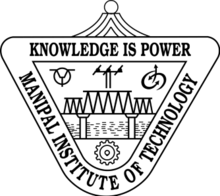
Manipal Institute of Technology

Manipal Academy of Higher Education

Manipal-576104. Karnataka, India



ADSA Mini Project Report

1st Semester M.Tech CSIS

Red Black Tree

Date of Submission: 02-11-19

**Submitted By**:

|  |  |
| --- | --- |
| Amit K. Pradhan | 190948027 |
| Mohammad Shagil Siddiqui | 190948019 |
| Pradyot S. Nadig | 190948012 |

Under the Guidance of

Prof. Geetha M

Manipal Institute of Technology

**Table of Contents**

**Introduction1**

**Algorithm2**

**Experimental Output5**

**Analysis8**

**Amortized Analysis9**

**Result9**

1. **Introduction**

Red Black Tree is a self-balancing Binary Search Tree(BST) where every node follows certain rules. It has one extra bit of storage per node which store its color (either red or black).

Each node of the tree contains the attributes **color, key, left, right, and p.**

A red-black tree is a binary tree that satisfies the following properties:

1. Every node is either red or black
2. The root is always black
3. Every leaf (NIL) is black.
4. If a node is red, then both its children are black
5. Every path from a node (including root) to any of its descendant NULL node has the same number of black nodes.

Why Red-Black Tree?

Most of the BST operations (e.g., search, max, min, insert, delete etc. take O(h) time where h is the height of the BST. The cost of these operations may become O(n) for a skewed Binary tree. If we make sure that height of the tree remains after every insertion and deletion, then we can guarantee an upper bound of for all these operations. The height of a Red-Black tree is always where n is the number of nodes in the tree.

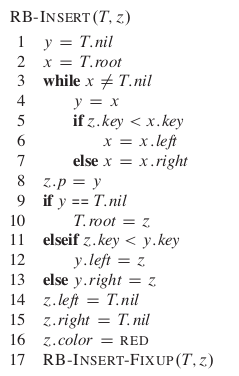
Comparison with AVL Tree

The AVL trees are more balanced compared to Red-Black Trees, but they may cause more rotations during insertion and deletion. So if the application involves many frequent insertions and deletions, then Red Black trees should be preferred. And if the insertions and deletions are less frequent and search is a more frequent operation, then AVL tree should be preferred over Red-Black Tree.

Every Red Black Tree with n nodes has height <= 2Log2(n+1)

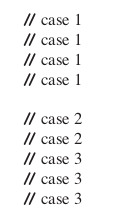
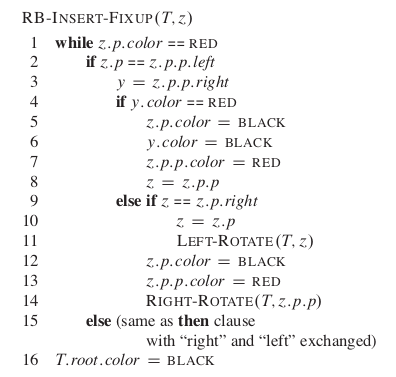
Applications:

1. Most of the self-balancing BST library functions like map and set in C++ (OR TreeSet and TreeMap in Java) use Red Black Tree
2. It is used to implement CPU Scheduling Linux.
3. **Algorithm**



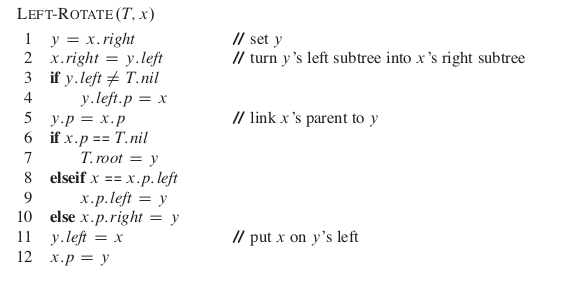
Insert node z (with key ‘key’) into tree T as if it were an ordinary binary search tree and then colour z red.

To guarantee the red-black properties are preserved, an auxiliary procedure to RB-INSERT-FIXUP(T,z) to recolour and perform rotation is called.



The while loop in lines 1–15 maintains the following three-part invariant at the start of each iteration of the loop:

1. Node z is red.
2. If z.p is the root, then z.p is black.
3. If the tree violates any of the red-black properties, then it violates at most one of them, and the violation is of either property 2 or property 4. If the tree violates property 2, it is because z is the root and is red. If the tree violates property 4, it is because both z and z.p are red.



The code for RIGHT-ROTATE is symmetric. Both LEFT-ROTATE and RIGHT-ROTATE run in time. Only pointers are changed by a rotation, all other attributes in a node remain the same.

1. **Experimental Output**

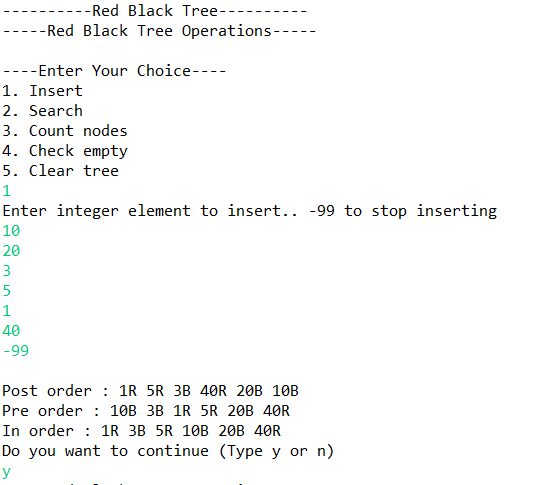
****

Figure 1:Insertion Of Nodes

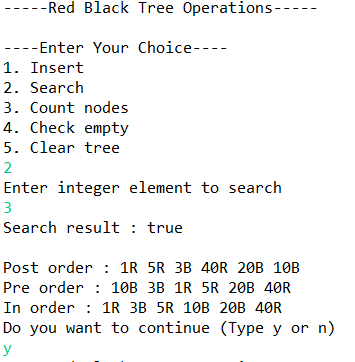
****

Figure 2:Searching of an element in the tree

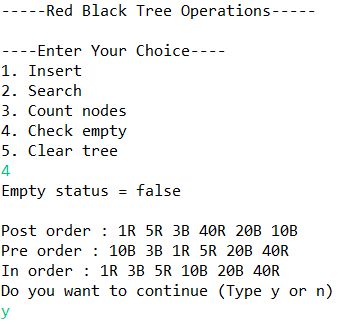
****

Figure 3: Checking if the tree is empty

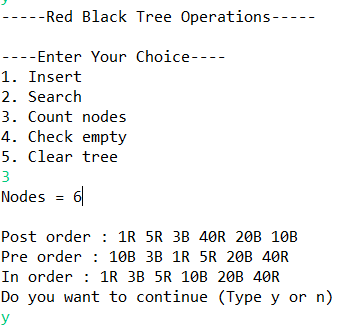
****

Figure 4: Counting the number of nodes

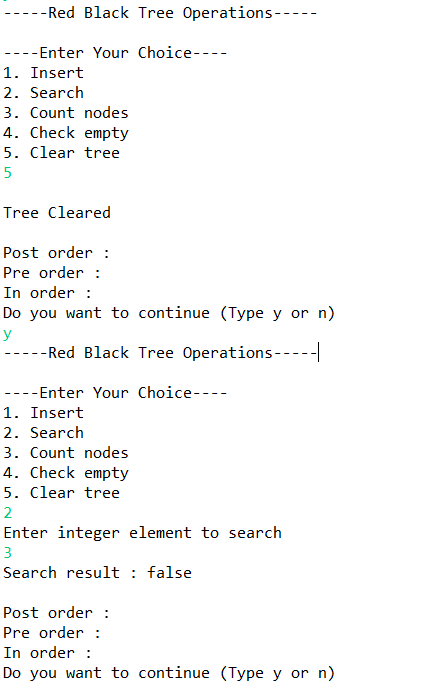


Figure 5: Clearing the tree and Searching for an element.

1. **Analysis**

Since the height of a red-black tree on n nodes is , lines 1–16 of RB-INSERT take time. In RB-INSERT-FIXUP, the while loop repeats only if case 1 occurs, and then the pointer z moves two levels up the tree. The total number of times the while loop can be executed is therefore . Thus, RB-INSERT takes a total of time. Moreover, it never performs more than two rotations, since the while loop terminates if case 2 or case 3 is executed.

Running time: Insertion takeswith rotations. Searching takes as the height of tree is always

1. **Amortized Cost**

Aggregation cost by aggregation method is given by:

Amortized cost for n Search or Insert or Search operation is:

By L Hospital rule

Therefore, Amortized cost of Insertion and Searching operation is Red-Black Tree is in .

1. **Results:**
   1. As the tree is always balanced, the running time of Insert, Search is
   2. If application involves many frequent insertions and deletions, then Red Black trees should be preferred.