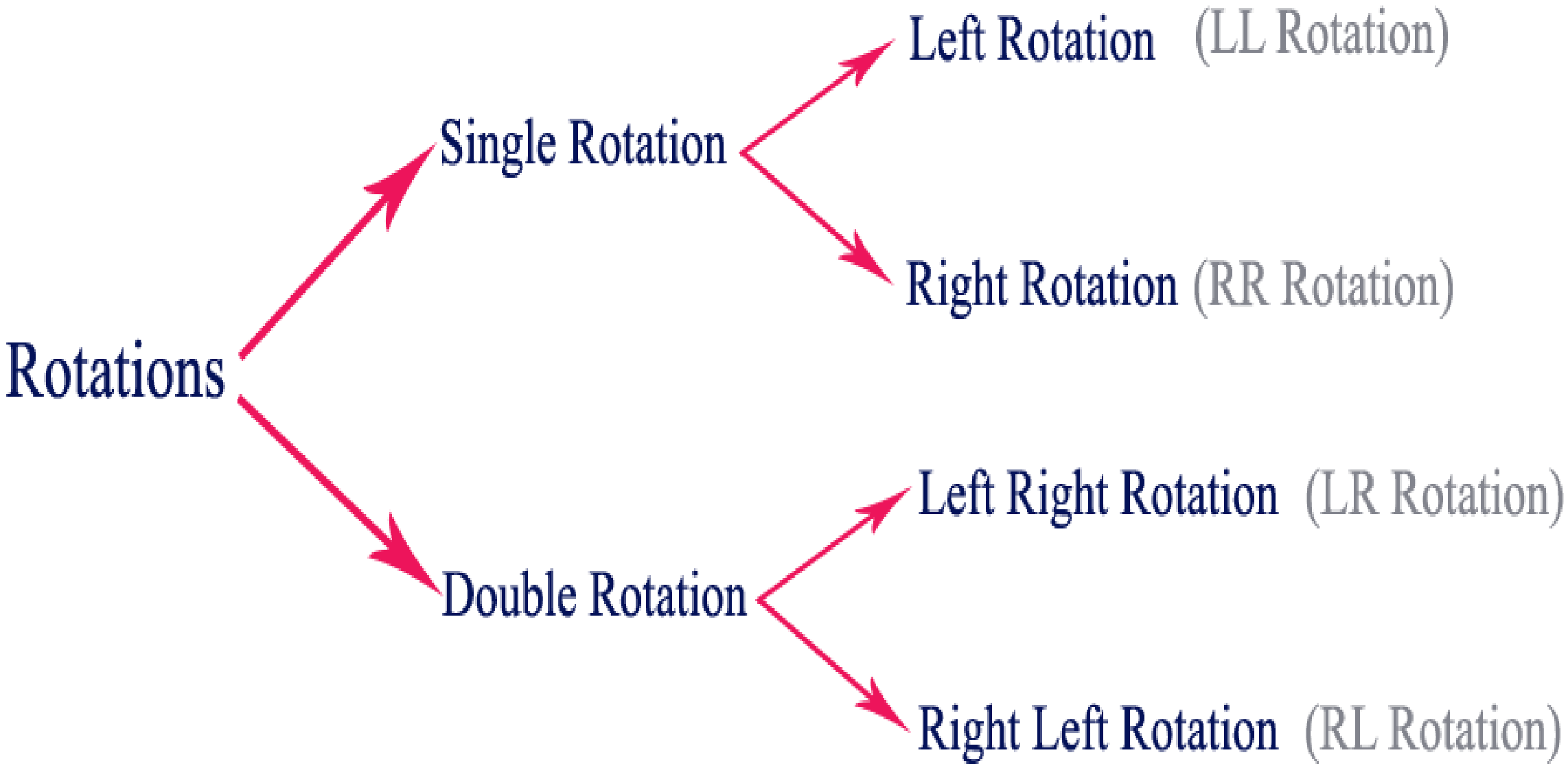
AVL Trees Insertion Oprerations

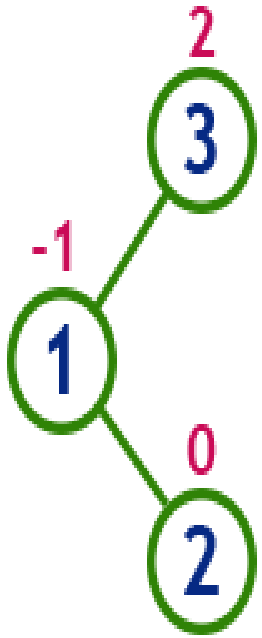
By
Aditya Tiwari
Assistant Professor
CSVTU Bhilai


Rotations
Single Rotation
Left Rotation (LL Rotation)
Right Rotation (RR Rotation)
Double Rotation
Left Right Rotation (LR Rotation)
Right Left Rotation (RL Rotation)

Left Right Rotation (LR Rotation)

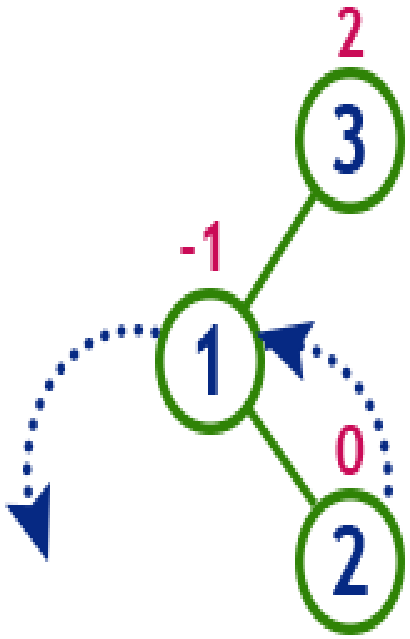
- The LR Rotation is a sequence of single left rotation followed by a single right rotation. In LR Rotation, at first, every node moves one position to the left and one position to right from the current position. To understand LR Rotation, let us consider the following insertion operation in AVL Tree...

insert 3, 1 and 2



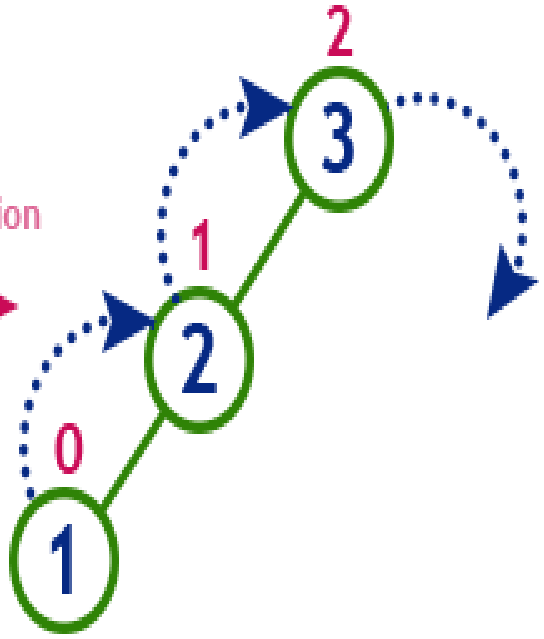
Tree is imbalanced

because node 3 has balance factor 2



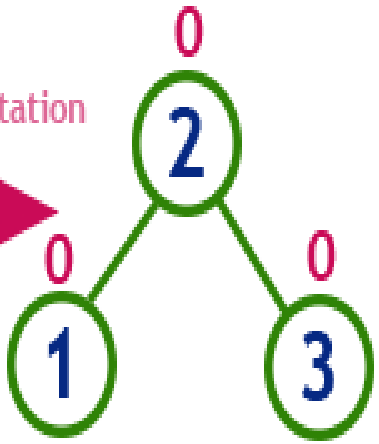
LL Rotation

After LL Rotation



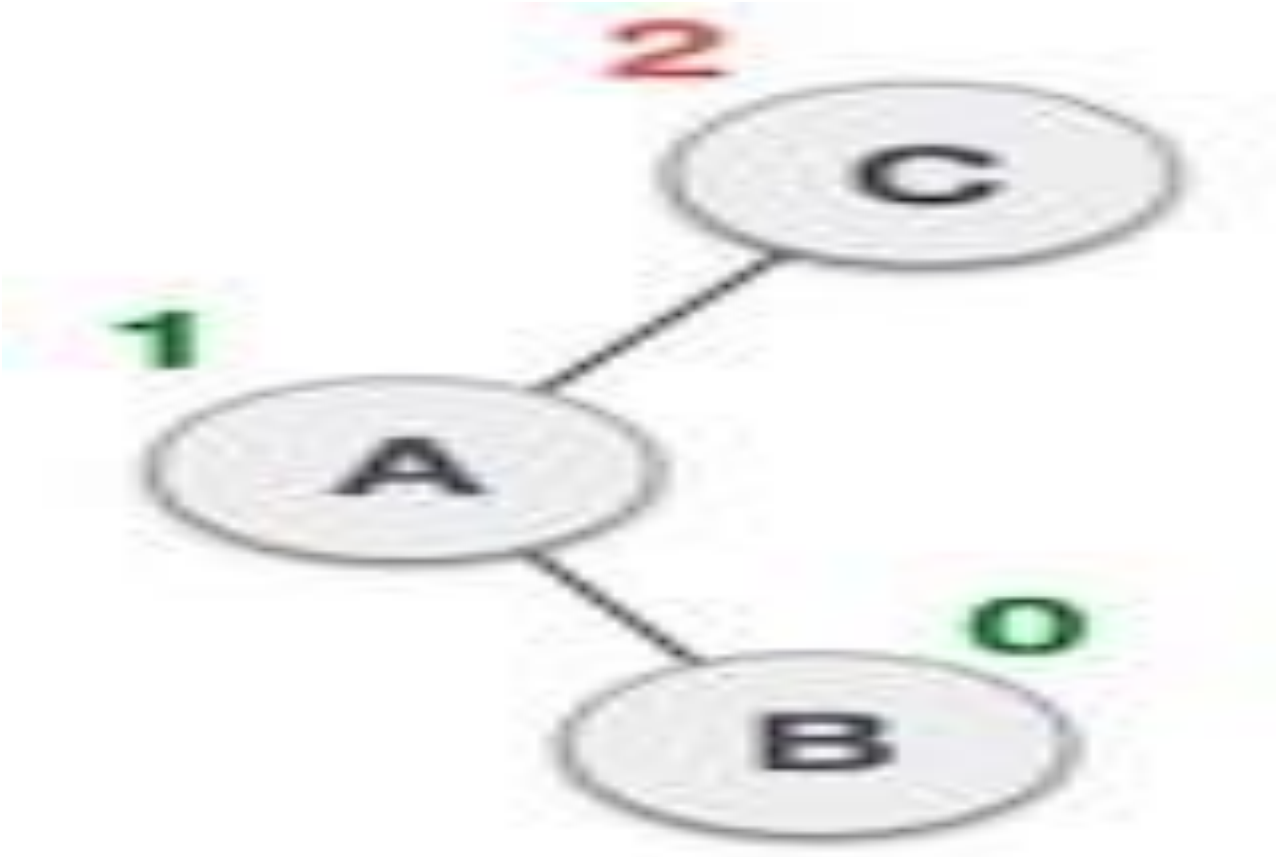
RR Rotation

After RR Rotation

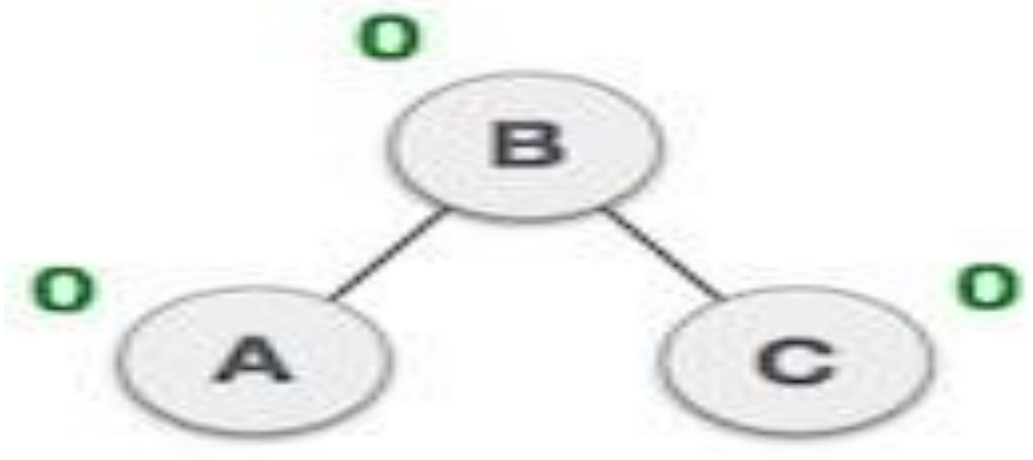


After LR Rotation
Tree is Balanced

- Double rotations are slightly complex version of already explained versions of rotations. To understand them better, we should take note of each action performed while rotation. Let's first check how to perform Left-Right rotation. A left-right rotation is a combination of left rotation followed by right rotation.



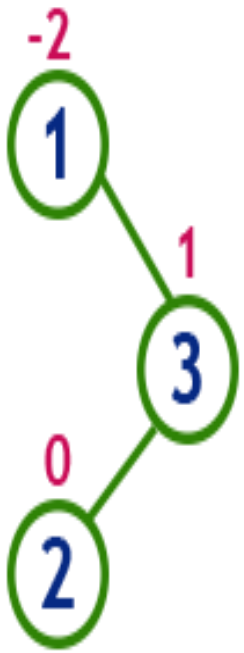
- A node has been inserted into the right subtree of the left subtree. This makes **C** an unbalanced node. These scenarios cause AVL tree to perform left-right rotation.
- We first perform the left rotation on the left subtree of **C**. This makes **A**, the left subtree of **B**.
- Node **C** is still unbalanced, however now, it is because of the left-subtree of the left-subtree.
- We shall now right-rotate the tree, making **B** the new root node of this subtree. **C** now becomes the right subtree of its own left subtree.
- The tree is now balanced.



Right Left Rotation (RL Rotation)

- The RL Rotation is sequence of single right rotation followed by single left rotation. In RL Rotation, at first every node moves one position to right and one position to left from the current position. To understand RL Rotation, let us consider the following insertion operation in AVL Tree...

insert 1, 3 and 2



Tree is imbalanced

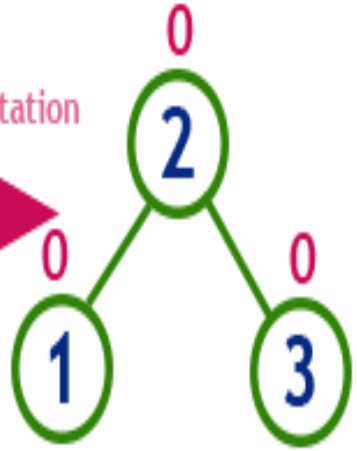
because node 1 has balance factor -2

RR Rotation

After RR Rotation

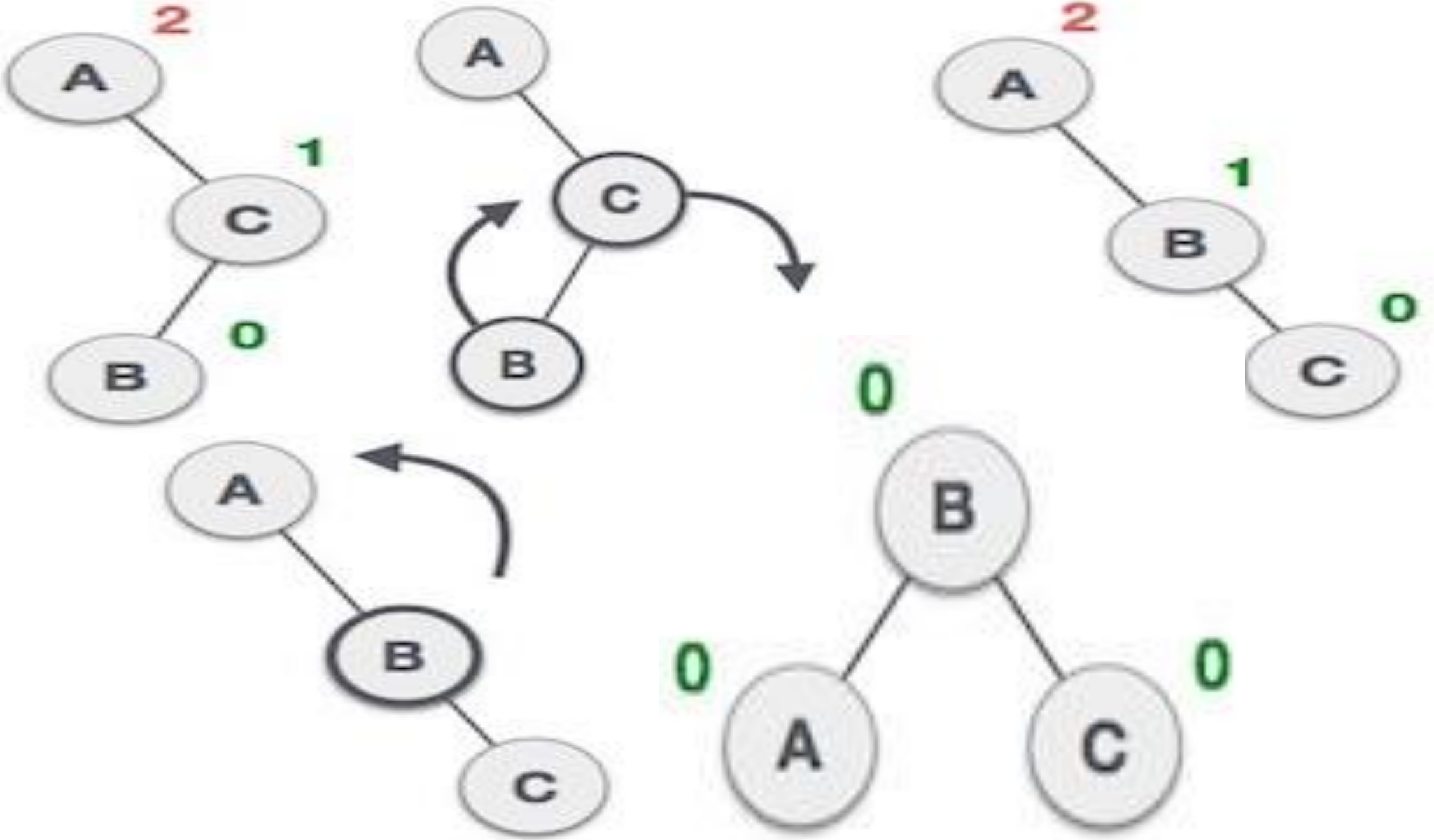
LL Rotation

After LL Rotation



After RL Rotation
Tree is Balanced

- The second type of double rotation is Right-Left Rotation. It is a combination of right rotation followed by left rotation.
- A node has been inserted into the left subtree of the right subtree. This makes **A**, an unbalanced node with balance factor 2.
- First, we perform the right rotation along **C** node, making **C** the right subtree of its own left subtree **B**. Now, **B** becomes the right subtree of **A**.
- Node **A** is still unbalanced because of the right subtree of its right subtree and requires a left rotation.
- A left rotation is performed by making **B** the new root node of the subtree. **A** becomes the left subtree of its right subtree **B**.
- The tree is now balanced.


2
A
1
C
0
B
A
C
B
2
A
1
B
0
C
A
B
C
0
B
0
A
0
C

Operations on an AVL Tree

- The following operations are performed on AVL tree...
- 1.Search
- 2.Insertion
- 3.Deletion

Search Operation in AVL Tree

- In an AVL tree, the search operation is performed with O(log n) time complexity. The search operation in the AVL tree is similar to the search operation in a Binary search tree. We use the following steps to search an element in AVL tree...
- Step 1 - Read the search element from the user.
- Step 2 - Compare the search element with the value of root node in the tree.
- Step 3 - If both are matched, then display "Given node is found!!!" and terminate the function.
- Step 4 - If both are not matched, then check whether search element is smaller or larger than that node value.

- Step 5 - If search element is smaller, then continue the search process in left subtree.
- Step 6 - If search element is larger, then continue the search process in right subtree.
- Step 7 - Repeat the same until we find the exact element or until the search element is compared with the leaf node.
- Step 8 - If we reach to the node having the value equal to the search value, then display "Element is found" and terminate the function.
- Step 9 - If we reach to the leaf node and if it is also not matched with the search element, then display "Element is not found" and terminate the function.

Insertion Operation in AVL Tree

- In an AVL tree, the insertion operation is performed with O(log n) time complexity. In AVL Tree, a new node is always inserted as a leaf node. The insertion operation is performed as follows...
- Step 1 - Insert the new element into the tree using Binary Search Tree insertion logic.
- Step 2 - After insertion, check the Balance Factor of every node.
- Step 3 - If the Balance Factor of every node is 0 or 1 or -1 then go for next operation.
- Step 4 - If the Balance Factor of any node is other than 0 or 1 or -1 then that tree is said to be imbalanced. In this case, perform suitable Rotation to make it balanced and go for next operation.

Example: Construct an AVL Tree by inserting numbers from 1 to 8.

insert 1

Tree is balanced

insert 2

Tree is balanced

insert 3

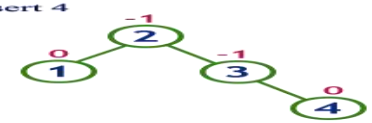
Tree is imbalanced

LL Rotation

After LL Rotation

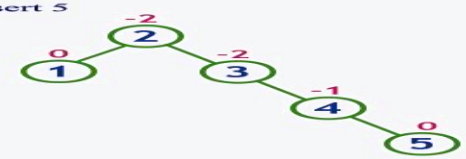
Tree is balanced

insert 4

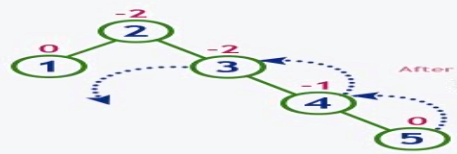


Tree is balanced

insert 5



Tree is imbalanced

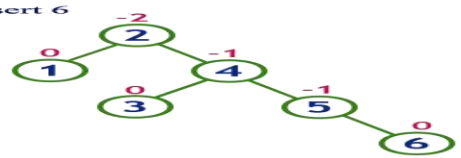


LL Rotation at 3

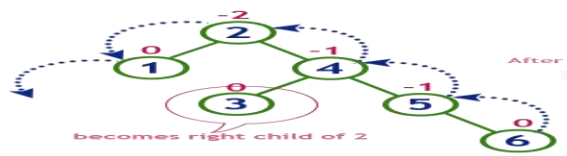
After LL Rotation at 3

Tree is balanced

insert 6



Tree is imbalanced



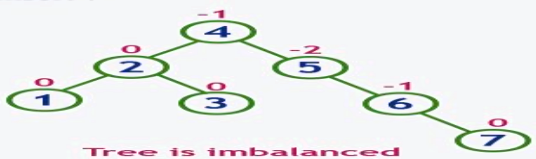
becomes right child of 2

LL Rotation at 2

After LL Rotation at 2

Tree is balanced

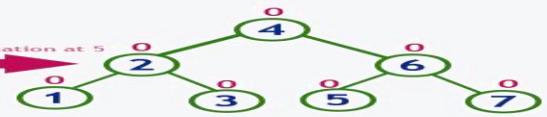
insert 7



Tree is imbalanced

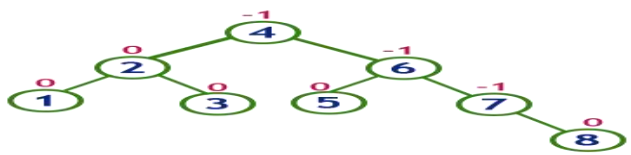
LL Rotation at 5

After LL Rotation at 5



Tree is balanced

insert 8



Tree is balanced

- THANK YOU