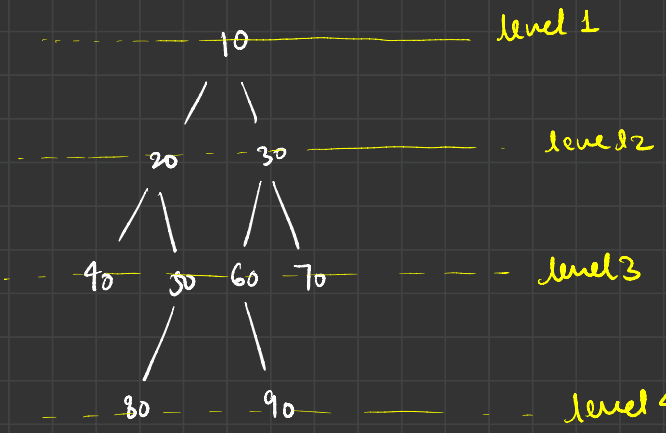
level order traversal

Run que for each level
1. remove from que
2. print
3. add children



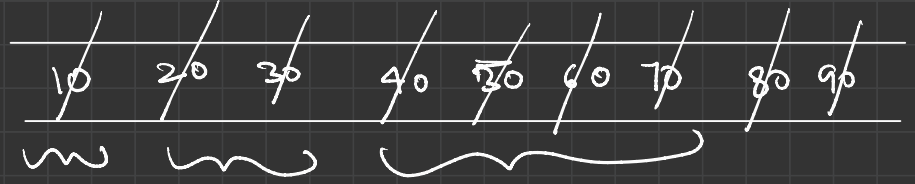
O/P

10

20 30

40 50 60 70

80 90



10

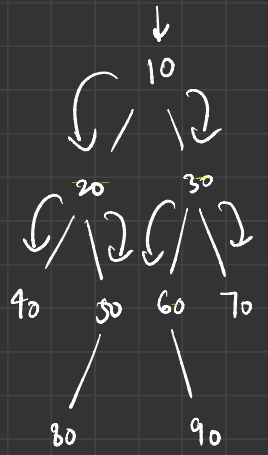
20 30

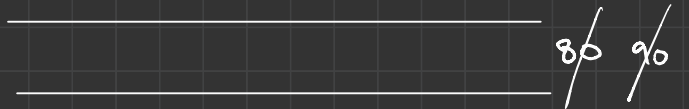
40 50 60 70

80 90

TC : O(N) , SC : O(N)

```java
public void levelOrderTraversal(TreeNode root) {
    if (root == null) {
        return;
    }

    Queue<TreeNode> que = new ArrayDeque<>();
    que.add(root);

    while (que.size() != 0) {
        int size = que.size();

        while (size > 0) {
            TreeNode rnode = que.remove();

            System.out.print(rnode.val + " ");

            if (rnode.left != null) {
                que.add(rnode.left);
            }

            if (rnode.right != null) {
                que.add(rnode.right);
            }

            size--;
        }
    }
}
```





que

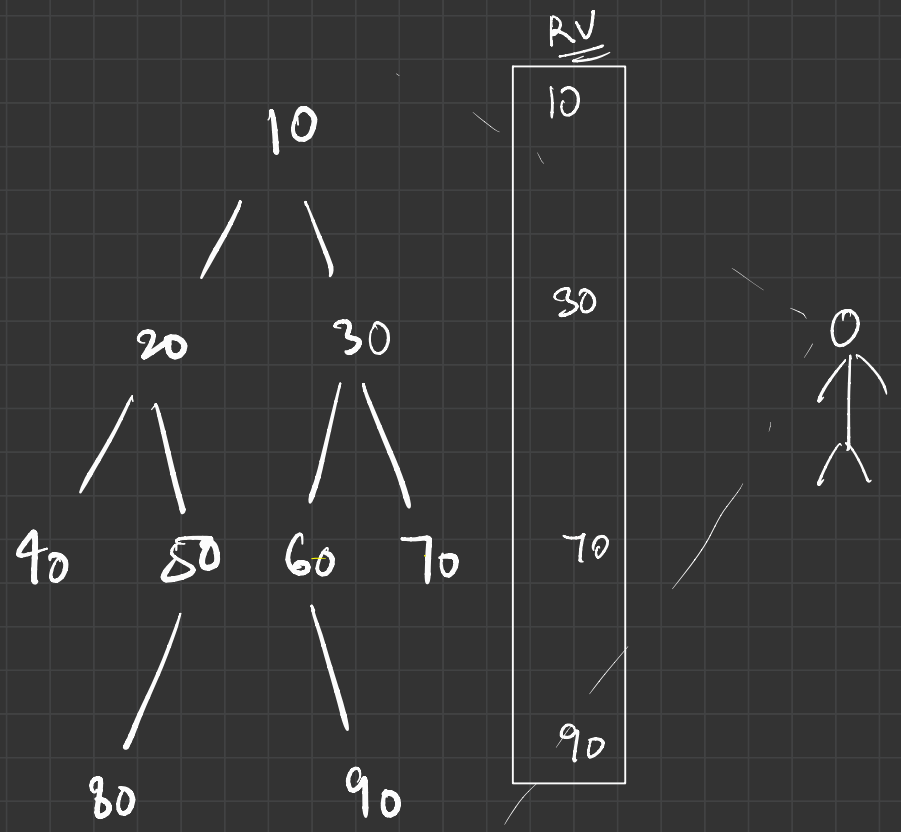
80 90

size = ~~2~~ 1 0

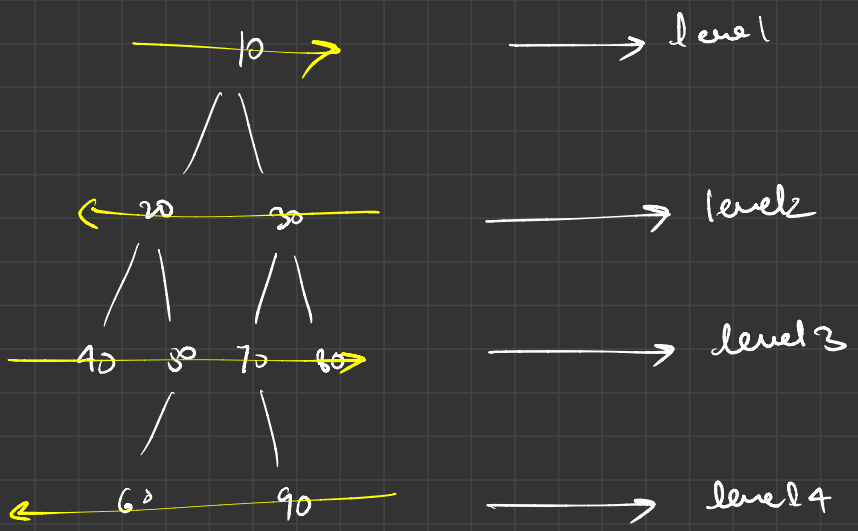
o/p

10	20	30	40	50	60	70	80	90

Right View



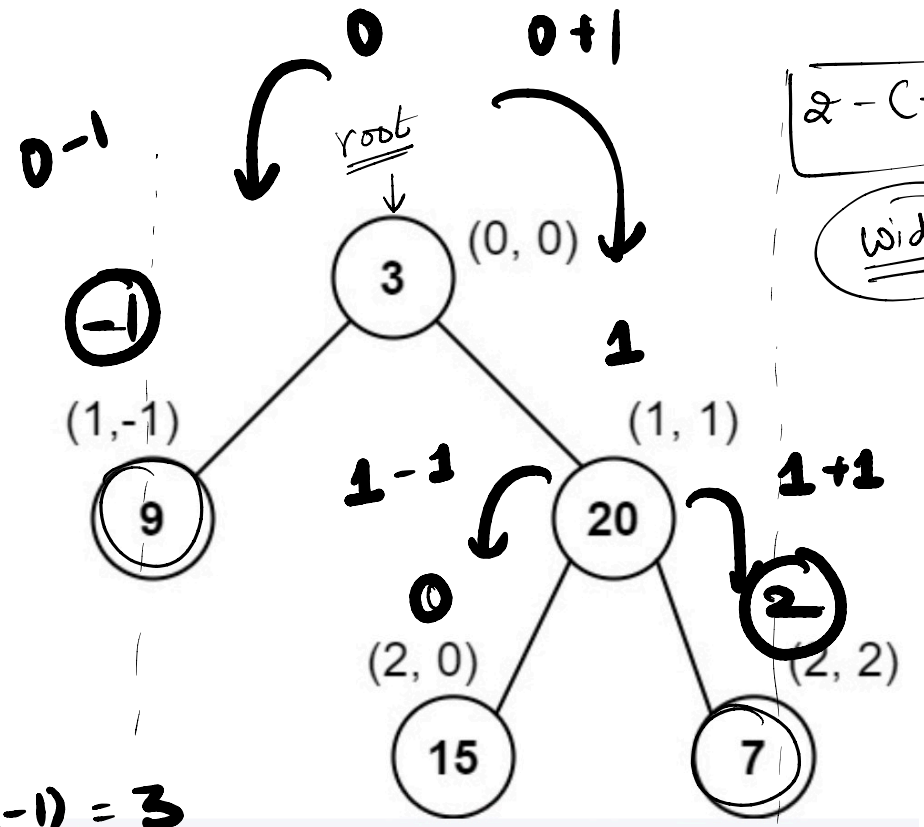
Zig Zag



Ans

{
10
30 20
40 80 70 80
90 60

width.



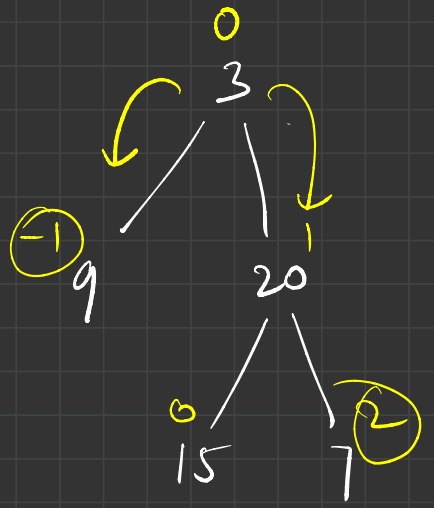
2 - (-1) = 3

width

dist

2 - (-1) = 3

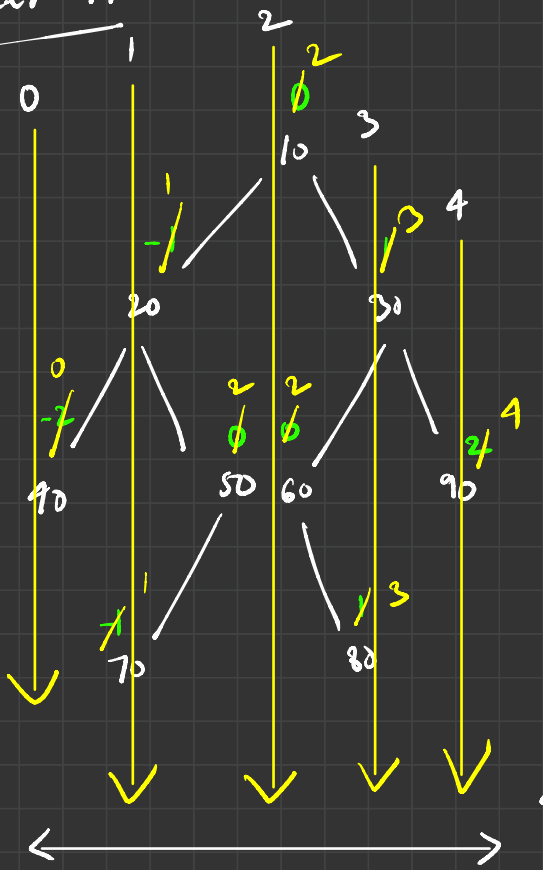
↳ Width!



call (root.left, pos-1);

call (root.right, pos+1);

Vertical Order Traversal



o/p

0 40

1 20 70

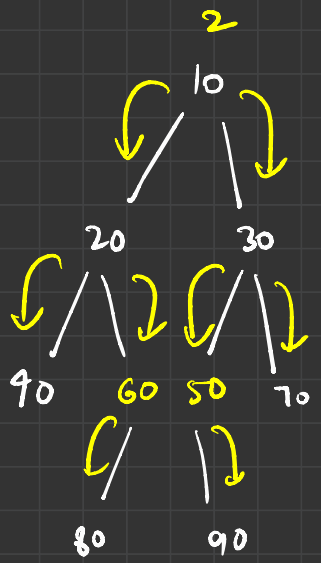
2 10 50 60

3 30 80

4 90

pos = (− minPos)

{No. of levels = width+1}



0	40
1	20 80
2	10 50 60
3	30 90
4	70

(10,2) (20,1) (30,3) (40,0) (50,2) (60,2) (70,4) (80,1) (90,3)

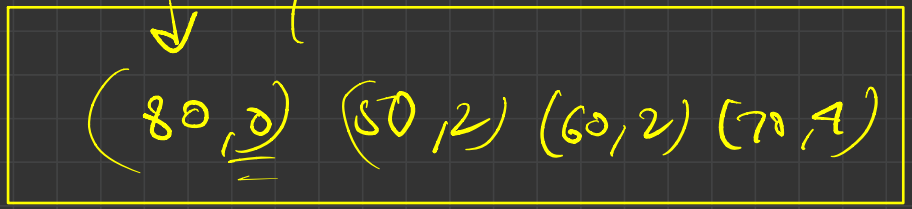
Priority Queue

(80,0) (60,2) (50,2) (70,4)

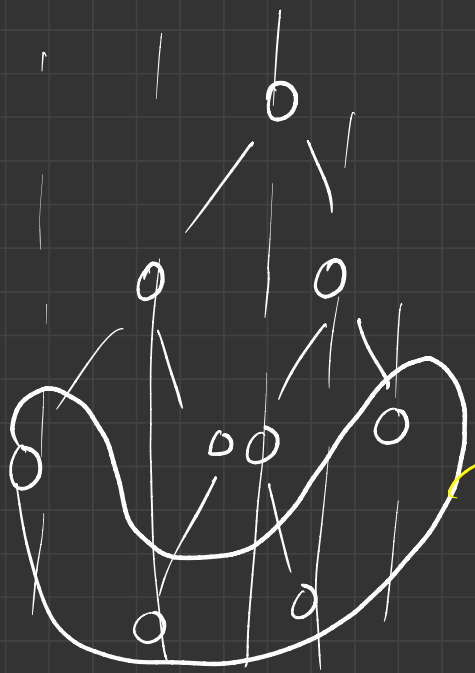
O(N log N)

① get lowest pos in frow

② if lowest pos is same then need lowest value first.



Bottom View



→ last Nodes in VO