TREES 1

Today's content

- → Trees Intro
- → Naming convention
- → Trees traversal
- → Iterative inorder
- → Construct tree with in[] & post[]

✓ Mock Interview

✓ DSA Full syllabus contest (5 question) → 3 attemps

↓

Mock Interview (2 questions)

Fixed Mock credits

Floaters

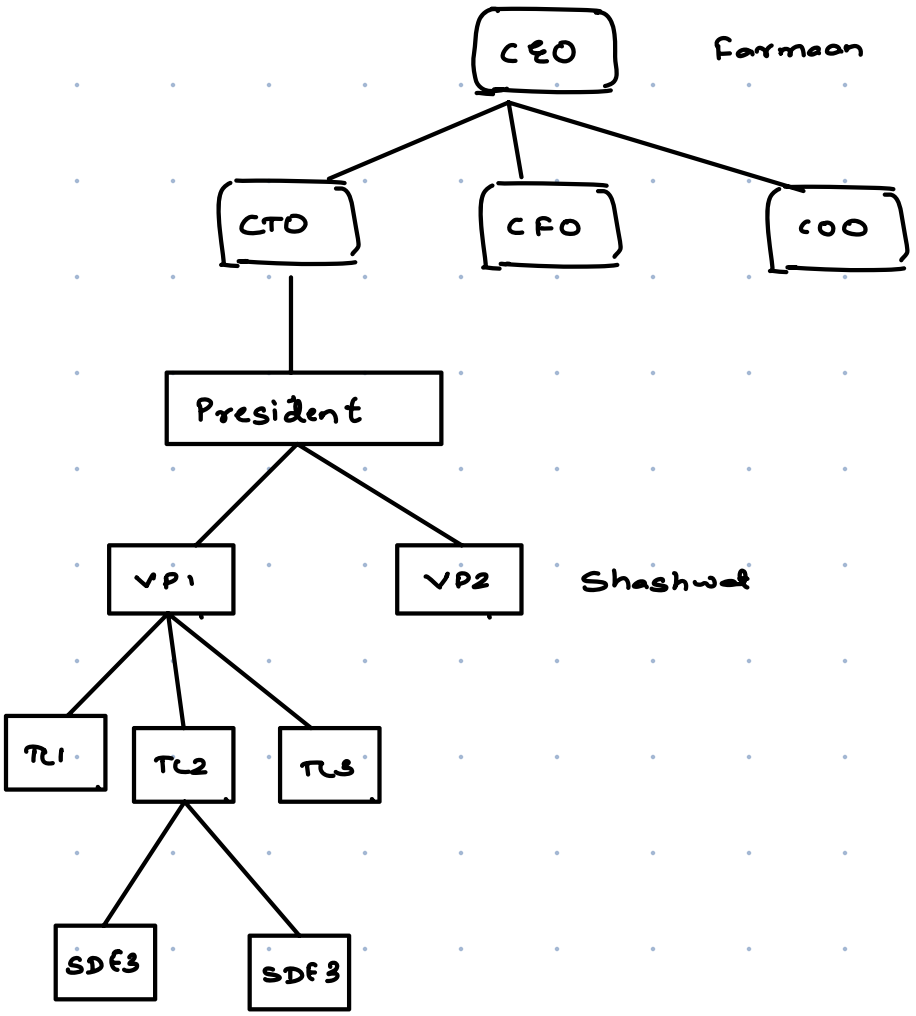
Mock credits without expiry

DSA → clear Full contest / clear Mock Interview → +30 days

Linear Data structure

01. Array
02. Linkedlist
03. Arraylist
04. Stack & Queue

Trees → Data structure which stores hierarchical data.

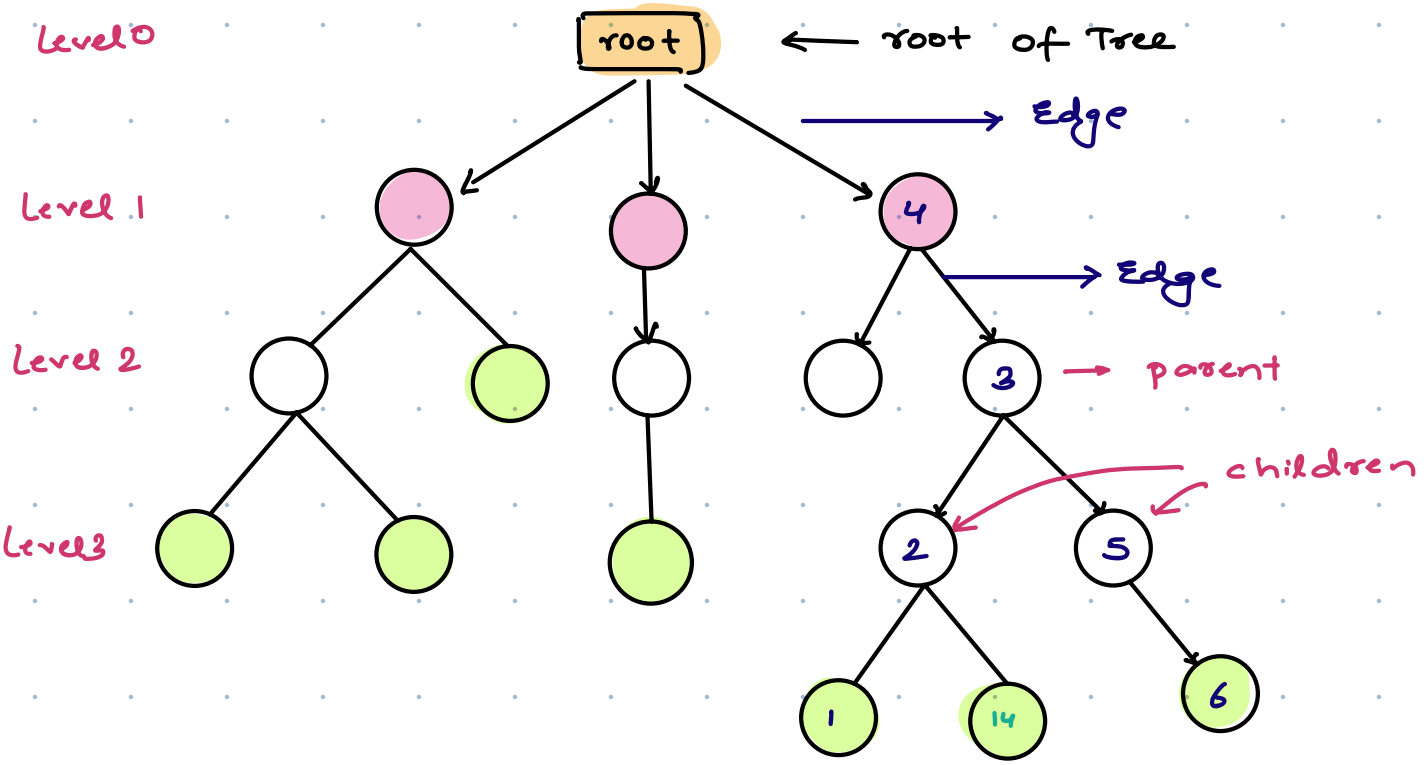


02 Family tree

03. Folder structure

03. Git commits

Naming convention of Trees



→ sibling nodes → share the same parent.

→ leaf nodes → Nodes with 0 child

* Every node will have one parent except the root node.

Total no. of Nodes = N

No. of Edges = N-1

Ancestor → All nodes from parent to root node in upward direction are going to be ancestors

Ancestor(1) = 2, 3, 4, root

Descendants → All nodes from child to leaf nodes along the path are descendants

Descendant(3) = 1, 2, 14, 5, 6

* Height of a node → Maximum distance (in terms of edges) between the node & its descendant leaf nodes

* Height of tree = Height of root node ⇒ 4

* Depth of a Node → No. of edges between the given node & the root node depth(root) = 0

Depth of tree = Depth of farthest leaf node

Height of Tree = Depth of tree

* Subtree ⇒ A portion of a tree
→ Node along with its descendant, which are part of a tree.

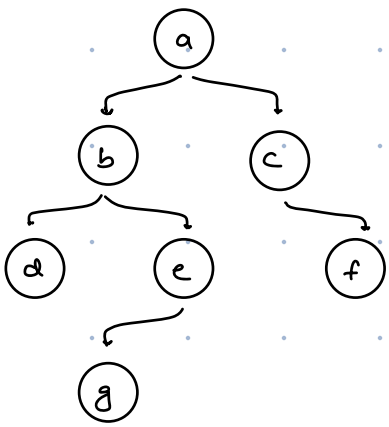
Quiz 1 = Can a leaf node be a subtree? ⇒ Yes

Quiz 2 = Do all nodes have a parent Node? ⇒ No, root node doesn't have parent node.

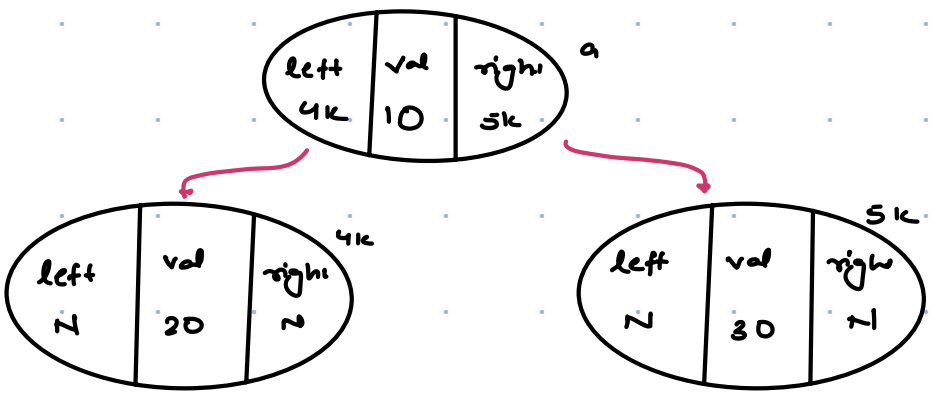
Quiz 3 = Height of leaf node = 0

* Binary Tree → Tree in which node can have atmost 2 children

{0, 1, 2}



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

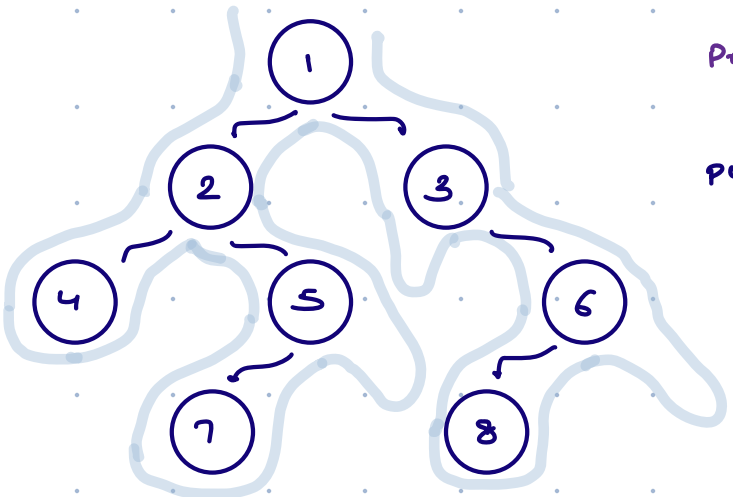


a.left = 4K

a.right = 5K

* Binary Tree Traversal

01. Preorder { Node LST RST }
02 Inorder { LST Node RST }
03 Post order { LST RST Node }



Pre = { 1 2 4 5 7 3 6 8 }

post = { 4 7 5 2 8 6 3 1 }

10:28 pm → 10:33 pm

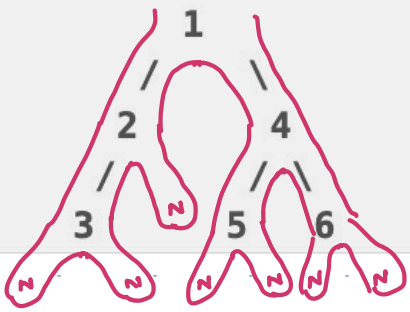
Preorder Traversal

{ N L R } - For every node

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

TC : O(n)
SC : O(ht of tree)

What is the inorder traversal sequence of the below tree?



3 2 1 5 4 6
LST Root RST

Inorder → { L N R }

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

TC : O(n)
SC : O(ht of tree)

Scenerio

In an e-commerce platform like **Flipkart**, when orders are placed, they need to be processed efficiently based on their time of placing the order. **Flipkart's** approach is to assign **OrderIDs** in increasing order for orders placed based on their times.

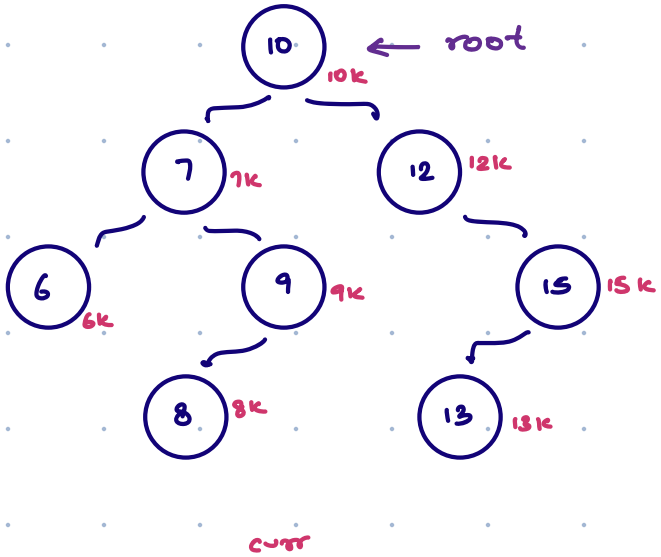
Problem Statement

You are given a Binary Tree, each order can be represented as a node in a binary tree, and having a value denoting the **OrderID** of the order.

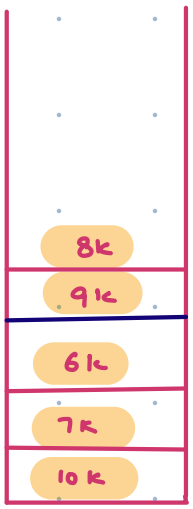
- The **left child** represents the order placed before it and,
- The **right child** represents the order placed after it.

Q Need to tell the order in which all these booking will get processed

{ Inorder Traversal }



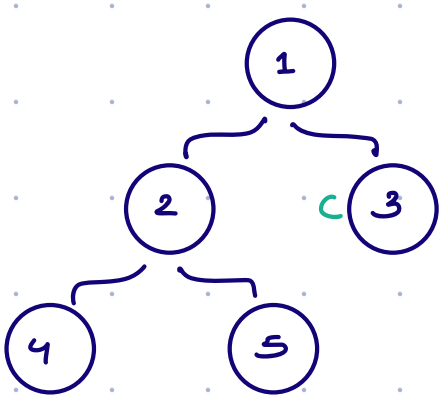
L N R

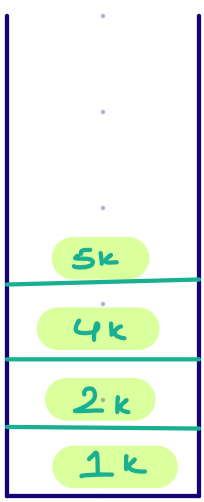



```
while (st.size() > 0 || curr != NULL) {
    while (curr != null)
    |   st.push(curr);
    |   curr = curr.left
    }
    curr = st.pop();
    print (curr.val);
    curr = curr.right;
}
```

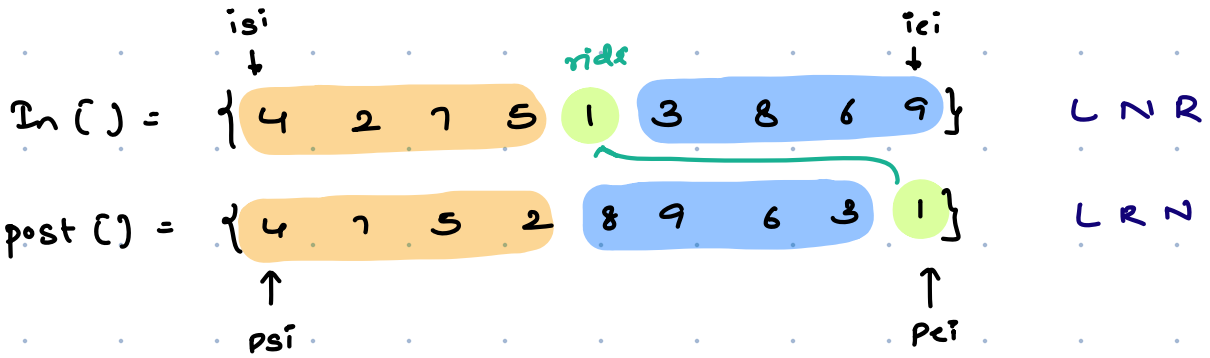
} LST as far as you can

4 2 5 1





Q Construct the binary tree using inorder() & postorder()



* Find the root node = post[pei]
* Find this root node in inorder array = ridx
  - → Get the count of nodes in LST
  - → Get the count of nodes in RST