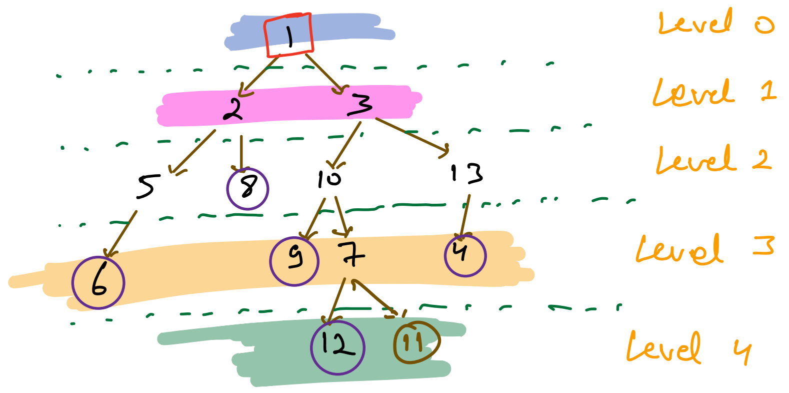
Welcome ☺

Agenda : Level order Traversal
Question
Top View / Bottom View
Type of trees
1 question.

Iterative PreOrder / PostOrder.

Level Order Traversal.



✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

⇒ 1 2 3 5 8 10 13 6 9 7 4 12 11

Code Level by level ⇒ Queue D.S

~~1~~ ~~2~~ ~~3~~ ~~5~~ ~~8~~ ~~10~~ ~~13~~ ~~6~~ ~~9~~ ~~7~~ ~~4~~ ~~12~~ ~~11~~

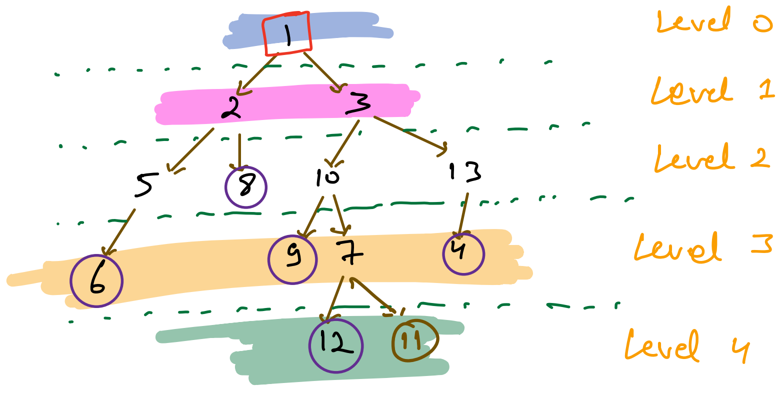
Code

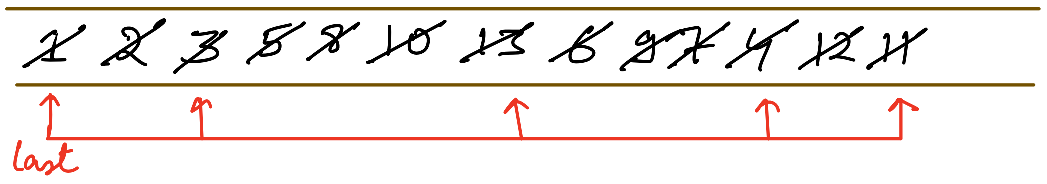
```
q.enqueue(root)
while ( ! q.isEmpty() )
{
    n = q.dequeue() ;
    print( n.data)
    if( n.left != NULL) q.enqueue( n.left)
    if(n.right != NULL) q. enqueue( n.right)
}
```

T.C → O(N)
S.C → O(N)

Print level by level in seperate line

✓1
✓2 ✓3
✓5 ✓8 ✓10 ✓13
✓6 ✓9 ✓7 ✓4
✓12 ✓11

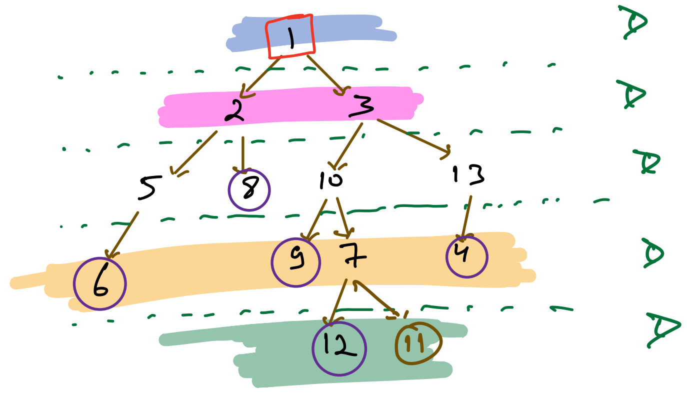




```
q.enqueue(root)
 last = root
while ( !q.isEmpty() )
{
    n = q.dequeue();
    print(n.data)
    if(n.left != NULL) q.enqueue(n.left)
    if(n.right != NULL) q.enqueue(n.right)
    if( n == last && !q.isEmpty() ) {
            print("\n")
            last = q.rear()
    }
}
```

T.C → O(N)
S.C → O(N)

Q Print right view of binary tree.



⇒ 1 3 13 4 11

Soln → print last node of every level

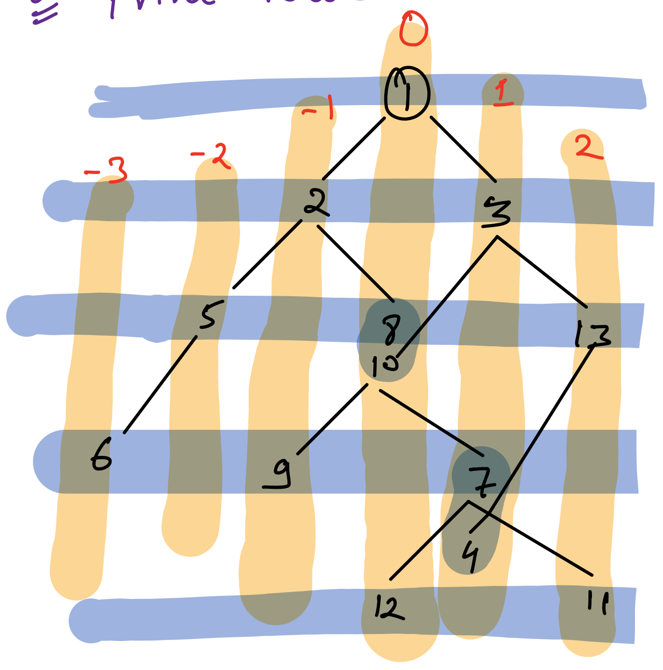
```
q.enqueue (root)
last = root
while ( ! q.isEmpty() )
{
    n = q.dequeue() ;
    if( n.left != NULL) q.enqueue ( n.left)
    if(n.right != NULL) q. enqueue ( n.right)

    if( n == last     ) {
        print ( n.data)
        if ( !q.isEmpty ())     last = q.rear()
    }
}
```

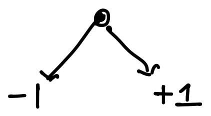
T.C → O(N)

S.C → O(N)

Swap h.w left view

Q Print vertical order traversal.



1) 

2) Overlap → first print data from left & then from right

3) Print from top to bottom

o/p ⇒

-3	→	6			
-2	→	5			
-1	→	2	9		
0	→	1	8	10	12
1	→	3	7	4	
2	→	13	11		

1) Need to know distance of node from root node.

→ Use Hashmap

2) Use level order traversal.

(node, ds)

~~(1,0)~~ ~~(2,-1)~~ ~~(3,1)~~ ~~(5,-2)~~ ~~(8,0)~~ ~~(10,0)~~ ~~(13,2)~~ ~~(6,-3)~~ ~~(9,-1)~~ ~~(7,1)~~ ~~(4,1)~~ ~~(12,0)~~ ~~(11,2)~~

Key Vertical ds	Value List < Node >
0	1, 8, 10, 12
-1	2, 9
1	3, 7, 4
-2	5
2	13, 11
-3	6

→ maintain min level & max level

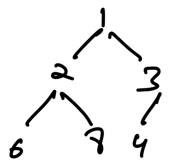
Top View → first node for each vertical distance

Bottom View ? → last node for each vertical distance.

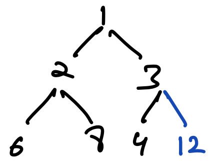
Type of trees

1) Proper binary tree → Every node has either 0 or 2 children.

2) Complete binary tree → Every node has 2 children except maybe the last level. All nodes of last level are as left as possible.



3) Perfect Binary Tree → All levels are complete
→ All are also Complete Binary Tree
→ Also Proper Tree.



$\boxed{\text{BST} ✓}$