AVL Trees

→ These are height balanced binary search trees.

How the height of a tree is balanced?

it is balanced using balance factor.

Balance factor is height of leftsubtree − height of right subtree

balance factor = height of leftsubtree − height of right subtree

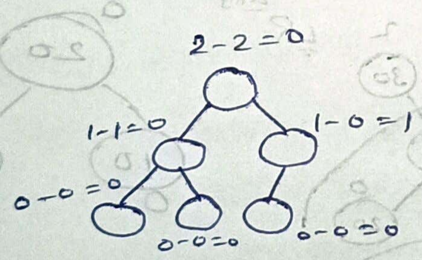
$bf = hl - hr = \{-1, 0, 1\}$ These are valid bf

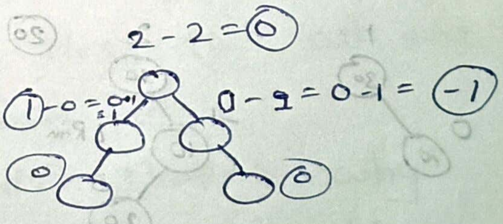
So This balance factor we calculate on every node of a BST, (or) now we will call it as AVL tree.

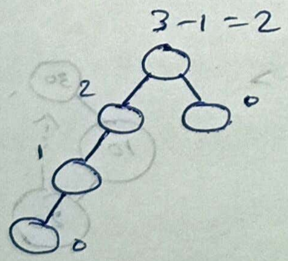
if $|bf| = |hl - hr| \leq 1$ then node balanced

if $|bf| = |hl - hr| > 1$ then node in "

→ if any '1' node is imbalance Then Tree is imbalance.





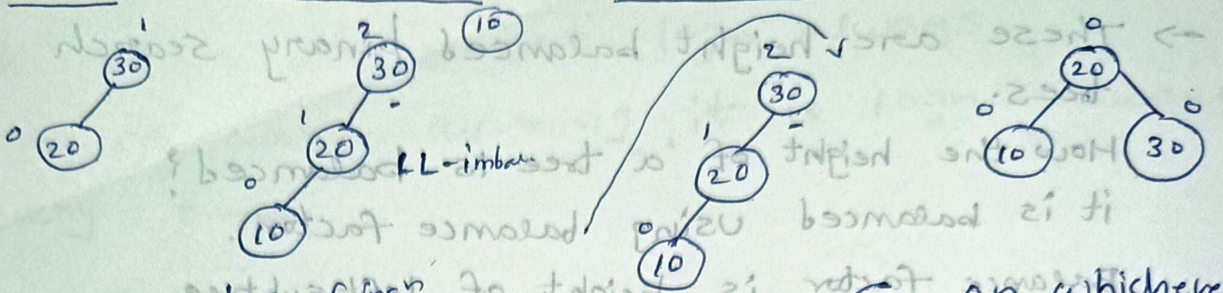


(2 Nodes imbalance).

inserting in AVL with Rotations

Some nodes may become imbalance. so tree become imbalanced. So to balance That, we perform Rotations.

initial After inserting Perform LL (or) Right (or) clockwise Rotn After Rotn



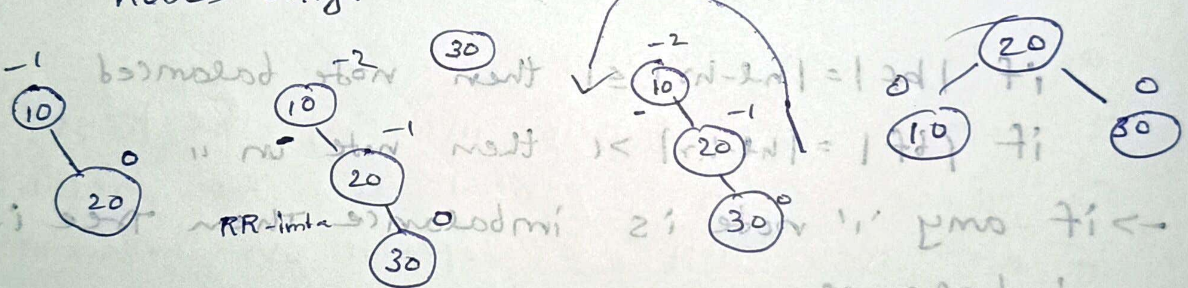
bcz of insn of 10 imbal in which dirn that 10 is inserted.

So, on whichever node u perform rotn The balance factor should become '0' for That one.

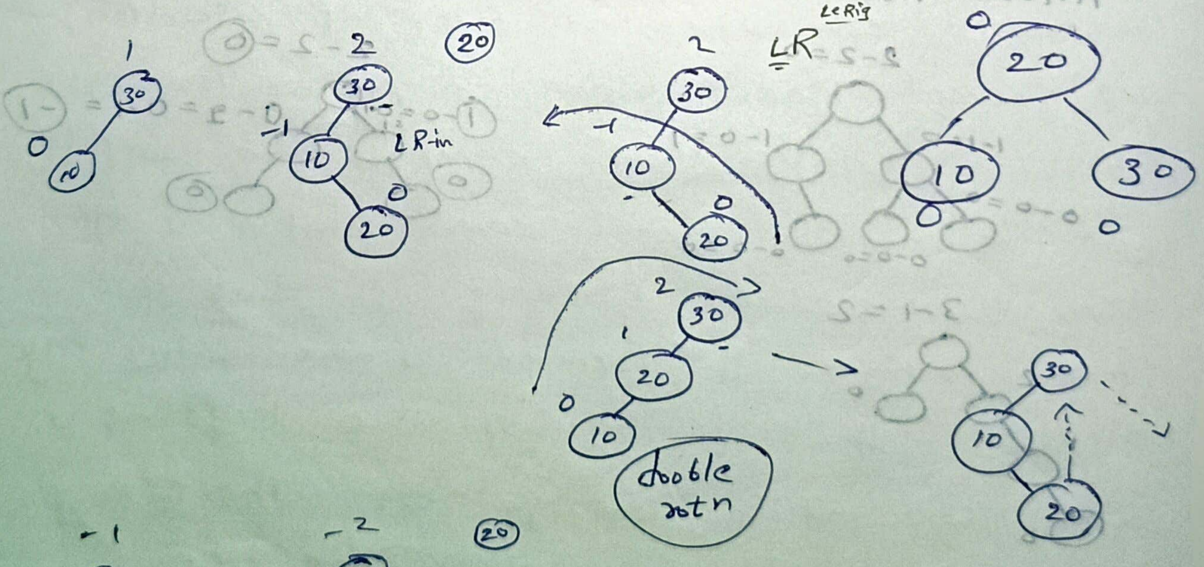
→ what i should do if the tree is very big.

⇒ Rotans are always done with the three nodes only.

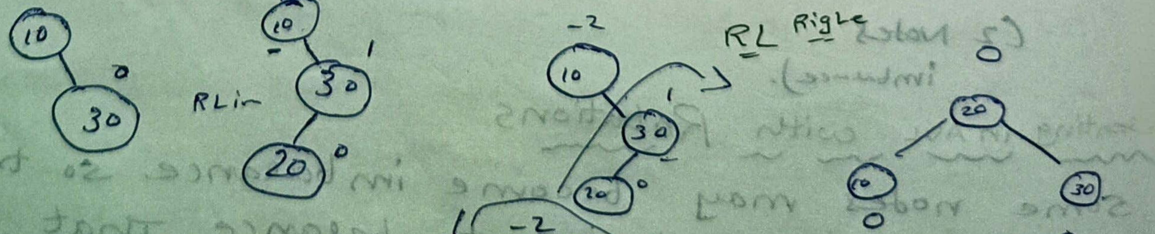
RR (or) left (or) anticlockwise.

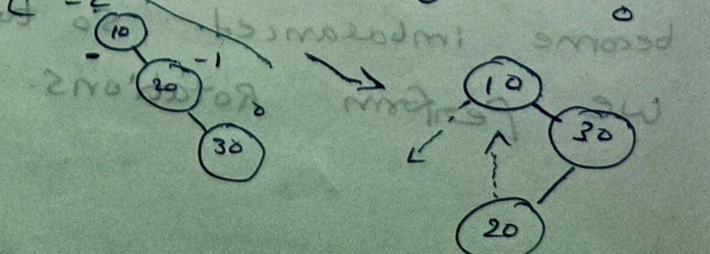


LR LeRig



RL Rigle





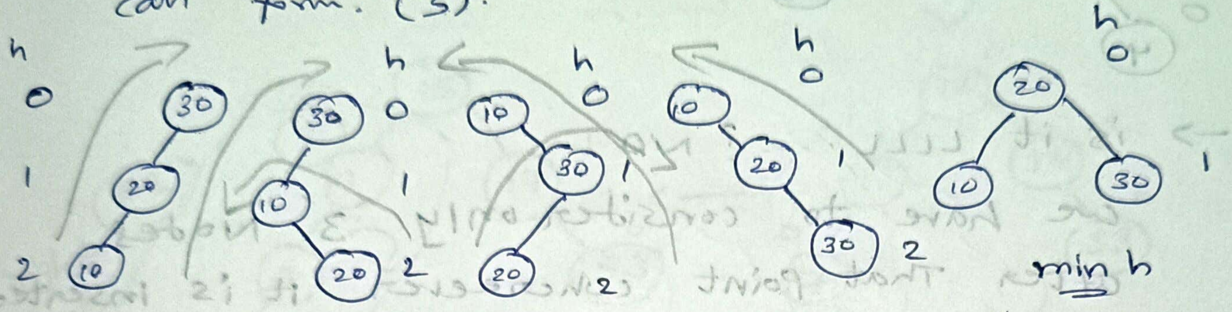
LL, RR → Single Rn

RL, LR → double Rotns

-> out of '4' Rotⁿ's -> one observⁿ That may be logically useful

n = 3

using '3' Keys how many diff BST's (keys we can form. (5).



So, four trees of larger height, & This is of Smaller height. AVL trees means 'h' balance BST. so, which is more h balance? last one.

So when u have '3' Keys, u can draw a BST of any shape, all are BST's only.

-> so, if u have first one, why don't u take last one? change first to last one

* - AVL Tree idea originates from here when u can have '5' diff shapes, why don't u select that last shape?, even if u got first one, modify it.

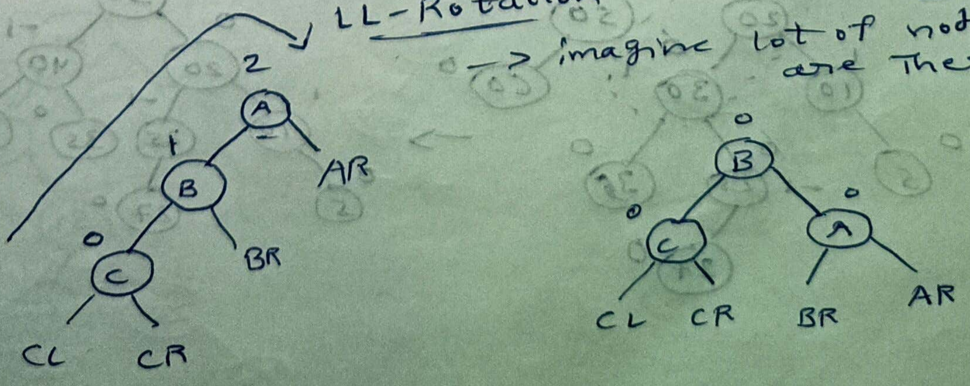
General form of AVL Rotations

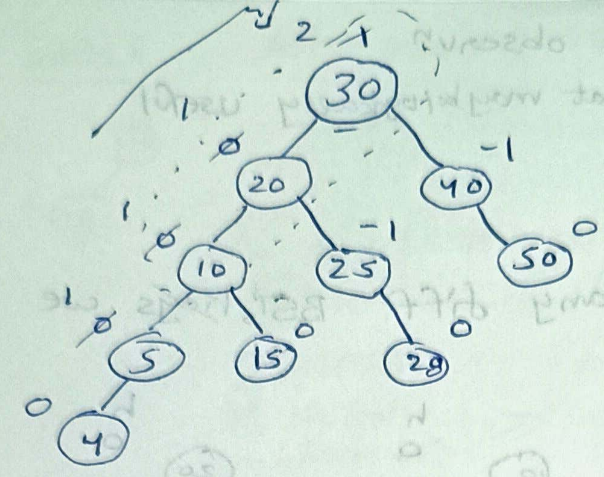
Formula of Rotⁿs for insⁿ

+ve left
-ve right

LL-Rotation

-> imagine lot of nodes are There.



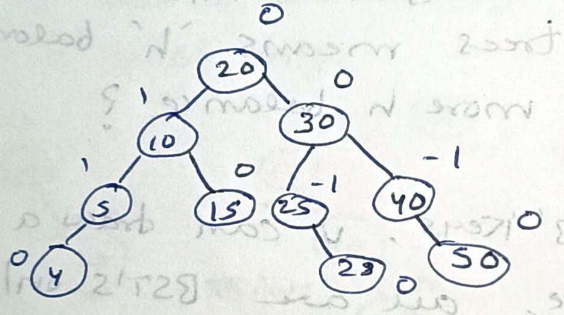


balanced.

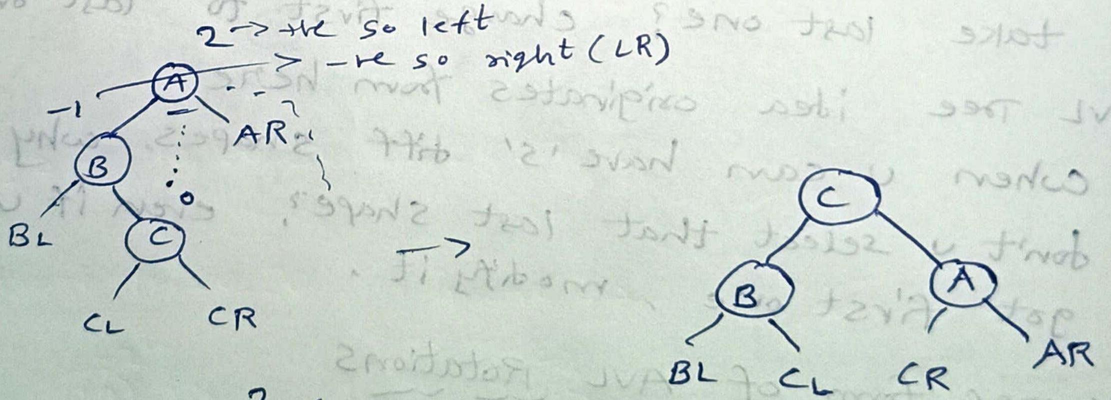
now inserting 4

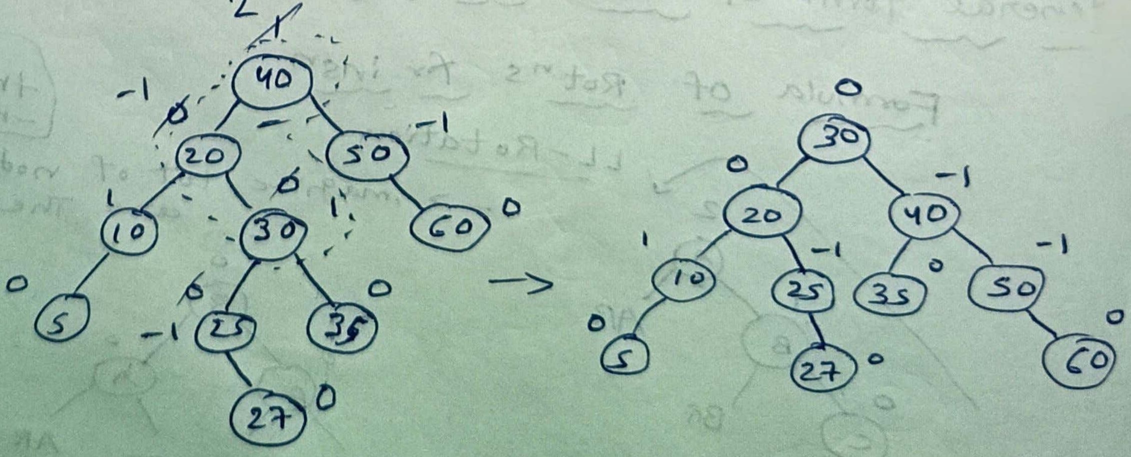
-> is it LLLL....? NO

we have to consider only '3' Nodes after That point wherever it is inserted it is LL only.

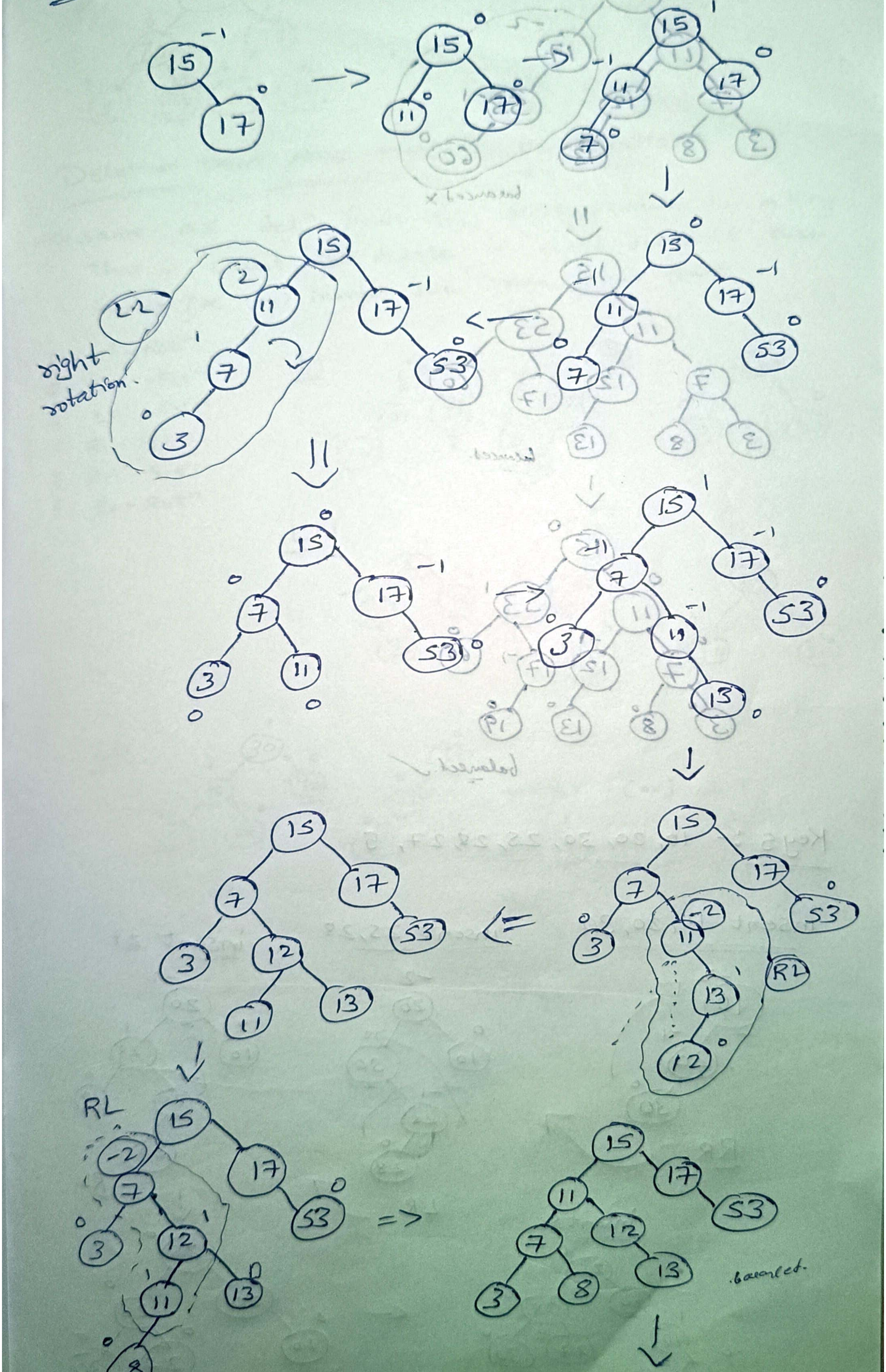


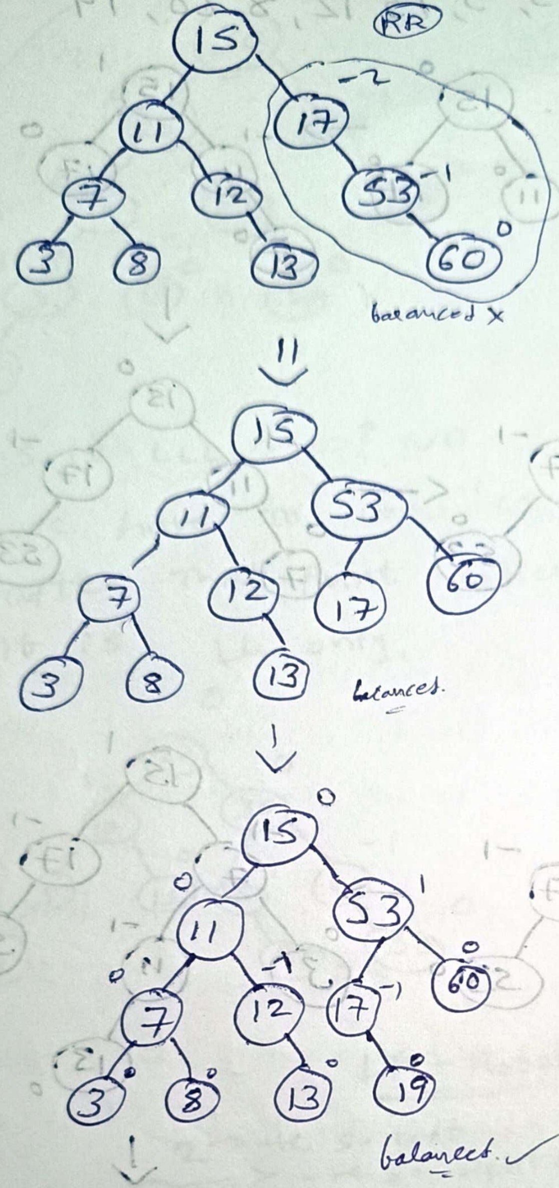
LR - Rotation



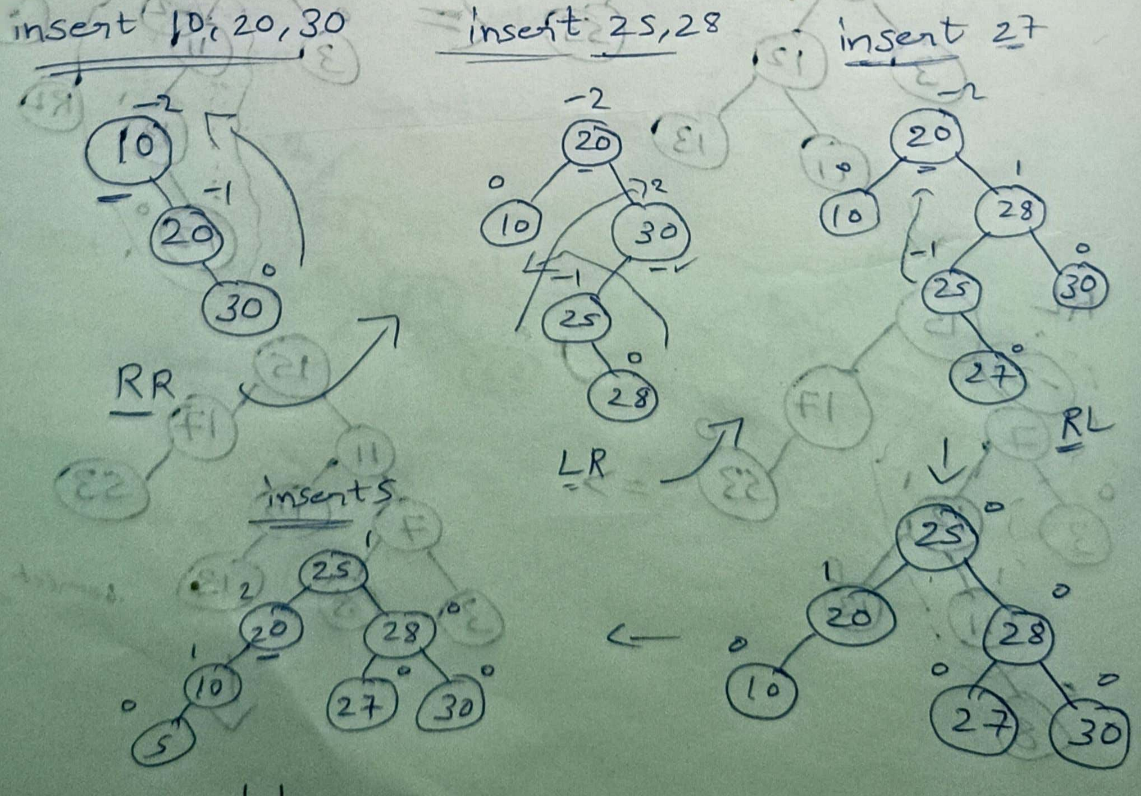


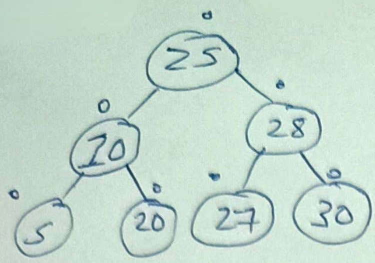
eg:- 15, 17, 11, 7, 53, 3, 13, 12, 8, 60, 19





Keys :- 10, 20, 30, 25, 28, 27, 5



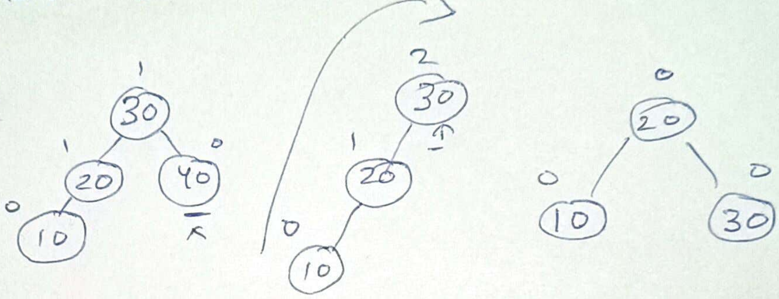


Deletion from AVL Tree with Rotations

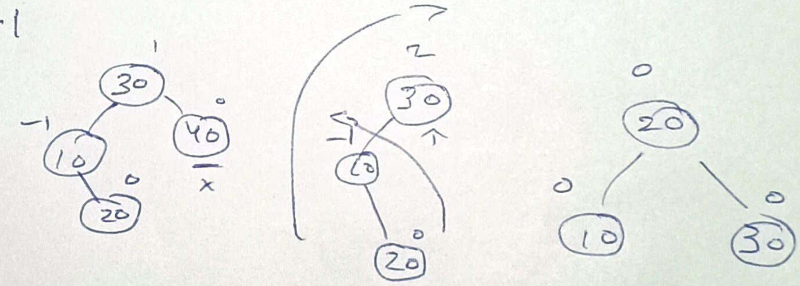
→ same as deln from BST, first search for a key That u want to delete. if found delete it Then inorder pre (or) inorder suc Takes That place.

1. L1 – Rotn
2. L-1 – Rotn
3. L0 – Rotn
4. R1 – Rotn
5. R-1 – Rotn
6. R0 – Rotn

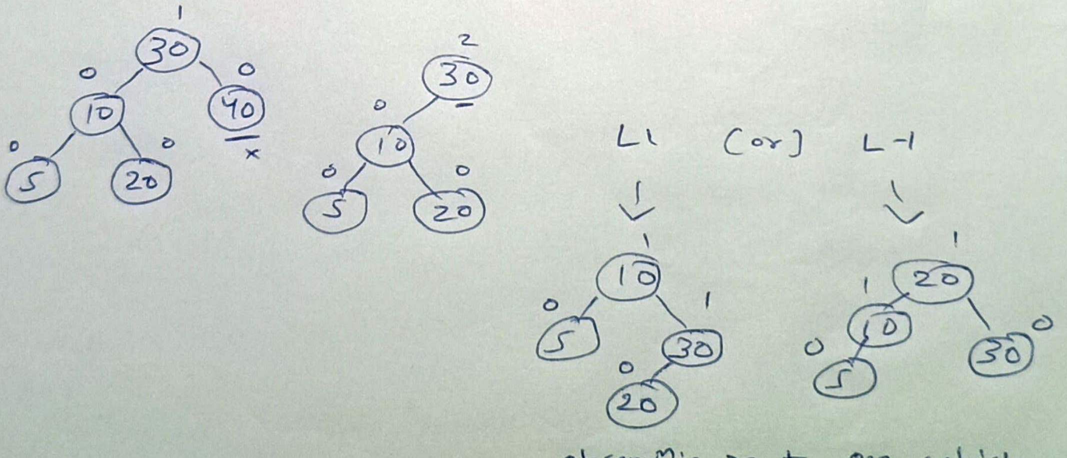
L1



L-1



L0



observn:- root, on which we have performed rotn not becoming zero.