Today's agenda

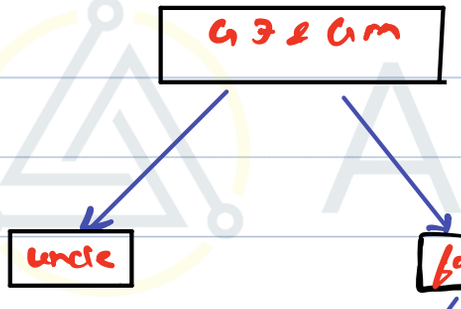
- Trees Intro
- Naming convention
- Tree traversal
- Basic tree problems

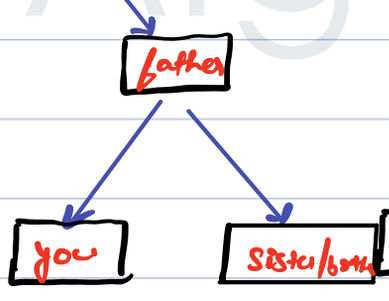
Linear

↳ Arrays, LL, HM, stacks queues etc.

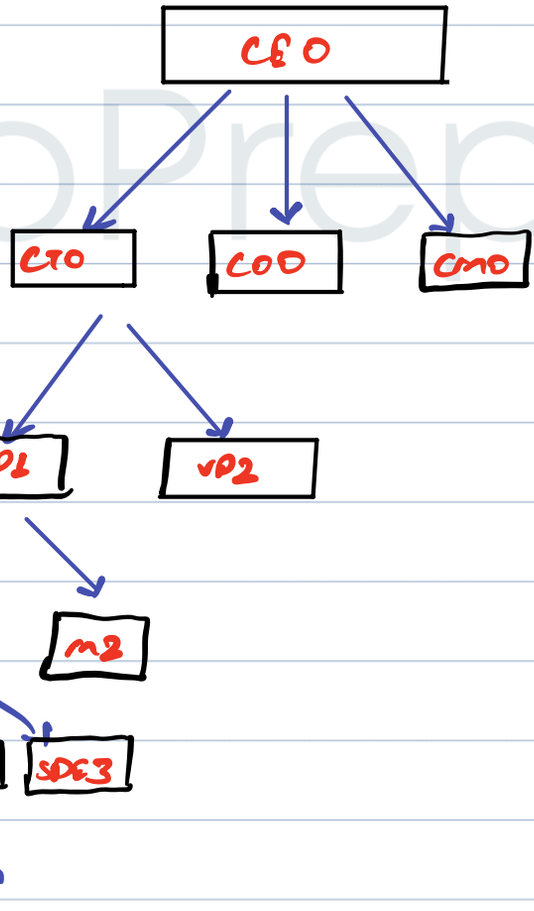
Hierarchical

family tree





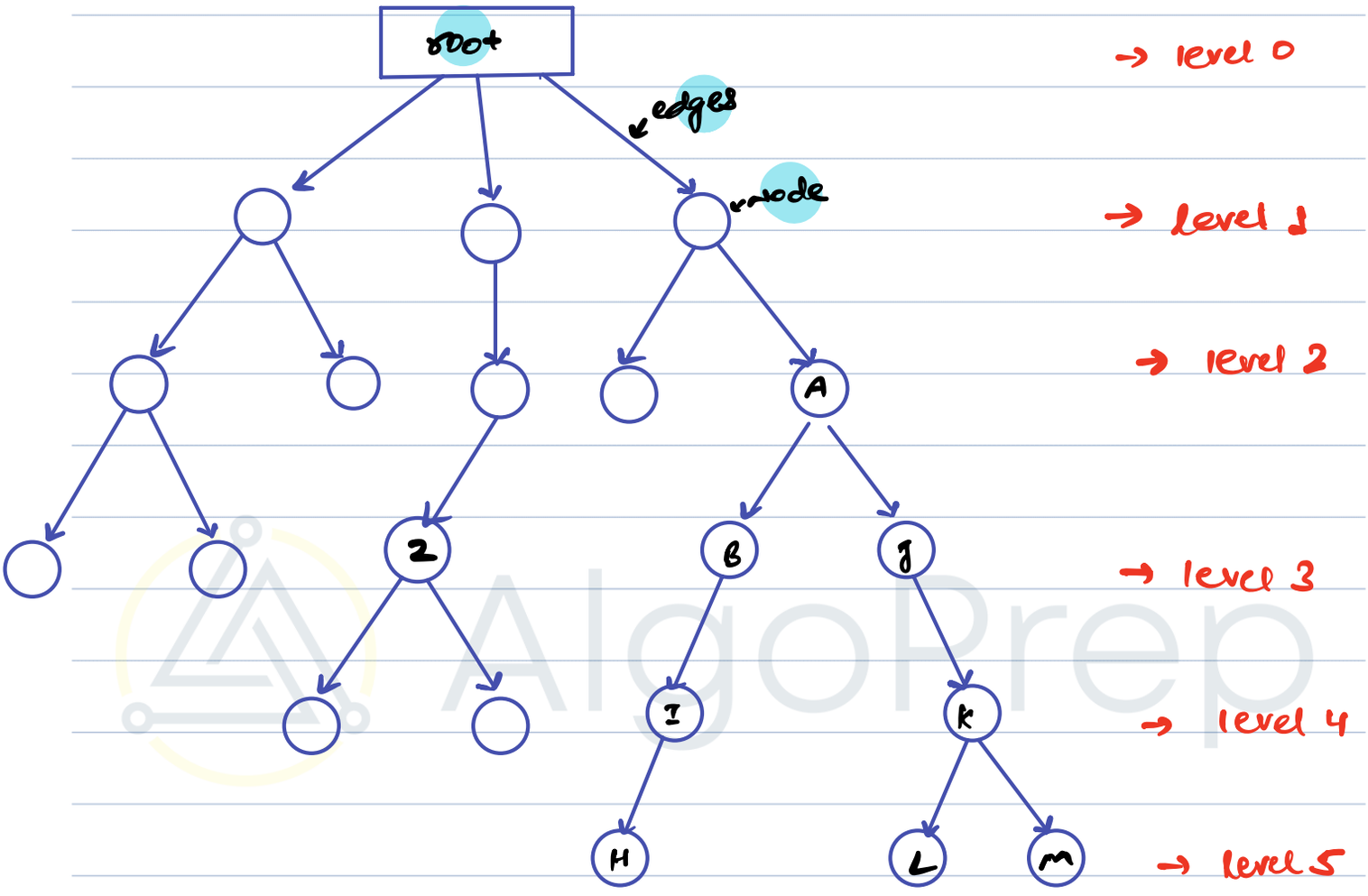
org structure



File System

// Tree

↳ Naming Convention



Parent ↔ Child: Parent is immediate above & child is immediate below.

ex: J is Parent of K, L and M are child of K.

Ancestors: All the nodes in Path from given node to root node.

Descendants: All the nodes below given node.

leaf nodes: nodes having 0 child nodes.

* **Property of tree:**

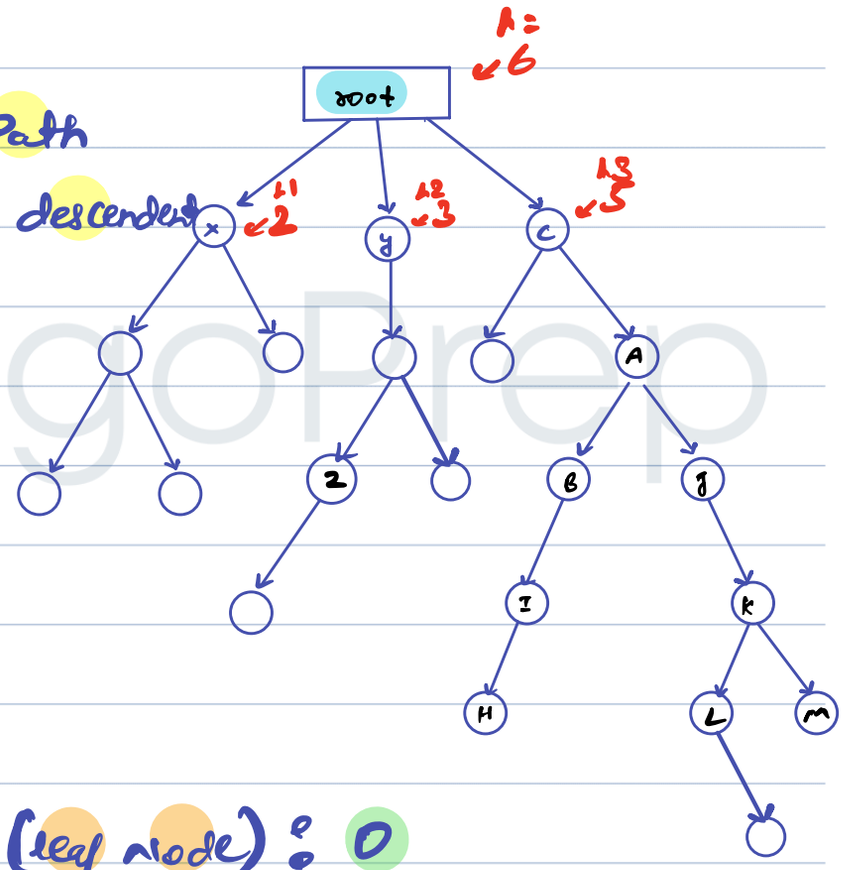
(i) We can have exactly 1 root node.

(ii) For any node, it can have exactly 1 parent.

(iii) It can't have cycle.

* **Height (node)**

Is length of longest path from node to any of its descendent leaf nodes.



Height (B): 2

Height (A): 4

Obs 1:

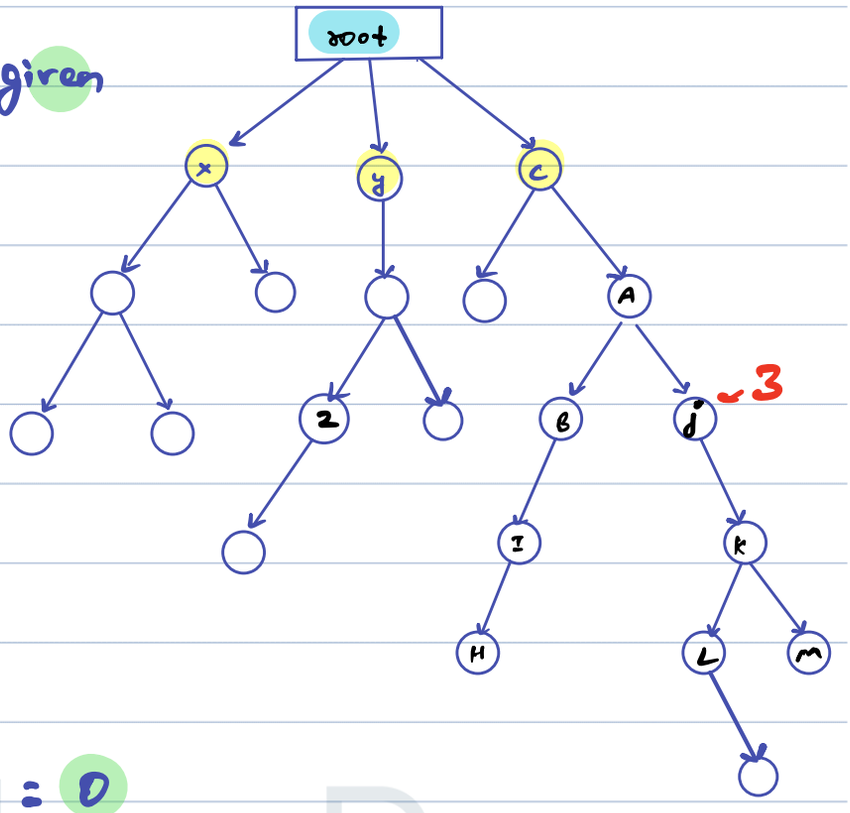
Height (leaf node): 0

Obs 2:

H(node): max(Height of child nodes) + 1;

* Depth (node)

↳ length of Path from given node to root Node.



Depth (j) : 3

Depth (H) : 5

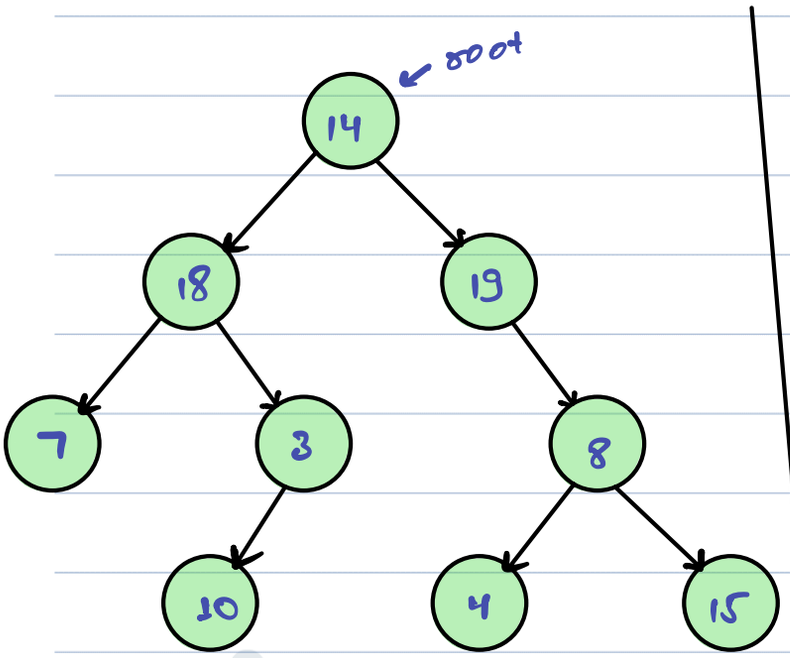
Obs1:

Depth (root node) : 0

Obs2:

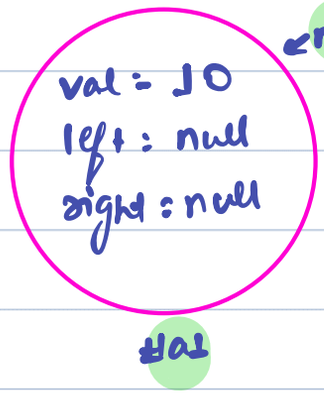
Depth (node) = Depth (Parent node) + 1

Binary tree: Every node can have atmax 2 child nodes.

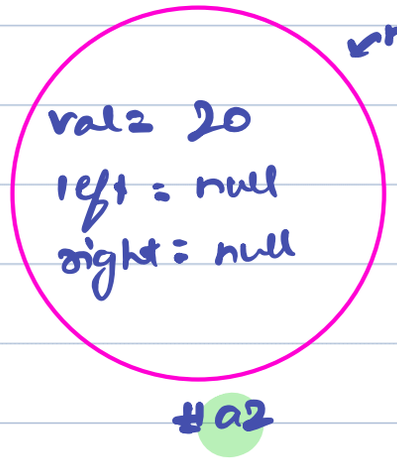



```
class Node {
    int val;
    Node left;
    Node right;
    Node(int n) {
        val = n;
    }
}
```

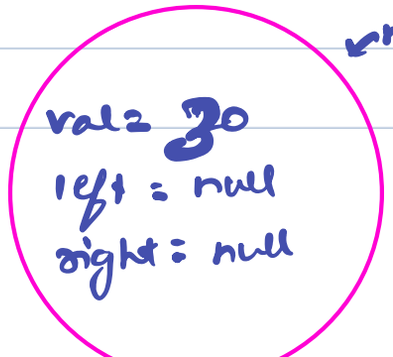
```
Node n1 = new Node(10);
```




```
Node n2 = new Node(20);
```



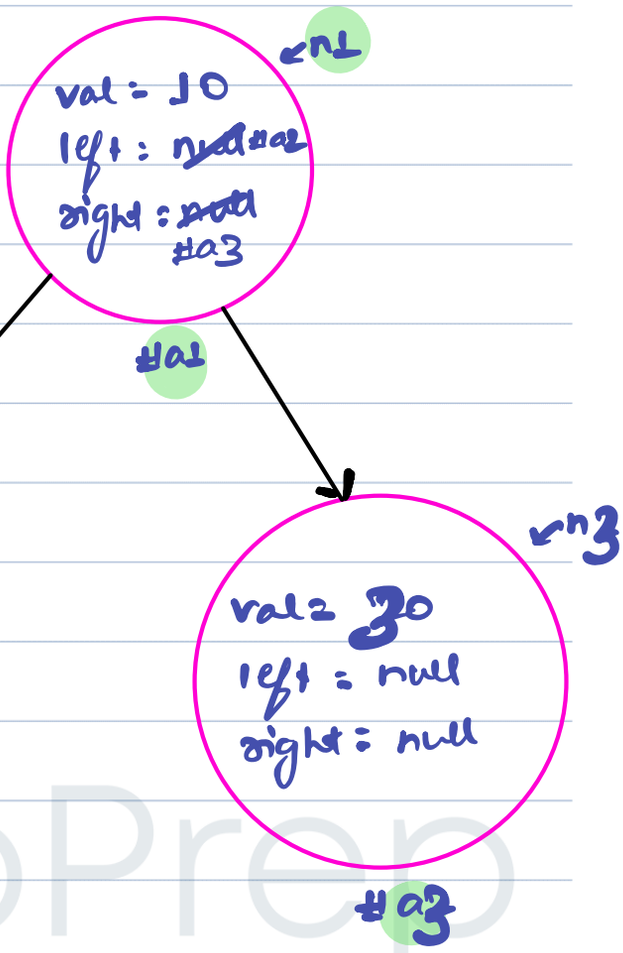

```
Node n3 = new Node(30);
```



#a3

n1.left = n2;

n1.right = n3;



Print (n1.left.val); → 20

Print (n1.right.val); → 30

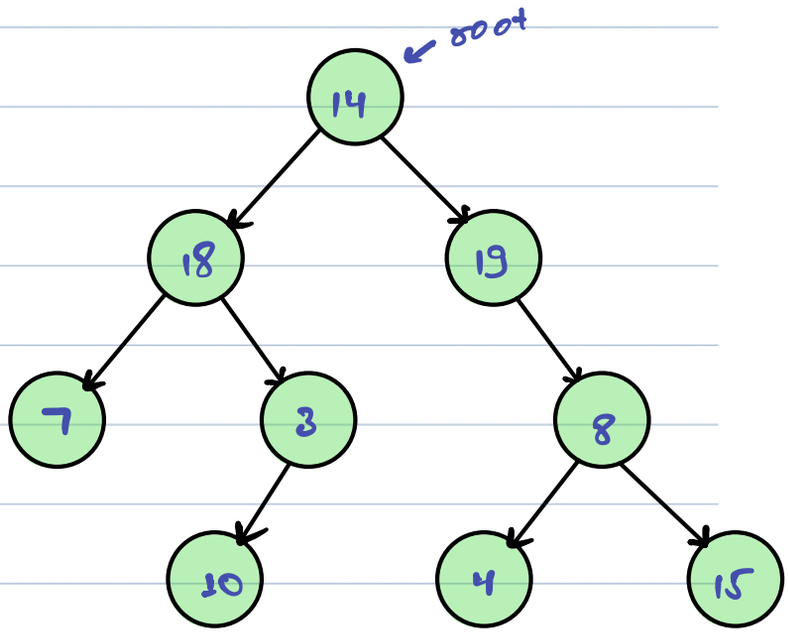
→ Don't worry How the tree got constructed. → {Serialization / deserialization}

It

Solve the given Problem.

//Tree traversal

Recursion traversal

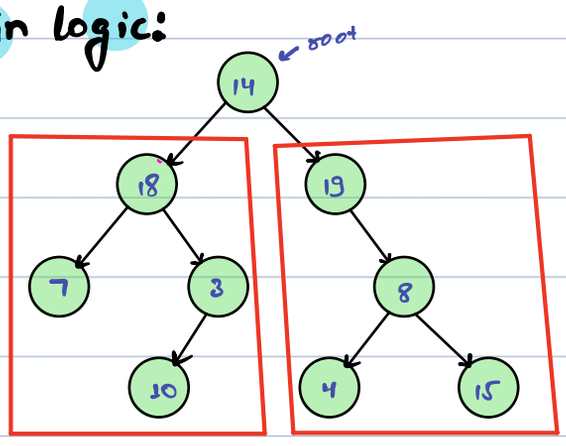



```
void traversal (Node root){
    if (root == null){
        return;
    }

    traversal (root.left);
    traversal (root.right);
}
```

Faith: Given root node, travel every descendent nodes.

main logic:

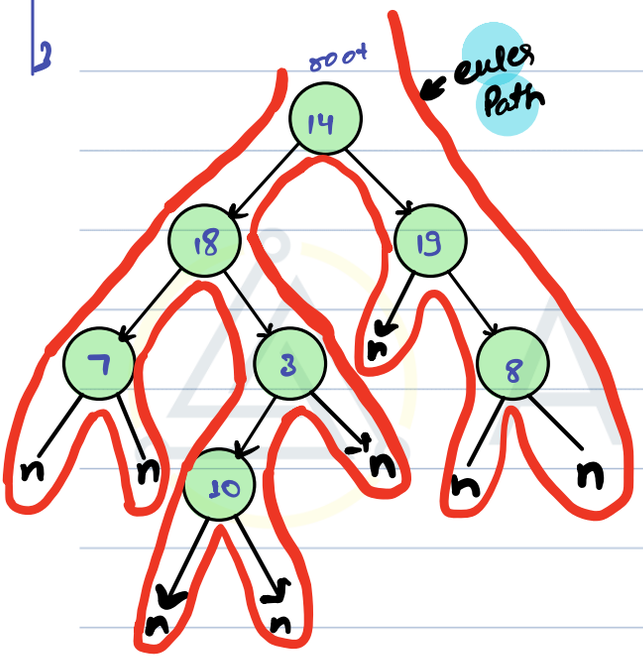


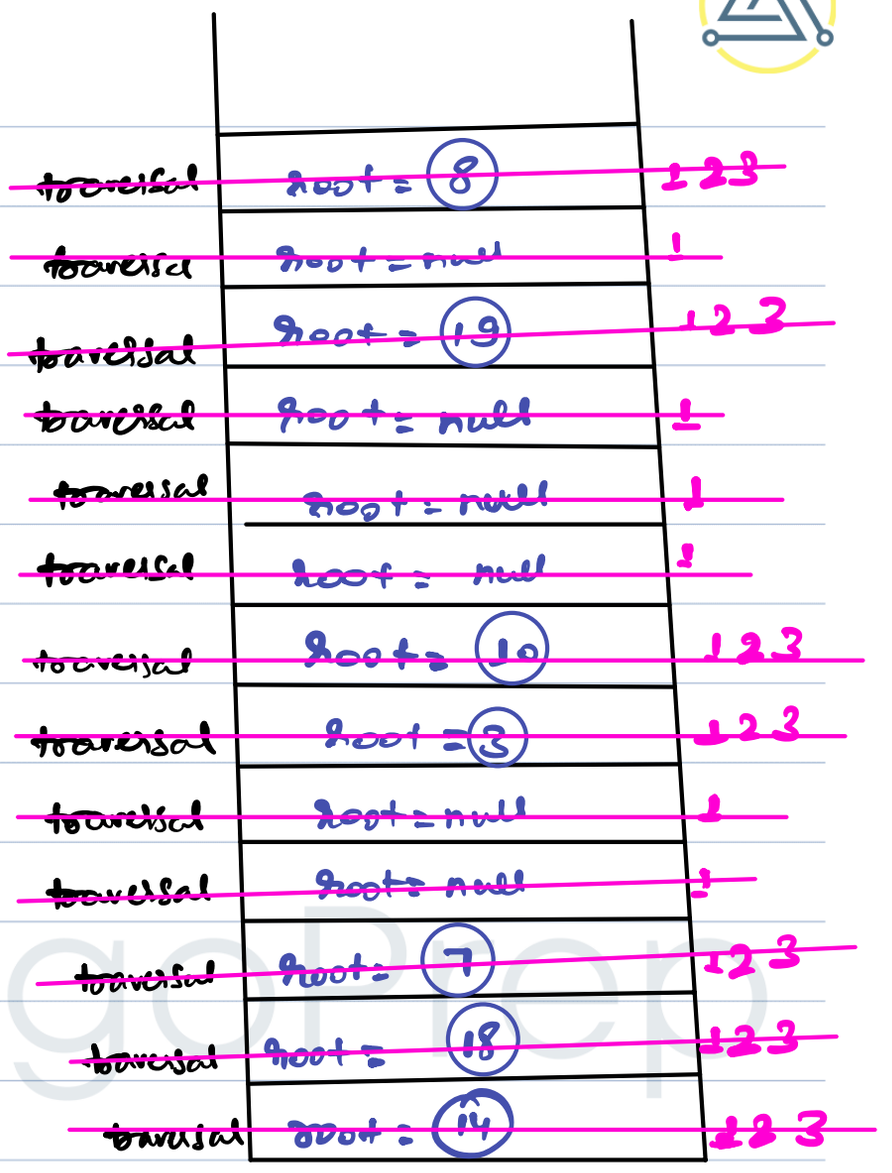
Base case:

```
if (node == null){
    return;
}
```

```
void traversal (Node root){
    if (root == null){
1       return;
    }

2   traversal (root.left);
3   traversal (root.right);
}
```



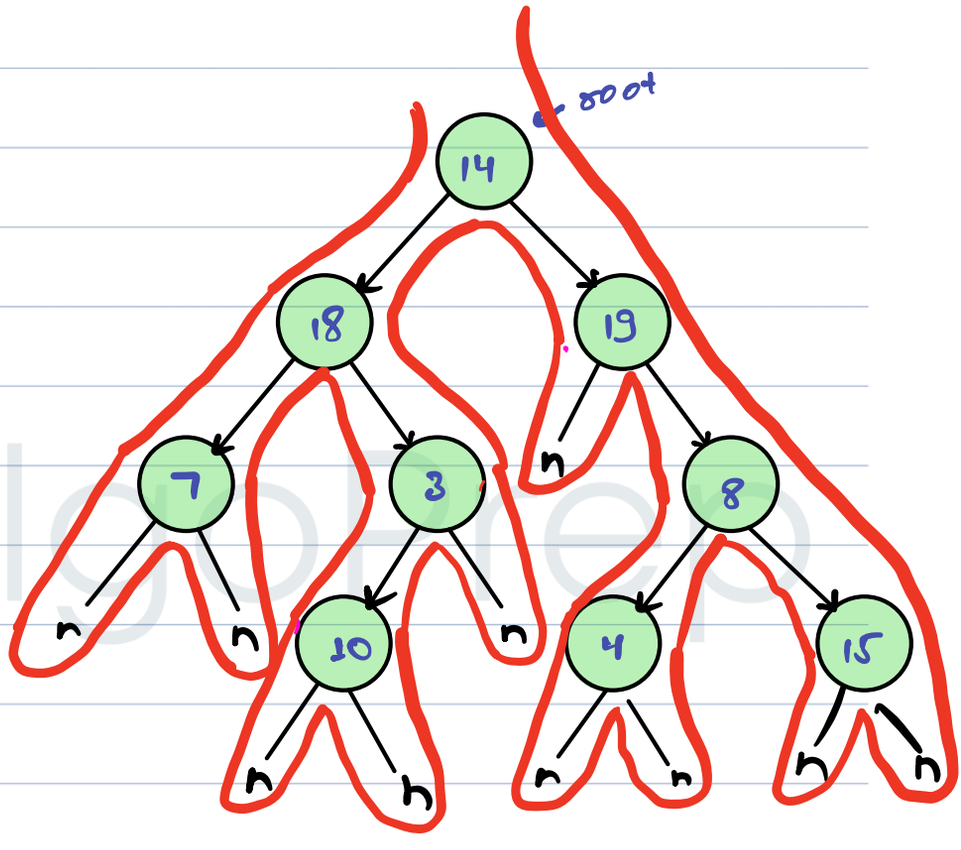


Traversal

- Preorder traversal — NLR
- Inorder traversal — LNR
- Postorder traversal — LRN

Preorder traversal

```
void traversal (node root){
    if (root == null){
        return;
    }
    s.o.p (root.val);
    traversal (root.left);
    traversal (root.right);
}
```



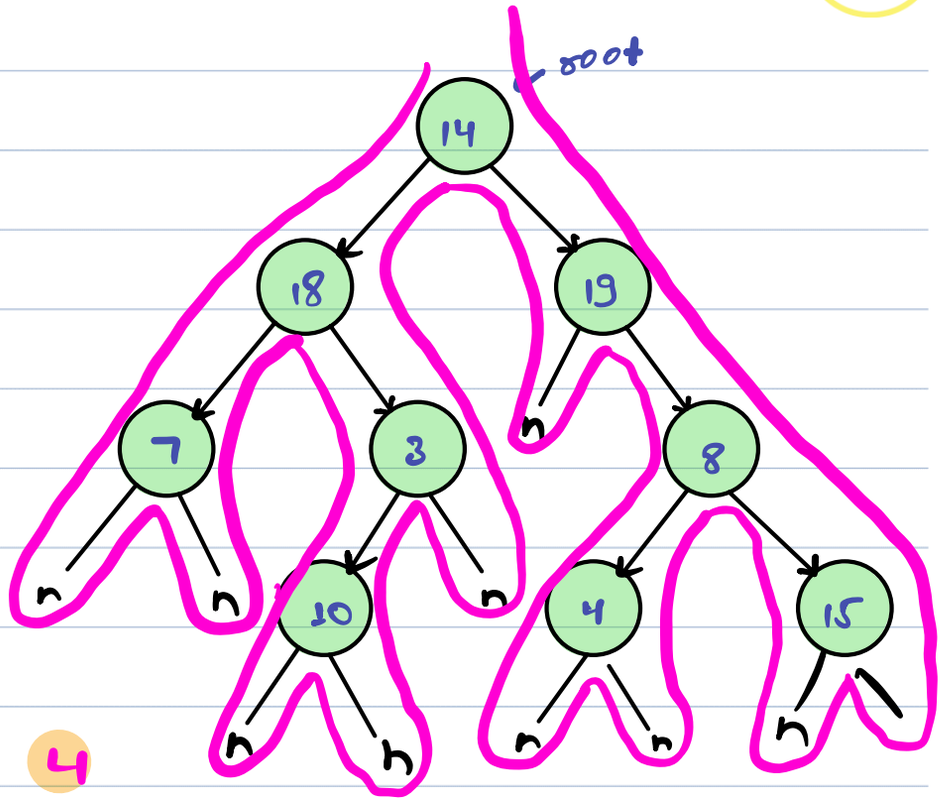
14 18 7 3 10 19 8 4 15

Inorder traversal

↳ LNR

```
void traversal (node root){
    if (root == null){
        return;
    }

    traversal (root.left);
    S.O.P (root.val);
    traversal (root.right);
}
```



7 18 10 3 14 19 4 8 15

Postorder traversal