Data Structure & Algorithms

Nilesh Ghule

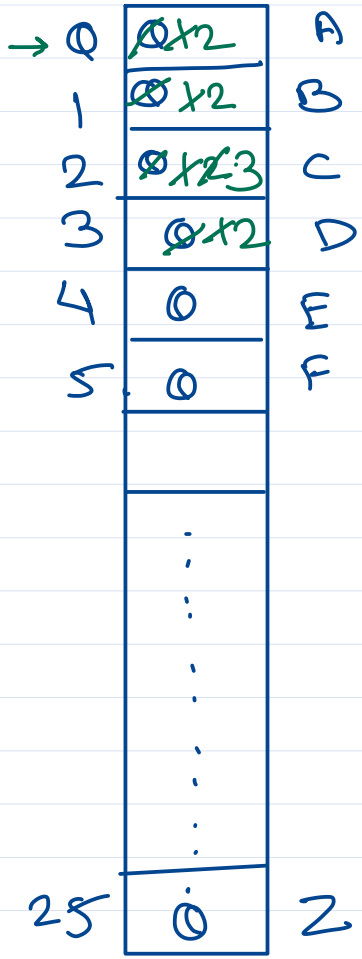
Linked List – Competitive programming

- Sort the singly linked list.

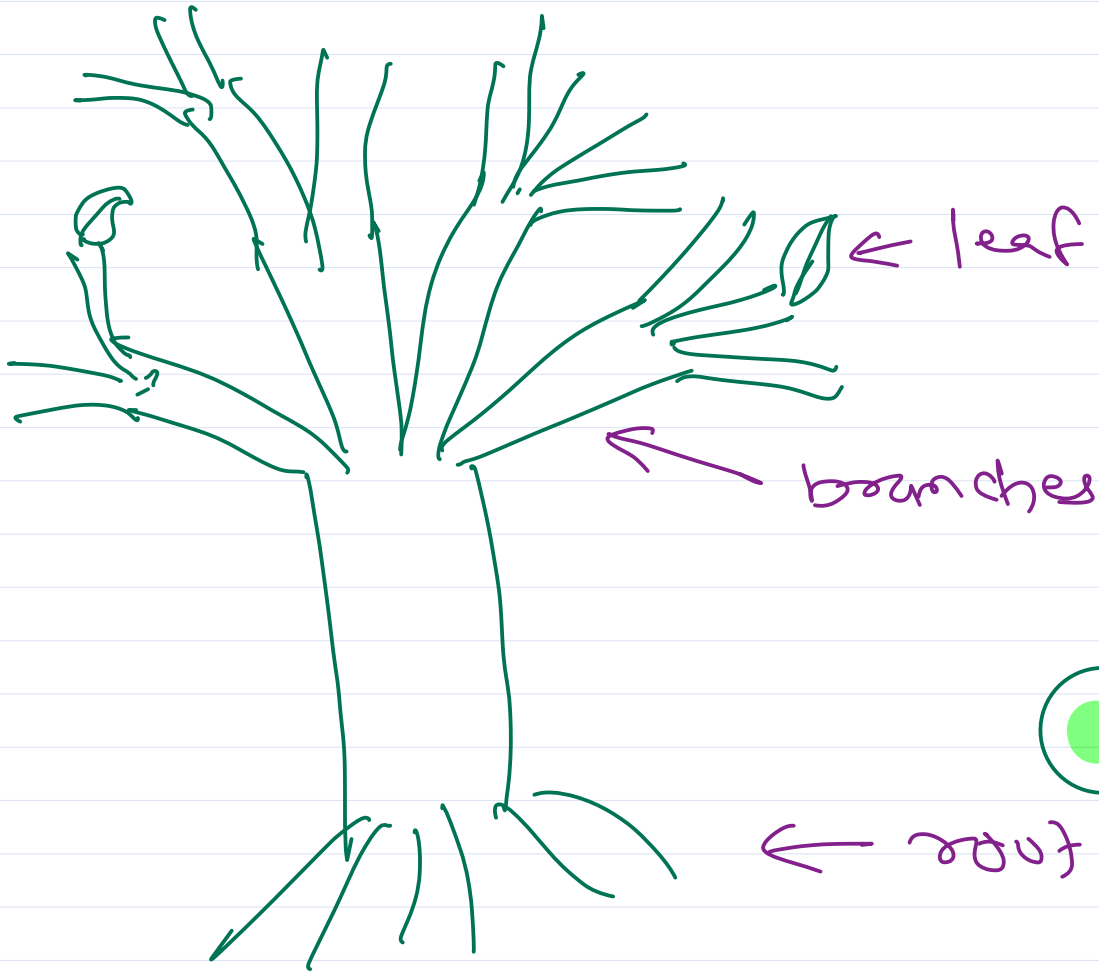
Given a string, count occurence of each alphabet.

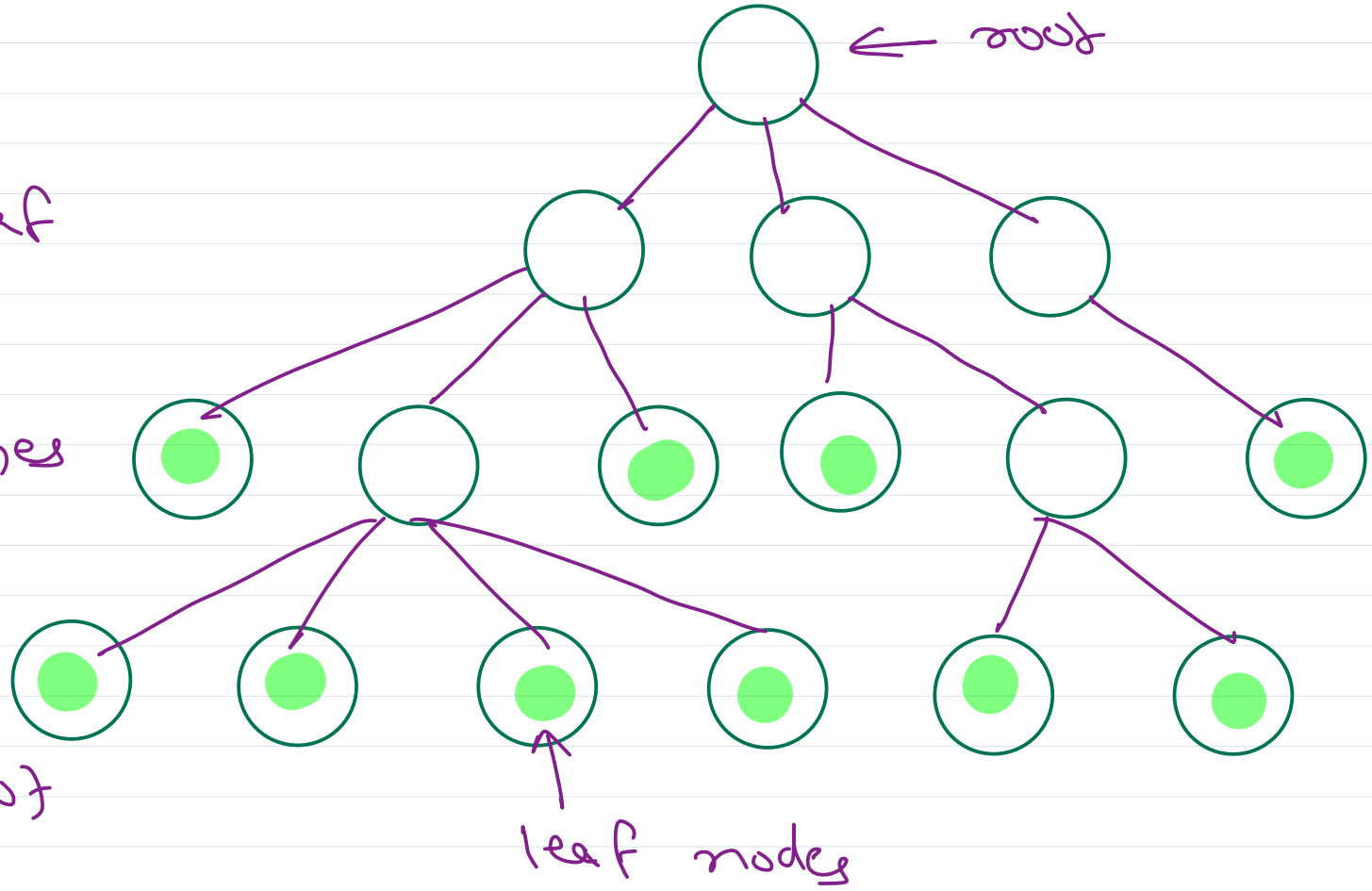
A B C B C D A D E

A → 2
B → 2
C → 3
D → 2



$T \propto n$ (n is length of string).
$\rightarrow O(n)$.

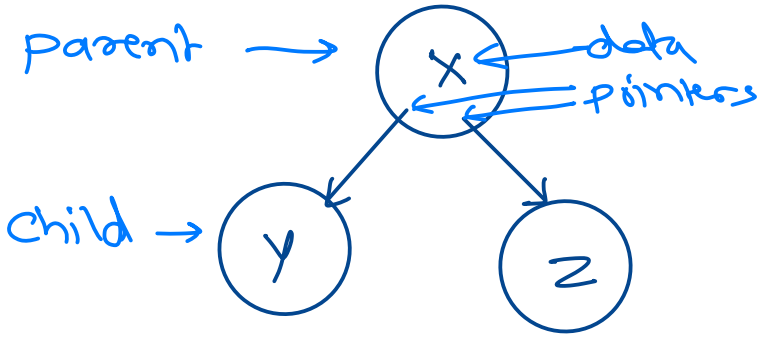

leaf
branches
root


root
leaf nodes

Tree Definition

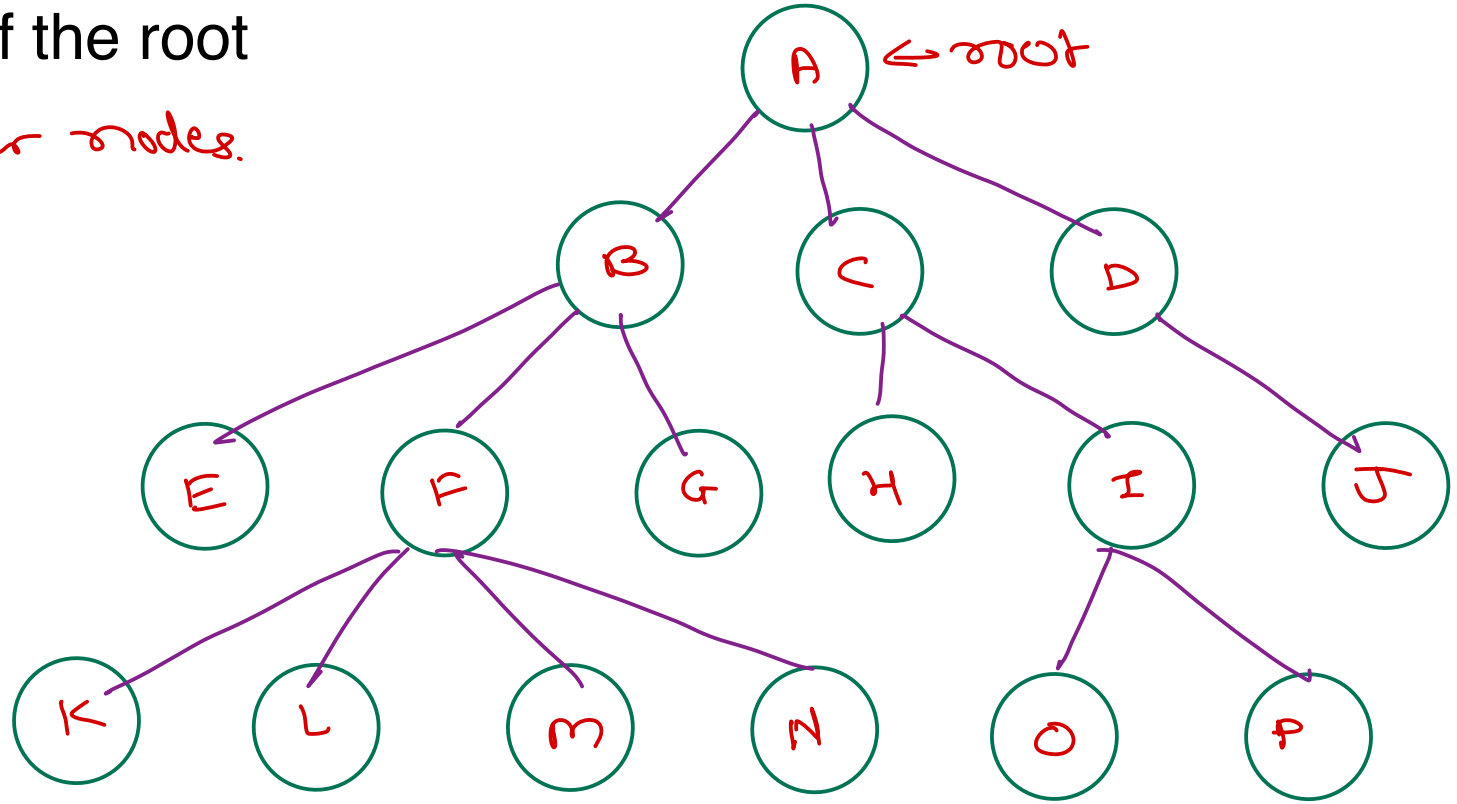
- **Tree** is a finite set of nodes with one specially designated node called the "**root**" and the remaining node are partitioned into disjoints sets T1 to Tn, where each of those sets is a TREE.
- T1 to Tn are called **sub-trees** of the root

Node = Data + pointers to other nodes.



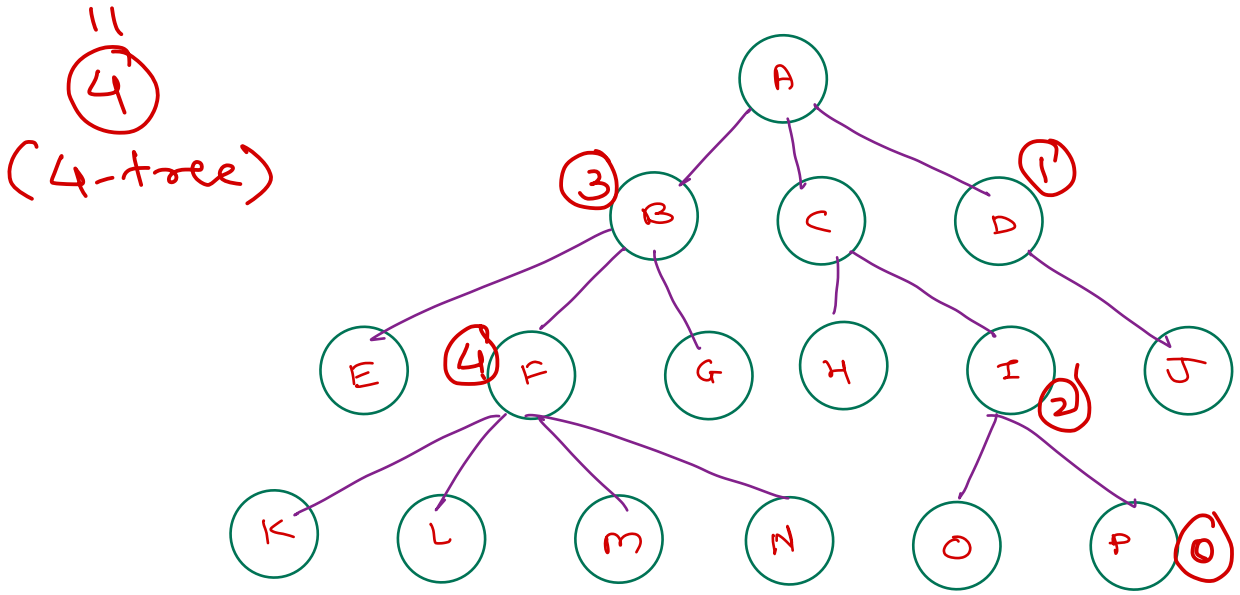
Tree is a non-linear datastructure.

Tree is used to represent hierarchial data.



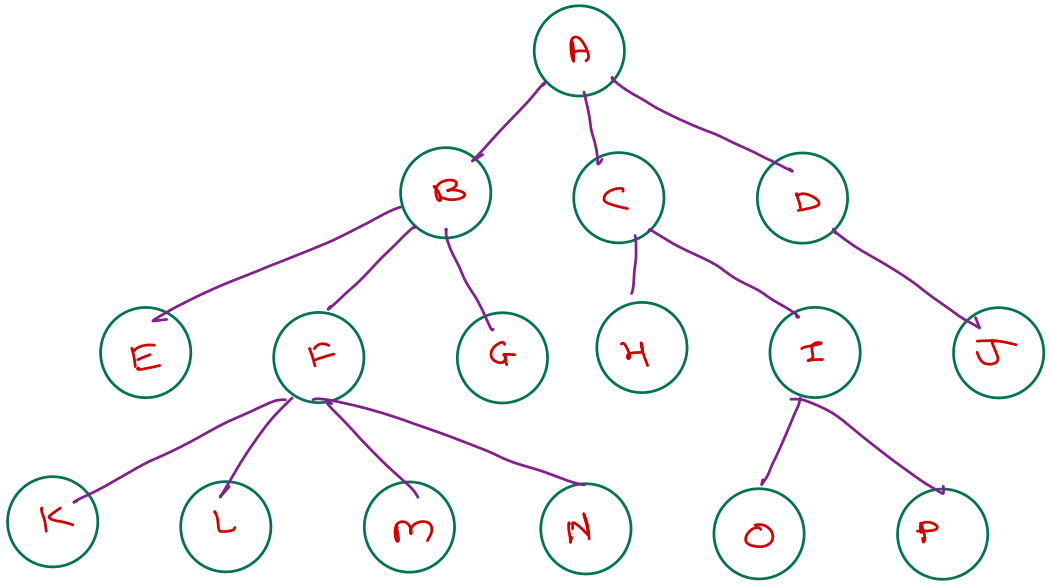
Tree terminologies

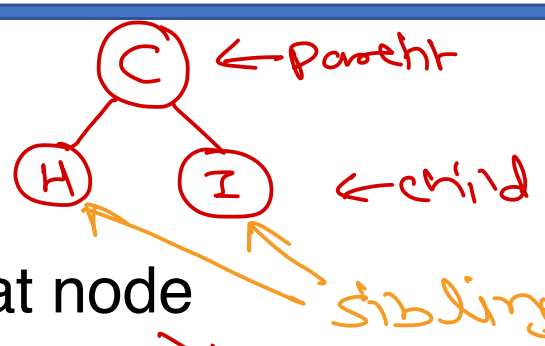
- Node: A item storing information and branches to other nodes
- Null Tree: Tree with no node (empty tree)
- Leaf Node: Terminal node of a tree & does not have any child node connected to it → degree = 0
- Degree of a Node: No of sub trees of a node (No of child nodes)
- Degree of a tree: Degree of a tree is maximum degree of a node in the tree



Tree terminologies

- Parent Node: node having other child nodes connected to it
- Siblings: Children of the same parents
- Descendants: all those node which are reachable from that node
- Ancestor: all the node along the path from the root to that node



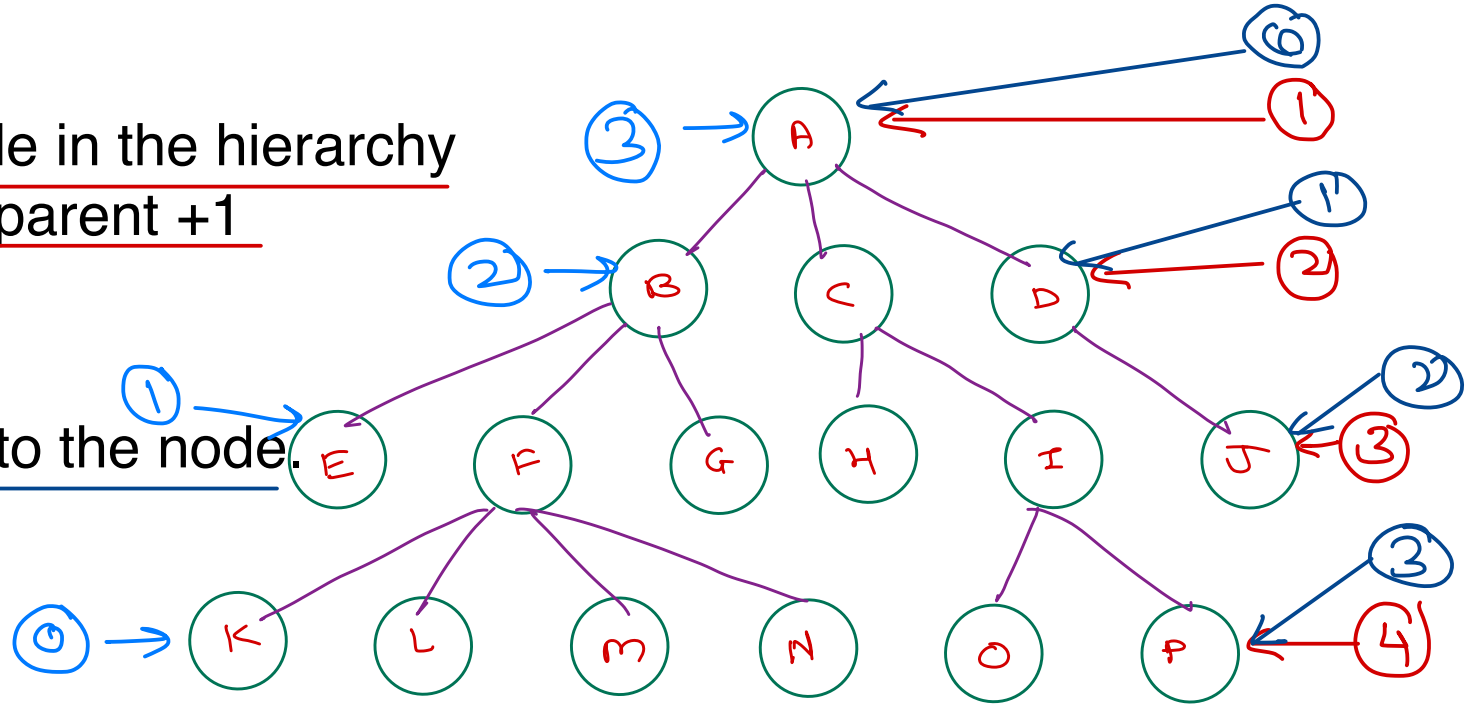


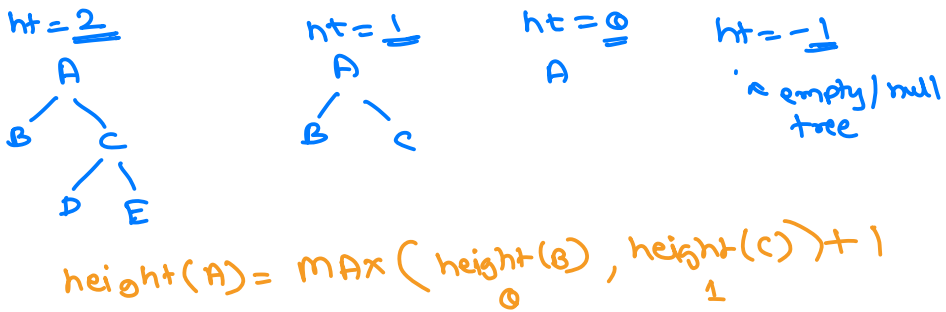
e.g. desc of C:
H, I, O, P

e.g.
ancests of H
A, B, F

Tree terminologies

- Level of a Node:
  - Indicates the position of the node in the hierarchy
  - Level of any node is level of its parent +1
  - Level of root is 1
- Depth of a node:
  - Number of nodes from the root to the node.
  - Depth of root is 0
  - Level = Depth + 1
- Height of a node:
  - Number of nodes from the node to its deepest leaf.
  - Height of node = max height of its child + 1
  - Height of empty/null tree is -1
- Height of a tree: Height of root of the tree.
- Traversal: Visiting each node of tree exactly once

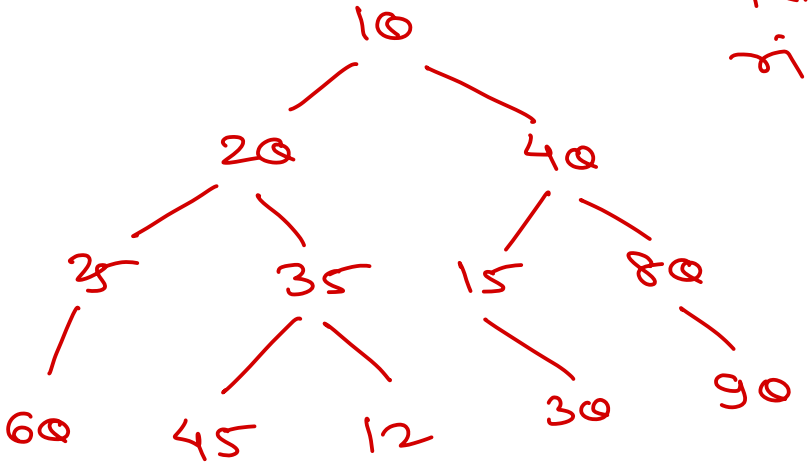




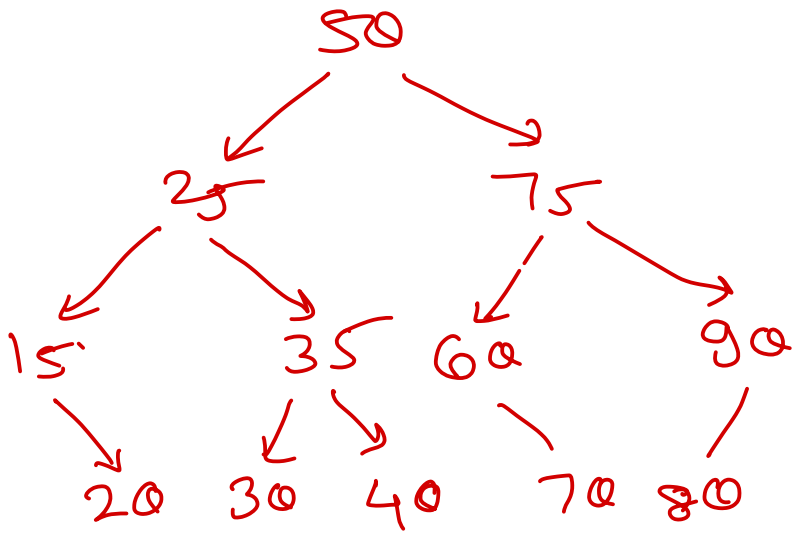
Types of trees

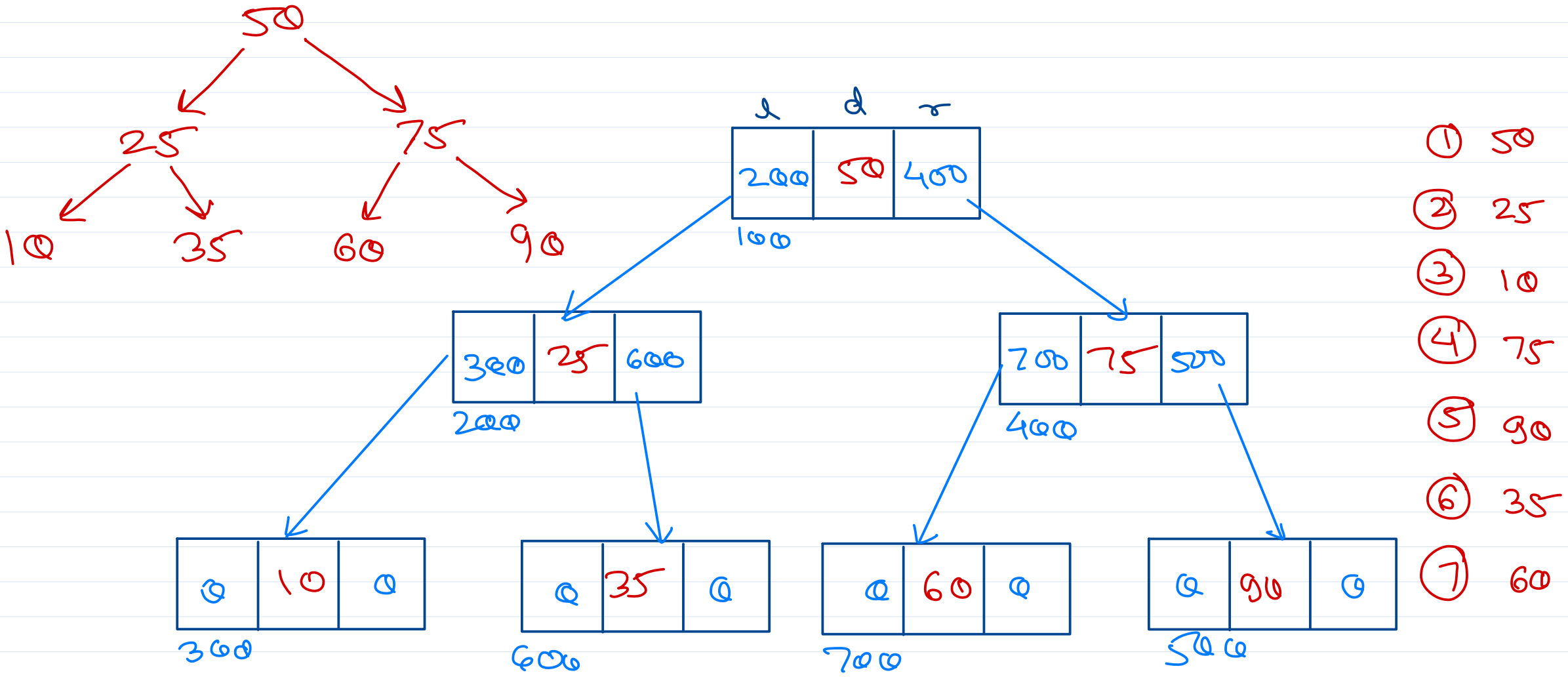
- Binary Trees → each node has max 2 child nodes.
  - It is a finite set of nodes partitioned into three sub sets:- Root, Left sub tree, Right sub tree
- Binary Search tree
  - A binary search tree is a binary tree in which the nodes are arranged according to their values.

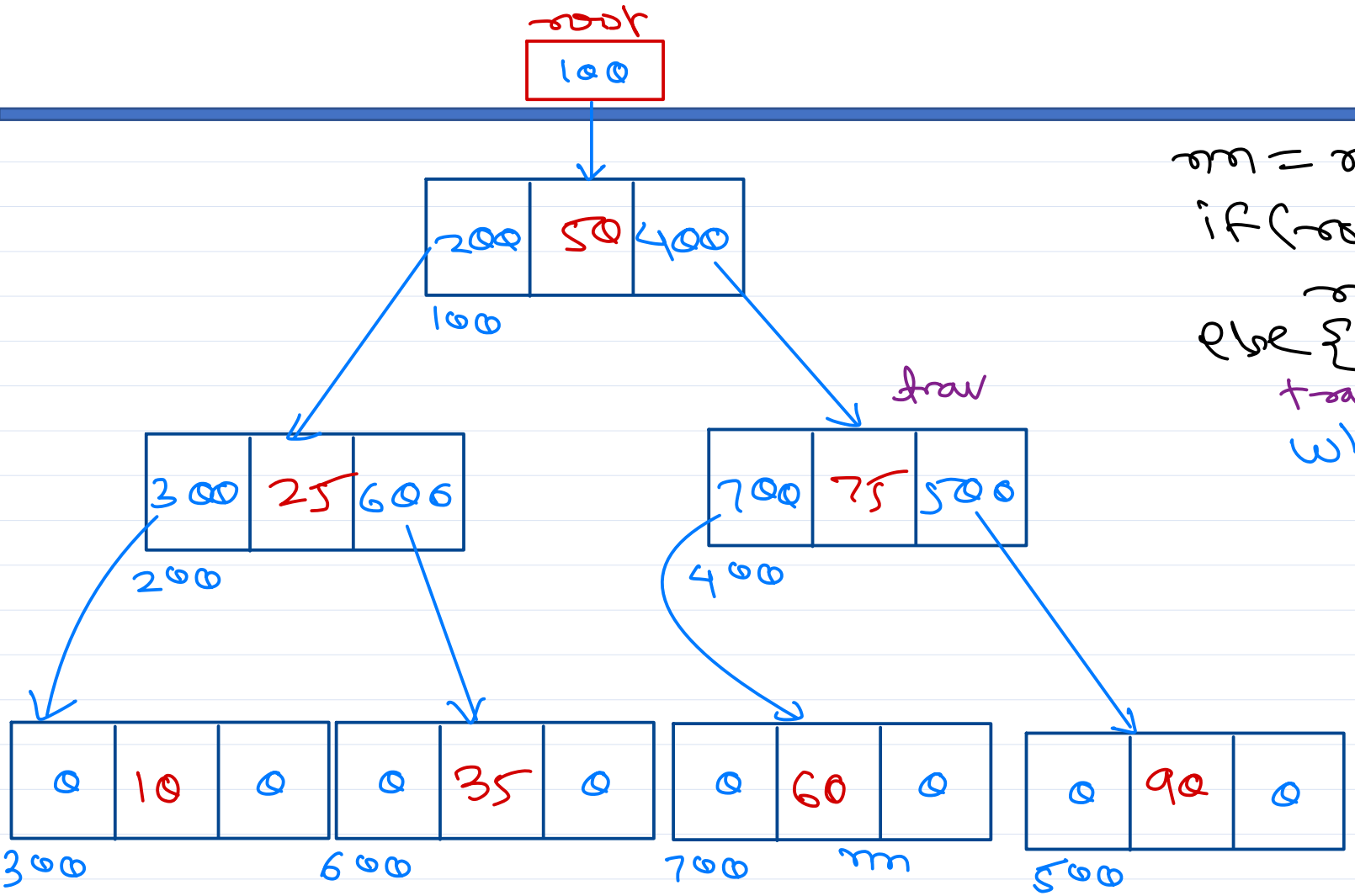
↳ left node is smaller than parent,
right node is greater or equal to parent.



Bin Tree (2-Tree)
Degree of tree = 2




50
25
75
10
35
60
90
l
d
r
2000
50
4000
1000
3000
25
6000
2000
7000
75
5000
4000
0
10
0
3600
0
35
0
6000
0
60
0
7000
0
90
0
5000
1 50
2 25
3 10
4 75
5 90
6 35
7 60

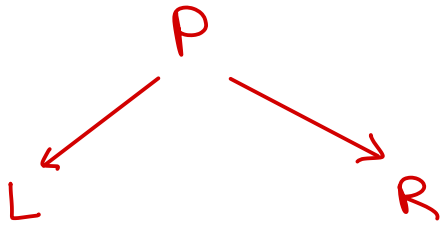



```
nn = new Node(val);
if(root == null)
    root = nn;
else{
    trav = root;
    while(true){
        if(val < trav.data){
            if(trav.left == null){
                trav.left = nn;
                break;
            }else
                trav = trav.left;
        }else{
            if(trav.right == null){
                trav.right = nn;
                break;
            }else
                trav = trav.right;
        }
    }
}
```

50 25 10 75 90 35 60

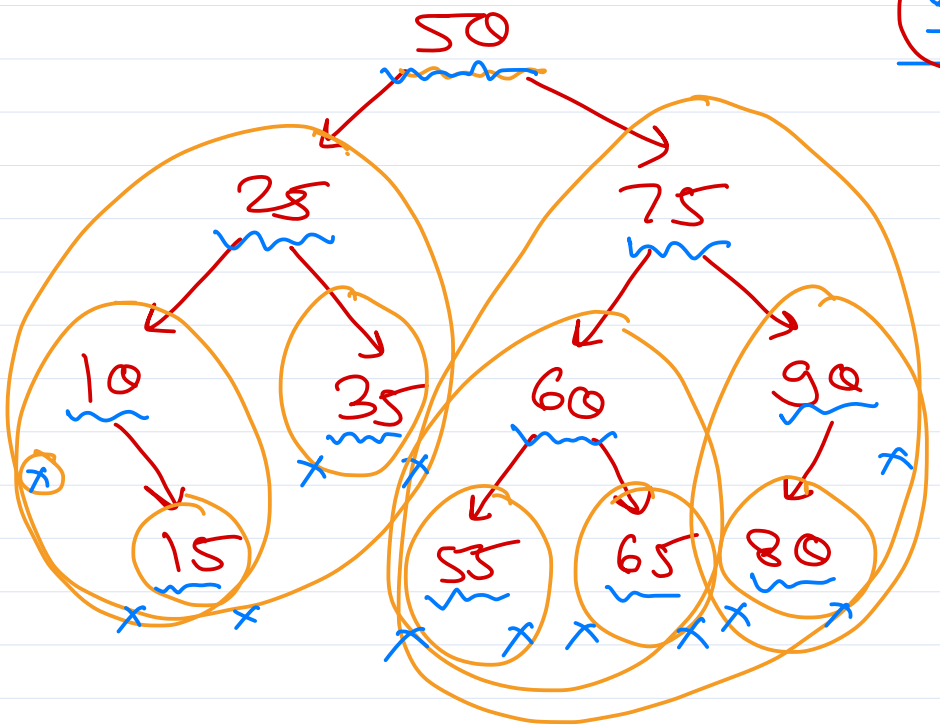
Binary Tree Traversal

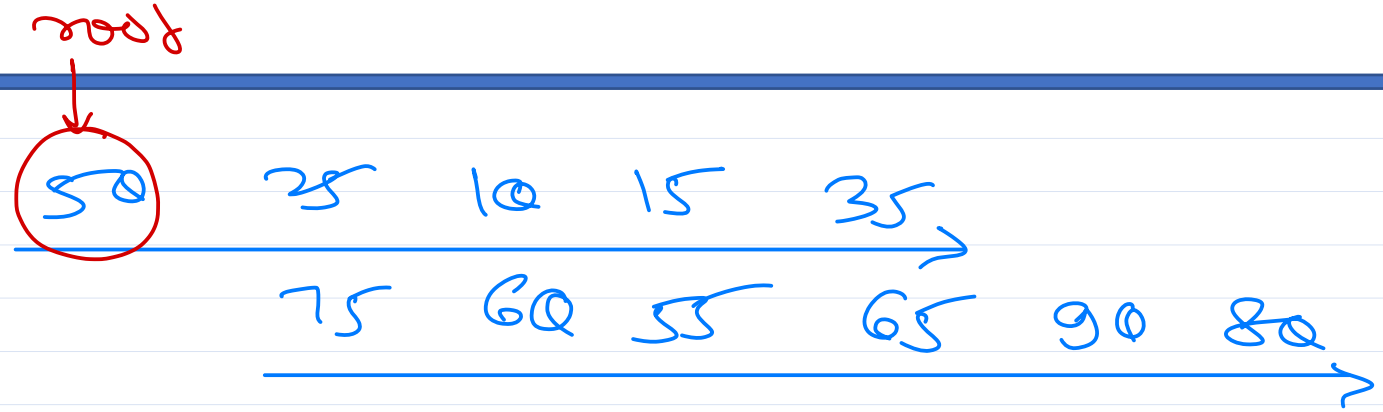
- In-order: L P R
- Pre-Order: P L R
- Post-Order: L R P



- The traversal algorithms can be implemented easily using recursion.
- Non-recursive algorithms for implementing traversal needs stack to store node pointers.

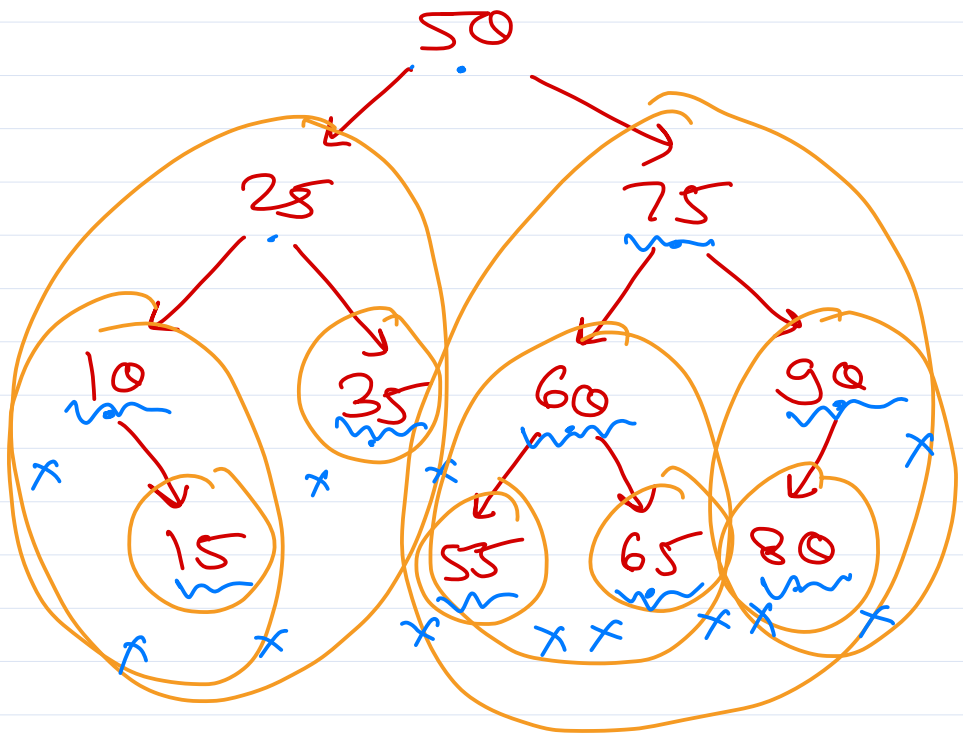
PreOrder






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

InOrder

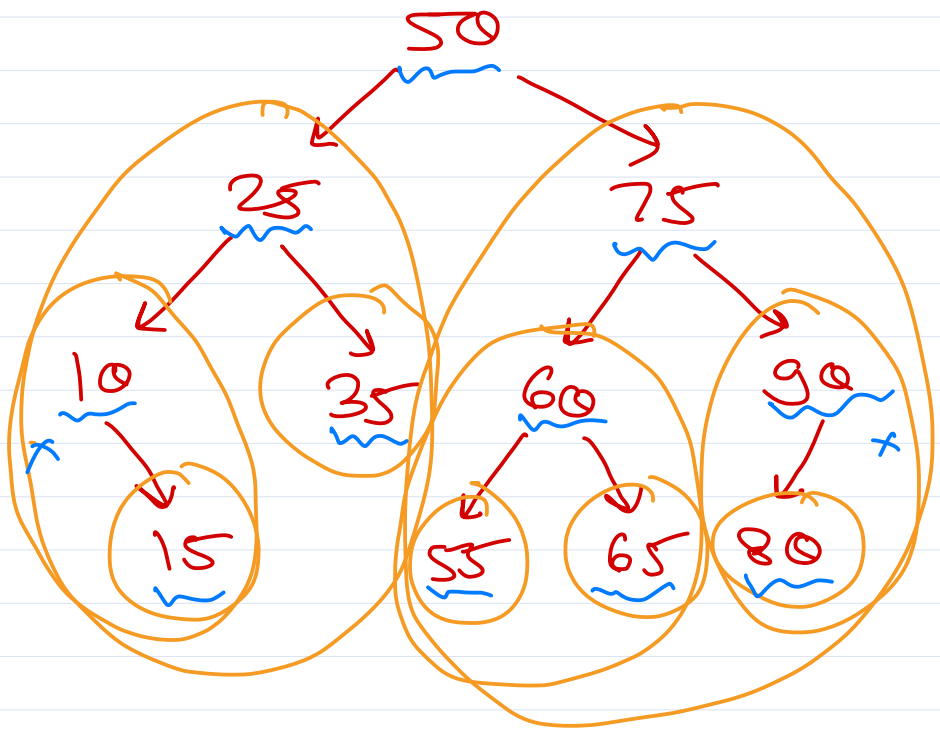


10 15 25 35 50 → Sorted output

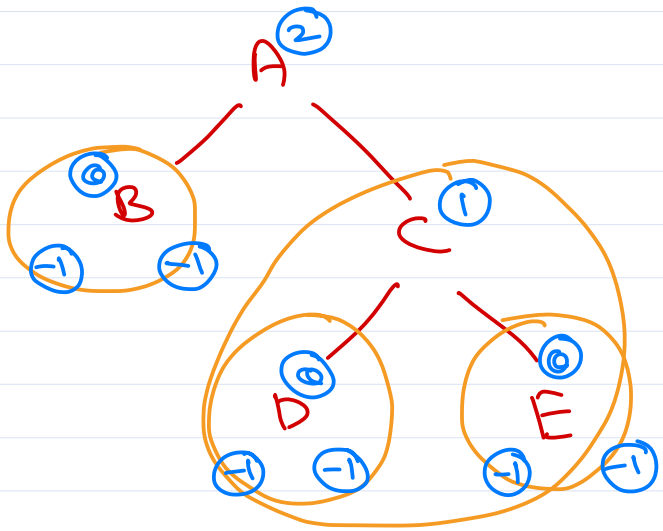
55 60 65 75 80 90 →

```
inorder(tnav){
    if(tnav==null)
        return;
    inorder(tnav.left);
    print(tnav.data);
    inorder(tnav.right);
}
```

PostOrder



15 10 35 25 55 65 60
80 90 75 50




```
int height(trav) {
    if(trav == null)
        return -1;

    hl = height(trav.left);
    hr = height(trav.right);
    max = hl > hr ? hl : hr;
    return max + 1;

}
```

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>