AVL(Adelson-Velsky and Landis)

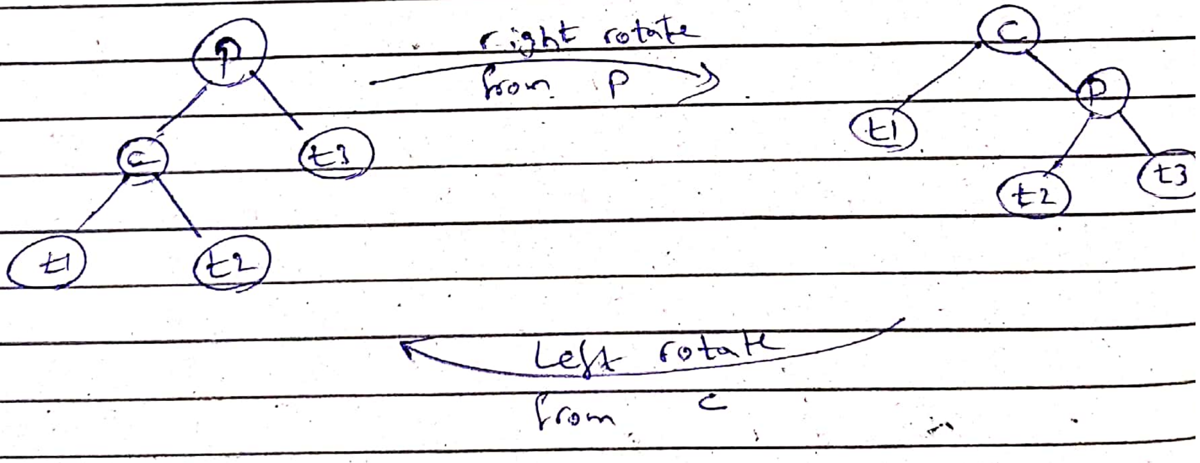
*) Algorithm Used for

→ Self Balancing Binary Tree

*) Two approaches or operations

- → Left Rotate
- → Right Rotate

*) Example of operations



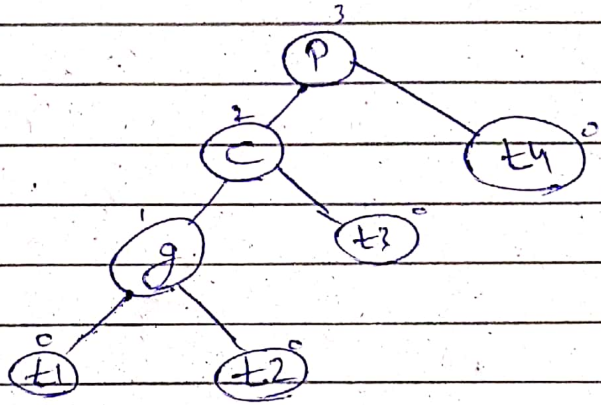
*) 4 rules to perform operations

- → Left-left case
- → Left-Right case
- → Right-left case
- → Right-Right case

1 Left - Left Case

→ Child is at left side of parent.
GrandChild is at Left side of child

→ Representation



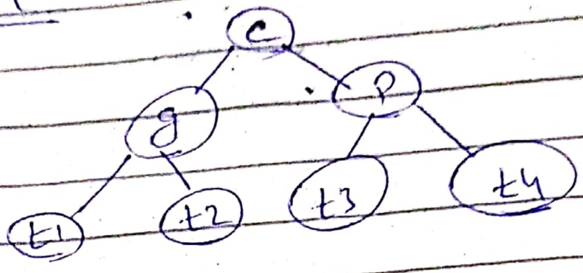
→ Terminologies

p → parent
c → child
g → grand child
t1 → SubTree 1
t2 → SubTree 2
t3 → SubTree 3
t4 → SubTree 4

→ Operations Performed in this case

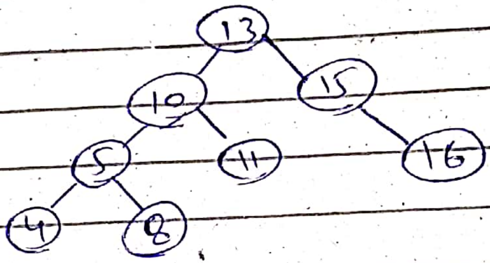
→ Right Rotate (p)

Result

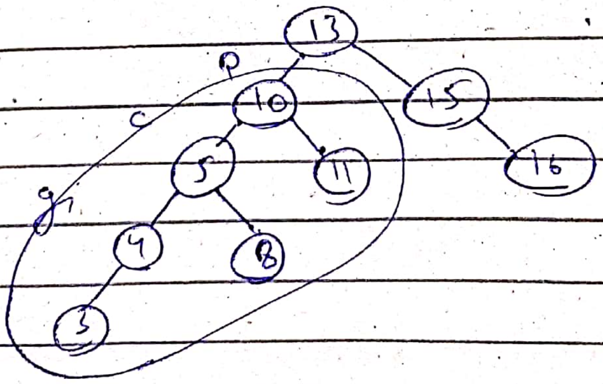


Example:-

Tree Given as

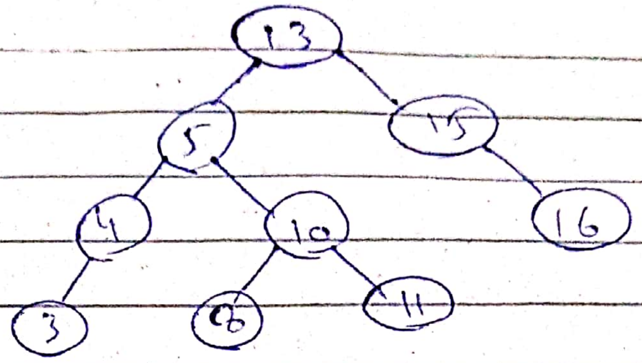


Adding 3 in this



The rounded subtree is unbalanced Grand child will be at the side where changes are made. Since 3 is getting added on left side so grandchild will be at left

Right rotating (p)

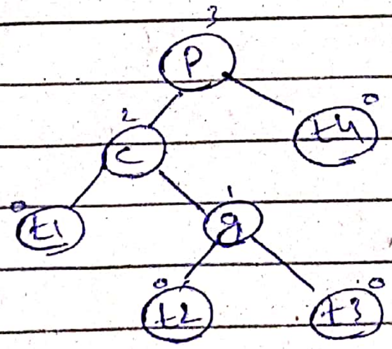


It is now Balanced

② Left Right Case :

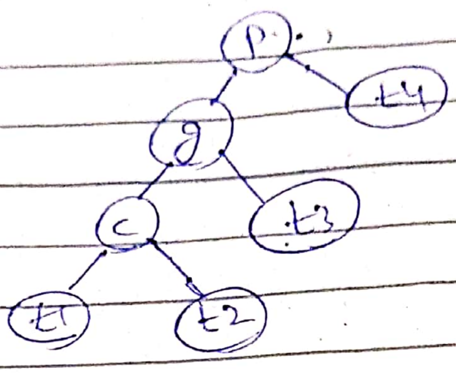
- Child is at left side of parent
  GrandChild is at right side of child

- Representation

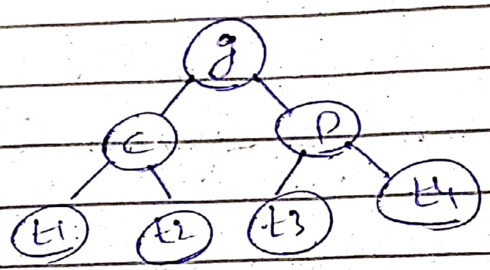


- Operations Performed in this case
  - Left Rotate (c)
    Right Rotate (p)

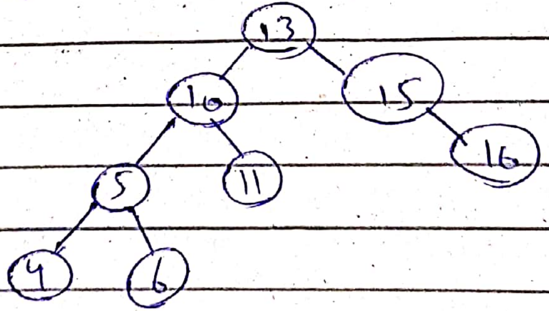
- Result After Left rotating (c)



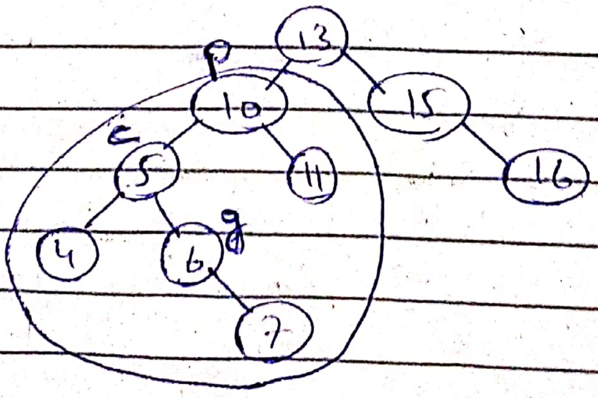
After Right rotating (p)



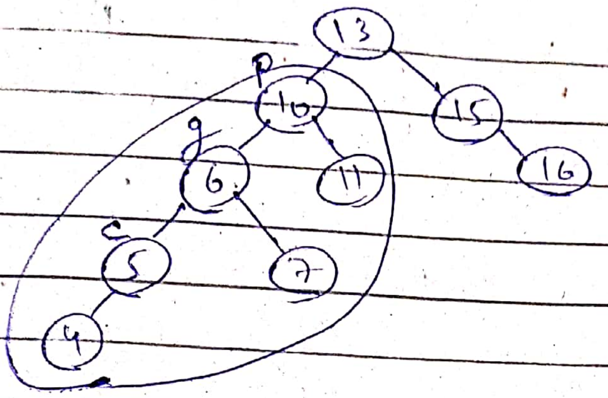
→ Example



Adding (7) into it



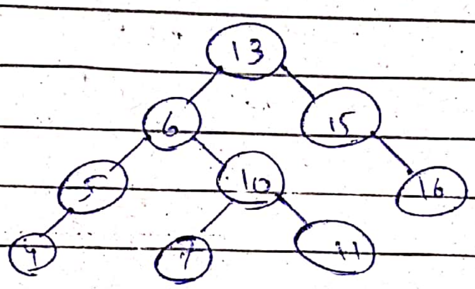
→ Applying operation:



Note

Again following Left-Left case so right rotating (p)

2 – Right rotating (p)

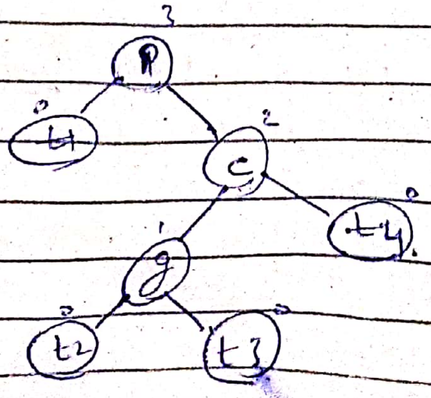


It is Balanced now

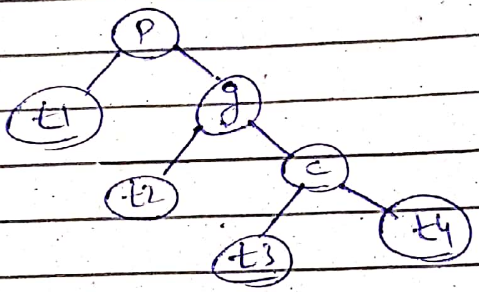
(3) Right Left Case

→ Child is at right side of parent.
grandchild is at Left side of child.

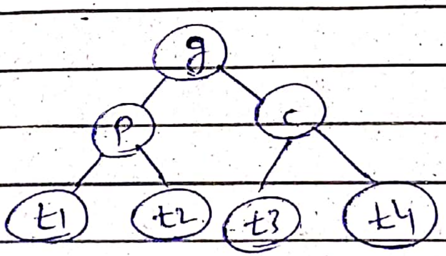
→ Representation



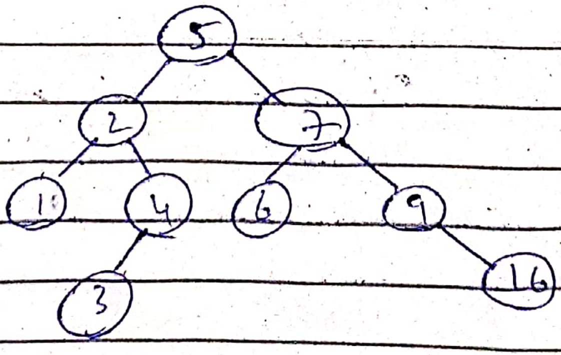
- Operations Performed in this case
  - Right Rotate (c)
  - Left Rotate (p)
- Results
  - After Right Rotating (c)



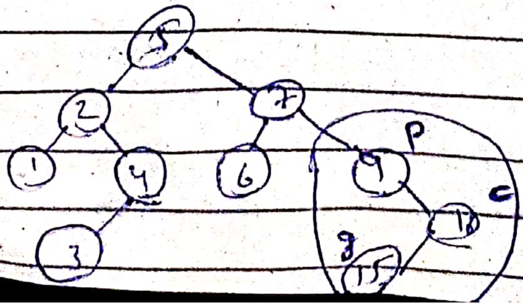
  - After Left Rotating (p)



- Example



Adding 15 into it

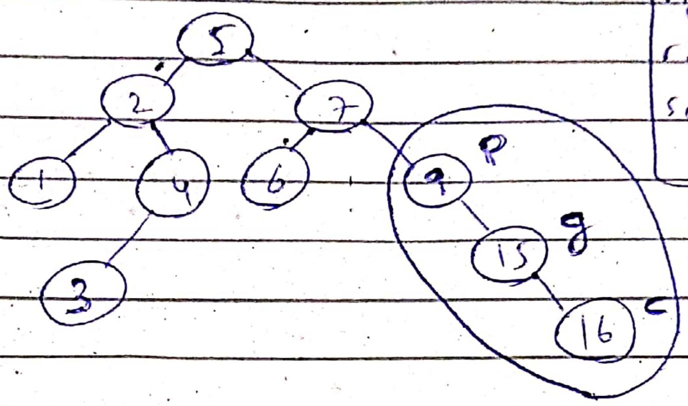


Applying Operations

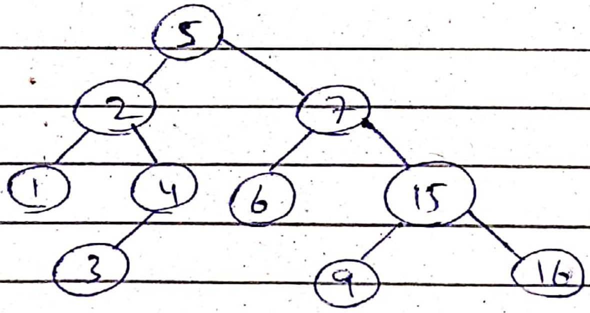
1 — Right Rotating (c)

Note

Again following right-right case so left rotating p



2 — Left Rotating (p)

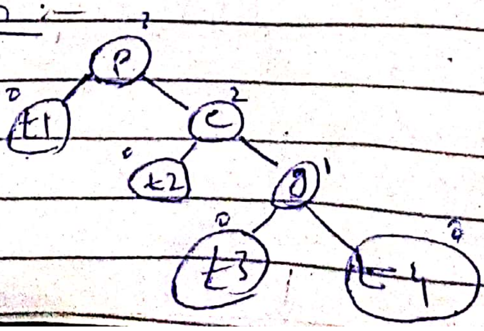


It is now Balanced

④ Right Right Case

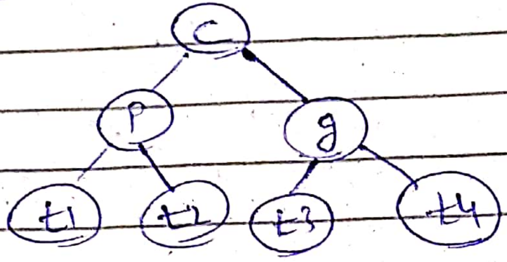
→ Child is at right side of parent
Grand Child is at right side of child

→ Representation :—

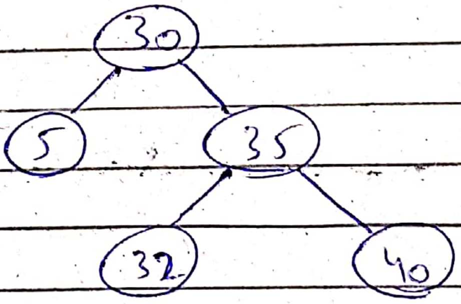


→ Operations Performed in this case
→ Left Rotate (p)

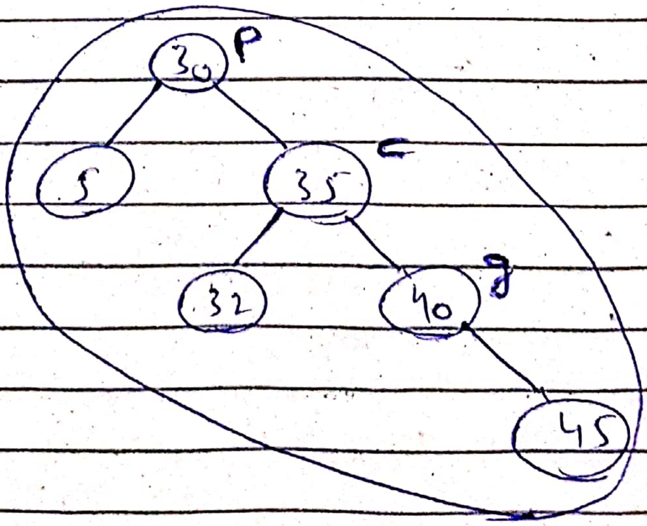
→ Results



→ Example

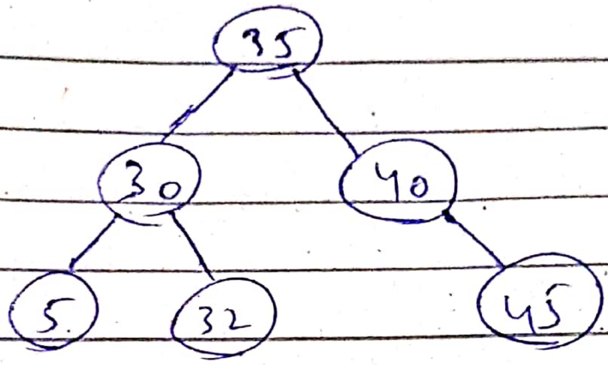


Adding 45 into this



↳ Applying Operation

Left Rotating (P)



It is now Balanced

A) Time Complexity

↳ Adding → Log (N) + O(1)

↓
Rotation

↳ O(Log(N))
Because the tree is balanced every time.