Implementing a Red Black Tree
Report #09

By
Eduardo Castro

CS 303 Algorithms and Data Structures

October 22, 2014

1. Problem Specification

The goal of this assignment is to implement a Red Black Tree and to use it.

2. Program Design

The first part of the assignment was basically to create a Red Black Tree, insert some elements on it and check if it is right.

The following steps were required to develop this program:

a) write the RBNode class
b) write the RBTree class
c)use the driver class to display and measure the results

The following methods were defined within the RBNode:

a) RBNode()
    a. Constructor
b) RBNode(T theData)
    a. Constructor
c) RBNode(T theData, T description)
    a. Constructor
d) getValue()
    a. returns the value from the node
e) getDescription()
    a. returns the description from the node
f) getParent()
    a. returns the parent from the node
g) setParent(RBNode newNode)
    a. sets a new parent node for the actual node
h) getLeft()
    a. returns the left child from the node
i) getRight()
    a. returns the right child from the node
j) setLeft(RBNode newNode)
    a. sets a new left child node for the actual node
k) setRight(RBNode newNode)
    a. sets a new right child node for the actual node
l) preOrder()
    a. preorder the node
m) height()
    a. returns the height from the tree
n) toStringPreOrder(String pathString)
    a. prints the tree
o) makeBlack()
    a. transforms the node into a black node

p) makeRed()
    a. transforms the node into a red node
q) isBlack()
    a. returns true if the node is black

The following methods were defined within the RBTree and BinaryTreeNode classes:

a) RBTree()
    a. constructor
b) newNode(int theData)
    a. creates a new node
c) RBNode getRoot()
    a. returns the root node
d) getRootValue()
    a. returns the value from the root
e) searchTree(RBNode nodeToBeSearch, int elementToBeSearch)
    a. search for an element
f) leftRotate(RBNode x)
    a. left rotate into a node
g) rightRotate(RBNode x)
    a. right rotate into a node
h) rbInsert(RBNode z)
    a. insert a new node
i) toString()
    a. prints the tree
j) rbInsertFixUp(RBNode z)
    a. given class for inserting a node

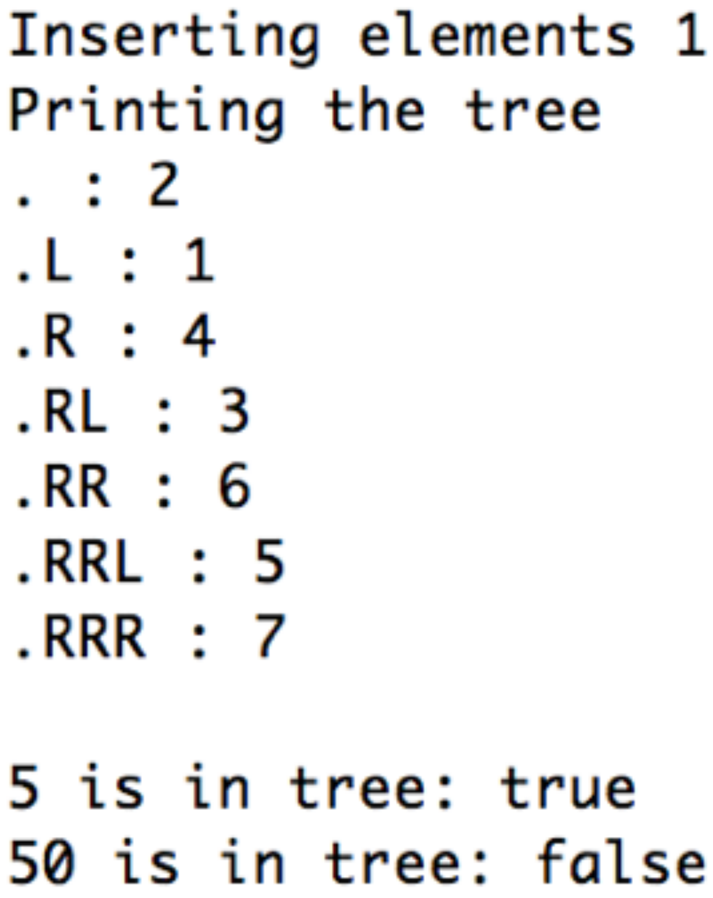
The Scanner class provided by Java was used to read in the necessary values within the provided driver program. The println method of the System.out object was used to display the inputs and results for the provided driver program.

3. Testing Plan

Insert some elements and prints the tree to check if the tree is right.

4. Test Cases

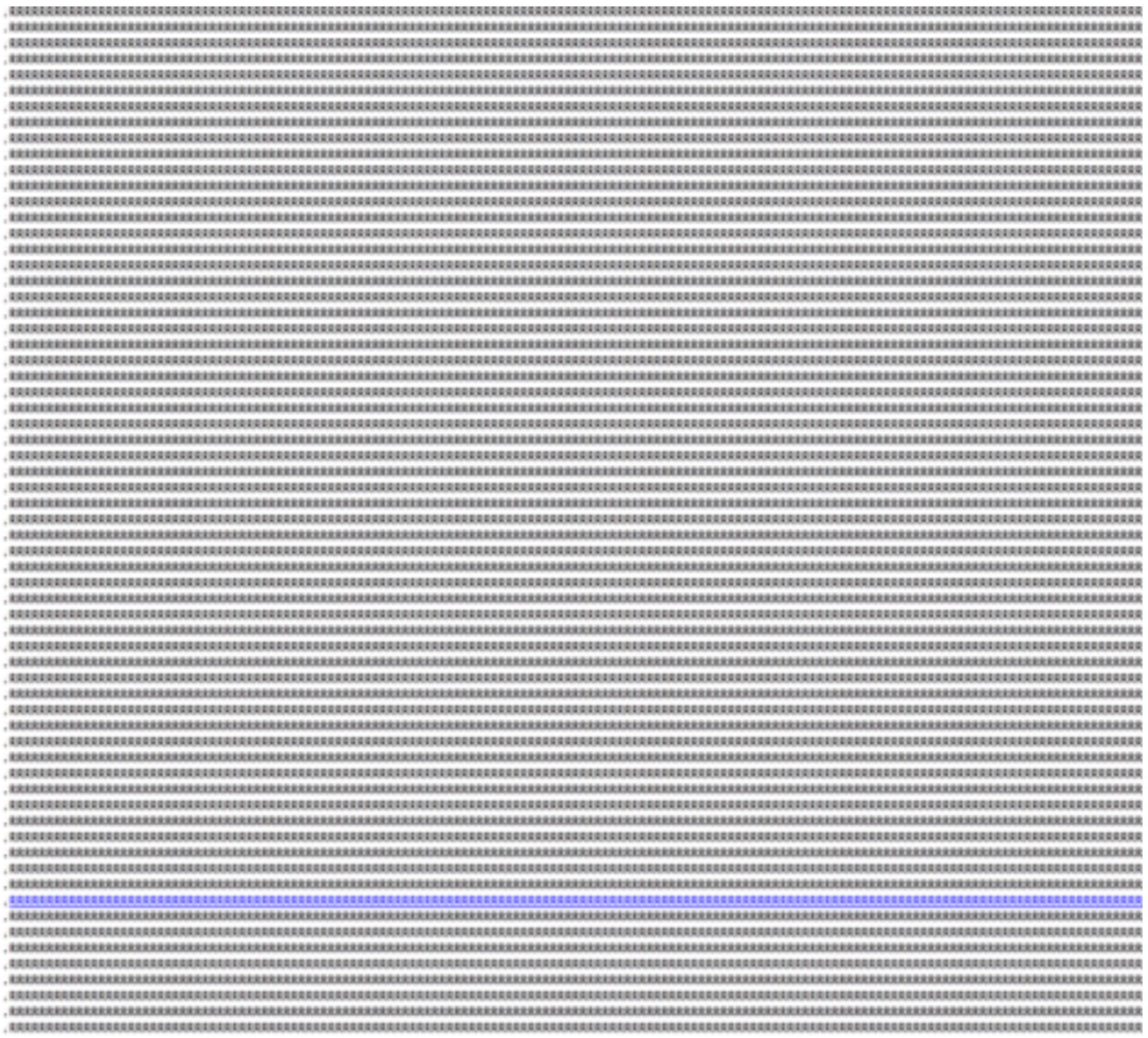
The tests case:



```
Inserting elements 1,2,3,4,5,6,7
Printing the tree
. : 2
.L : 1
.R : 4
.RL : 3
.RR : 6
.RRL : 5
.RRR : 7

5 is in tree: true
50 is in tree: false
```

And for the second part the tree as the tree is so big the console doesn't show everything properly but for smaller inputs it is working and showing correctly.


[illegible]

5. Analysis and Conclusions

The implementation for the Red Black Tree is a little bit confusing and I had a lot of difficult trying to do it but thinking now, after conclusion, is not as much different as a binary search tree. The hard part is to understand the color thing and the rotates. As the search was not working properly it was not possible to catch the times but the functions were written and I think that it is right. According to the theory we know that the Search, insertion and deleting in Red Black Tree nodes are O(log n) in average and worst case so this is great.

6. References

The parameters used was from the homework assignment provided in class, I also used as reference for learning how to work with compareTo the Java documentation and another references from an Algorithms course from Princeton.