Assignment no - 8

Title - AVL Tree

Problem Statement - A Dictionary stores keyword its meanings. Provide facility for adding new keywords, deleting keywords, updating values of any entry. Provide facility to display whole data stored in ascending/desending order. Also find how many max comparisons may require for finding any keyword. Use height balance tree and find the complexity for finding a keyword.

Objective - To understand construction of AVL tree and its rotation techniques.

Outcome - At this end of this assignment stuends will be able to construct a AVL tree and perform rotation.

Theory -

AVL Tree (Self Balancing Tree) is a binary search tree in which the difference of heights of left and right subtrees of any node is less than or equal to 1. The techniques of balancing the height of binary tree was developed by Addson, Velscii and Candis and hence given the short form as AVL tree.

The tree is balanced if

① T_L and T_R are height balanced

② $h_L - h_R <= 1$ where $h_L - h_R$ are the heights of T_L & T_R.

Scanned with CamScanner

Advantages of AVL Tree

since AVL trees are height balance trees, operations like insertion and deletion have low time complexity

Balance factor = height (left subtree) - height (right subtree)

AVL Rotations

To balance itself, an AVL tree may perform the following four types of rotations

i) Left Rotation
ii) Right Rotation
iii) Left-Right Rotation
iv) Right-Left Rotation

The first two rotations are single rotations and the next two rotations are double rotation

Representation of AVL Tree

```
struct AVLNode
{
    int data;
    struct AVLNode *left, *right;
    int balfactor;
};
```

Scanned with CamScanner

Algorithm

① Insertion-

a) First insert a new element into the tree using BST's insertion logic

b) After inserting the elements you have to check the balance factor of each node.

c) When balance factor of every node will be found like 0, -1, 1 then algorithm will proceed for next operation

d) When the balance factor of any node comes other than the above three values then the tree is said to be imbalanced. Therefore the suitable rotation to make it balanced and then the algorithm will proceed for the next operation.

② Delete

a) First find that node where k is stored

b) Secondly delete those contents of the node

c) Claim: Deleting a node in an AVL tree can be reduced by deleting a leaf. There are three possible cases

→ When x has no children then delete x

→ When x has one child. Let 'x' be child of x

→ Notice 'x' cannot have a child, since subtree of T can differ in height by atmost one.

→ then replace the contents of x with x'

→ then delete x' (a leaf)

d) When x has 2 children

→ the find x's successor z (which has no left child)

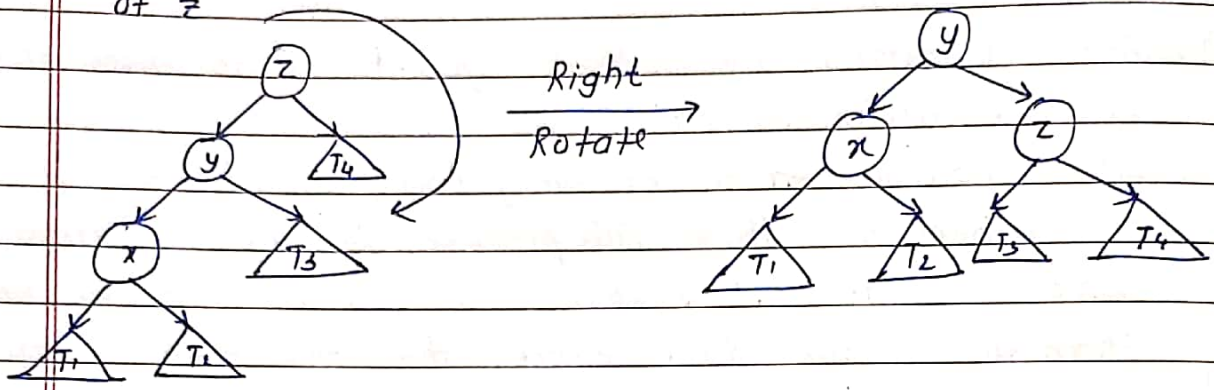
→ then replace x's content with z's contents and

→ delete z

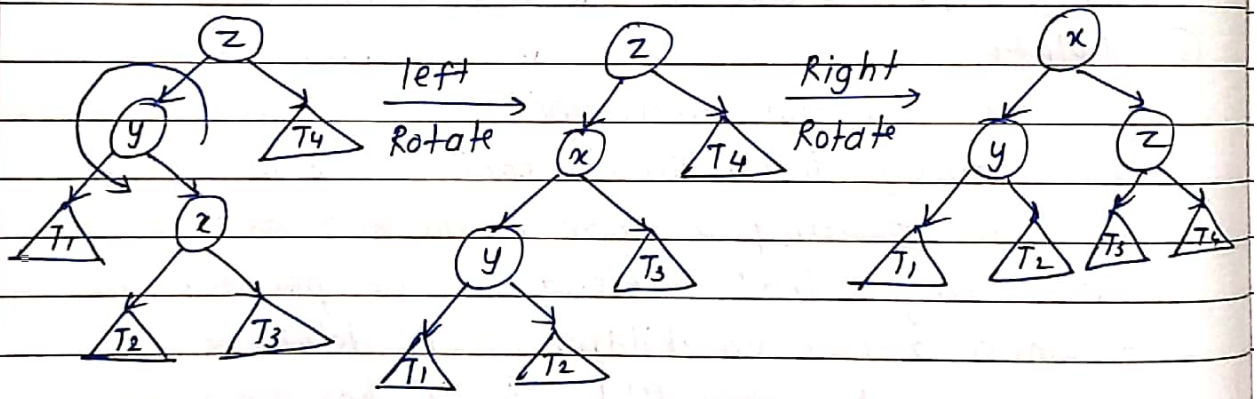
Scanned with CamScanner

Rotations

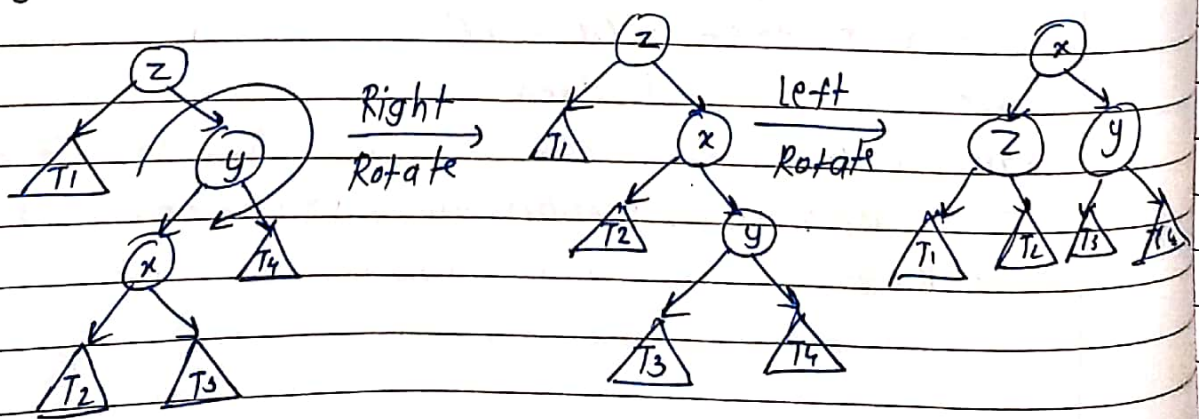
① Left-Left case - x is the left child of y & y is left child of z



② Left-Right case - x is the right child of y & y is the left child of z.

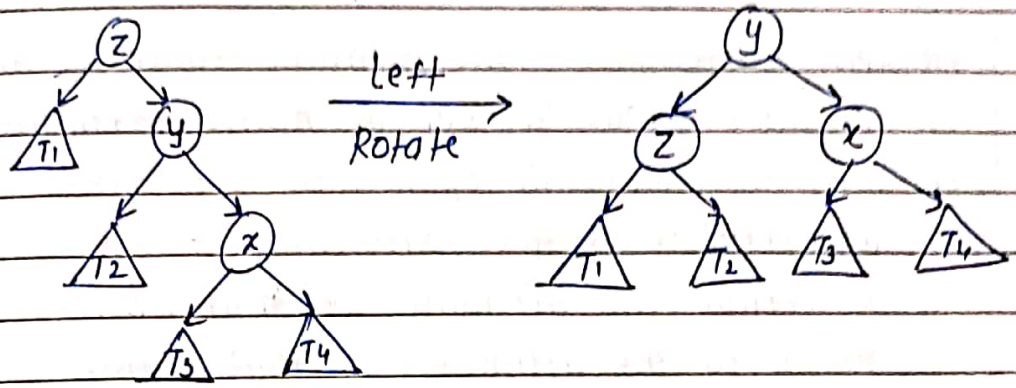


③ Right-Left case - x is the left child of y & y is the right child of z

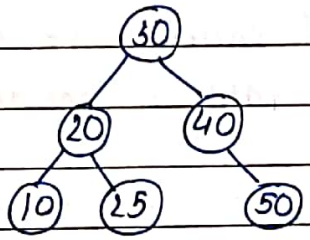
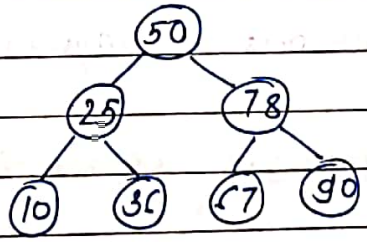
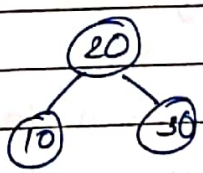


Scanned with CamScanner

④ Right-Right case - x is the right child of y & y is the right child of z



Test case

Test case		Expected O/P	Actual O/P	Result
①	10, 20, 30, 40, 50, 25		As expected	Pass
②	50, 36, 78, 67, 25, 10, 90		As expected	Pass
③	10, 20, 30		As expected	Pass

Conclusion -

We understood the concept of AVL tree and rotation techniques and successfully complete the assignment.

Scanned with CamScanner