SNS COLLEGE OF TECHNOLOGY

(Autonomous)

COIMBATORE-35

AVL trees – Single Rotation

V.Savitha

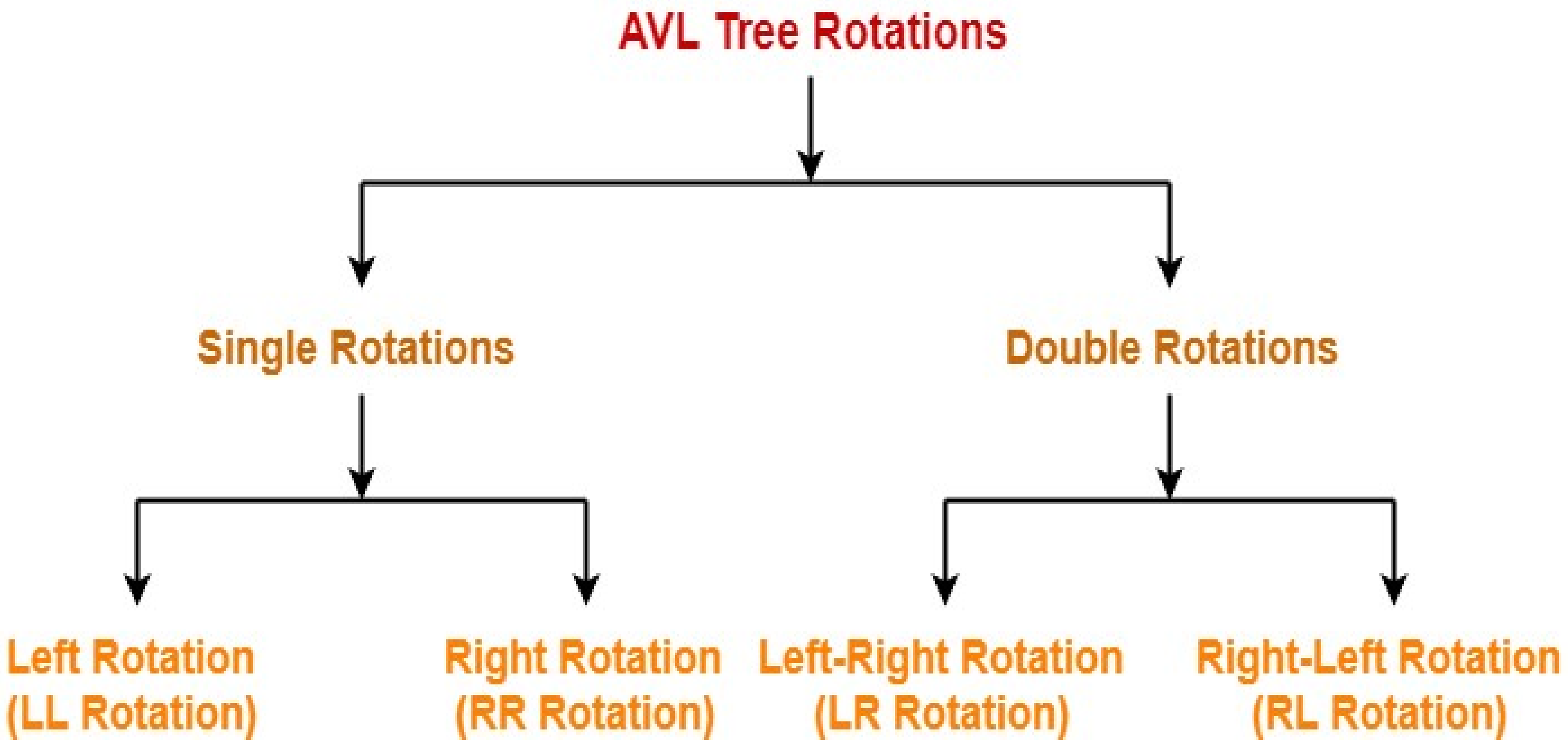
AP/CSE

Introduction

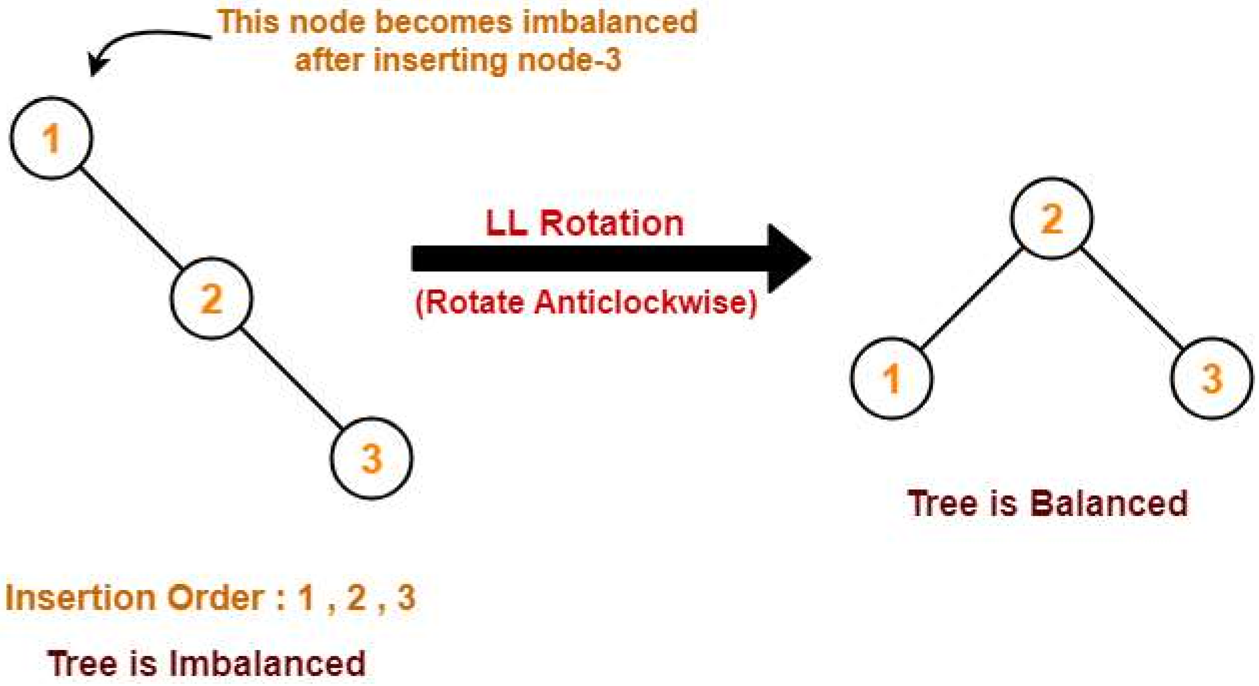
- AVL tree is **a self-balancing** Binary Search Tree (BST)

- Difference between heights of left and right subtree cannot be more than one for all nodes.

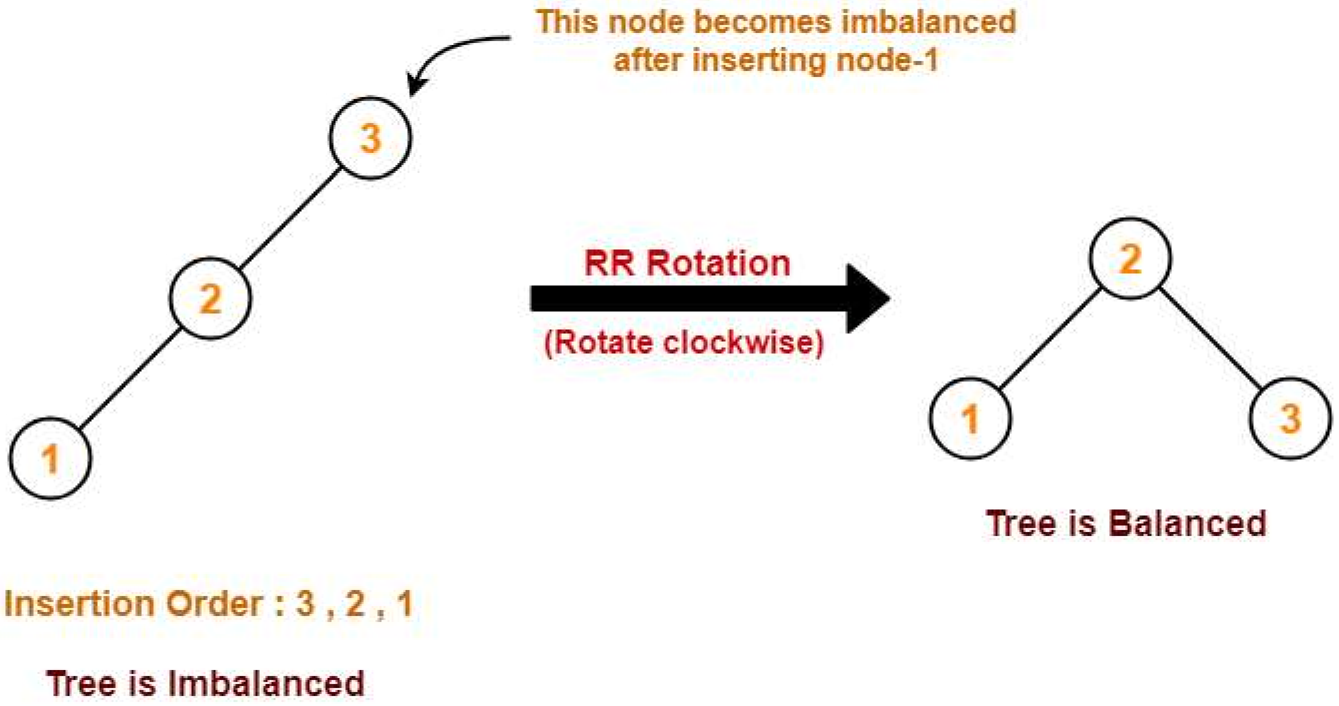
Types of Rotations



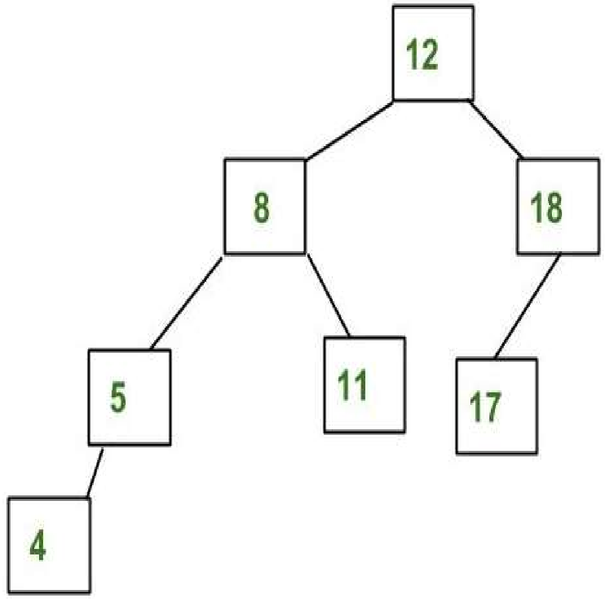
Cases Of Imbalance And Their Balancing Using Rotation Operations- Case 1

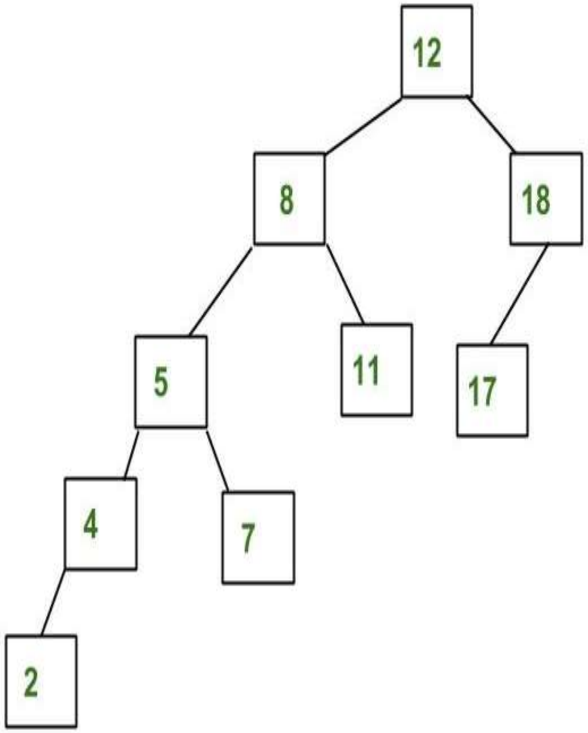

This node becomes imbalanced after inserting node-3
1
2
3
LL Rotation
(Rotate Anticlockwise)
2
1
3
Tree is Balanced
Insertion Order : 1 , 2 , 3
Tree is Imbalanced

Cases Of Imbalance And Their Balancing Using Rotation Operations- Case 2



Introduction - Examples





The above tree is AVL
Height difference between left and right subtree =1.

Is **not AVL**

Insertion - Rebalancing

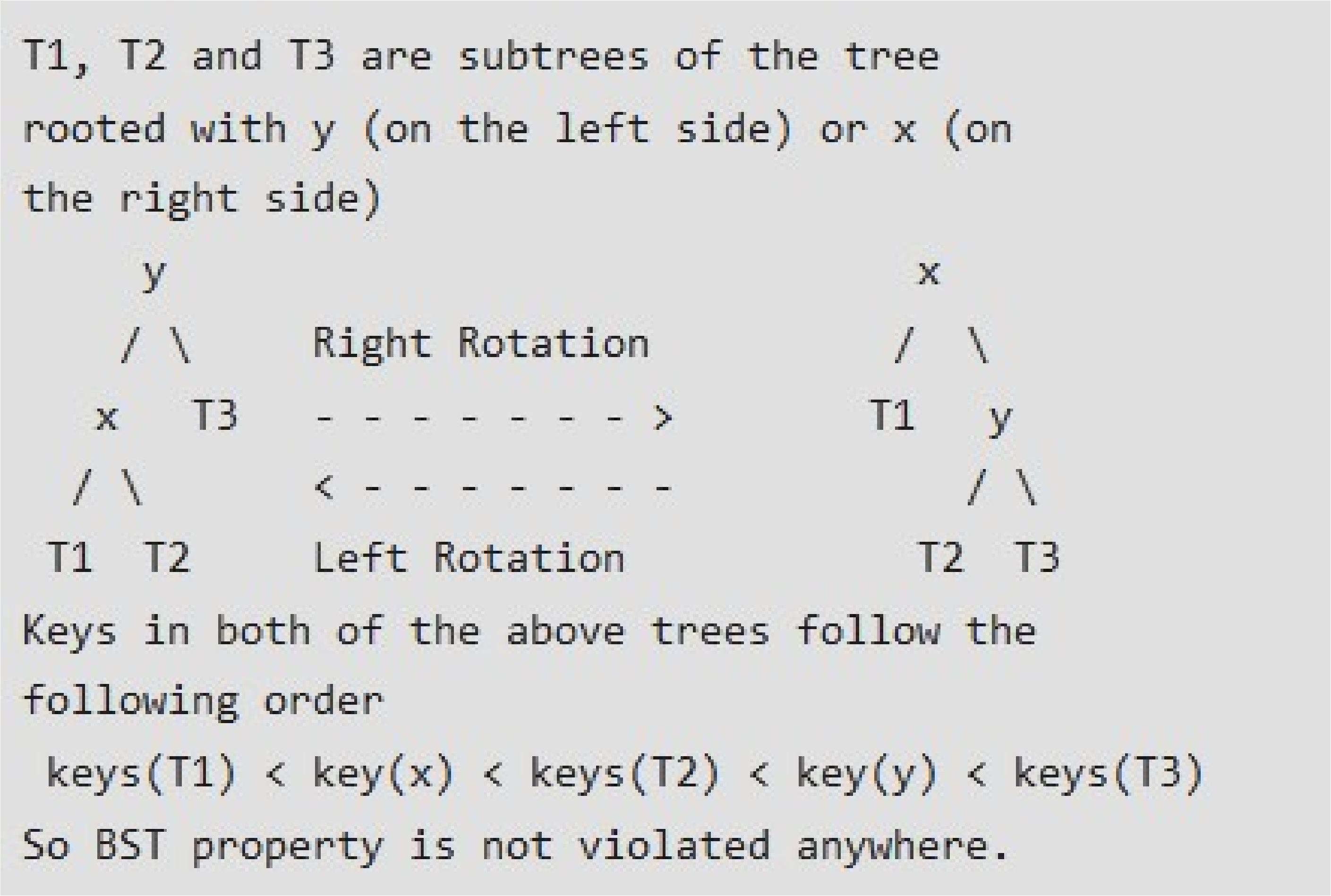
- Make sure that the given tree remains AVL after every insertion – Hence need to rebalance

- **(keys(left) < key(root) < keys(right))**

- To achieve this

  1) Left Rotation
  2) Right Rotation

Rebalancing of trees on insertion



```
T1, T2 and T3 are subtrees of the tree
rooted with y (on the left side) or x (on
the right side)
     y                               x
    / \     Right Rotation          /  \
   x   T3   - - - - - - - >        T1   y
  / \       < - - - - - - -            / \
 T1  T2     Left Rotation            T2  T3
Keys in both of the above trees follow the
following order
 keys(T1) < key(x) < keys(T2) < key(y) < keys(T3)
So BST property is not violated anywhere.
```