AVL Trees (Balanced Trees)

Do optimize the search time for insertion and deletion.

It is a kind of BST whose heights of left and right subtrees will be with maximum difference of 1.

Each node has balance factor defined as:

Difference b/w the height of left subtree and right subtree of a node

Right heavy (Balance factor -1)

If height of its right subtree is one more than the height of its left subtree

Left heavy (Balance factor +1)

If height of its left subtree is one more than the height of its right subtree.

Balanced node (Balance factor 0)

Same height of left and right subtrees.

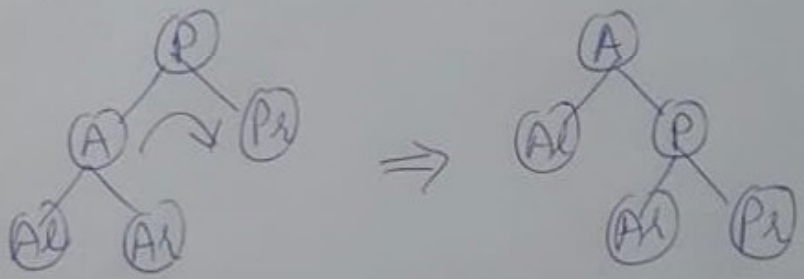
Examples:

Pivot node is unbalanced node (having balance factor other than 0, -1, +1)

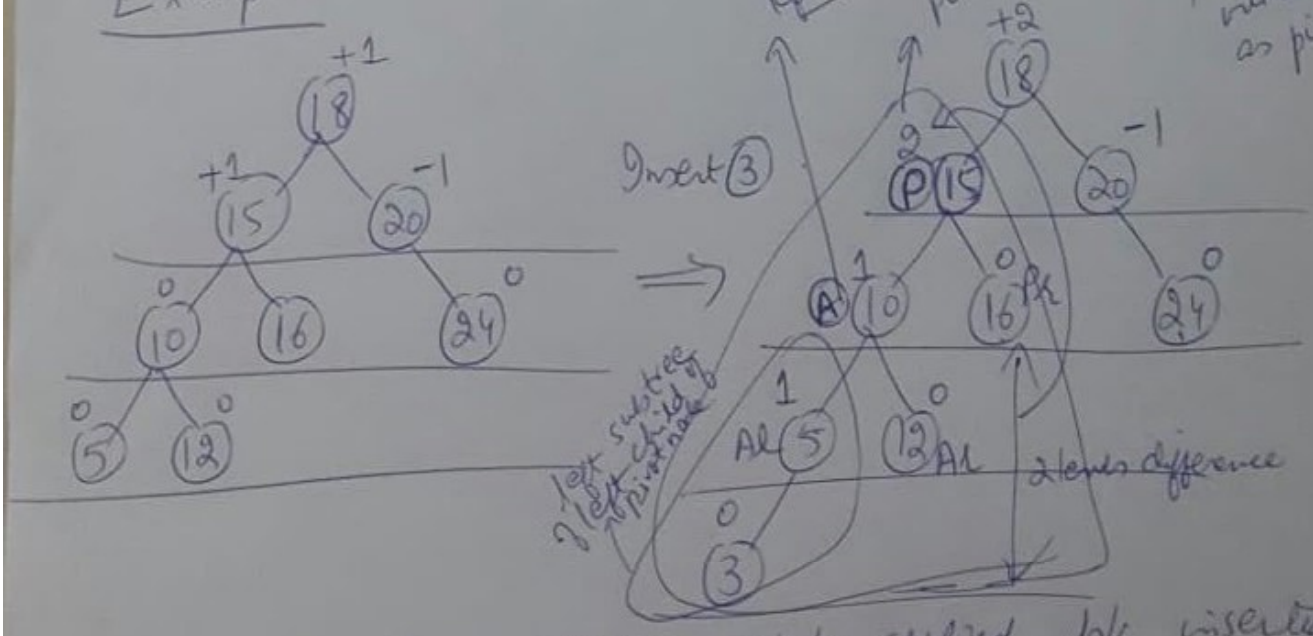
AVL Rotations

1. Left to left rotation
   (Insertion in left subtree of left child of pivot node)
2. Right to right rotation
   (Insertion in right subtree of right child of pivot node)
3. Right to left rotation
   (Insertion in left subtree of right child of pivot node)
4. Left to right rotation
   (Insertion in right subtree of left child of pivot node)

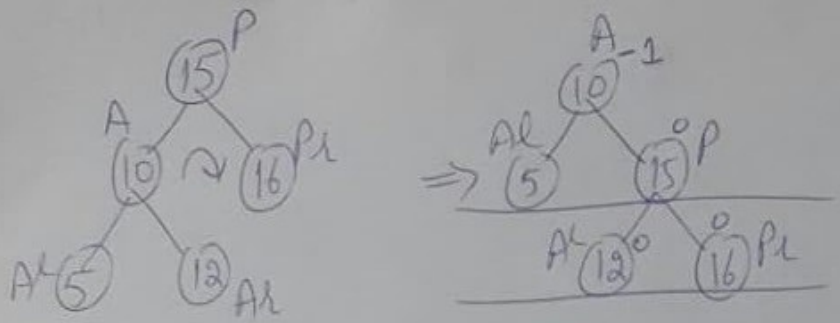
Left to left rotation



Example:

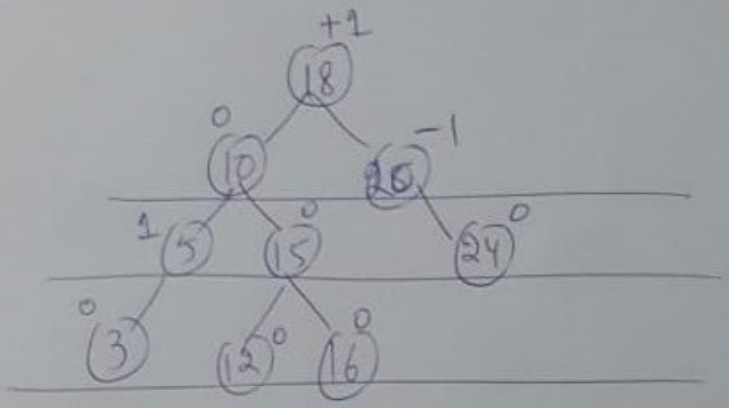


Left to left rotation will be applied b/c insertion is in left subtree of left child of pivot node.

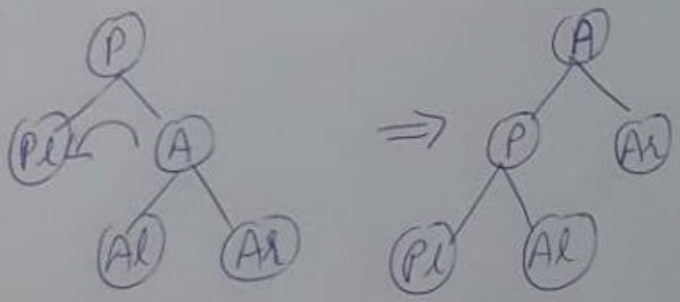


Balanced now.

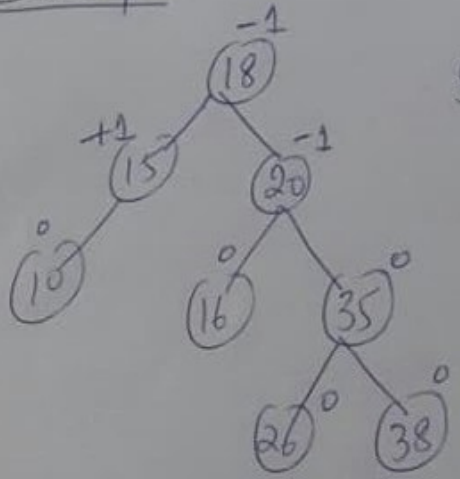
So, the whole tree is now:



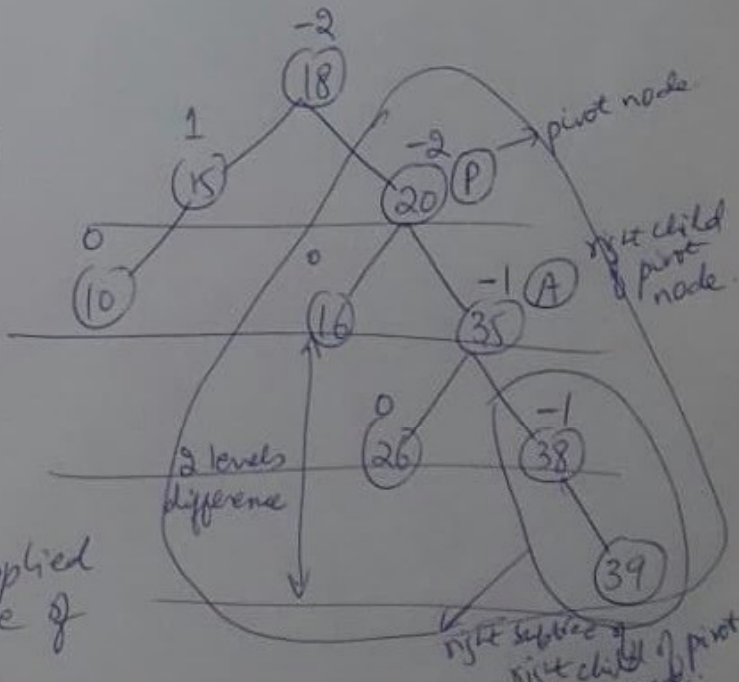
Right to right rotation



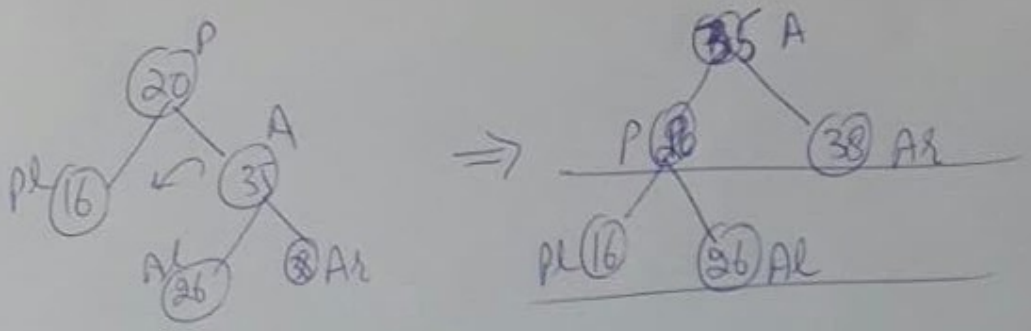
Example:



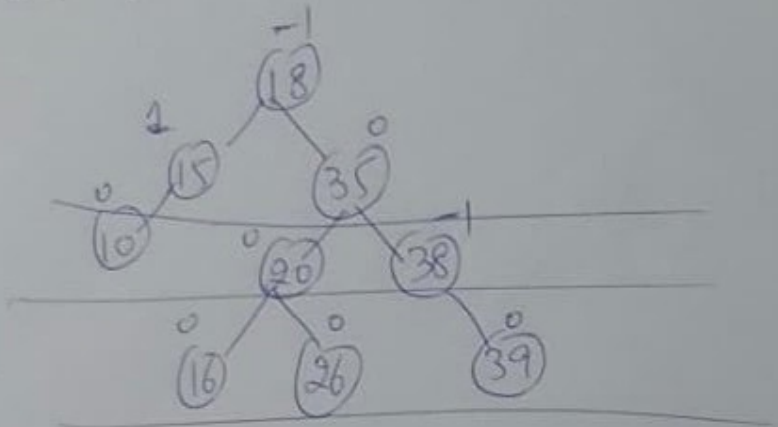
Insert 39



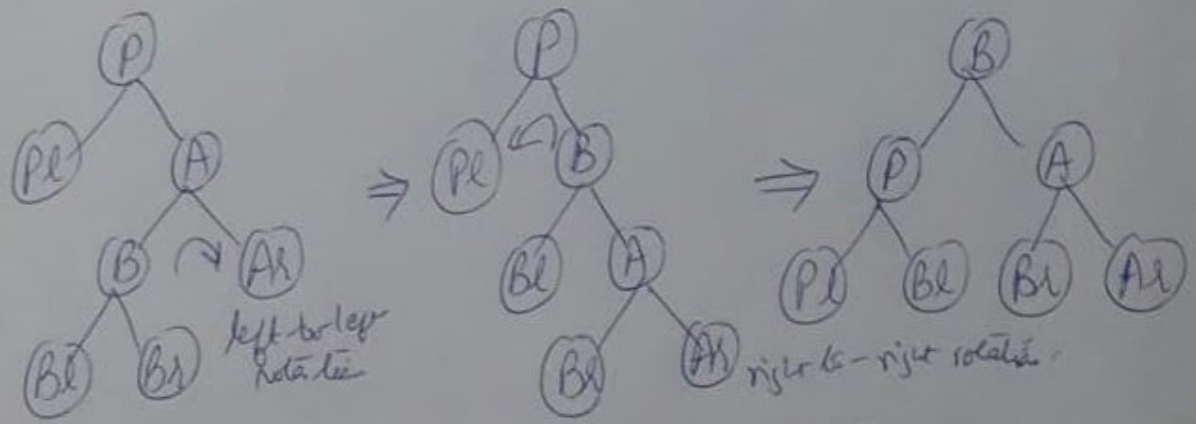
Right-to-right rotation will be applied b/c insertion in the right subtree of right child of pivot node.



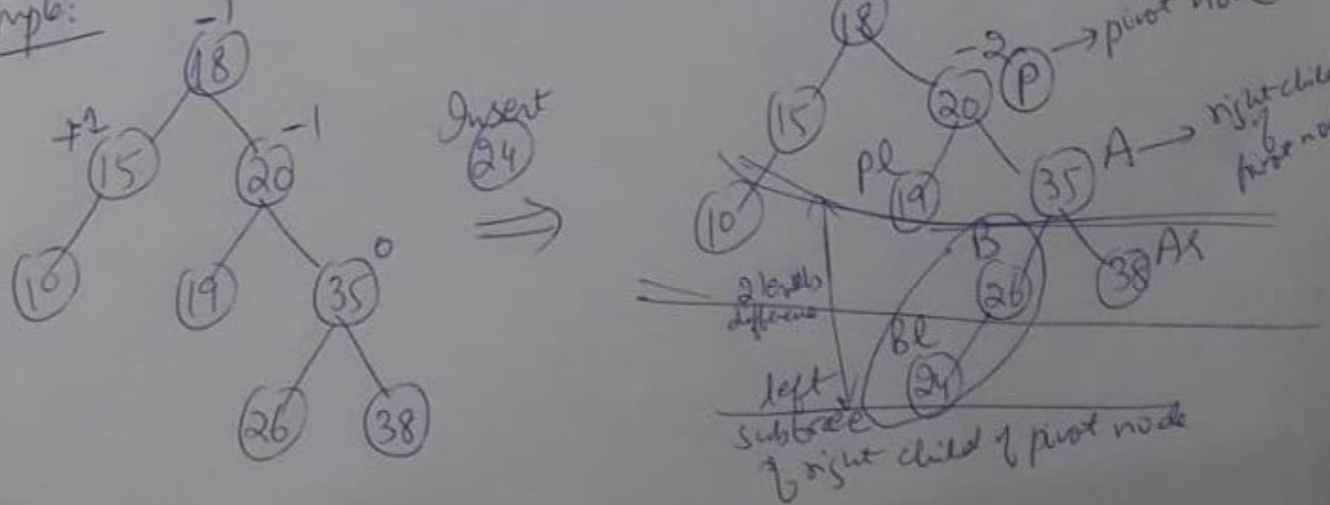
So, the whole tree is now:



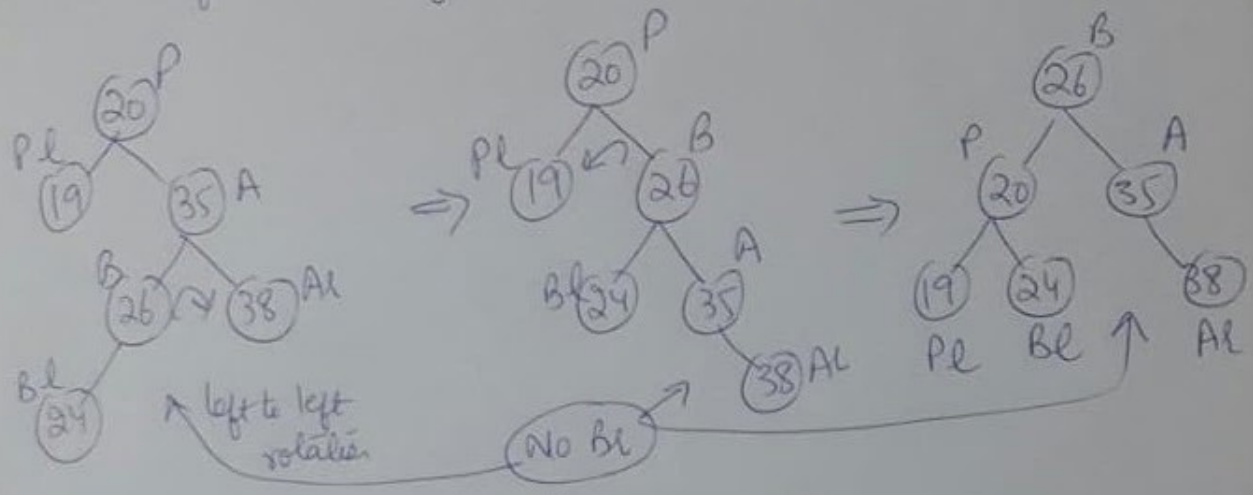
Right to left rotation ① First left to left rotation
② Then right to right rotation.



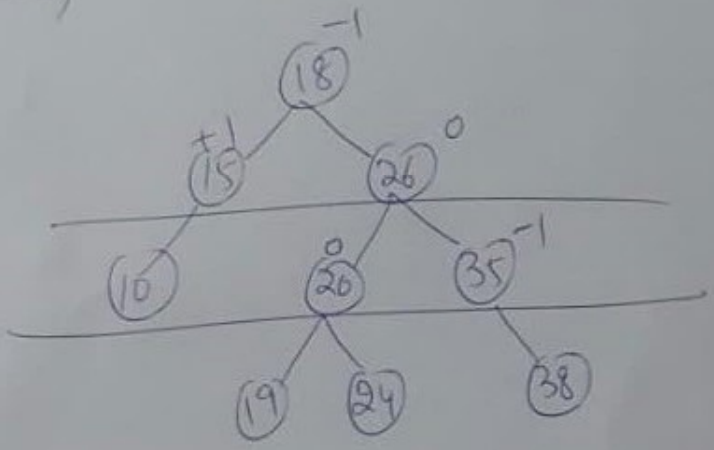
Example:



Right to left rotation will be applied b/c insertion is in left subtree of right child of first node.

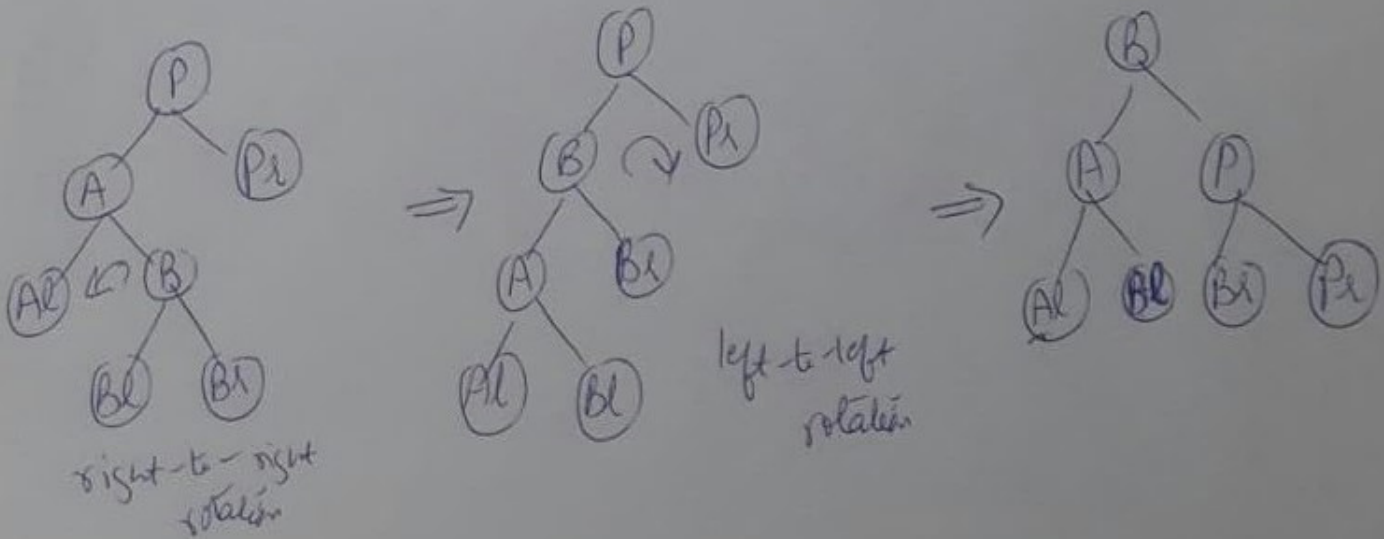


So, the whole tree is now:

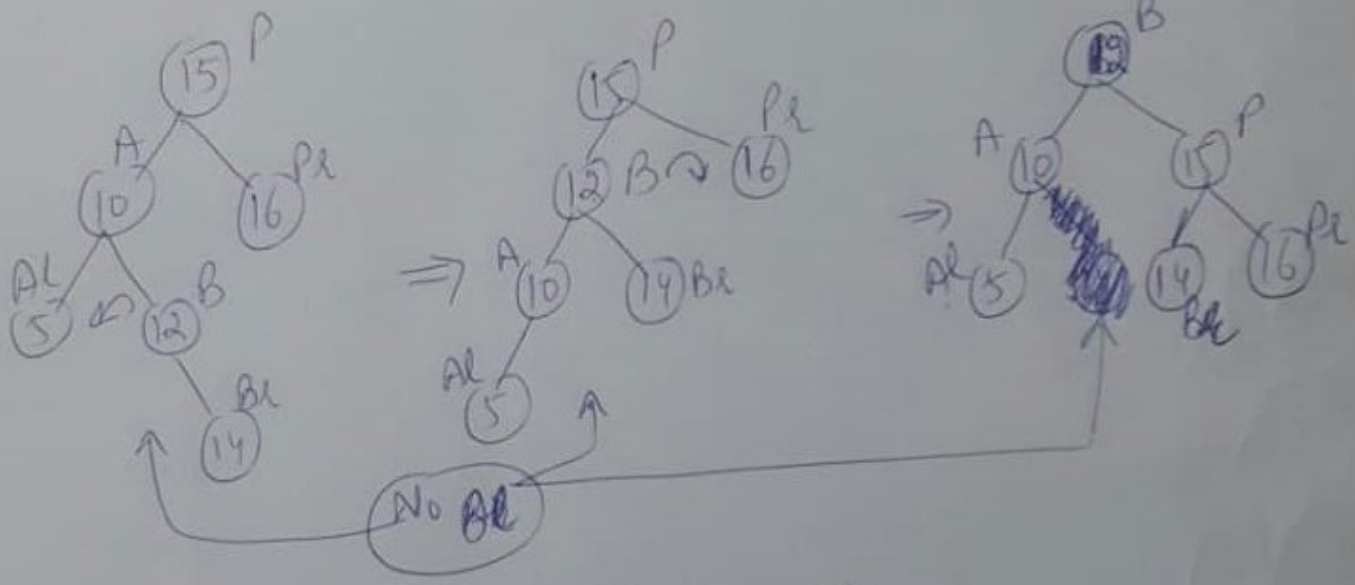


Left-to-Right rotation

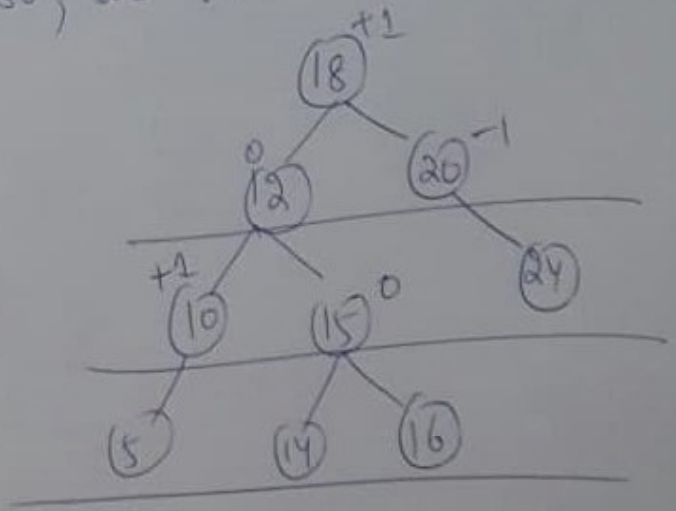
① First right-to-right rotation
② Then left-to-left rotation



Example.

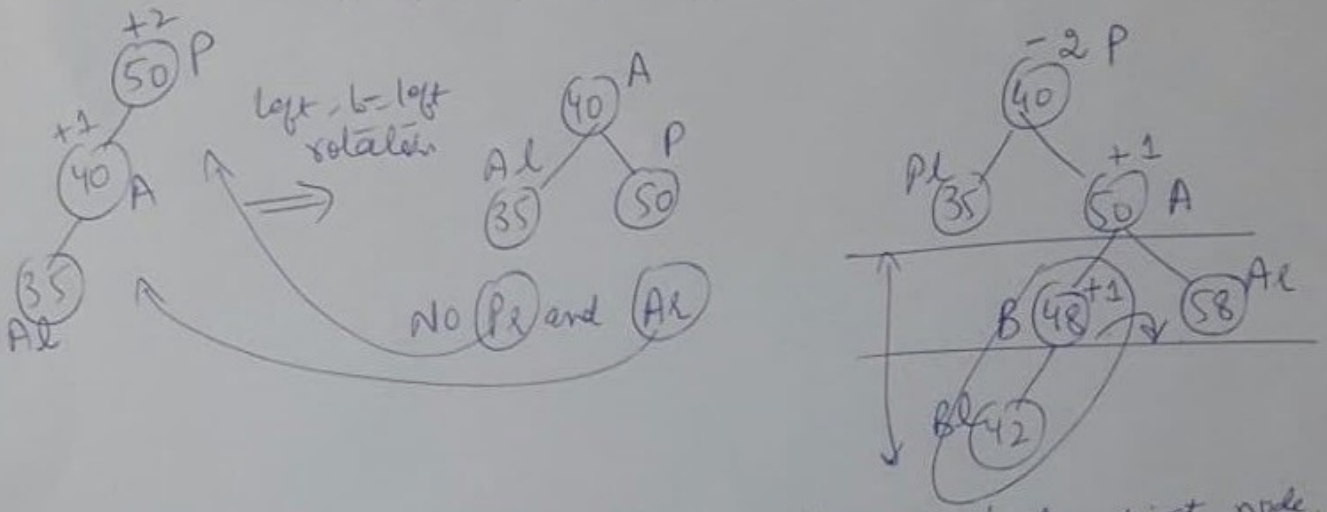


So, the whole tree is now:

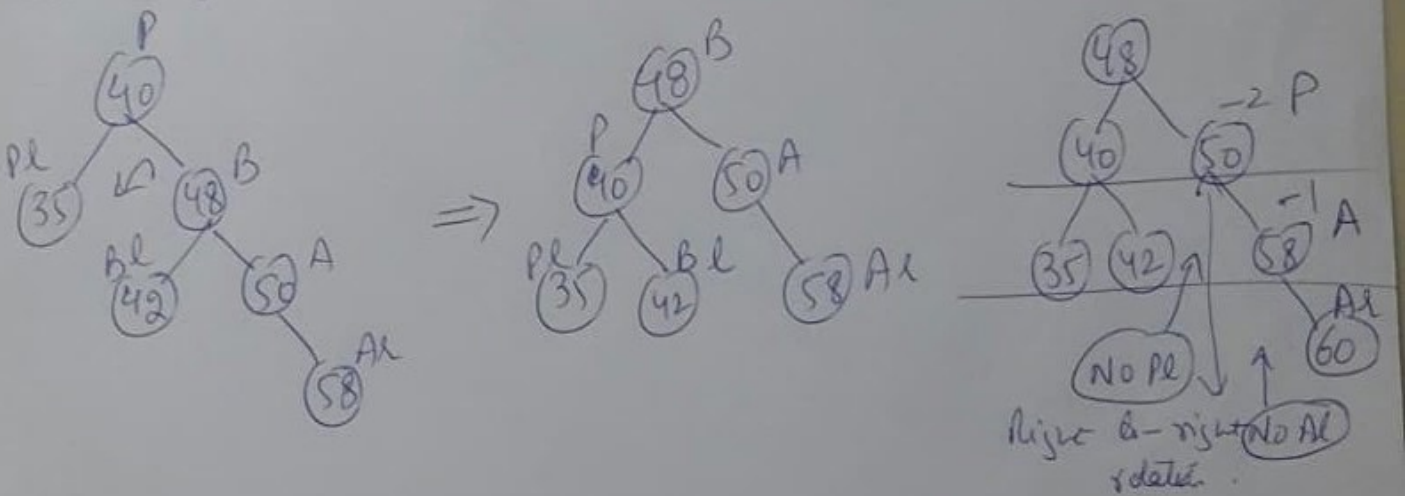


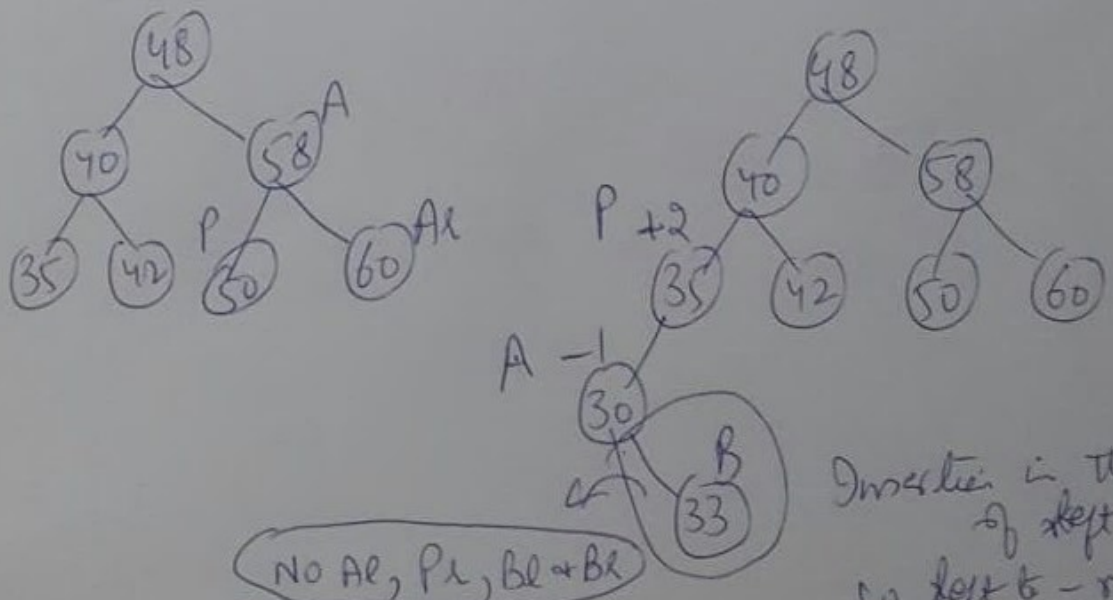
Example. Let us take some numbers and construct an AVL tree from them.

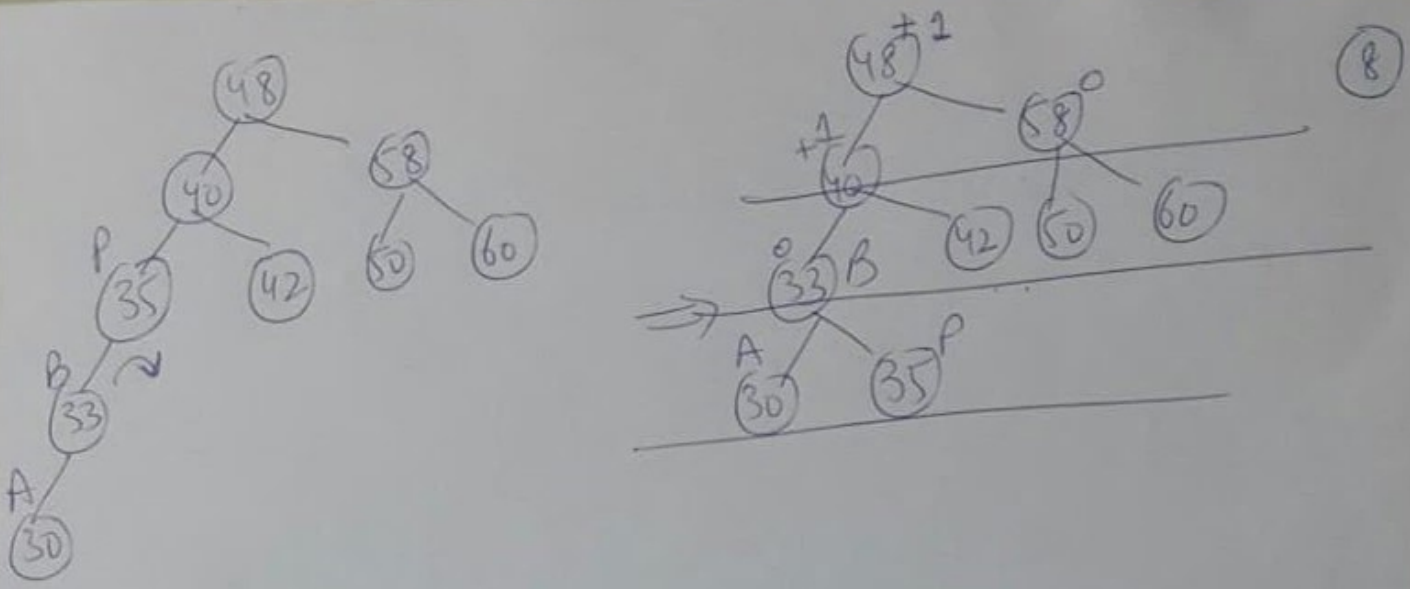
50, 40, 35, 58, 48, 42, 60, 30, 33, 25 (n=10)

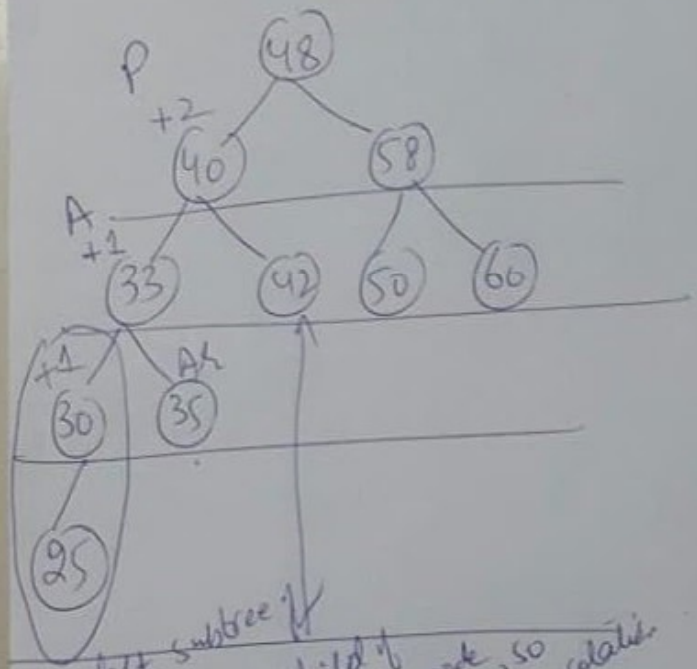


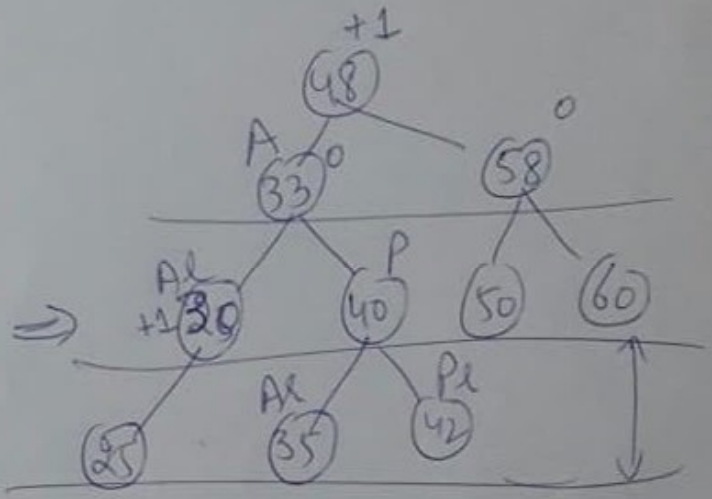
42 is inserted in left subtree of right child of pivot node, so Right - Left rotation will be applied.











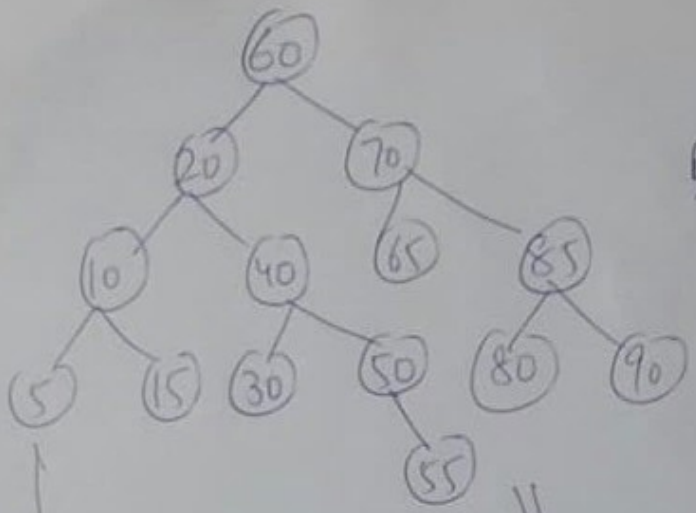
Deletion in AVL Trees

Case (1) → terminal node (same as BST)

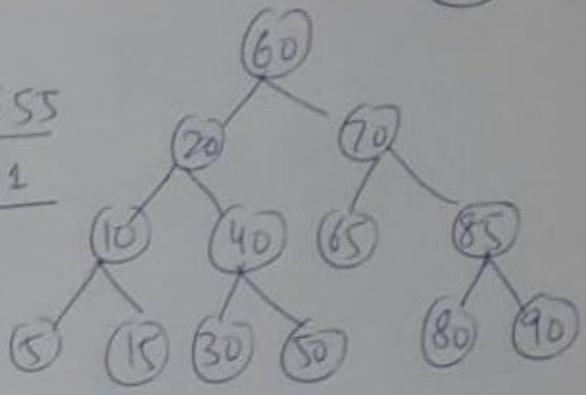
Case (2) → one child node (same as BST)

Case (3) → two children node (replace with inorder predecessor).

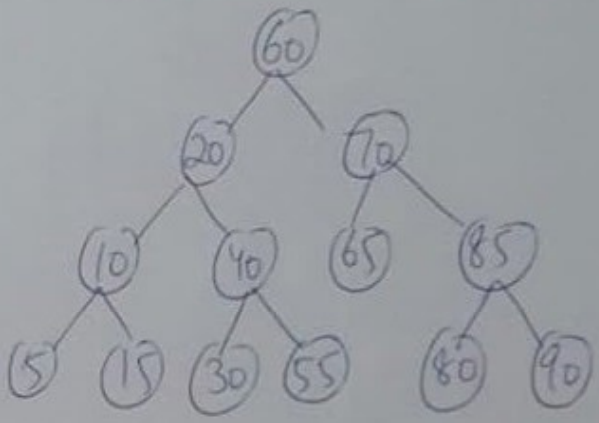
Rebalancing.



Delete 55
case 1



Delete 50
case 2.



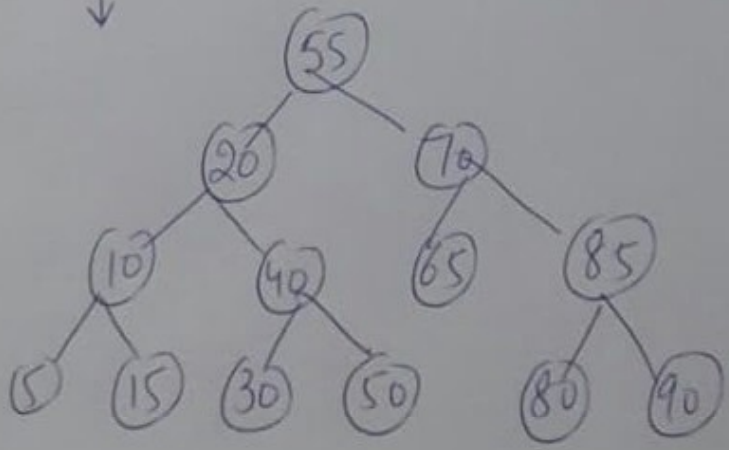
Delete 60
case 3.

(Replace it with its inorder predecessor).

Inorder traversal

5, 10, 15, 20, 30, 40, 50, 55, 60, 65, 70, 80, 85, 90

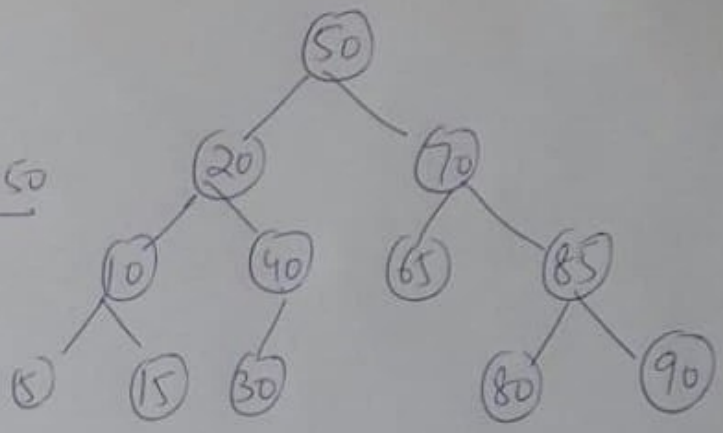
55 ← predecessor.



Delete 55

Case 3

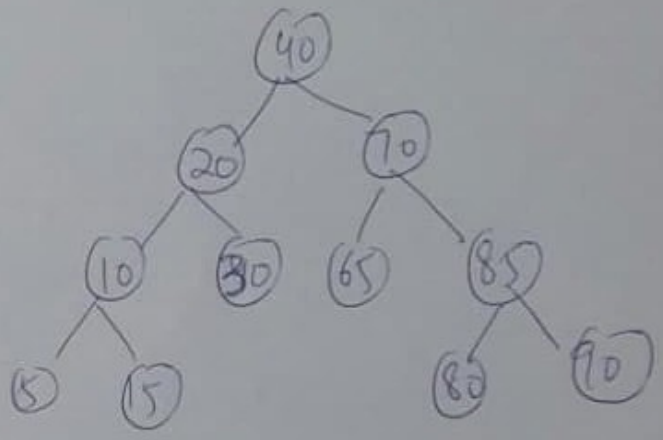
predecessor is 50



Delete 50

Case 3

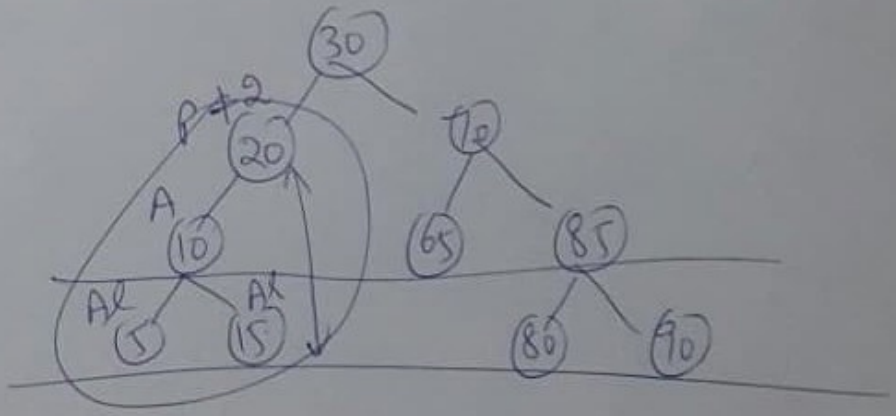
predecessor is 40



Delete 40

Case 3

predecessor is 30



Left-to-right rotation