

Gebze Technical University  
Computer Engineering

CSE 222  
2017 Spring

HOMEWORK 8 REPORT

VAKHID BETRAKHMADOV  
141044086

Course Assistant:

## 1. Problem Solutions Approach

In order to complete AVL Tree class as required, we had to provide missing methods for removal. In the book's source code found online, the author uses strategy where he manipulates both increase and decrease flags, in order to keep the tree balanced while removing elements from the tree, and provides extra helper methods ( `rebalanceLeftR`, `rebalanceRightL` ). This source code turned out to be buggy, and didn't work as expected. I decided to accomplish all the job using just decrease flag, and only 2 helper methods ( `rebalanceLeft` and `rebalanceRight` ) which are symmetric to each other.

While removing an element from the tree that causes the critical imbalance to occur, we might face two additional situations, when left-right ( right-left ) child is balanced or left ( right ) child is balanced. In the first case after the double rotation is performed, all balances are zero, so we add one extra condition to the `rebalanceLeft` and `rebalanceRight` methods. In the second case, after the single rotation is performed in `rebalanceLeft` method, old left child becomes right heavy, and the old root becomes left heavy ( in the `rebalanceLeft` method old right child now is left heavy and the old root is now right heavy ) .

Delete method is almost the same as in the Binary Search Tree class, except for that we use extra flag 'decrease' to track changes in the tree, and rebalance it on run when necessary.

After each return from the recursive call, we check if the decrease flag is true, if so, that means that the height of the sub-tree has changed, and we decrease ( increase ) local root's balance. If now balance of the local root is 0 we leave decrease flag true, if it has changed to 1 or -1 we set it false, if it has caused critical imbalance ( -2 or +2 ) we set decrease to false and rebalance the sub-tree starting from the local root. We check again after rebalancing if the local root is now balanced ( 0 ) , if so we set decrease flag to true again and return.

## 2. Test Cases

I have extensively tested the program, found and solved all bugs I have noticed.

As a test case I provided a piece of code which builds an AVL Tree from scratch with 20 random elements in the range from 1 to 50. It prints the state of the tree to the screen after each insertion. Then it removes 20 random elements from the same range and prints the tree state to the screen after every removal.

### 3. Running and Results

Project: -/Desktop/Semester 4/Data Structures/HW/HW6/Project - [Project] - ~/Desktop/Semester 4/Data Structures/HW/HW6/Project/src/com/restermans/AVLTree.java - IntelliJ IDEA 2017.1.2

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Project src com restermans AVLTree

Run Main

```

↑ Inserting: 29
My tree:
0: 33
| 0: 17
| | 1: 3
| | | 0: 2
| | | | null
| | | -1: 14
| | | 0: 7
| | | | null
| | | | null
| | 1: 21
| | | 0: 20
| | | | null
| | | | null
| | | 0: 29
| | | | 0: 25
| | | | | null
| | | | | null
| | | | 0: 31
| | | | | null
| | | | | null
| | 1: 45
| | | 0: 40
| | | | 0: 37
| | | | | null
| | | | | null
| | | | 0: 43
| | | | | null
| | | | | null
| | | 1: 47
| | | | 0: 46
| | | | | null
| | | | | null
| | | 1: 48
| | | | null
| | | | 0: 49
| | | | | null
| | | | | null

```

Removing 47 from the myTree: true

My tree:

```

-1: 33
| 0: 17
| | 1: 3
| | | 0: 2
| | | | null
| | | | null
| | | -1: 14
| | | 0: 7

```

Run TODO Terminal

All files are up-to-date (moments ago)

Vakhid Betrakhmadov 11:34:1 CRLF UTF-8 4:43 PM

[illegible]

The screenshot shows the IntelliJ IDEA 2017.1.2 IDE. The main editor displays a Java file named `AVLTree.java` with the following code:

```

import java.util.*;

public class AVLTree {
    static class Node {
        int data;
        Node left;
        Node right;
    }

    static Node root;

    static void insert(int data) {
        root = insertRec(root, data);
    }

    static Node insertRec(Node root, int data) {
        if (root == null) {
            root = new Node();
            root.data = data;
            return root;
        }
        if (data < root.data)
            root.left = insertRec(root.left, data);
        else if (data > root.data)
            root.right = insertRec(root.right, data);
        return root;
    }

    static void remove(int data) {
        root = removeRec(root, data);
    }

    static Node removeRec(Node root, int data) {
        if (root == null)
            return null;
        if (data < root.data)
            root.left = removeRec(root.left, data);
        else if (data > root.data)
            root.right = removeRec(root.right, data);
        else {
            // Node to be deleted
            if (root.left == null || root.right == null)
                return root.right != null ? root.right : root.left;
            else {
                // Node has both left and right children
                Node succ = findMin(root.right);
                root.data = succ.data;
                root.right = removeRec(root.right, succ.data);
            }
        }
        return root;
    }

    static Node findMin(Node root) {
        while (root.left != null)
            root = root.left;
        return root;
    }

    static void printTree() {
        printRec(root, 0);
    }

    static void printRec(Node root, int level) {
        if (root == null)
            return;
        printRec(root.left, level + 1);
        System.out.printf("%-10s", level == 0 ? "My tree:" : "");
        System.out.printf("%-10s", level == 1 ? "-l: 33" : "");
        System.out.printf("%-10s", level == 2 ? "0: 17" : "");
        System.out.printf("%-10s", level == 3 ? "1: 3" : "");
        System.out.printf("%-10s", level == 4 ? "0: 2" : "");
        System.out.printf("%-10s", level == 5 ? "null" : "");
        System.out.printf("%-10s", level == 6 ? "null" : "");
        System.out.printf("%-10s", level == 7 ? "-l: 14" : "");
        System.out.printf("%-10s", level == 8 ? "0: 7" : "");
        System.out.printf("%-10s", level == 9 ? "null" : "");
        System.out.printf("%-10s", level == 10 ? "null" : "");
        System.out.printf("%-10s", level == 11 ? "null" : "");
        System.out.printf("%-10s", level == 12 ? "1: 21" : "");
        System.out.printf("%-10s", level == 13 ? "0: 26" : "");
        System.out.printf("%-10s", level == 14 ? "null" : "");
        System.out.printf("%-10s", level == 15 ? "null" : "");
        System.out.printf("%-10s", level == 16 ? "0: 29" : "");
        System.out.printf("%-10s", level == 17 ? "0: 25" : "");
        System.out.printf("%-10s", level == 18 ? "null" : "");
        System.out.printf("%-10s", level == 19 ? "null" : "");
        System.out.printf("%-10s", level == 20 ? "0: 31" : "");
        System.out.printf("%-10s", level == 21 ? "null" : "");
        System.out.printf("%-10s", level == 22 ? "null" : "");
        System.out.printf("%-10s", level == 23 ? "0: 45" : "");
        System.out.printf("%-10s", level == 24 ? "0: 40" : "");
        System.out.printf("%-10s", level == 25 ? "0: 37" : "");
        System.out.printf("%-10s", level == 26 ? "null" : "");
        System.out.printf("%-10s", level == 27 ? "null" : "");
        System.out.printf("%-10s", level == 28 ? "0: 43" : "");
        System.out.printf("%-10s", level == 29 ? "null" : "");
        System.out.printf("%-10s", level == 30 ? "null" : "");
        System.out.printf("%-10s", level == 31 ? "-l: 48" : "");
        System.out.printf("%-10s", level == 32 ? "0: 46" : "");
        System.out.printf("%-10s", level == 33 ? "null" : "");
        System.out.printf("%-10s", level == 34 ? "null" : "");
        System.out.printf("%-10s", level == 35 ? "null" : "");
    }

    public static void main(String[] args) {
        insert(33);
        insert(17);
        insert(3);
        insert(2);
        insert(14);
        insert(7);
        insert(21);
        insert(26);
        insert(29);
        insert(25);
        insert(31);
        insert(45);
        insert(40);
        insert(37);
        insert(43);
        insert(48);
        insert(46);
        remove(49);
        printTree();
        remove(21);
        printTree();
    }
}

```

The console output shows the execution of the program. It first prints the tree structure after inserting all nodes, then removes node 49, and finally removes node 21. The tree structure is printed before and after each removal operation.

```

Removing 49 from the myTree: true
My tree:
-l: 33
0: 17
1: 3
0: 2
null
null
-l: 14
0: 7
null
null
1: 21
0: 26
null
null
0: 29
0: 25
null
null
0: 31
null
null
0: 45
0: 40
0: 37
null
null
0: 43
null
null
-l: 48
0: 46
null
null
null

Removing 21 from the myTree: true
My tree:
-l: 33
0: 17
1: 3
0: 2
null
null
-l: 14
0: 7
null
null
null

```