

Lab 14

AVL Trees



**What is an AVL Tree?**

AVL tree is a self-balancing binary search tree in which each node maintains extra information called a balance factor whose value is either -1, 0 or +1.

AVL tree got its name after its inventor Georgy Adelson-Velsky and Landis.

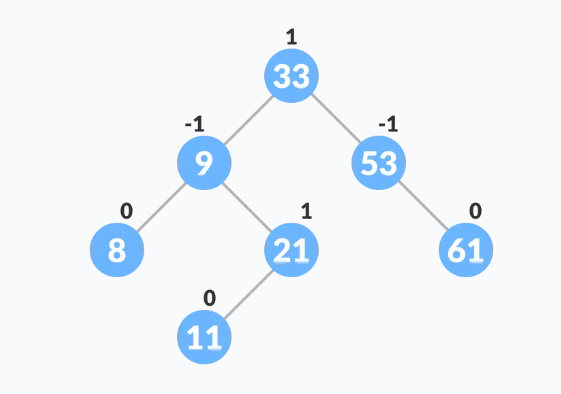
**What is Balance Factor?**

Balance factor of a node in an AVL tree is the difference between the height of the left subtree and that of the right subtree of that node.

Balance Factor = (Height of Left Subtree - Height of Right Subtree) or (Height of Right Subtree - Height of Left Subtree)

The self-balancing property of an AVL tree is maintained by the balance factor. The value of balance factor should always be -1, 0 or +1.

**An example of a balanced AVL tree is:**



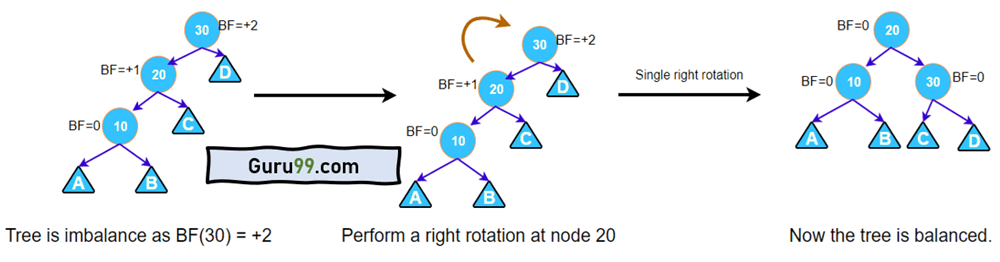
**AVL Rotations:**

To make the AVL Tree balance itself, when inserting or deleting a node from the tree, rotations are performed.

We perform the following LL rotation, RR rotation, LR rotation, and RL rotation.

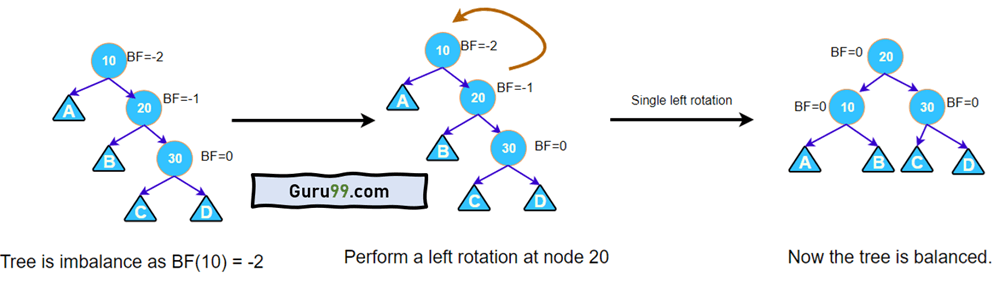
* Left – Left Rotation
* Right – Right Rotation
* Right – Left Rotation
* Left – Right Rotation

**Left – Left Rotation**

This rotation is performed when a new node is inserted at the left child of the left subtree

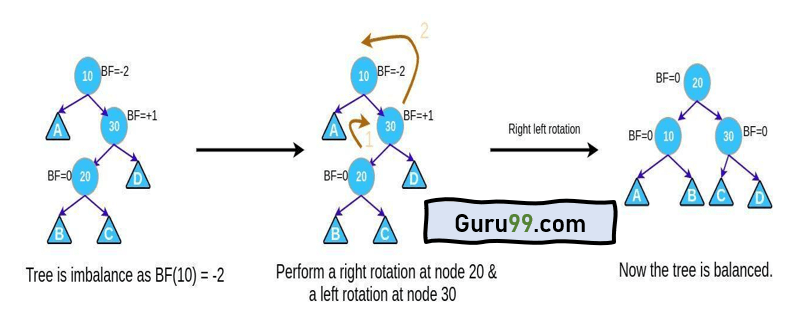
**Right – Right Rotation**

This rotation is performed when a new node is inserted at the right child of the right subtree.



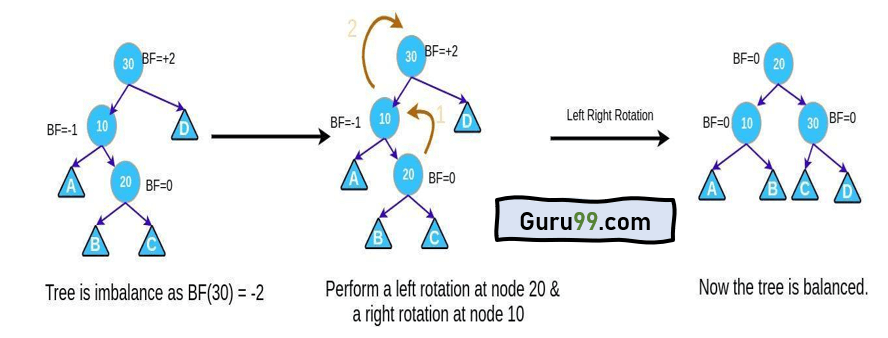
**Right – Left Rotation**

This rotation is performed when a new node is inserted at the right child of the left subtree.



**Left – Right Rotation**

This rotation is performed when a new node is inserted at the left child of the right subtree.



**Lab Task**

Write a C++ program to implement AVL trees. The program should

* insert a new node,
* delete a node, and
* search for a node.