:40 –16:50 Uhr)

Aufgabe 1: AVL-Bäume - *Algorithmen* - 30 Punkte

(a) Fügen Sie die Schlüssel 1, 3, 5, 6, 7, 4 und 2 in dieser Reihenfolge in einen initial leeren AVL Baum ein. Geben Sie den Baum jeweils direkt nach dem Einfügen sowie nach jeder Rotation an.

(b) Gegeben sei der folgende AVL-Baum:



Überlegen Sie sich, welche Rotationen eine Lösch-Operation auf AVL-Bäumen bewirken können. Löschen Sie aus dem Baum das Element 5, indem Sie es durch das größte Element im linken Teilbaum ersetzen. Markieren Sie den betroffenen Knoten, falls das AVL-Kriterium durch das Löschen verletzt wird, und rebalancieren sie den Baum unter Angabe der notwendigen Rotationen.

(c) In Aufgabenteil (a) war jeweils nur eine Rebalancierung notwendig. Muss das immer so sein? Wenn nein: Konstruieren Sie einen Fall, bei dem ein Schlüssel eingefügt wird und zwei oder mehr Rebalancierungen notwendig werden. Wenn ja: Begründung!.

(d) Gilt die Aussage von (c) auch für das Löschen? Bitte eine plausible, kurze Begründng!

Aufgabe 2: B-Bäume - *Algorithmen* - 30 Punkte

(a) Fügen sie nacheinander die Werte 4, 6, 1, 7, 8, 2, 5, 9, 10 in einen B-Baum der **Ordnung 1** (2-3 Baum) ein.

Malen sie den Baum nach jeder einzelnen Einfügeaktion, welche die B-Baum-Eigenschaft verletzt, hin.

Sofern beim Einfügen eines Wertes eine Maßnahme zur Wiederherstellung der B-Baum-Eigenschaft notwendig ist, geben sie die entsprechende Maßnahme an und anschließend den veränderten B-Baum.

(b) Löschen Sie im B-Baum aus Teil (a) den Wert 7, indem Sie ihn durch den kleinsten Wert des rechten Teilbaums ersetzen.
Sofern ein Maßnahmen zur Wiederherstellung der B-Baum-Eigenschaft notwendig ist, geben sie die entsprechenden Maßnahmen an und anschließend den veränderten B-Baum.

Aufgabe 3: B-Baum **90 Punkte**

Im B-Baum werden alle Arten von Objekten gespeichert, die miteinander vergleichbar, also vom Typ `Comparable` sind. Es sind immer nur Elemente derselben Klasse miteinander vergleichbar!
Der in der Vorlesung vorgestellte ADT BTree (B-Baum) stellt über ein `Interface` die folgende Funktionalität zur Verfügung:

- `boolean insert (Comparable o)` - fügt o in den B-Baum ein.
- `boolean insert (String filename)` - fügt die Elemente, die in der Datei stehen in den Baum ein.
- `boolean contains(Comparable o)` - testet, ob o im Baum vorhanden ist.
- `void delete (Comparable obj)` - löscht das Objekt obj, sofern es im Baum enthalten ist.
- `int size()` - ermittelt die Anzahl der Elemente im Baum.
- `int height()` - ermittelt die Höhe des Baums.
- `Comparable getMax()` - liefert das größte Element im Baum.
- `Comparable getMin()` - liefert das kleinste Element im Baum.
- `booelan isEmpty()` - ist true genau dann, wenn der Baum leer ist.
- `addAll(BTree otherTree)` - fügt alle Elemente des übergebenen B-Baums (`otherTree`) in den aktuellen Baum ein.
- Ausgabemethoden:
  - `void printInorder()` - Ausgabe des Baums in Inorder.
  - `void printPostorder()` - Ausgabe des Baums in Postorder.
  - `void printPreorder()` - Ausgabe des Baums in Preorder.
  - `void printLevelorder()` - Ausgabe des Baums in Levelorder.

  Überlegen Sie genau, wie die jeweilige Ausgabe aussehen muss. Es ist nicht ganz so einfach, wie bei Binärbäumen!

Da auf den Elementen des Baums eine Ordnungsrelation definiert ist, dürfen immer nur Elemente vom selben Typ in dem Baum abgelegt werden. Da noch nicht alle technischen Voraussetzungen in der Vorlesung besprochen wurden, dürfen Sie davon ausgehen, dass diese Bedingung (automatisch) erfüllt ist. Beachten Sie, dass Sie alle Methoden so, schreiben, dass sie für Elemente vom Typ `Comparable` funktionieren. Insbesondere betrifft das den Vergleich von 2 Elementen und die Ausgabe eines Elements.

Implementieren Sie in der Klasse `BTree` die Methoden des Interface. Dem Konstruktor wird die Ordnung des B-Baums als Parameter übergeben.

Um das Testen zu erleichtern, ist die Methode `boolean insert (String filename)` vorgesehen. Dabei werden alle Elemente auf der Datei in den Baum eingefügt.

Denken Sie daran, dass Sie private Methoden zur Behandlung von Knoten, die **platzen** bzw. **defizitär** werden, schreiben müssen.

20 Zusatzpunkte erhalten Sie, wenn Sie die Methode `BTree clone ()` für die Erstellung einer tiefen Kopie des B-Baums implementieren.

Schreiben Sie ein Rahmenprogramm, das es ermöglicht, über ein geeignetes Menü die einzelnen Operationen auszuführen.

Gehen Sie so vor, dass Sie zuerst die Methode `boolean insert (Comparable o)` implementieren und dann die Methoden zur Ausgabe der Werte des Baums. Dann lassen sich die restlichen Methoden einfacher testen.

Abgabe – Mit Javadoc kommentiertes Programm inkl. JUnit-Tests (soweit wie möglich). Die sind bei B-Bäumen i.d.R. nicht für alle Test-Konstellationen sinnvoll anzugeben!