ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 8, August 2019

A Study of Non Linear Data Structure: Red Black Tree

Pooja Rani

Research Scholar, Department of Computer Science and Technology, CUP Bathinda, Punjab, India

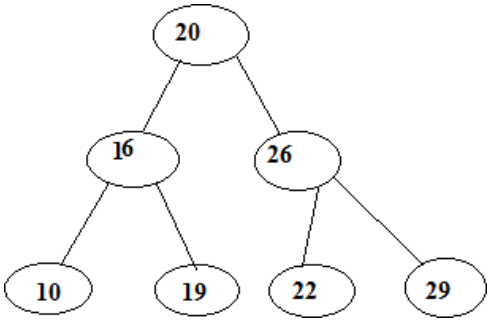
ABSTRACT: Trees are widely used abstract data type or we can say trees are the data structure that implementing abstract data type. The reason behind to use the trees are that whenever we want to store information in forms of hierarchy we can use trees. Trees provide the moderate insertion/ deletion but quicker than arrays and slower than unordered linked lists. Trees don't have an upper limit on number of nodes as nodes are linked using pointers. Binary trees are special case of tree where every node has almost two children. Binary search tree is a node-based binary tree data structure in which left sub tree of a node contains nodes with key value less than the root node key and right sub tree that contains nodes with key value greater than root key. Binary Search Tree provides moderate access or search and it is quicker than linked list but slower than arrays .With the binary search tree, tree shape depends on insertions order and that can be degenerated. So, with binary search tree we can't guarantee efficient insertion and retrieval. Finally we look at red-black trees, a variation of binary search trees that overcome binary search tree's limitations through a logarithmic bound on insertion and retrieval.

KEYWORDS*:* Binary Tree, Search Tree, Red Black Tree.

I. INTRODUCTION

Trees are the non-linear data structure. Trees represent the hierarchical data. Trees have the nodes and edges where nodes are connected with each other through edges. The topmost node called as root node. The immediate left and right nodes of root node called as children and so on for rest of the nodes. **Binary Tree** is the data structure that is used to store the data. In the binary tree there is a condition.i.e there should be maximum of two children of the node in the tree.

Binary Search Tree: Binary search tree are the tree that have a special behavior: The node left child value should be minimum value than its parent node and the node right child value should be greater than its parent node. Likewise the left sub tree of the root node contains the key values less than root node and the right sub tree of the root node contains the key values greater than the root nodes. The left sub tree and right sub tree of the root node also the binary search tree. The binary search tree represented as:



Binary search tree allows us to print the elements of binary search tree in sorted order by using the algorithm called as inorder tree traversal. With this algorithm we get that it first traverse the left child of the node after that the root node and last is the right child. That's why it is called as inorder traversal because it traverse root node in between its left

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 8, August 2019

and right child. The algorithm for the traversal of tree in inorder traversal is:

Inorder-Tree-Traversal(x):

1. If x! = NIL
2. Inorder-tree-walk(x, left)
3. Print x, key
4. Inorder-tree-walk(x, right)

It takes O(n) time to walk an n-node binary search tree, since after the initial call, the procedure calls itself recursively exactly twice for each node in the tree-once for its left child and once of its right child.

Operations in Binary Search tree that discussed here:
Searching
Insertion

Searching:
For searching the element we start from the root node, if the key value(value that is to be searched)is less than the root node then we go to left sub tree otherwise go to the right subtree.The algorithm for searching is as follows:

Tree-Search(x, k)
1. **If** x==NIL or k==x.key
2. **Return** x
3. **If** k<x.key
4. **Return** Tree-Search (x.left, k)
5. **Else return** Tree-Search (x.right, k)

Here procedure begins its search at the root and trace a simple path downward in the tree. For each node x it encounters, it compares the key k with x.key.If two keys are equal, the search terminates. If k is smaller than x.key, the search continues in the left sub tree of x, since the binary search-search tree property implies that k could not be stored in the right subtree.So, if k is larger than x.key, the search continues in the right subtree.The running time of Tree-Search is O (h), where h is the height of the tree.

Insertion
For inserting the node into the binary search tree. First of all we require the existing binary search tree. Afterwards when we start insertion first locate its position. We start searching from the root node of the tree, and then if we get the node value less than the key value (inserted node value) search the empty location in the left sub tree of the node. Otherwise we have to search the empty location in the right sub tree of the binary search tree and insert the data there.

Tree-Insert (T, z)

1. Y=NIL 2.X=T.root
3. **While** x! =NIL
4. Y=x
5. **If** z.key<x.key
6. x=x.left
7. **else** x=x.right 8.z.p=y
9. **if** y==NIL
10. T.root=z 11.**Elseif** z.key<y.key
12. y.left=z 13.**Else** y.right=z

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 8, August 2019

II. RED BLACK TREE

Red Black tree is a binary search tree with one extra bit of storage per node; its color, which can be either RED or BLACK. By constructing the node colors on any simple path from the root to a leaf, red-black trees ensure that no such path is more than twice as long as any other, so that the tree is approximately balanced. Each node of the tree now contains the attributes color, left, key, right and p.if a child or the parent of a node does not exist, the corresponding pointer attribute of the node contains the value NIL. We shall regard these NIL as being pointers to leaves (external nodes) of the binary search tree and the normal, key-being nodes as being internal nodes of the tree.

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

1. Root is black.
2. No Red-Red parent-child.
3. No black nodes from root to node with less than 2 children are same.

III. ALGORITHM

Operation of Red-Black Tree Discussed here:

Insertion
Deletion

Insertion:

We can insert a node into an n-node red black tree in o (logn) time. We have following Algorithm to insert node into red- black tree.

Algorithm

1. If(empty)
2. Create a Black Node
3. 3.Else

4. Create red leaf node
5. If (parent is black) done
6. Else
7. If (parent sibling is red)
8. If (parent's parent is root) 9.Recolor
10. Else recolor & recheck
11. Else

Rotate If (LL) rotate right
If (LR) rotate rights->left If (RL) rotate left->right If (RR) rotate left
To insert node into red black tree we have following procedure:

1. If empty tree create black root node.
2. Insert new leaf node as red (a) if the parent is black then done. (b)If parent is red
   1. If black or absent sibling then rotate, recolor and done
   2. If red sibling then recolor and check again.

IV. RESULTS

Example:

1. We have the following tree where we insert the node with key value 4 into it. So, while we inserting node into the tree the node color should be red.

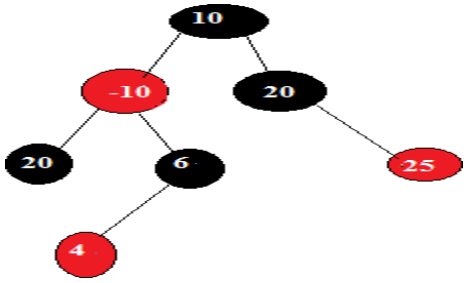
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

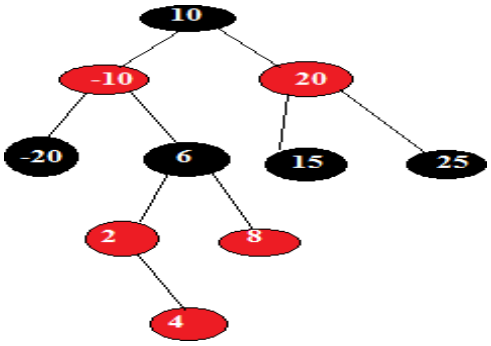
Website: www.ijircce.com

Vol. 7, Issue 8, August 2019

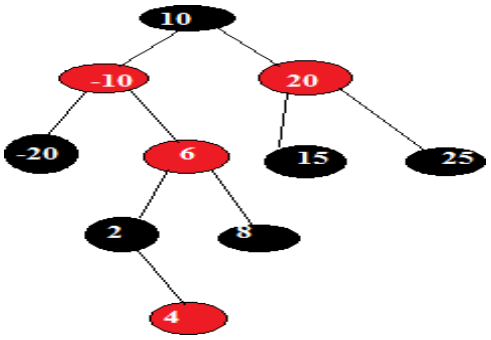


So it is not violating the red black tree properties.

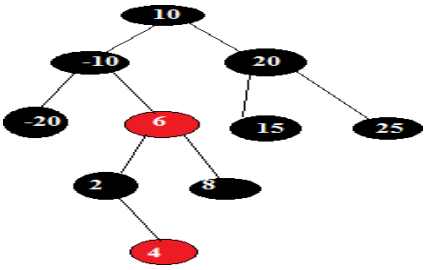
2. We have the following tree where we insert node with key value 4 into it but it violate the red-black tree property.



So we here simply change the color. Then the tree looks like this:



But here also violate red black tree property. So, we again recolor the nodes. The tree looks like as now:



ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

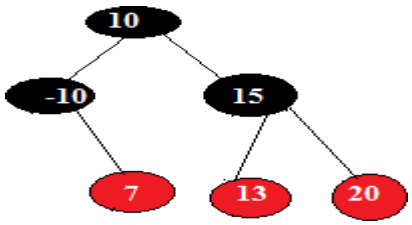
(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

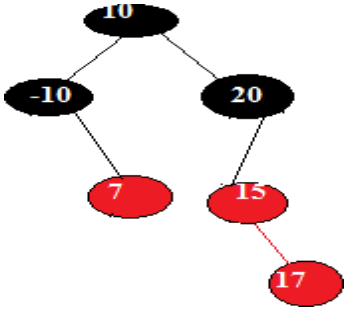
Vol. 7, Issue 8, August 2019

3. We have tree where we insert node with key value 13 into it. And it creates R-R conflict.

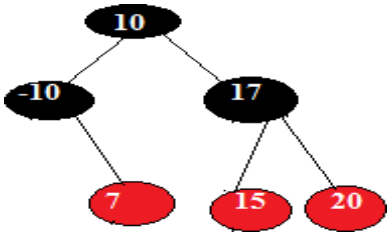
We did here the single rotation that is right rotation. Now after rotation tree looks like:



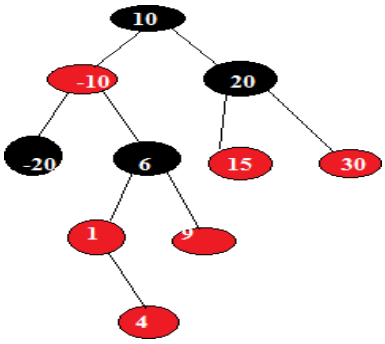
4. We have the tree where we insert node with key value 17.That creates the R-R conflict.



Here we did the double rotation. That is first we done the left rotation then the right rotation. After rotation the tree looks like as:



Here we have tree where we insert the node with key value 4.After insertion tree looks like as:



ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

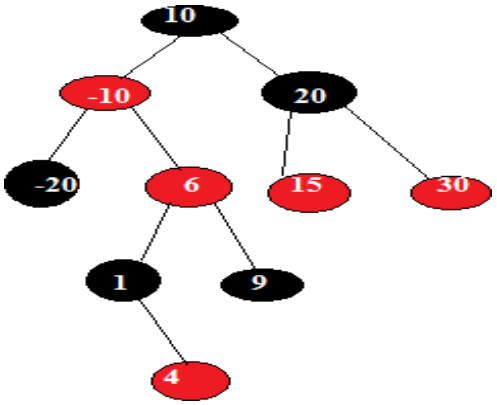
International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

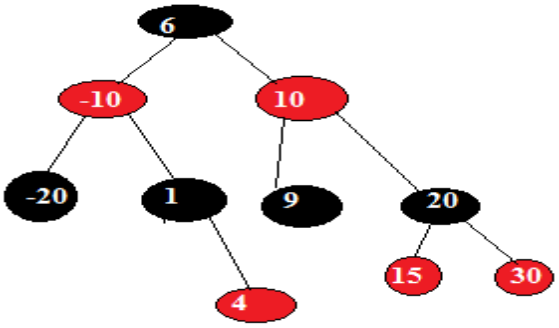
Website: www.ijircce.com

Vol. 7, Issue 8, August 2019

Now it creates R-R conflict. So, we done recoloring hereafter recoloring the tree looks like as:



Here it again R-R conflict. So, we did double rotation here. The first rotation is left and second rotation is right rotation. After rotation tree looks like as:



Now it satisfies the Red Black Property.

Deletion:
We can delete a node into an n-node red black tree in o (logn) time. We have following Algorithm to delete node from red- black tree.

1. Find the node to be deleted using binary search tree traversal.
2. If node to be deleted has 2 non-null children, replace it with its inorder successor, then delete inorder successor.
3. If node to be deleted is red then just delete it.
4. If node to be delete is black but has one red child replace it with that child & change color of child to black.
5. Otherwise double black situation is created and follow 6 cases.

Black double node means we have one less black nodes on this path than rest of the tree.
Steps:
1. Convert to 0 or 1 not null node.
2. If red node just delete it.
3. If black node with red child then delete black & turn red into black node.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

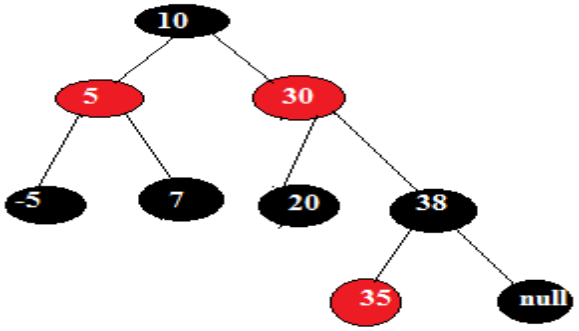
(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

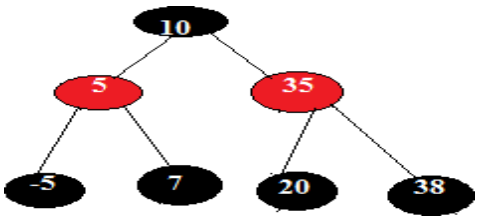
Vol. 7, Issue 8, August 2019

Example:

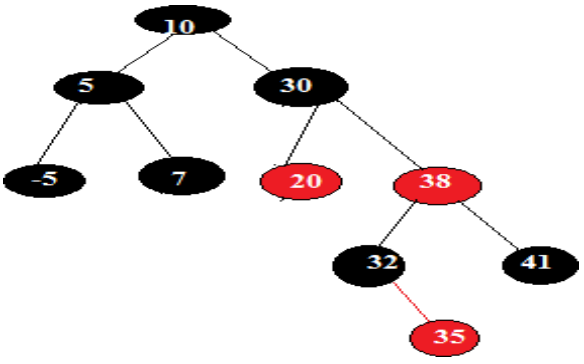
1. We have Red-Black tree where we have to delete 30.



For deleting 30 firstly we have to find the inorder successor of 30.i.e 35.Now replace 30 with 35.delete 35.After deletion tree look like as:



2. We have the tree where we have to delete 30.But now successor of 30 have the right child. The tree as follows:



For successful deletion we have to replace 30 with 32.after that replace 32 with 35 and change color. The tree looks like as follows after deletion:

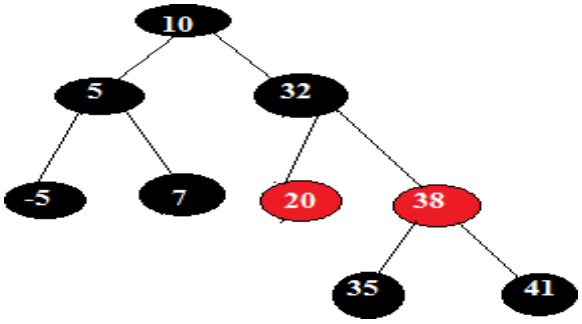
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

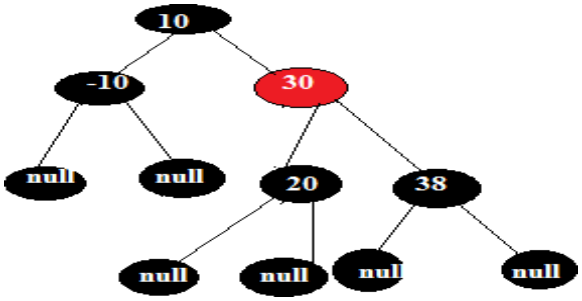
(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

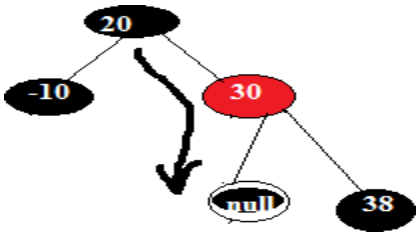
Vol. 7, Issue 8, August 2019



If we want to delete node and color of node is black and children of node also black. Then to delete that node we have 6 cases. Because while we delete node and replace with its child we have a double black node created.



Here we delete 10.Find the inorder successor.i.e 20.Replace 10 with 20.Now delete 20.Now it creates the double black node.



This path as shown in above figured directed by arrow has only one black node. So this leaf node is double black node. We have 6 cases to solve this. The 6 cases are as follows.

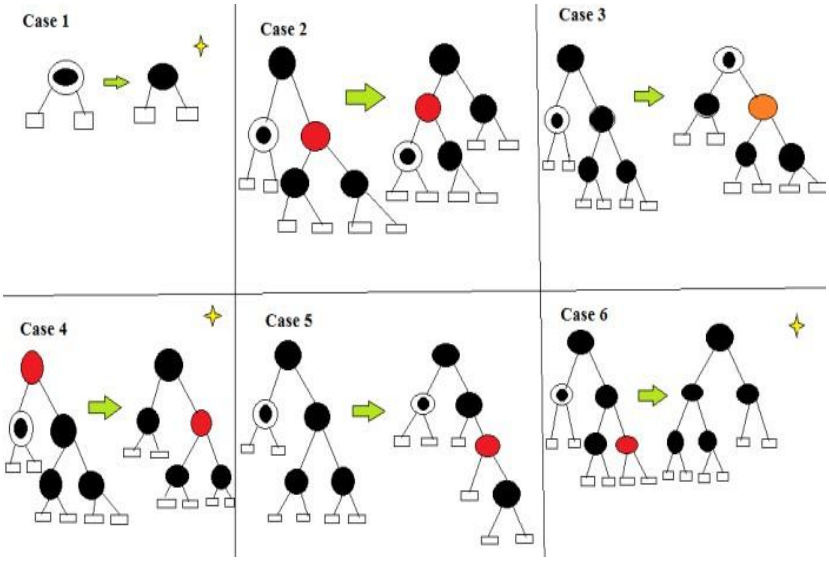
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 8, August 2019



We have seen in the above figure. The case 1, 4, 6 have the star sign at top of the cases. This shows that these cases are the terminal case. And the cases 2, 3, 5 are the non-terminal cases.

V. CONCLUSIONS

Trees are the non-linear data structure where data is stored in non-linear fashion. Trees are the abstract data structure. We used binary trees for data storage in hierarchical order. We use the binary search tree for constructing the abstract data structure such as sets, multisets, and associative arrays. But with Binary search tree disadvantage is that the shape of the binary search tree depends entirely on the order of insertions and deletions and become degenerate. So we use the red black tree. With the use of red black tree the insertion and deletion become faster as compared to AVL trees. Here it depends upon the cost of the structural changes to the tree, as this will depends a lot on the implementation and runtime.

REFERENCES

1. https://www.tutorialspoint.com/data_structures_algorithms/tree_data_structure.html
2. http://www.geeksforgeeks.org/binary-tree-set-1-introduction/
3. https://en.wikipedia.org/wiki/Data_structure
4. http://www.geeksforgeeks.org/red-black-tree-set-1-introduction-2/
5. http://www.geeksforgeeks.org/red-black-tree-set-2-insert/
6. http://www.geeksforgeeks.org/binary-search-tree-set-1-search-and- insertion/
7. https://www.tutorialspoint.com/data_structures_algorithms/binary_search_t ree.htm
8. https://www.cs.auckland.ac.nz/software/AlgAnim/red_black.html
9. https://www.cs.rochester.edu/~gildea/csc282/slides/C12-bst.pdf
10. http://www.cs.cmu.edu/~clo/www/CMU/DataStructures/Lessons//lesson4_1.htm
11. https://www.javatpoint.com/tree
12. https://www.hackerearth.com/practice/data-structures/trees/binary-search-tree/tutorial/