Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

AHMET YASIR NACAK
161044042

Course Assistant: Fatma Nur Esirci

1 Worst RedBlack Tree

Question 1 wants us to create the worst RedBlack tree with the height of 6. For this, we need to find a solution and simulate it in our main method.

1.1 Problem Solution Approach

The worst RedBlack tree is the tree that has the minimum number of nodes for a given height. The ideal (best) number of nodes in a RedBlack tree for height n is 2^(n+1)-1. Since height is the maximum distance from a given node to a leaf, we add 1 to n. In this problem, I found out that if we add 22 numbers that are in the smallest to largest order to our tree, we get the height value of 6 and there is not a smaller number of nodes that can achieve that. So, I created a number randomizer that generates a value between 1 and 5000 (this can go up) and added this number to the tree. After that, I added the following 21 numbers to the tree and left with a tree with height of 6 and node count of 22.

Pseudocode:

```
t = new RedBlack tree
num = randomly generated number between 1 – 5000
add num to t
loop from num to num+22
    add each value to t
print t
```

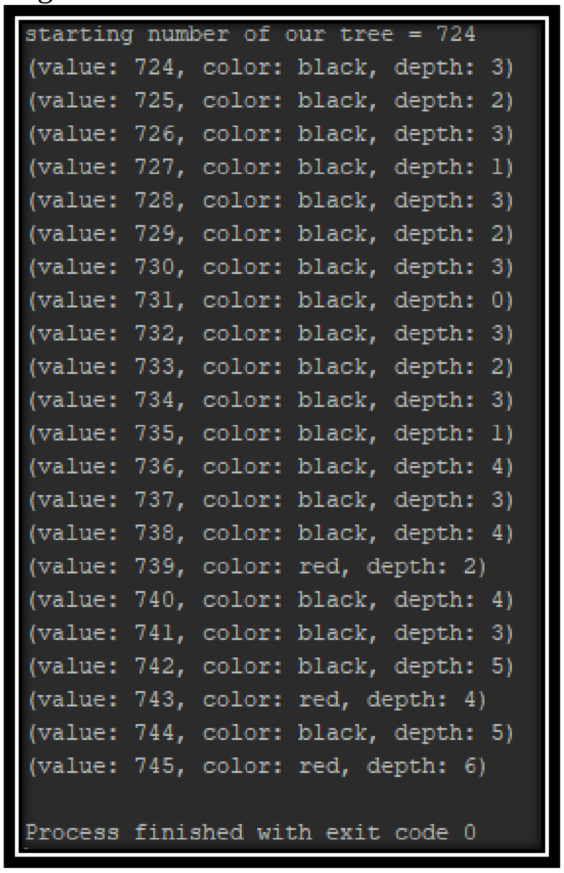
1.2 Test Cases

First RedBlack Tree is the one that starts from 724. The last value is 724+21 = 746. All the values get added to the tree from smallest to largest. In this case, the height of 746 becomes 6 and the tree is the worst RedBlack Tree.

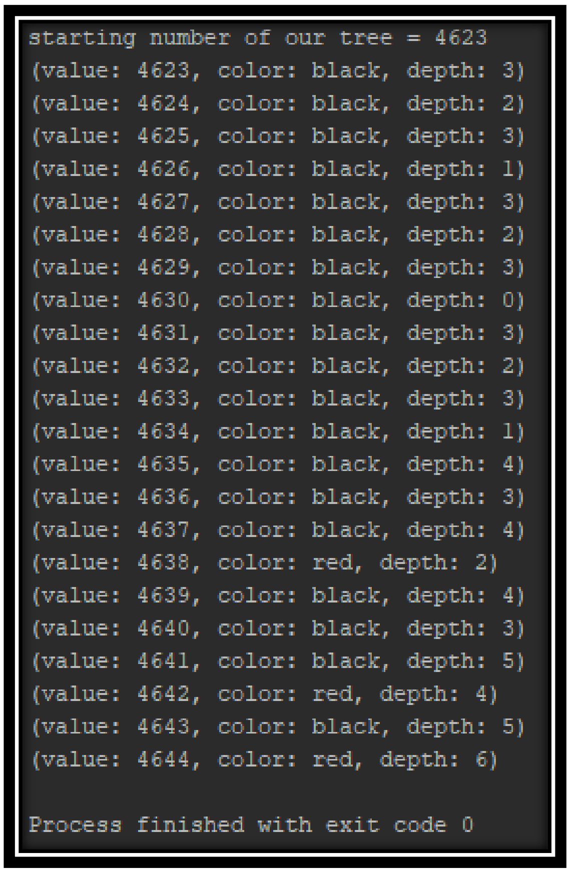
Second RedBlack Tree is the one that starts from 4623. The last value is 4644 in that case. If we add the values starting from 4623 and ending at 4644 to our tree, we get a RedBlack tree with the height of 6 and node count of 22. This also is the worst RedBlack Tree possible.

1.3 Running Commands and Results

Show that test case results using screenshots.



```
starting number of our tree = 724
(value: 724, color: black, depth: 3)
(value: 725, color: black, depth: 2)
(value: 726, color: black, depth: 3)
(value: 727, color: black, depth: 1)
(value: 728, color: black, depth: 3)
(value: 729, color: black, depth: 2)
(value: 730, color: black, depth: 3)
(value: 731, color: black, depth: 0)
(value: 732, color: black, depth: 3)
(value: 733, color: black, depth: 2)
(value: 734, color: black, depth: 3)
(value: 735, color: black, depth: 1)
(value: 736, color: black, depth: 4)
(value: 737, color: black, depth: 3)
(value: 738, color: black, depth: 4)
(value: 739, color: red, depth: 2)
(value: 740, color: black, depth: 4)
(value: 741, color: black, depth: 3)
(value: 742, color: black, depth: 5)
(value: 743, color: red, depth: 4)
(value: 744, color: black, depth: 5)
(value: 745, color: red, depth: 6)

Process finished with exit code 0
```



```
starting number of our tree = 4623
(value: 4623, color: black, depth: 3)
(value: 4624, color: black, depth: 2)
(value: 4625, color: black, depth: 3)
(value: 4626, color: black, depth: 1)
(value: 4627, color: black, depth: 3)
(value: 4628, color: black, depth: 2)
(value: 4629, color: black, depth: 3)
(value: 4630, color: black, depth: 0)
(value: 4631, color: black, depth: 3)
(value: 4632, color: black, depth: 2)
(value: 4633, color: black, depth: 3)
(value: 4634, color: black, depth: 1)
(value: 4635, color: black, depth: 4)
(value: 4636, color: black, depth: 3)
(value: 4637, color: black, depth: 4)
(value: 4638, color: red, depth: 2)
(value: 4639, color: black, depth: 4)
(value: 4640, color: black, depth: 3)
(value: 4641, color: black, depth: 5)
(value: 4642, color: red, depth: 4)
(value: 4643, color: black, depth: 5)
(value: 4644, color: red, depth: 6)

Process finished with exit code 0
```

2 binarySearch method

Question 2 wants us to write the missing binary searching method for the BTree class. This method helps the insert method to find the correct way to go. Since a node of BTree has order-1 elements, it has order connections to its children. Each connection shows a node that has smaller than elements after and greater than before it. Because of that, the binary search method needs to find where to advance in the tree to insert the given node. The insertion method takes care of the rest of it.

2.1 Problem Solution Approach

As the book explains, this binary search needs to return a valid index value no matter what. This means that if the item is in the array, we need to return the position of it, otherwise we need to return the location that item needs to go to in the given array. For example:

4, 12, 15, 37, 41

If we are searching for 12 in this array, the method returns 1. If we are looking for a value between 4 and 12, this method also returns 1 because if that value was in this array, it would be in the position with index of 1. Another example would be a value lesser than 4. The search returns 0 for that because any value lesser than 4 goes to that position. Any value greater than 41 returns 6, etc.

Pseudocode:

```
if the target value is greater than the last element of the array
    return size of the array
do a regular binary search in the given array
    return the value if target found
iterate through the array to find a value greater than target
    return the location before the found value
```

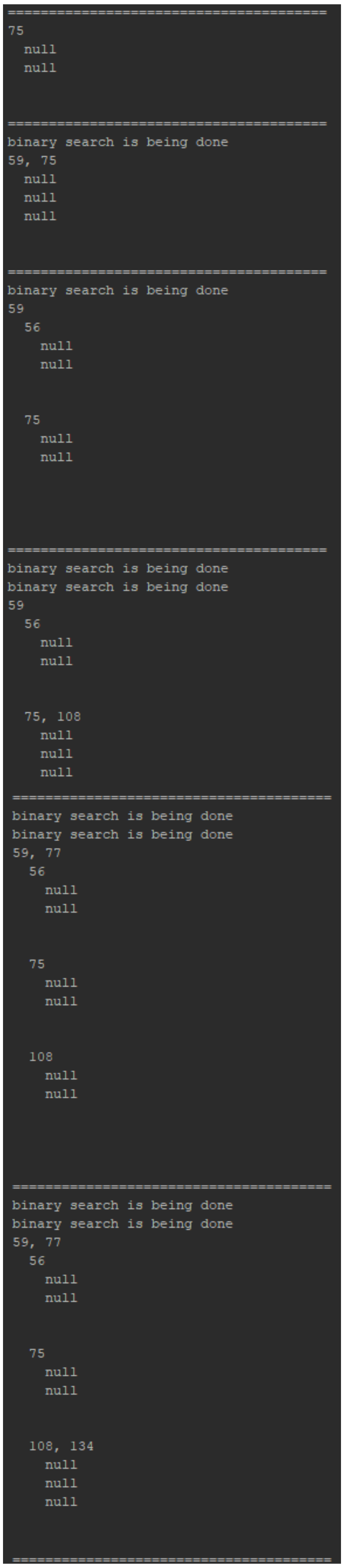
2.2 Test Cases

The test case in this part is consists of creating a BTree and tracking the method calls for binarySearch. I inserted the following numbers in the tree:

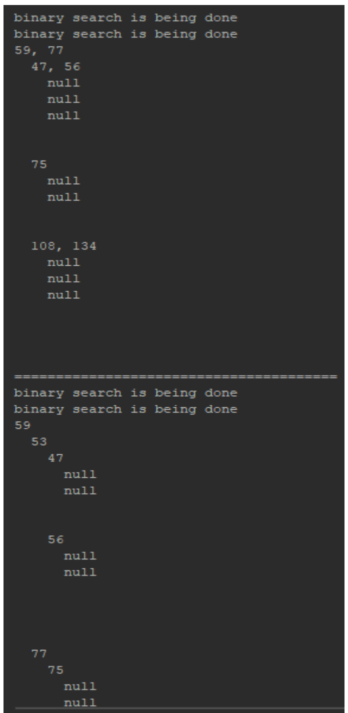
75, 59, 56, 108, 77, 134, 47, 53, 106, 127

After each addition, I print out the tree and tracked the number of binarySearch calls. This resulted in the number of calls being the depth of the tree. For example, if the depth is 3, the search method gets called 3 times and determines which path to follow after each search and stops when a leaf is reached. More detailed info about this can be showed in part 2.3.

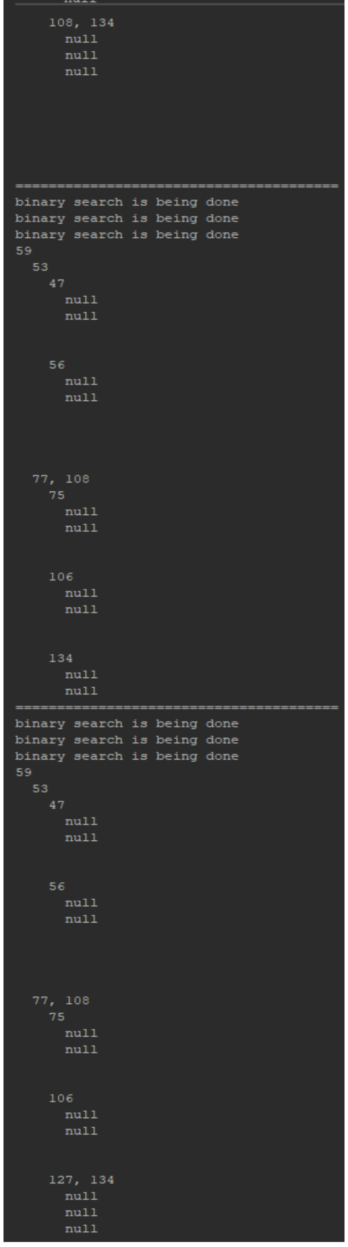
2.3 Running Commands and Results



```
=======================================
75
  null
  null


=======================================
binary search is being done
59, 75
  null
  null
  null


=======================================
binary search is being done
59
  56
    null
    null


  75
    null
    null




=======================================
binary search is being done
binary search is being done
59
  56
    null
    null


  75, 108
    null
    null
    null
=======================================
binary search is being done
binary search is being done
59, 77
  56
    null
    null


  75
    null
    null


  108
    null
    null




=======================================
binary search is being done
binary search is being done
59, 77
  56
    null
    null


  75
    null
    null


  108, 134
    null
    null
    null


=======================================
```



```
binary search is being done
binary search is being done
59, 77
  47, 56
    null
    null
    null


  75
    null
    null


  108, 134
    null
    null
    null




========================================
binary search is being done
binary search is being done
59
  53
    47
      null
      null


    56
      null
      null




  77
    75
      null
      null
```



```
    108, 134
      null
      null
      null






========================================
binary search is being done
binary search is being done
binary search is being done
59
  53
    47
      null
      null


    56
      null
      null




  77, 108
    75
      null
      null


    106
      null
      null


    134
      null
      null
========================================
binary search is being done
binary search is being done
binary search is being done
59
  53
    47
      null
      null


    56
      null
      null




  77, 108
    75
      null
      null


    106
      null
      null


    127, 134
      null
      null
      null
```

3 Project 9.5 in book

Question 3 wants us to complete the missing methods in the AVL Tree that the book provided. These methods are: delete, rebalanceRight, rebalanceLeft, rebalanceRightLeft, rebalanceLeftRight and incrementBalance. The delete method contains a user (public) version and the real deletion method that runs recursively that utilizes the rebalanceRightLeft and rebalanceLeftRight. These rebalance are used for special cases where the subtree of the subtree of root is imbalanced. This question also wants us to create a constructor that takes a Binary Search Tree and make it into an AVL Tree if it is balanced for an AVL Tree.

3.1 Problem Solution Approach

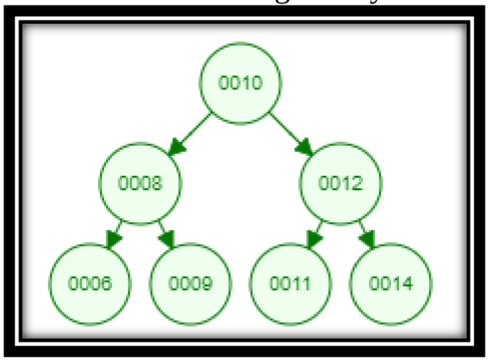
For the constructor, I checked the right and left heights of the given Binary Search Tree. I created a recursive method that finds the height of a node. If the absolute value of the difference of those heights are lesser or equal to 1, we can say that the given Binary Search Tree is a valid AVL Tree. After that, I traversed the given tree in level order and added the found values to an ArrayList. After that, I added this array to the tree from start to end. This addition does not change the balance of the tree and transfers the values correctly.

3.2 Test Cases

I firstly created two binary search trees. One of them was a valid AVL tree and the other was not. After that I sent them to the constructor of the AVL Tree. One of them successfully created an AVL Tree out of the given tree and other didn't. After that, I created another AVL Tree and added elements to them and removed elements from them. I tracked down the method calls and made sure that the methods run correctly.

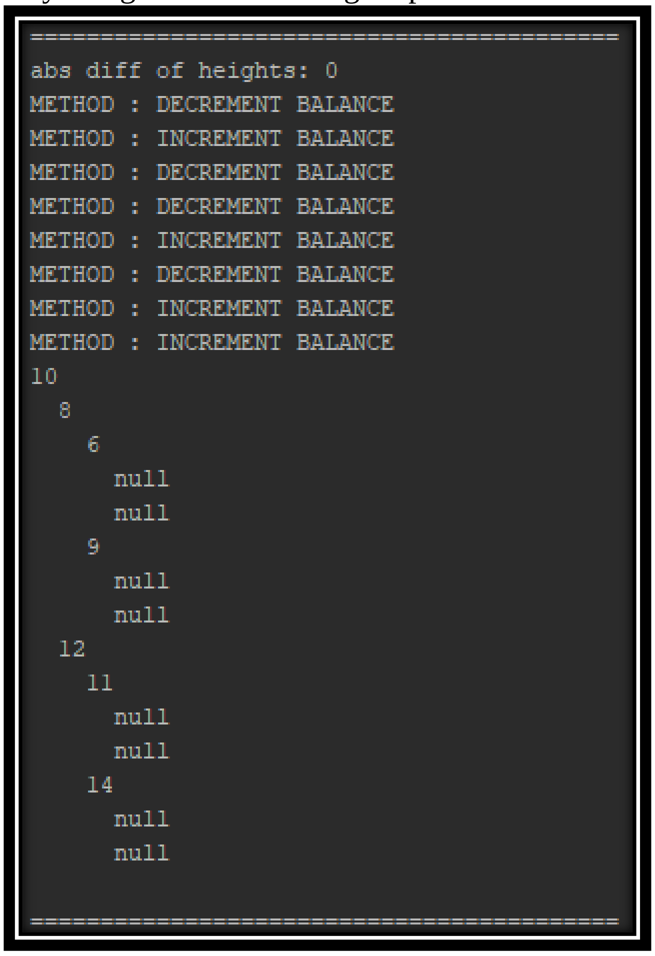
3.3 Running Commands and Results

I firstly tried to run the constructor with the following Binary Search Tree:



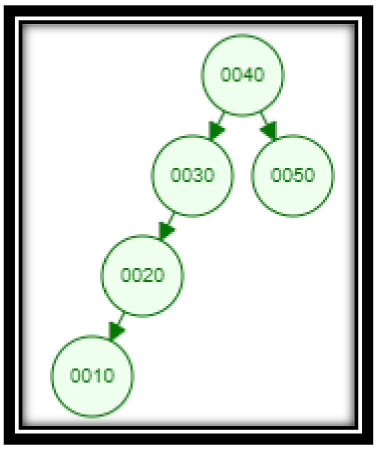
This is a balanced binary search tree and can be converted to an AVL Tree

The constructor ran correctly and gave the following output:

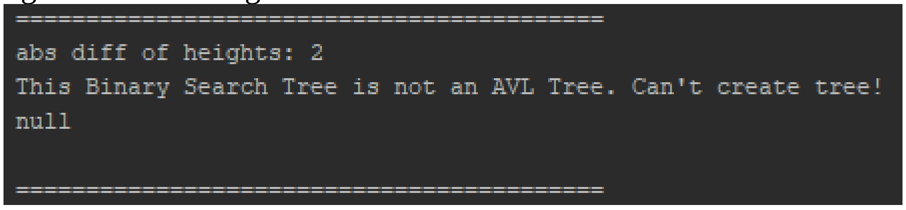


```
==========================================
abs diff of heights: 0
METHOD : DECREMENT BALANCE
METHOD : INCREMENT BALANCE
METHOD : DECREMENT BALANCE
METHOD : DECREMENT BALANCE
METHOD : INCREMENT BALANCE
METHOD : DECREMENT BALANCE
METHOD : INCREMENT BALANCE
METHOD : INCREMENT BALANCE
10
  8
    6
      null
      null
    9
      null
      null
  12
    11
      null
      null
    14
      null
      null

==========================================
```

After that, I gave the following tree:



The constructor gave the following error:

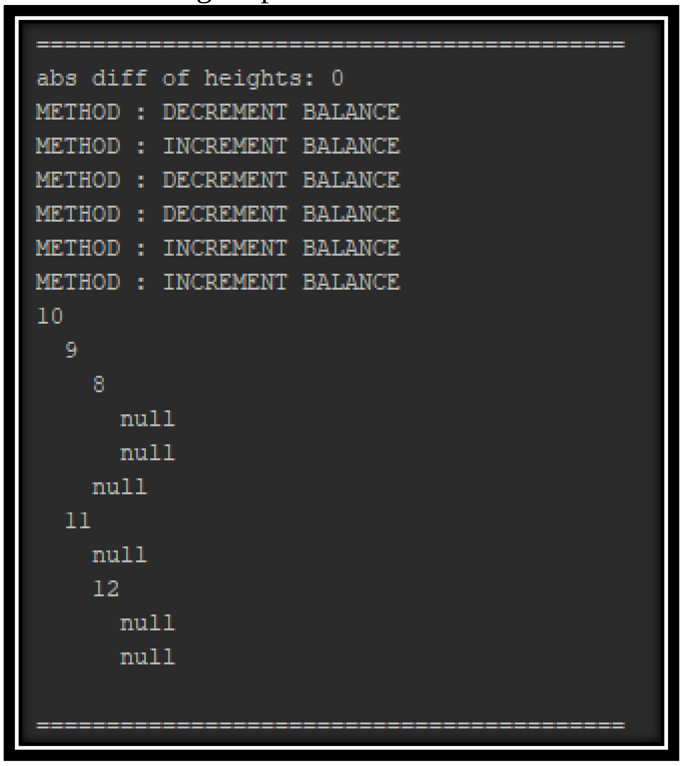


```
==========================================
abs diff of heights: 2
This Binary Search Tree is not an AVL Tree. Can't create tree!
null

==========================================
```

After that, I created a valid AVL Tree with adding following values in order:

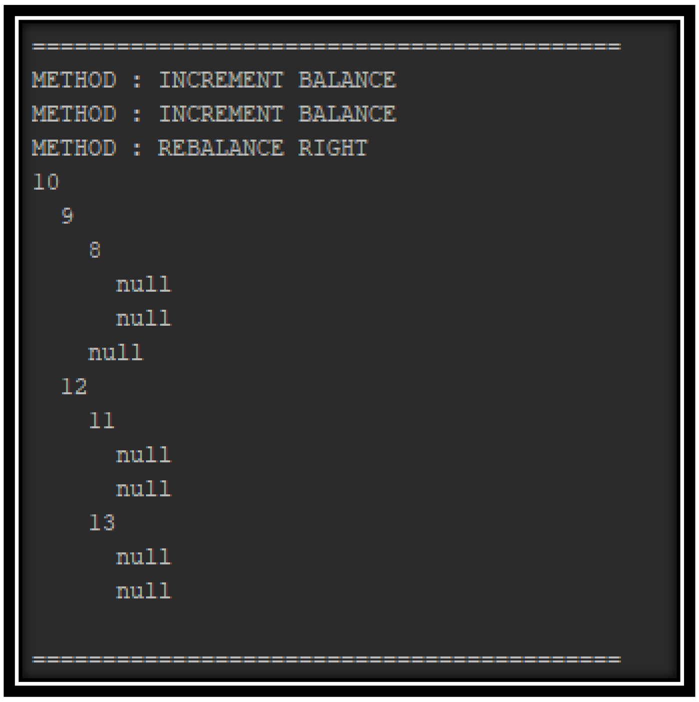
10, 9, 11, 8, 12

The creation process gave the following output:



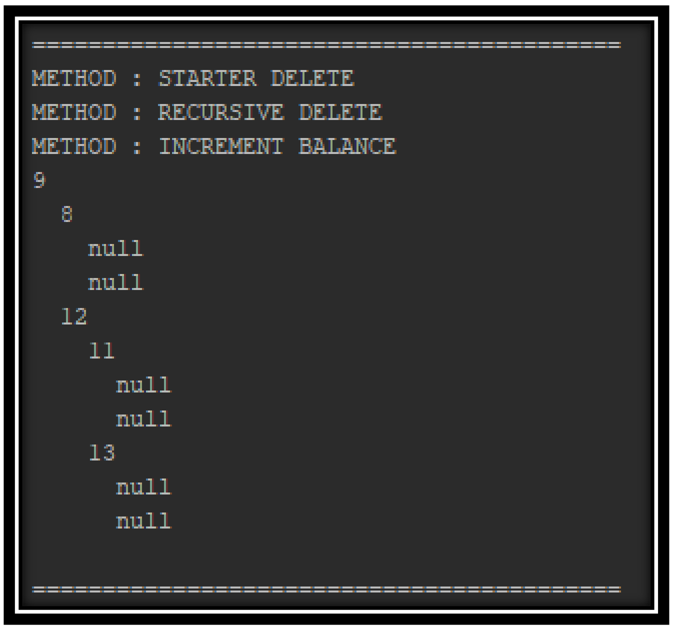
```
==========================================
abs diff of heights: 0
METHOD : DECREMENT BALANCE
METHOD : INCREMENT BALANCE
METHOD : DECREMENT BALANCE
METHOD : DECREMENT BALANCE
METHOD : INCREMENT BALANCE
METHOD : INCREMENT BALANCE
10
  9
    8
      null
      null
    null
  11
    null
    12
      null
      null

==========================================
```

Then I added 13:



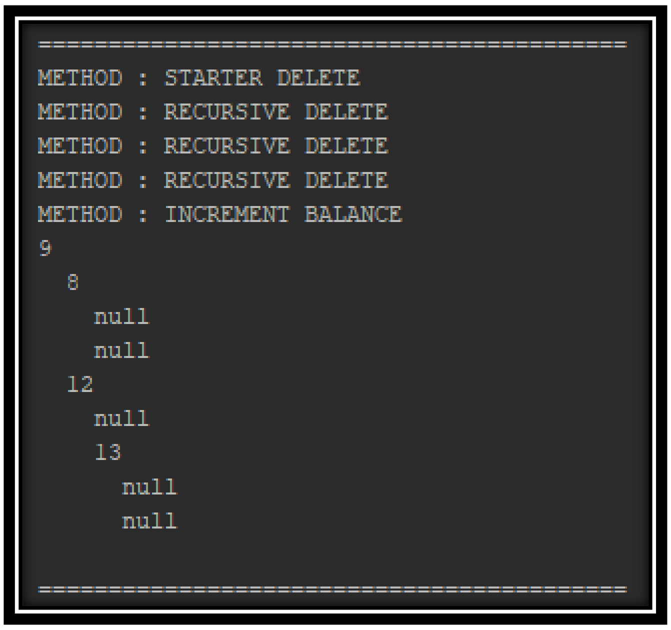
```
==========================================
METHOD : INCREMENT BALANCE
METHOD : INCREMENT BALANCE
METHOD : REBALANCE RIGHT
10
  9
    8
      null
      null
    null
  12
    11
      null
      null
    13
      null
      null

==========================================
```

I removed 10:



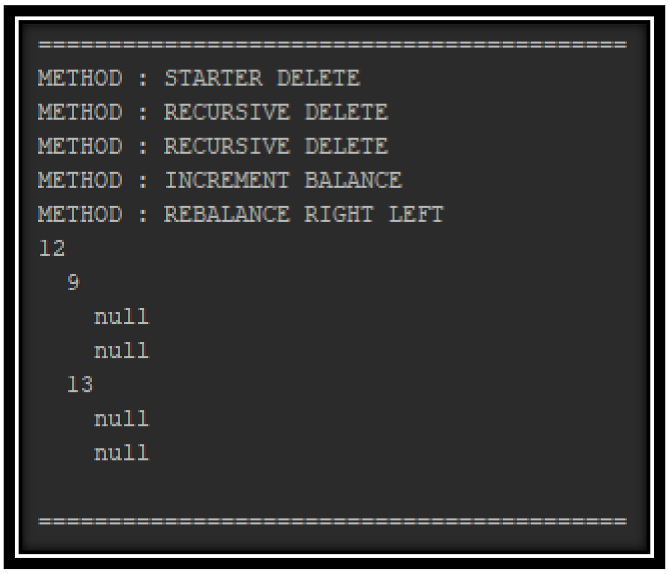
```
===========================================
METHOD : STARTER DELETE
METHOD : RECURSIVE DELETE
METHOD : INCREMENT BALANCE
9
  8
    null
    null
  12
    11
      null
      null
    13
      null
      null

===========================================
```

I removed 11:



```
===========================================
METHOD : STARTER DELETE
METHOD : RECURSIVE DELETE
METHOD : RECURSIVE DELETE
METHOD : RECURSIVE DELETE
METHOD : INCREMENT BALANCE
9
  8
    null
    null
  12
    null
    13
      null
      null

===========================================
```

I removed 13:



```
===========================================
METHOD : STARTER DELETE
METHOD : RECURSIVE DELETE
METHOD : RECURSIVE DELETE
METHOD : INCREMENT BALANCE
METHOD : REBALANCE RIGHT LEFT
12
  9
    null
    null
  13
    null
    null

===========================================
```