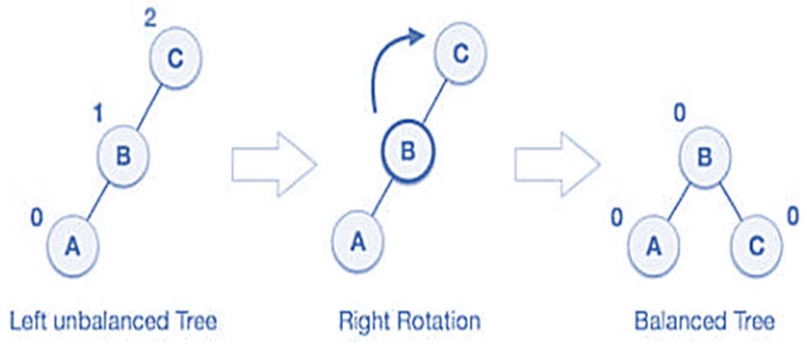
➢ AVL Tree: -

AVL Tree can be defined as height balanced Binary Search tree in which each node is associated with balance factor (BF) between (-1 to 1) or either -1, 0 or 1 AVL Tree introduced in 1962 by Adelson-Velski-Landis

Balance Factor = Height of left sub tree – Height of light sub tree

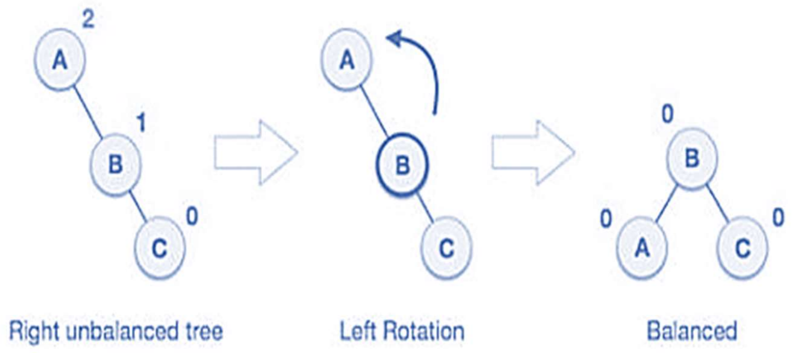
AVL Rotations: -

1. **LL Rotation: -**
   When BST becomes unbalanced, due to a node is inserted into the left subtree of the left subtree of C, then we perform LL rotation, LL rotation is clockwise rotation (right rotation), which is applied on the edge below a node having balance factor 2.



   In above example, node C has balance factor 2 because a node A is inserted in the left subtree of C left subtree. We perform the LL rotation on the edge below A.

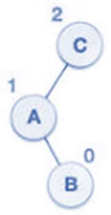
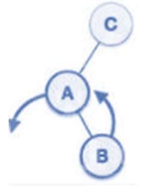
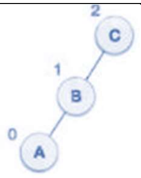
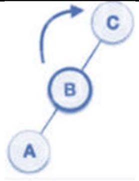
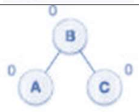
2. **RR Rotation: -**
   When BST becomes unbalanced, due to a node is inserted into the right subtree of the right subtree of A, then we perform RR rotation, RR rotation is an anticlockwise rotation, which is applied on the edge below a node having balance factor -2



   In above example, node A has balance factor -2 because a node C is inserted in the right subtree of A right subtree. We perform the RR rotation on the edge below A.

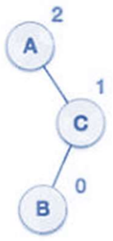
3. **LR Rotation: -**

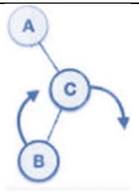
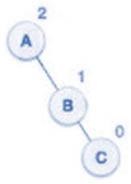
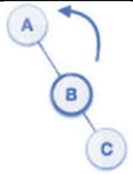
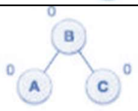
Double rotations are bit tougher than single rotation which has already explained above. LR rotation = RR rotation + LL rotation, i.e., first RR rotation is performed on subtree and then LL rotation is performed

State	Action
	A node B has been inserted into the right subtree of A the left subtree of C, because of which C has become an unbalanced node having balance factor 2. This case is L R rotation where: Inserted node is in the right subtree of left subtree of C
	As LR rotation = RR + LL rotation, hence RR (anticlockwise) on subtree rooted at A is performed first. By doing RR rotation, node A, has become the left subtree of B.
	After performing RR rotation, node C is still unbalanced, i.e., having balance factor 2, as inserted node A is in the left of left of C
	Now we perform LL clockwise rotation on full tree, i.e., on node C. node C has now become the right subtree of node B, A is left subtree of B
	Balance factor of each node is now either -1, 0, or 1, i.e., BST is balanced now.

4. **RL Rotation: -**

Double rotations are bit tougher than single rotation which has already explained above. RL Rotation = LL rotation + RR rotation, i.e., first LL rotation is performed on subtree and then RR rotation is performed

State	Action
	A node B has been inserted into the left subtree of C the right subtree of A, because of which A has become an unbalanced node having balance factor - 2. This case is RL rotation where: Inserted node is in the left subtree of right subtree of A

	As RL rotation = LL rotation + RR rotation, hence, LL (clockwise) on subtree rooted at C is performed first. By doing RR rotation, node C has become the right subtree of B.
	After performing LL rotation, node A is still unbalanced, i.e., having balance factor -2, which is because of the right-subtree of the right-subtree node A.
	Now we perform RR rotation (anticlockwise rotation) on full tree, i.e., on node A. node C has now become the right subtree of node B, and node A has become the left subtree of B.
	Balance factor of each node is now either -1, 0, or 1, i.e., BST is balanced now.