Tree Terminology

In linear data structure, data is organized in sequential order and in non-linear data structure, data is organized in random order. Tree is a very popular data structure used in wide range of applications. A tree data structure can be defined as follows...

Tree is a non-linear data structure which organizes data in hierarchical structure and this is a recursive definition.

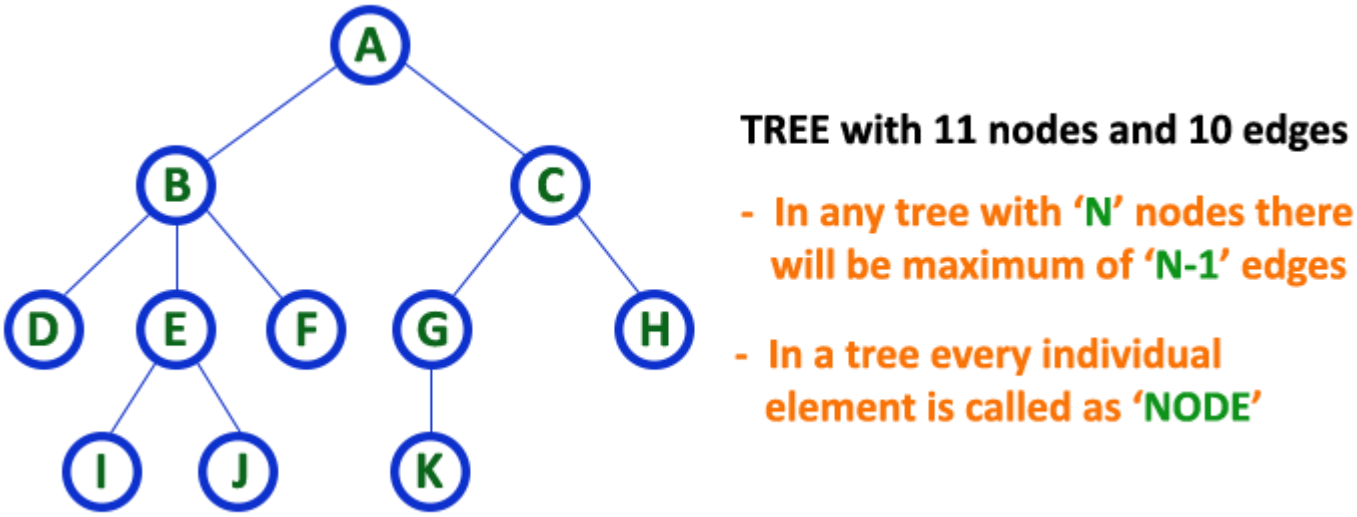
A tree data structure can also be defined as follows...

Tree data structure is a collection of data (Node) which is organized in hierarchical structure and this is a recursive definition

In tree data structure, every individual element is called as Node. Node in a tree data structure, stores the actual data of that particular element and link to next element in hierarchical structure.

In a tree data structure, if we have N number of nodes then we can have a maximum of N-1 number of links.

Example

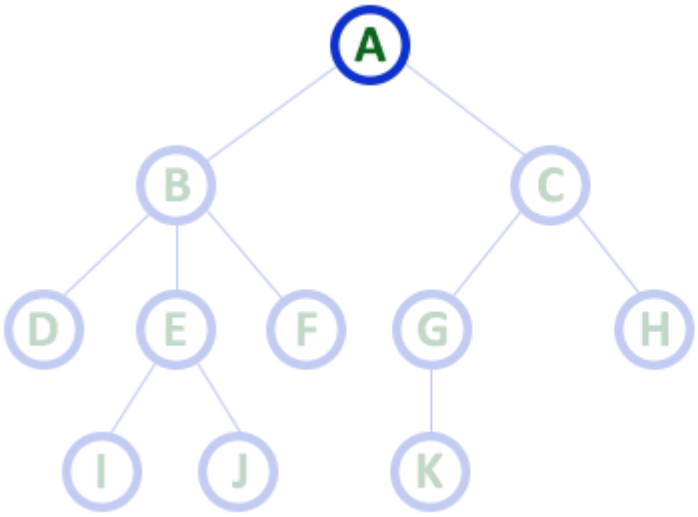


Terminology

In a tree data structure, we use the following terminology...

1. Root

In a tree data structure, the first node is called as Root Node. Every tree must have root node. We can say that root node is the origin of tree data structure. In any tree, there must be only one root node. We never have multiple root nodes in a tree.

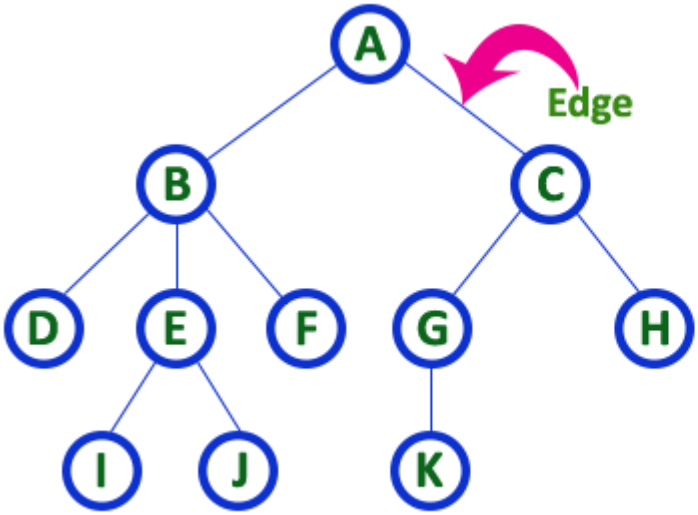


Here ‘A’ is the ‘root’ node

- In any tree the first node is called as ROOT node

2. Edge

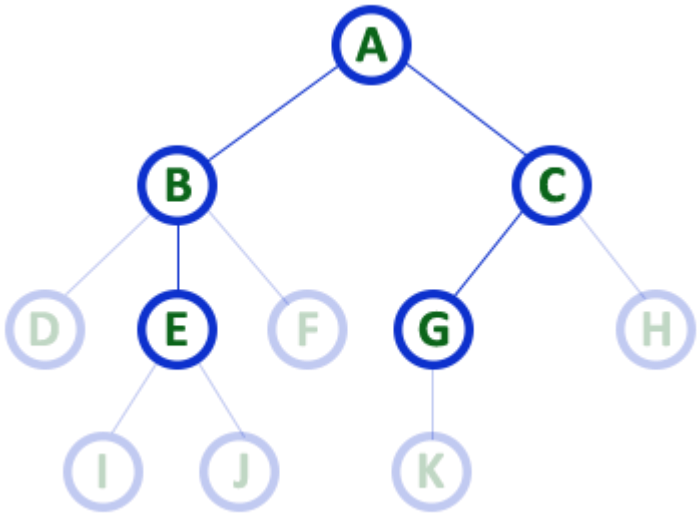
In a tree data structure, the connecting link between any two nodes is called as EDGE. In a tree with 'N' number of nodes there will be a maximum of 'N-1' number of edges.



- In any tree, ‘Edge’ is a connecting link between two nodes.

3. Parent

In a tree data structure, the node which is predecessor of any node is called as PARENT NODE. In simple words, the node which has branch from it to any other node is called as parent node. Parent node can also be defined as "The node which has child / children".

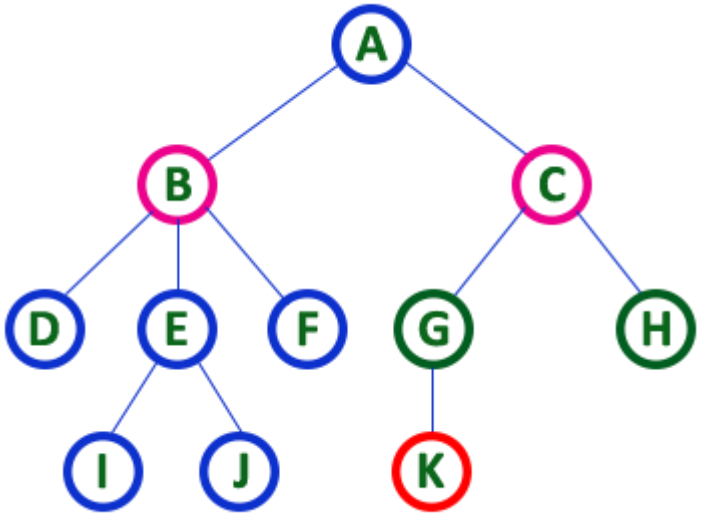


Here A, B, C, E & G are Parent nodes

- In any tree the node which has child / children is called 'Parent'
- A node which is predecessor of any other node is called 'Parent'

4. Child

In a tree data structure, the node which is descendant of any node is called as CHILD Node. In simple words, the node which has a link from its parent node is called as child node. In a tree, any parent node can have any number of child nodes. In a tree, all the nodes except root are child nodes.



Here B & C are Children of A

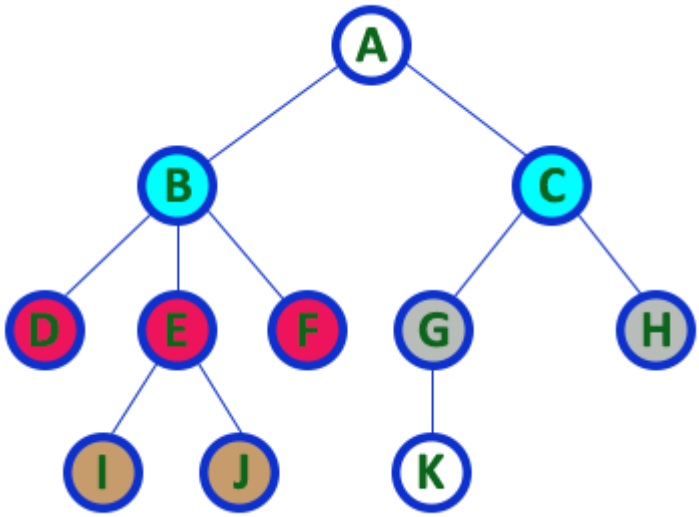
Here G & H are Children of C

Here K is Child of G

- descendant of any node is called as CHILD Node

5. Siblings

In a tree data structure, nodes which belong to same Parent are called as SIBLINGS. In simple words, the nodes with same parent are called as Sibling nodes.



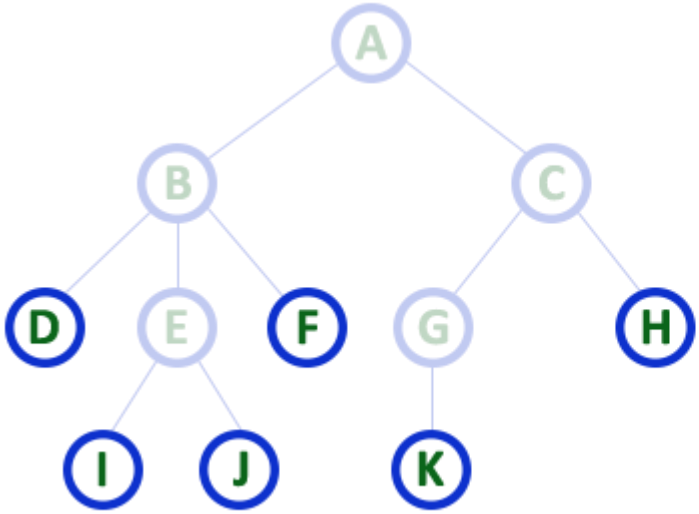
Here B & C are Siblings
Here D E & F are Siblings
Here G & H are Siblings
Here I & J are Siblings

- In any tree the nodes which has same Parent are called 'Siblings'
- The children of a Parent are called 'Siblings'

6. Leaf

In a tree data structure, the node which does not have a child is called as LEAF Node. In simple words, a leaf is a node with no child.

In a tree data structure, the leaf nodes are also called as External Nodes. External node is also a node with no child. In a tree, leaf node is also called as 'Terminal' node.

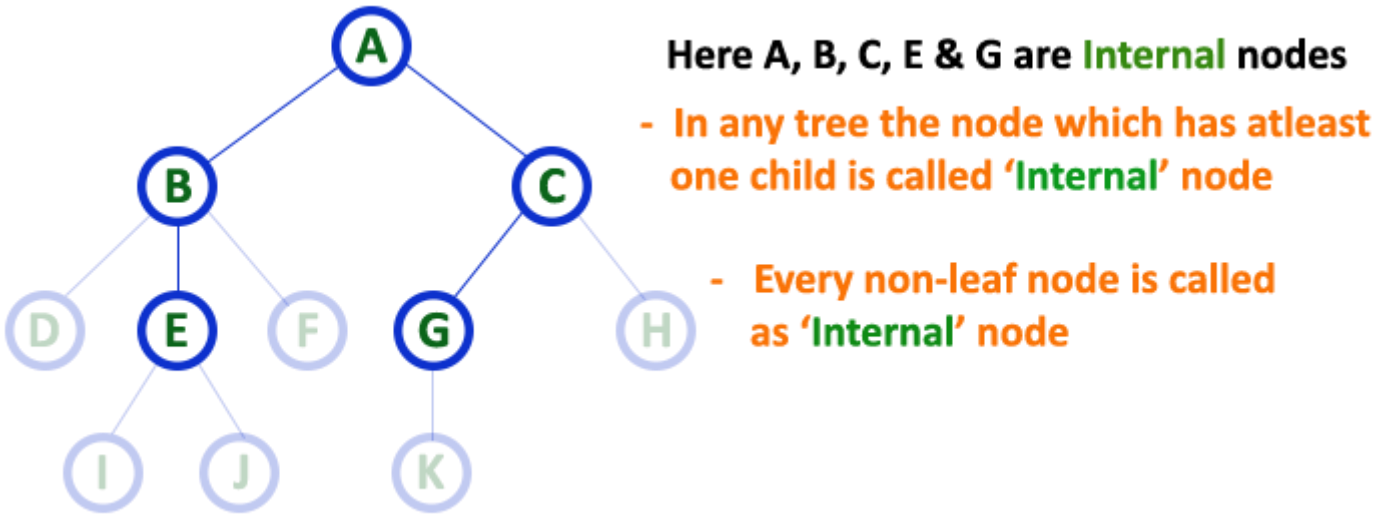


Here D, I, J, F, K & H are Leaf nodes

- In any tree the node which does not have children is called 'Leaf'
- A node without successors is called a 'leaf' node

7. Internal Nodes

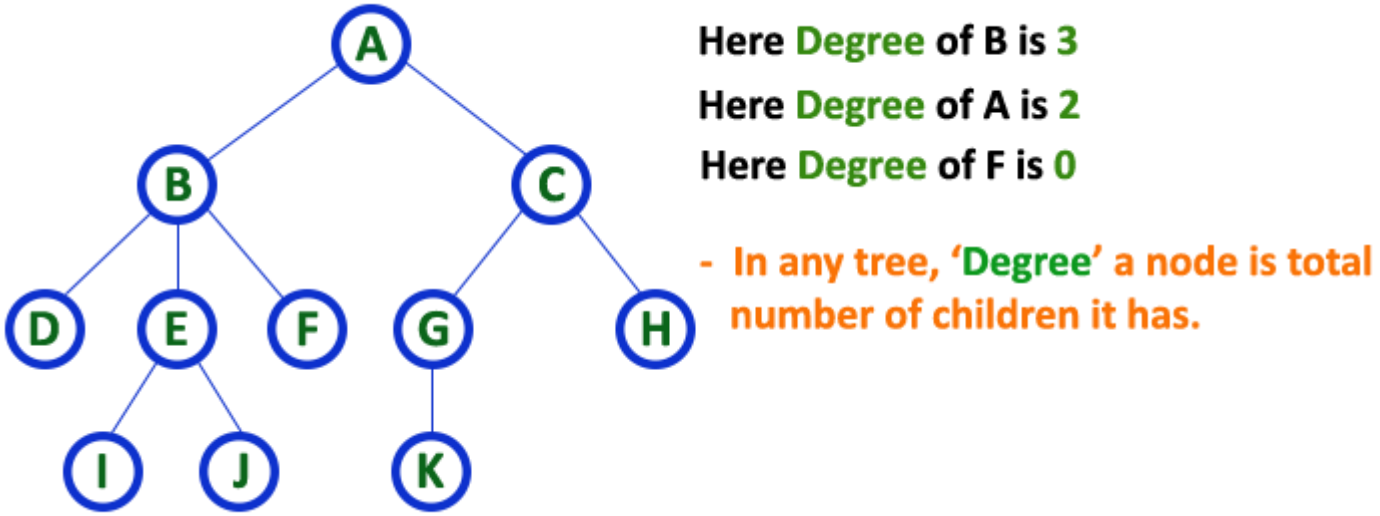
In a tree data structure, the node which has atleast one child is called as INTERNAL Node. In simple words, an internal node is a node with atleast one child.



In a tree data structure, nodes other than leaf nodes are called as Internal Nodes. The root node is also said to be Internal Node if the tree has more than one node. Internal nodes are also called as 'Non-Terminal' nodes.

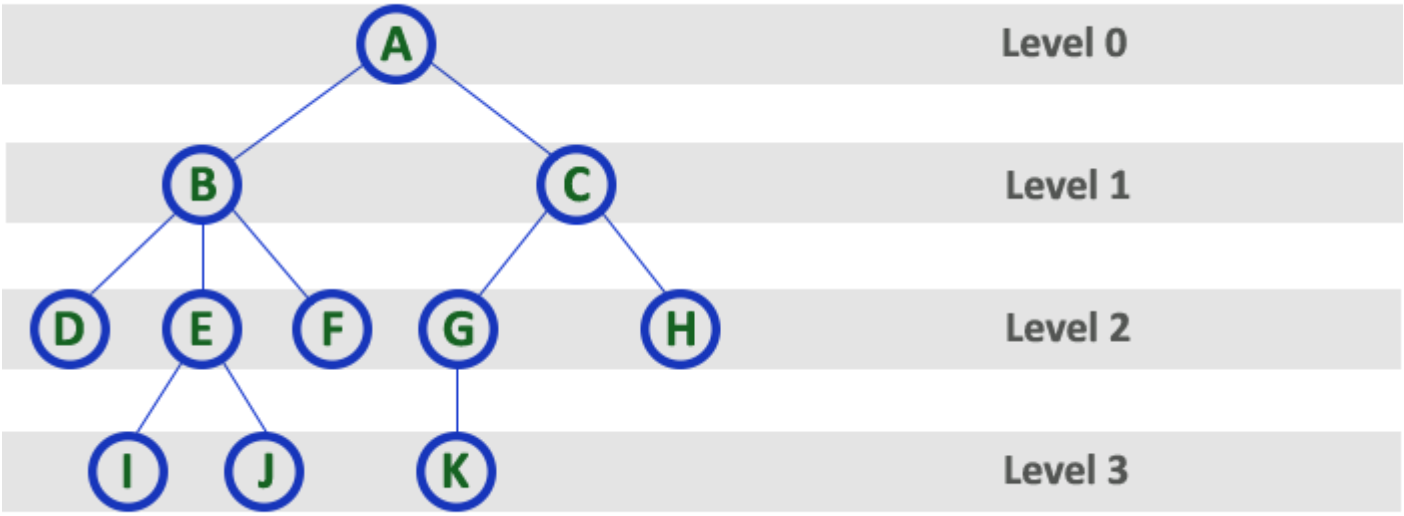
8. Degree

In a tree data structure, the total number of children of a node is called as DEGREE of that Node. In simple words, the Degree of a node is total number of children it has. The highest degree of a node among all the nodes in a tree is called as 'Degree of Tree'



9. Level

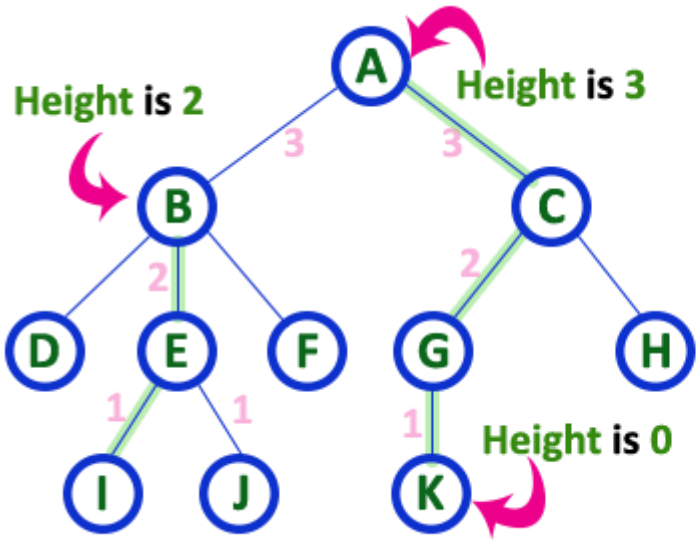
In a tree data structure, the root node is said to be at Level 0 and the children of root node are at Level 1 and the children of the nodes which are at Level 1 will be at Level 2 and so



on... In simple words, in a tree each step from top to bottom is called as a Level and the Level count starts with '0' and incremented by one at each level (Step).

10. Height

In a tree data structure, the total number of egdes from leaf node to a particular node in the longest path is called as HEIGHT of that Node. In a tree, height of the root node is said to be height of the tree. In a tree, height of all leaf nodes is '0'.

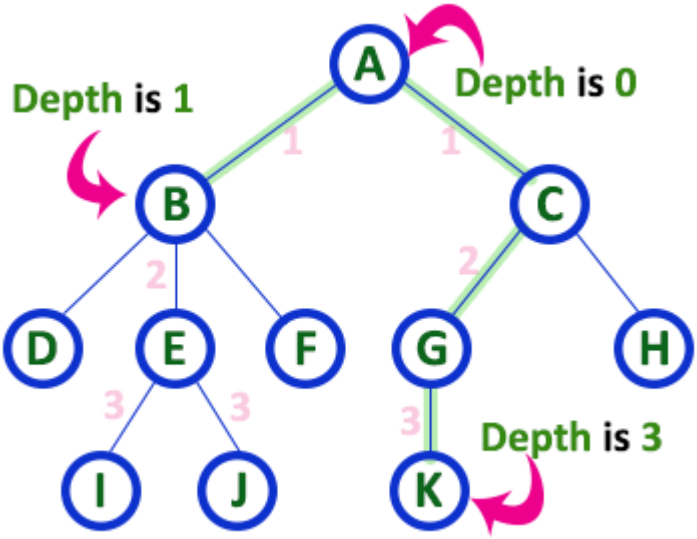


Here Height of tree is 3

- In any tree, 'Height of Node' is total number of Edges from leaf to that node in longest path.
- In any tree, 'Height of Tree' is the height of the root node.

11. Depth

In a tree data structure, the total number of edges from root node to a particular node is called as DEPTH of that Node. In a tree, the total number of edges from root node to a leaf node in the longest path is said to be Depth of the tree. In simple words, the highest depth of any leaf node in a tree is said to be depth of that tree. In a tree, depth of the root node is '0'.

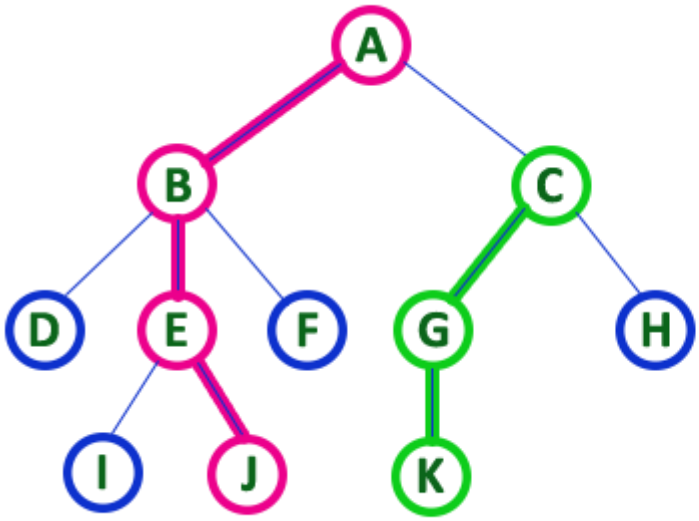


Here Depth of tree is 3

- In any tree, 'Depth of Node' is total number of Edges from root to that node.
- In any tree, 'Depth of Tree' is total number of edges from root to leaf in the longest path.

12. Path

In a tree data structure, the sequence of Nodes and Edges from one node to another node is called as PATH between that two Nodes. Length of a Path is total number of nodes in that path. In below example the path A - B - E - J has length 4.



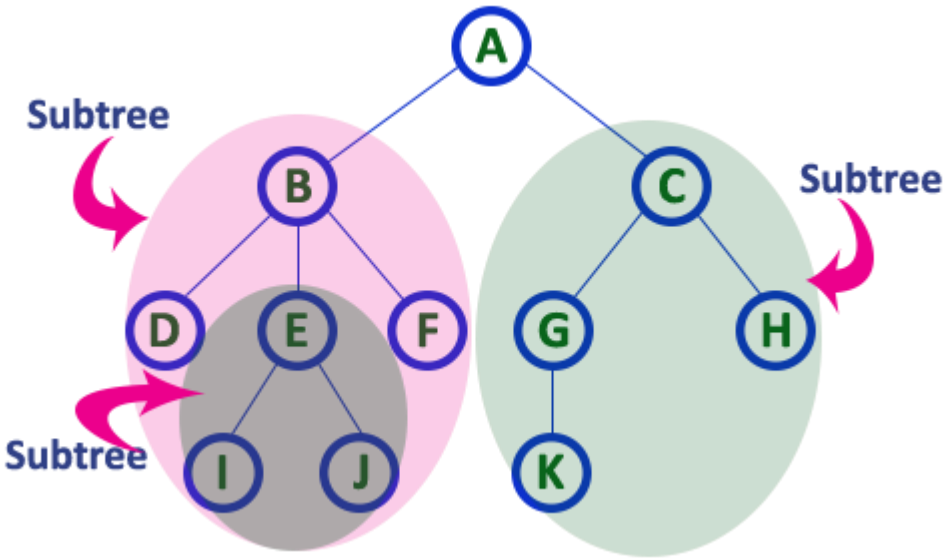
- In any tree, 'Path' is a sequence of nodes and edges between two nodes.

Here, 'Path' between A & J is
A - B - E - J

Here, 'Path' between C & K is
C - G - K

13. Sub Tree

In a tree data structure, each child from a node forms a subtree recursively. Every child node will form a subtree on its parent node.



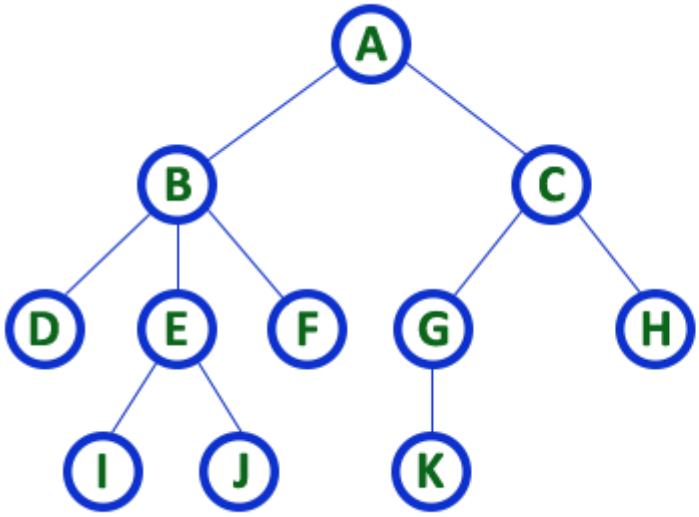
Tree Representations

A tree data structure can be represented in two methods. Those methods are as follows...

List Representation

Left Child - Right Sibling Representation

Consider the following tree...



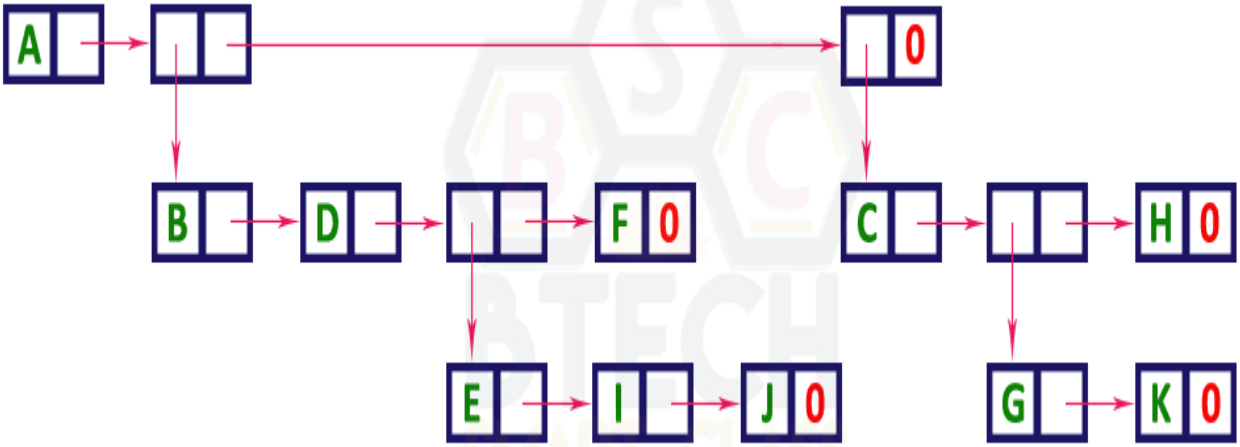
TREE with 11 nodes and 10 edges

- In any tree with 'N' nodes there will be maximum of 'N-1' edges
- In a tree every individual element is called as 'NODE'

1. List Representation

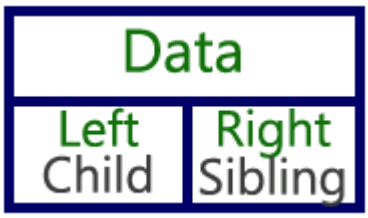
In this representation, we use two types of nodes one for representing the node with data and another for representing only references. We start with a node with data from root node in the tree. Then it is linked to an internal node through a reference node and is linked to any other node directly. This process repeats for all the nodes in the tree.

The above tree example can be represented using List representation as follows...



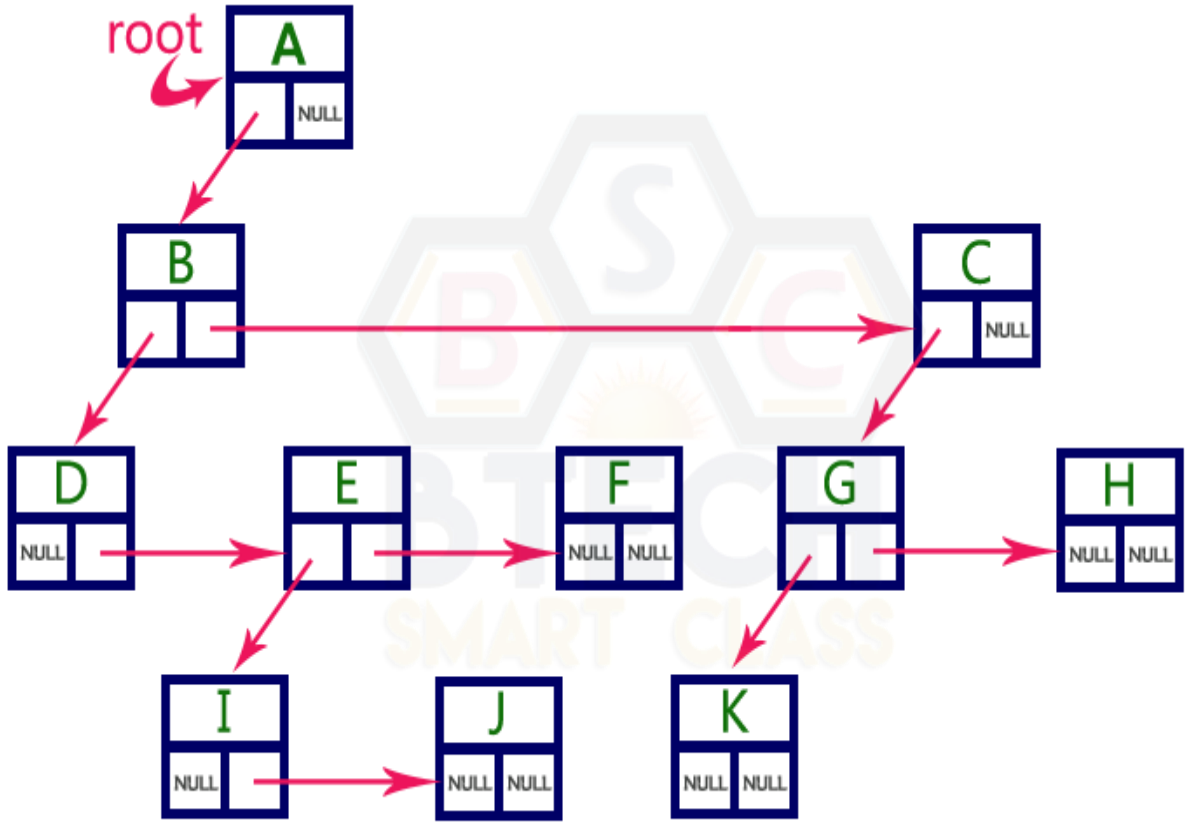
2. Left Child - Right Sibling Representation

In this representation, we use list with one type of node which consists of three fields namely Data field, Left child reference field and Right sibling reference field. Data field stores the actual value of a node, left reference field stores the address of the left child and right reference field stores the address of the right sibling node. Graphical representation of that node is as follows...



In this representation, every node's data field stores the actual value of that node. If that node has left child, then left reference field stores the address of that left child node otherwise that field stores NULL. If that node has right sibling then right reference field stores the address of right sibling node otherwise that field stores NULL.

The above tree example can be represented using Left Child - Right Sibling representation as follows...


root
A
NULL
B
C
NULL
D
NULL
E
F
NULL
NULL
G
H
NULL
NULL
I
NULL
J
NULL
NULL
K
NULL
NULL

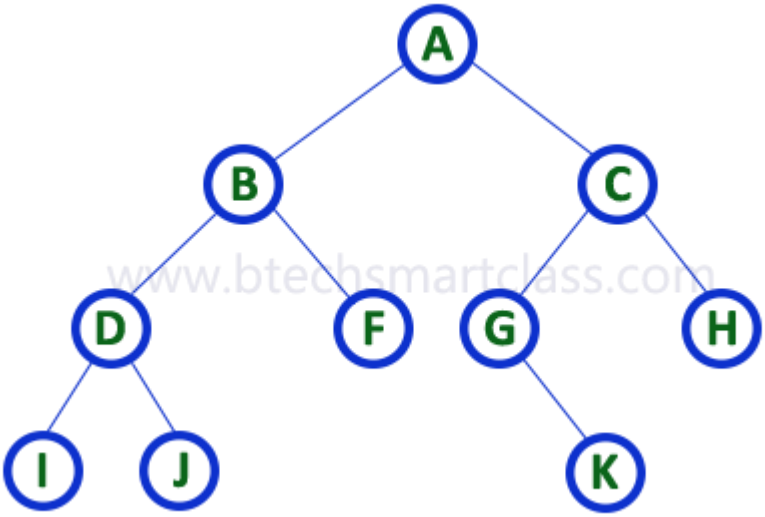
Binary Tree

In a normal tree, every node can have any number of children. Binary tree is a special type of tree data structure in which every node can have a maximum of 2 children. One is known as left child and the other is known as right child.

A tree in which every node can have a maximum of two children is called as Binary Tree.

In a binary tree, every node can have either 0 children or 1 child or 2 children but not more than 2 children.

Example



www.btechsmartclass.com

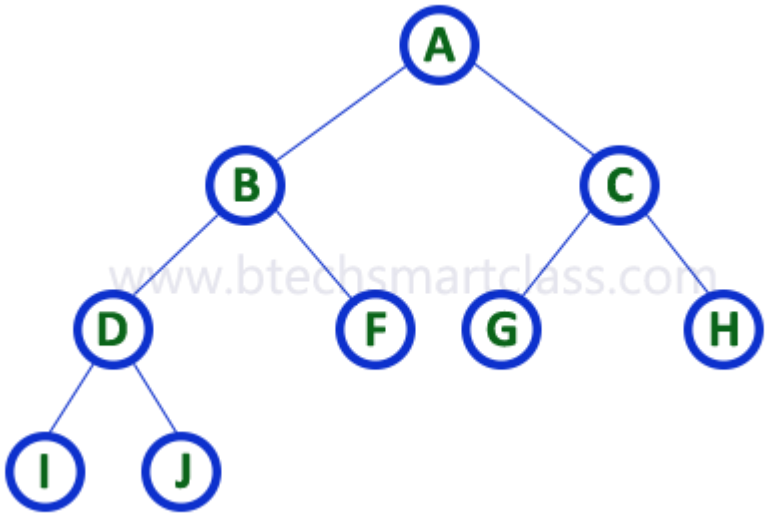
There are different types of binary trees and they are...

1. Strictly Binary Tree

In a binary tree, every node can have a maximum of two children. But in strictly binary tree, every node should have exactly two children or none. That means every internal node must have exactly two children. A strictly Binary Tree can be defined as follows...

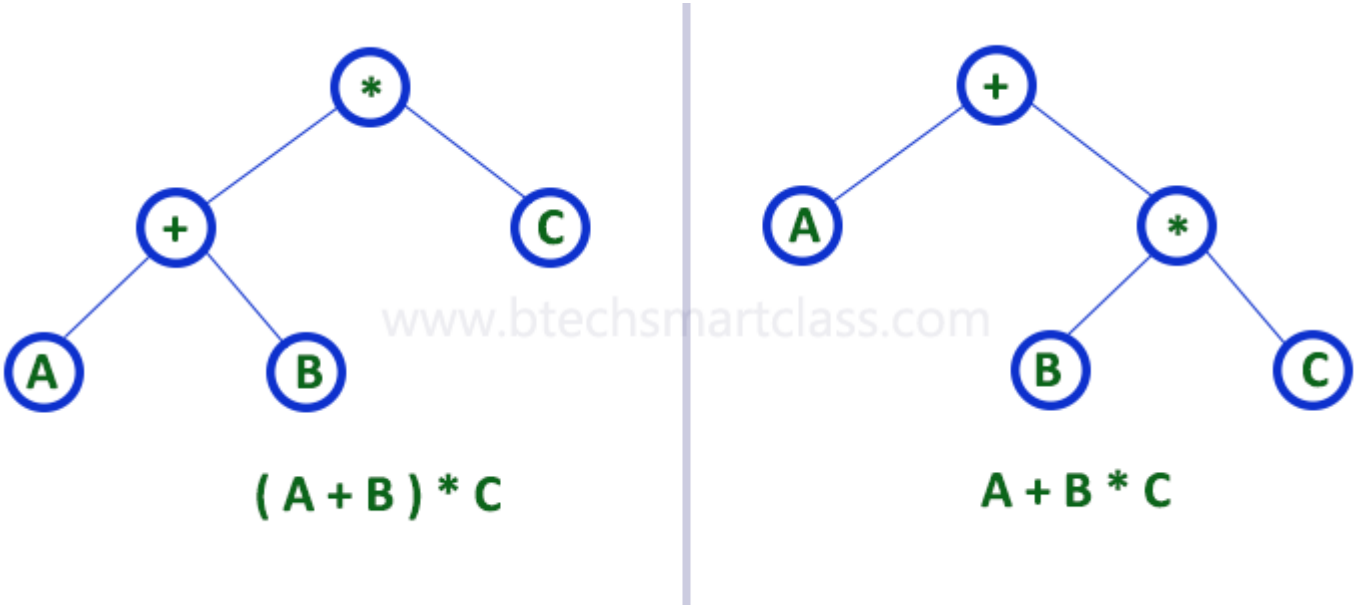
A binary tree in which every node has either two or zero number of children is called Strictly Binary Tree

Strictly binary tree is also called as Full Binary Tree or Proper Binary Tree or 2-Tree



Strictly binary tree data structure is used to represent mathematical expressions.

Example

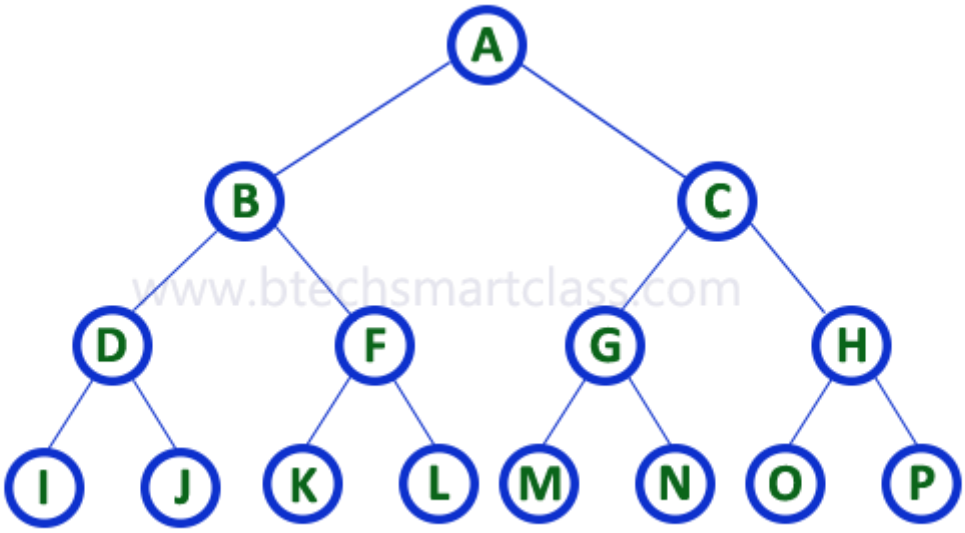


2. Complete Binary Tree

In a binary tree, every node can have a maximum of two children. But in strictly binary tree, every node should have exactly two children or none and in complete binary tree all the nodes must have exactly two children and at every level of complete binary tree there must be 2level number of nodes. For example at level 2 there must be 22 = 4 nodes and at level 3 there must be 23 = 8 nodes.

A binary tree in which every internal node has exactly two children and all leaf nodes are at same level is called Complete Binary Tree.

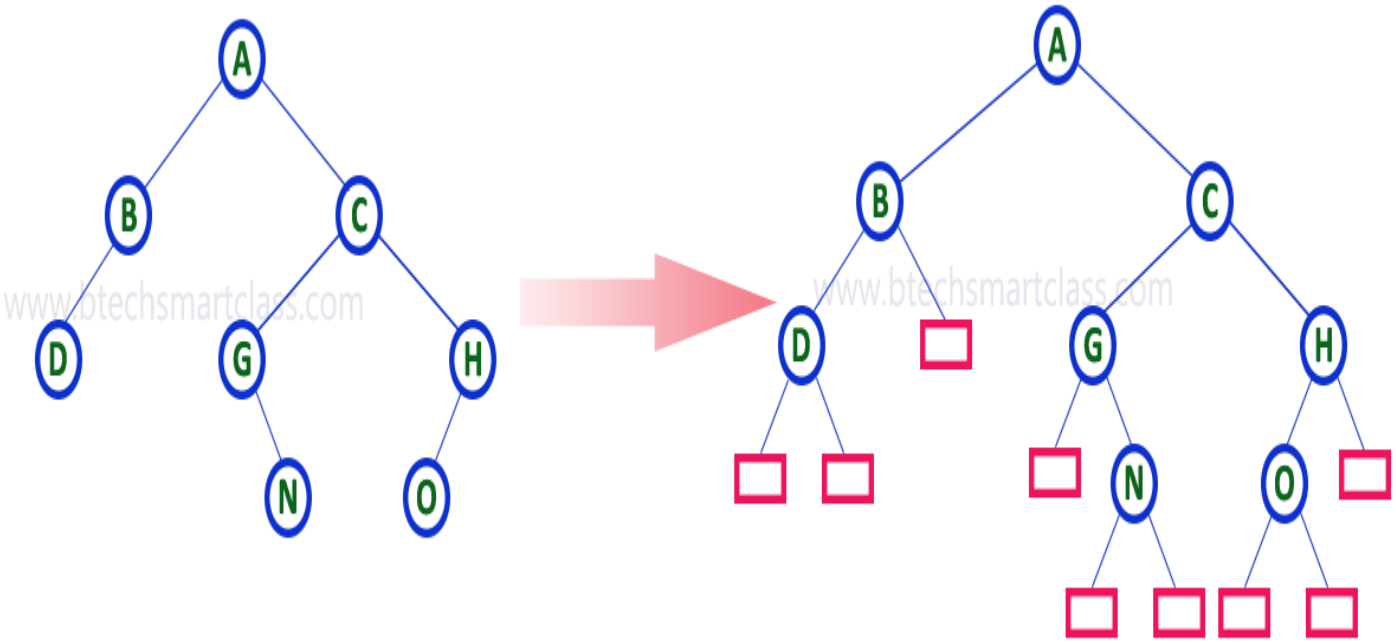
Complete binary tree is also called as Perfect Binary Tree



3. Extended Binary Tree

A binary tree can be converted into Full Binary tree by adding dummy nodes to existing nodes wherever required.

The full binary tree obtained by adding dummy nodes to a binary tree is called as Extended Binary Tree.



In above figure, a normal binary tree is converted into full binary tree by adding dummy nodes (In pink colour).

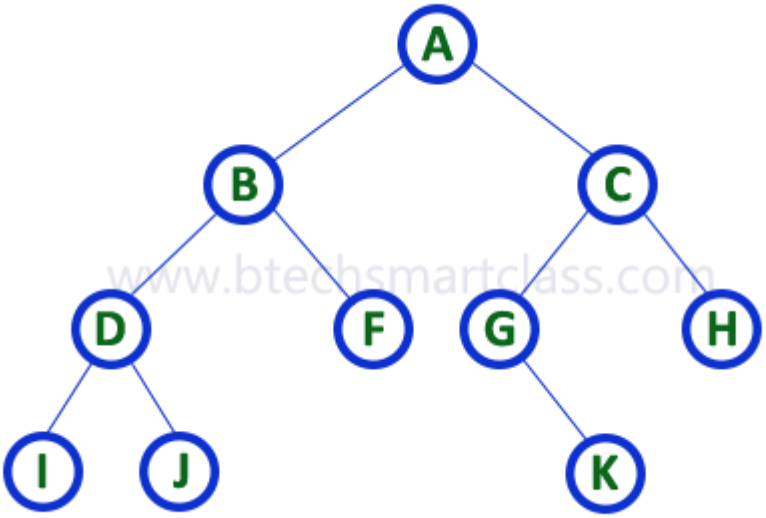
Binary Tree Representations

A binary tree data structure is represented using two methods. Those methods are as follows...

Array Representation

Linked List Representation

Consider the following binary tree...



www.btechsmartclass.com

1. Array Representation

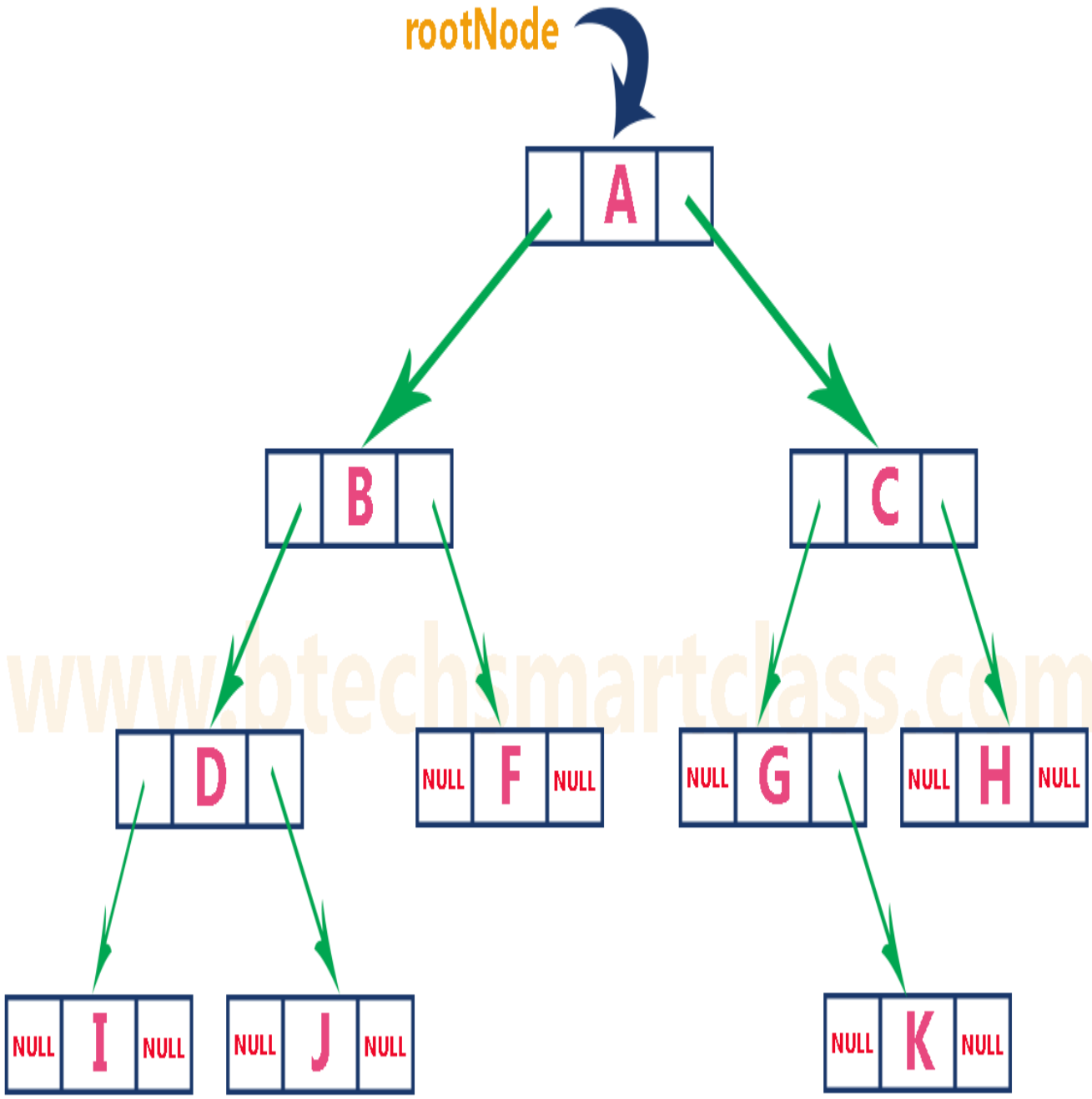
In array representation of binary tree, we use a one dimensional array (1-D Array) to represent a binary tree.
Consider the above example of binary tree and it is represented as follows...

To represent a binary tree of depth 'n' using array representation, we need one dimensional array with a maximum size of 2n+1 - 1.

2. Linked List Representation

We use double linked list to represent a binary tree. In a double linked list, every node consists of three fields. First field for storing left child address, second for storing actual data and third for storing right child address.
In this linked list representation, a node has the following structure...

The above example of binary tree represented using Linked list representation is shown as follows...



www.btechsmartclass.com