មហាវិទ្យាល័យវិស្វកម្ម
FACULTY OF ENGINEERING

# Data Structure & Algorithm II

## Lecture 5
## Binary Tree

Chhoeum Vantha, Ph.D.

Telecom & Electronic Engineering

# Content

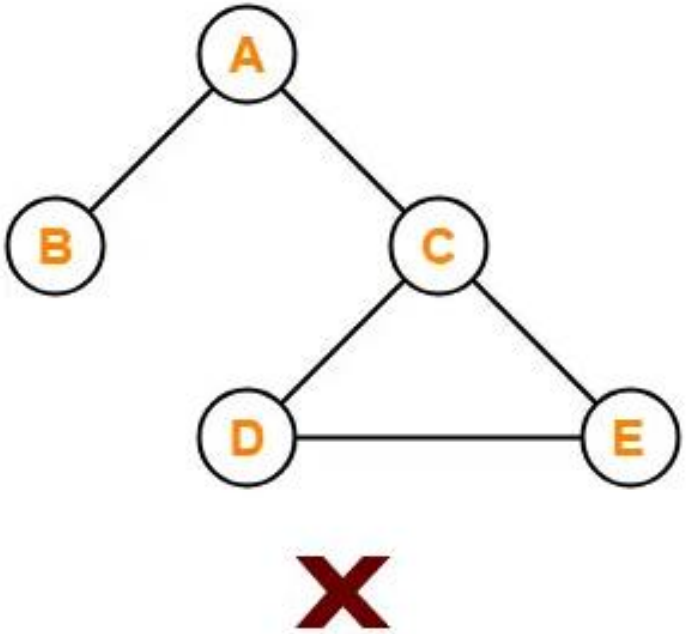
- Tree
- Terminology
- Traversal

# Tree Definition

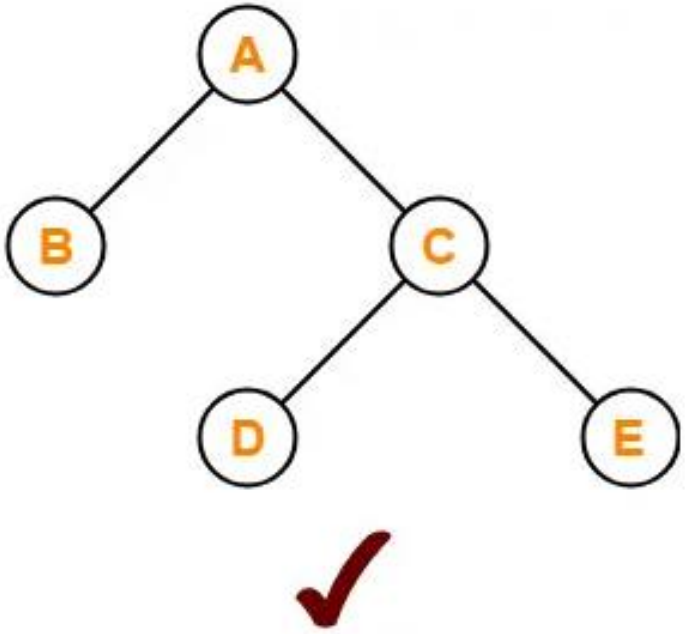
- Tree is a non-linear data structure which organizes data in a hierarchical structure and this is a recursive definition.
- If in a graph, there is one and only one path between every pair of vertices, then the graph is called a tree.

# Tree Definition

Example




This graph is not a Tree

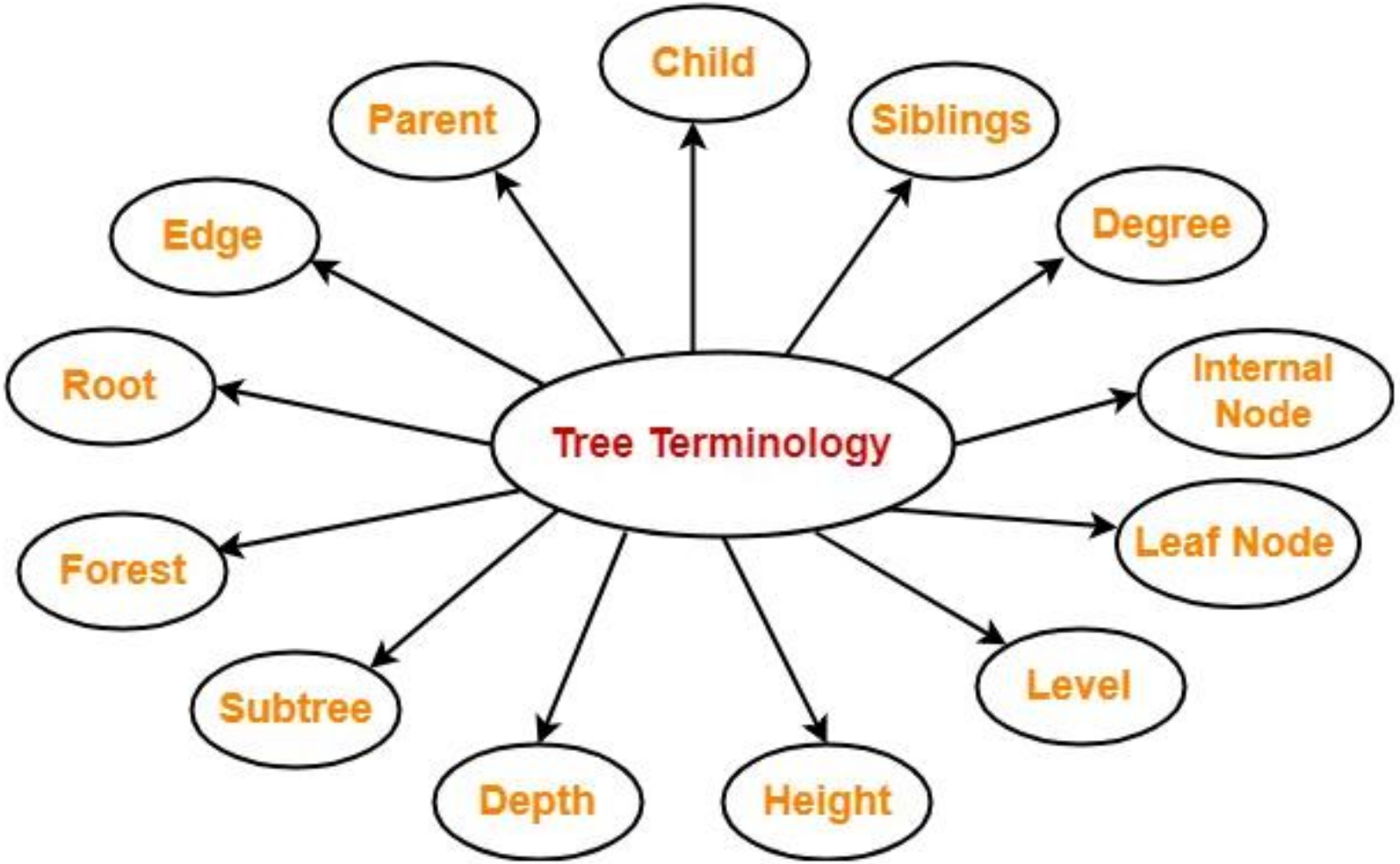



This graph is a Tree

# Tree Definition - Properties-

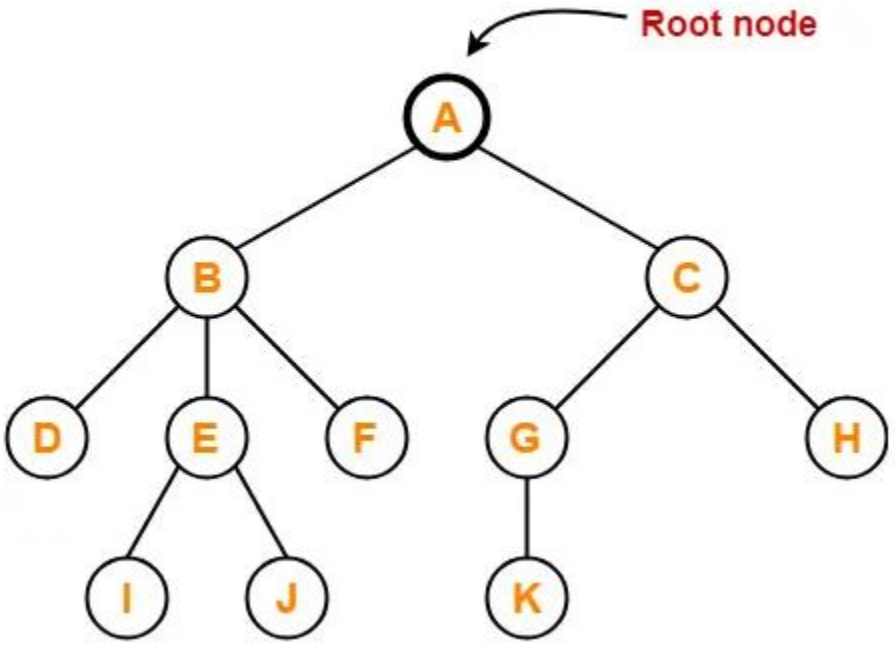
- There is one and only one path between every pair of vertices in a tree.
- A tree with n vertices has exactly (n-1) edges.

# Tree Terminology



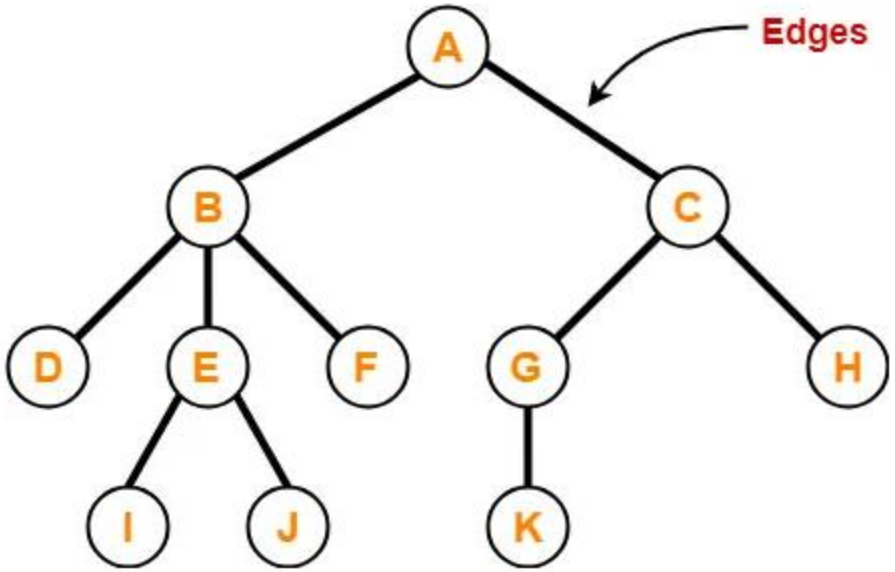
# Tree Terminology-Root

- The first node from where the tree originates is called as a root node.
- In any tree, there must be only one root node.
- We can never have multiple root nodes in a tree data structure.



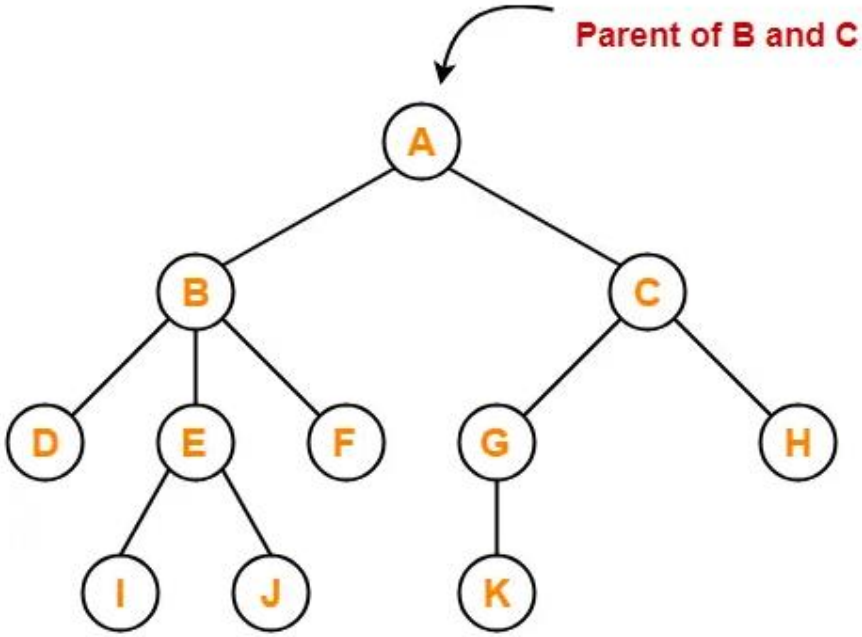
# Tree Terminology-Edge

- The connecting link between any two nodes
- In a tree with **n** number of nodes, there are exactly (n-1) number of edges.



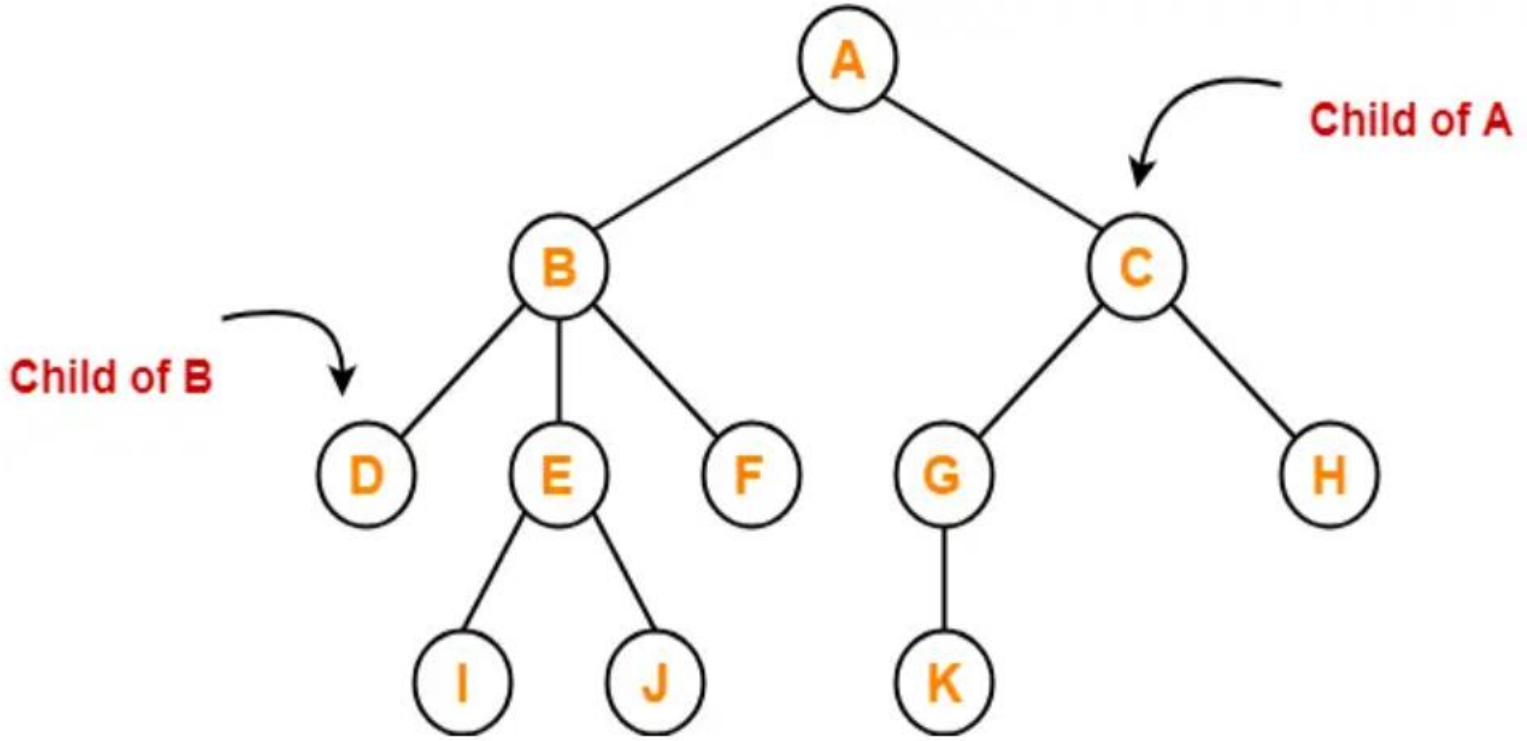
# Tree Terminology-Parent

- The node which has a branch from it to any other node
- the node which has one or more children
- In a tree, a parent node can have any number of child nodes.



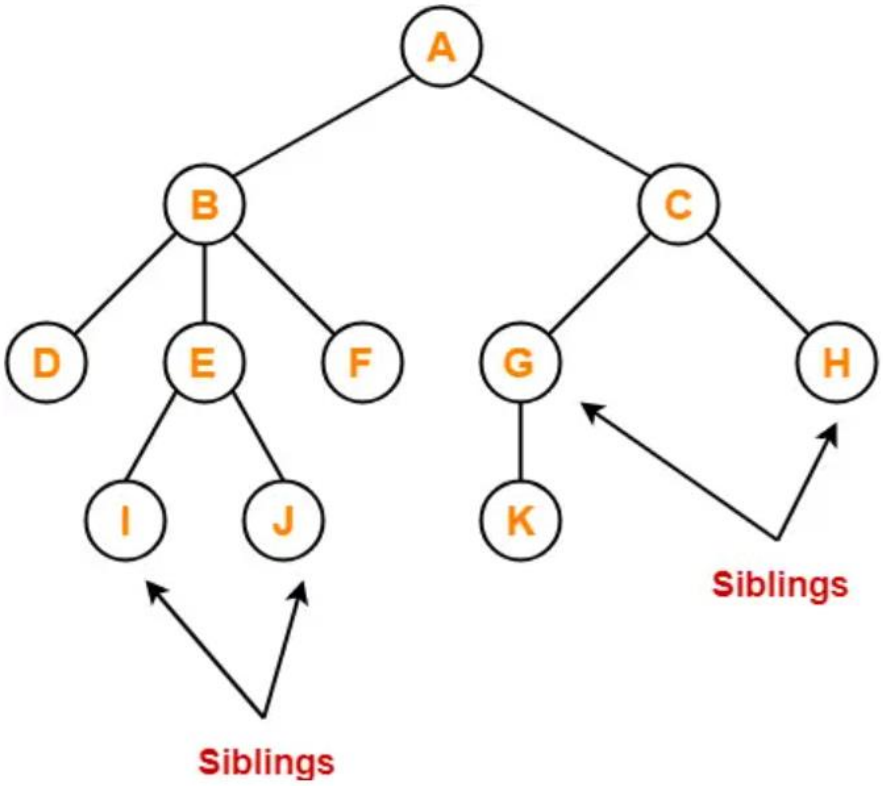
# Tree Terminology-Child

- The node which is a descendant of some node
- All the nodes except root node are child nodes.



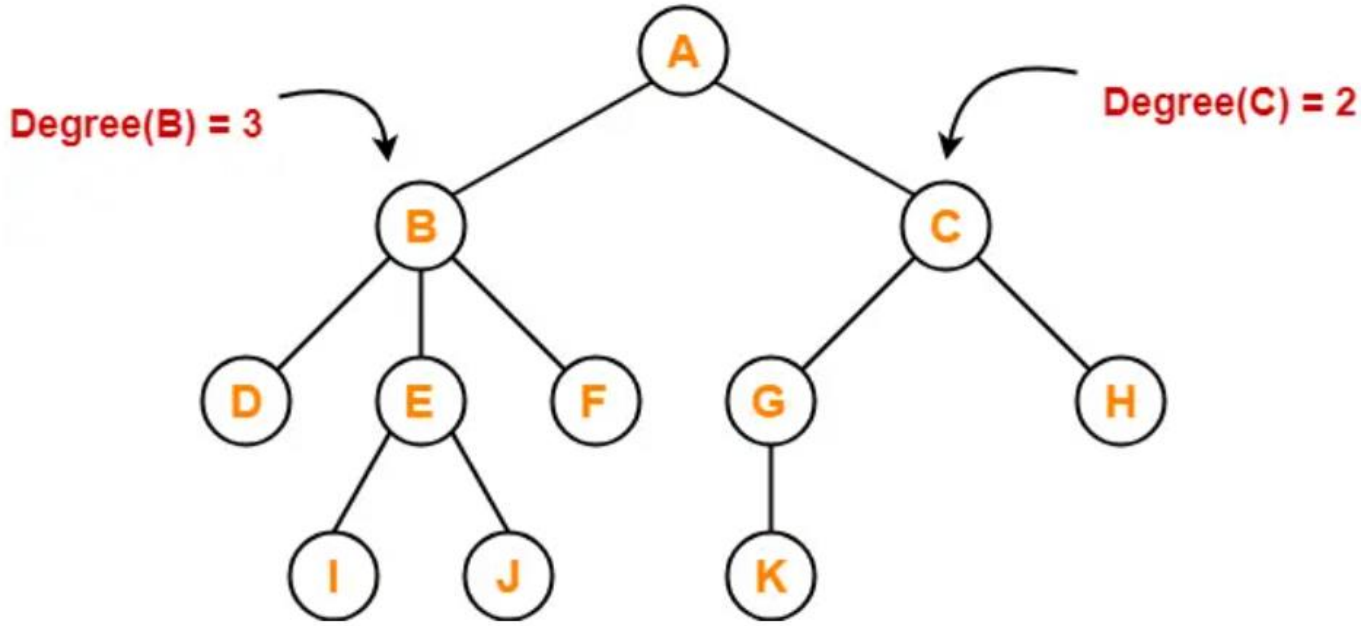
# Tree Terminology - Siblings

- Nodes which belong to the same parent
- nodes with the same parent are sibling nodes.



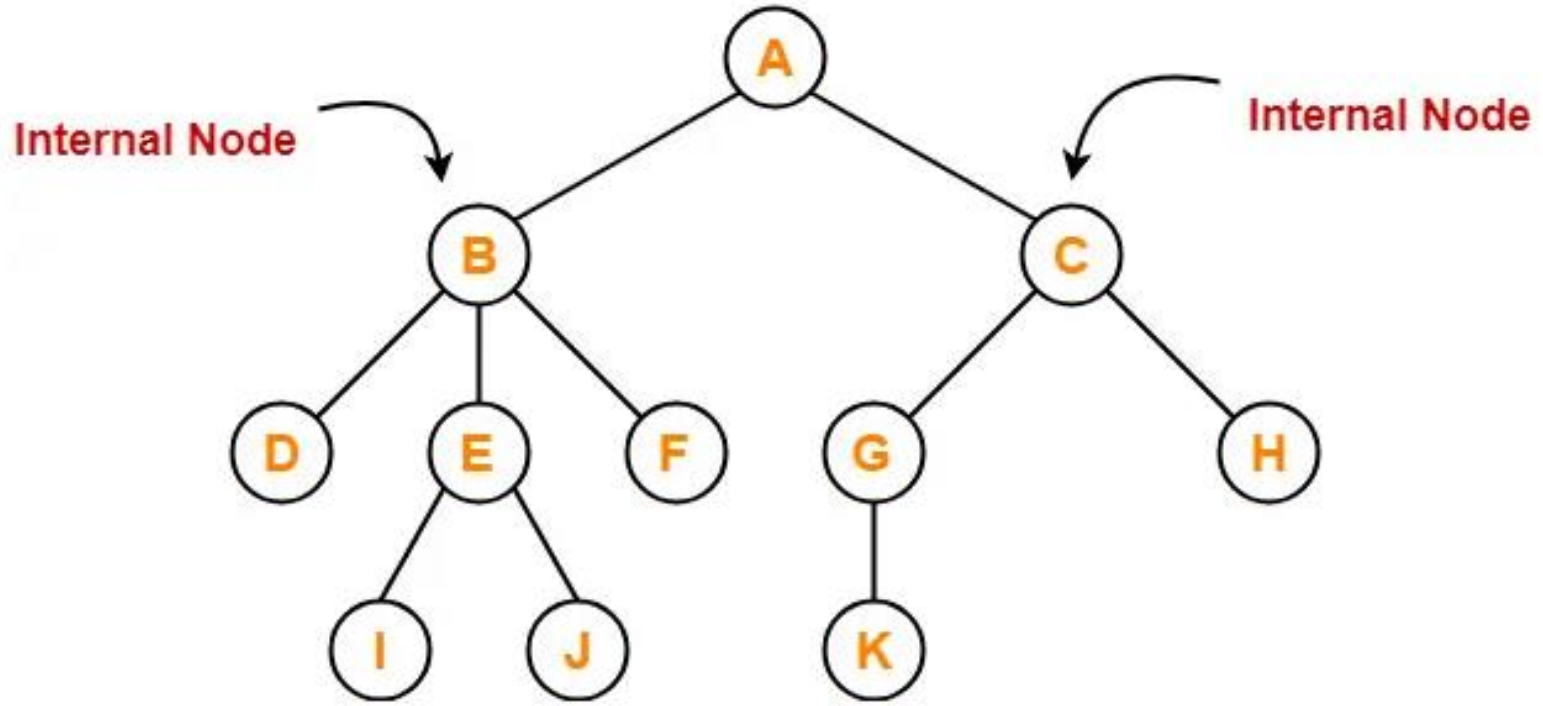
# Tree Terminology- Degree

- Degree of a node is the total number of children of that node.
- Degree of a tree is the highest degree of a node among all the nodes in the tree.



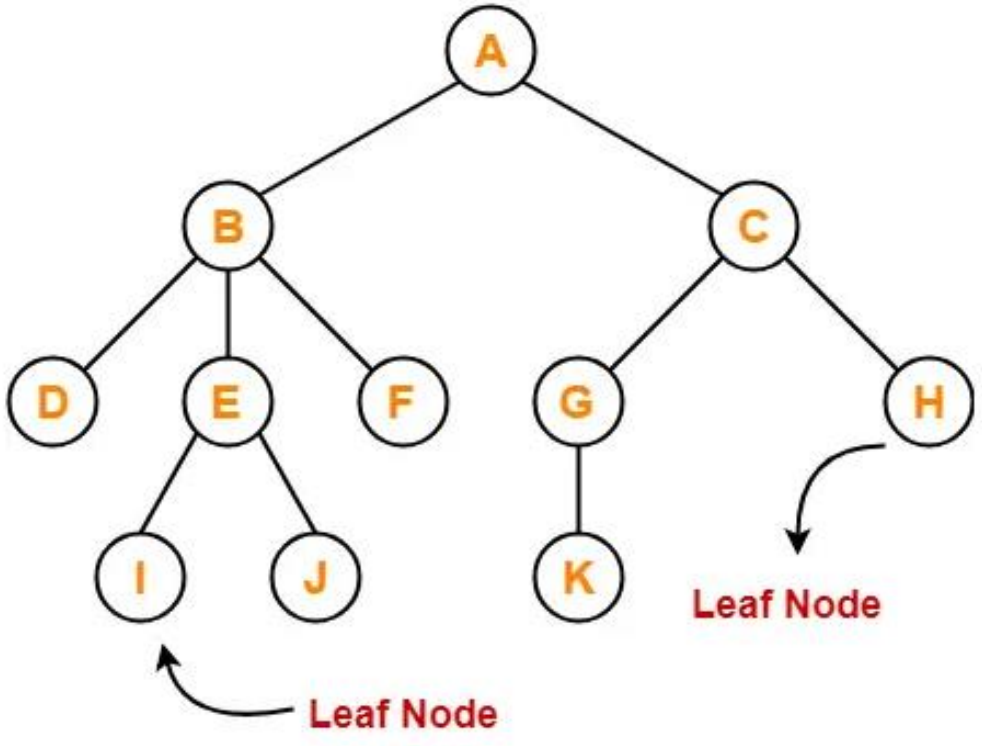
# Tree Terminology - Internal Node

- The node which has at least one child



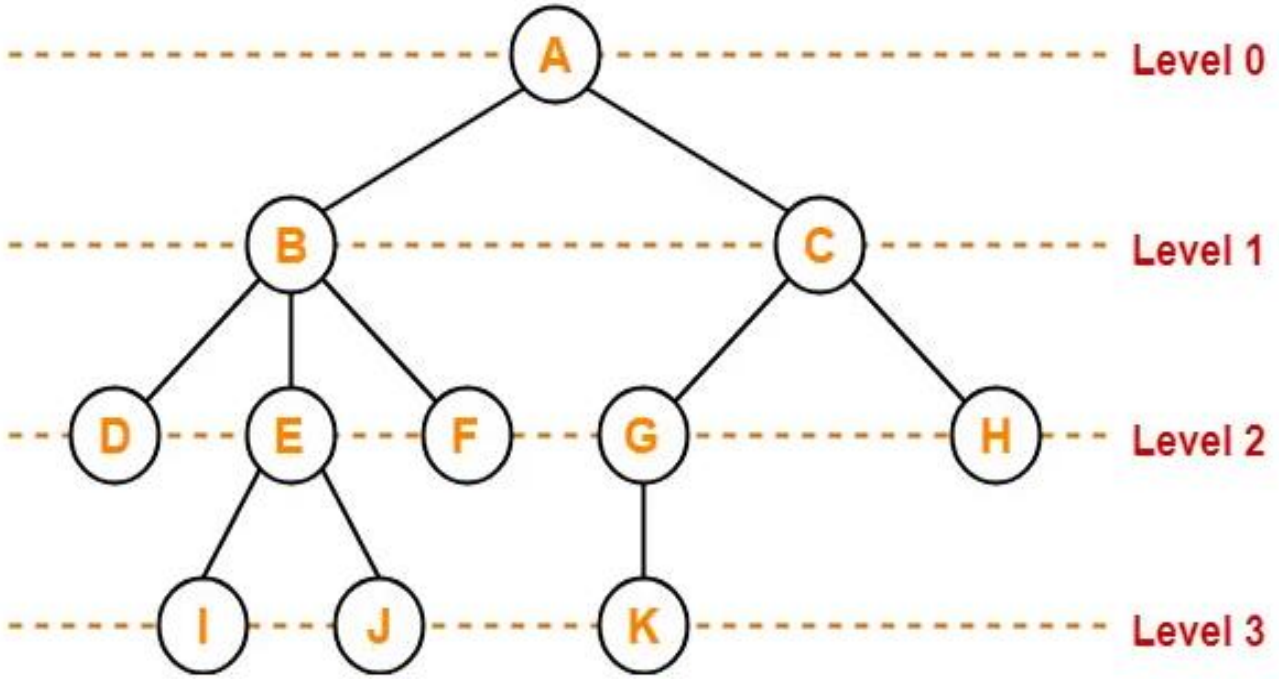
# Tree Terminology - Leaf Node

- The node which does not have any child
- Leaf nodes are also called as external nodes or terminal nodes.



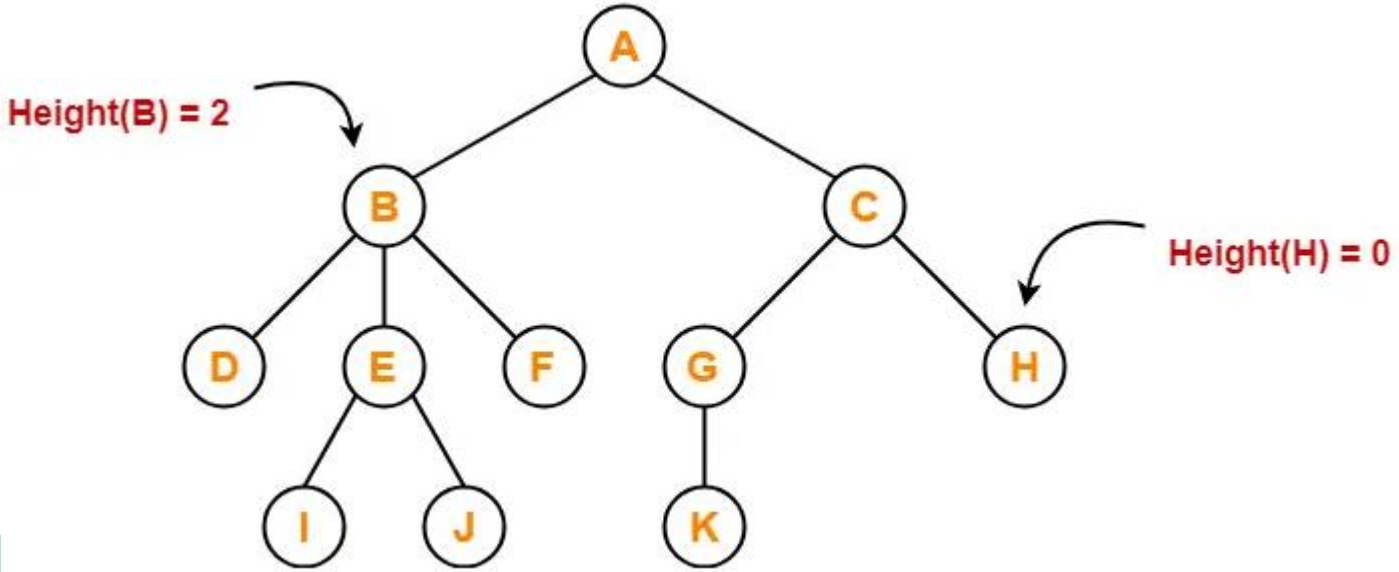
# Tree Terminology - Level

- In a tree, each step from top to bottom
- The level count starts with 0 and increments by 1 at each level or step.



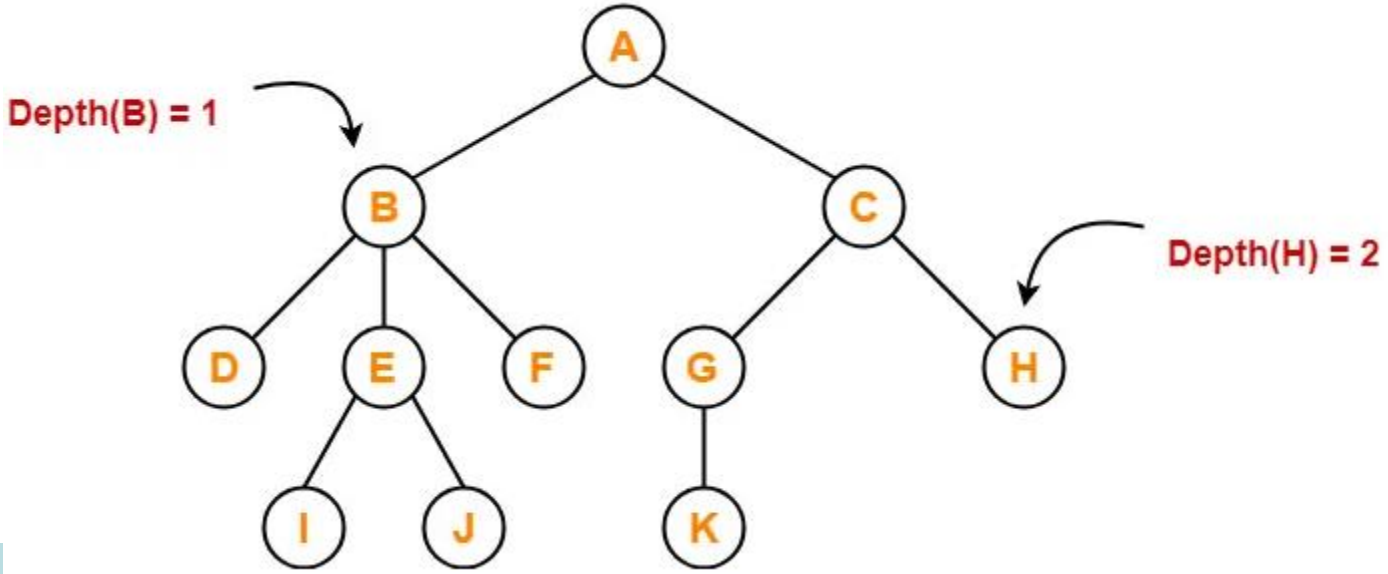
# Tree Terminology - Height

- Total number of edges that lies on the longest path from any leaf node to a particular node is called as height of that node.
- Height of a tree is the height of root node.
- Height of all leaf nodes = 0



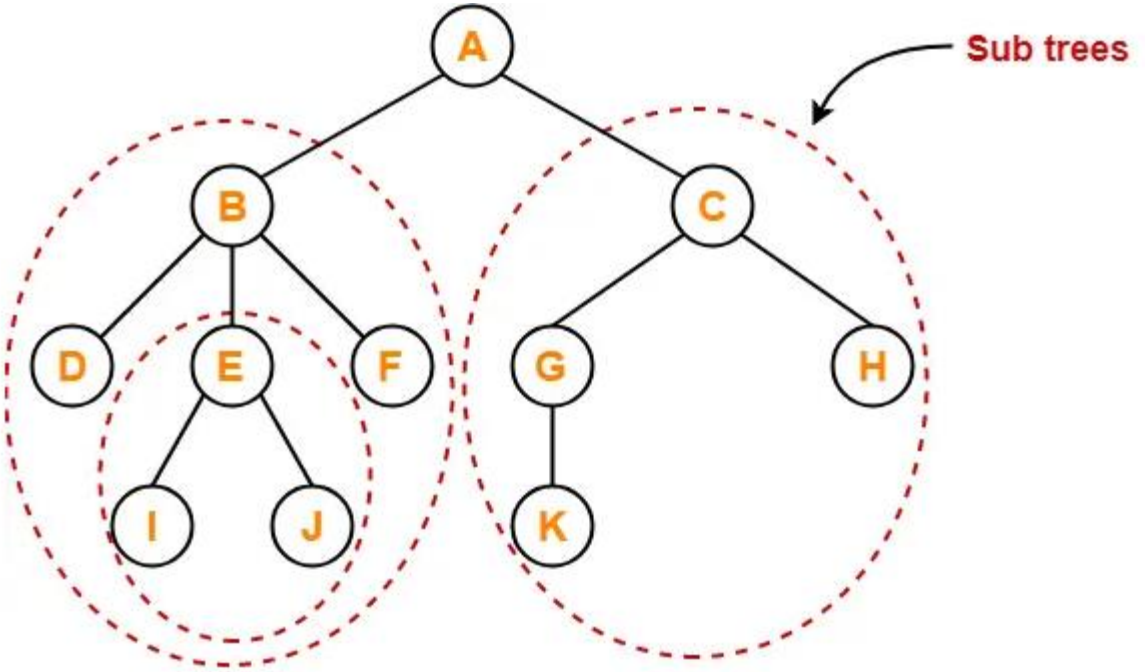
# Tree Terminology - Depth

- Total number of edges from root node to a particular node is called as depth of that node.
- Depth of a tree is the total number of edges from root node to a leaf node in the longest path.
- Depth of the root node = 0



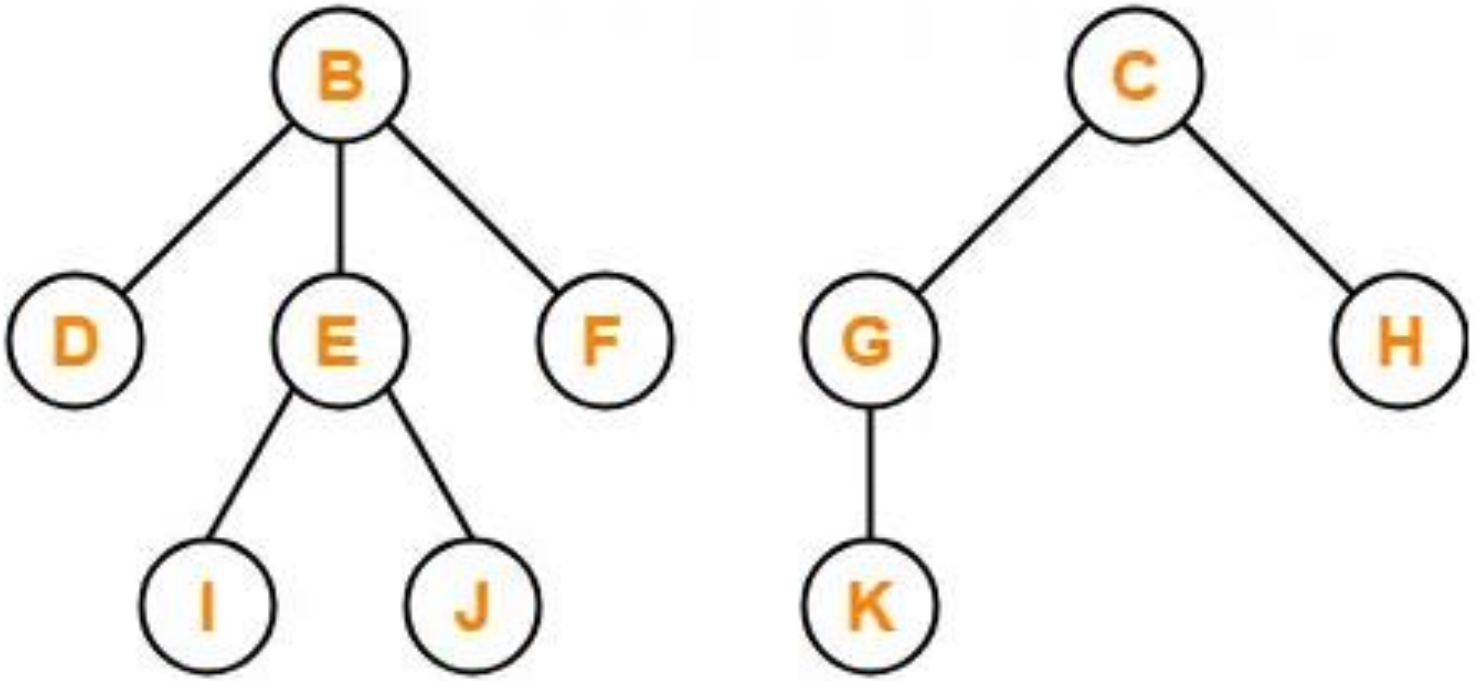
# Tree Terminology - Subtree

- In a tree, each child from a node forms a subtree recursively.
- Every child node forms a subtree on its parent node.



# Tree Terminology - Forest

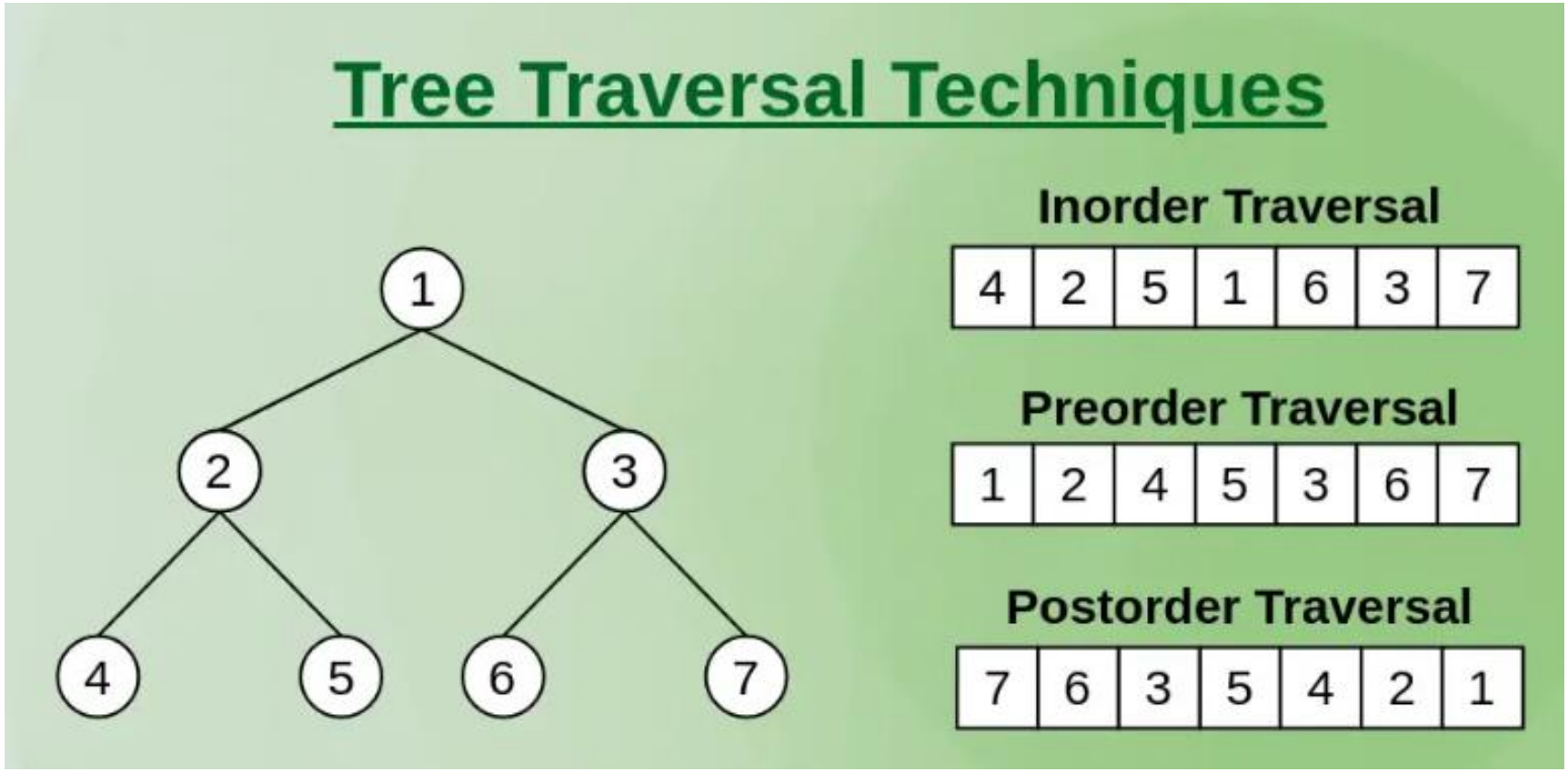
- A forest is a set of disjoint trees.



# Tree Traversal

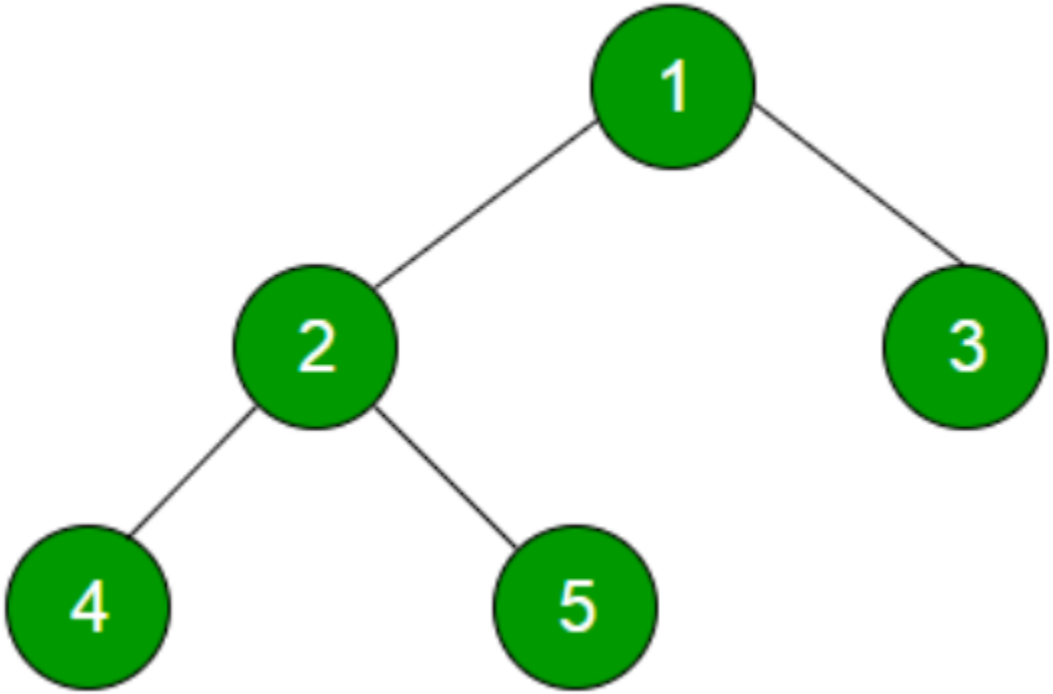
- Depth First Search (DFS)
  - Inorder Traversal
  - Preorder Traversal
  - Postorder Traversal
- Level Order Traversal or Breadth First Search or BFS
- Boundary Traversal
- Diagonal Traversal

# Tree Traversal -



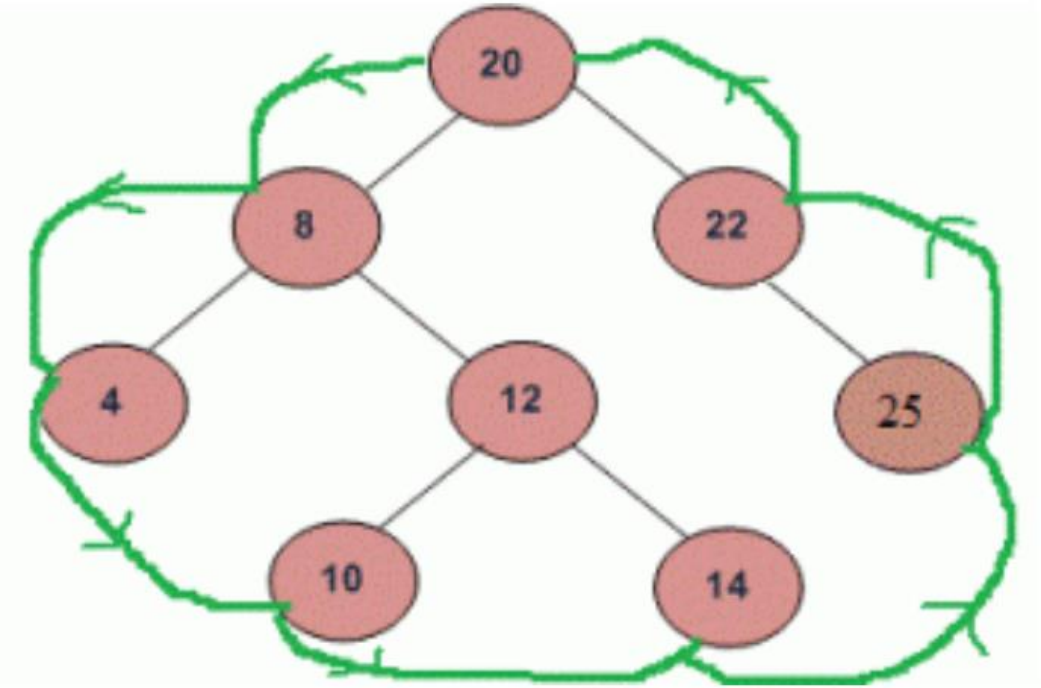
# Tree Traversal - Level Order Traversal

- 1
- 2 3
- 4 5



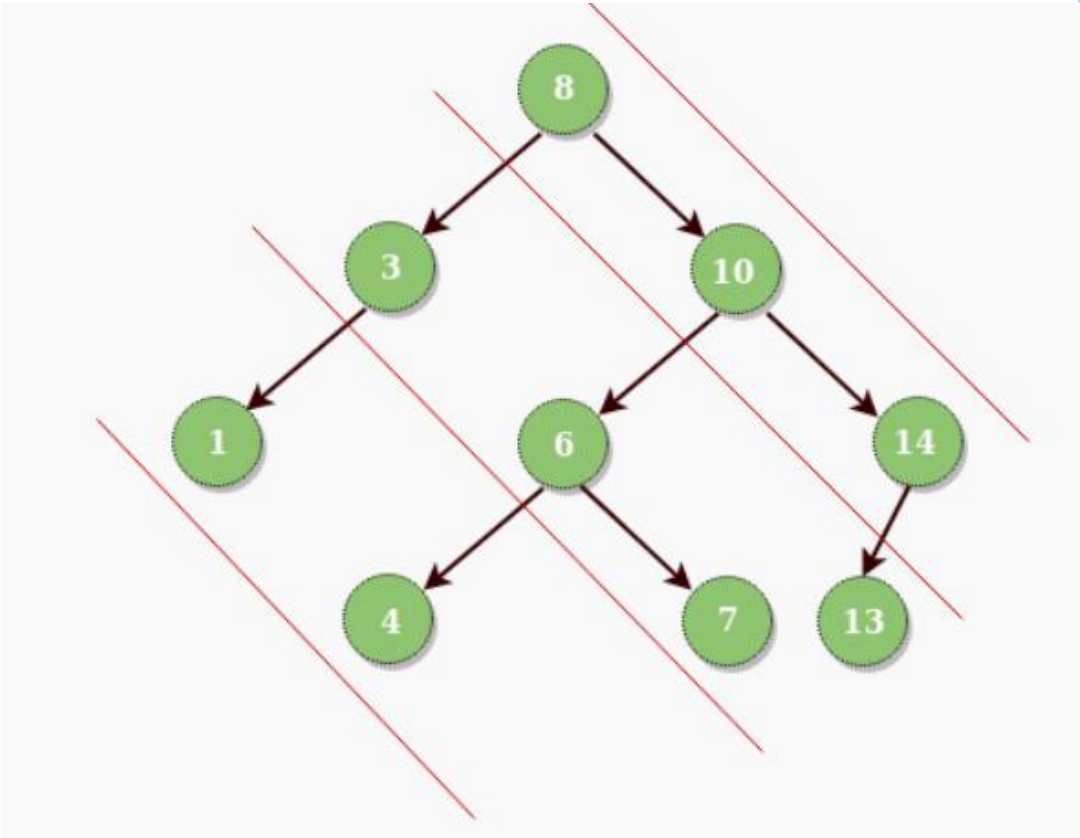
# Tree Traversal - Boundary Traversal

- left boundary (nodes on left excluding leaf nodes)
- leaves (consist of only the leaf nodes)
- right boundary (nodes on right excluding leaf nodes)



# Tree Traversal - Diagonal Traversal

- all the nodes in a single diagonal will be printed one by one.
  - 8 10 14
  - 3 6 7 13
  - 1 4



# W5 – Lab

# Struct node of Tree

```
1   // Binary Tree in C++
2   #include <stdlib.h>
3   #include <iostream>
4   using namespace std;
5   struct node
6   {
7     int data;
8     struct node *left;
9     struct node *right;
10  };
```

# Struct node of Tree

```
11  // New node creation
12  struct node *newNode(int data)
13  {
14      struct node *node = (struct node *)malloc(sizeof(struct node));
15      node->data = data;
16      node->left = NULL;
17      node->right = NULL;
18      return (node);
19  }
```

# Struct node of Tree

```
20  // Traverse Inorder
21  void traverseInOrder(struct node *temp)
22  {
23    if (temp != NULL)
24    {
25      traverseInOrder(temp->left);
26      cout << " " << temp->data;
27      traverseInOrder(temp->right);
28    }
29  }
```

# Struct node of Tree

```
30  // Traverse Preorder
31  void traversePreOrder(struct node *temp)
32  {
33    if (temp != NULL)
34    {
35      cout << " " << temp->data;
36      traversePreOrder(temp->left);
37      traversePreOrder(temp->right);
38    }
39  }
```

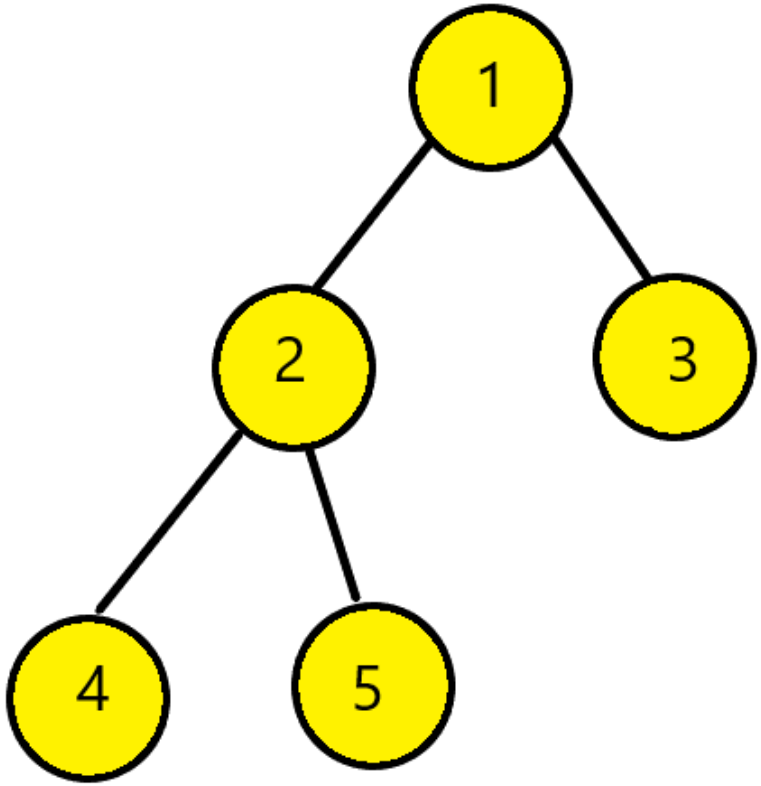
# Struct node of Tree

```
40  // Traverse Postorder
41  void traversePostOrder(struct node *temp)
42  {
43    if (temp != NULL)
44    {
45      traversePostOrder(temp->left);
46      traversePostOrder(temp->right);
47      cout << " " << temp->data;
48    }
49  }
```

# Ex. 1 – Tree 1

Execute in Main() trees in the figure below and access the tree with three traversal algorithms

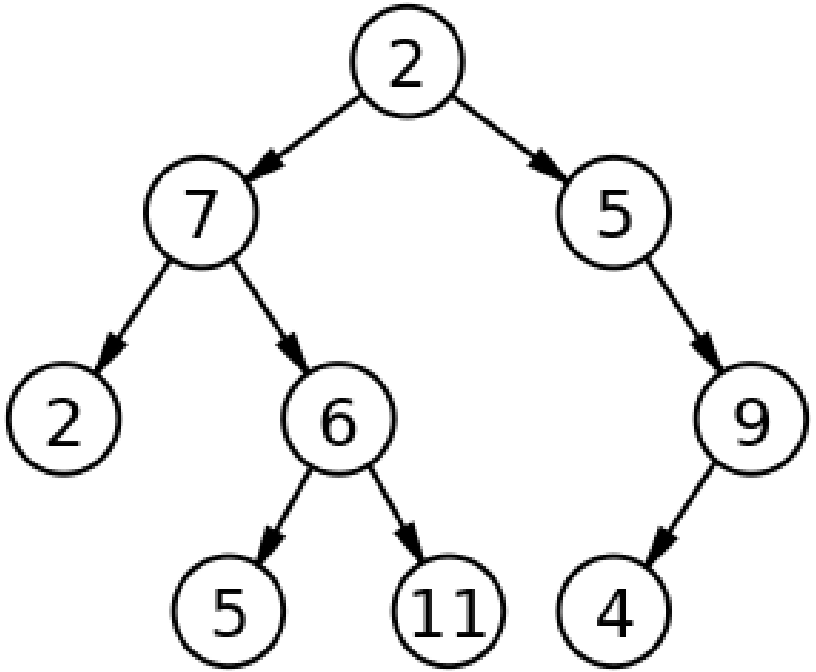
- Inorder Traversal
- Preorder Traversal
- Postorder Traversal



# Ex. 2 – Tree 2

Execute in Main() trees in the figure below and access the tree with three traversal algorithms

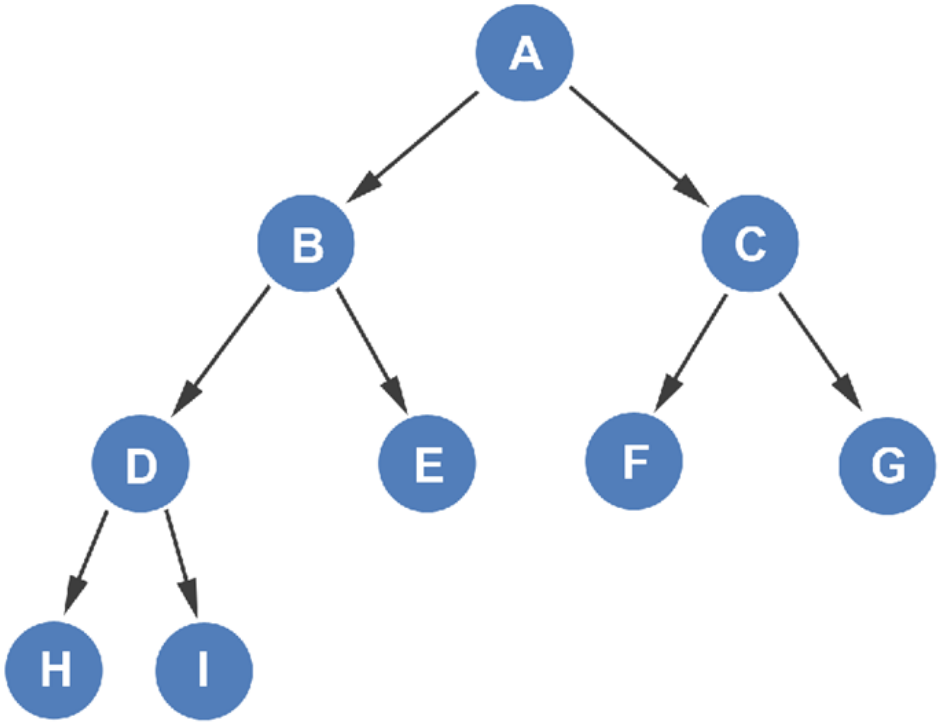
- Inorder Traversal
- Preorder Traversal
- Postorder Traversal



# Ex. 3 – Tree 3

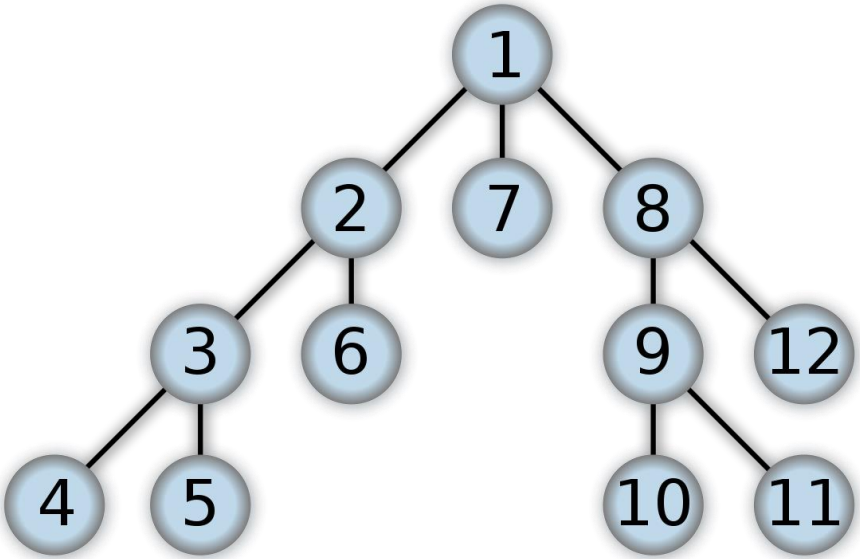
Execute in Main() trees in the figure below and access the tree with three traversal algorithms

- Inorder Traversal
- Preorder Traversal
- Postorder Traversal



# Ex. 4 – Teamwork

1. Execute in Main() trees in the figure below and access the tree with three traversal algorithms
   - Inorder Traversal
   - Preorder Traversal
   - Postorder Traversal




2. Analysis of tree step of each function in tree

# Thanks!