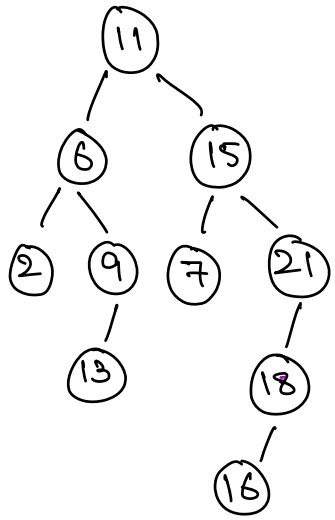
Q. Given a Binary Tree, Print the level order traversal.



11, 6, 15, 2, 9, 7, 21, 13, 18, 16

Queue < TreeNode > Q;

Q

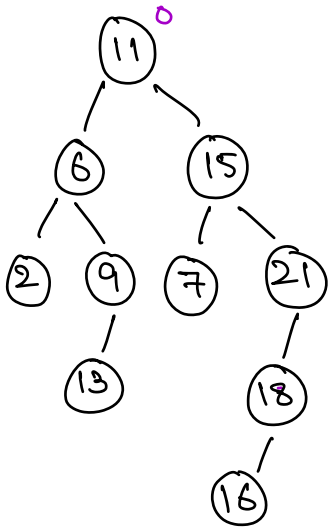
~~11~~, ~~6~~, ~~15~~, ~~2~~, ~~9~~, ~~7~~, ~~21~~, ~~13~~, ~~18~~, ~~16~~

11, 6, 15, 2, 9, 7, 21, 13, 18, 16

* List < List < int > >

[[11],
[6,15],
[2,9,7,21],
[13,18],
[16]]

Approach 1 :



Q.

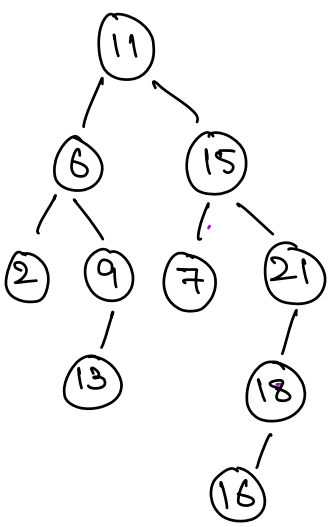
<11,0>, <6,1>, <15,1>, <2,2>, <9,2>, <7,2>, <21,2>, <13,3>

---

[11]
[6,15]
[2,9,7,21]
⋮

Approach 2 :-

⇒ Adding a marker / delimeter after every level.



Queue<TreeNode> Q

11, Null, 6, 15, Null, 2, 9, 7, 21, Null, 13, 18, Null, 16, Null

[11]
[6, 15]
[2, 9, 7, 21]
[13, 18]
[16]

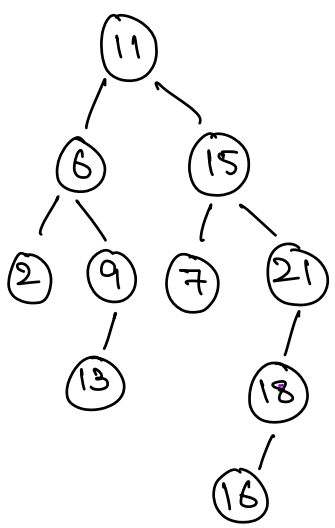
TC: O(N)

SC: O(Width) ⇒ O(N)

↳ Max. no. of Nodes in any level (Width of B.T)

11, 6, 15, 2, 9, 7, 21, 13, 18, 16

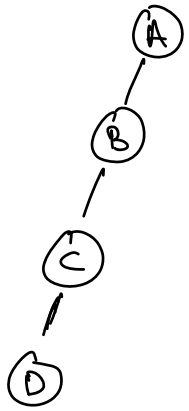
Size = 4



Skew Tree

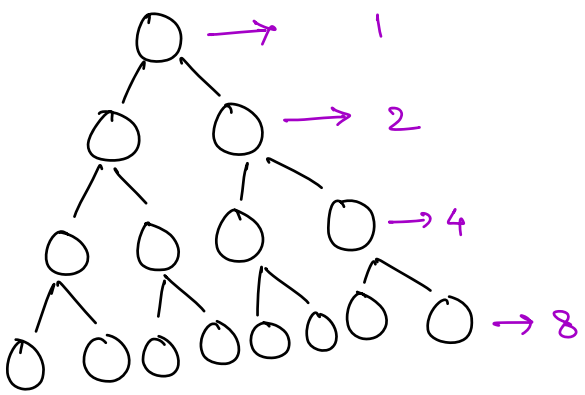
A, B, C, D

Size = 4



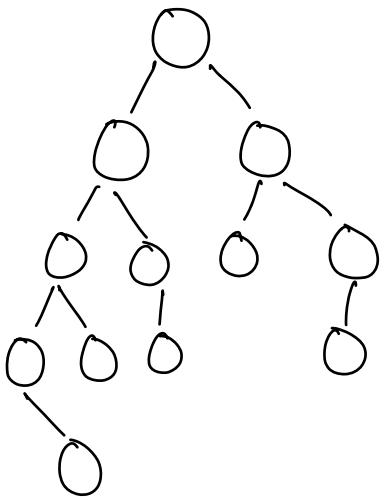
Complete Binary Tree

⇒ All the levels are completely filled.

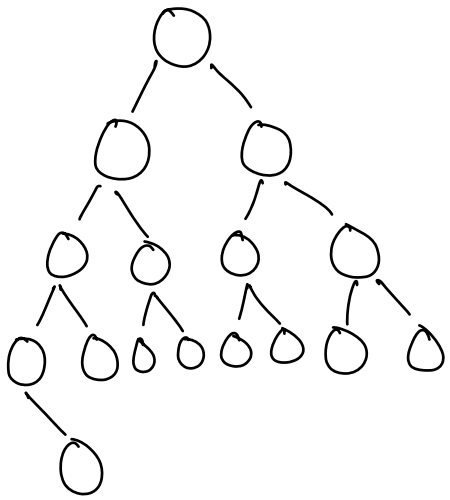


A binary tree where all the levels are completely filled

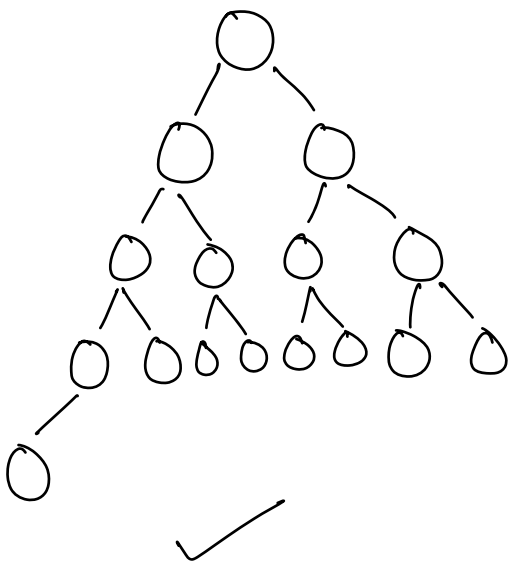
- ↳ except possibly the last level
- ↳ Nodes in the last level are left aligned.

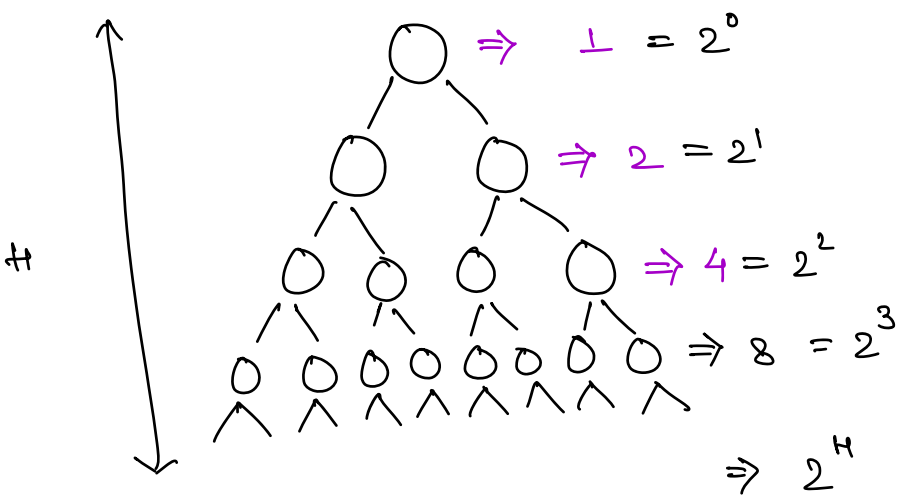


Not complete B.T



Not complete B.T





No. of nodes in the last level $= 2^H$

Total no. of nodes in the Tree $\Rightarrow$

$$2^0 + 2^1 + 2^2 + 2^3 + \cdots\cdots + 2^H$$

$\Rightarrow$ G.P

$a = 1$

$r = 2$

# of terms $= H+1$

$$\text{Sum} = \frac{1(2^{H+1} - 1)}{(2-1)} = 2^{H+1} - 1$$

$$\text{No of nodes} = 2^{H+1} - 1 = N.$$

$$2^{H+1} - 1 = N$$

$$2^{H+1} = N+1$$

$$H+1 = \log_2 (N+1)$$

$$\boxed{H = \log (N+1) - 1}$$

$$H \approx O(\log N)$$

No. of nodes in the last level $= 2^{H}$

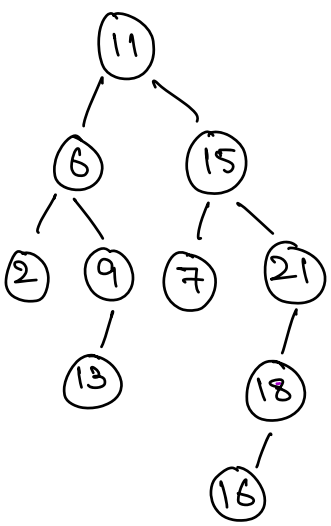
$= 2^{\log_2(N+1) - 1}$

$= \frac{2^{\log_2(N+1)}}{2}$

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

Worst Case Space Complexity of level order $\Rightarrow O(N)$

Approach 3 :



Q	No. of dequeues	Size
11	1	1
6, 15	2	2
2, 9, 7, 21	4	4
13, 18	2	2
16	1	1

No. of dequeue's for every level = size of the Queue.

Code :-

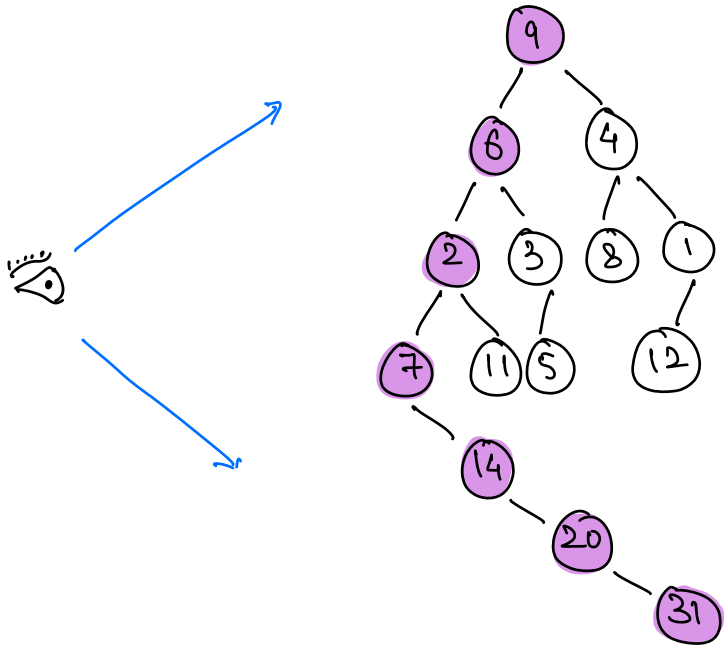
```
List<List<int>> levelOrder(root) {
    if(root == null)
        return null/empty list;
    List<List<int>> ans;
    Queue<TreeNode> Q;
    Q.enqueue(root);
    while(!Q.isEmpty()) {
        int size = Q.size();
        List<int> level;
        for(i=0; i<size; i++) {
            TreeNode temp = Q.poll();
            level.add(temp.data);
            if(temp.left != null)
                Q.enqueue(temp.left)
            if(temp.right != null)
                Q.enqueue(temp.right)
        }
        ans.add(level);
    }
    return ans;
}
```

TC: O(N)

SC: O(N) (worst case)

Q: Given a Binary Tree, Print the left view of it.

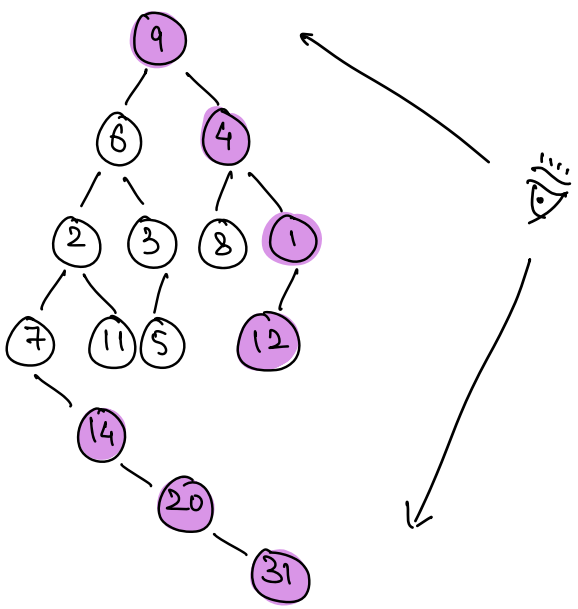
Amazon
MS
Adobe



Left View ⇒ 9, 6, 2, 7, 14, 20, 31

↳ First node of every level in Level Order traversal.

Q: Given a Binary Tree, Print the right view of it.

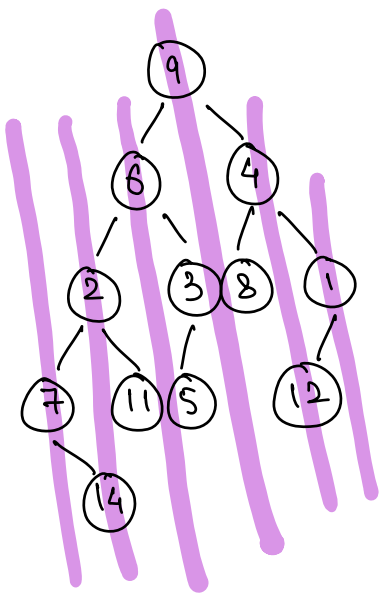


Right view ⇒ 9, 4, 1, 12, 14, 20, 31

↳ Last node of every level in Level Order traversal.

* print the node only for i == size-1 in prev. code.

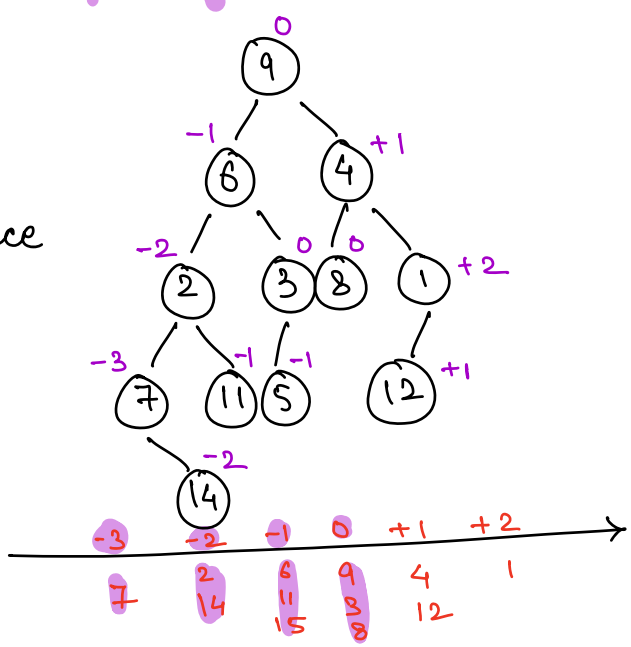
Q. Vertical Order Traversal.



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

Vertical Order traversal

⇒ Nodes with same distance from root node will come under same Vertical line.



HashMap < int, List < int > > map

↓ dist

↓ Nodes which are dist apart from root node

```
preOrder(root, dist) {
        if( root == Null)
                return ;
        if(!map.contains(dist)){
            map.insert( dist, new ArrayList<int>());
        }
        map.get(dist).add(root.val);
        preOrder(root.left, dist-1);
        preOrder(root.right, dist+1);
}
```

HashMap

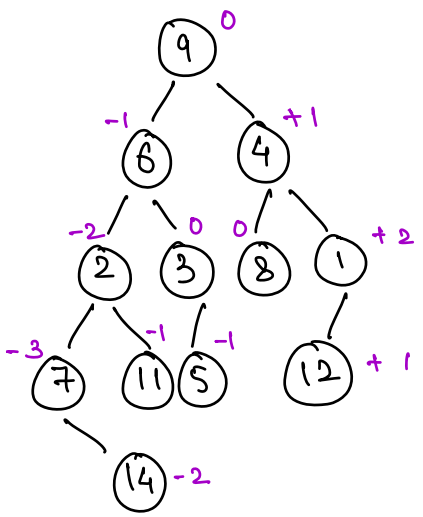
int : List of Node value

0 : 9

-1 : 6, 20

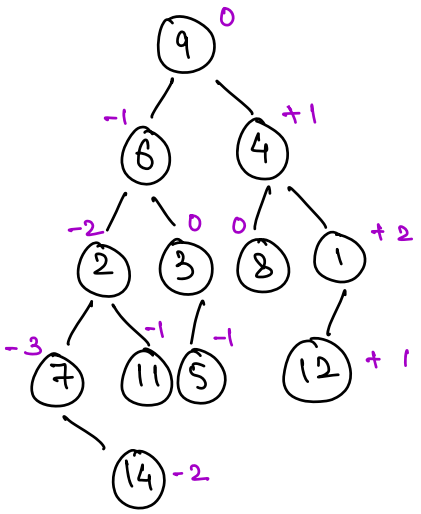
-2 : 2, 14

-3 : 7



* Post Order
* In Order
* Pre Order

} DFS ✗




```
Queue < Info > Q;

Class Info {
    TreeNode node;
    int dist;
}
```

~~{9,0}~~ , ~~{6,-1}~~ , ~~{4,1}~~ , ~~{2,-2}~~ , ~~{3,0}~~ , ~~{8,0}~~ , ~~{1,2}~~ , ~~{7,-3}~~

~~{11,-1}~~ ~~{5,-1}~~ ~~{12,1}~~ {14,-2}

~~HashMap~~ :-

int : List of Node value

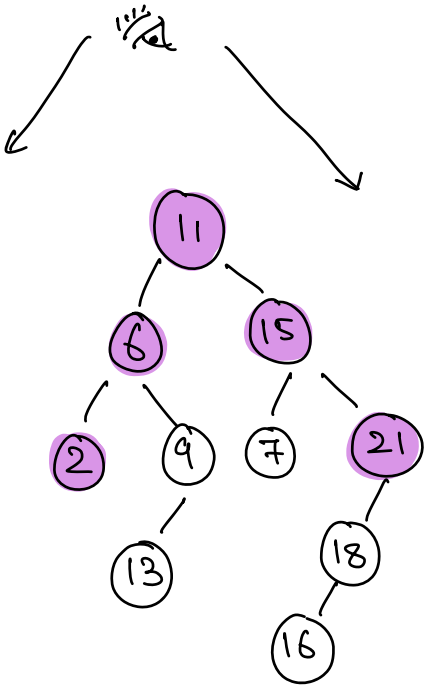
0 : [9, 3, 8]
-1 : [6, 11, 5]
1 : [4, 12]
-2 : [2, 14]
2 : [1,]
-3 : [7,]

⇒

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

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

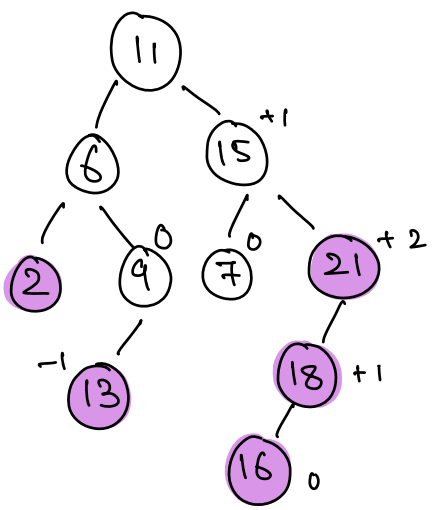
Q: Top View



⇒ First node in the list against every dist.

Q: Bottom View.

⇒ Last node in the list against every in map.



*