Nama : Andyan Yogawardhana

NIM : 21/482180/PA/21030

Kelas : KOMB1

Tugas 4 - AVL

1. Implementasi AVL

```java
package Tree;

public class AVL {
    public static void main(String[] args) {
        int[] data = {0,1,2,3,4,5,6,7,8,9};
        Tree tree = new Tree();

        for(int i = 0; i < data.length; i++) {
            tree.setRoot(tree.insert(tree.getRoot(), data[i]));
        }

        tree.printTree();
    }
}

class Node {
    private int data, height;
    private Node left, right;

    public Node(int d) {
        this.data = d;
        this.height = 1;
    }

    public int getData() {
        return this.data;
    }

    public int getHeight() {
        return this.height;
    }

    public void setHeight(int h) {
        this.height = h;
    }

    public Node getLeft() {
        return this.left;
```

```
39      }
40
41      public Node getRight() {
42          return this.right;
43      }
44
45      public void setLeft(Node n) {
46          this.left = n;
47      }
48
49      public void setRight(Node n) {
50          this.right = n;
51      }
52  }
53
54  class Tree {
55      private Node root;
56
57      public Node getRoot() {
58          return this.root;
59      }
60
61      public void setRoot(Node n) {
62          this.root = n;
63      }
64
65      public int height(Node n) {
66          return n == null ? 0 : n.getHeight();
67      }
68      public int max(int a, int b) {
69          return a > b ? a : b;
70      }
71
72      public int getBalance(Node n) {
73          return n == null ? 0 : (height(n.getLeft()) -
    height(n.getRight()));
74      }
75
76      public Node rotateLeft(Node x) {
77          Node y = x.getRight();
78          Node T2 = y.getLeft();
79
80          x.setRight(T2);
81          y.setLeft(x);
82
83          x.setHeight(1 + max(height(x.getLeft()),
    height(x.getRight())));
```

```
84          y.setHeight(1 + max(height(y.getLeft()),
    height(y.getRight())));
85
86          return y;
87      }
88
89      public Node rotateRight(Node y) {
90          Node x = y.getLeft();
91          Node T2 = x.getRight();
92
93          x.setRight(y);
94          y.setLeft(T2);
95
96          y.setHeight(1 + max(height(y.getLeft()),
    height(y.getRight())));
97          x.setHeight(1 + max(height(x.getLeft()),
    height(x.getRight())));
98
99          return x;
100           }
101
102           public Node insert(Node n, int d) {
103               if(n == null) {
104                   return new Node(d);
105               }
106
107               if(d < n.getData()) {
108                   n.setLeft(insert(n.getLeft(), d));
109               }
110               else if(d > n.getData()) {
111                   n.setRight(insert(n.getRight(), d));
112               }
113               else {
114                   return n;
115               }
116
117               n.setHeight(1 + max(height(n.getLeft()),
    height(n.getRight())));
118
119               int balance = getBalance(n);
120
121               if(balance > 1 && d < n.getLeft().getData()) {
122                   return rotateRight(n);
123               }
124
125               if(balance < -1 && d > n.getRight().getData()) {
126                   return rotateLeft(n);
127               }
```

```
128
129             if(balance > 1 && d > n.getLeft().getData()) {
130                 n.setLeft(rotateLeft(n.getLeft()));
131                 return rotateRight(n);
132             }
133
134             if(balance < -1 && d < n.getRight().getData()) {
135                 n.setRight(rotateRight(n.getRight()));
136                 return rotateLeft(n);
137             }
138
139             return n;
140         }
141
142         public void printTree() {
143             System.out.println("Tree Graph: ");
144             printTree("", root, false);
145         }
146
147         public void printTree(String prefix, Node n, boolean
    isLeft) {
148             if (n != null) {
149                 printTree(prefix + "      ", n.getRight(),
    false);
150                 System.out.println (prefix + ("|----[") +
    n.getData() + "]");
151                 printTree(prefix + "      ", n.getLeft(),
    true);
152             }
153         }
154     }
155
```

Output Terminal

```
Tree Graph:
                    |----[9]
               |----[8]
          |----[7]
                    |----[6]
               |----[5]
                    |----[4]
|----[3]
               |----[2]
          |----[1]
               |----[0]
```

2. Cek AVL atau bukan

```java
package Tree;

public class AVLCheck {
    public static void main(String[] args) {
        int[] data = {6,8,9,4,7,2,1,5,3};

        BSTree tree = new BSTree();

        for(int i = 0; i < data.length; i++) {
            Node node = new Node(data[i]);
            tree.addNode(node);
        }

        tree.checkAVL();

        int[] data2 = {5,2,6,3,1,4,8,7,9};

        BSTree tree2 = new BSTree();

        for(int i = 0; i < data2.length; i++) {
            Node node2 = new Node(data2[i]);
            tree2.addNode(node2);
        }

        tree2.checkAVL();
    }
}

class Node {
    private int data, height;
    private Node left, right;

    public Node(int d) {
        this.data = d;
        this.height = 1;
    }

    public int getData() {
        return this.data;
    }

    public int getHeight() {
        return this.height;
    }

    public void setHeight(int h) {
```

```
47          this.height = h;
48      }
49
50      public Node getLeft() {
51          return this.left;
52      }
53
54      public Node getRight() {
55          return this.right;
56      }
57
58      public void setLeft(Node n) {
59          this.left = n;
60      }
61
62      public void setRight(Node n) {
63          this.right = n;
64      }
65  }
66
67  class BSTree {
68      private Node root;
69      private boolean isAVL = true;
70
71      public Node getRoot() {
72          return this.root;
73      }
74
75      public void addNode(Node n) {
76          if(root == null) {
77              root = n;
78          }
79          else {
80              insertNode(root, n);
81          }
82      }
83
84      public void insertNode(Node parent, Node n) {
85          if(parent.getData() > n.getData()) {
86              if(parent.getLeft() == null) {
87                  parent.setLeft(n);
88              }
89              else {
90                  insertNode(parent.getLeft(), n);
91              }
92          }
93          else {
94              if(parent.getRight() == null) {
```

```
95                  parent.setRight(n);
96              }
97              else {
98                  insertNode(parent.getRight(), n);
99              }
100               }
101         }
102
103         public int height(Node n) {
104             if(n == null) {
105                 return -1;
106             }
107             else {
108                 int leftHeight = height(n.getLeft());
109                 int rightHeight = height(n.getRight());
110
111                 return leftHeight > rightHeight ? leftHeight -
  1 : rightHeight - 1;
112             }
113
114         }
115
116         public int max(int a, int b) {
117             return a > b ? a : b;
118         }
119
120         public Node rotateLeft(Node x) {
121             Node y = x.getRight();
122             Node T2 = y.getLeft();
123
124             y.setLeft(x);
125             x.setRight(T2);
126
127             x.setHeight(1 + max(height(x.getLeft()),
  height(x.getRight())));
128             y.setHeight(1 + max(height(y.getLeft()),
  height(y.getRight())));
129
130             return y;
131         }
132
133         public Node rotateRight(Node y) {
134             Node x = y.getLeft();
135             Node T2 = x.getRight();
136
137             x.setRight(y);
138             y.setLeft(T2);
139
```

```
140             y.setHeight(1 + max(height(y.getLeft()),
  height(y.getRight())));
141             x.setHeight(1 + max(height(x.getLeft()),
  height(x.getRight())));
142 
143             return x;
144         }
145 
146         public int getBalance(Node n) {
147             return n == null ? 0 : height(n.getLeft()) -
  height(n.getRight());
148         }
149 
150         public Node insert(Node n, int d) {
151             if(n == null) {
152                 return new Node(d);
153             }
154 
155             if(d < n.getData()) {
156                 n.setLeft(insert(n.getLeft(), d));
157             }
158             else if(d > n.getData()) {
159                 n.setRight(insert(n.getRight(), d));
160             }
161             else {
162                 return n;
163             }
164 
165             n.setHeight(1 + max(height(n.getLeft()),
  height(n.getRight())));
166 
167             int balance = getBalance(n);
168 
169             if(balance > 1 && d < n.getLeft().getData()) {
170                 return rotateRight(n);
171             }
172 
173             if(balance < -1 && d > n.getRight().getData()) {
174                 return rotateLeft(n);
175             }
176 
177             if(balance > 1 && d > n.getLeft().getData()) {
178                 n.setLeft(rotateLeft(n.getLeft()));
179                 return rotateRight(n);
180             }
181 
182             if(balance < -1 && d < n.getRight().getData()) {
183                 n.setRight(rotateRight(n.getRight()));
```

```
184                 return rotateLeft(n);
185             }
186 
187             return n;
188         }
189 
190         public boolean isAVL(Node n) {
191             if(n.getLeft() != null) {
192                 isAVL(n.getLeft());
193             }
194             if(n.getRight() != null) {
195                 isAVL(n.getRight());
196             }
197 
198             if(getBalance(n) > 1 || getBalance(n) < -1) {
199                 isAVL = false;
200             }
201 
202             return isAVL;
203         }
204 
205         public void checkAVL() {
206             boolean isAVL = this.isAVL(root);
207             printTree();
208             printLevelOrder();
209             System.out.println();
210 
211             if(isAVL) {
212                 System.out.println("This is an AVL Tree");
213             }
214             else {
215                 System.out.println("This is not an AVL Tree");
216             }
217             System.out.println("\n- - - - - - - - - - - - - -
   \n");
218         }
219 
220         public void printTree() {
221             System.out.println("Tree Graph:");
222             printTreeFunction("", root, false);
223         }
224 
225         public void printTreeFunction(String prefix, Node n,
   boolean isLeft) {
226             if (n != null) {
227                 printTreeFunction(prefix + "      ",
   n.getRight(), false);
```

```
228                 System.out.println (prefix + ("|----[") +
   n.getData() + "]");
229                 printTreeFunction(prefix + "      ",
   n.getLeft(), true);
230             }
231         }
232 
233         public void printLevelOrder() {
234             System.out.println("\nTree Level Order:");
235             for(int i = 1; i <= 99; i++) {
236                 levelOrderFunction(root, i);
237             }
238             System.out.println();
239         }
240 
241         public void levelOrderFunction(Node node, int level) {
242             if(node != null) {
243                 if(level == 1){
244                     System.out.print(node.getData() + " ");
245                 }
246                 else if(level > 1) {
247                     levelOrderFunction(node.getLeft(), level -
   1);
248                     levelOrderFunction(node.getRight(), level
   - 1);
249                 }
250             }
251         }
252     }
```

Output Terminal

```
Tree Graph:
            |----[9]
      |----[8]
            |----[7]
|----[6]
            |----[5]
      |----[4]
                  |----[3]
            |----[2]
                  |----[1]

Tree Level Order:
6 4 8 2 5 7 9 1 3

This is an AVL Tree

- - - - - - - - - - - - -

Tree Graph:
                  |----[9]
            |----[8]
                  |----[7]
      |----[6]
|----[5]
                  |----[4]
            |----[3]
      |----[2]
            |----[1]

Tree Level Order:
5 2 6 1 3 8 4 7 9

This is not an AVL Tree

- - - - - - - - - - - - -
```