UNIT-2 BALANCED TREES

AVL- Trees

- Insertion
- Deletion

2-3- Trees

- Insertion
- Deletion

UNIT: 2 BALANCED TREES

AVL-TREES :

AVL tree is a self-balancing binary search tree invented by Adelson-velsky-Landis.

In an AVL Tree, the heights of the two sub-trees of a node may differ by at most one. Due to this property, the AVL Tree is also known as a "Height-Balanced-Tree".

The structure of an AVL Tree is same as that of binary search tree, and in additional it stores a variable called "Balance Factor". Thus every node has a balance factor associated with it.

The Balance Factor of a node is calculated by subtracting the height of its right sub tree from the height of its left sub tree.

"Balance Factor" = Height (left sub Tree) – Height (Right sub tree)

→ A Binary search TREE in which every node has a balance factor of "-1, 0, 1" is said to be height balanced.

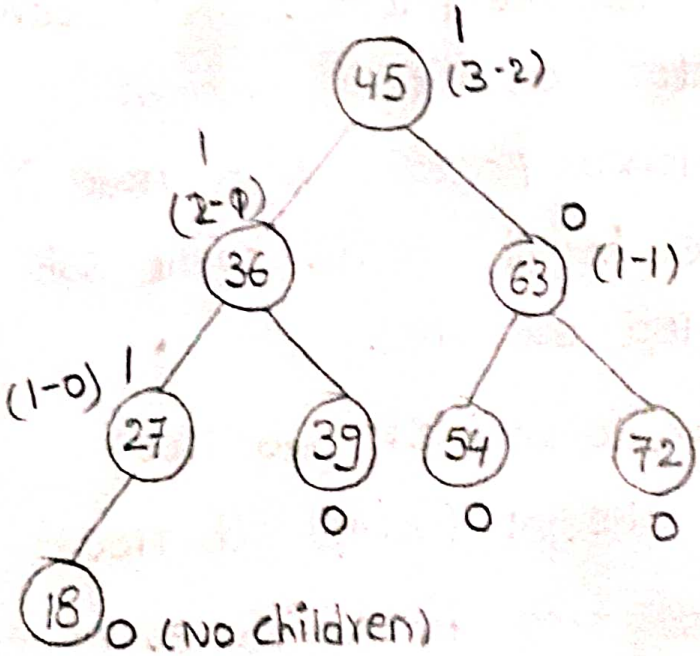
→ A node with any other balance factor is consider to be "unbalanced" and requires balancing of the tree.

* If the "balance factor" of a node is "1", then it means that the left-sub tree of the tree is one level higher than that of the right sub-tree. such a tree is therefore known as a "left-heavy tree"

* If the "balance factor" of a node is 0 then it means that the height of the left sub tree is equal to the height of the right sub tree.

* If the "balance factor" of a node is "-1" then it means that the left sub tree of the tree is one level Lower than that of the right sub tree. such a tree is therefore known as "right-heavy-tree".

Ex :-



There are 2 types of operations performed on AVL Tree.

1) Insertion
2) Deletion.

Insertion operation

In the AVL Tree, the new node is always inserted as the "leaf node".

But the step of Insertion is usually followed by an additional step of "rotation".

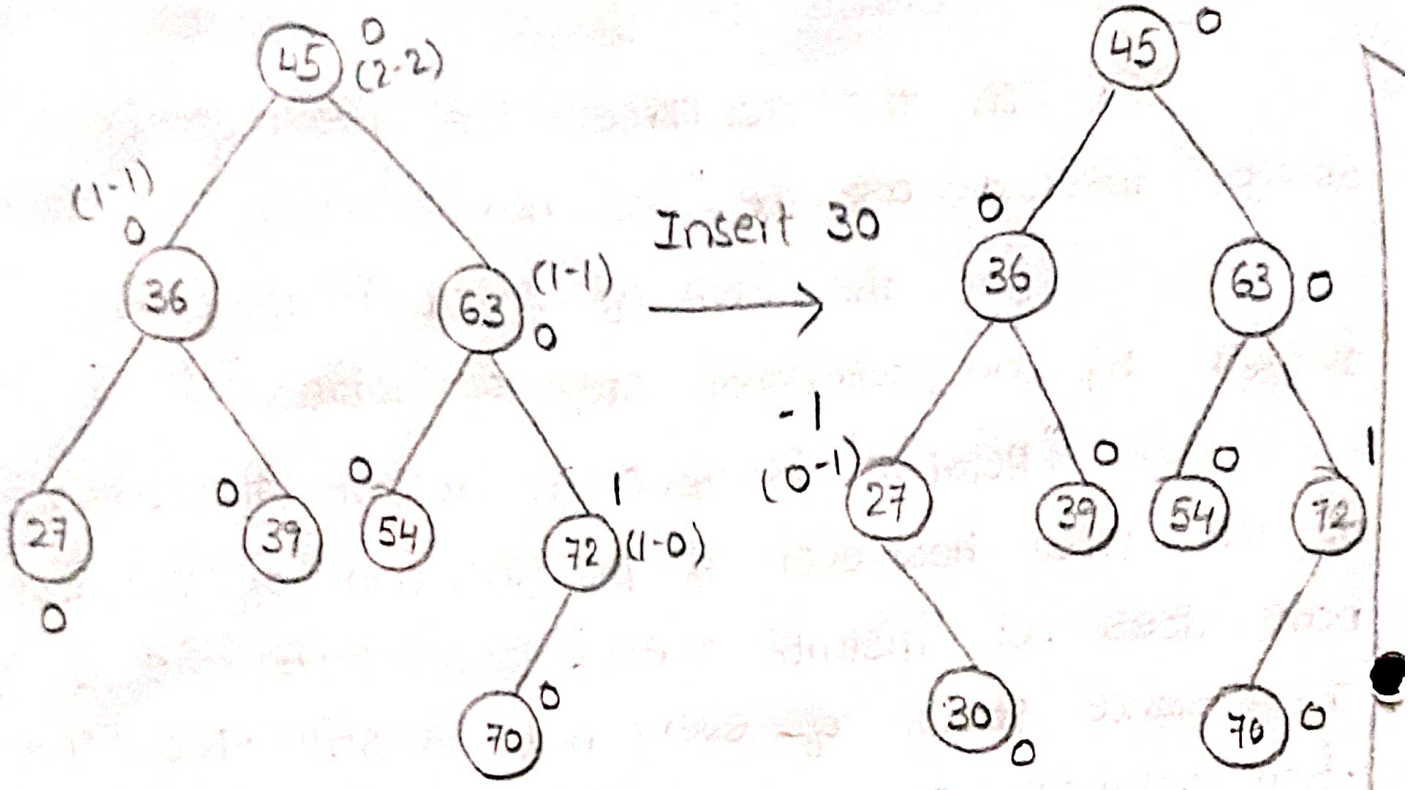
"Rotation" is done to restore the "balance" of the tree. How ever if the insertion of the new node does not disturb the balance factor i.e if the Balance Factor of every node is still -1, 0 or 1, then rotations are not required.

During Insertion, the new node is inserted as the leaf node, so it will always have Balance Factor equal to zero.

The only nodes whose balancing factor will change are those which lie in the path between the root of the tree, and the newly inserted node.

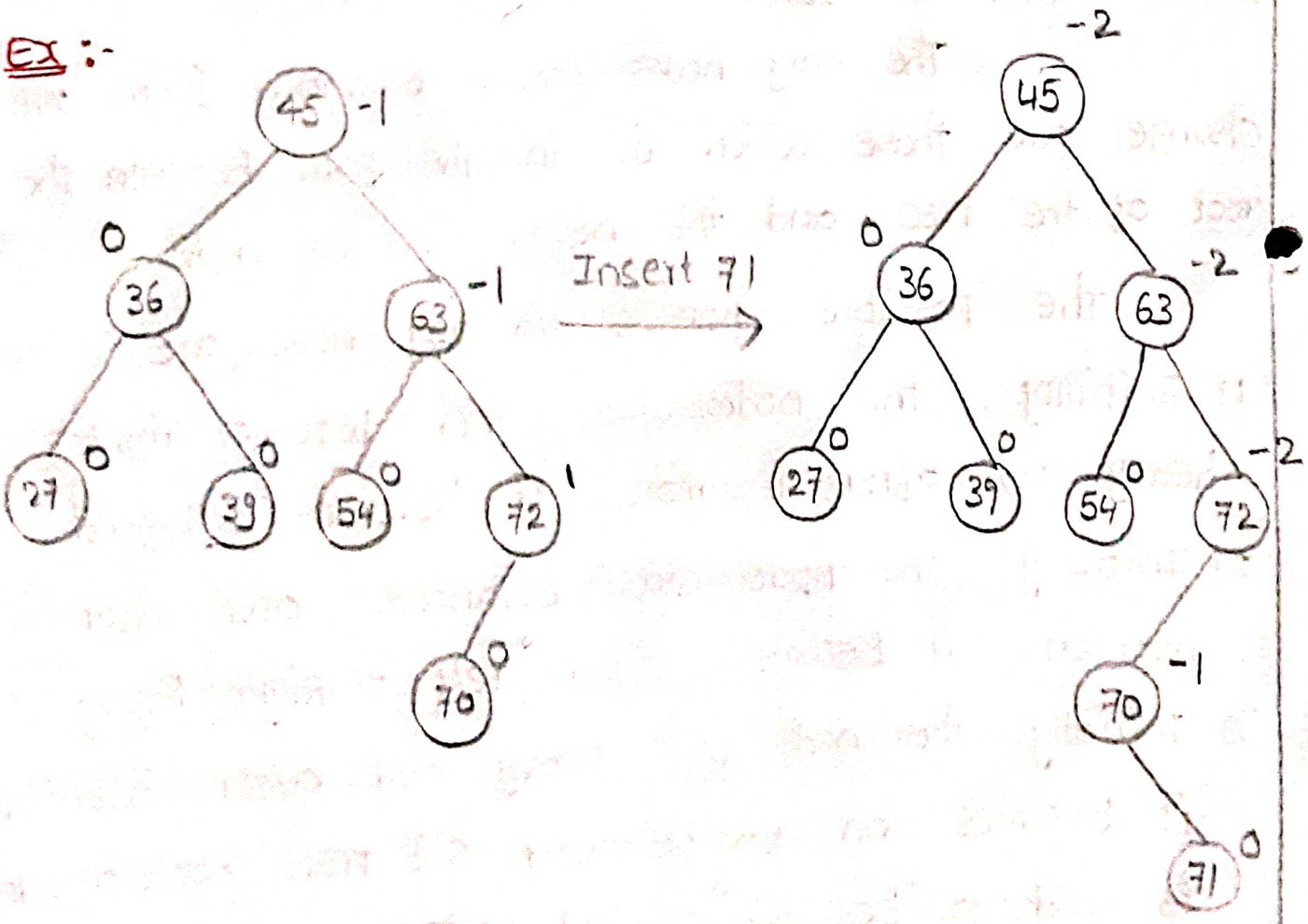
The possible changes in any node are

1) Initially, the node was either left- or right heavy and after insertion, it becomes "balanced".
2) Initially the node was balanced, and after insertion. it becomes either "left or Right heavy".
3) Initially the node was heavy and after insertion it becomes an un-balanced sub tree. Such a node is said to be a "critical node"



If we insert a new node with value 30 then the new tree will still be balanced and no rotations will be required.

Ex :-



There are 4 types of rebalancing rotations and application of these rotations depends on the position of the inserted node with reference to the critical node.

The 4 types of rotations are

1) LL Rotation : The new node is inserted in the left sub tree of the left sub tree of the critical node.

2). RR Rotation : The new node is inserted in the right sub tree of the right sub tree of the critical node

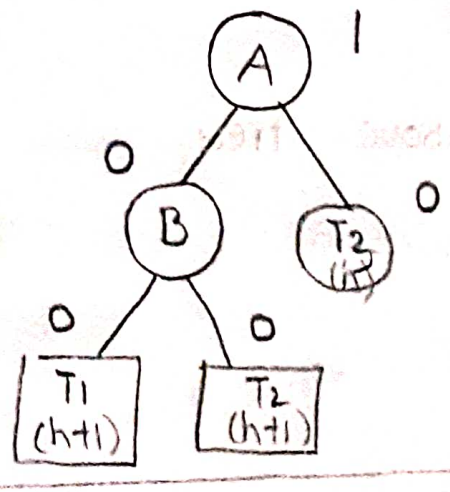
3) LR Rotation: The new node is inserted in the right sub tree of the left sub tree of the critical node

4) RL Rotation: The new node is inserted in the left sub tree of the right sub tree of critical node.

1) LL Rotation :

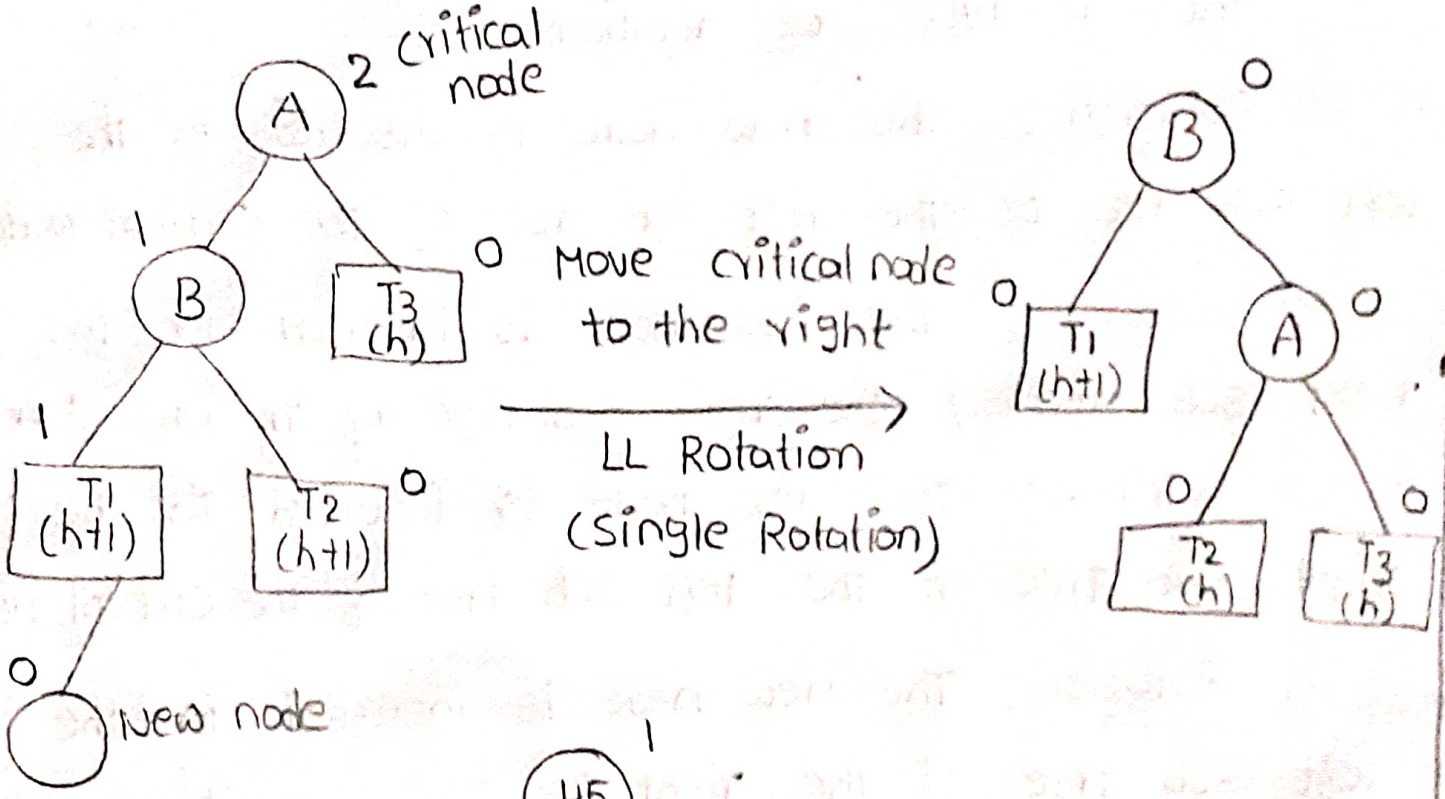
when the out of balance condition has been created by a left high subtree of a left high tree, we must balance the tree by rotating the out-of-balance node to the right.

Ex :-

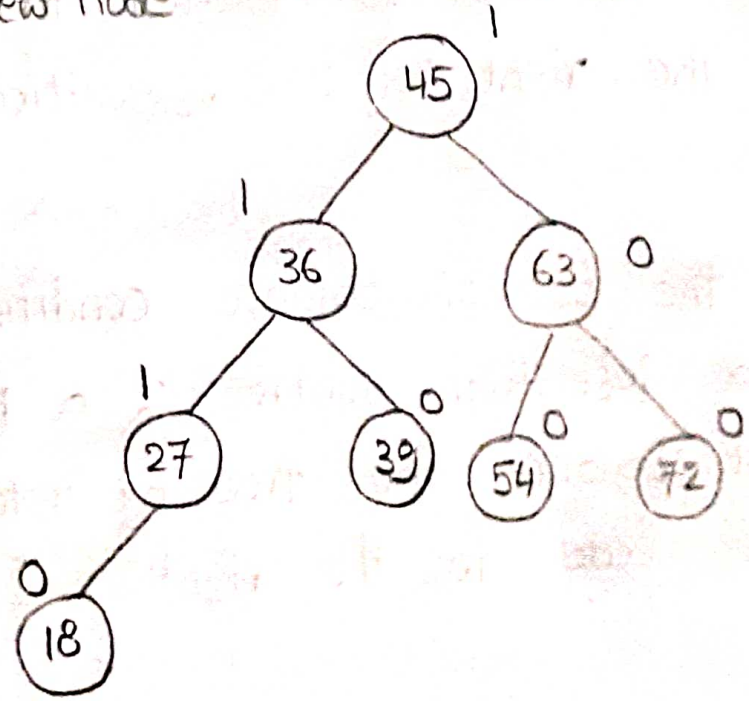


A new node is inserted in the left sub tree of the left sub tree of the critical node A.

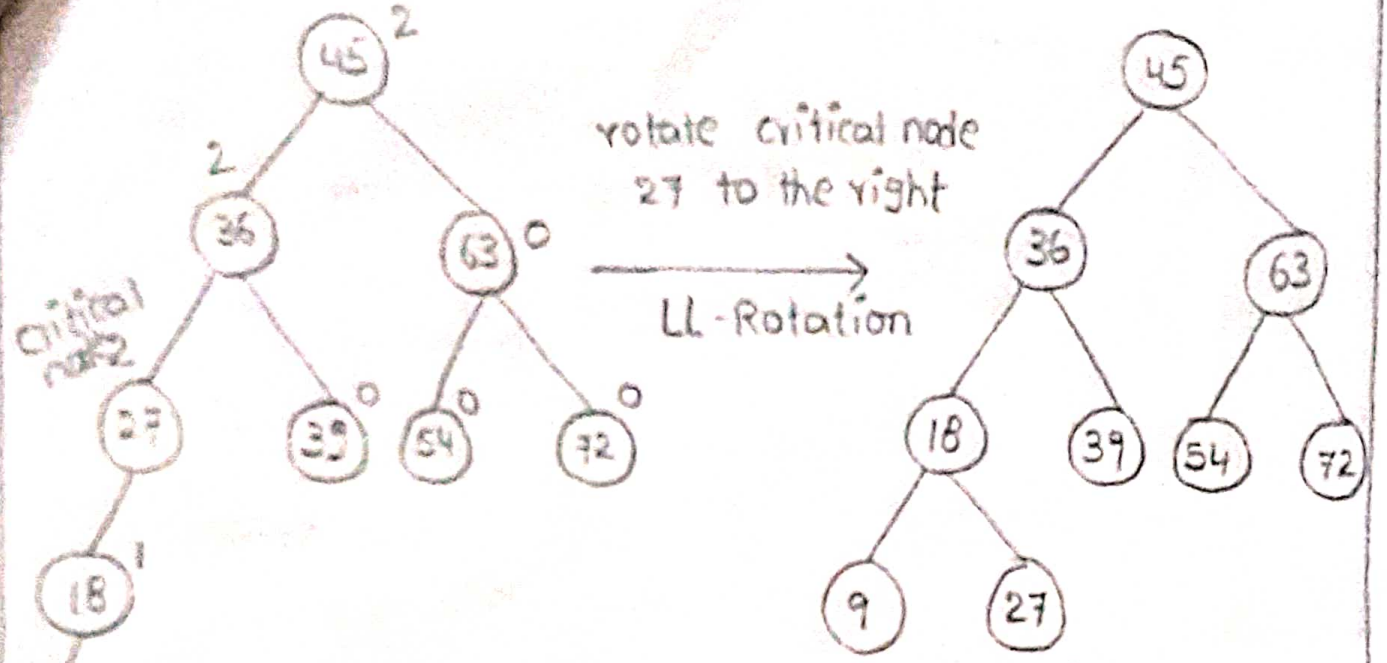
Node A is critical node because it is the closest ancestor whose balance factor is not -1, 1 & 0.



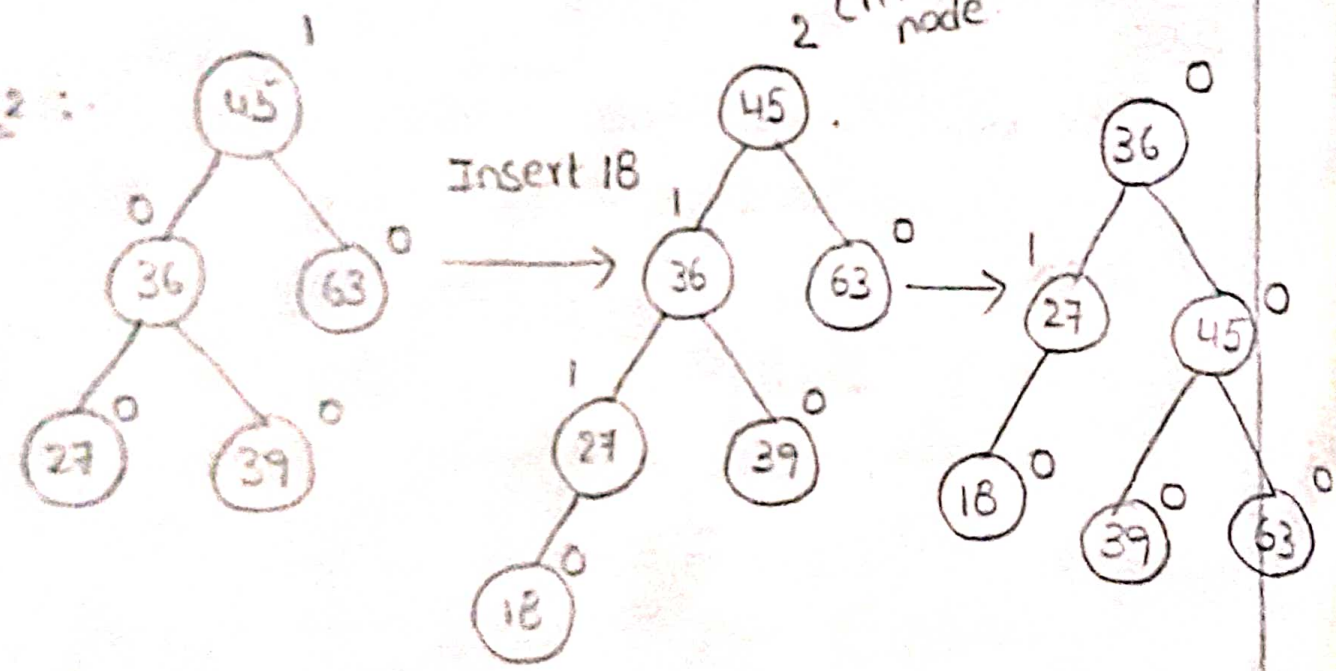
Ex:



Insert 9 to the above tree

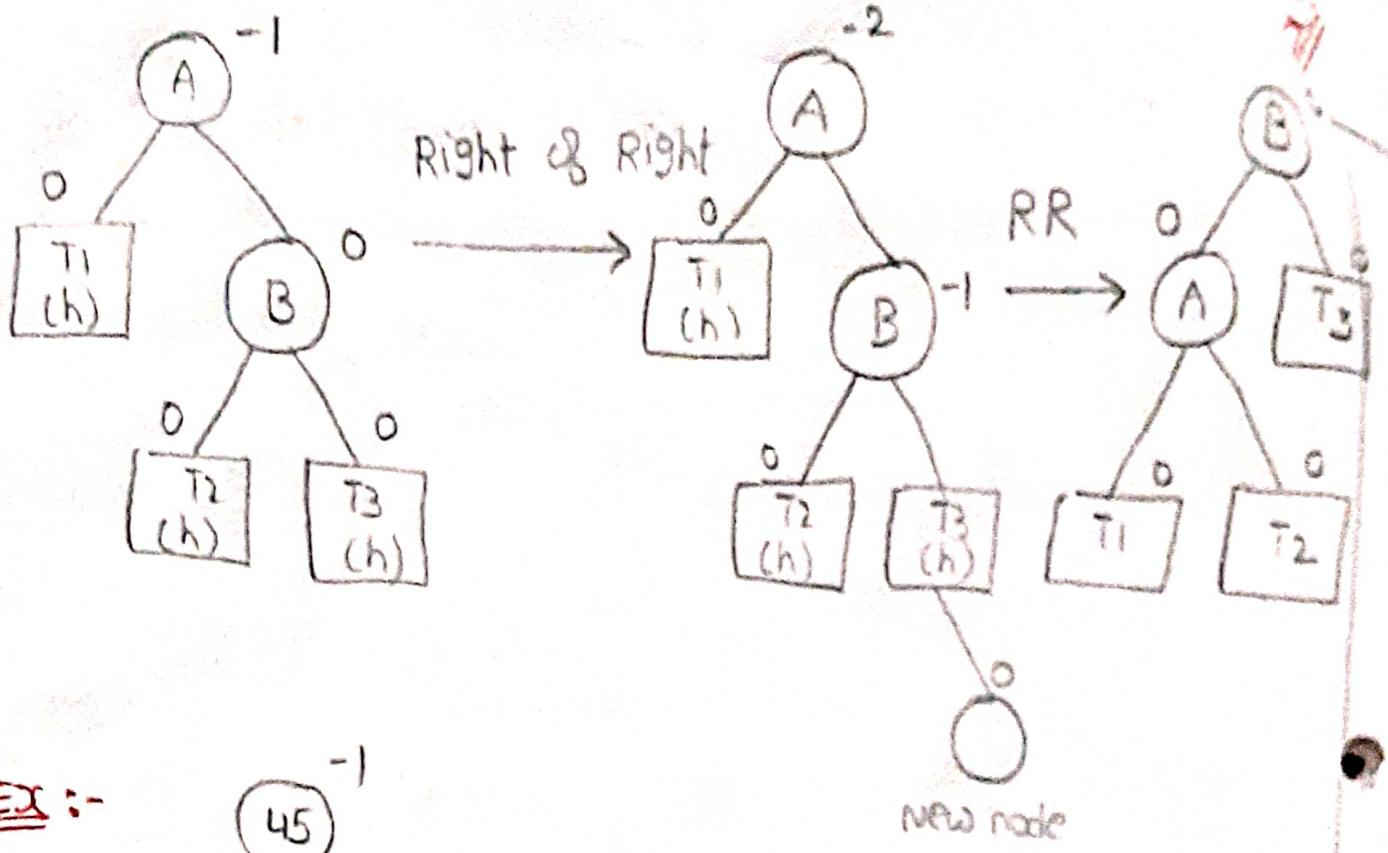


Ex 2 :

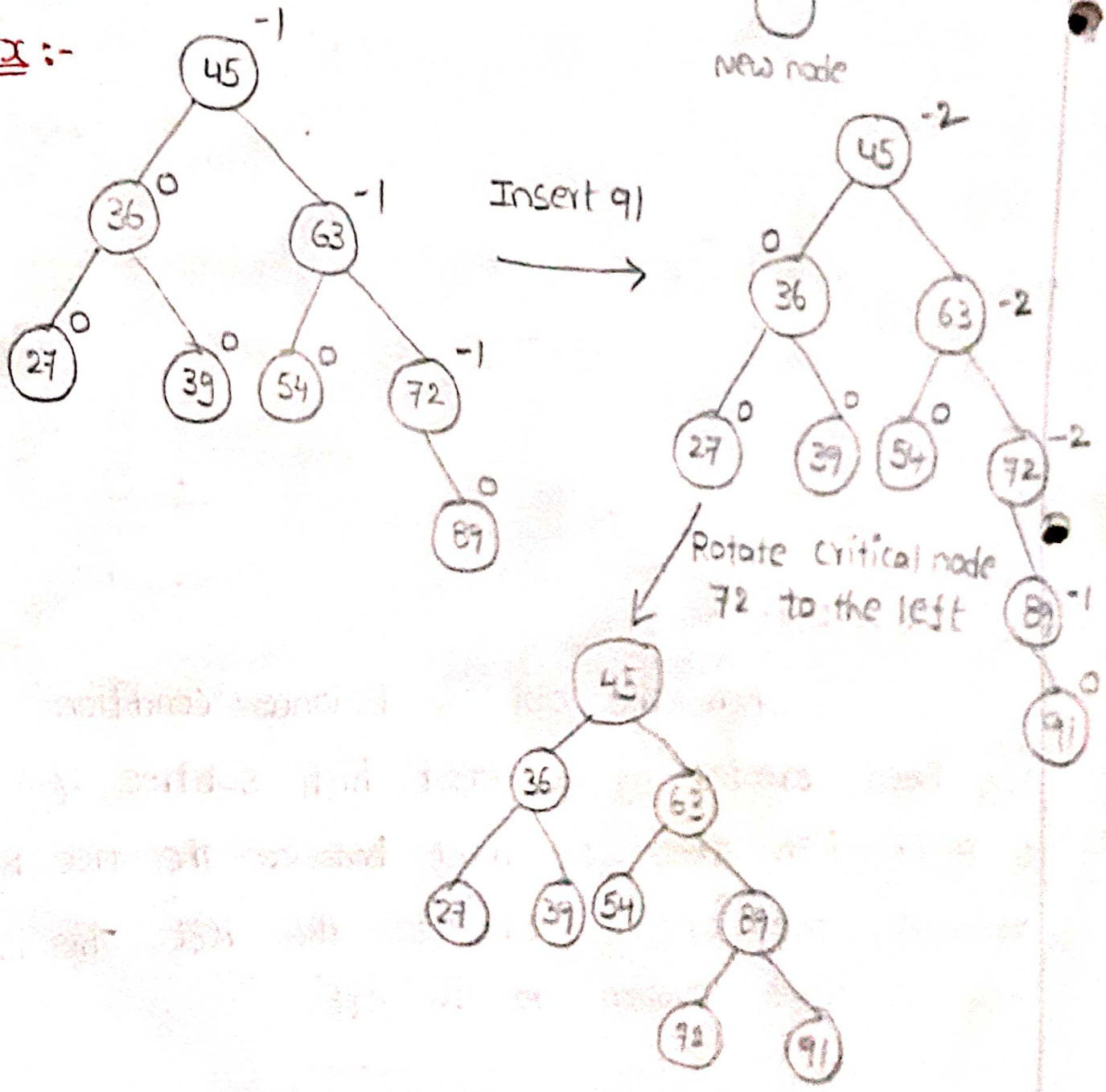


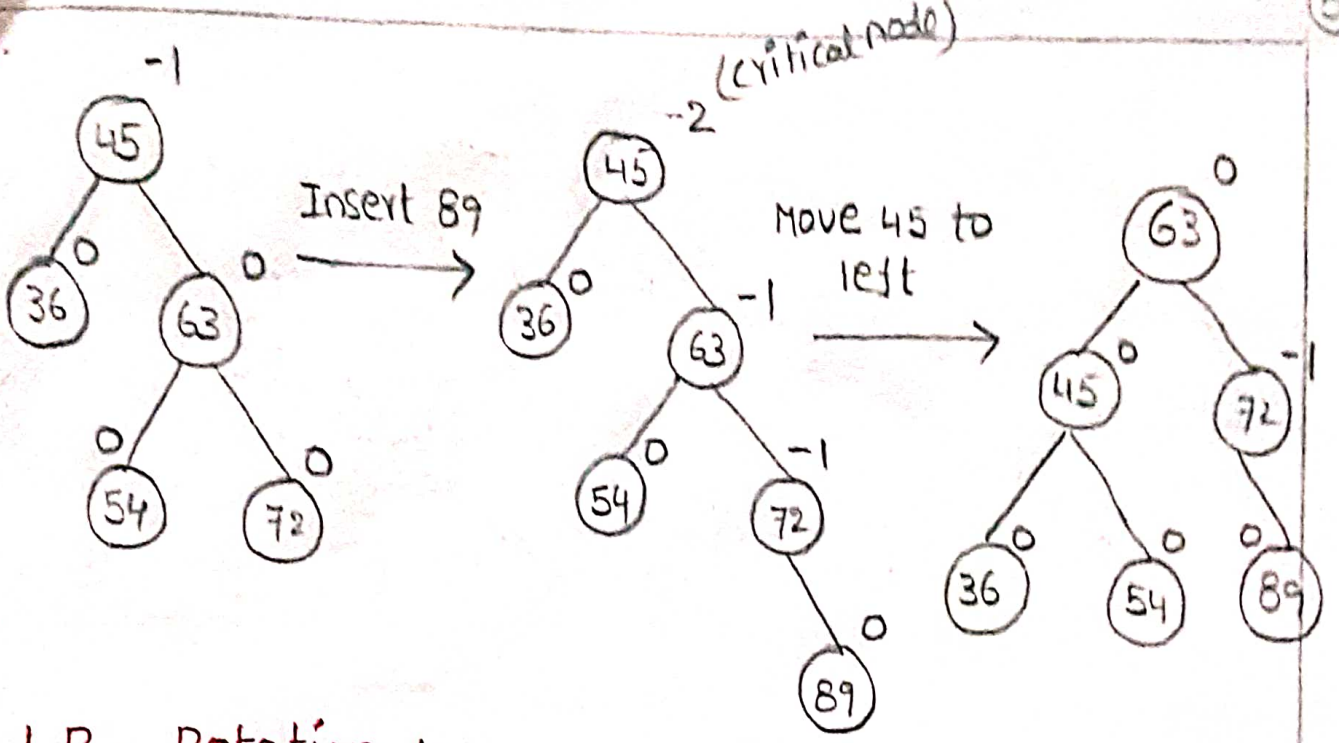
2) RR - Rotation :

when the out of balance condition has been created by a Right high subtree of a Right high tree, we must balance the tree by rotating the "critical node" to the "left". This is also a "single rotation" to the "left".


A
-1
0
T1
(h)
B
0
0
T2
(h)
0
T3
(h)
Right of Right
A
-2
0
T1
(h)
B
-1
0
T2
(h)
T3
(h)
0
New node
RR
B
0
A
0
T3
0
T1
0
T2

Ex :-


45
-1
36
0
63
-1
27
0
39
0
54
0
72
-1
89
0
Insert 91
45
-2
36
0
63
-2
27
0
39
0
54
0
72
-2
89
-1
91
0
Rotate critical node
72 to the left
45
36
63
27
39
54
89
72
91



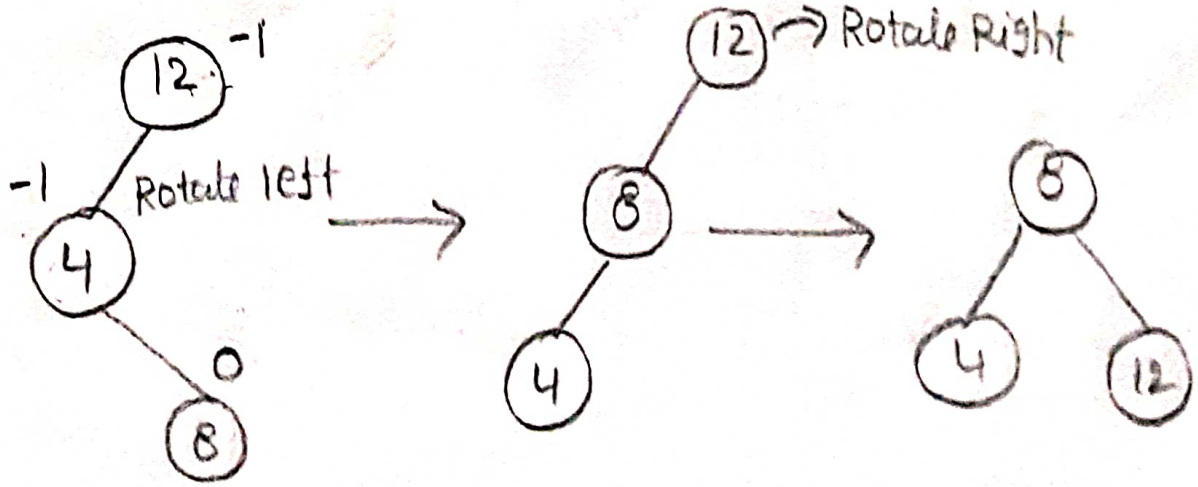
3) LR - Rotation :-

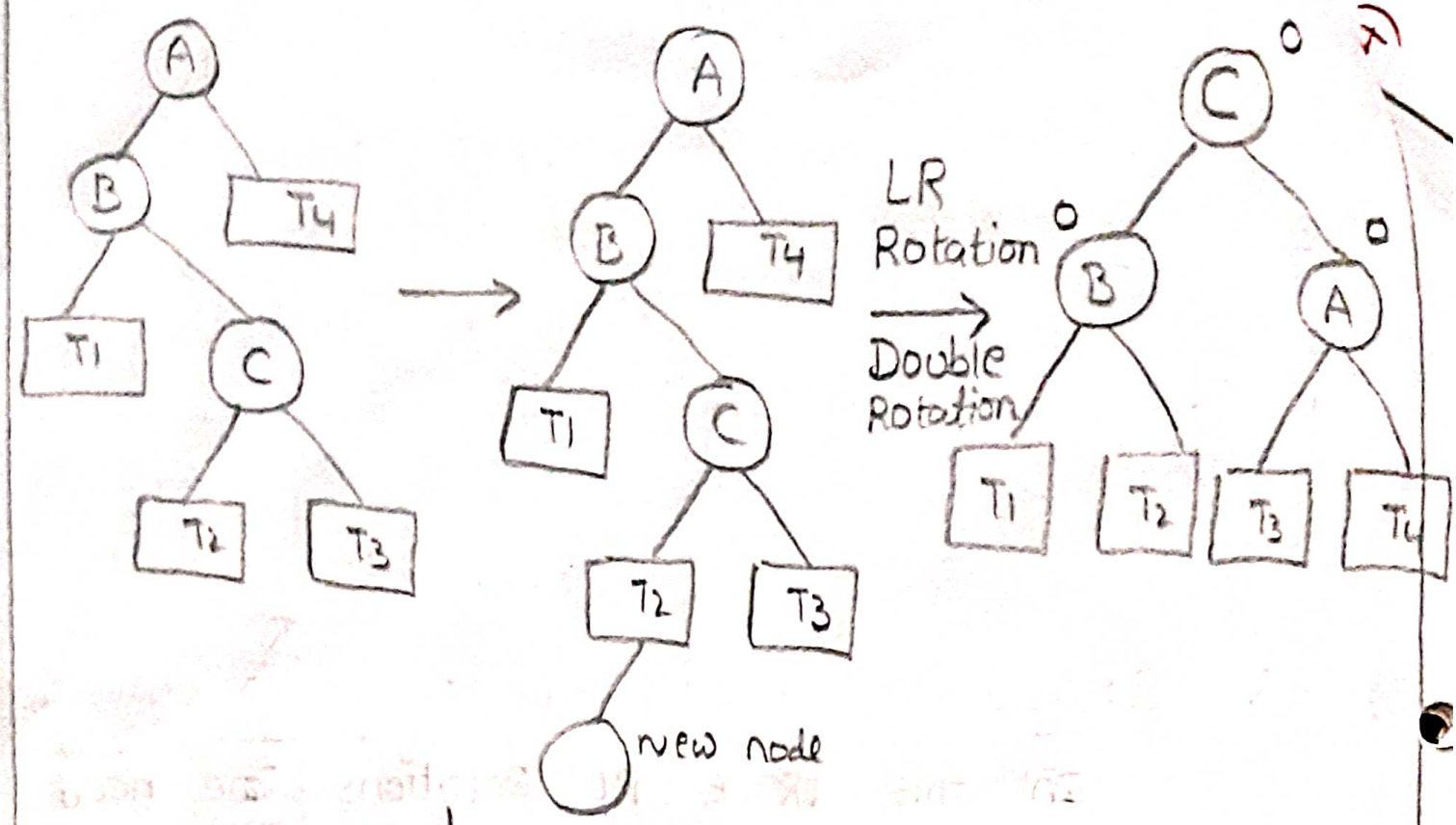
In this LR & RL Rotations we need to rotate two nodes. one to the left and one to the right. to balance the tree.

The out of balance tree in which the root is left high and the left sub tree is right high then it is a Right of left Tree.

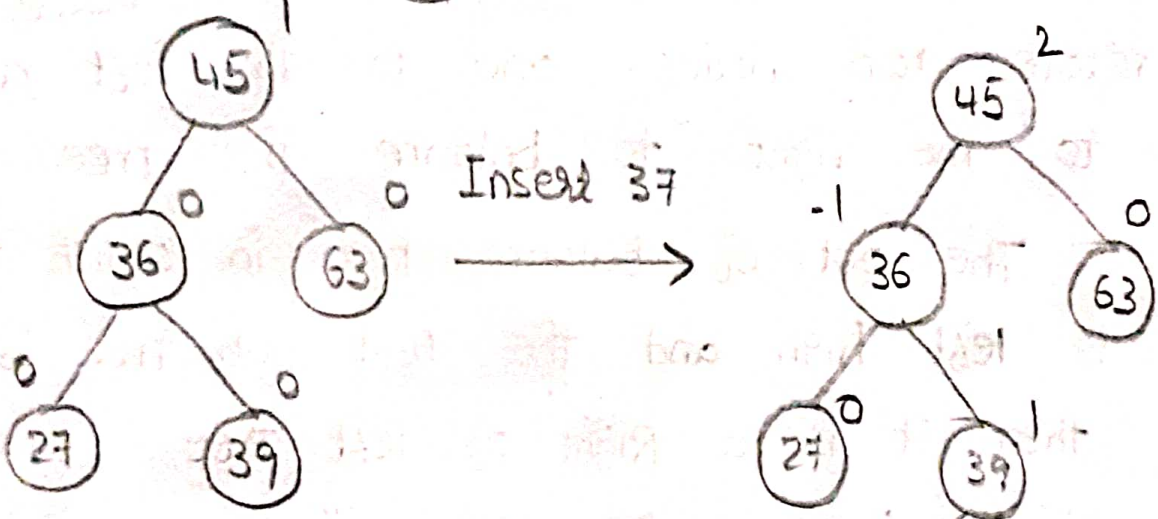
To balance this tree we first rotate the "left sub tree" to the "left", then we rotate "root" to the "right", making the "left node" the new "root"

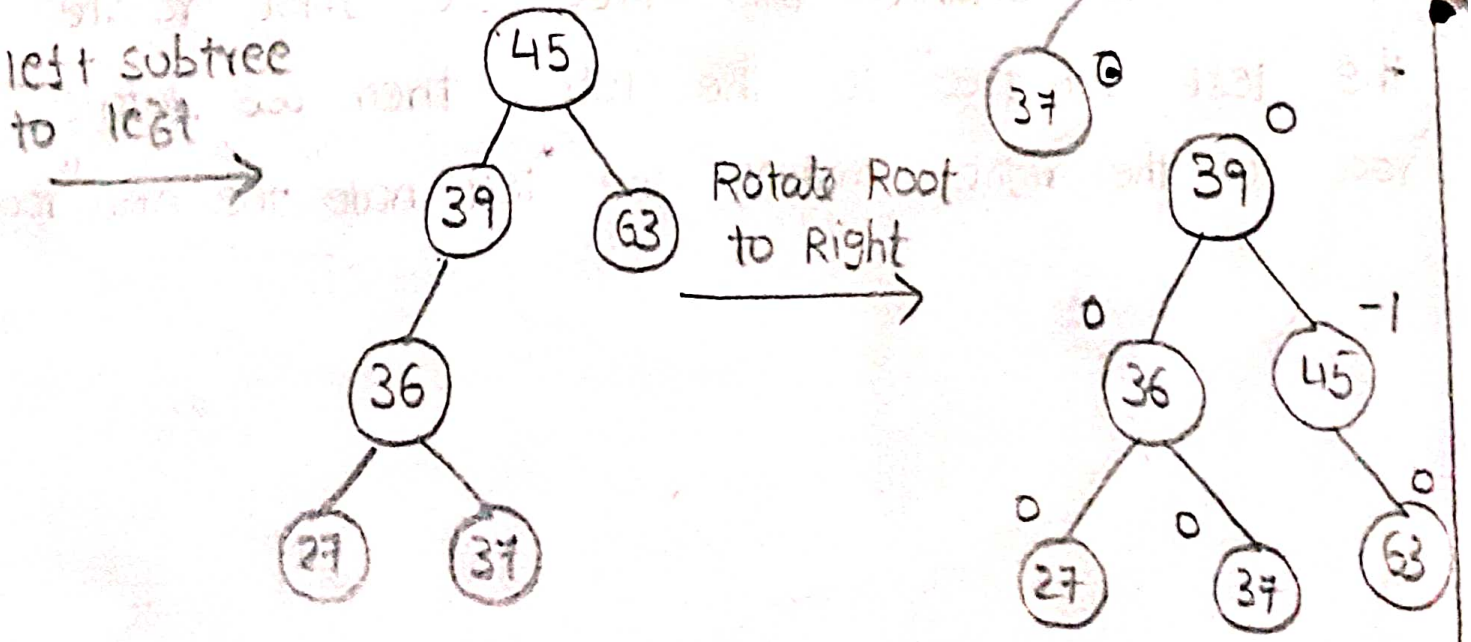
Ex :-




A
B
T4
T1
C
T2
T3
LR
Rotation
Double
Rotation
new node

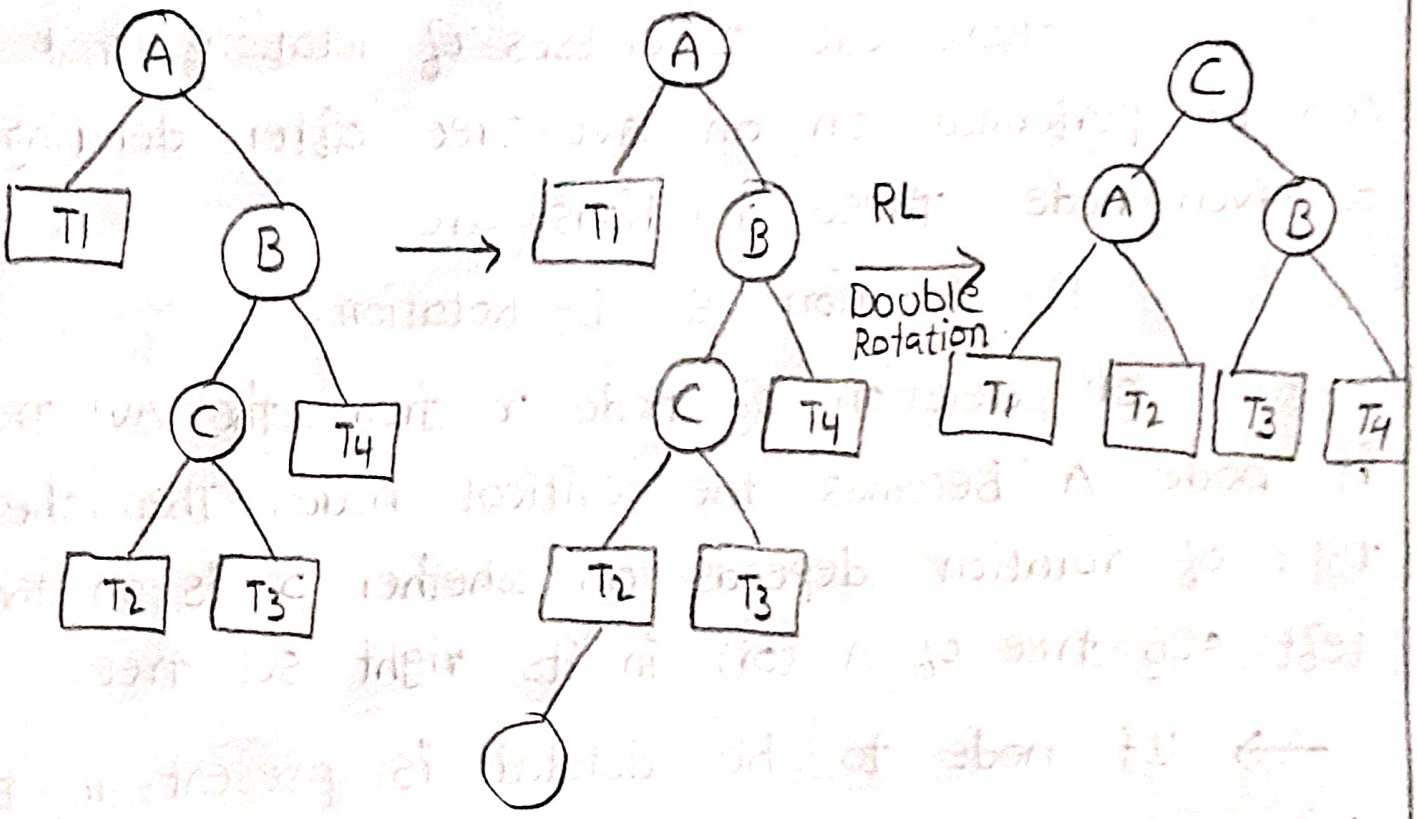
Ex:-


45
36
63
27
39
Insert 37

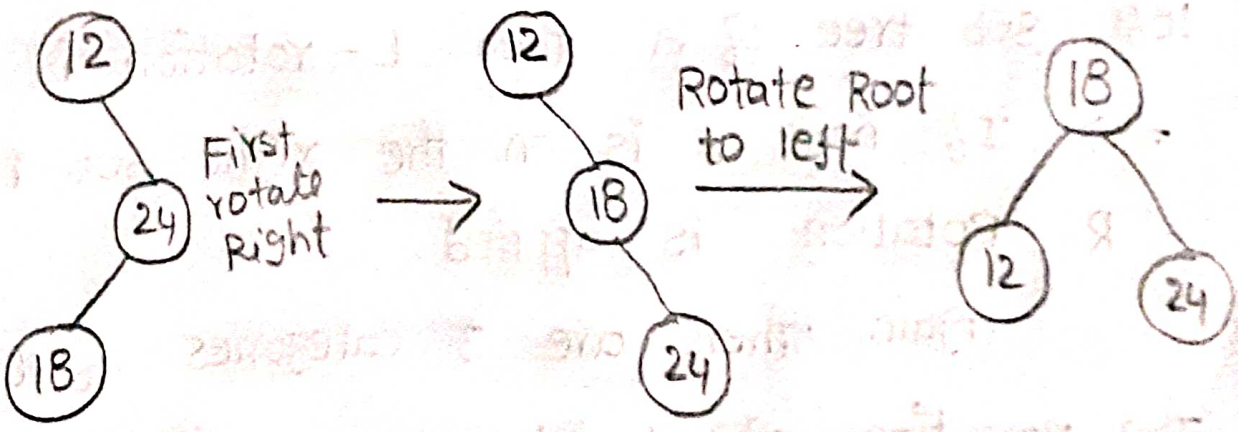

left subtree
to left
45
39
63
36
27
37
Rotate Root
to Right

4) RL - Rotation :

In this RL Rotation to balance the tree first rotate the right sub tree right and then rotate the root left.



Ex :-



AVL Tree - Deletion :

Deletion of a tree may disturb the AVLness of the tree, so to rebalance the AVL tree we need to perform rotations.

There are 2 classes of rotations that can be performed on an AVL Tree after deleting a given node. These rotations are

R- Rotation & L- Rotation.

on deletion of node x from the AVL Tree if node A becomes the critical node. Then the type of rotation depends on whether x is on the left sub tree of A (or) in its right sub tree.

→ If node to be deleted is present in the left sub tree of A then L- rotation is applied.

→ If node x is on the right sub tree, then R- Rotation is applied.

Again there are 3 categories of L & R rotation

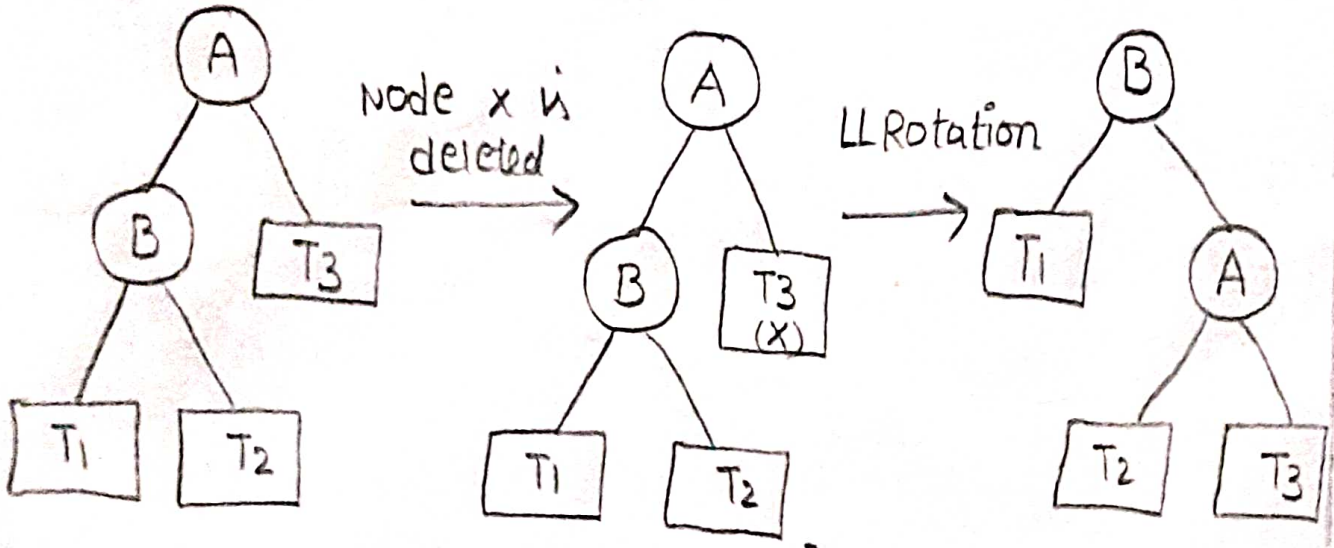
The variations of L rotation are L-1 rotation
Lo rotation &
L1 rotation

The variations of R- rotation are R-1 rotation
Ro rotation
R1 rotation.

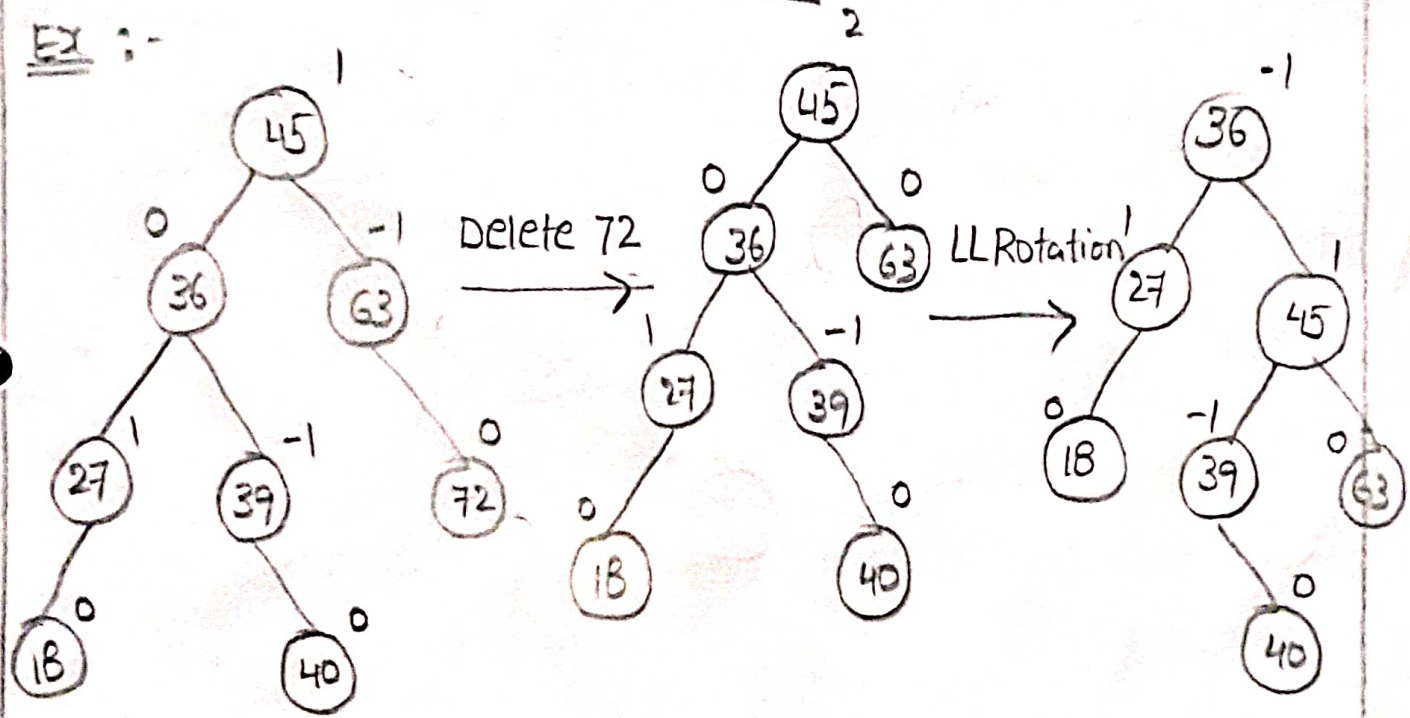
1) R0 Rotation :

let B be the root of the left & Right sub tree of A (critical node).

R0 - Rotation is applied if the Balance Factor of B is "0".



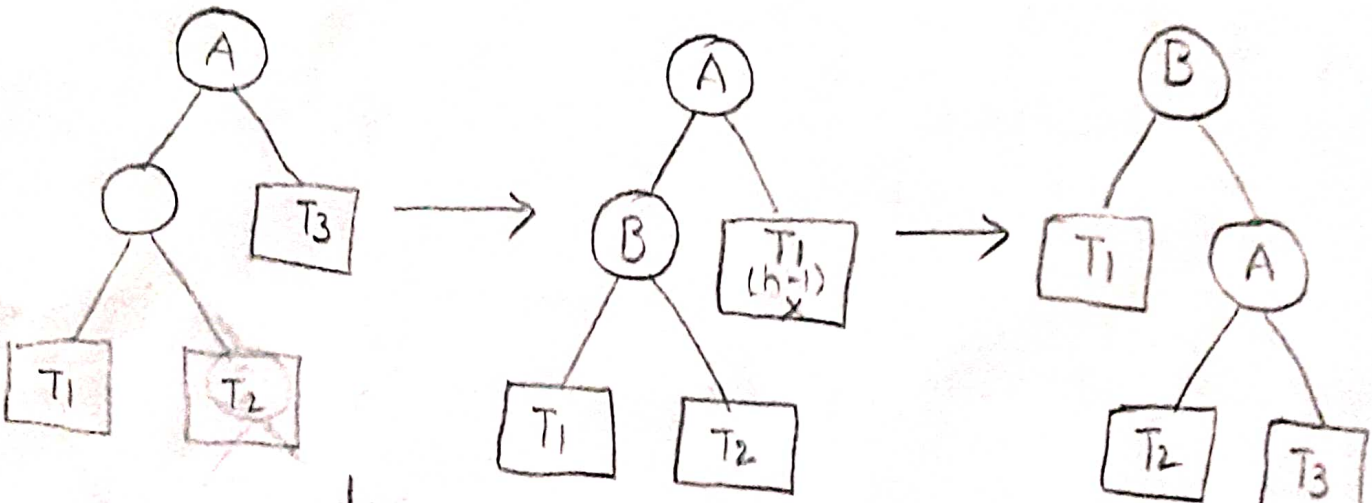
Ex :-



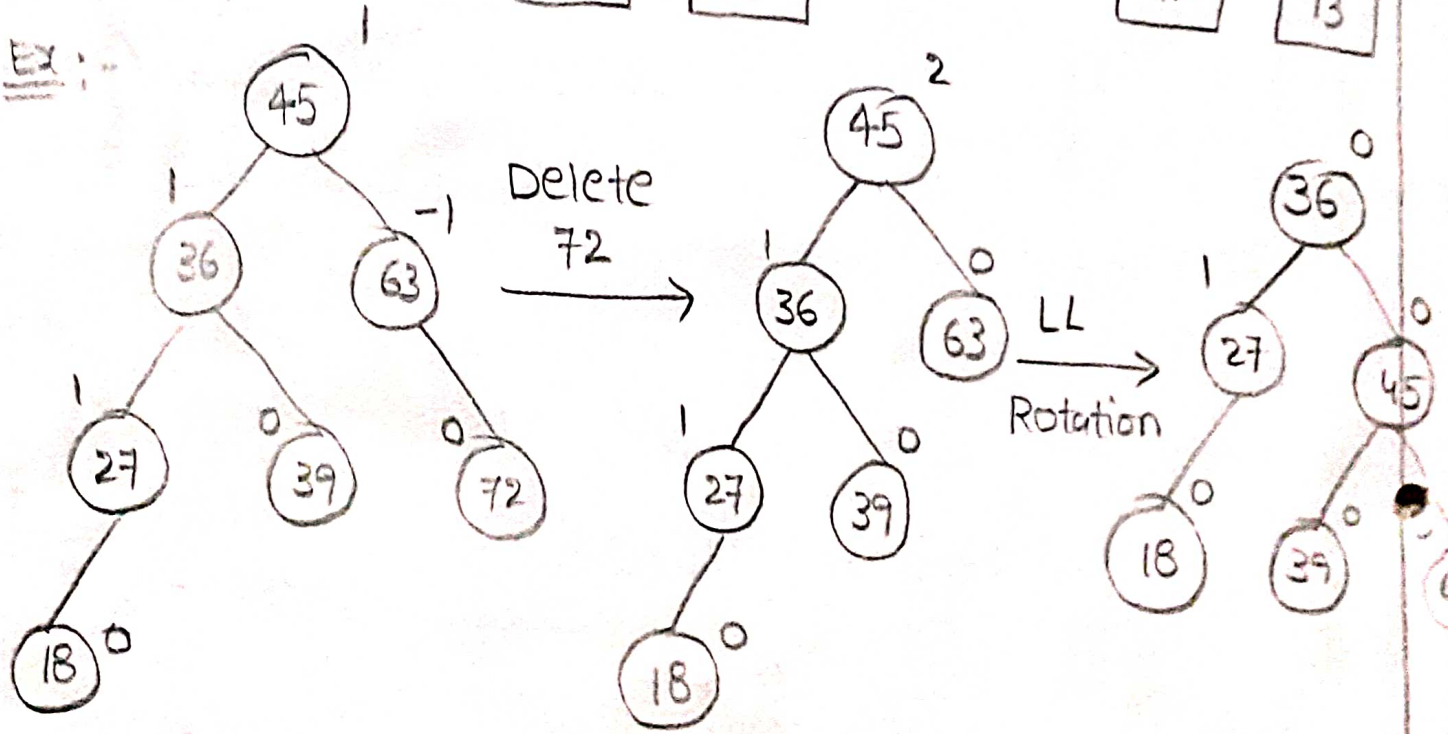
In both R0 & R1 Rotations we follow the same procedure as LL Rotation.

2) RI- Rotation :

let B be the root of left (or) Right sub tree of A (critical node). Then RI - Rotation is applied if the balance factor of B is "1"



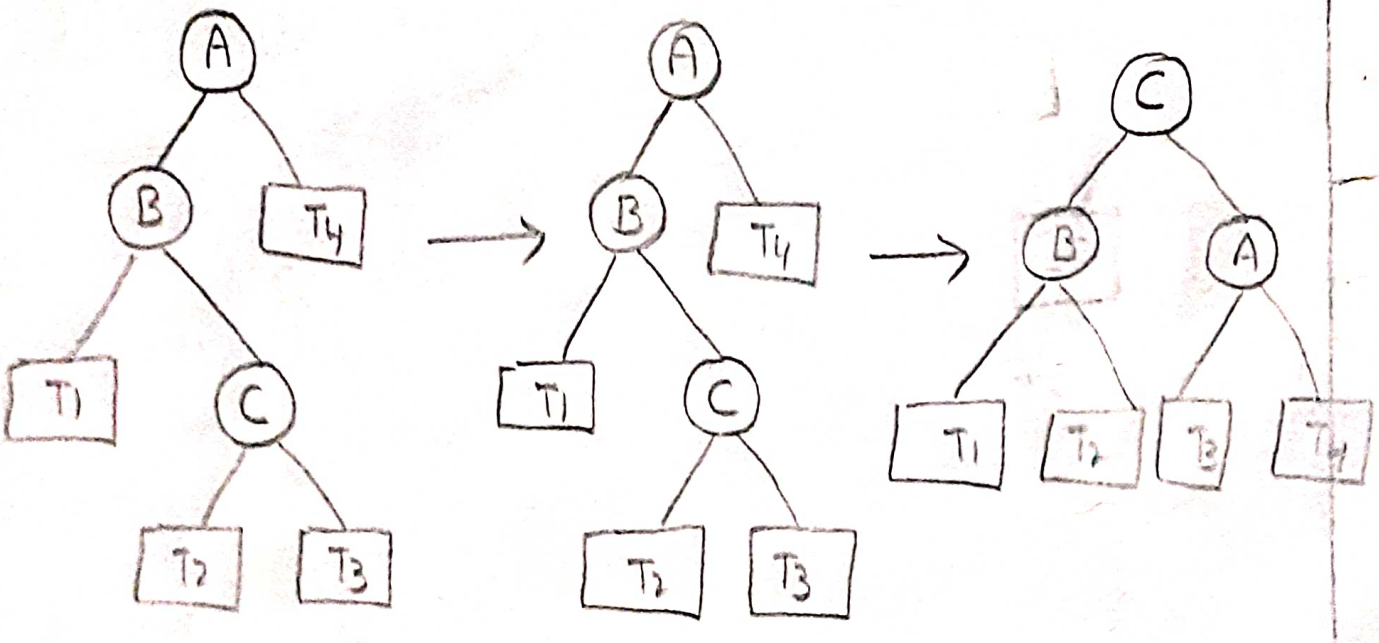
Ex :-



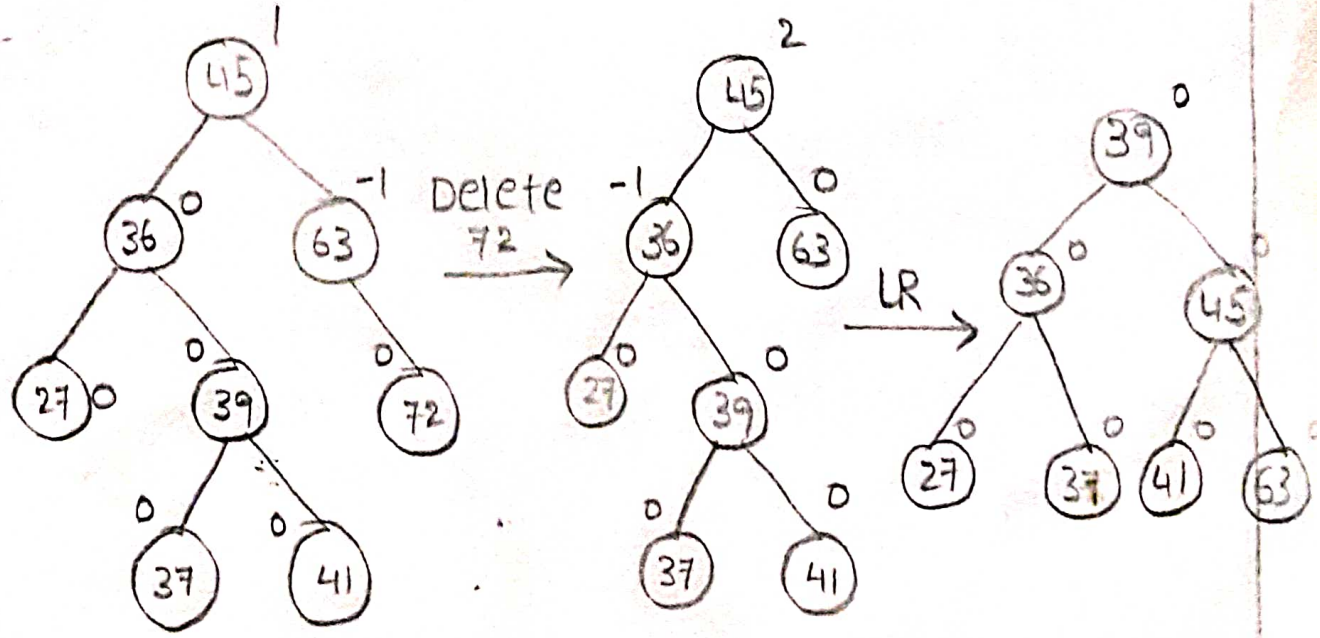
3) R -1 Rotation :-

let B be the root of the left or right sub tree of A (critical node). Then R-1 rotation is applied if the balance factor of B is -1.

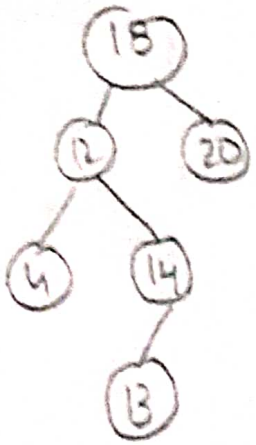
R-1 Rotation is similar to LR - Rotation.



EX :-



Method-4: use a 'keyword = value' expression to identify Fields.



LR

RL