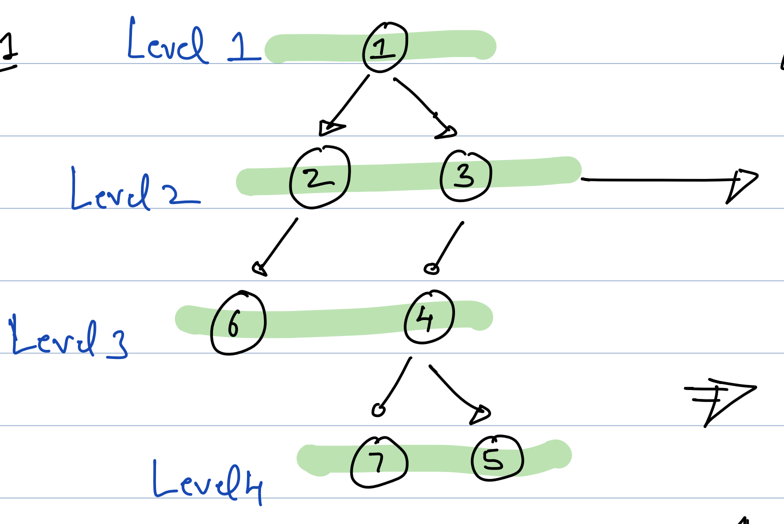
Level Order Traversal [At the same level, traverse L-R]

Ex 1

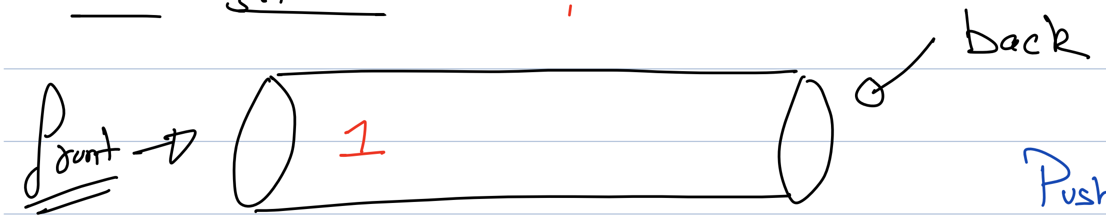


[[1],
[2 3],
[6, 4],
[7, 5]]

⇒ 1, 2, 3, 6, 4, 7, 5

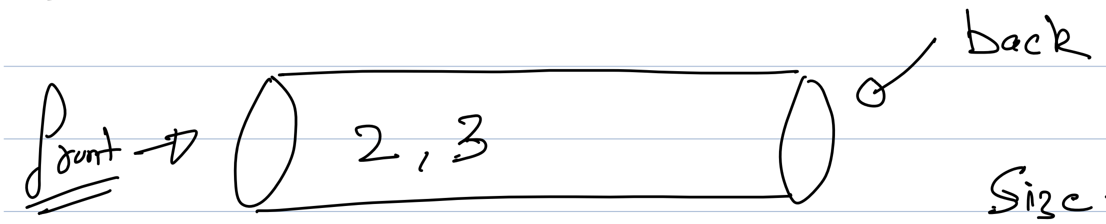
1,

Initialization



Push root

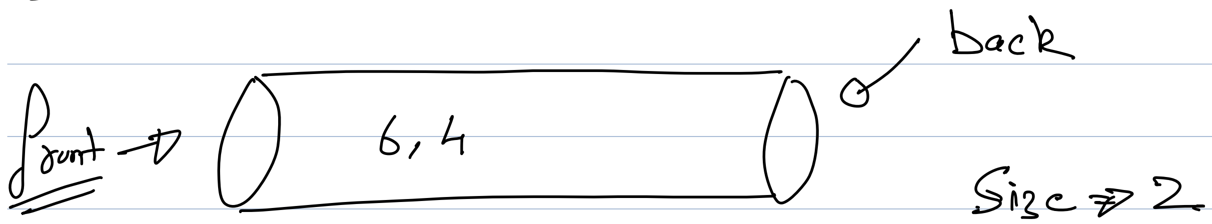
Level 1



Size ⇒ 1

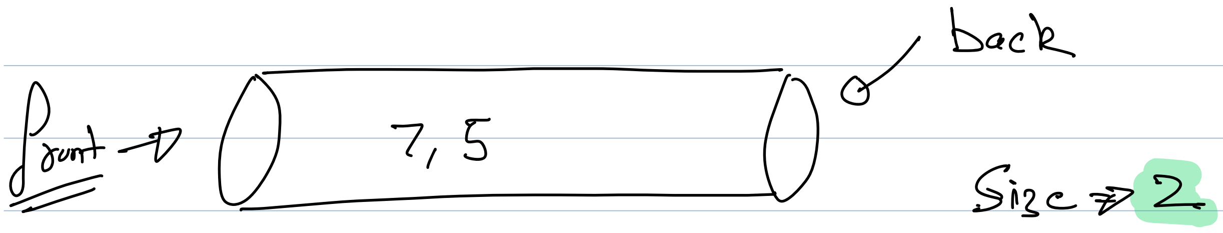
[1]

Level 2



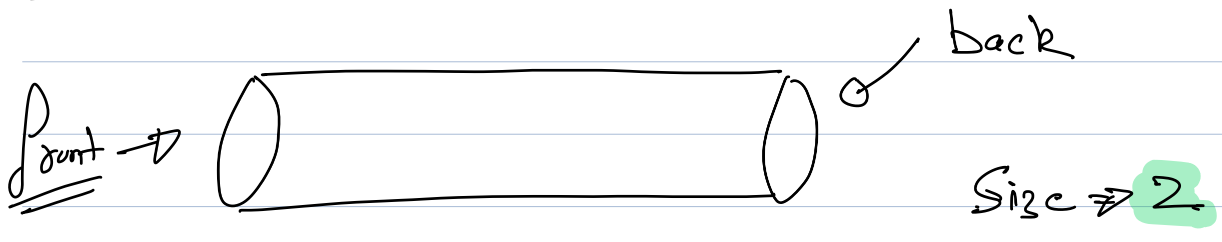
[2, 3]

Level 3



[6, 4]

Level 4



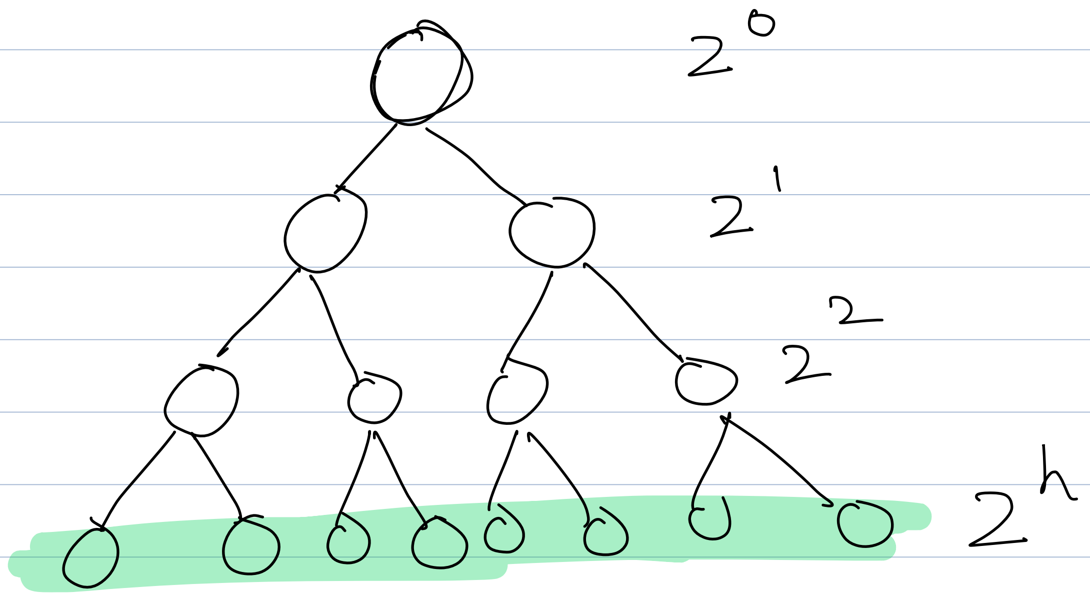
[7, 5]

Pseudo Code.

```
list <list <int>> ans;                 -> if (root == null)
Queue <Node> q;                               return ans;
q.push (root);

while (!q.empty ()) {

    int size => q.size();
    list<int> level_ans;

    for (int i=0 ; i< size ; i++) {
        Node n => q.front();
        q.pop();
        level_ans (n.val);
        if (n.left != null)
            q.push (n.left);
        if (n.right != null)
            q.push (n.right);

    }

    ans.add (level_ans);                       return ans;

}
```

TC: $O(n)$

SC: $O(n)$



No of leaf nodes $= 2^h$

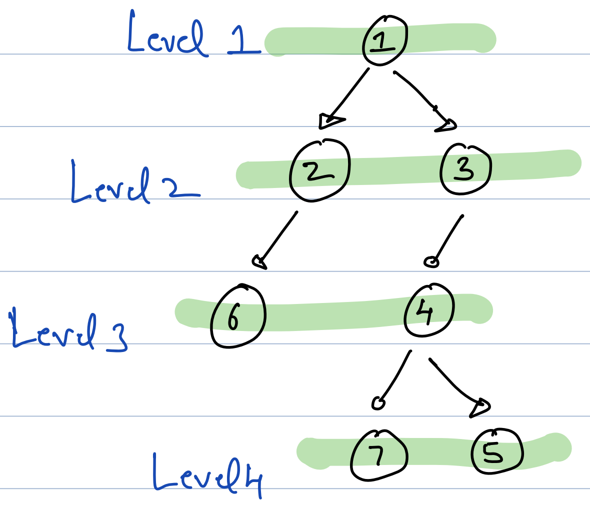
$h \Rightarrow \log n$.

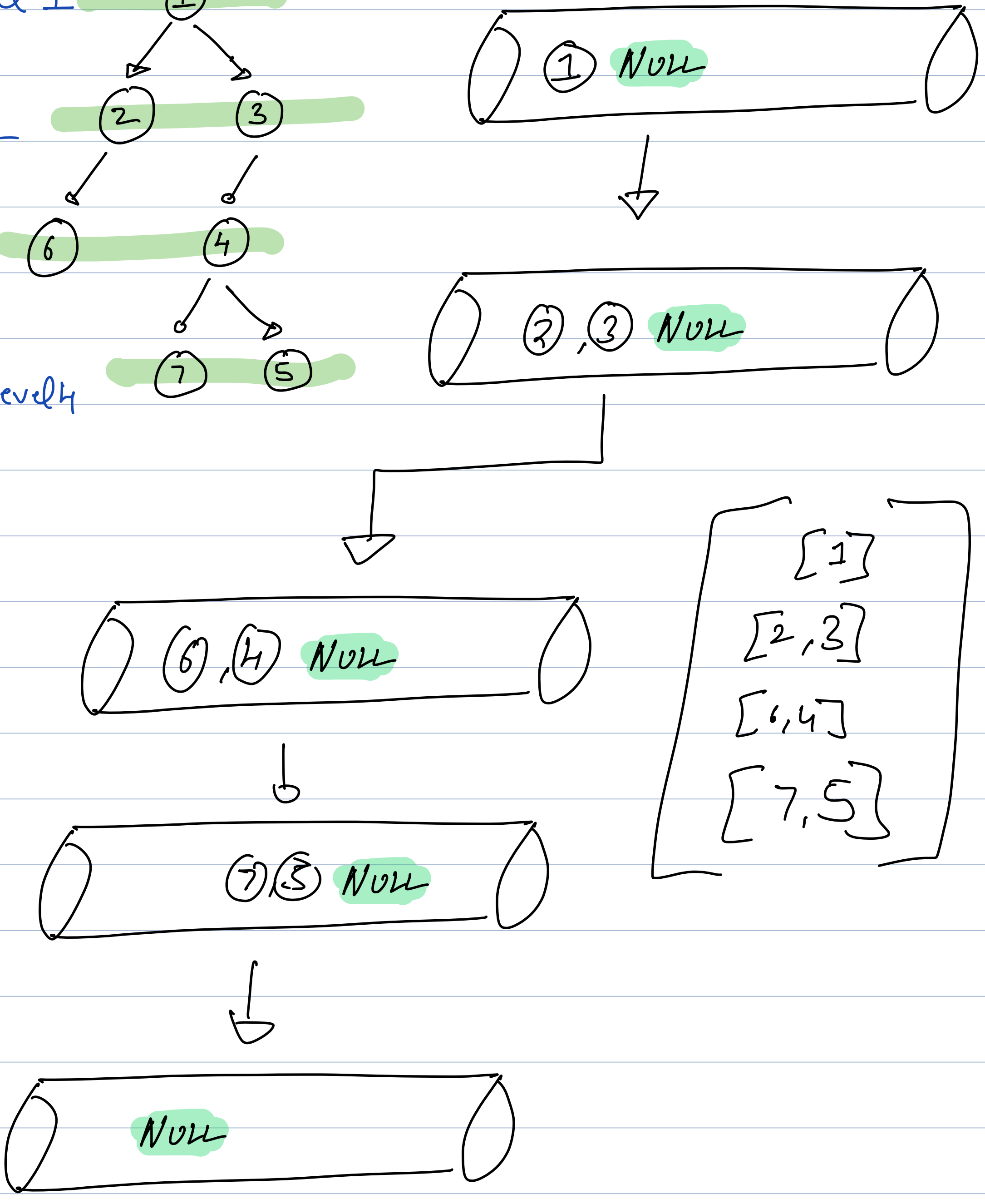
$\Rightarrow 2^{\log_2 n}$

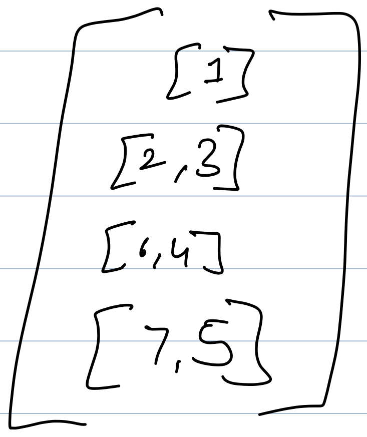
$\boxed{h \Rightarrow \frac{\log n}{2}}$

$\Rightarrow (n)$

$2^{\log n - 1} \Rightarrow \frac{2^{\log n}}{2} \Rightarrow n/2$

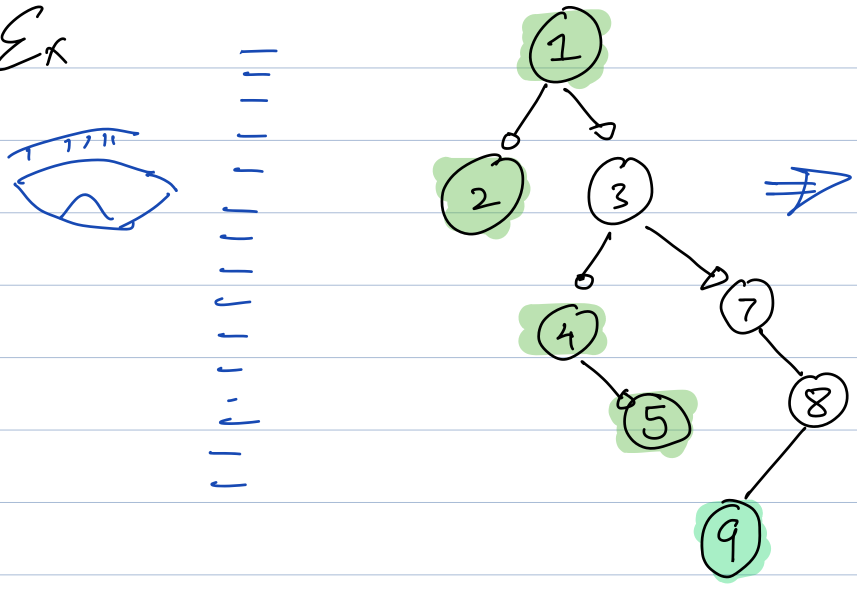

Level 1
1
Level 2
2
3
Level 3
6
4
Level4
7
5


1 NULL
2, 3 NULL
6, 4 NULL
7, 5 NULL
NULL


[1]
[2, 3]
[6, 4]
[7, 5]

Q2 Print left view of a BT

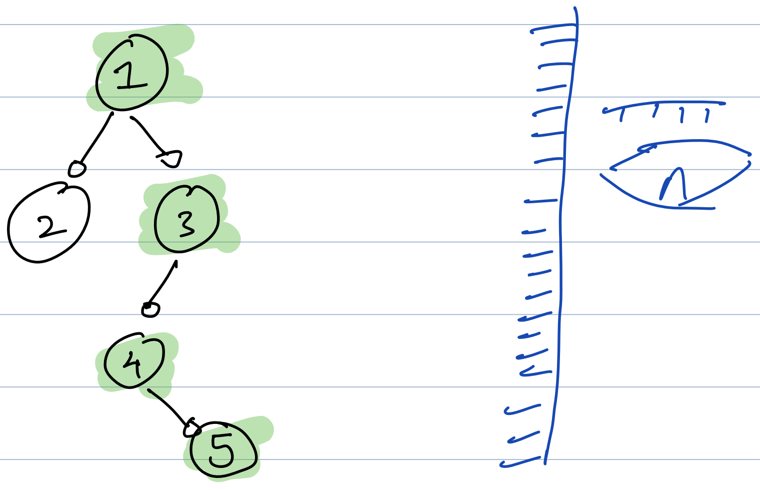
Ex

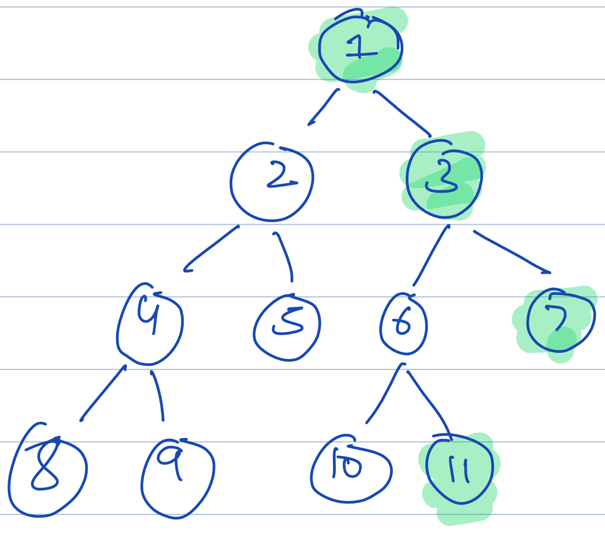


$\Rightarrow$ [1, 2, 4, 5, 9]

Q2 Print right view of a BT

Ex

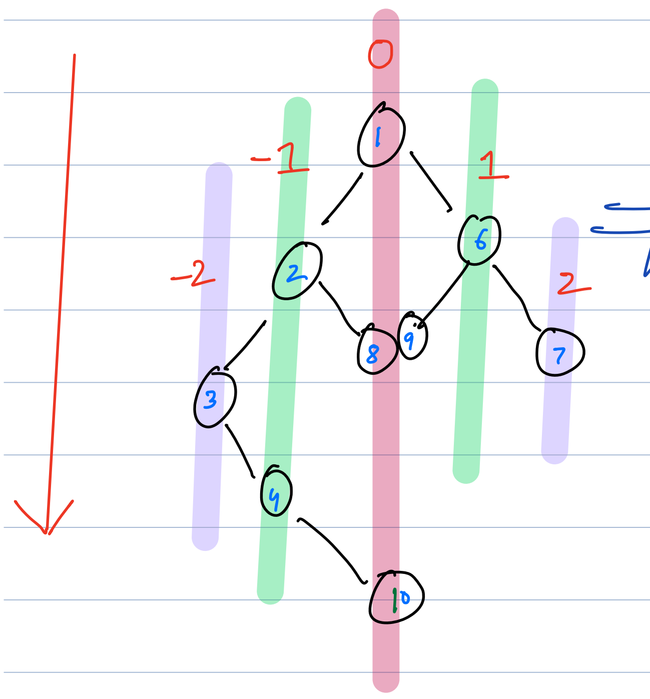




$\Rightarrow$ [1, 3, 7, 11]

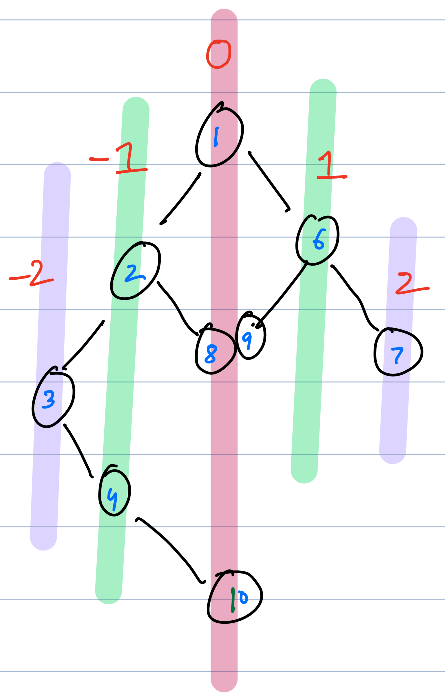
Level Order traversal (Right to left)

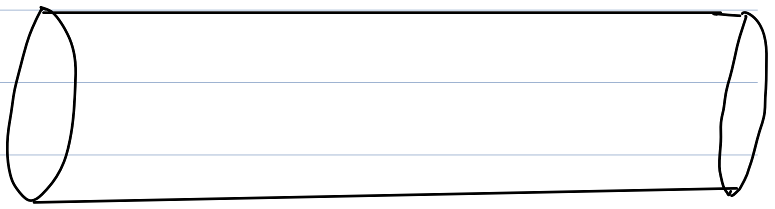
Vertical Order traversal.



$\Rightarrow$

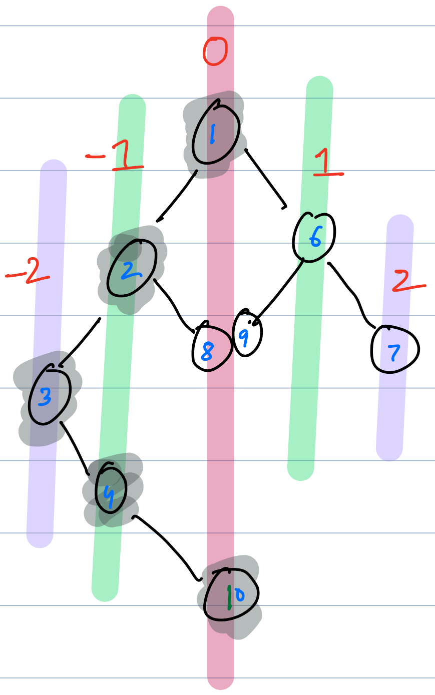
[
[3]
[2 4]
[1, 8, 9, 10]
[6] , [7]
]





Preorder

[0] → [1, 8, 9, 10]
[-1] → [2, 4]
[1] → [6]
[-2] → [3]
[2] → [7]

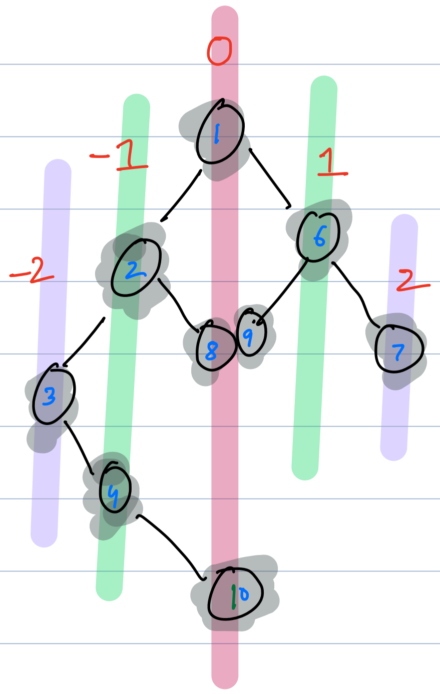


~~0 → 1, 10~~

-1 → 2, 4

-2 → 3

Preorder doesn't work !!



Postorder

→ $\boxed{1, 9, 8, 10}$

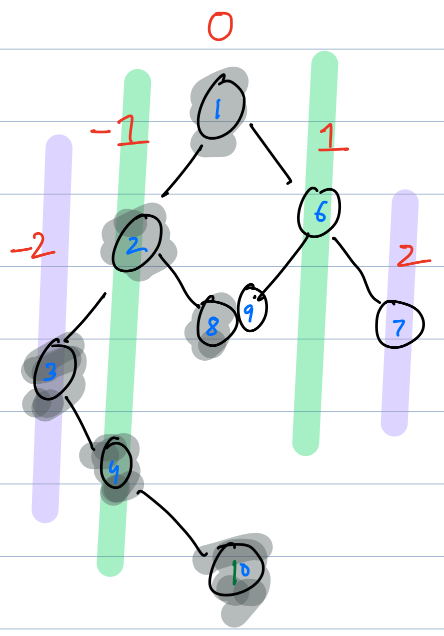
0 → 10, 8, 9, 1

-1 → 4, 2

-2 → 3

2 → 7

1 → 6



⇒ Left Root Right

-2 ⇒ 3

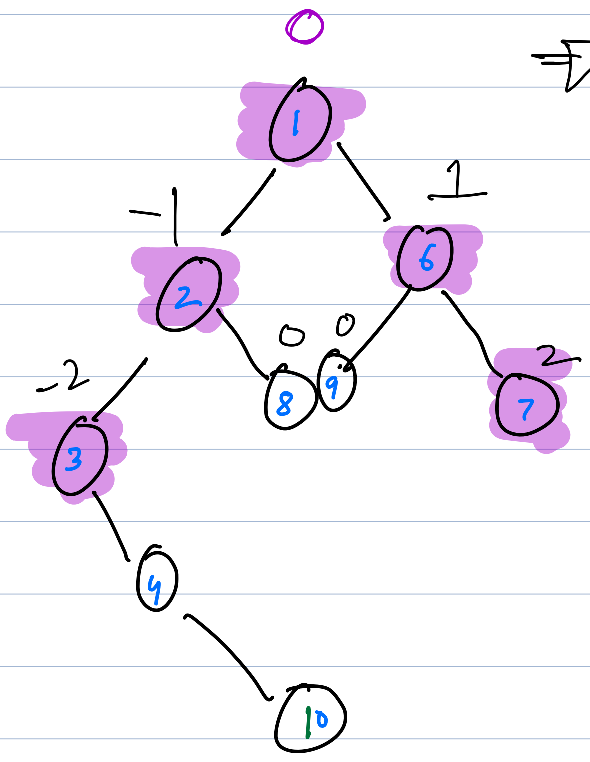
-1 ⇒ 4, 2

0 ⇒ 1, 10, 8 — 1

Pseudo Code

```
list <list <int>> ans ;
map < int, list<int>> mp;          => n
Queue < pair <Node, int >> q;      => n

q.push ( root, 0) ;
int  max_level   =>  Integer.min ;
int  min_level   =>  Integer.max ;

while ( !Q.empty()) {

   Pair p  =>  Q.front();
   int level  =>  p.second;
   Node  n  =>  p.first ;
   q.pop();

   if (mp.containsKey (level))
          mp.get (level).add (n.val) ;
   else
        mp. insert ( level, new list);
        mp.get (level).add (n.val);
if ( n.left != null) {
    q.push ( make_pair (n.left, level -1));

}

if (n.right != null) {
    q.push ( make_pair (n.right , level +1));

}

min_level => min (min_level, level);
max_level => max (max_level , level);

}
for (int i = min_level ; i <= max_level ; i++) {

        ans.add (mp.get(i));
}

                                    TC: O(n)
   return ans;
                                    SC: O(n)
```

Q4 Print the top view of BT



$\Rightarrow$ [3, 2, 1, 6, 7]