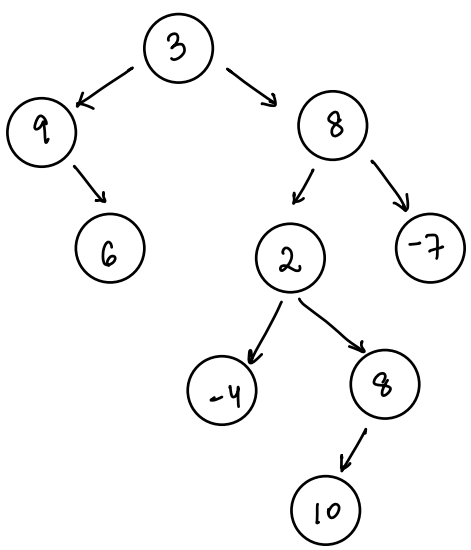
Level order Traversal
- ↳ From left to right
- ↳ From right to left

Some more views
- ↳ Left view
- ↳ Right view
- ↳ Top view
- ↳ Bottom view
- ↳ Diagonal view

* Level order traversal (Traverse from left to right)

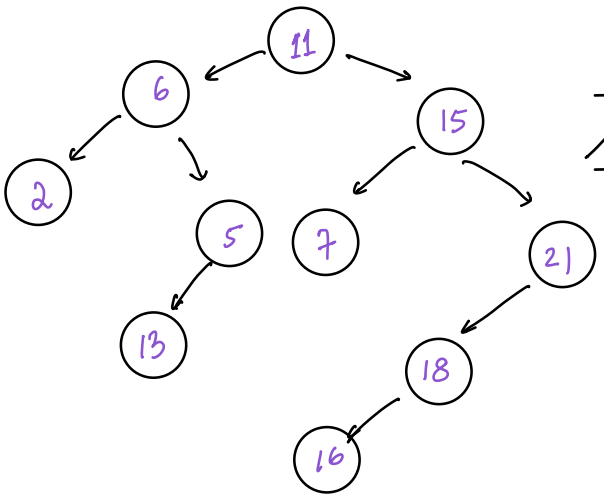
GainSight

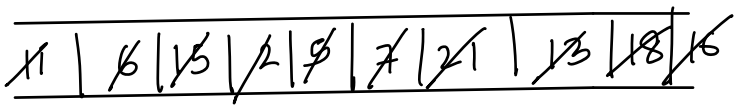


3 9 8 6 2 -7 -4 8 10

o/p → 3 9 8 6 2 -7 -4 8 10

Expected o/p : 3 9 8 6 2 -7 -4 8 10





q → 11 6 15 2 5 7 21 13 18 16

Queue<Node> q;

What should we push in QUEUE?

VAL

Node
[So as to get its children]

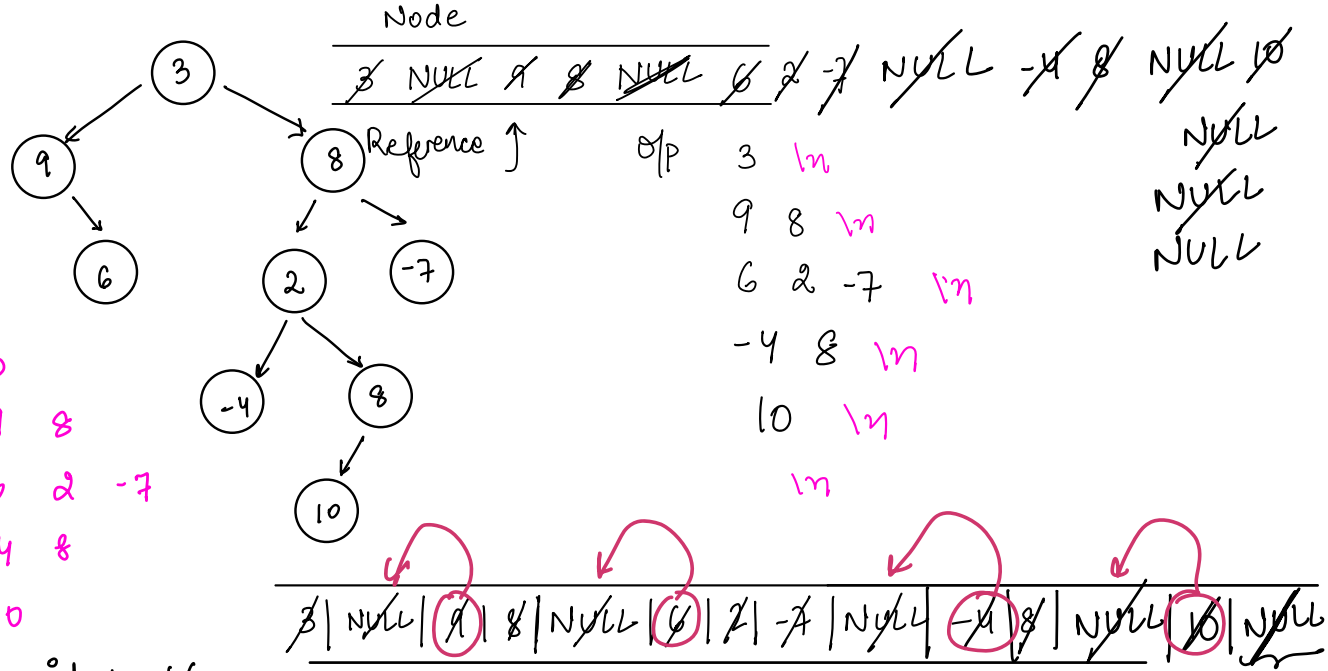
```
void level(root){
    queue<Node> q
    q.push(root)
    while(q.size()>0){
        Node f = q.front()
        q.pop()
        print(f.data)
        if(f.left != NULL){
            q.push(f.left)
        }
        if(f.right != NULL){
            q.push(f.right)
        }
    }
}
```

DFS : Depth First Search
inorder | Preorder | Postorder

BFS : Breadth First Search
Level order

TC : O(N)

SC : O(N)

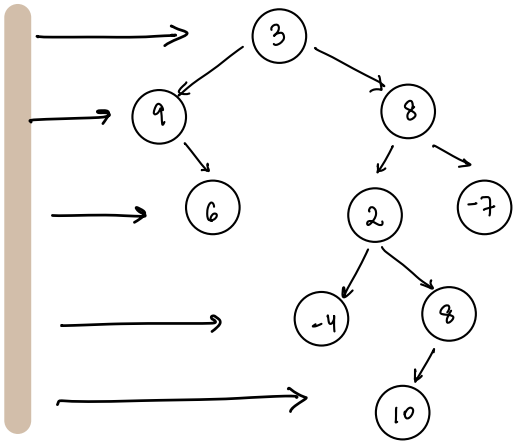



```
void level(root){
    queue<Node> q
    q.push(root)
    q.push(NULL)
    while(q.size()>0){
        Node f = q.front()
        q.pop()
        if(f == NULL){
            print('\n')
            if(q.size()>0) q.push(NULL)
            continue
        }
        print(f.data)
        if(f.left != NULL){
            q.push(f.left)
        }
        if(f.right != NULL){
            q.push(f.right)
        }
    }
}
```

```
3 \n
9 8 \n
6 2 -7 \n
-4 8 \n
10 \n
```

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

* Left View



3	9	6	-4	10

{3, 9, 6, -4, 10}

Observations:

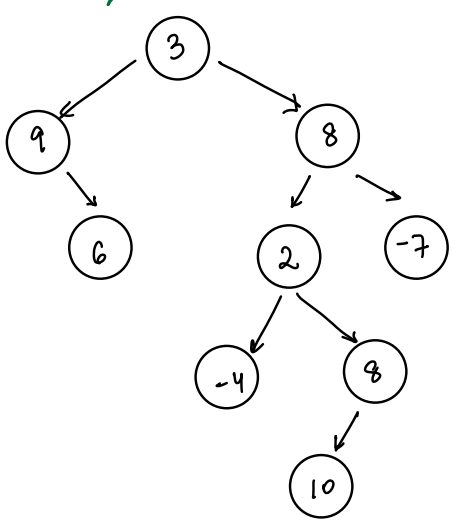
1. We need to print the first ele of each level.

How can we identify the first ele of every level?

2. If before a certain ele, NULL is present; that ele should be included in the answer.

Edge: Root:

3 | NULL | 9 | 8 | NULL | 6 | 2 | -7 | NULL | -4 | 8 | NULL | 10 | NULL

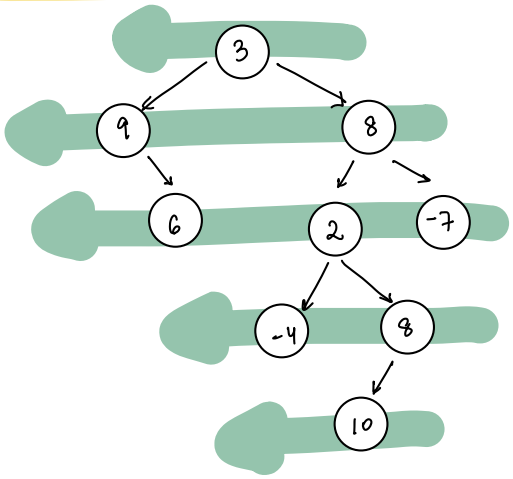


Ans list

{3, 9, 6, -4, 10}

⇓

* level order (Traverse the tree from right to left)



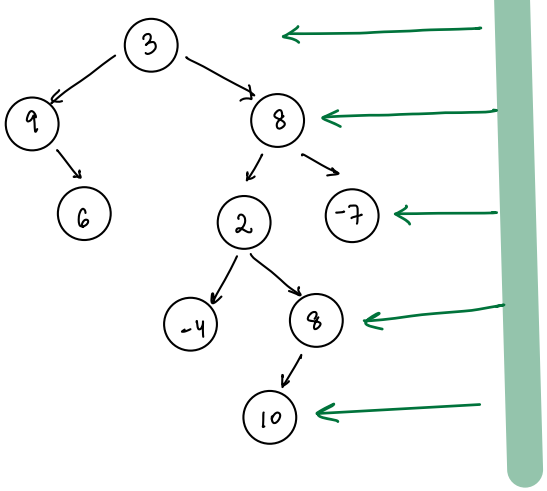
3
8 9
-7 2 6
8 -4
10

3 \n
8 9 \n
-7 2 6 \n
8 -4 \n
10 \n

3 | NULL | 8 | 9 | NULL | -7 | 2 | 6 | NULL | 8 | -4 | NULL | 10 | NULL

---

* Right view

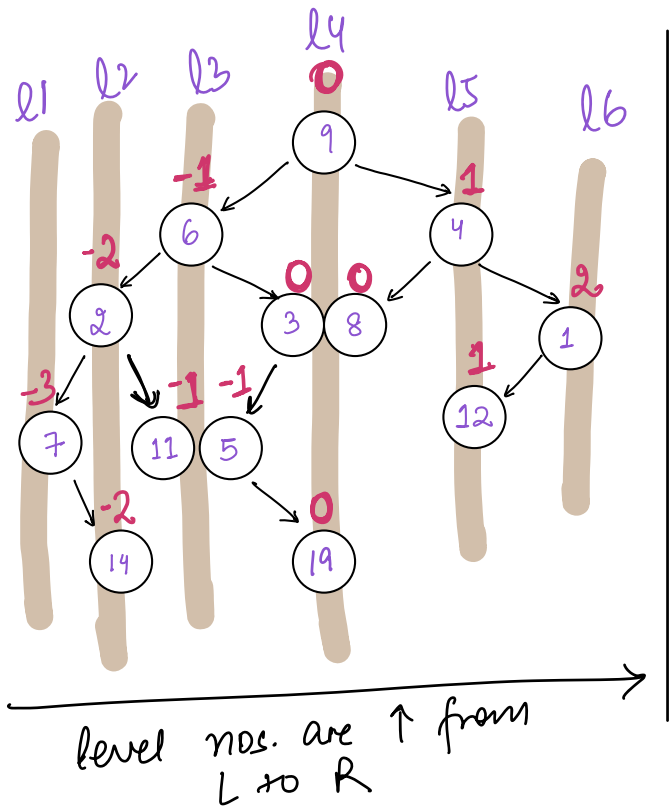


3 8 -7 8 10

First child in level order from right to left

Break → 10:30 PM

* vertical level order view

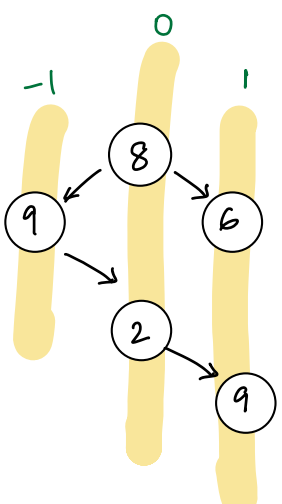


Expected O/P :

7
2 14
6 11 5
9 3 8 19
4 12
1

HASHMAP
<Key, value>
<level, list<Node>>

-3 : 7
-2 : 2 14
-1 : 6 11 5
0 : 9 3 8 19
1 : 4 12
2 : 1



-1 : 9
0 : 8 2
1 : 6 9

Preorder HM ⇒ DLR

-1 : 9
0 : 8 2
1 : 9 6

} no order

inorder ⇒ LDR

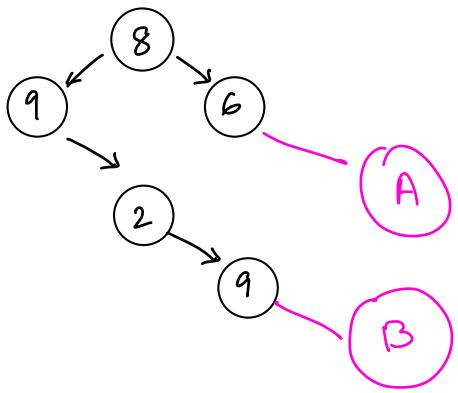
-1 : 9
0 : 2 8
1 : 9 6

} no order

postorder ⇒ LRD

-1 : 9
0 : 2 8
1 : 9 6

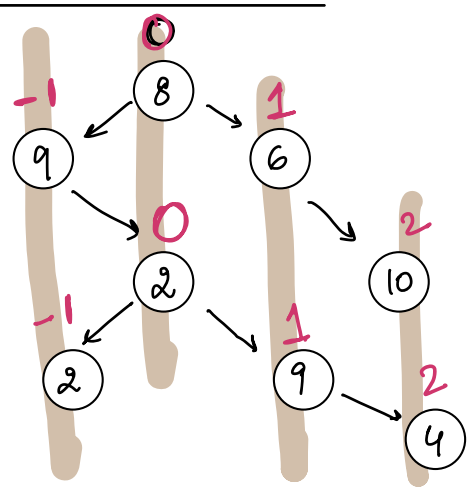
} no order



Not possible with In, Pre & Post order.
They don't care about level order

Hence, we'll go with level order traversal.

Level order Traversal



HM :

{
0 : 8 2
-1 : 9 2
1 : 6 9
2 : 10 4
}

~~<8,0>~~ ~~<9,-1>~~ ~~<6,1>~~ ~~<2,0>~~ ~~<10,2>~~ ~~<2,-1>~~ ~~<9,1>~~ ~~<4,2>~~

pair < d[1], d[2] > P ;

can be two different Data Types.

pair < Node , int >
ref level

---

queue< pair <Node, int>> q
map < int , list <Node> > mp
q. insert ({ 8, 0 })
minLevel = +∞
maxLevel = −∞
while (q.size() > 0) {

```
pair<Node, int > p = q.front()
q.pop()
Node t = p.first
int level = p.second
minLevel = min( minLevel, level )
maxLevel = max(maxLevel, level )
mp[level].add(t)
if (t.left != NULL){
        q.insert ({t.left , level - 1})
}
if(t.right != NULL){
        q.insert ({t.right, level + 1})
}
}
```

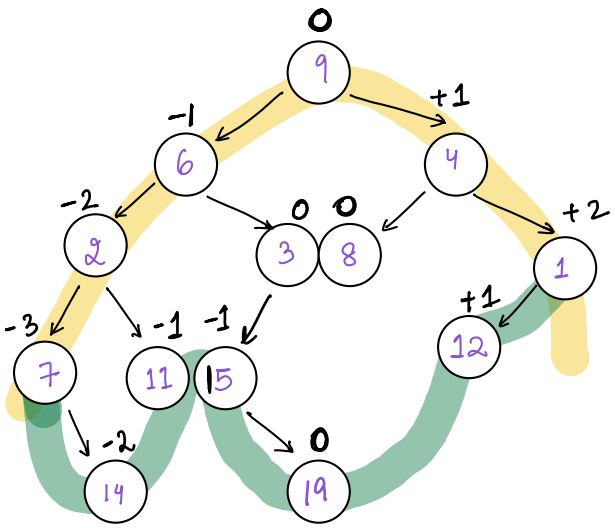
```
for (i = minLevel; i <= maxLevel; i++) {
    // mp[i]
    list<Node> temp = mp[i]
    for (j = 0 ; j < temp.size(); j++) {
        print (temp[j])
    }
    print ('\n')
}
```

TC: O(N)

SC: O(N)

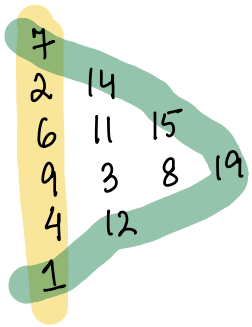
* TOP VIEW

7 2 6 9 4 1



* Bottom View

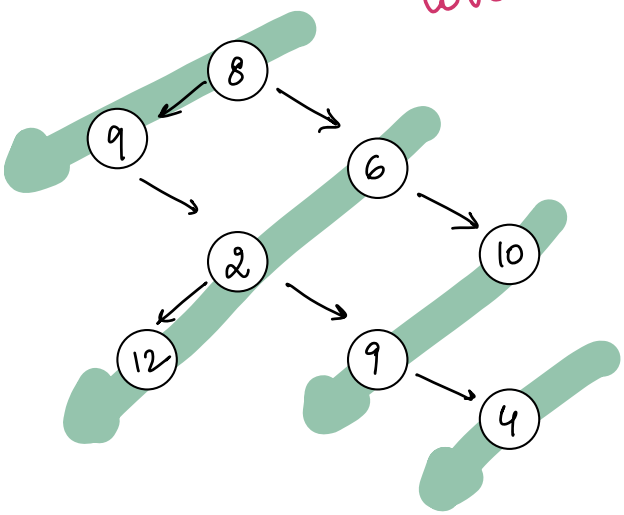
7 14 11 or 15 19 12 1



DIY / TODO

Diagonal View

TODO
⇓
Love it ♡.



Expected o/p

8 9
6 2 12
10 9
4