AVL Trees (Not imp for interviews)

↓

Georgy Adelson-Velsky & Landis

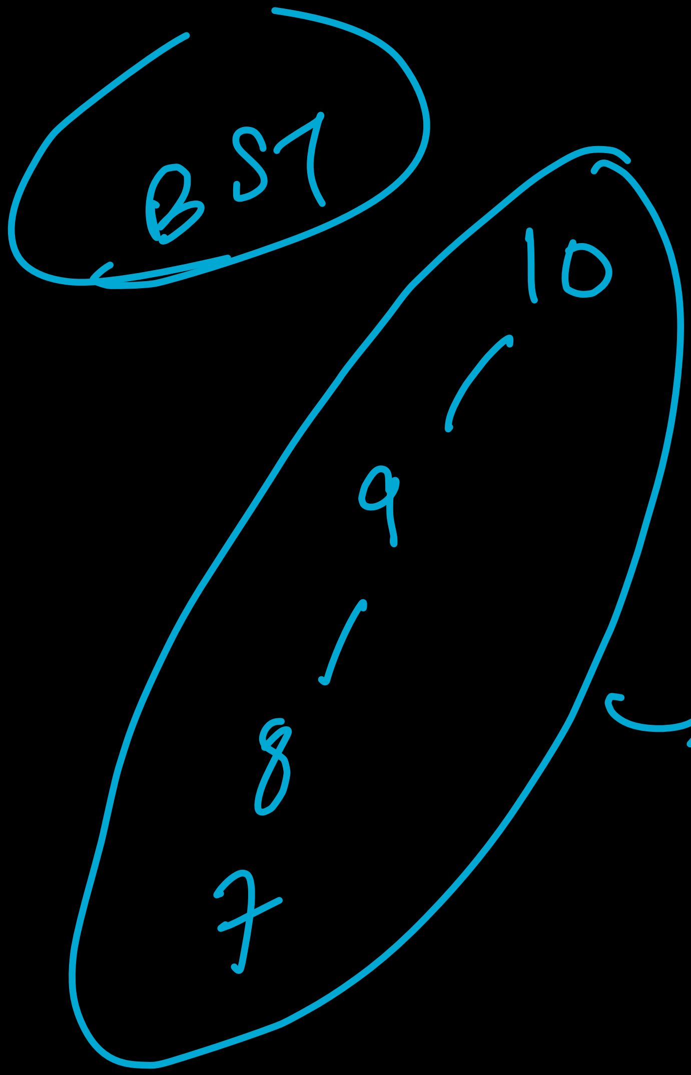
AVL trees are Self Balancing BST

Balanced B-T → for every subtree of the given binary tree the balancing factor should be either 0, 1, -1.

height → O(log n) always

Balancing Factor → $|h_{leftsubtree} - h_{rightsubtree}| \leq 1$

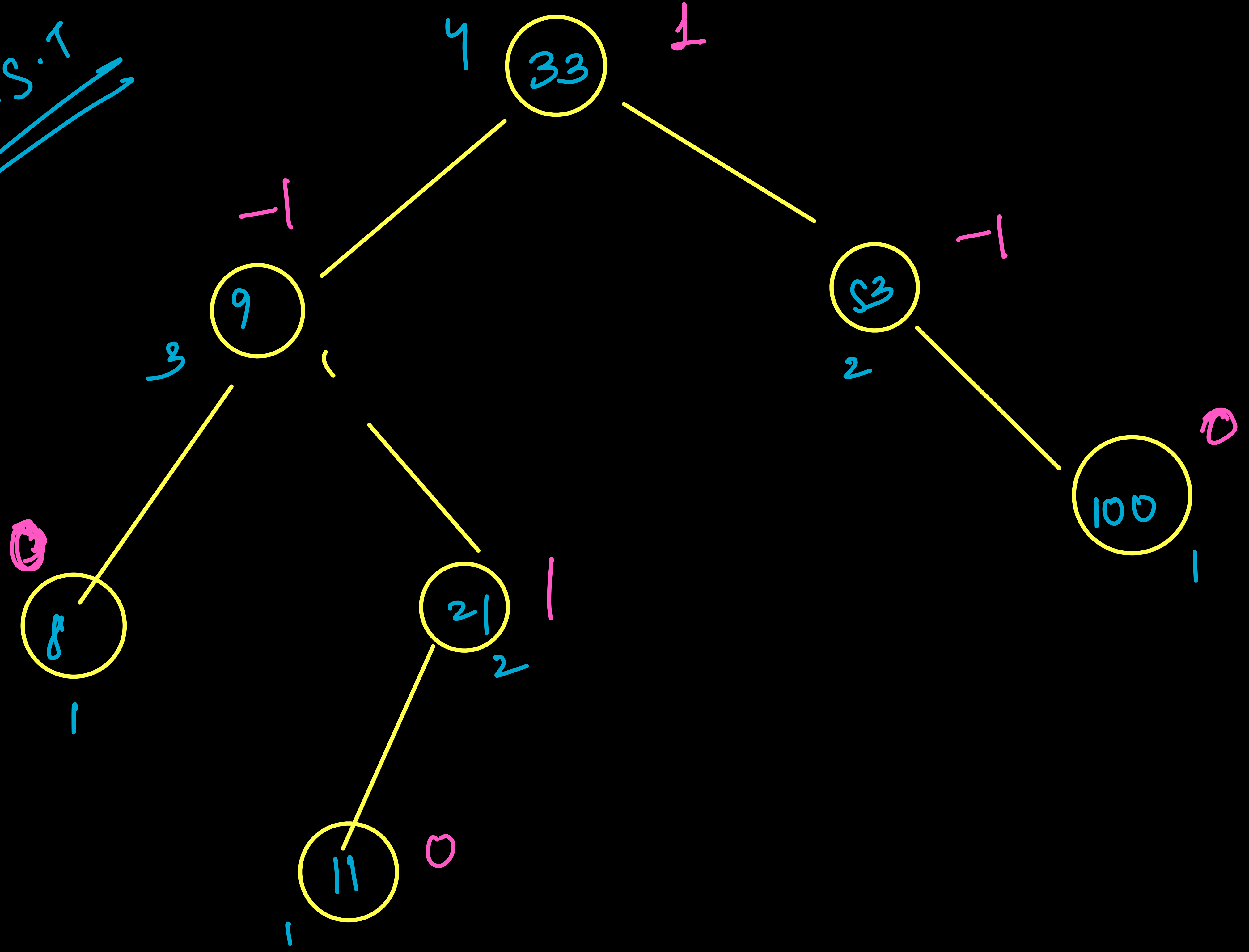
→ insert (10) (9) (8) (7)



→ skewed tree

X not balanced X

Balanced B.S.T



Imp operations in AVL Trees

1. Right Rotation
2. Left Rotation
3. Right Left Rotation
4. Left Right Rotation

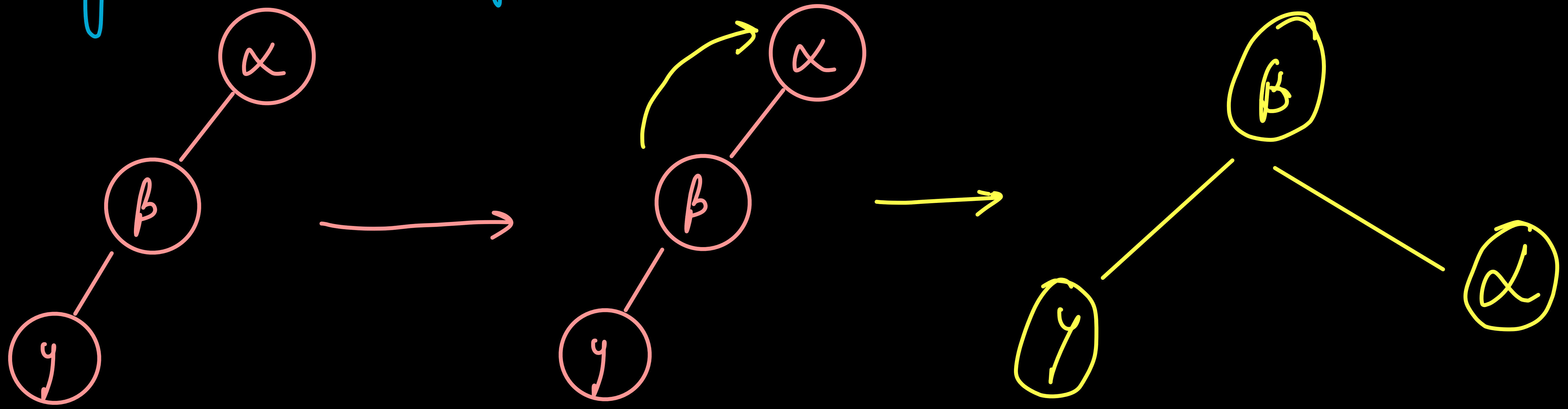
these will be applicable if and only if we know α is unbalanced

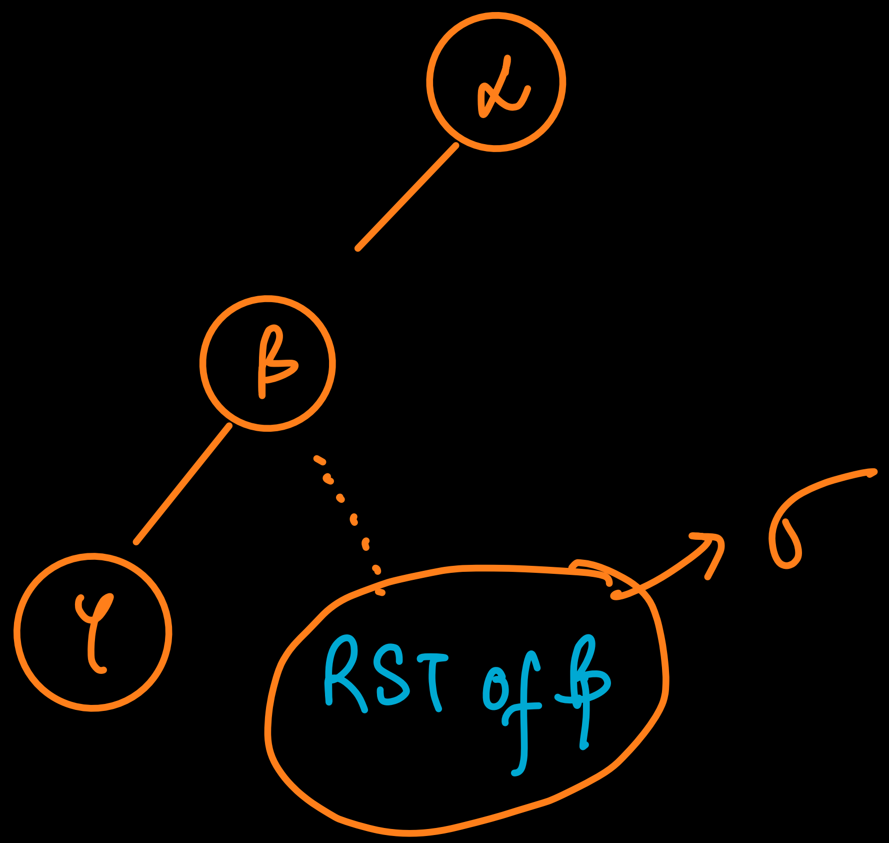
&

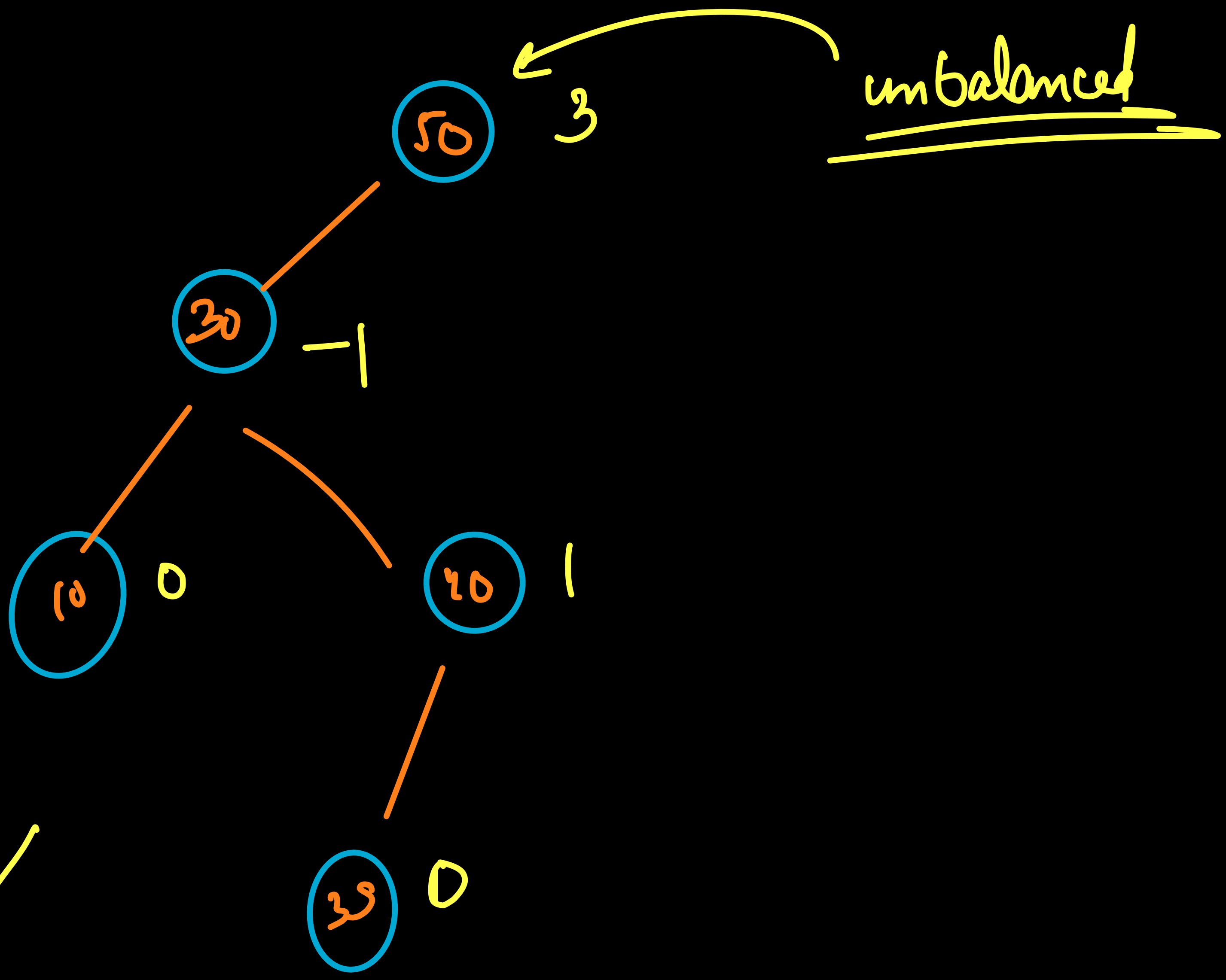
for checking balancing of any node we need height.

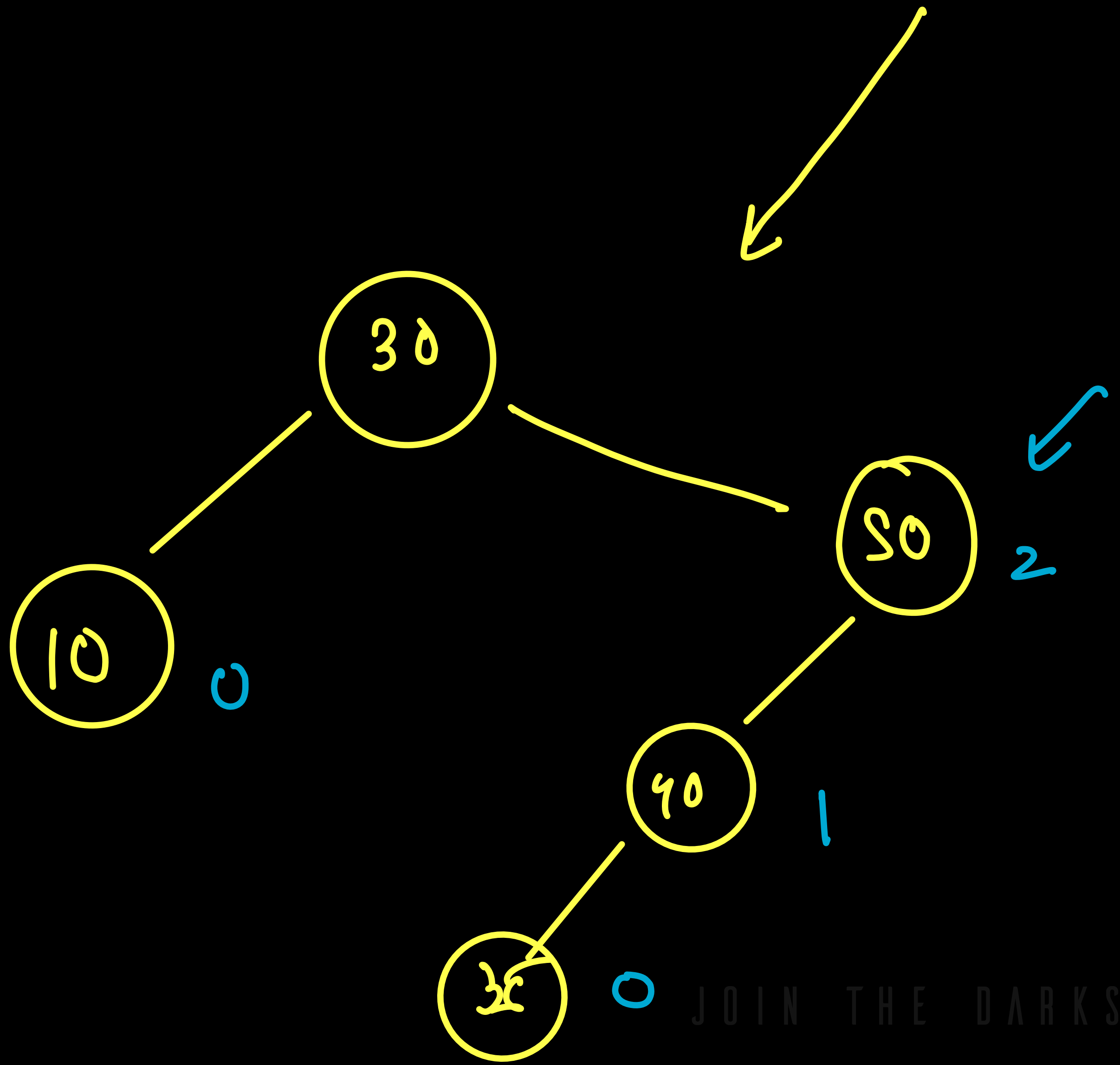
$\boxed{bf}$

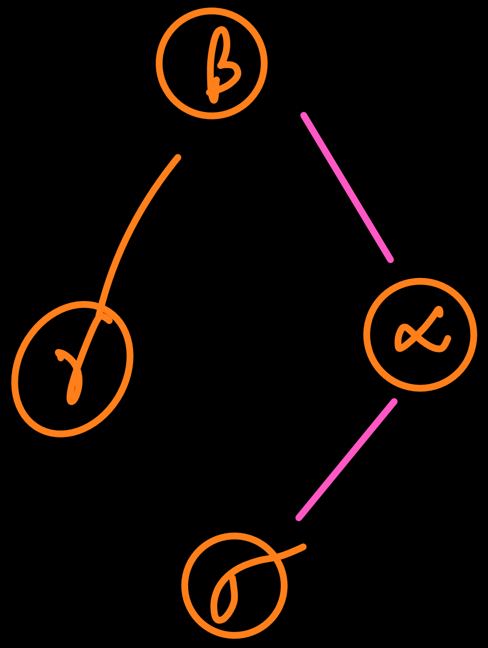
Right Rotation → if an element is added to the left side of my left subtree, then the tree becomes heavy on the left side.



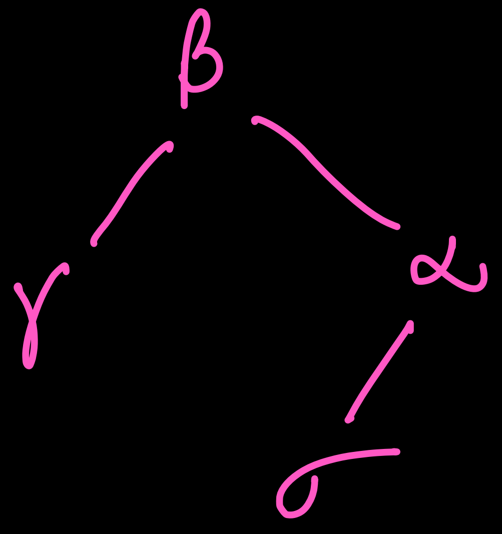

α
β
γ
RST of β
σ


unbalanced
50
3
30
-1
10
0
40
1
35
0

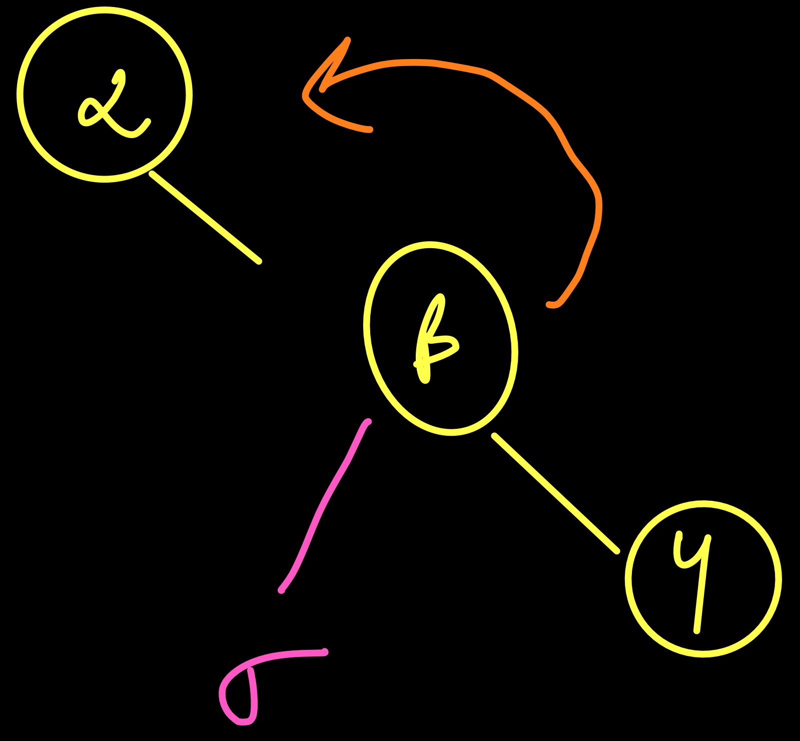

30
10
0
50
2
40
1
35
0

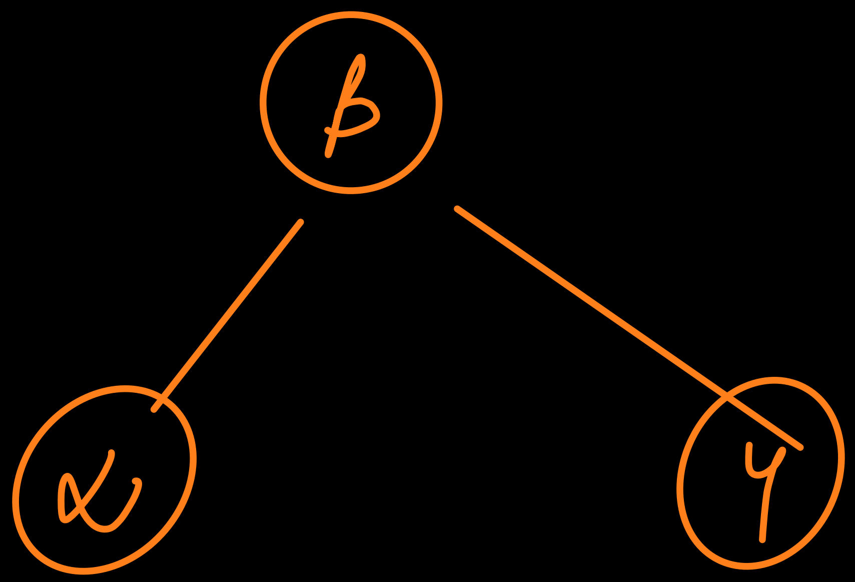


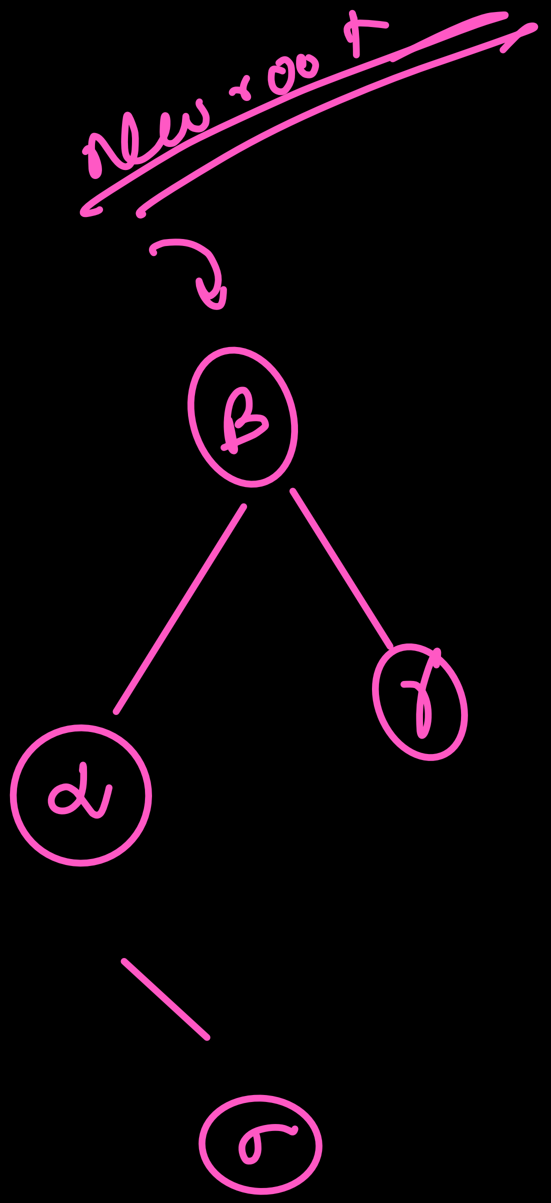
```
rightRotate (alpha) {
     → beta = alpha.left
       sigma = beta.right
       beta.right = alpha
       alpha.left = sigma
       // more ops

       return beta;
}
```



Left Rotation → if node is added to right side of my RST , we have a right - heavy tree

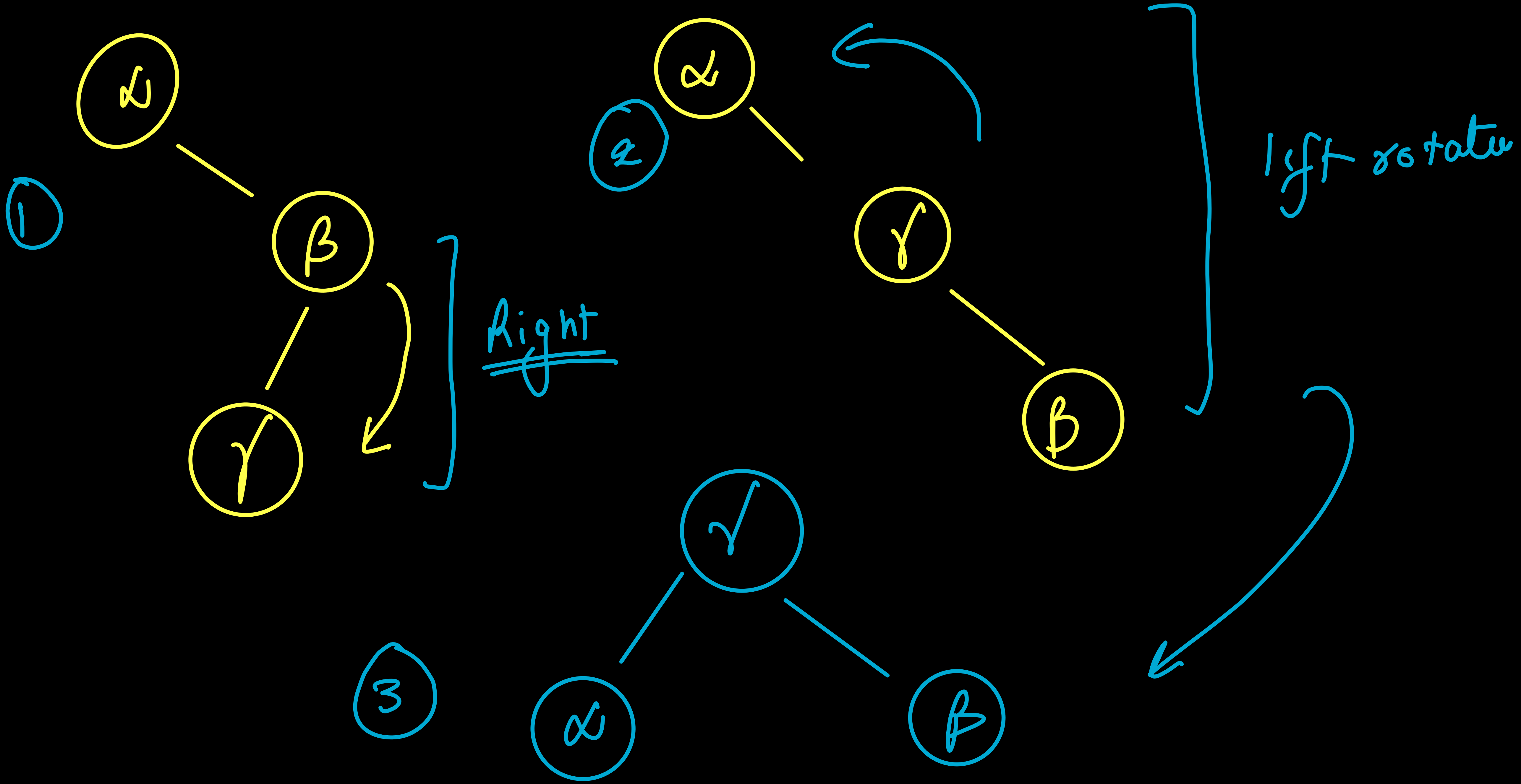






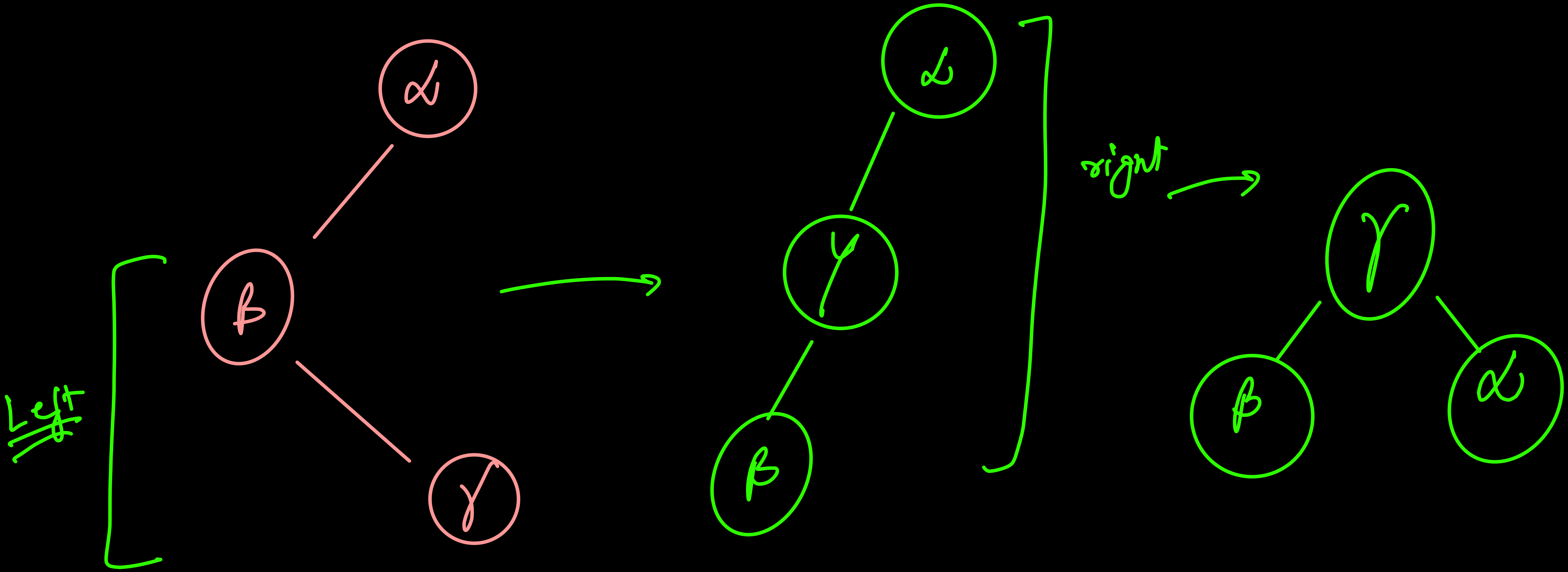
```
leftRotate (alpha) {
        beta = alpha.right;
        Sigma = beta.left;
        beta.left = alpha
        alpha.right = Sigma
        //more ops
        return beta;
}
```

Right -Left Rotation → Right heavy tree but with diff orientation



```
(
    α.right = rightRotata (α.riget)
    return  leftRotate (α)
)
```

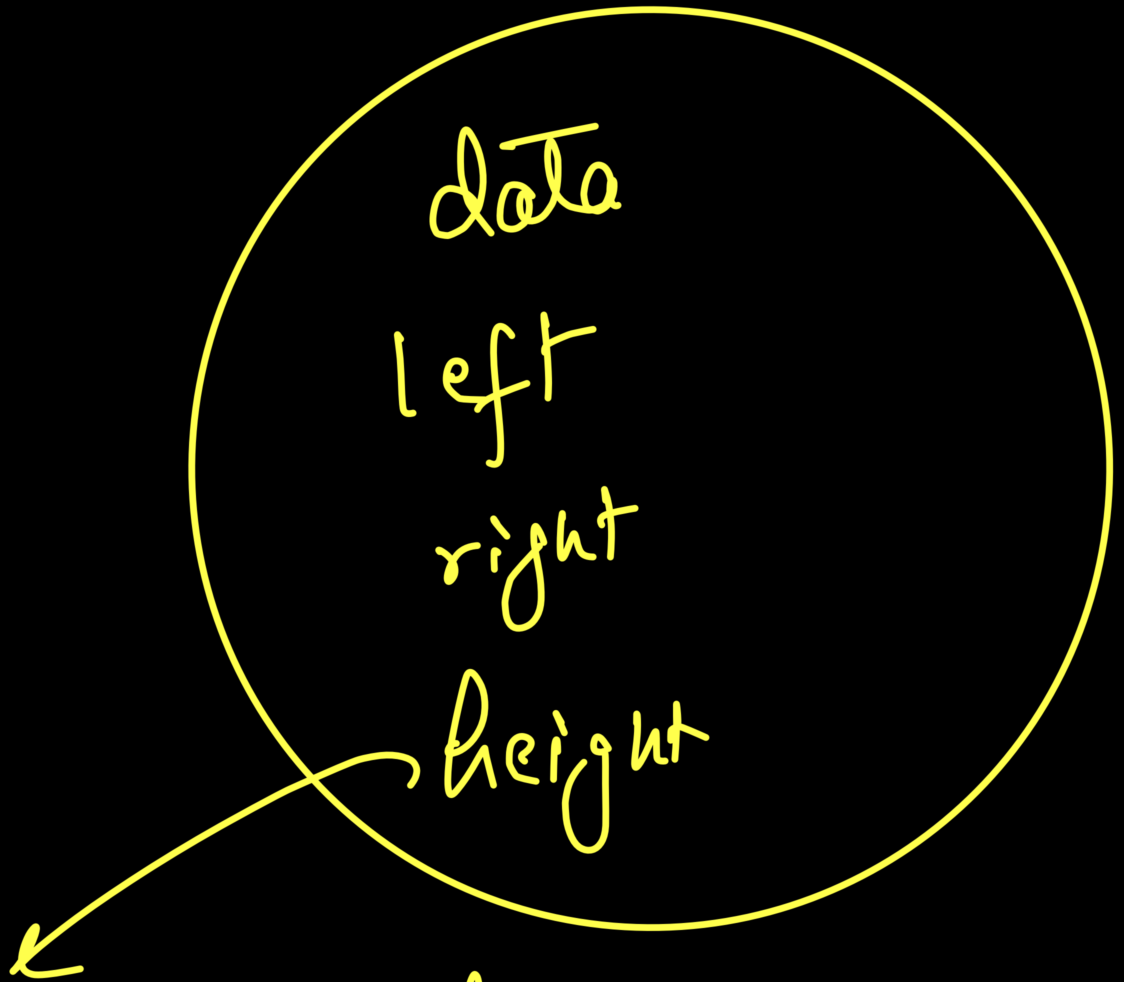
→ Left - Right Rotation



α.left = leftRotate (α.left)

return rightRotate (α)

Node



height of tree rooted here