Data Structures

By
Siddharth.T

Why recursion?

- ➢ Recursive thinking is really important in programming and it helps you break down big problems into smaller ones and easier to use.

When to choose recursion

- ➢ If you can divide the problem into smaller subproblems.
- ➢ It is especially good for working on things that have many possible branches and are too complex for an iterative approach.
- ➢ Prominent usage of recursion in data structure like trees and graphs.
- ➢ It is used in many algorithms like (divide and conquer, greedy and dynamic programming.

Recursion

- ➢ Recursion is a widely used phenomenon in computer science used to solve complex problems by breaking them down into simpler ones. Recursion is a way of solving problem by having a function calling itself.
- ➢ Performing the same operation multiple times with different inputs
- ➢ In every step we try smaller inputs to make the problem smaller
- ➢ Base condition is needed to stop the recursion or infinite loop will occur.

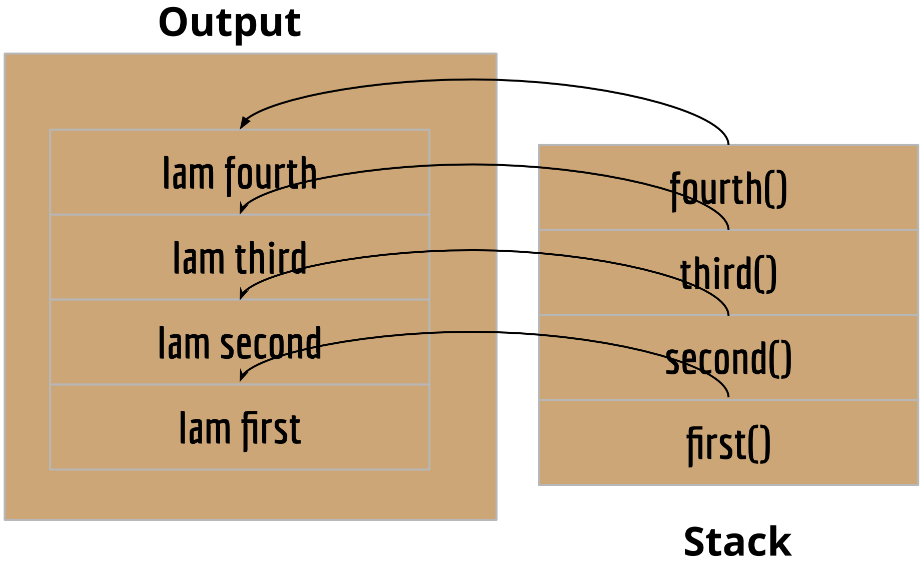
Parts of a Recursive Algorithm

All recursive algorithms must have the following:

1. Base Case (i.e., when to stop)
2. Work toward Base Case
3. Recursive Call (i.e., call itselves)

```
def first():
  second()
  print("Iam first")
def second():
  third()
  print("Iam first")
def third():
  fourth()
  print("Iam third")
def fourth():
  print("iam fourth")


first()
```



Output

Iam fourth

Iam third

Iam second

Iam first

fourth()

third()

second()

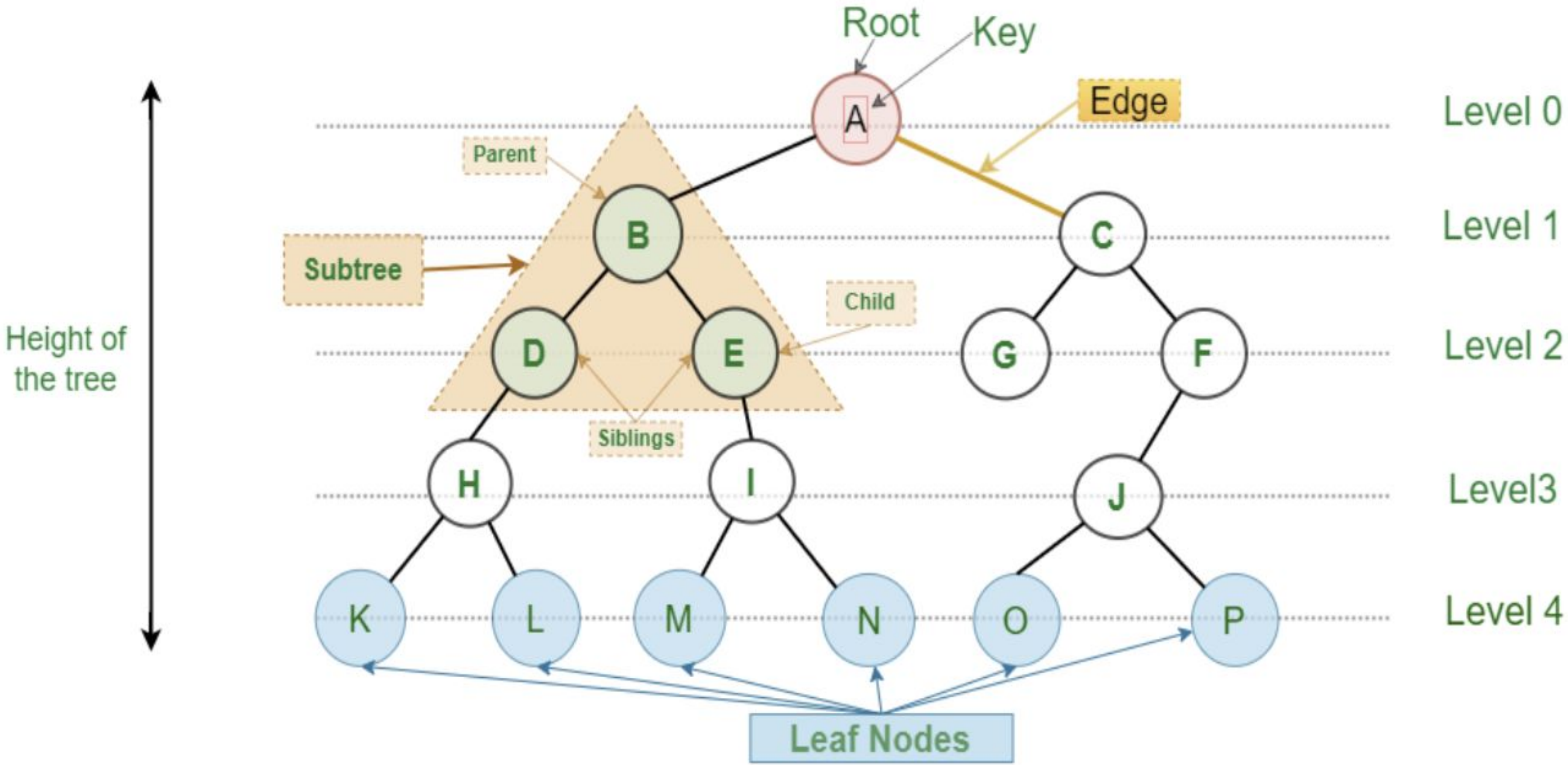
first()

Stack

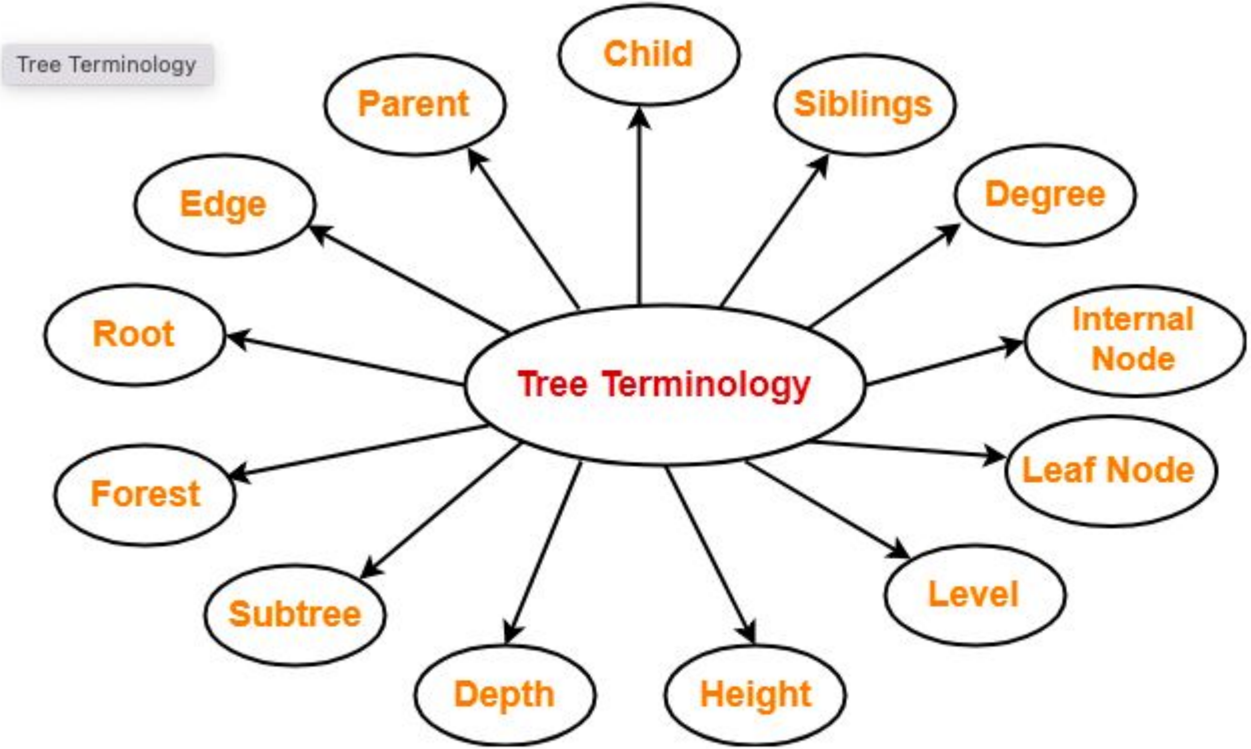
Tree

- Tree is a non-linear data structure (organises data non sequentially). It is a collection of entities called nodes connected by edges.
- Tree is also called as hierarchical data structure
- It is unidirectional
- Represents the relationship between nodes
- Sub node are called childrens and all nodes are connected by edges
- The last nodes are called as leaves or terminal nodes or external nodes
- Nodes which belong to same nodes are called siblings

Tree Data Structure



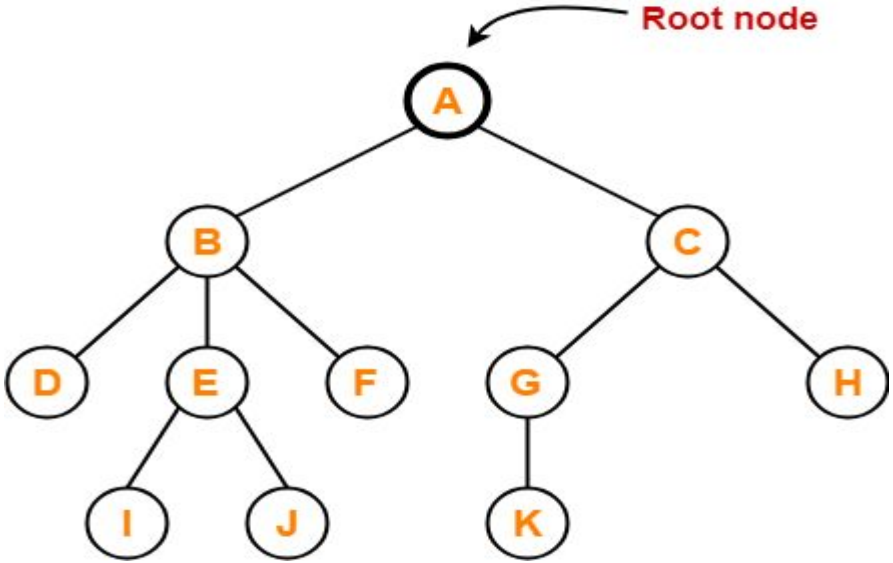
Terminologies



https://www.gatevidyalay.com/tree-data-structure-tree-terminology/#:~:text=Tree%20data%20structure%20is%20a,terms%20are%20discussed%20with%20examples.

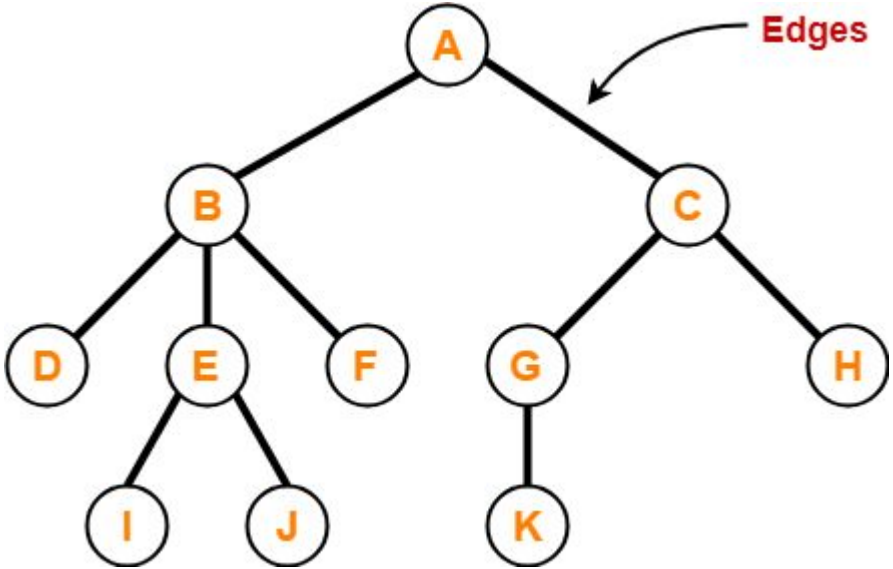
Root

- 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.



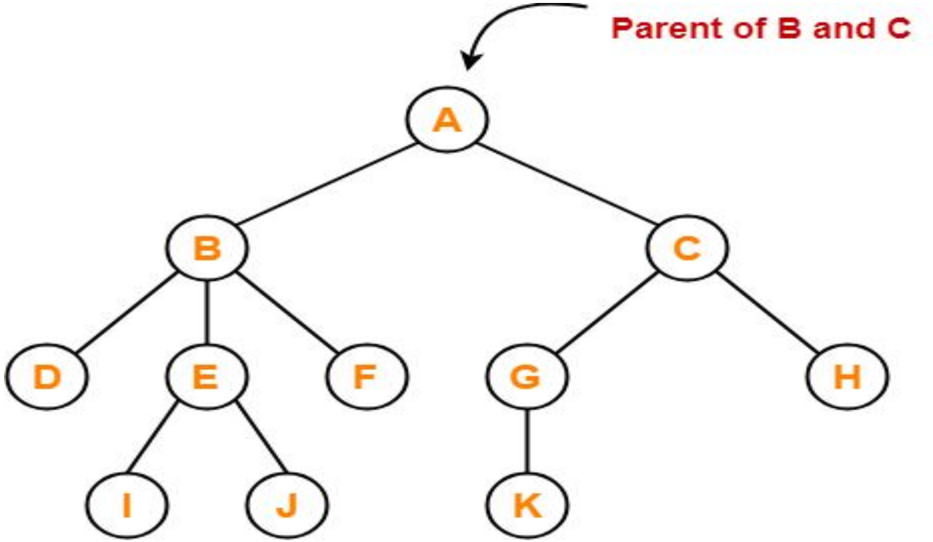
Edge

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



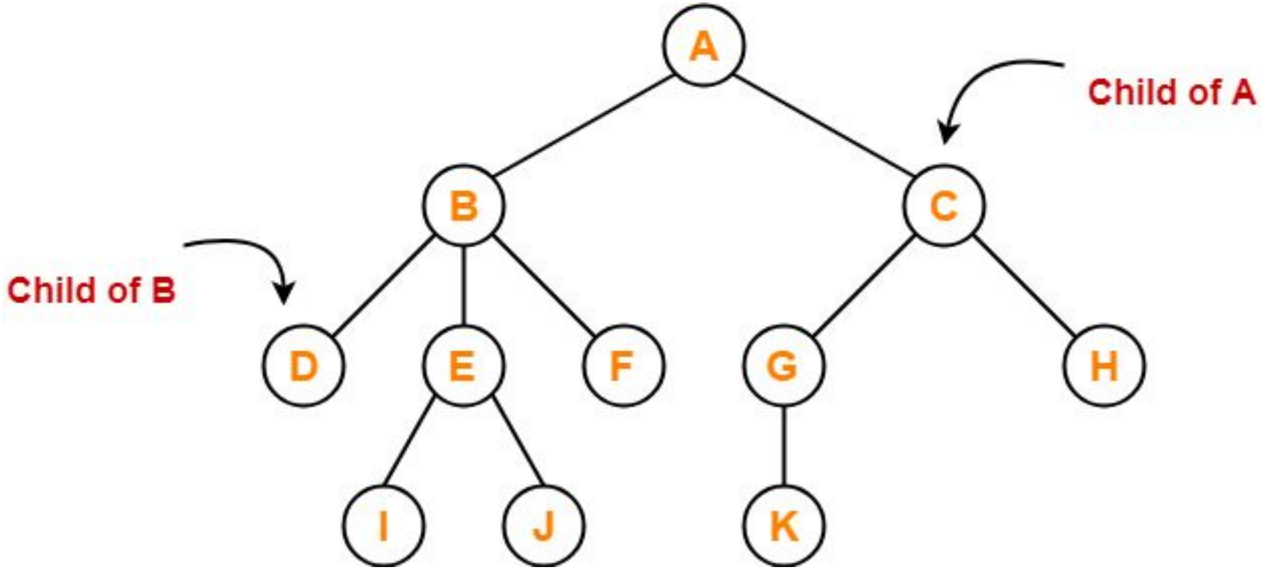
Parent

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



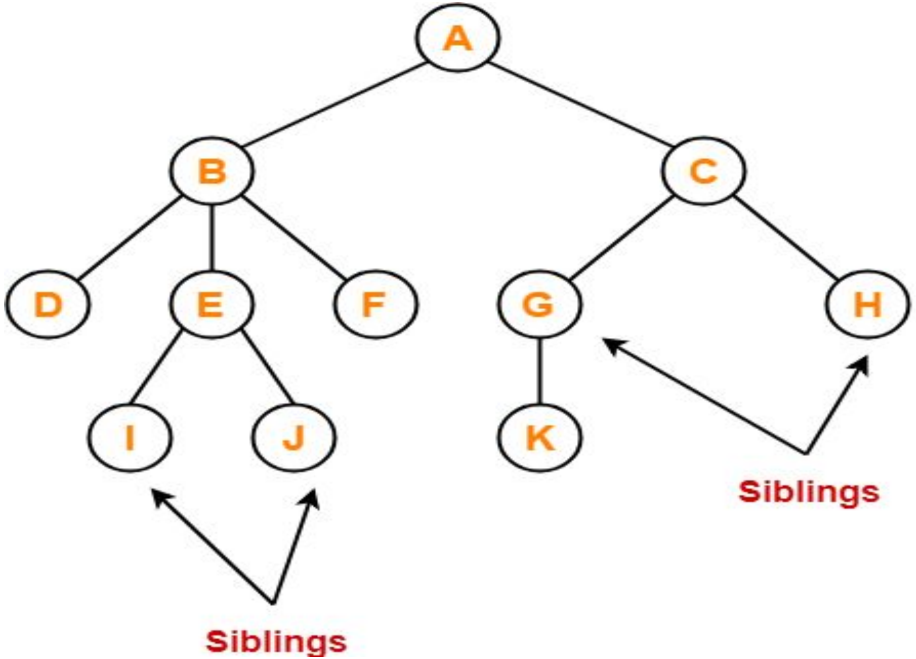
Child

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



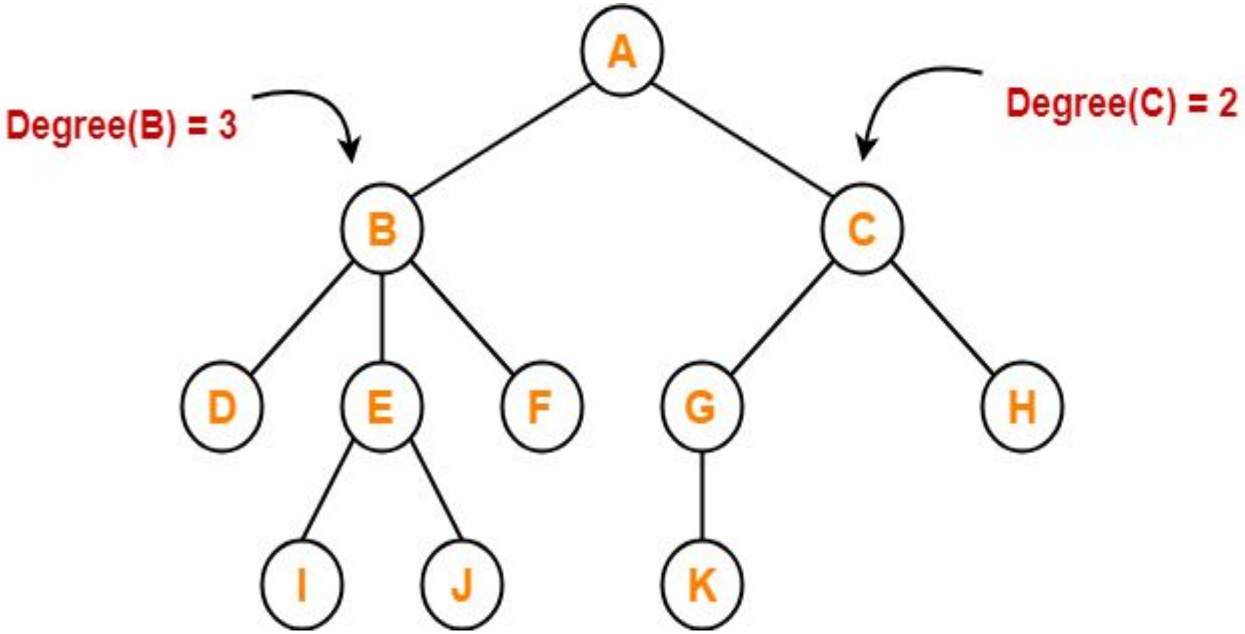
Siblings

- Nodes which belong to the same parent are called as siblings.
- In other words, nodes with the same parent are sibling nodes.



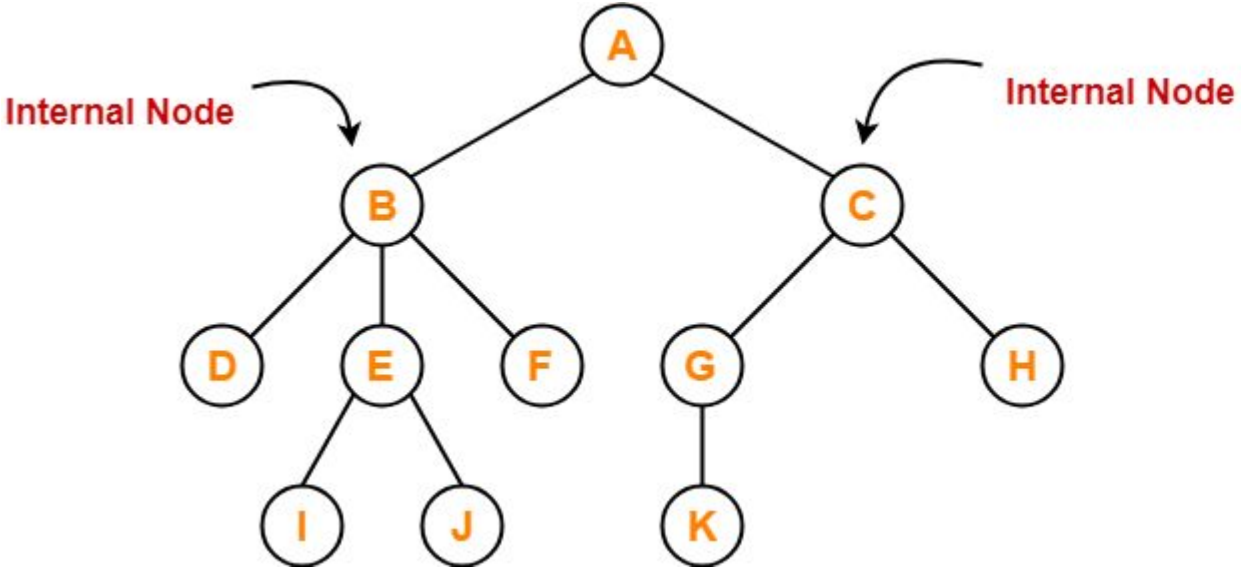
Degree

- 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.



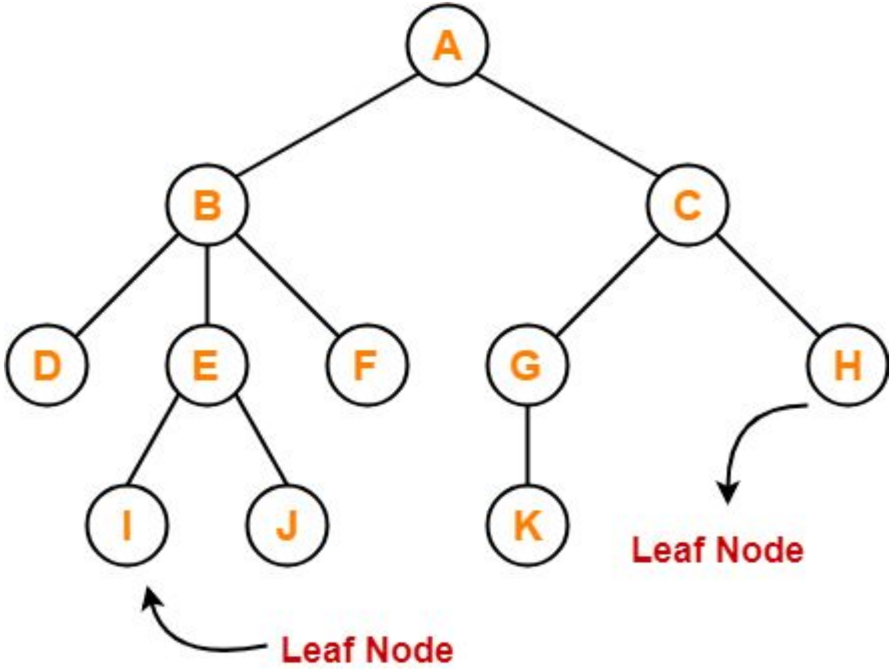
Internal Node

- The node which has at least one child is called as an internal node.
- Internal nodes are also called as non-terminal nodes.
- Every non-leaf node is an internal node.



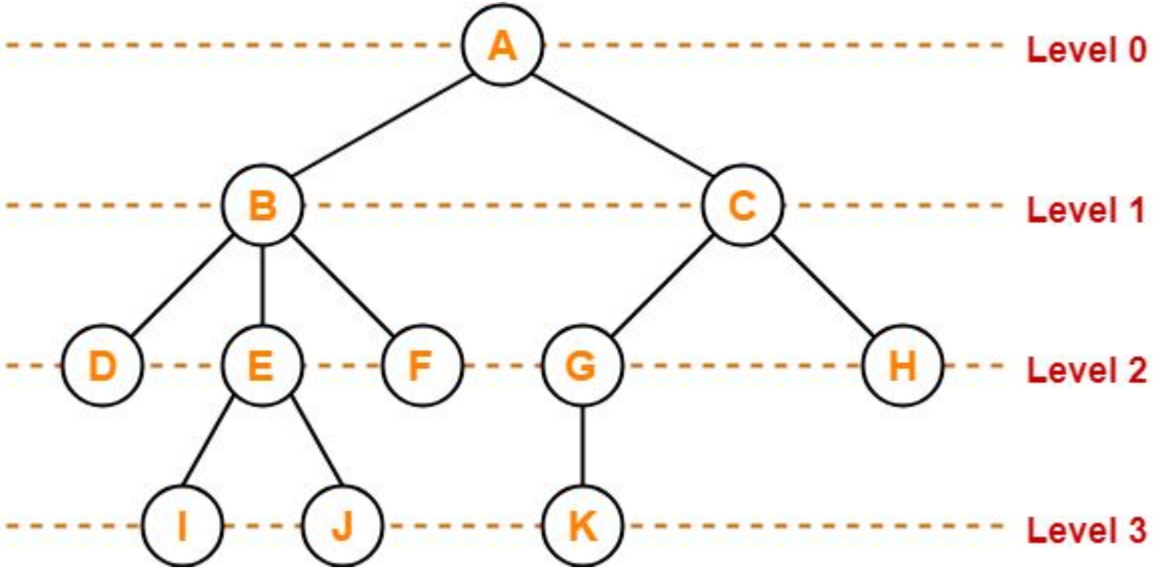
Leaf Node

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



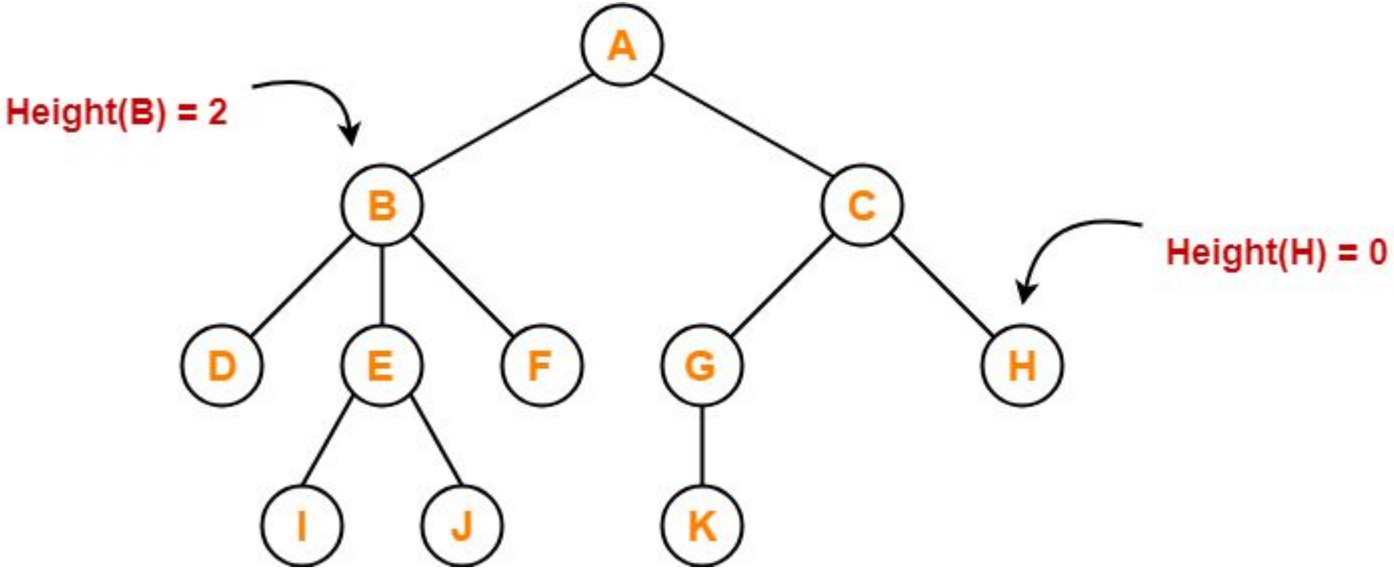
Level

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



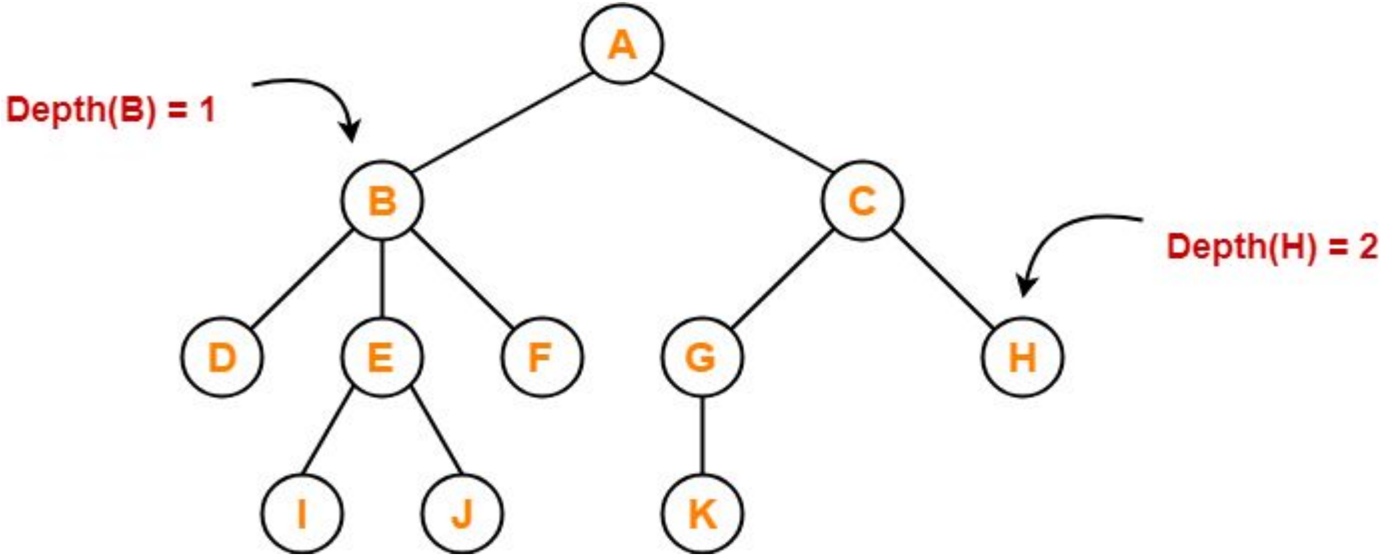
Height

- 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



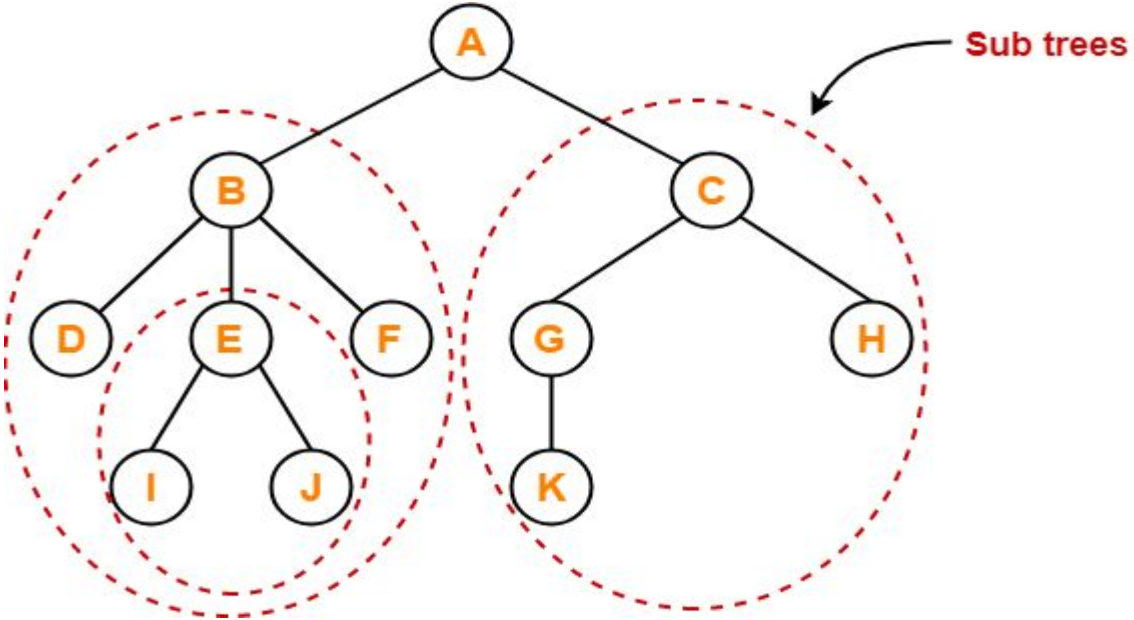
Depth

- 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
- The terms "level" and "depth" are used interchangeably.



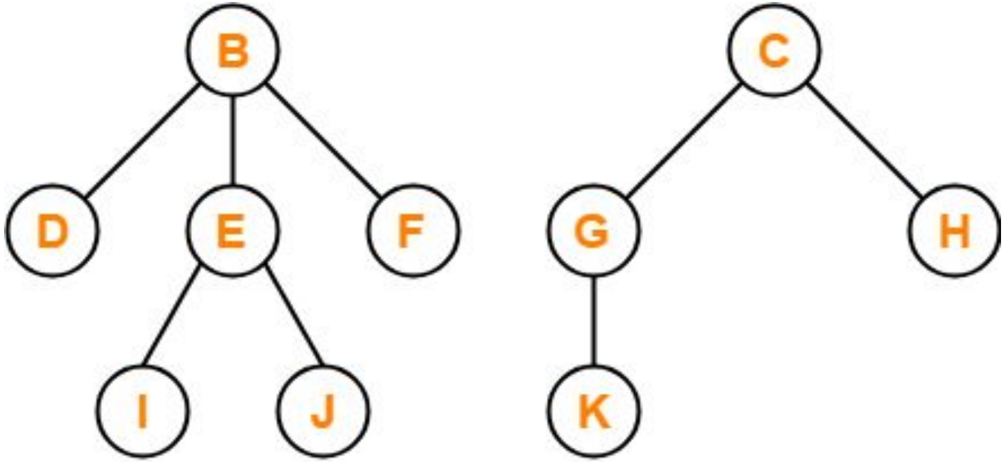
Subtree

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



Forest

A forest is a set of disjoint trees.



Forest

Types of Tree

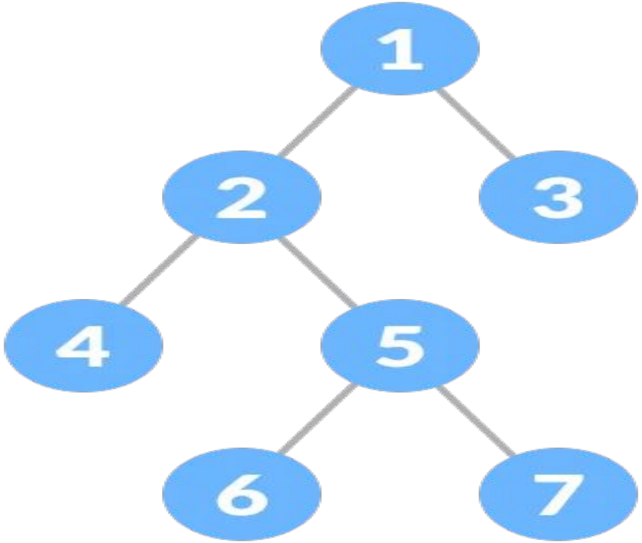
1. Binary Tree
2. Binary Search Tree
3. AVL Tree
4. B-Tree

Types

- Full binary tree
- Complete binary tree
- Perfect binary tree
- Balanced binary tree
- Pathological binary tree

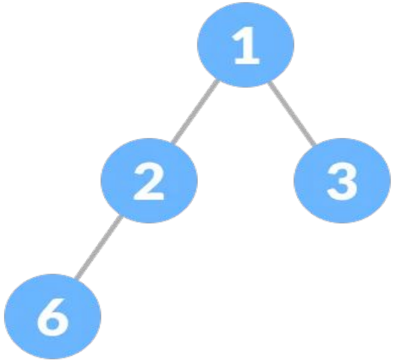
Full Binary tree

- A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children.
- It is also known as a proper binary tree.

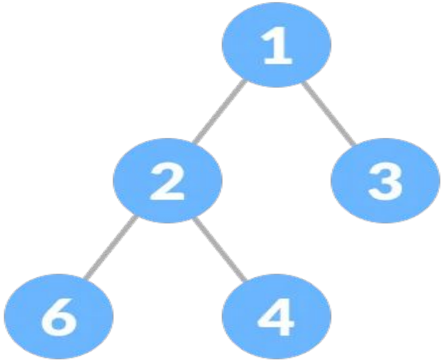


Complete Binary tree

In a complete binary tree, all the levels of a tree are filled entirely except the last level. In the last level, nodes might or might not be filled fully. Also, let's note that all the nodes should be filled from the left.



✗ Full Binary Tree
✓ Complete Binary Tree

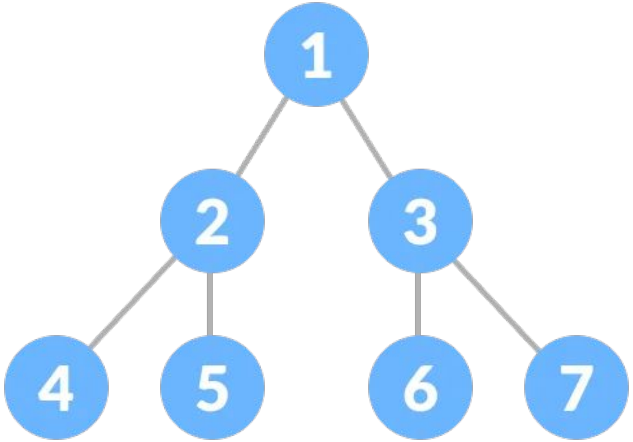


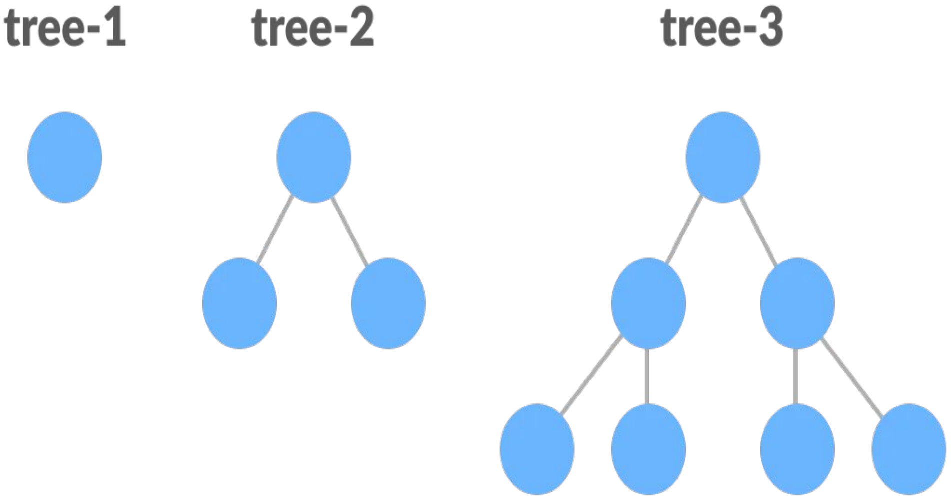
✓ Full Binary Tree
✓ Complete Binary Tree

Perfect binary tree

A perfect binary tree is a type of binary tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.

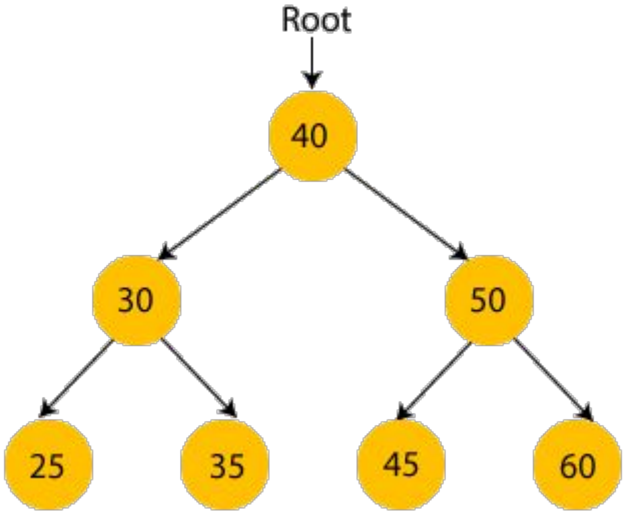
1. If a single node has no children, it is a perfect binary tree of height $h = 0$,
2. If a node has $h > 0$, it is a perfect binary tree if both of its subtrees are of height $h - 1$ and are non-overlapping





Binary search tree

- A binary search tree follows some order to arrange the elements.
- In a Binary search tree, the value of left node must be smaller than the parent node, and the value of right node must be greater than the parent node.
- This rule is applied recursively to the left and right subtrees of the root.



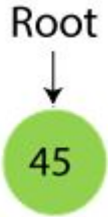
Advantages of Binary search tree

- Binary Search Trees(BSTs) are used to quickly check whether an element is present in a set or not.
- Binary trees can be used to store data in a database system, with each node representing a record. This allows for efficient search operations and enables the database system to handle large amounts of data.
- A modified version of a tree called Tries is used in modern routers to store routing information.
- Binary trees can be used to implement decision trees, a type of machine learning algorithm used for classification and regression analysis.
- Compilers use a syntax tree to validate the syntax of every program you write.

Suppose the data elements are - 45, 15, 79, 90, 10, 55, 12, 20, 50

Step:1

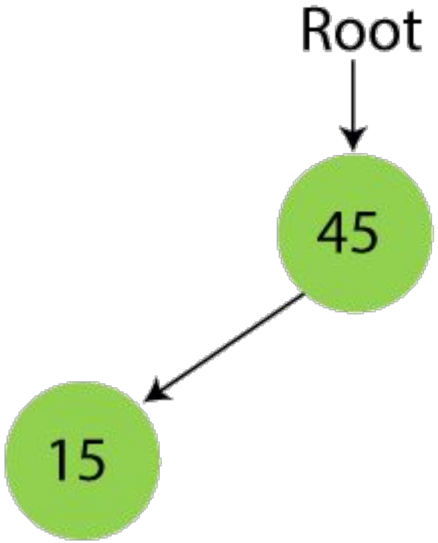
Insert 45 as root node



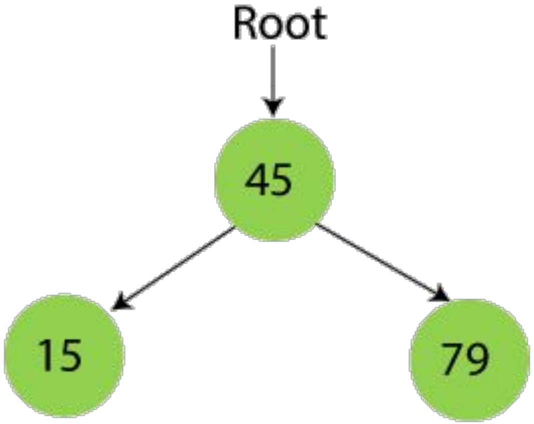
Step 2 - Insert 15.

As 15 is smaller than 45, so

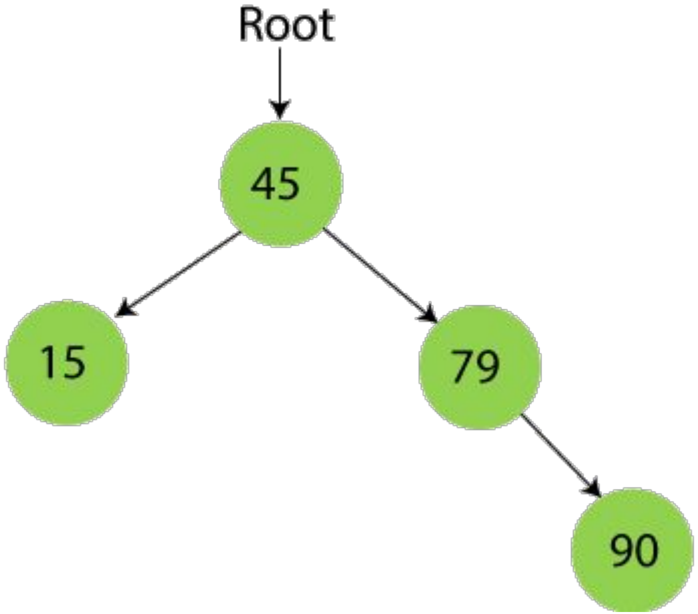
insert it as the root node of the left subtree.



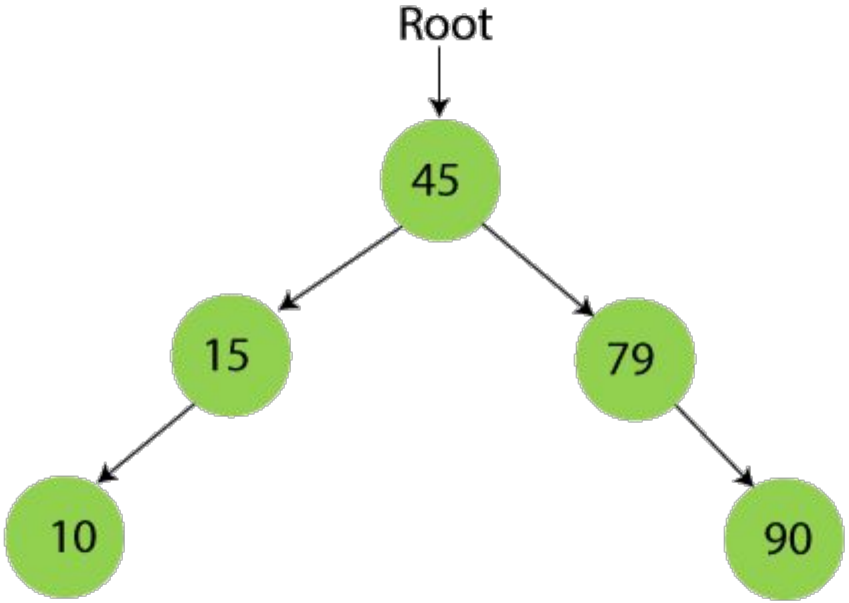
Step 3 - Insert 79.



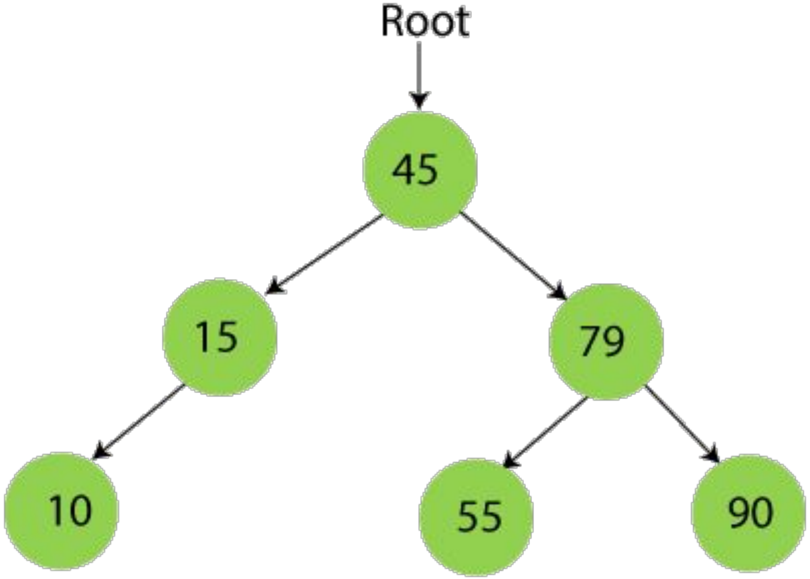
Step 4 - Insert 90.



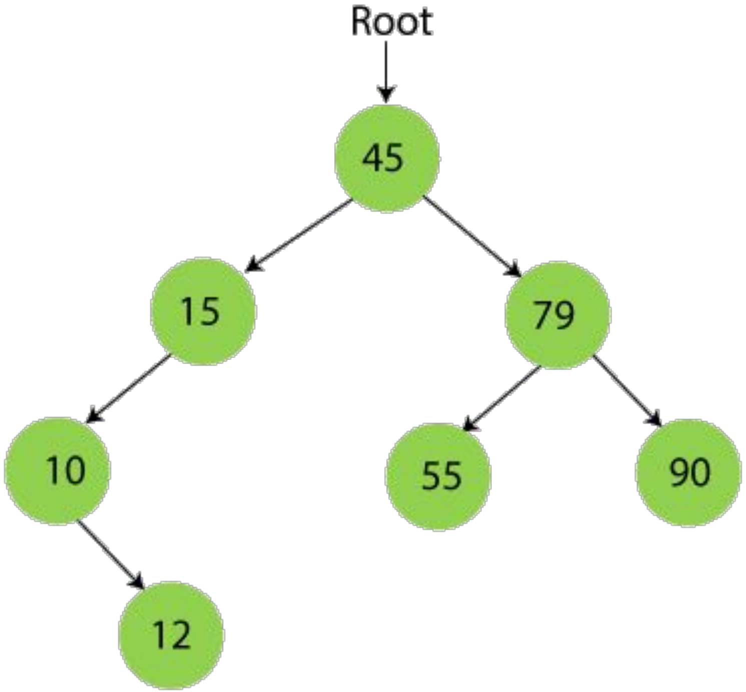
Step 5 - Insert 10.



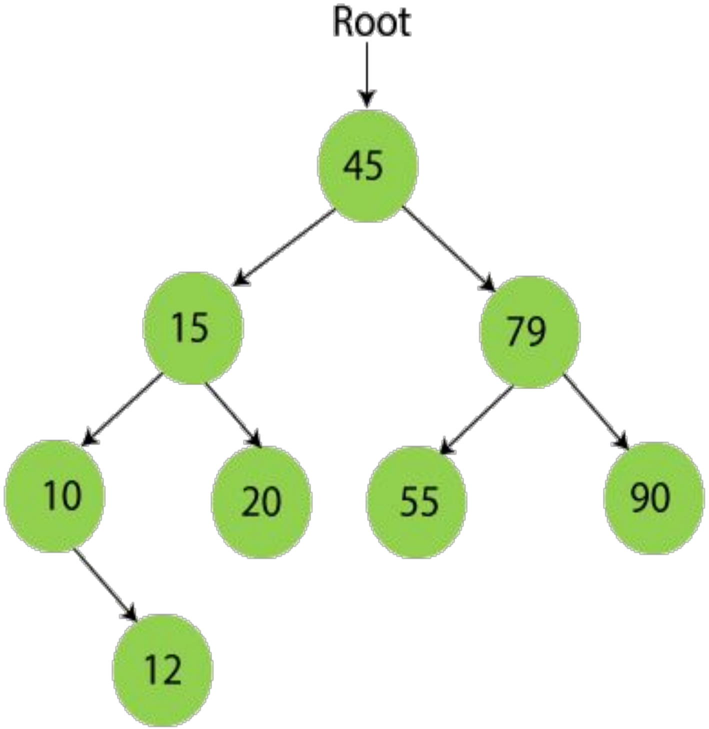
Step 6 - Insert 55.



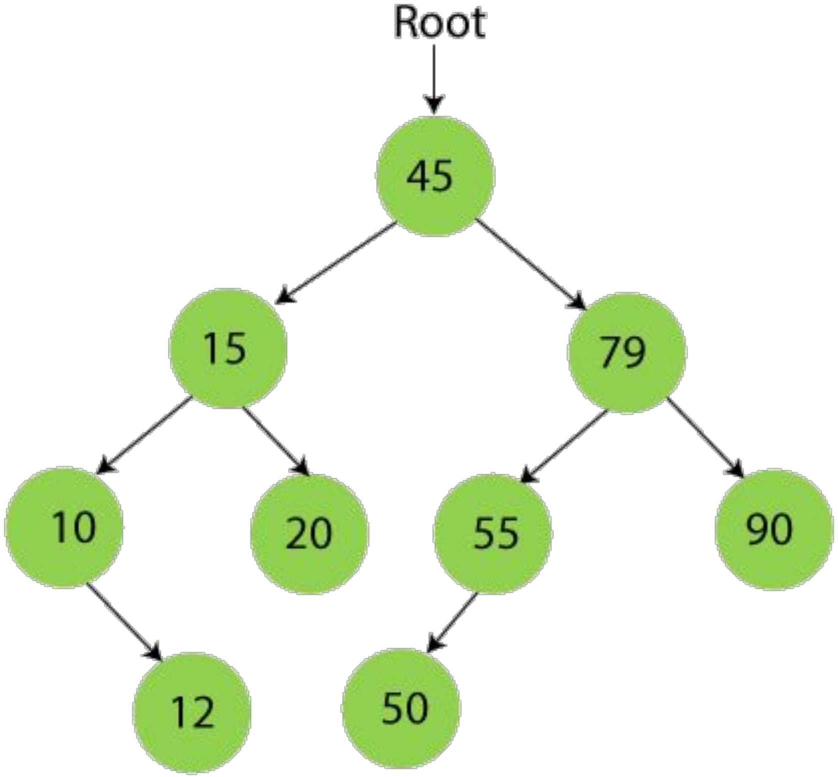
Step 7 - Insert 12.



Step 8 - Insert 20.

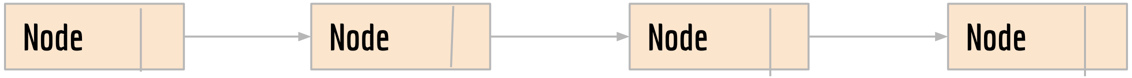


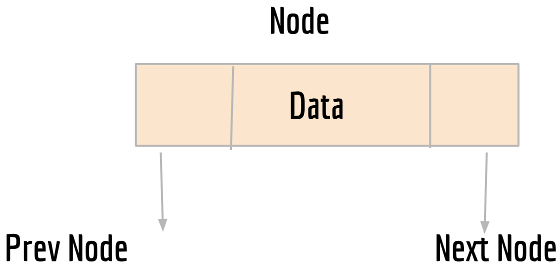
Step 9 - Insert 50.



Linked List

Linked list is a linear data structure, which is made up of chain of nodes, were each nodes consist of data and memory address of another node.





Terms of linked list

Node ⟶ The element in linked list is called node, each element contains data and reference of next node.

Head ⟶ The first element of linked list is called head.

tail/end ⟶ The last node of linked list.

➢ Linked list is a dynamic data structure, its size is not fixed, it can be modified based on number of elements.

Advantages

- Its a dynamic data structure, so its size don’t want to be mentioned.
- Inserted or removed easily.
- We can use linked list to implement stack, queue and graph.
- Used to represent and manipulate polynomials.
- In real life its used in ⟶ web browsers, Music player, image viewer

Disadvantages

- It need extra memory to store elements and reference.
- Accessing elements without the reference is not possible.

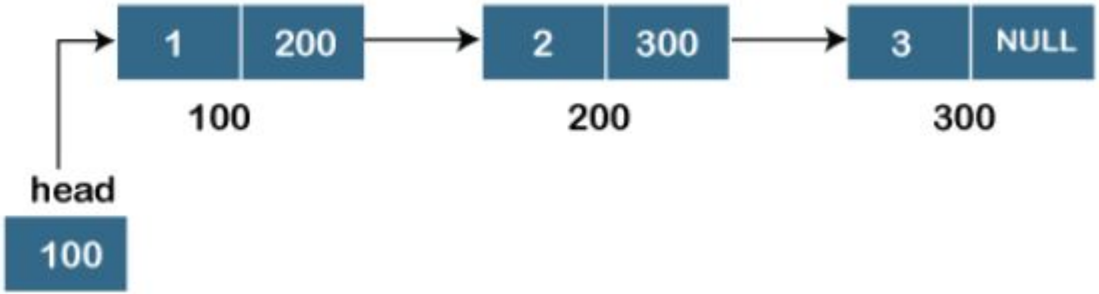
Types of linked list

The following are the types of linked list:

- Singly Linked list
- Doubly Linked list
- Circular Linked list
- Doubly Circular Linked list

Singly linked list

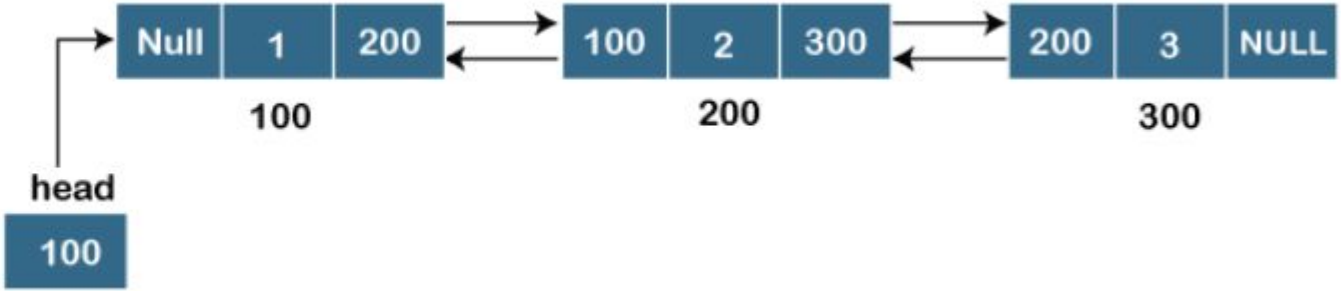
- The singly linked list is a data structure that contains two parts, i.e., one is the data part, and the other one is the address part, which contains the address of the next or the successor node.
- The address part in a node is also known as a pointer.



- In this list, only forward traversal is possible; we cannot traverse in the backward direction as it has only one link in the list.

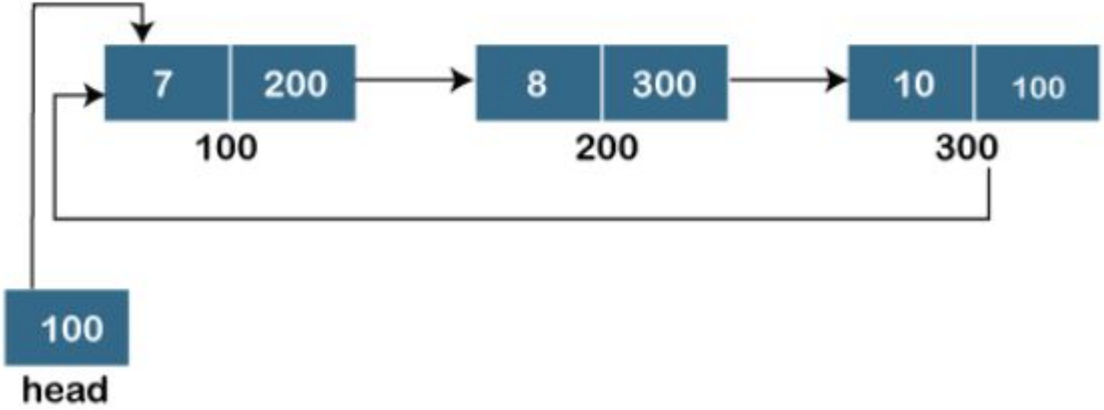
Doubly linked list

The doubly linked list contains two pointers. We can define the doubly linked list as a linear data structure with three parts: the data part and the other two address part(includes one data part, a pointer to its previous node, and a pointer to the next node).



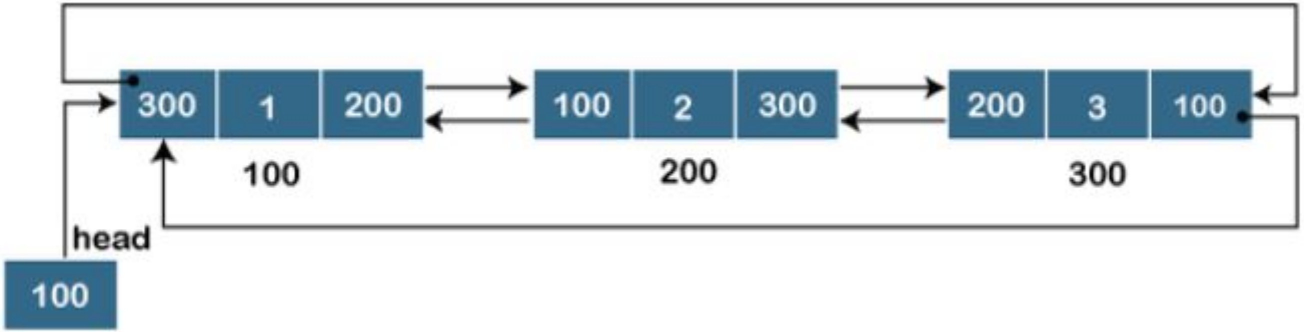
Circular linked list

- A circular linked list is a variation of a singly linked list.
- The only difference between the *singly linked list* and a *circular linked* list is that the last node does not point to any node in a singly linked list, so its link part contains a NULL value.
- On the other hand, the circular linked list is a list in which the last node connects to the first node, so the link part of the last node holds the first node's address. We can traverse in any direction.



Doubly Circular linked list

- The doubly circular linked list has the features of both the *circular linked list* and *doubly linked list*.
- The doubly circular linked list in which the last node is attached to the first node and thus creates a circle. It is a doubly linked list also because each node holds the address of the previous node also.



Graph

Graphs are those types of non-linear data structures which consist of a definite

Quantity of vertices/nodes and edges. The vertices or the nodes are involved in storing data and the edges show the vertices relationship.

2.)Nodes or vertices are data elements of graph

3.) Diff between Tree and Graph

Every tree is a graph, but all graph is not tree

Use of graph

# a.) GPS

# b.)Google Maps

# c.)Google search

# d.)Ecommerce for recommendations

# e.)To show the connection between the users

# f.)Frends suggestions.

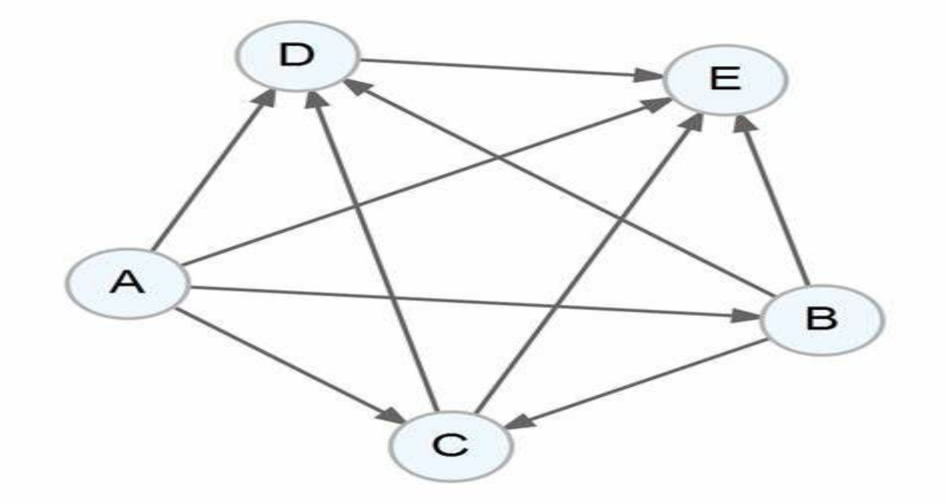
Types of graph

# 1.)Directed Graph(Edges are unidirectional)

# 2.)undirected Graph(All edges are bidirectional)

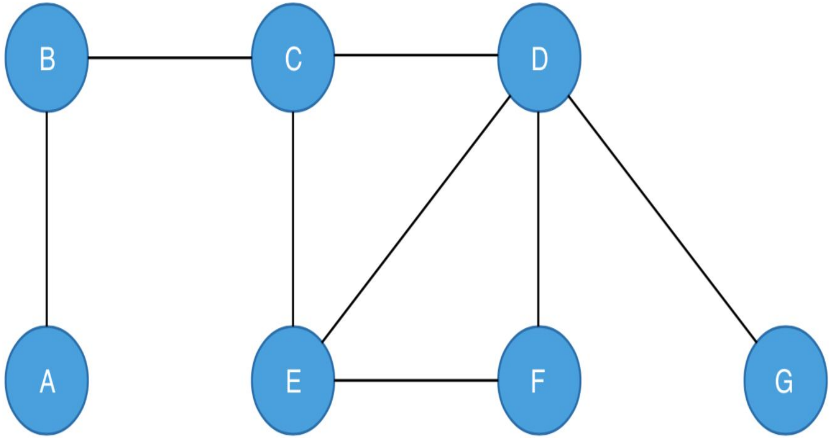
Directed Graph

A directed graph is graph, i.e., a set of objects (called vertices or nodes) that are connected together, where all the edges are directed from one vertex to another.



Undirected Graph

An undirected graph is graph, i.e., a set of objects (called vertices or nodes) that are connected together, where all the edges are bidirectional.



DFS

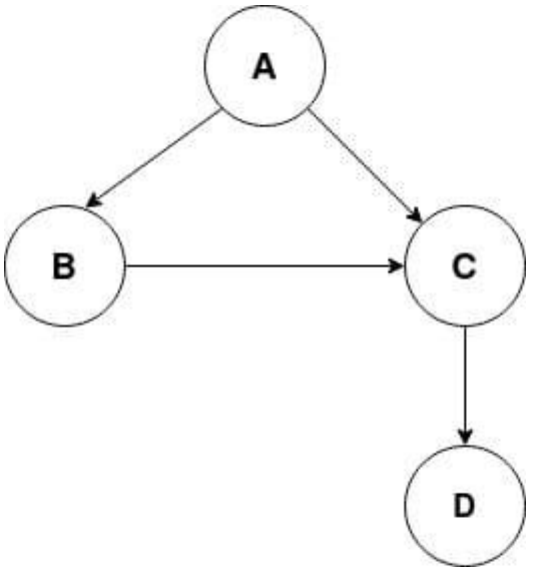
Depth first Search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph or tree data structure. Traversal means visiting all the nodes of a graph.

Adjacency list view of graph to list form

Step:1

A -> B
A -> C
B -> C
C -> D

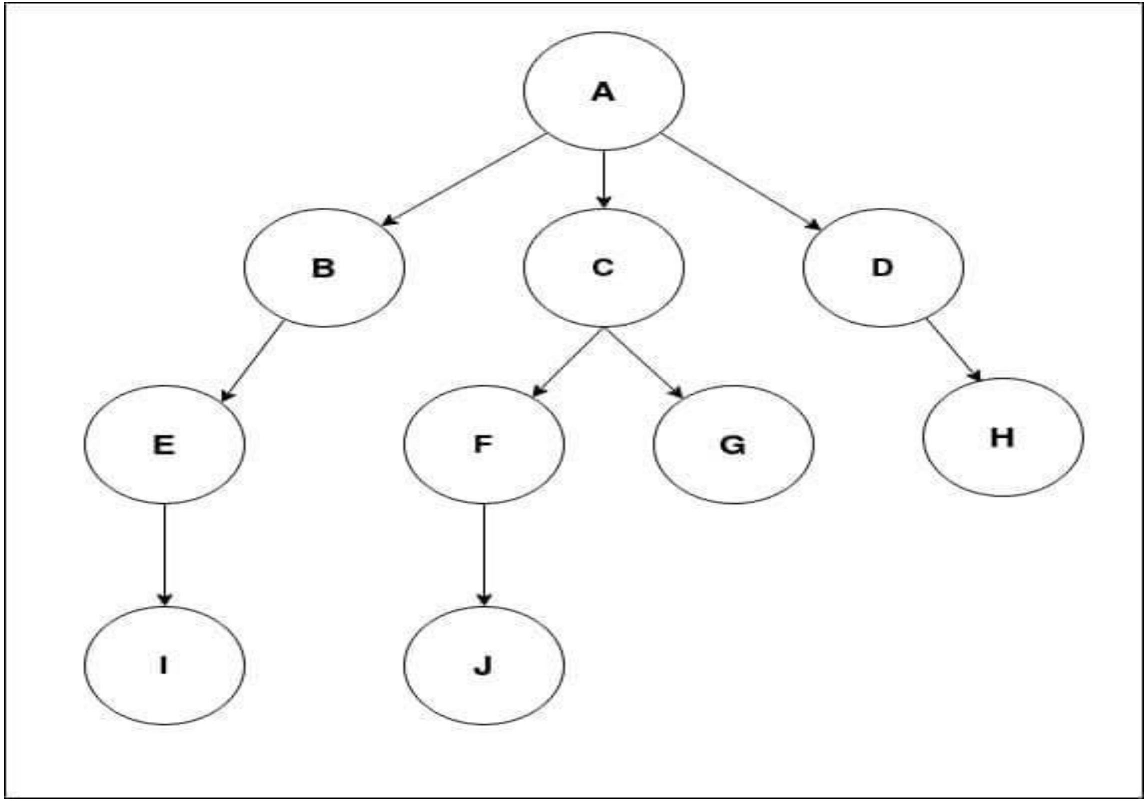
Step:2



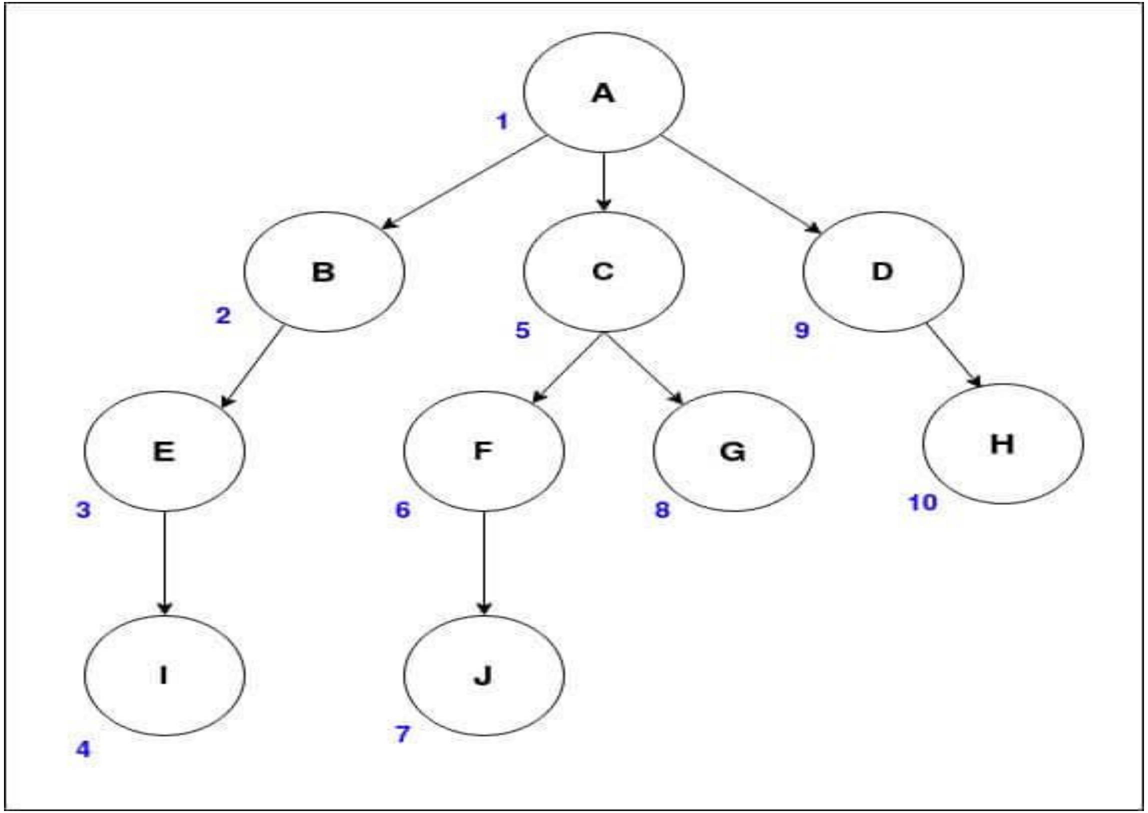
Step:3

```
graph = {
            "A": ["B", "C"],
            "B": ["C"],
             "C": ["D"]
                }
```

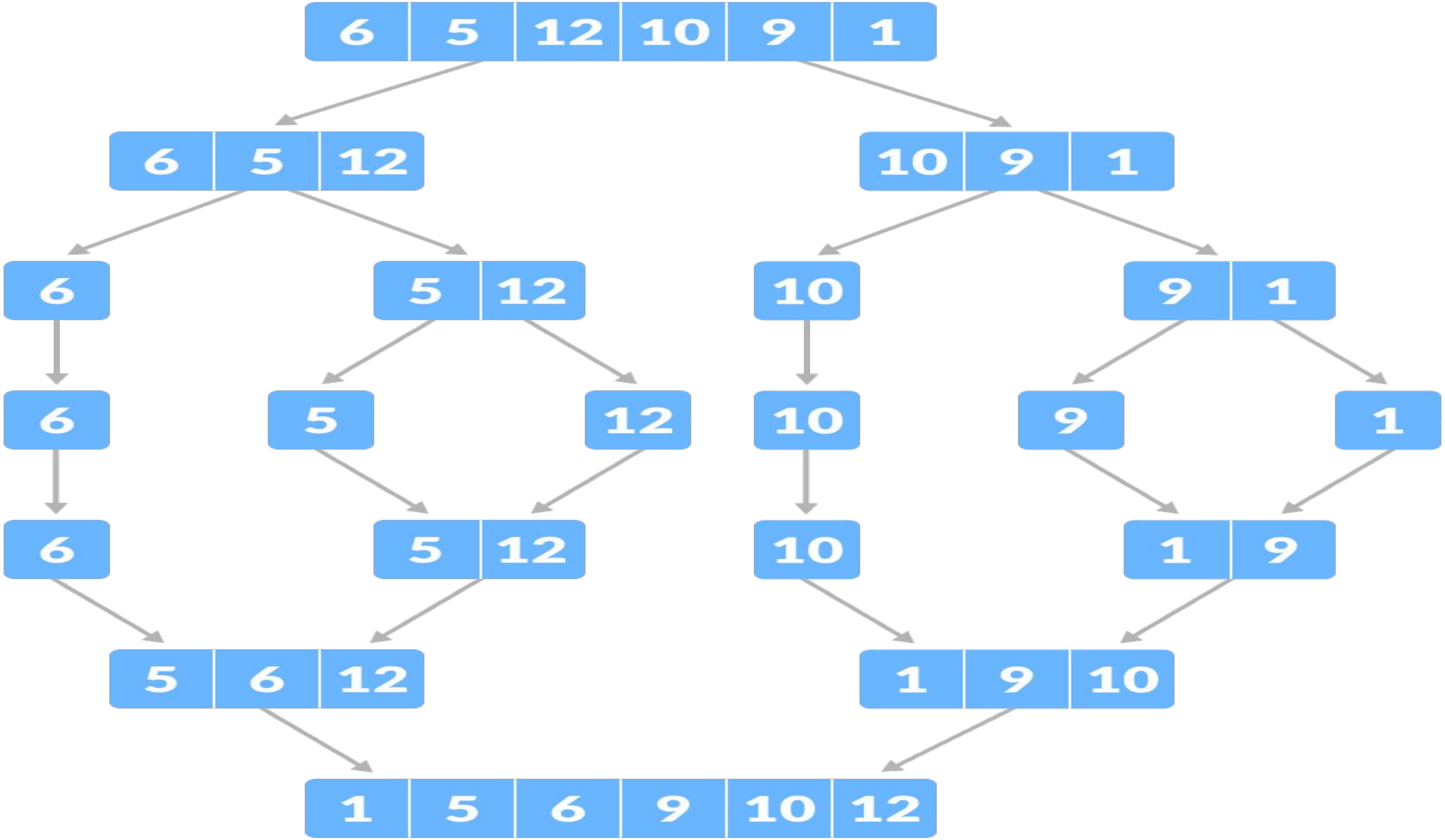
Question



Path of DFS



Divide and conquer


6 5 12 10 9 1
6 5 12
10 9 1
6
5 12
10
9 1
6
5
12
10
9
1
6
5 12
10
1 9
5 6 12
1 9 10
1 5 6 9 10 12