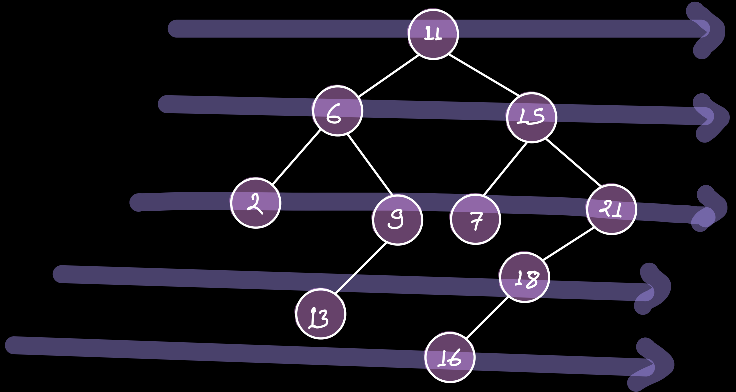
Q Given a BT. Print the level order traversal.

```
[  [11],         ← 0
   [6, 15],      ← 1
   [2, 9, 7, 21],
   [13, 18],
   [16]
]
```



App 1   Add level along with node in the queue.

~~<11, 0>~~ <6, 1> <15, 1>

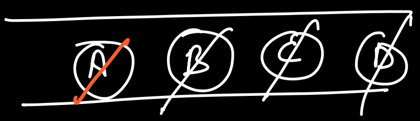
TC: O(N)
SC: O(Width of tree)
: O(N)
↳ $\frac{(N+1)}{2}$

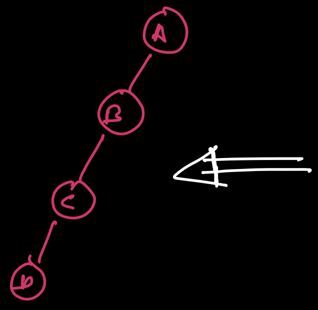
App 2   Add a marker after every level

```
[
  [11],
  [6, 15]
  [.          ]
  ⋮
]
```

~~11~~ Null ~~6~~ ~~15~~ Null 2 9 7 21 Null

TC: O(N)
SC: O(N)
↳ $(\frac{N+1}{2})$

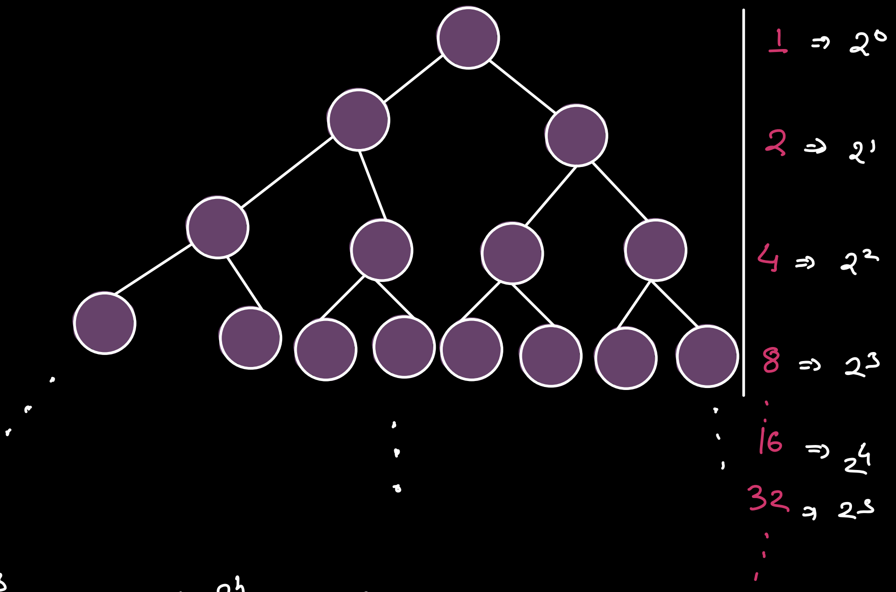




Complete Binary Tree

→ A Binary tree where all the levels are completely filled
except possibly the last level.
Nodes in the last level are left aligned.

Ht(CBT) ⇒ log N



→ 2^h

$$2^0 + 2^1 + 2^2 + 2^3 + \dots + 2^h = N$$

$a = 2^0$
$r = 2$
$n = h+1$

$$\frac{2^{(h+1)} - 1}{2-1} = N$$

$$\Rightarrow 2^{(h+1)} - 1 = N$$

$$2^{(h+1)} = N+1$$

$$h+1 = \log(N+1)$$

$$h = \log(N+1) - 1$$

$$\boxed{h = O(\log N)}$$

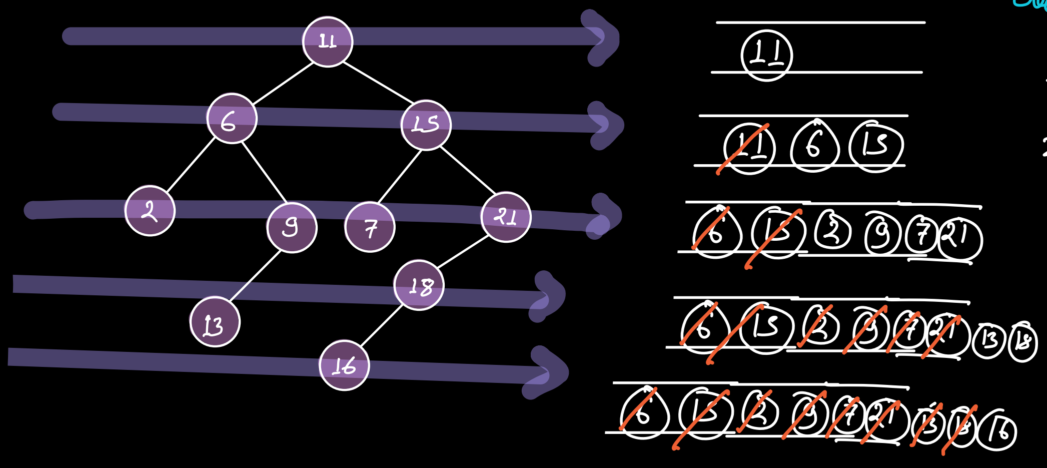
If all levels are completely filled,

Count of nodes in the last level $= 2^h$

$$= 2^{\log(N+1) - 1}$$

$$= \frac{2^{\log(N+1)}}{2}$$

$$= \frac{N+1}{2}$$

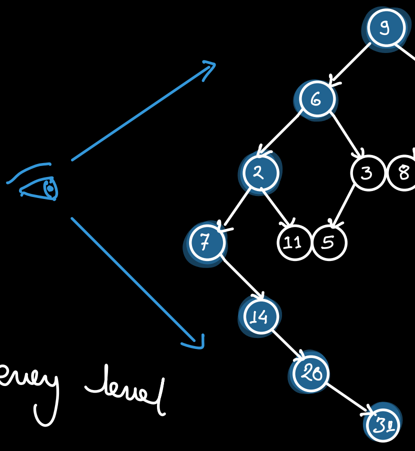


No of deque	Size of Q
1	1
2	2
4	4
2	2
1	1

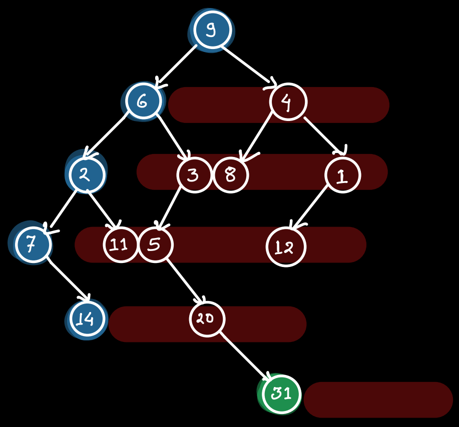
```
List < List < Int>>  levelOrder ( root) {
        if( root = = null) ret null/ empty list ;
        List<List< Int>> ans  =  - - - ;
        Queue < TreeNode > Q = - - - ;
        Q.add( root);
        While ( ! Q.isEmpty())  {
            List < Int> level = new ArrayList <Int>();
            Size =  Q.size();
            for ( i=0;  i < size ;  i++ ) {
                TreeNode temp = Q. poll();
                level.add( temp.val);
                if( temp. left  != null) {
                        Q.add (temp . left);
                }
                if ( temp. right  != null) {
                        Q. add (temp. right);
                }
            }
            ans. add( level);
        }
        ret  ans;
}
```

Amazon
MS
Adobe

Q Given a BT. Print the left view of the tree.

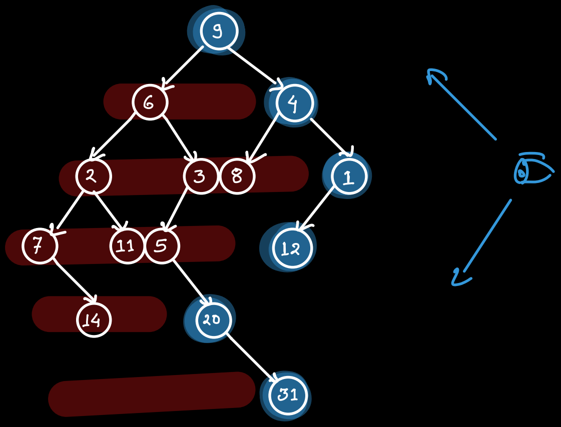


Left view ⇒ First node of every level
(leftmost)



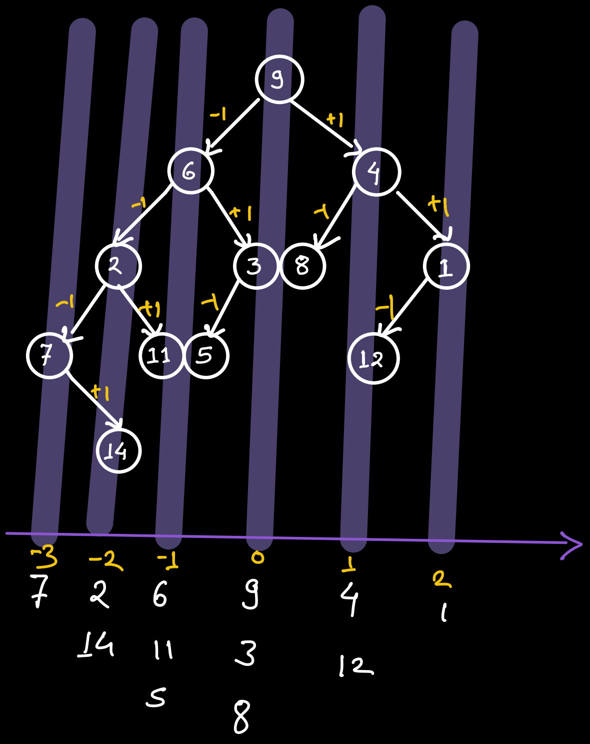
Q Given a BT. Print the right view of the tree

Right view ⇒ Right most node
of all levels.



Q Vertical order traversal.

```
[
   [7],
   [2, 14],
   [6, 11, 5],
   [9, 3, 8],
   [4, 12],
   [1]
]
```



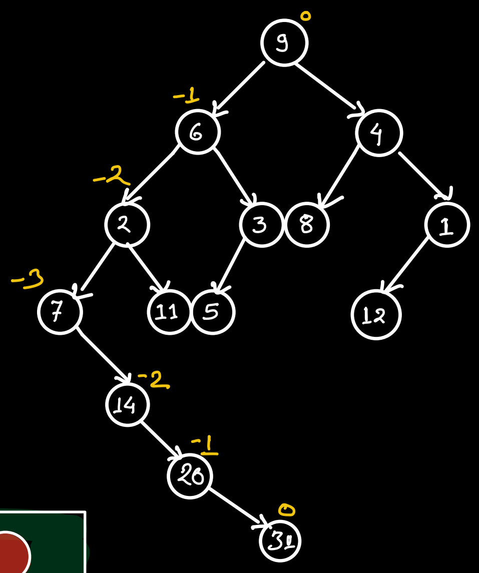
HashMap< Int, List<Int>> map;
↓ dist
↓ List of nodes 'dist' apart from root.

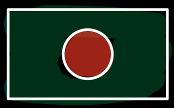
```
PreOrder ( root, dist ) {   0
    // Base case

    if ( ! map.containsKey ( dist ) ) {
          map.put( dist, new ArrayList< Int>() );
    }
    map.get (dist). add ( root.val);
    preOrder ( root. left, dist -1);
    PreOrder ( root. right, dist +1);
}
```

Hash Map

dist	:	list of nodes	
0	:	[9, 31]	?
-1	:	[6, 20	?
-2	:	[2, 14]	
-3	:	[7]	

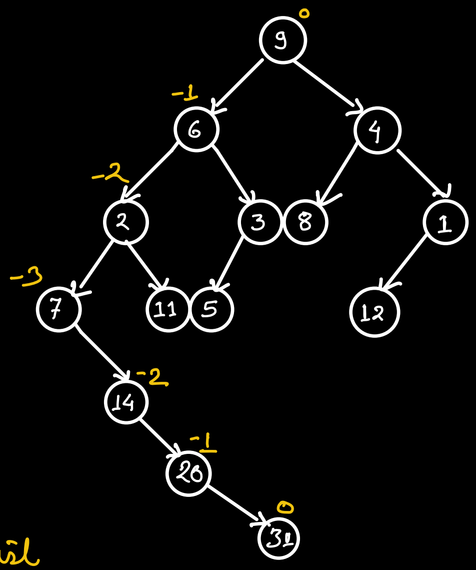




Level Order

Queue < TreeInfo >

~~{9, 0}~~ ~~{6, -1}~~ ~~{4, 1}~~



```
TreeInfo {
    TreeNode node;
    Int dist,
    Int level;
}
```

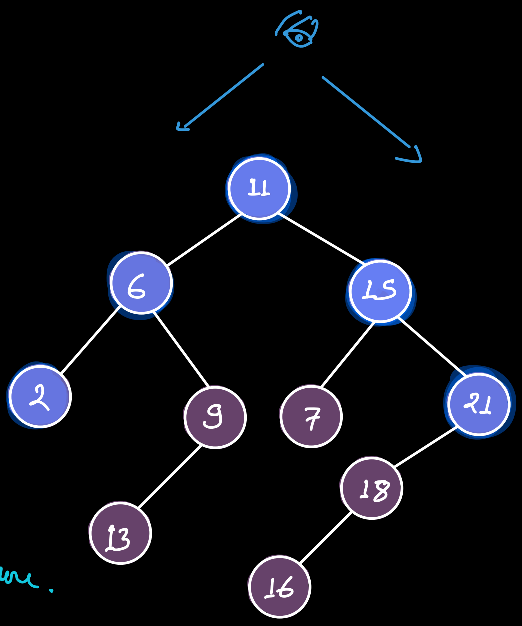
HashMap

dist	:	List
0	:	[9]
-1	:	[6]
1	:	[4 . . .
-3	:	

Top View

First node in the list againt every key in map.

If list against a key is present & has a node → don't add more.

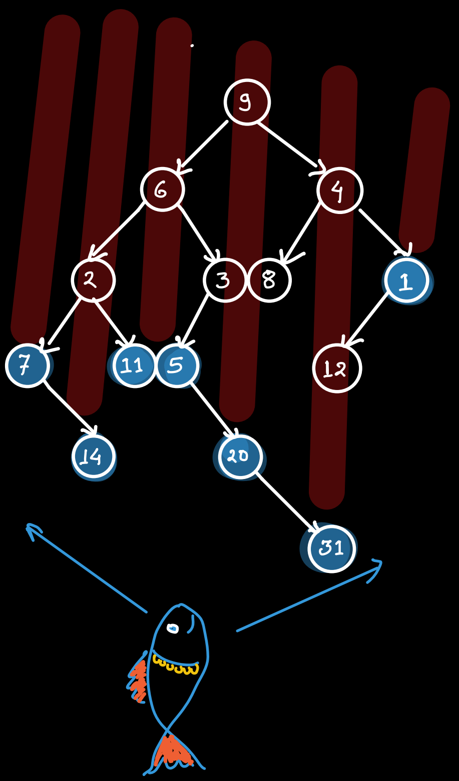


Bottom View

Last value in the list against every key in the map.

$-1 : [6, \underbrace{11, 5}_{\text{Same level}}]$

$-2 : [2, 14]$


9
6
4
2
3
8
1
7
11
5
12
14
20
31