Explanation

Trees

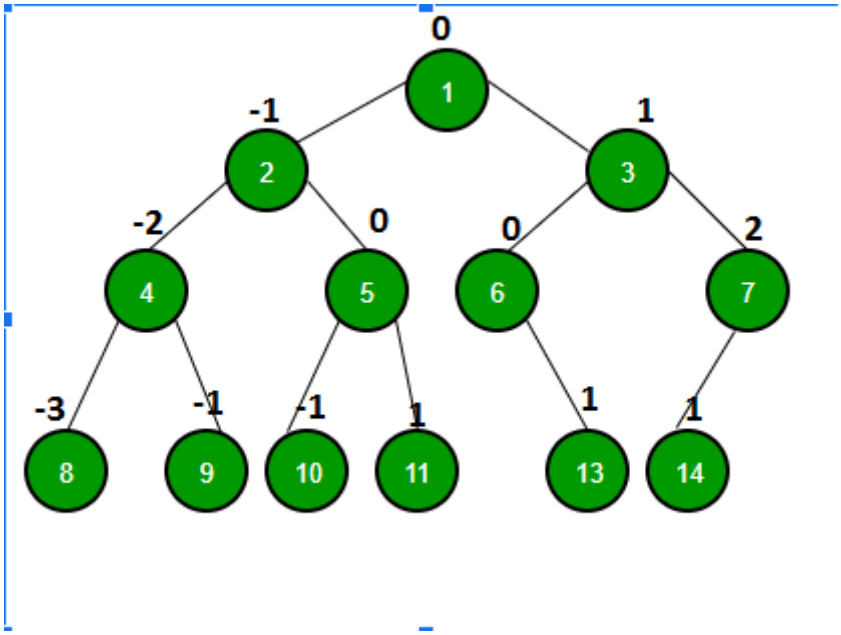
Non - linear data structure
Hierarchical nature
Binary and n-ary tree
Bst
Different types - full, perfect, complete
Traversals - in, pre, post, level dfs, bfs
A tree is a graph
Balanced binary tree

N - nodes n-1 edges => tree (directed)
N - nodes x edges => graph

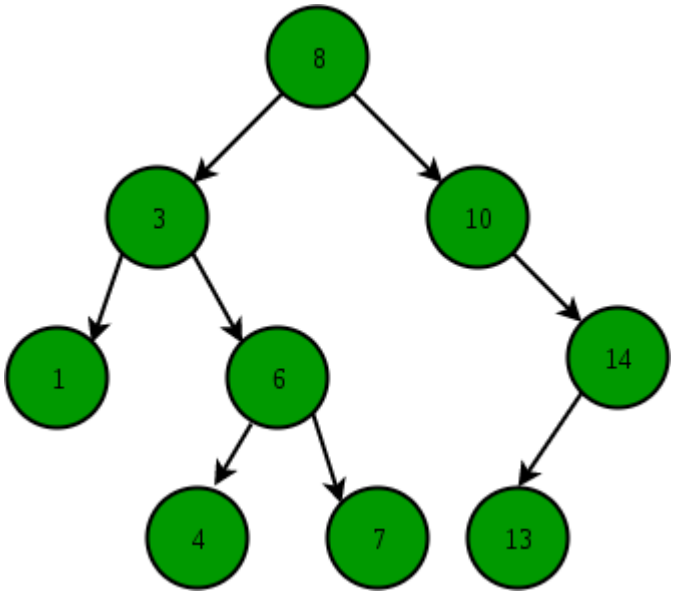
```
Class node{
Public:
        Int val;
        node* left;
        node* right;
        node(int d)
        {
                Val = d;
                Left  right = NULL;
        }
};
Class tree{
public:
        node* root;
        // FUNCTIONS
};
```

(a) Inorder (Left, Root, Right) : 4 2 5 1 3
(b) Preorder (Root, Left, Right) : 1 2 4 5 3
(c) Postorder (Left, Right, Root) : 4 5 2 3 1

Que. level order and pre order are given, can you construct a unique binary tree



Left view: 1, 2, 4, 8
Right view: 1, 3, 7, 14
Top view: 8, 4, 2, 1, 3, 7
Bottom view: 8, 4, 10, 6, 14, 7



Left: 8, 3, 1, 4
Right: 8, 10, 14, 13
Top: 1, 3, 8, 10, 14
Bottom: 1, 4, 6, 13, 14

Level - 8, 3, 10, 1, 6, 14, 4, 7, 13
Level spiral - 8, 10, 3, 1, 6, 14, 13, 7, 4

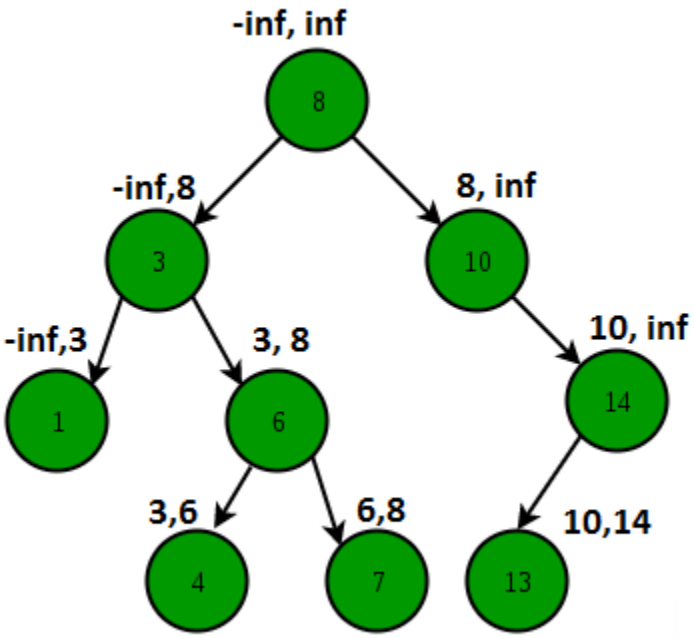
Lca - lowest common ancestor
3 cases -

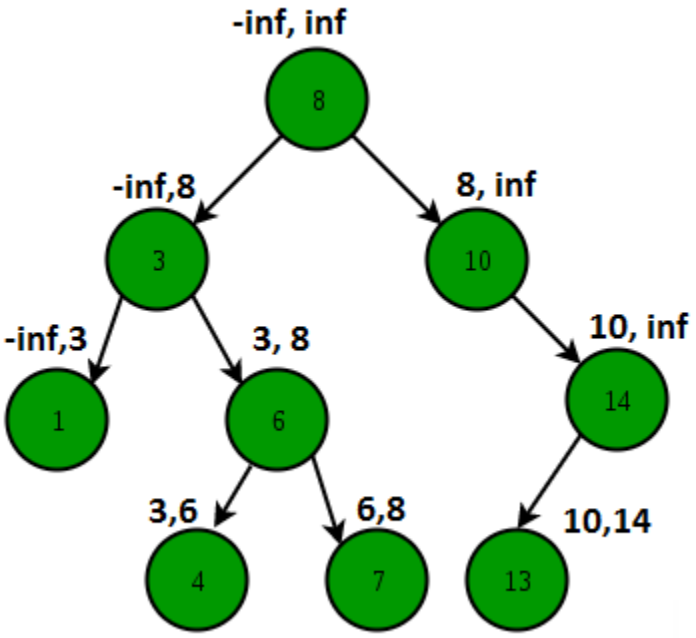
Diameter of tree - 3 cases
Identical or mirror trees

Construct tree from inorder and preorder -
Find root from preorder and then check in inorder left and right subtree and do same for them also

BST - binary search tree
Search
Insert
Delete
Inorder successor
Inorder predecessor

Check if binary tree is bst or not
By using preorder



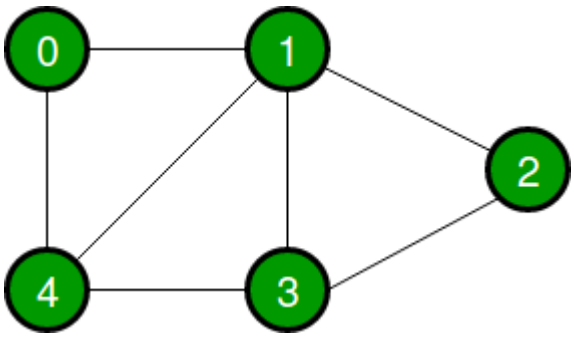


Tree construction
Level order
Pre order 8 3 1 N N 6 4 N N 7 N N 10 N 14 13 N N N

node* root
root->left root->right root->val

N nodes/vertices
Tree - n-1
Graph - (n-1) - (n(n-1)/2)

Matrix = O(v*v)
List = O(e)



Bfs - 0, 4, 1, 3, 2
Dfs - 0, 4, 1, 3, 2

Adj list
0 -> 1,4
1 -> 0,2,3,4
2 -> 1,3
3 -> 1,2,4
4 -> 0,1,3

N - vertices
M - edges

```
vector<vector<int>> v(n+1)
for(int i=0;i<m;i++)
{
        Int x,y;
        cin>>x>>y;
        v[x].push_back(y);
        v[y].push_back(x);
}
```