

7. Übungsaufgaben zur LV Algorithmen & Datenstrukturen

Abgabetermin: Mi, 12.05.04

1. Aufgabe:

- Implementation der Methode *Hoehe()* der Klasse *AVLbaum* (rekursiv):

```
public int Hoehe (AVLKnoten k) {  
    if (k==null) return 0;  
    int hLS=Hoehe (k.lSohn) , hRS=Hoehe (k.rSohn) ;  
    if (hLS>hRS)  
        return hLS+1;  
    return hRS+1;  
}
```

2. Aufgabe:

- Implementation der Methode *EtagOrd()* der Klasse *AVLbaum* (iterativ):

```
public int [] EtagOrd (String s, AVLKnoten start) {  
    int erg [] = new int [2];  
    AVLKnoten tmp = start;  
    while (tmp != null && !tmp.getSchluessel (). equals (s)) {  
        if (tmp.getSchluessel (). compareTo (s) < 0)  
            { tmp = tmp.rSohn; erg [1] = erg [1] * 2 + 1; }  
        else  
            { tmp = tmp.lSohn; erg [1] = 2 * erg [1]; }  
        erg [0] ++;  
    }  
    if (tmp == null) return new int [] { -1, -1 };  
    return erg;  
}
```

3. Aufgabe:

- Beispielausgabe:

The screenshot shows a window titled "AVLbaum - Test" with a text input field containing "+24" and an "Ausführen" button. Below the input is a legend: "+S für einfüegen(S), -S für entfernen(S), ?S für suchen(S)". The main area is a 32x32 grid representing a binary search tree. The root node is at index 0 with value 16. Its left child is at index 1 (0) and its right child is at index 8 (22). Node 8 has a left child at index 4 (12) and a right child at index 20 (20). Node 4 has a left child at index 2 (6) and a right child at index 25 (25). Node 2 has a left child at index 1 (3) and a right child at index 23 (23). Node 1 has a left child at index 0 (0) and a right child at index 3 (5). Node 3 has a left child at index 0 (0) and a right child at index 7 (10). Node 7 has a left child at index 0 (0) and a right child at index 14 (14). Node 14 has a left child at index 0 (0) and a right child at index 18 (18). Node 18 has a left child at index 0 (0) and a right child at index 21 (21). Node 21 has a left child at index 0 (0) and a right child at index 27 (27). Node 27 has a left child at index 0 (0) and a right child at index 31 (31). Node 31 has a left child at index 0 (0) and a right child at index 30 (30). Node 30 has a left child at index 0 (0) and a right child at index 29 (29). Node 29 has a left child at index 0 (0) and a right child at index 28 (28). Node 28 has a left child at index 0 (0) and a right child at index 26 (26). Node 26 has a left child at index 0 (0) and a right child at index 24 (24). Node 24 has a left child at index 0 (0) and a right child at index 28 (28). The grid also shows balance factors in the second row of each node's children. For example, node 8 has a balance factor of 0, node 4 has -1, node 2 has 1, node 1 has 0, node 3 has 0, node 7 has 0, node 14 has 0, node 18 has 0, node 21 has 1, node 27 has 0, node 31 has 0, node 30 has 0, node 29 has 0, node 28 has 0, and node 26 has 0. An "Ende" button is at the bottom.

- Dialog_AVLBaum(auszugsweise):

```

public class Dialog_AVLBaum extends JDialog {
    String[] TabKopf = new String[] {
        "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16",
        "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31"};
    String[][] Tab = new String[15][32];
    JTable jTable1 = new JTable(Tab, TabKopf);
    //...

    void OKBtn_actionPerformed(ActionEvent e1) {
        int i; boolean ok=false;
        String S = textfeld.getText();
        JOptionPane jOptPan = new JOptionPane();
        if (S.length() > 0) {
            char c = S.charAt(0);
            S = S.substring(1);
            ElementtypS e=new ElementtypS(S);
            jTable1.clearSelection();
            switch (c) {
                case '+':
                    if(!baum.einfuegen(e)) jOptPan.showMessageDialog(this,
                        "Element '"+S+"' schon im AVLBaum vorhanden.");
                    else ok=true;
                    break;
                case '-':
                    if(!baum.entfernen(e.getSchluessel())) jOptPan.showMessageDialog(this,
                        "Element '"+S+"' konnte nicht entfernt werden!");
                    else jOptPan.showMessageDialog(this, "'"+S+"' wurde entfernt...");
                    break;
                case '?':
                    Elementtyp d=baum.suchen(e.getSchluessel());
                    if(d==null || !d.getSchluessel().equals(e.getSchluessel()))
                        jOptPan.showMessageDialog(this,
                            "Element '"+S+"' wurde nicht gefunden!");
                    else ok=true;
                    break;
            }
            if(ok && (c=='+' || c=='?')){
                int[] iKn=baum.EtagOrd(e.getSchluessel(),baum.Wurzel);
                jTable1.changeSelection(3*iKn[0],
                    iKn[1]*baum.ZweiHoch(baum.Hoehe(baum.Wurzel))/baum.ZweiHoch(iKn[0]+1),
                    false, false);
            }
            if(c=='+' || c=='-')
                baum.BaumAusgabe(jTable1,baum.Wurzel);
        }
    }
}

```