AVL trees

AVL Trees

1 .These are height balanced binary search trees, We balance height of a BST, because we don't want trees with nodes which have large height

2. This can be attained if both subtrees of each node have roughly the same height.

3. AVL tree is a binary search tree where the height of the two subtrees of a node differs by at most one

Height of a null tree is -1

4. an AVL tree (named after inventors Adelson-Velsky and Landis) is a self-balancing binary search tree.

AVL Tree

Que. How to find out balance of a BST.
Ans: By finding the balance factor of a BST.

Balance factor = height of left subtree – height of right subtree

All node's balance factor should be {-1, 0, 1}

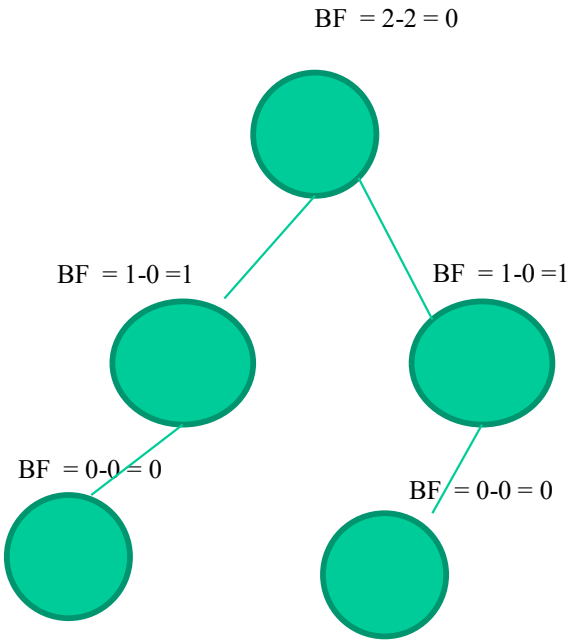
If any node's balance factor is less than -1 or more than 1 then it is not balanced search tree.

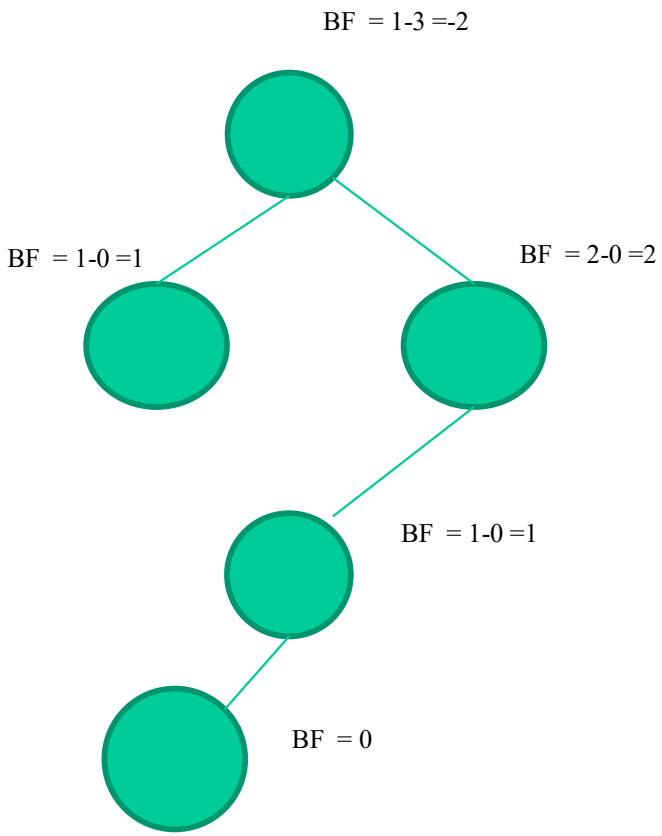
If an insertion cause an imbalance, which nodes can be affected?

Nodes on the path of the inserted node.

Let U be the node nearest to the inserted one which has an imbalance.

- insertion in the left subtree of the left child of U
- insertion in the right subtree of the left child of U
- insertion in the left subtree of the right child of U
- insertion in the right subtree of the right child of U

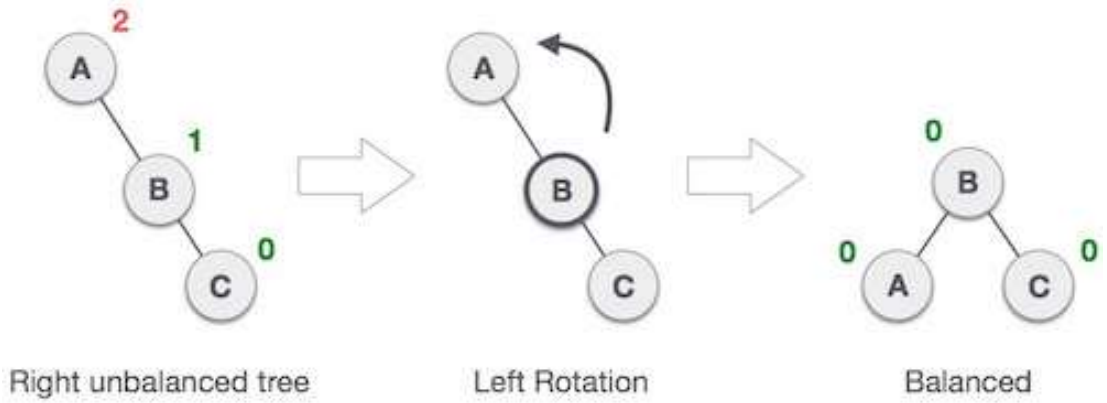




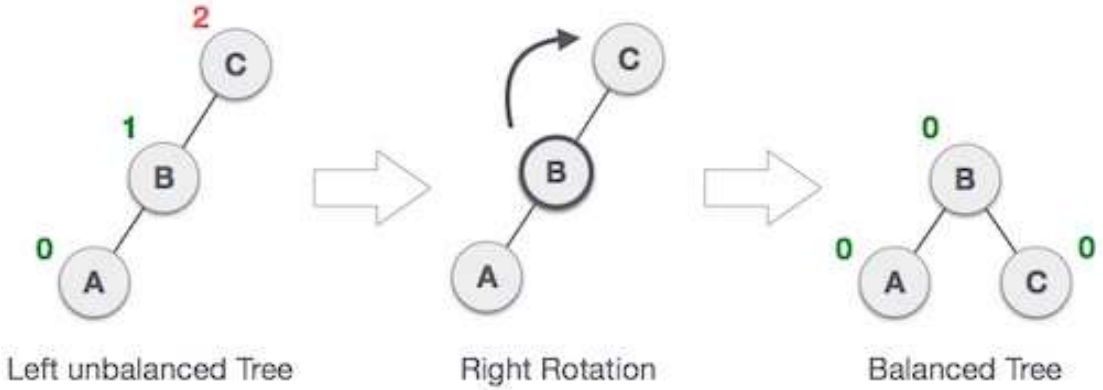
So, at the time of insertion we need to check balance factor of node and if BF is more than 1, then we need to perform rotation.

Rotation is performed always on 3 nodes

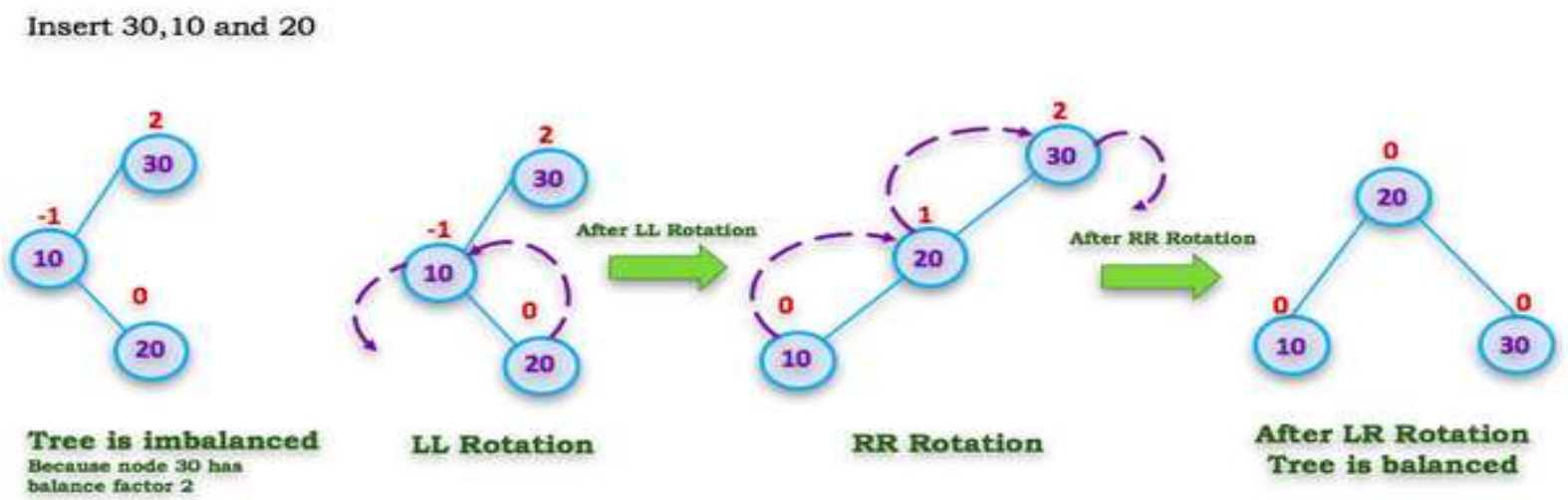
LL rotation ---------- RR imbalance



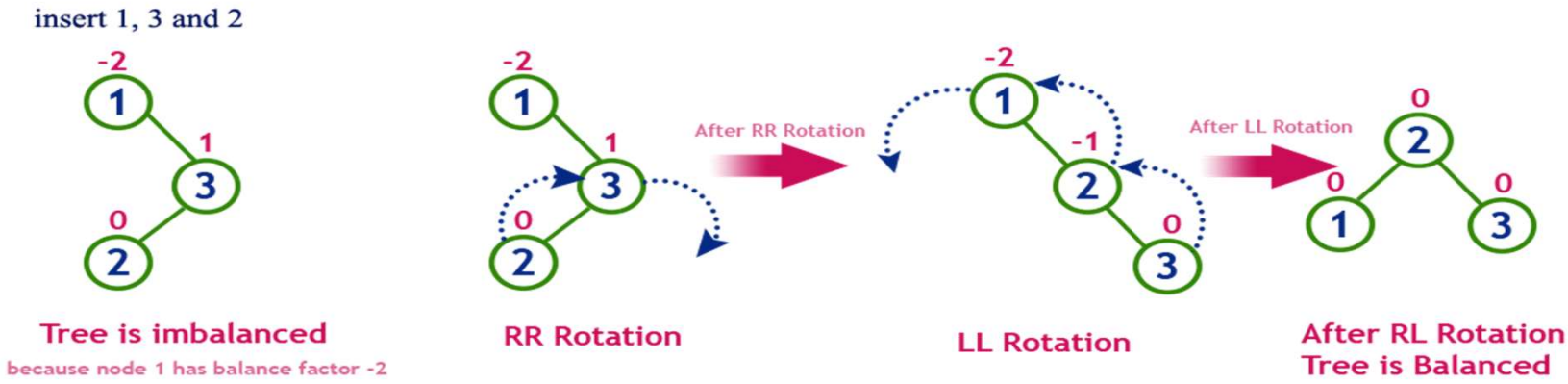
RR rotation ------ LL imbalance

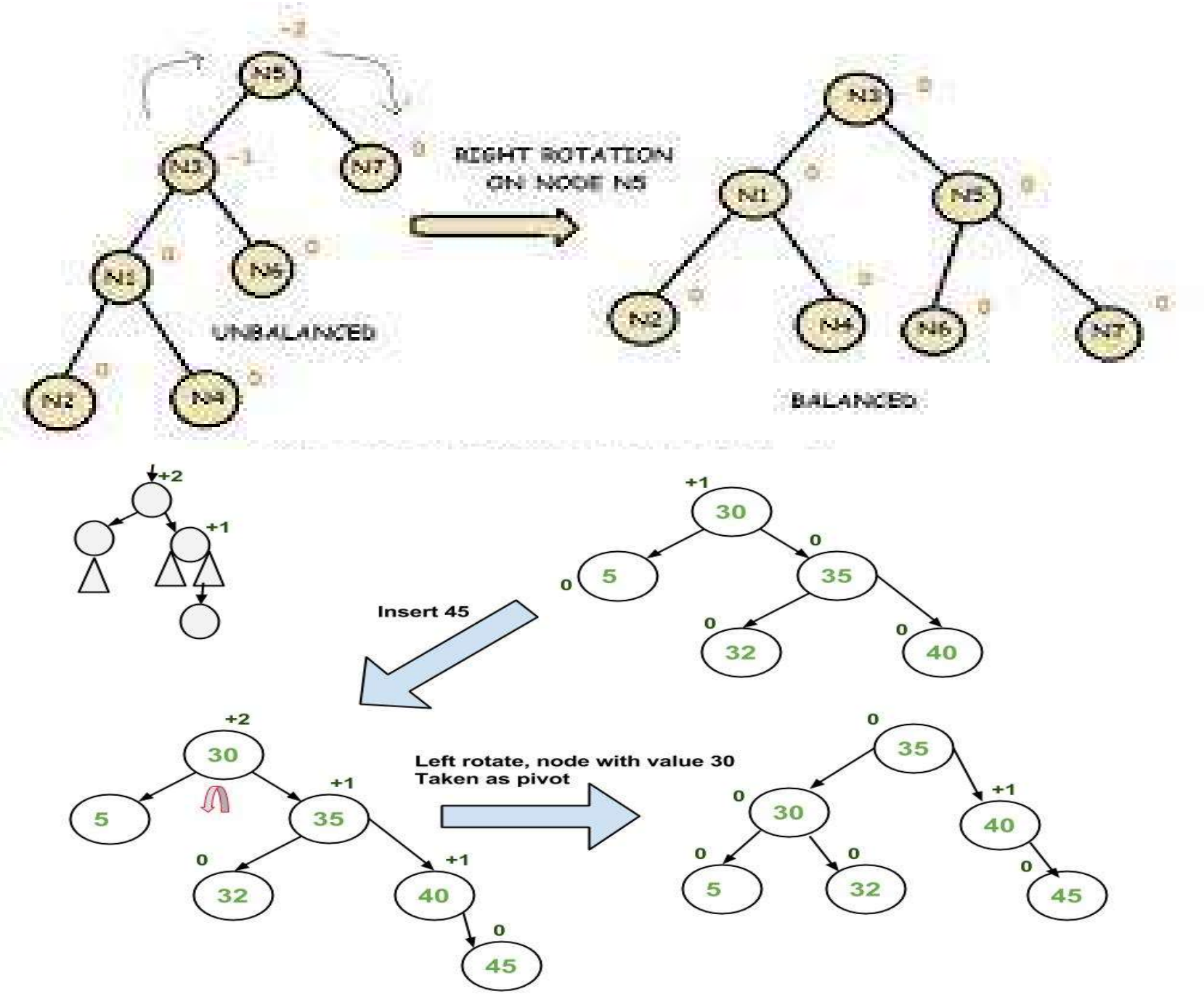


LR rotation



RL rotation




N5
N3
N7
N1
N6
N2
N4
UNBALANCED
RIGHT ROTATION
ON NODE N5
N3
N1
N5
N2
N4
N6
N7
BALANCED
+2
+1
30
5
35
32
40
Insert 45
+2
30
5
+1
35
0
32
+1
40
0
45
Left rotate, node with value 30
Taken as pivot
0
35
0
30
+1
40
0
0
5
32
0
45