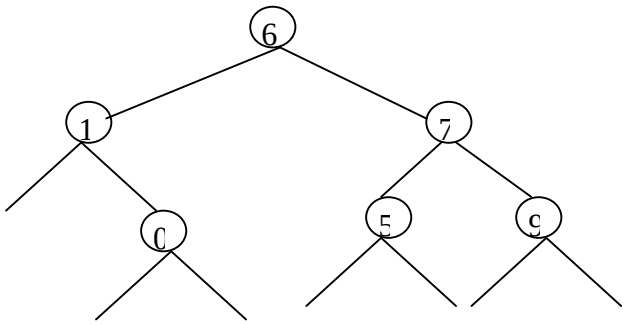
Kanaeva / Schweppe, FUB Informatik

Übung zu Algorithmen und Programmieren III, WS 2001/2

Übung 11

Ausgabe: 15.01.02
Abgabe: 24.01.02 bis 14.00 Uhr

Aufgabe 1 (2+4 P)
a) Gegeben sei folgender binärer Baum (kein Suchbaum):



Traversieren Sie den Baum mit Preorder, Inorder, Postorder und Levelorder und geben Sie die entsprechenden Folgen der Baumelemente an.

b) Schreiben Sie eine Haskell-Funktion namens `pre_post2tree`, die aus den Preorder- und Postorder-Traversierungen eines binären Baumes T den Baum T rekonstruiert. Im Baum T kommen keine Duplikate vor. Benutzen Sie dabei den folgenden Typ `BinTree`:

```
data BinTree = Leaf |  Node BinTree Int BinTree deriving (Show)
```

`pre_post2tree [2,1,7,3,4,5] [7,1,4,5,3,2]` soll z.B. die folgende Ausgabe haben:

```
Node (Node Leaf 1 (Node Leaf 7 Leaf)) 2 (Node (Node Leaf 4 Leaf) 3 (Node Leaf 5 Leaf))
```

Aufgabe 2 (3+5 P)
a) Gegeben seien drei Schlüsselfolgen:

(i) 1, 2, 3, 4, 5, 6, 7
(ii) 7, 2, 8, 11, 22, 9, 0, 4, 5, 3
(iii) 1, 6, 7, 2, 4, 3, 5, 9

Konstruieren Sie zu jeder Folge einen AVL-Baum, indem Sie die Schlüssel in der angegebenen Reihenfolge in den Baum einfügen. Benutzen Sie dabei den in der Vorlesung angegebenen Algorithmus. Zeichnen Sie den Baum nach jeder Einfügeoperation, geben Sie die notwendige Rotation (einfach oder doppelt) an und veranschaulichen Sie die Rotation in dem Baum. Annotieren Sie dabei in jedem Knoten den Balancefaktor.

b) Schreiben Sie ein Programm, das für einen gegebenen binären Suchbaum T entscheidet, ob T ein AVL-Baum ist. Geben Sie den Zeitaufwand Ihres Programms mit der O-Notation an.

Bitte wenden

Aufgabe 3 (4+2+4 P)

a) Erweitern Sie bitte die angegebene Implementierung der binären Suchbäume (s. `http://www.inf.fu-berlin.de/lehre/WS01/ALP3/uebungen/u11/BinSTree.java`) um eine interne Klasse `LevelOrderIterator` mit den folgenden Eigenschaften:

```
class LevelOrderIterator implements java.util.Iterator {
  ...
  public LevelOrderIterator() { ... }
  public boolean hasNext() { ... }
  public Comparable next() { ... }
  public void remove() { ... }
}
```

Schreiben Sie eine Testklasse, die einen Baum erzeugt und mit Hilfe der Methoden der Klasse `LevelOrderIterator` diesen Baum durchläuft.

b) Implementieren Sie eine Funktion `postOrderIteration`, die das Durchlaufen eines binären Suchbaumes ermöglicht.

c) Implementieren Sie eine iterative Funktion `boolean delete(Comparable el)` und bestimmen Sie die Laufzeit dieser Funktion.