CSE 3010 - Lecture #42 — AVL Tree functions in C

Note Title 3/5/2020

— How do we balance an unbalanced tree

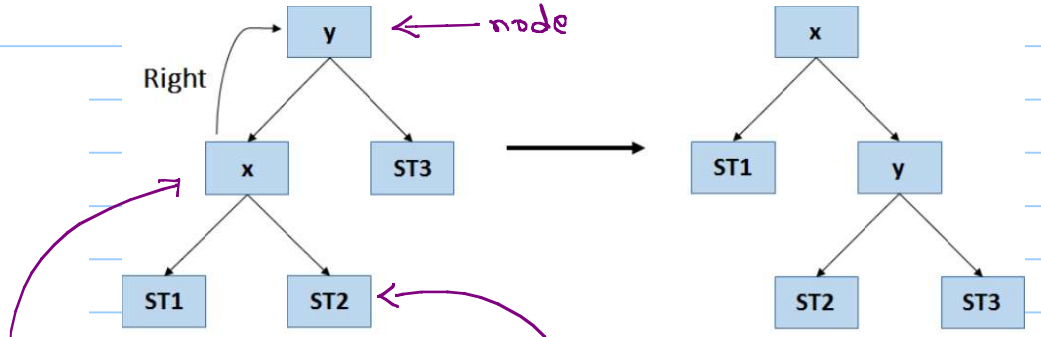
```
//Right rotates the sub-tree rooted at y
AVLNODE* rightRotate(AVLNODE* y)
{
    //Saving the nodes which will be lost first
    AVLNODE* x = y->left;
    AVLNODE* ST2 = x->right;

    //Performing the rotation
    x->right = y;
    y->left = ST2;

    //No need to change x->left and y->right as they still
point to the same nodes
    return(x);
}
```

Two points to consider:

1) Child of node being rotate
2) Grandchild of node being rotated

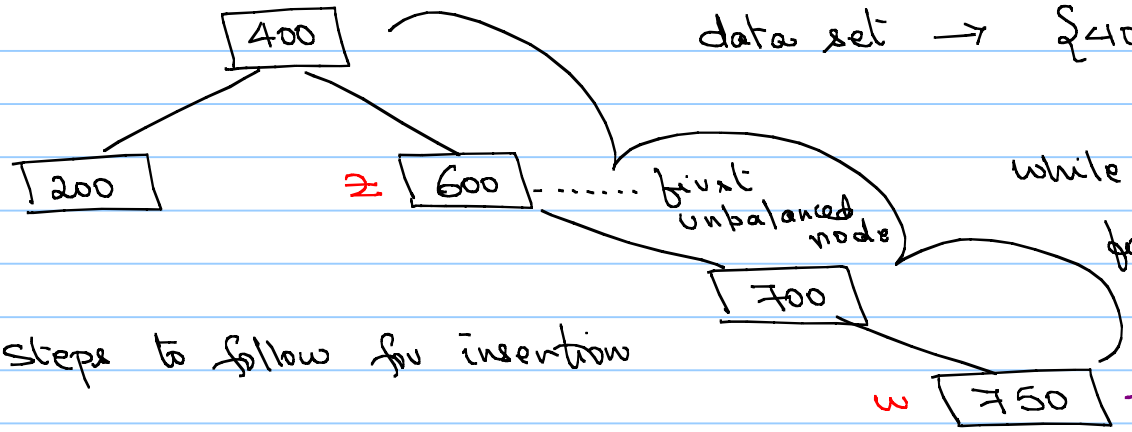


rightRotate ([y])

x = y → left

ST2 = x → right

ST2 is no more the right child of x



data set → {400, 200, 600, 700, 750

while traversing up to the root, find the first unbalanced node

(new) node of interest

y → child of z
x → grand child of z
→ right child of z
→ right child of y

Right Right Case

Steps to follow for insertion

↳ Add the node as you would in a BST

↳ check for the balance factor
if OK, proceed with another insertion
if not OK, balance the tree

bf = 2 - 0 = 2

perform a left rotation