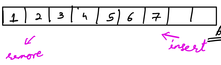
Trees 2

Wednesday, 22 September 2021 9:05 PM

Level Order Traversal

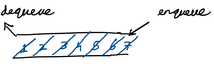
o/p 1 → 1 2 3 4 5 6 7

Array ✓



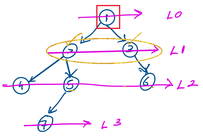
T O(N)
SC = O(N)

Queues ✓ dequeue enqueue



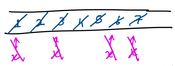
o/p → 1 2 3 4 5 6 7 (Ans)

o/p 2 →
1
2 3
4 5 6
7 ✓



T =

Queues

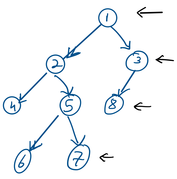


o/p → 1 2 3 4 5 6 7 ✓ (Ans)

```
void levelOrder(root) {
    if(root == null) return;
    q = new Queue();
    q.enqueue(root);
    last = root;
    while(!q.isEmpty()) {
        x = q.dequeue();
        if(x.left != null) q.enqueue(x.left);
        if(x.right != null) q.enqueue(x.right);
        print(x.data);
        if(last == x) {
            print("\n");
            last = q.rear();
        }
    }
}
```

TC = O(N)
SC = O(N)

Right View

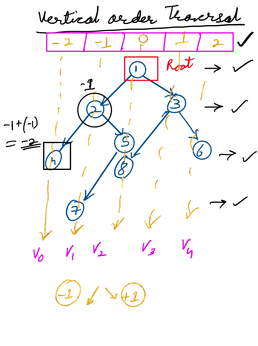


o/p → 1 3 8 7

o/p is last node of each level.

HW → Left View of Binary Tree

Vertical order Traversal



o/p → 4 ✓
2 7 ✓
1 5 8 ✓
3 ✓
6 ✓

TC = O(N)
SC = O(N)

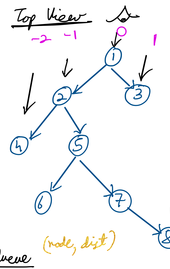
Map

Key (distance)	Value (list of nodes)
0	{1, 5, 8} ✓
-1	{2, 7} ✓
1	{3} ✓
-2	{4} ✓
2	{6} ✓

Value → node

min, min+1, min+2, ---, max-1, max ✓

Top View



o/p → 4 2 1 3 8 ✓ (Ans)

↳ First element in vertical order traversal.

TC = O(N)
SC = O(N)

Map

Key (distance)	Value (Node)
0	1 ✓
-1	2 ✓
1	3 ✓
-2	4 ✓
2	8 ✓

min

Queue (node, dist)

(1,0) (2,-1) (3,1) (5,-2) ...

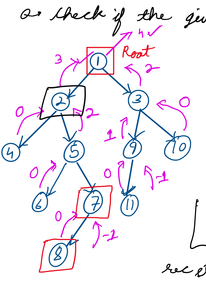
HW → Bottom View

Height Balanced Tree

∀ nodes → |Height(left subtree) − Height(right subtree)| ≤ 1

Height of tree → Distance from root to farthest leaf.

Q. Check if the given tree is height balanced ?



rec stack

```
isBalanced = true;
int height(root) {
    if(root == null) return -1;
    l = height(root.left);
    r = height(root.right);
    if(abs(l-r) > 1) isBalanced = false;
    return max(l, r) + 1;
}
```

TC = O(N)
SC = O(H)