TREES 1

Today's content

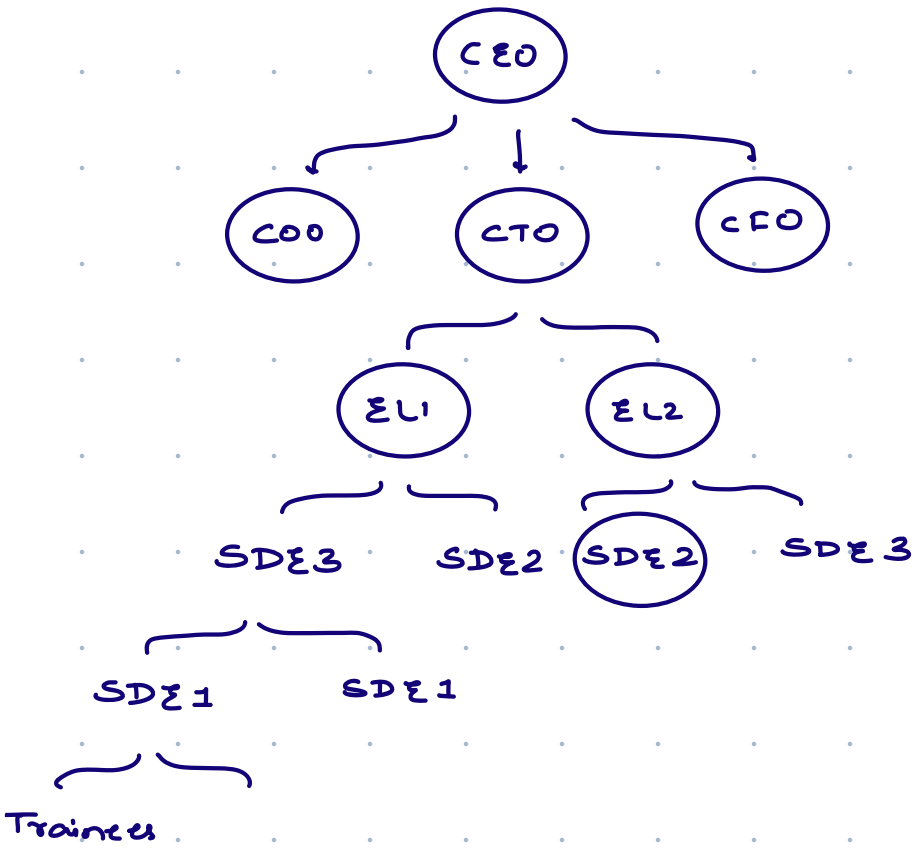
- → Trees Intro
- → Naming convention
- → Trees traversal
- → Iterative inorder
- → Level Order Traversal
- → left view & right view

01. Arrays & ArrayList
02. Stack & Queue
03. LinkedList

} Linear data structures

Trees → Non linear data structure

↳ stores information in hierichal order



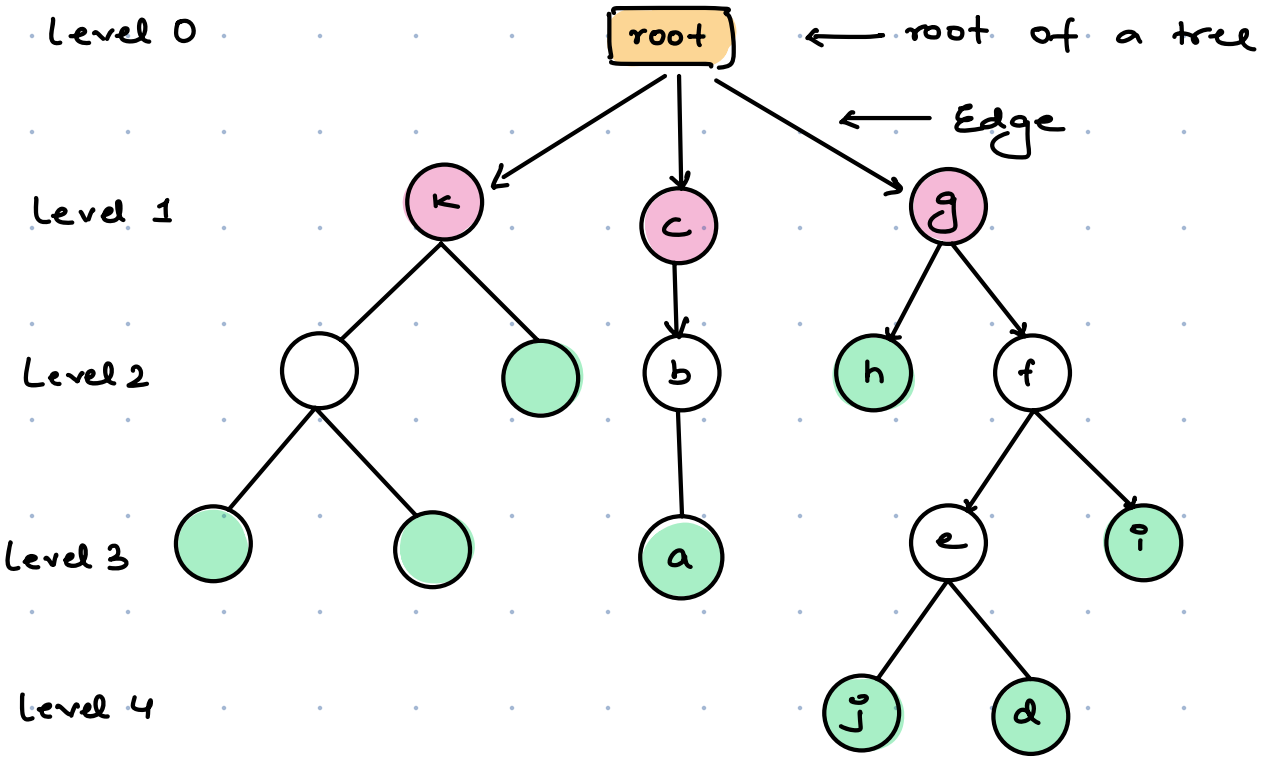
→ Family tree

→ Config tree in windows

→ File structure

→ MLM market

Naming convention for Trees



Node → An element of a tree that contains data & it may have some children

sibling nodes → children of same parent

leaf nodes → Nodes with 0 child

Total no. of Nodes = N

No. of Edges = N - 1

Ancestor → All nodes from parent node to root node are going to be ancestor.

Ancestor (e) = f, g, root

Ancestor (b) = c, root

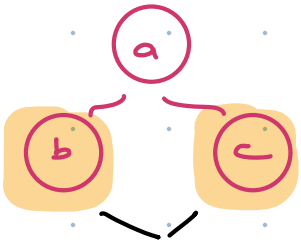
Descendant → All the nodes from child to leaf nodes along the same path will be descendants

Descendants (f) = d, e, i, j

Subtree → A part of a tree

→ Child along with descendants is a subtree.

Quiz 1 → Can leaf node be a subtree?



Leaf node is also a subtree.

Is single node a valid tree? → Yes

Quiz 2 Do all nodes have parent node?

↳ No, root node is an exception.

Height (Node) → Maximum distance (in terms of edges) between the node & its farthest leaf child node.

Height (b) = 1

Height (g) = 3

Depth (Node) → No. of edges b/w the given node to the root node.

Depth (g) = 1

Depth (root) = 0

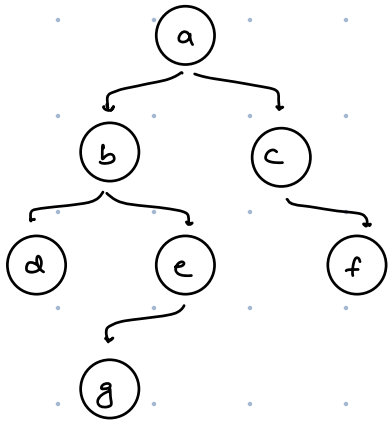
Quiz 3 → Height of leaf Node → 0

Binary Tree

→ Each node can go & have atmost 2 children.

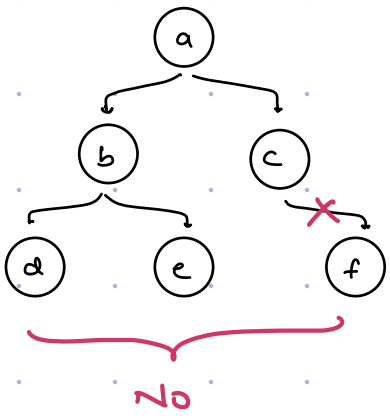
0, 1, 2 ✓

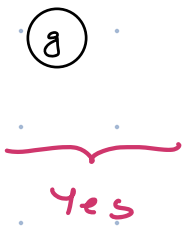
```
class Node {
    int val;
    Node left;
    Node right;
}
```

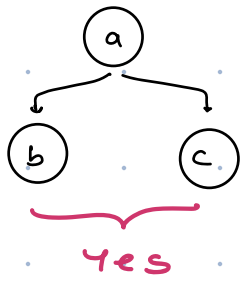


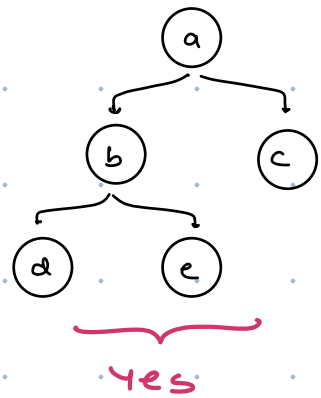
Types of Binary trees

01. **Proper binary tree** → Every node will have either 0 or 2 children

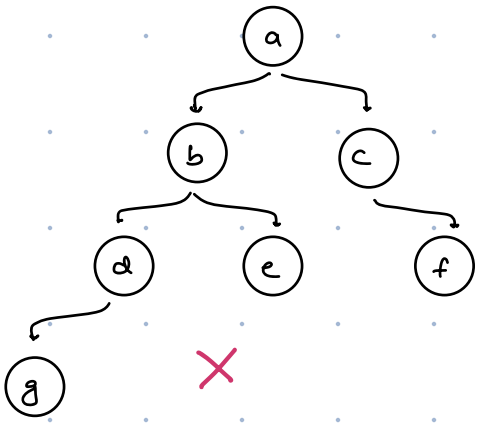


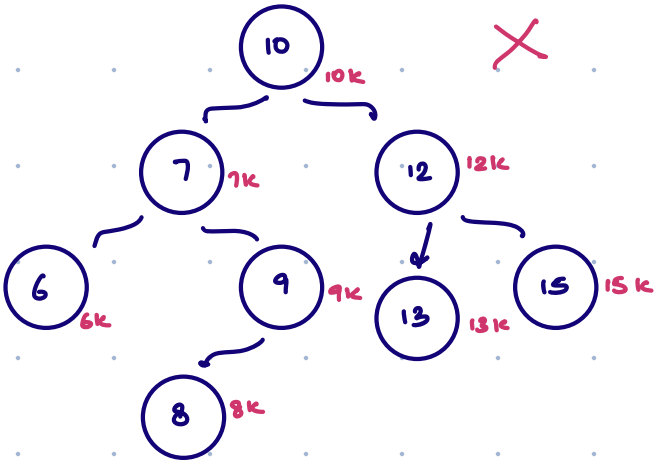


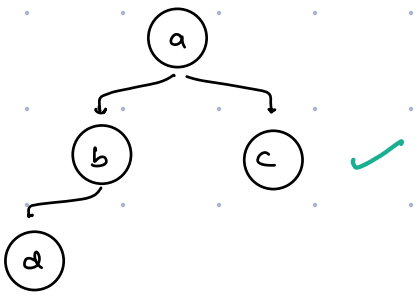




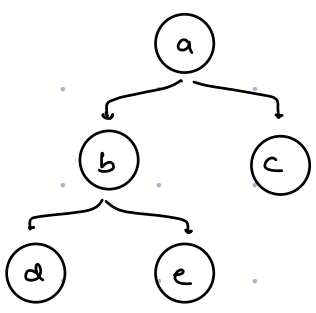
02. **Complete Binary tree** → All the levels must be completely filled. the last level can be an exception but for last level as well, child should be added from left to right.



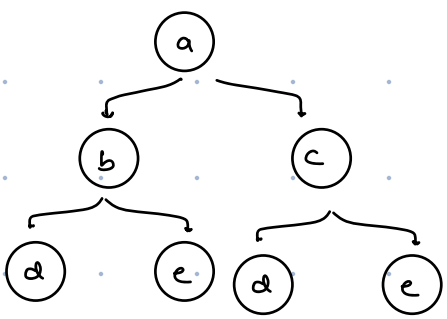




* **Perfect binary tree** → All internal nodes must have 2 children. All leaf nodes should lie on same level.



NO



Yes

→ Perfect binary tree have property of both complete binary tree & proper binary tree

Traversals

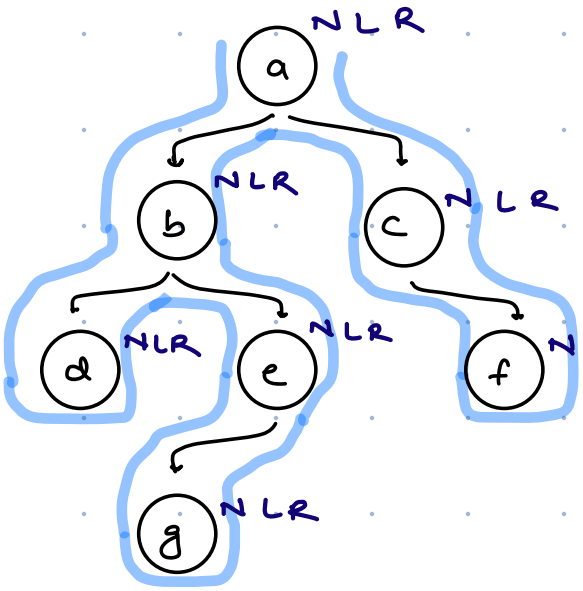
⇒ Preorder (N L R)

⇒ Inorder (L N R)

⇒ Postorder (L R N)

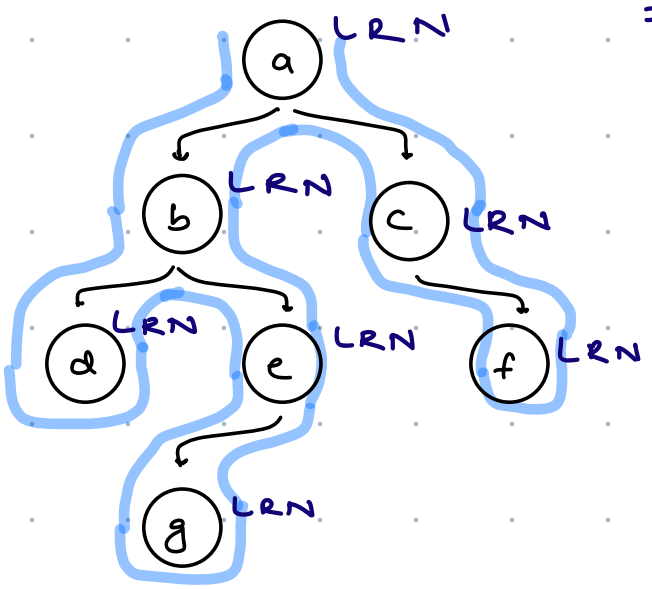
} Depth first Traversal

⇒ Level Order → Breadth first Search



preorder = a b d e g c f

↳ N LR for every node



postorder = d g e b f c a

↳ L R N

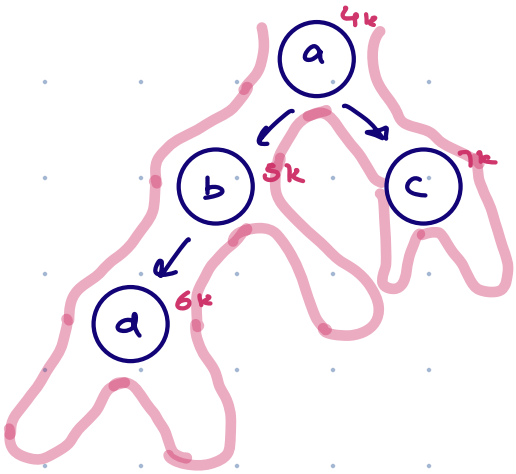
Quiz 5 - Inorder Traversal → 3 2 1 5 4 6

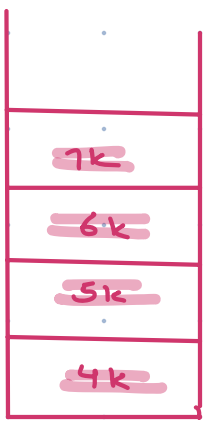

```
void inorder (Node root)
{
    if (root == Null) return;
    inorder (root.left)
    print (root.val);
    inorder (root.right);
}
```

Tc: O(n)

Sc: O(Ht)

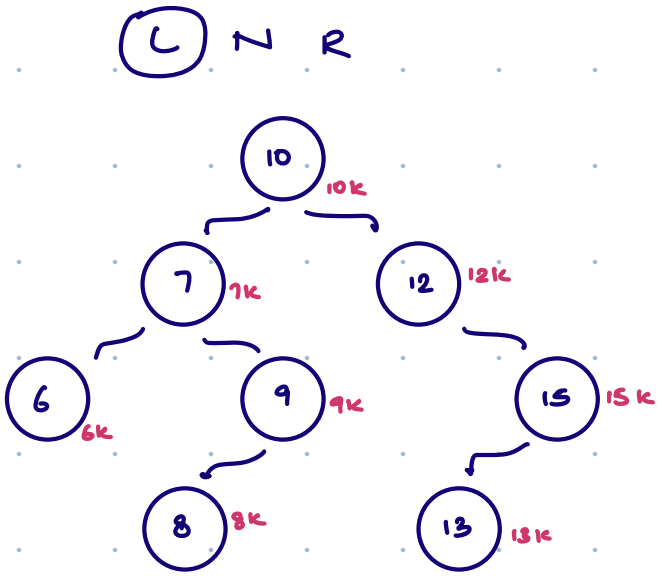
- d b a c



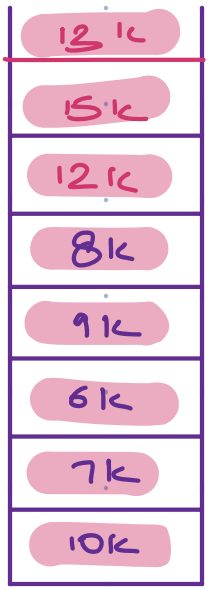


8:18 AM → 8:28 AM

Iterative Inorder Traversal



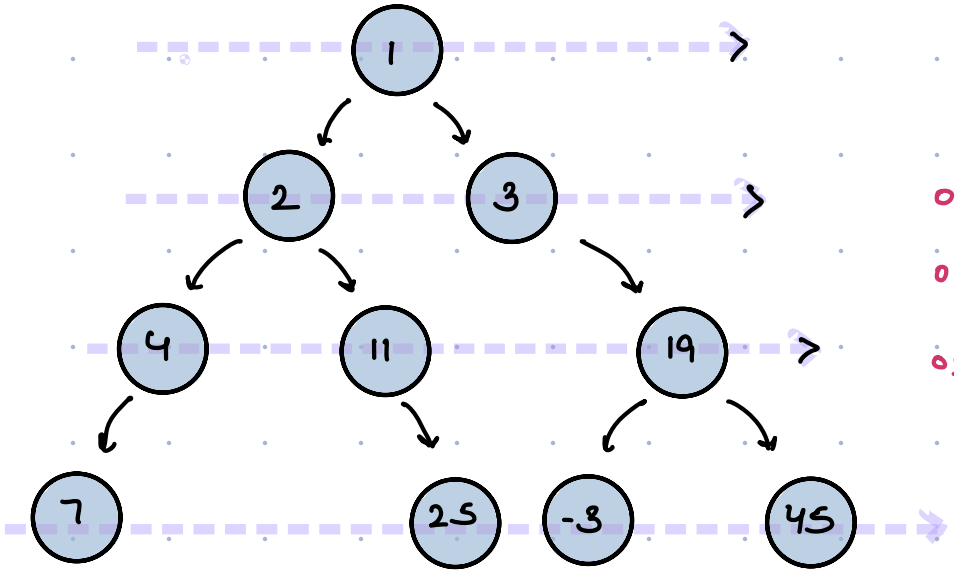
C



6 7 8 9 10 12 13 15

```
Node curr = root
while ( st.size()>0 || curr != Null )
    while (curr != Null)
        st.push(curr);
        curr = curr.left;
    curr = st.pop();
    Print (curr.val);
    curr = curr.right;
```

Level Order Traversal (Breadth first search)



Queue

01. remove Node
02. Process the data
03. Add the children

1 2 3 4 11 19 7 25 -3 45

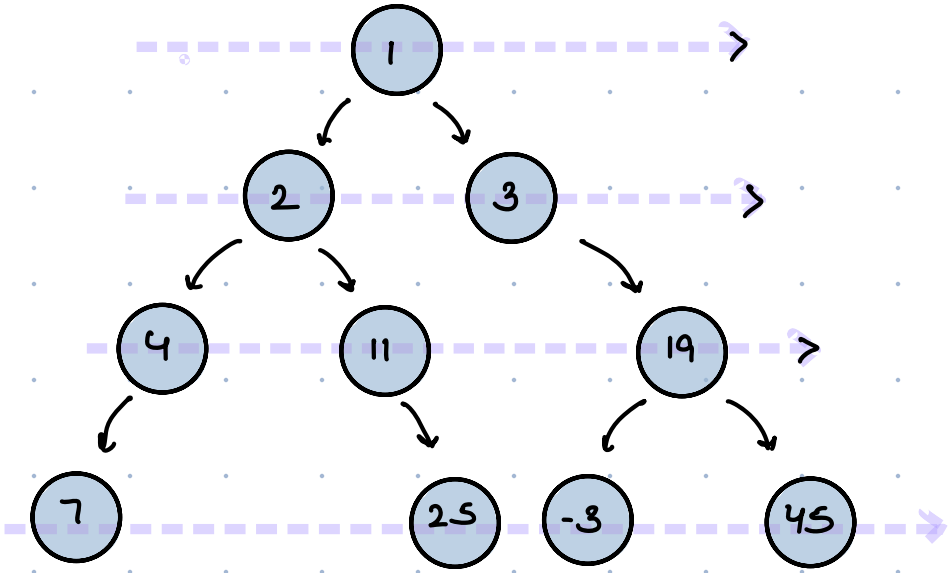
Output → 1 2 3 4 11 19 7 25 -3 45

```
Queue<Node> q;
q.add(root);

while(q.size()>0)
{
    Node rem= q.remove();
    Print(rem.val);
    if(rem.left != Null) q.add(rem.left);
    if(rem.right != Null) q.add(rem.right);
}
```

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

* Level Order



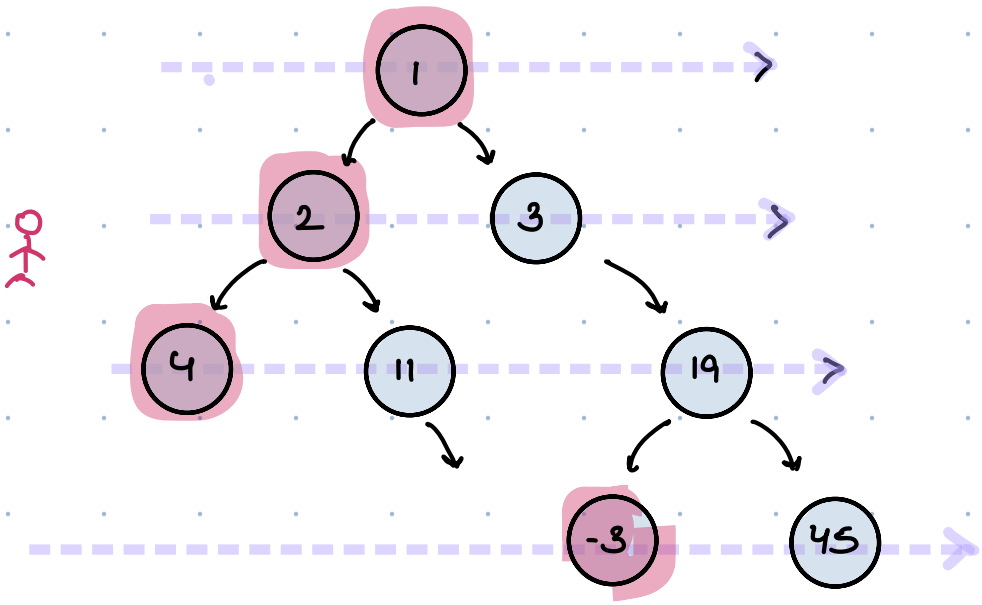
1	2	3	4	11	19	7	25	-3	45

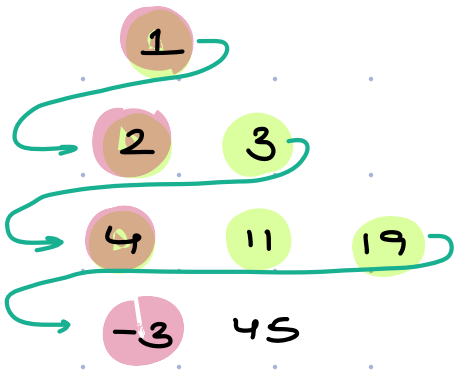
sz = ~~1~~ 0 → sz = ~~2~~ 1 → 0 → sz = ~~3~~ ~~2~~ + 0

```
Queue<Node> q;
q.add(root);
while(q.size()>0)
{
    int sz= q.size();
    for(int i=1; i<=sz; i++){
        Node rem= q.remove();
        Print(rem.val);
        if(rem.left != Null) q.add(rem.left);
        if(rem.right != Null) q.add(rem.right);
    }
    Println();
}
```

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

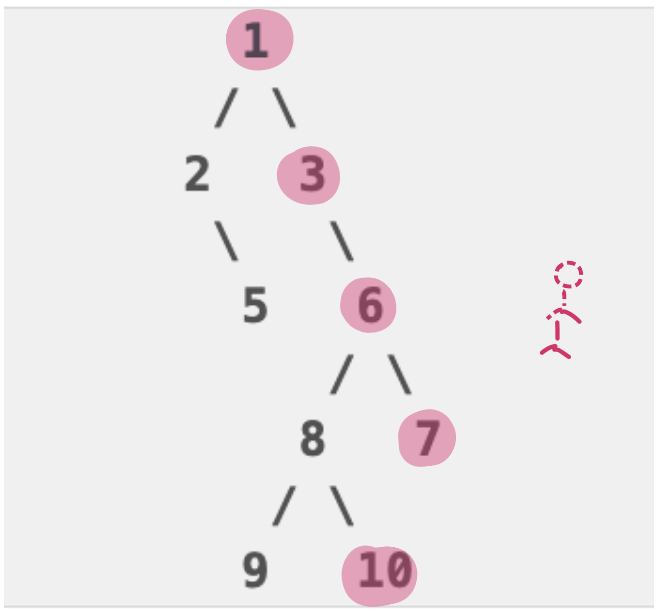
* left view





if (i == 1)
↓
print (rem.val);

right view



if (i == sz)
↓
print (rem.val)