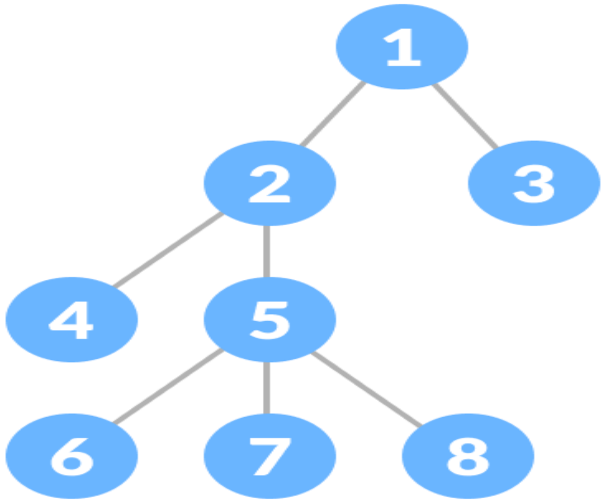
Chapter 7

Tree and its Application

Introduction t o Tree

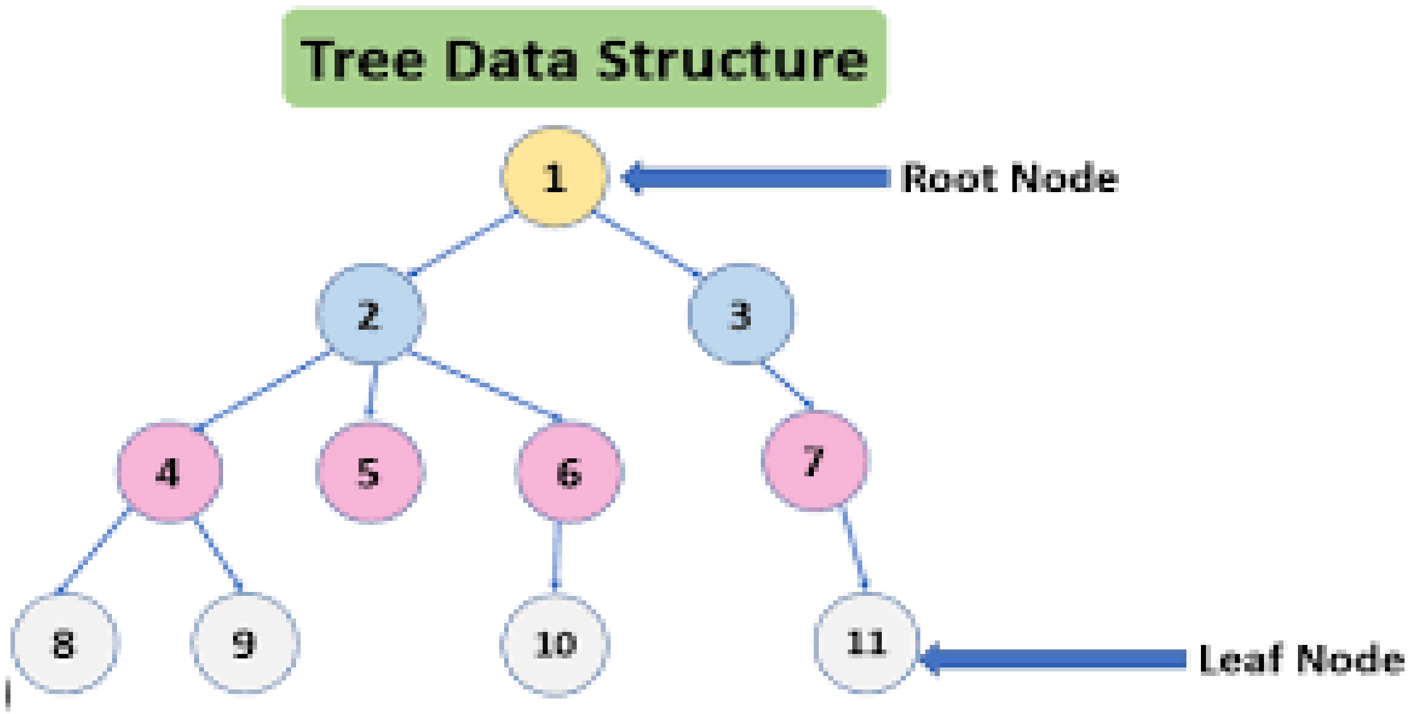
- Tree is a hierarchical data structure that consists of nodes connected by edges. It's a non-linear data structure where each node can have zero or more child nodes.
- Each node has only one parent (except for the root node, which has no parent).



Tree is a collection of data (Node) which is organized in hierarchical structure.

Introduction t o Tree (2)

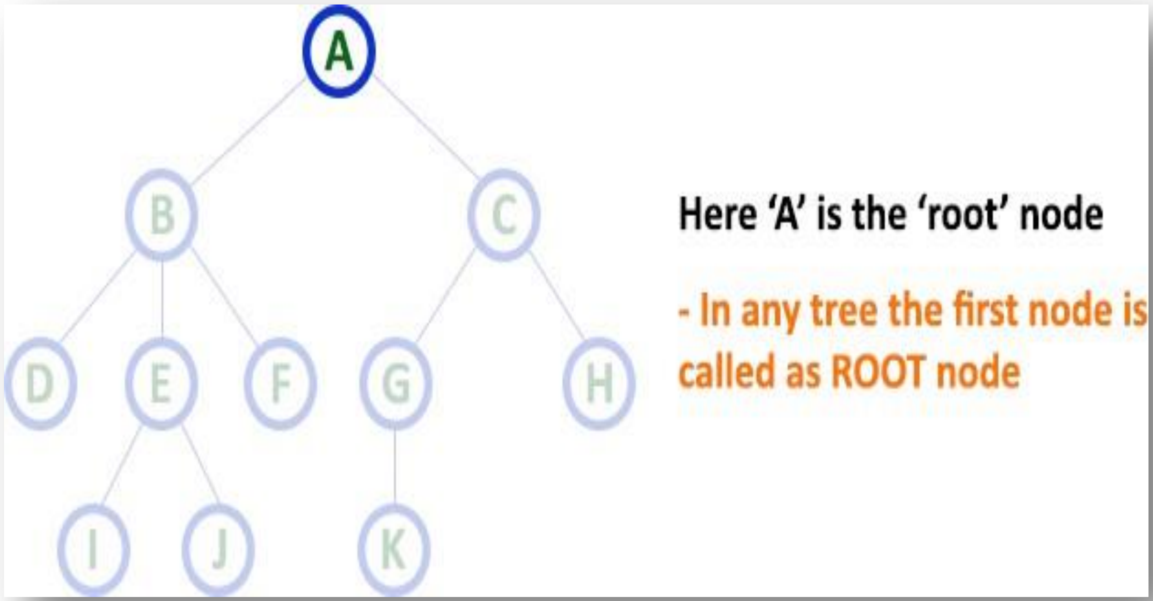
- Node in a tree data structure stores the actual data of that particular element and link to next element in hierarchical structure.
- In a tree, if we have N number of nodes then we can have a maximum of N-1 number of links



Tree Terminology

Root Node

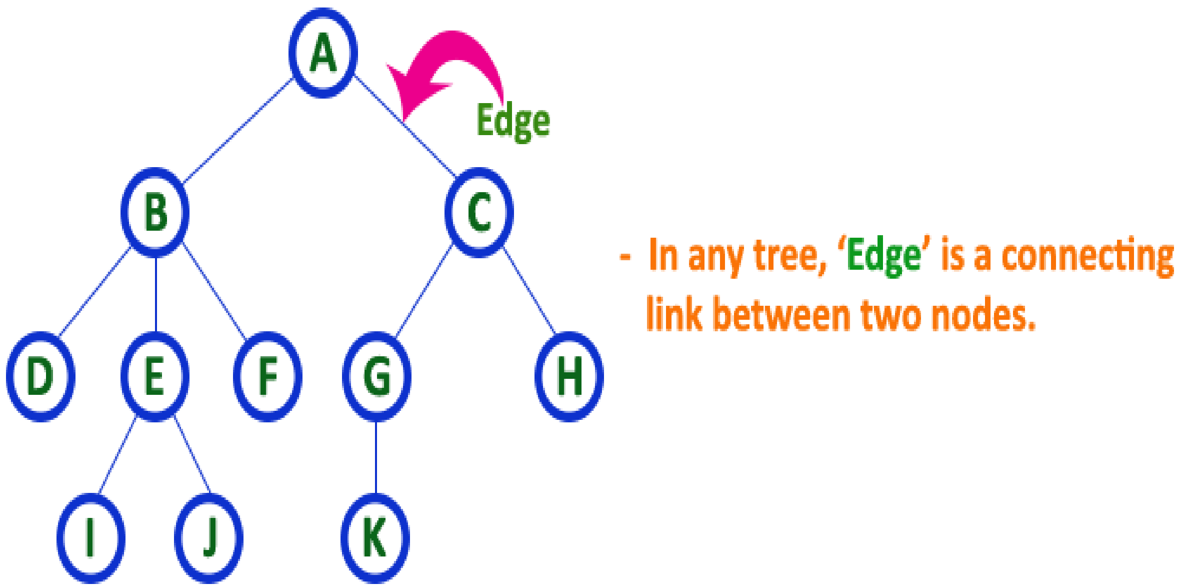
- In a tree data structure, the first node is called as Root Node.
- Every tree must have a root node. We can say that the root node is the origin of the tree data structure.
- In any tree, there must be only one root node. We never have multiple root nodes in a tree.


A
B
C
D
E
F
G
H
I
J
K
Here 'A' is the 'root' node
- In any tree the first node is called as ROOT node

Tree Terminology (2)

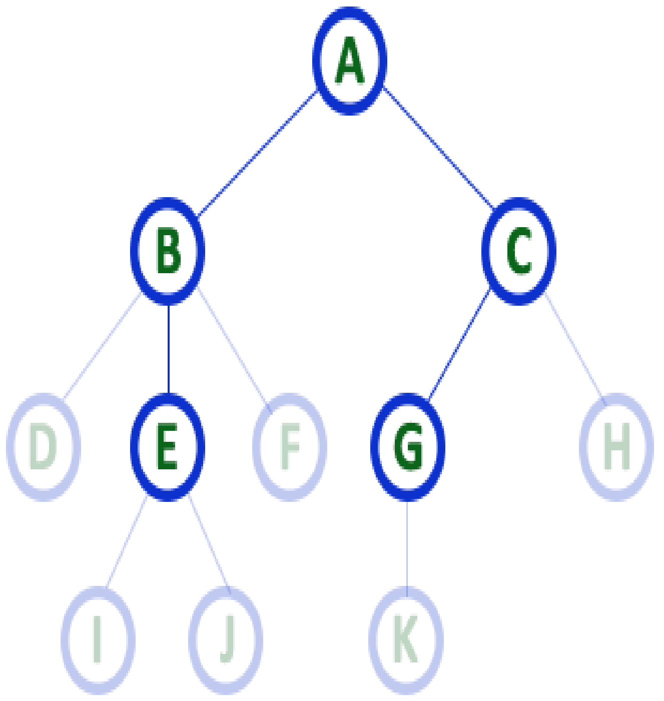
Edge

- 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.


A
B
C
D
E
F
G
H
I
J
K
Edge
- In any tree, 'Edge' is a connecting link between two nodes.

Parent

- The node which is a predecessor of any node is called as PARENT NODE. In simple words, the node which has a branch from it to any other node is called a parent node.

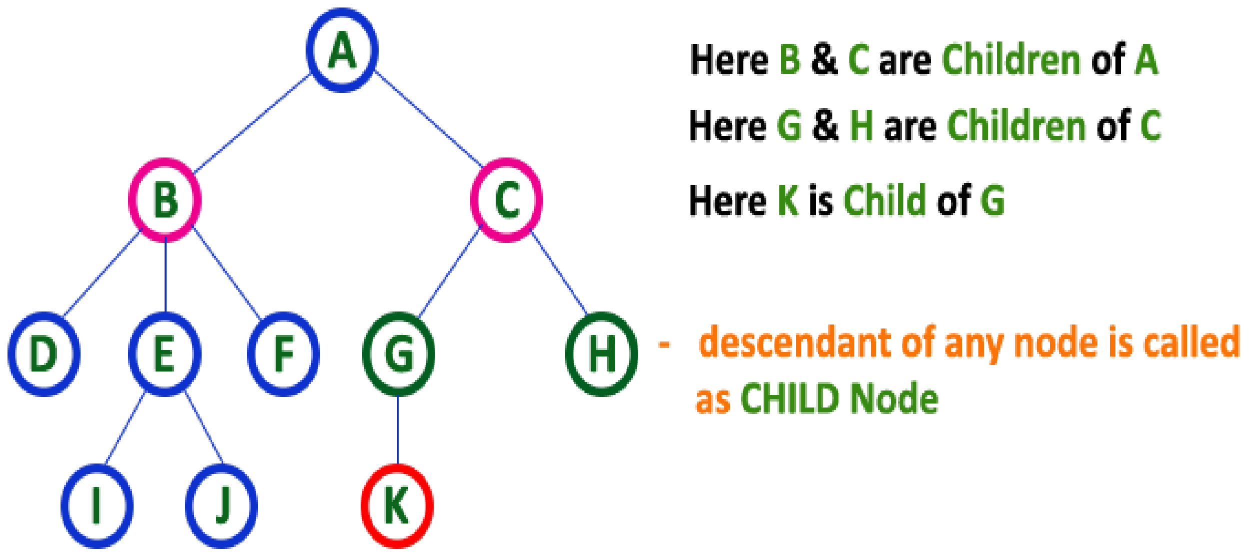


Here A, B, C, E & G are Parent nodes

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

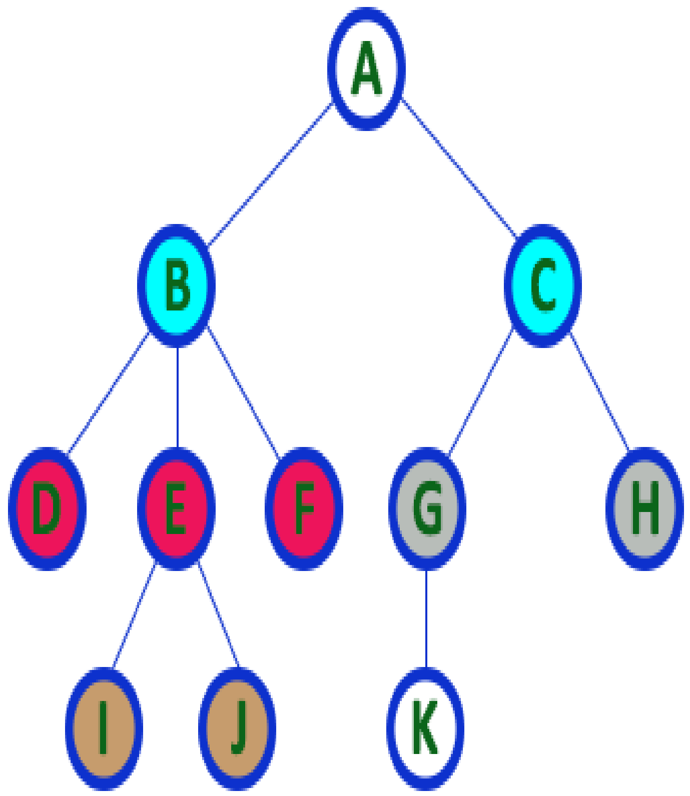
Child

- The node which is successor (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.


A
B
C
D
E
F
G
H
I
J
K
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

Siblings

- The nodes which belong to same Parent are called as SIBLINGS. In simple words, the nodes with the same parent are called Sibling nodes.



Here B & C are Siblings

Here D E & F are Siblings

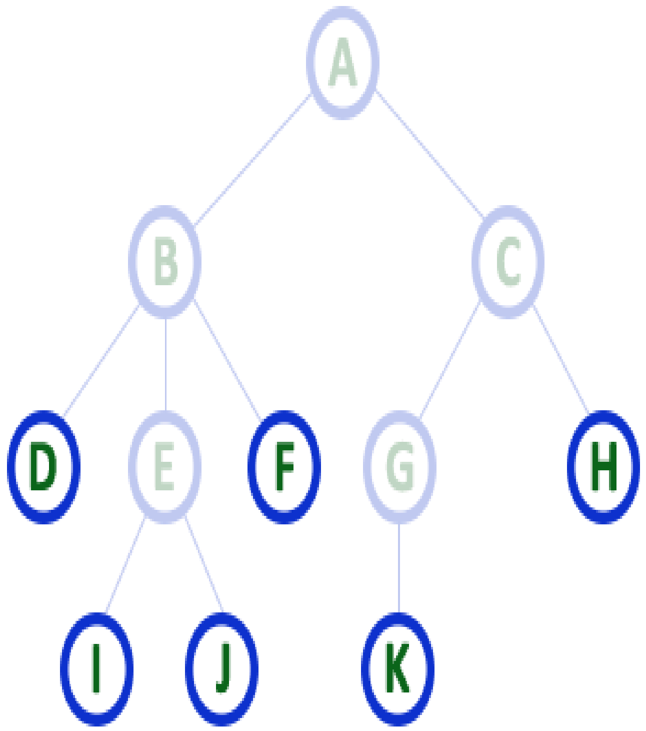
Here G & H are Siblings

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'

Leaf

- 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.
- In a tree, leaf node is also called as 'Terminal' node.

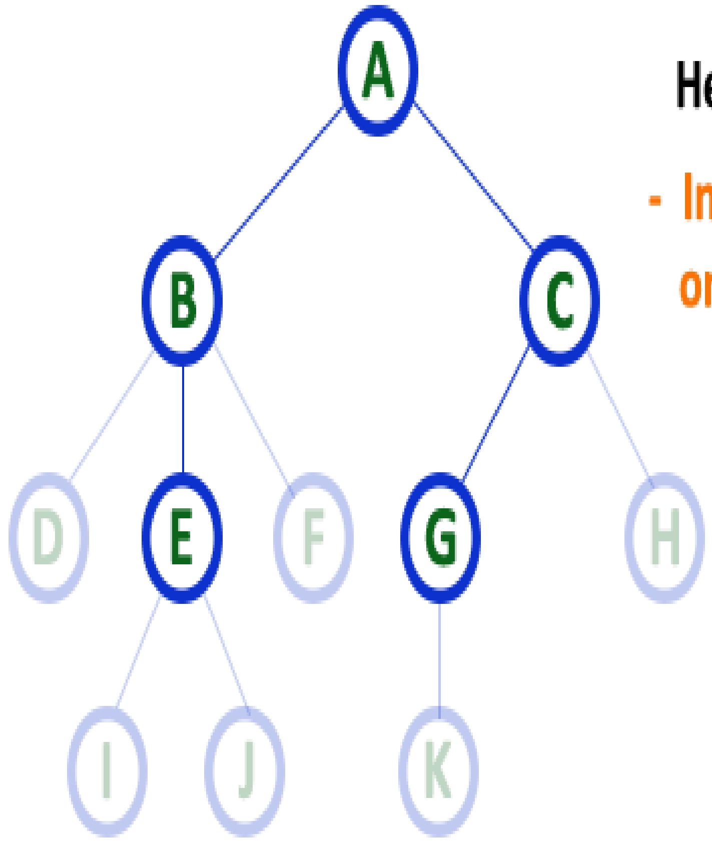


Here D, I, J, F, K & H are Leaf nodes

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

Internal Nodes

- The node which has at least one child is called as INTERNAL Node. Internal nodes are also called as 'Non-Terminal nodes.



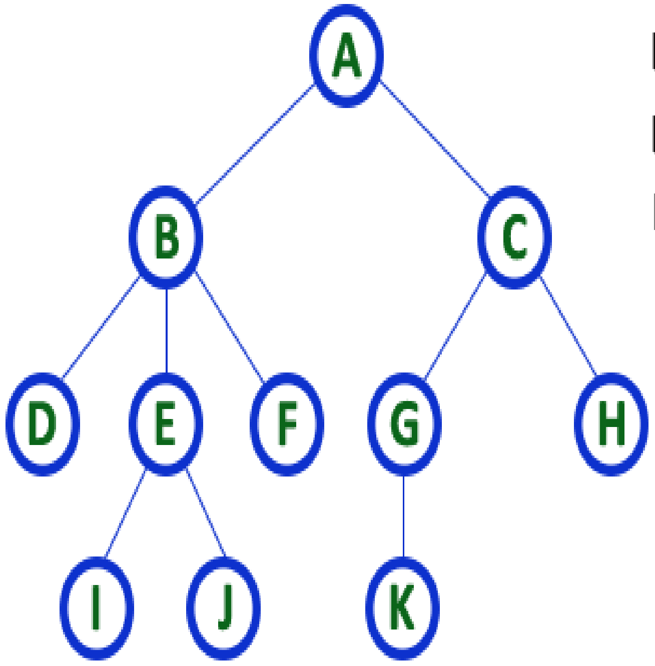
Here A, B, C, E & G are Internal nodes

- In any tree the node which has atleast one child is called 'Internal' node

- Every non-leaf node is called as 'Internal' node

Degree

- 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'



Here Degree of B is 3

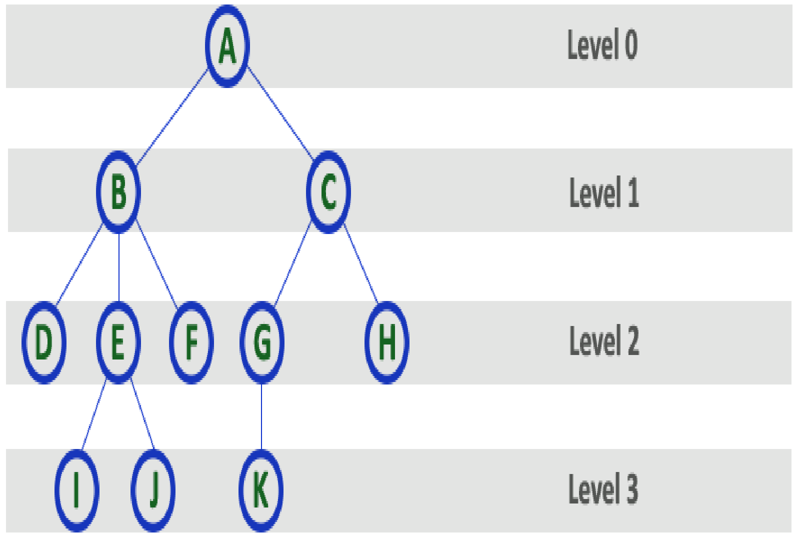
Here Degree of A is 2

Here Degree of F is 0

- In any tree, 'Degree' of a node is total number of children it has.

Level

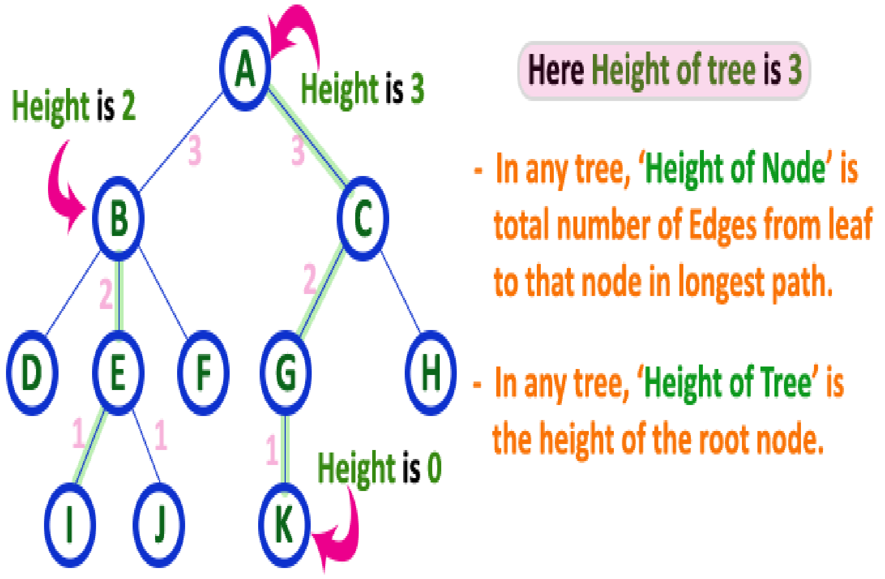
The root node is said to be at Level 0 and the children of root node are at Level 1 and the children of the nodes which are at Level 1 will be at Level 2 and so on... In simple words, in a tree each step from top to bottom is called as a Level and the Level count starts with '0' and incremented by one at each level (Step).



Graph Terminology (4)

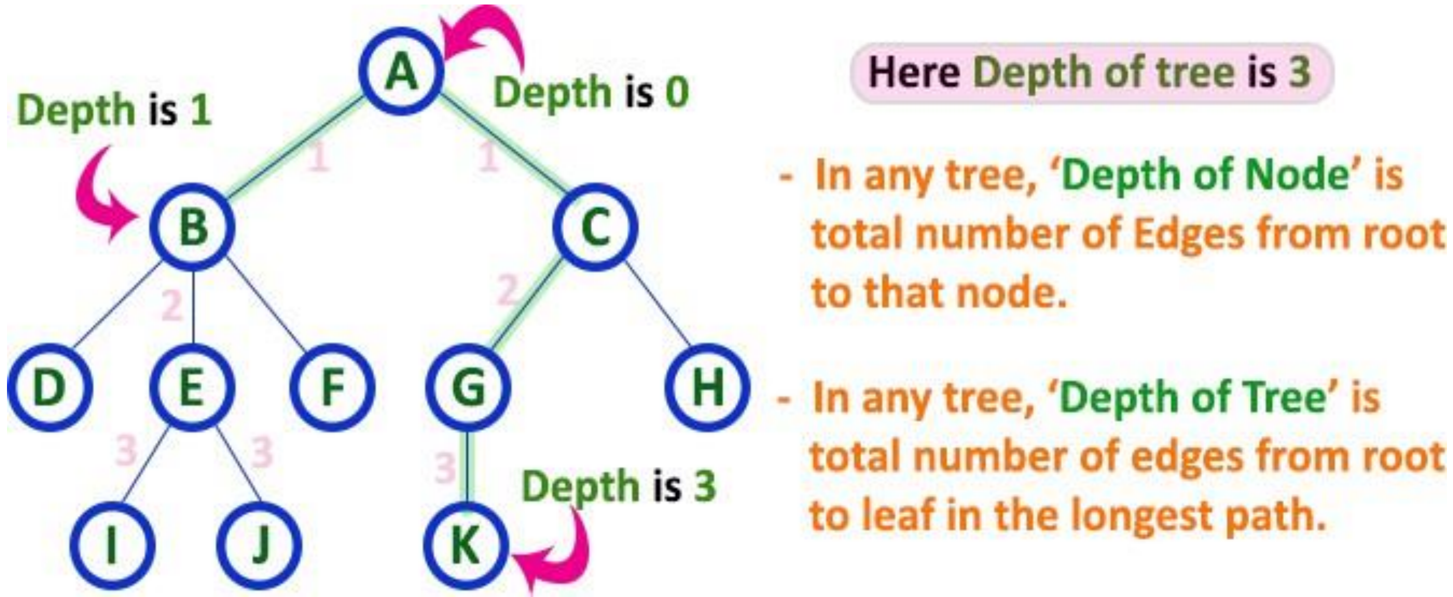
Height

- In a tree data structure, the total number of edges 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
Height is 3
Height is 2
Height is 0
A
B
C
D
E
F
G
H
I
J
K
3
3
2
2
1
1
1
- 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.

Depth

- The total number of egdes 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'.

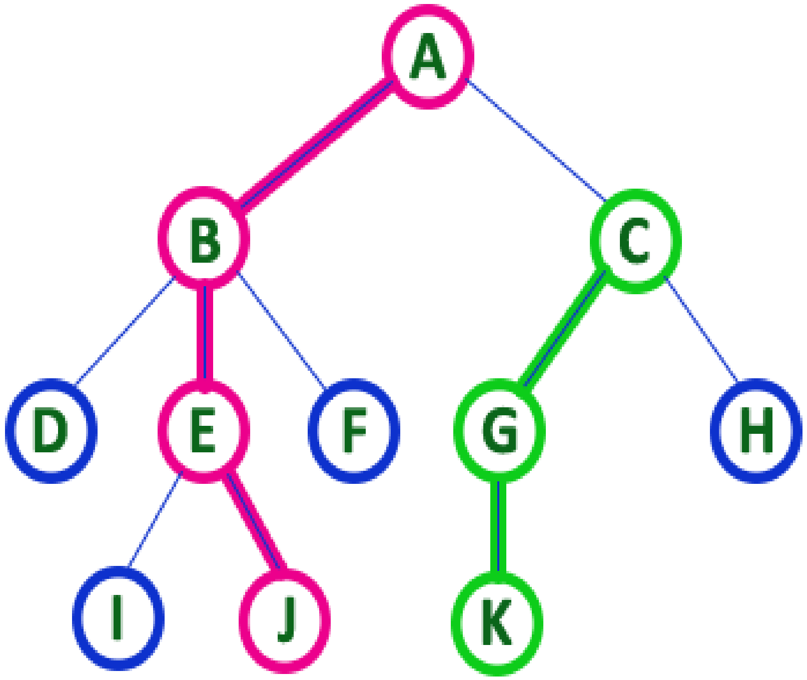


Depth is 1
Depth is 0
Depth is 3
A
B
C
D
E
F
G
H
I
J
K
1
1
2
2
3
3
3
Here Depth of tree is 3
- In any tree, 'Depth of Node' is total number of Edges from root to that node.
- In any tree, 'Depth of Tree' is total number of edges from root to leaf in the longest path.

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

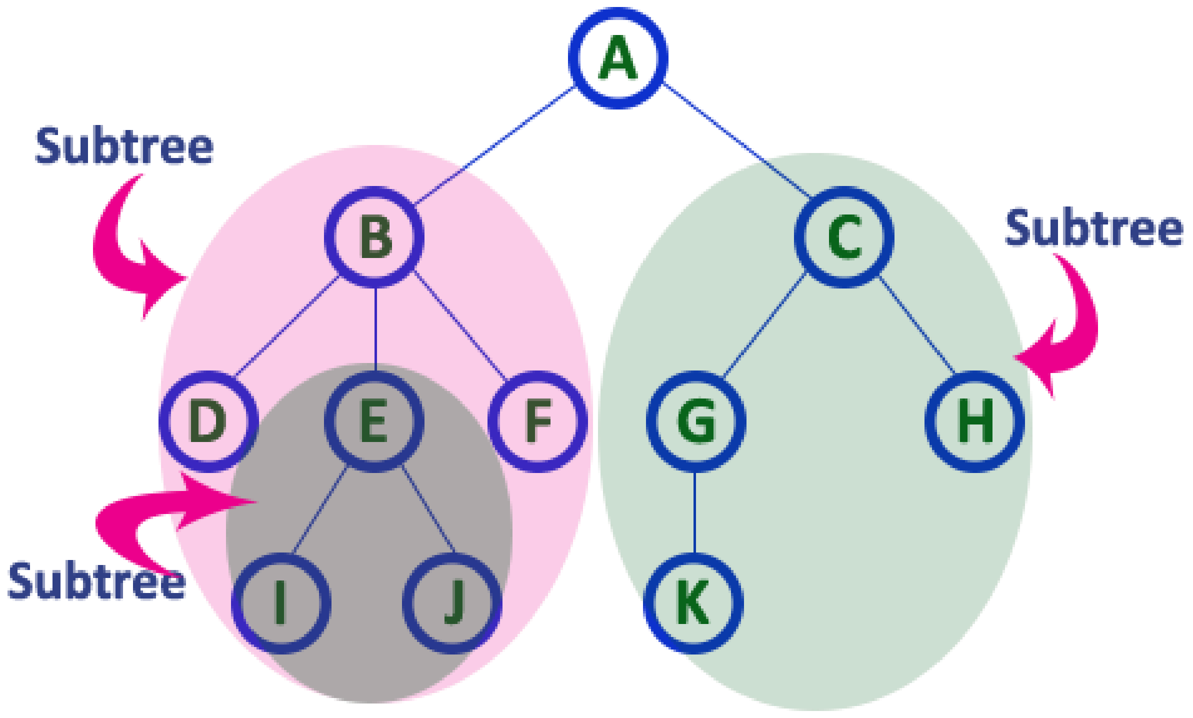
Here, 'Path' between C & K is
C - G - K

Graph Terminology (5)

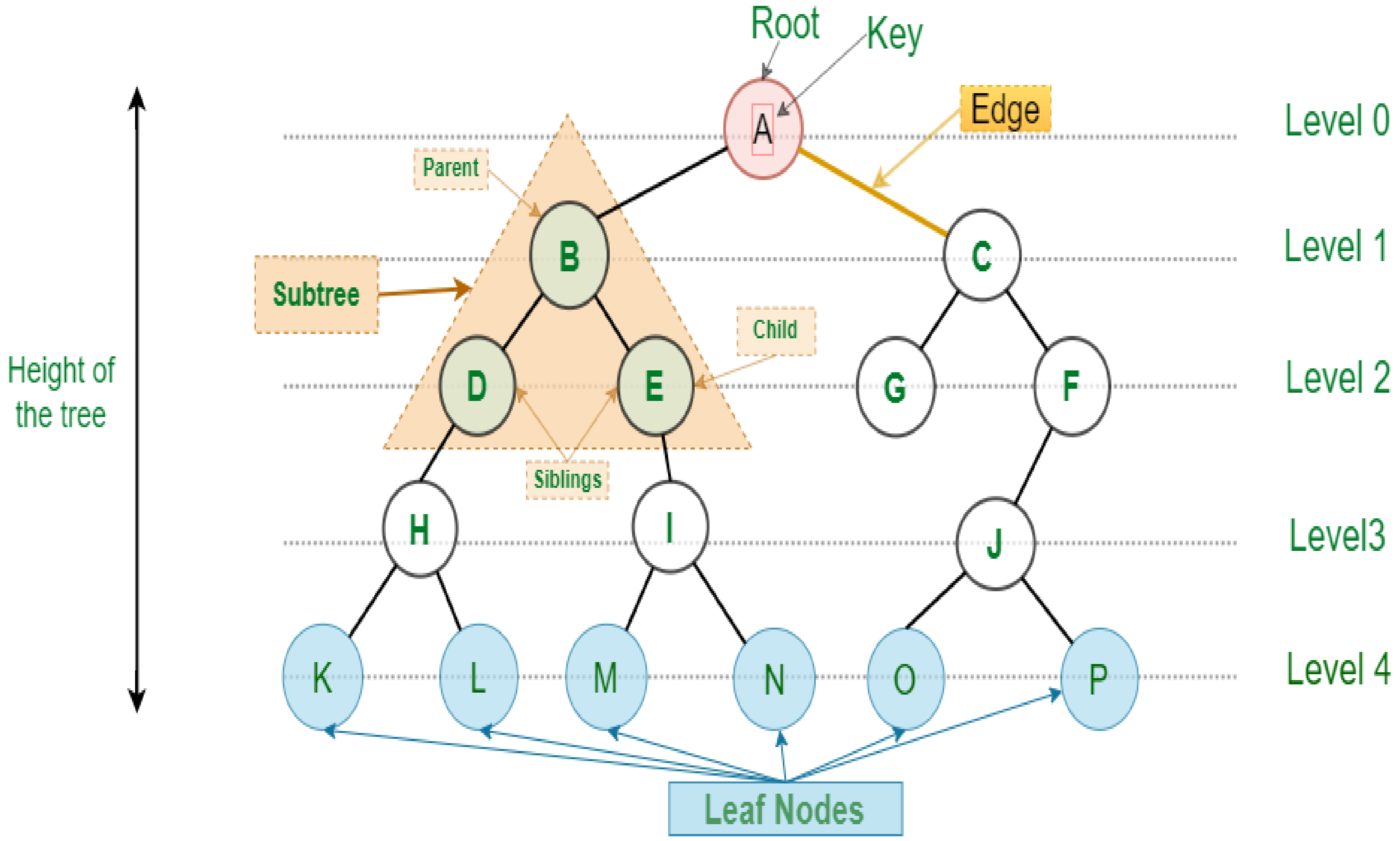
Sub Tree

In a tree data structure,Every child node will form a sub tree

on its parent node.



Tree Data Structure



Terminologies used in Tree:

According to the above example image of a tree

- **Nodes:** A B C D E F G H I J K L M N O P
- **Root:** A
- **Internal Nodes:** A B C D E F H I J
- **External nodes:** K L M N O P
- **(Parent , Child) :** (A, B and C), (B, D and E), (C, G and F),(H, K and L), (I, M and N) , (J, O AND P).
- Siblings : (B,C) , (D, E), (G, F), (K, L), (M, N) ,(O, P)

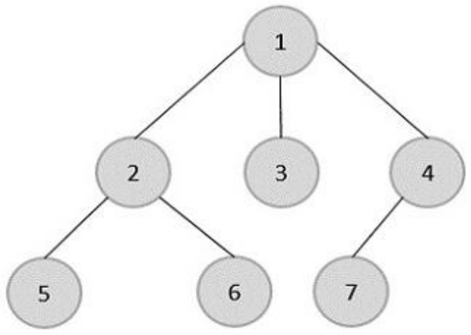
Types of Trees

There are three types of trees –

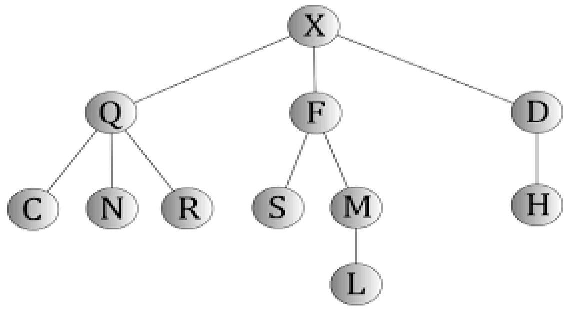
1. General Trees
2. Binary Trees
3. Binary Search Trees

General trees

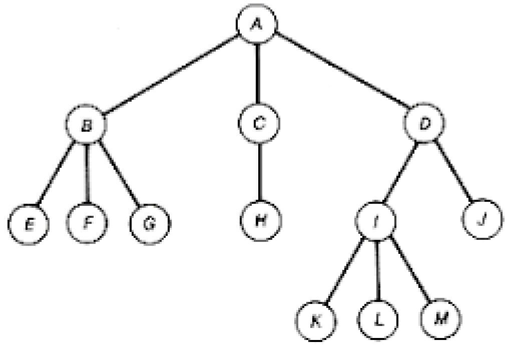
- General trees data structures that don't have limitations on the number of children each node can have, unlike binary trees where nodes have at most two children.
- Each node in a general tree can have multiple children.
- These trees don't necessarily follow a specific order among their children, so the arrangement can vary



(a)



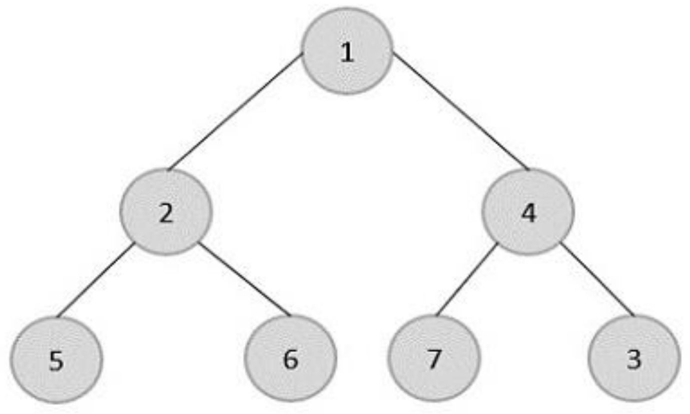
(b)



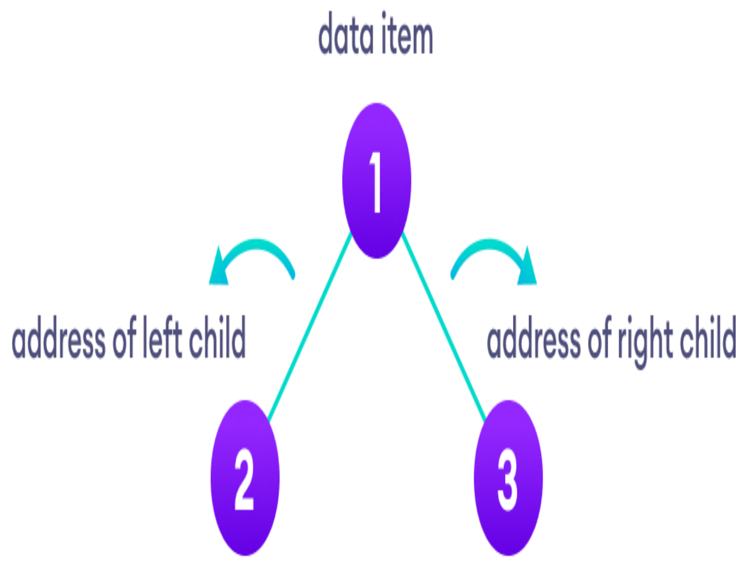
(c)

Binary Trees

- Binary Trees are general trees in which the root node can only hold up to maximum 2 subtrees: left subtree and right subtree.
- A binary tree is a tree data structure in which each parent node can have at most two children.



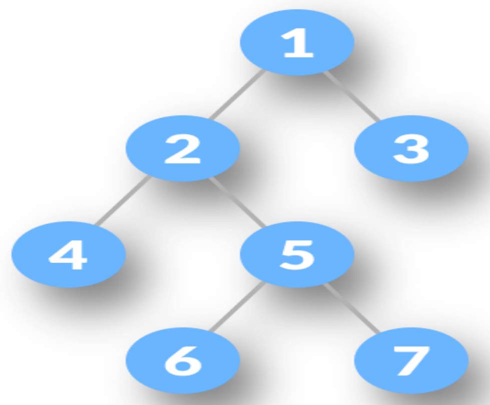
Binary Tree Data Structure



Types of Binary Tree

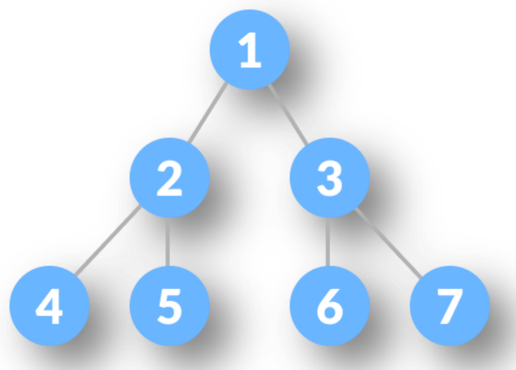
1. Full Binary Tree

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



2. Perfect Binary Tree

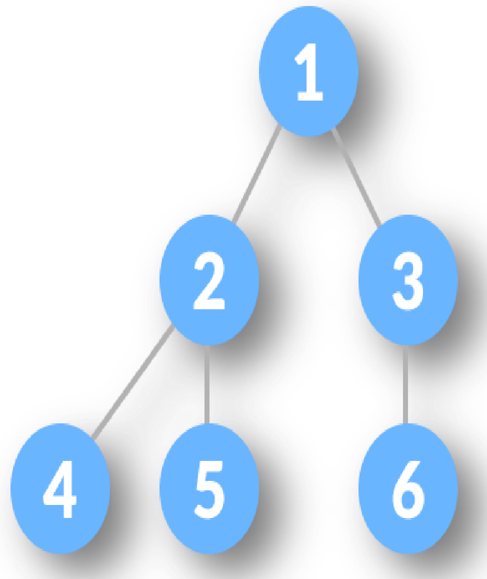
A perfect binary tree is a type of binary tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.



3. Complete Binary Tree

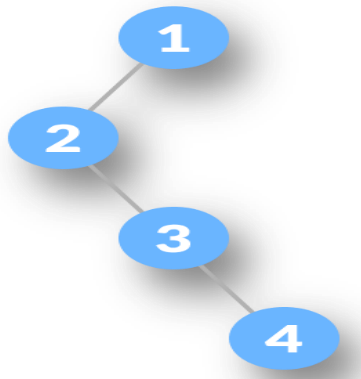
A complete binary tree is just like a full binary tree, but with the major differences

- ❖ All the leaf elements must lean towards the left.
- ❖ The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree



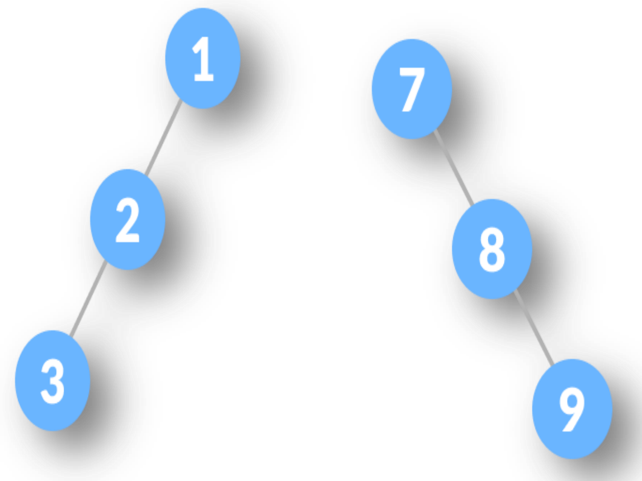
4. Degenerate or Pathological Tree

A degenerate or pathological tree is the tree having a single child either left or right



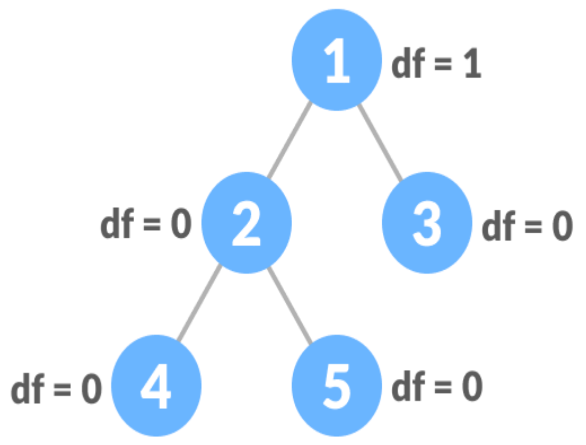
5. Skewed Binary Tree

A skewed binary tree is a pathological/degenerate tree in which the tree is either dominated by the left nodes or the right nodes. Thus, there are two types of skewed binary tree: left-skewed binary tree and right-skewed binary tree

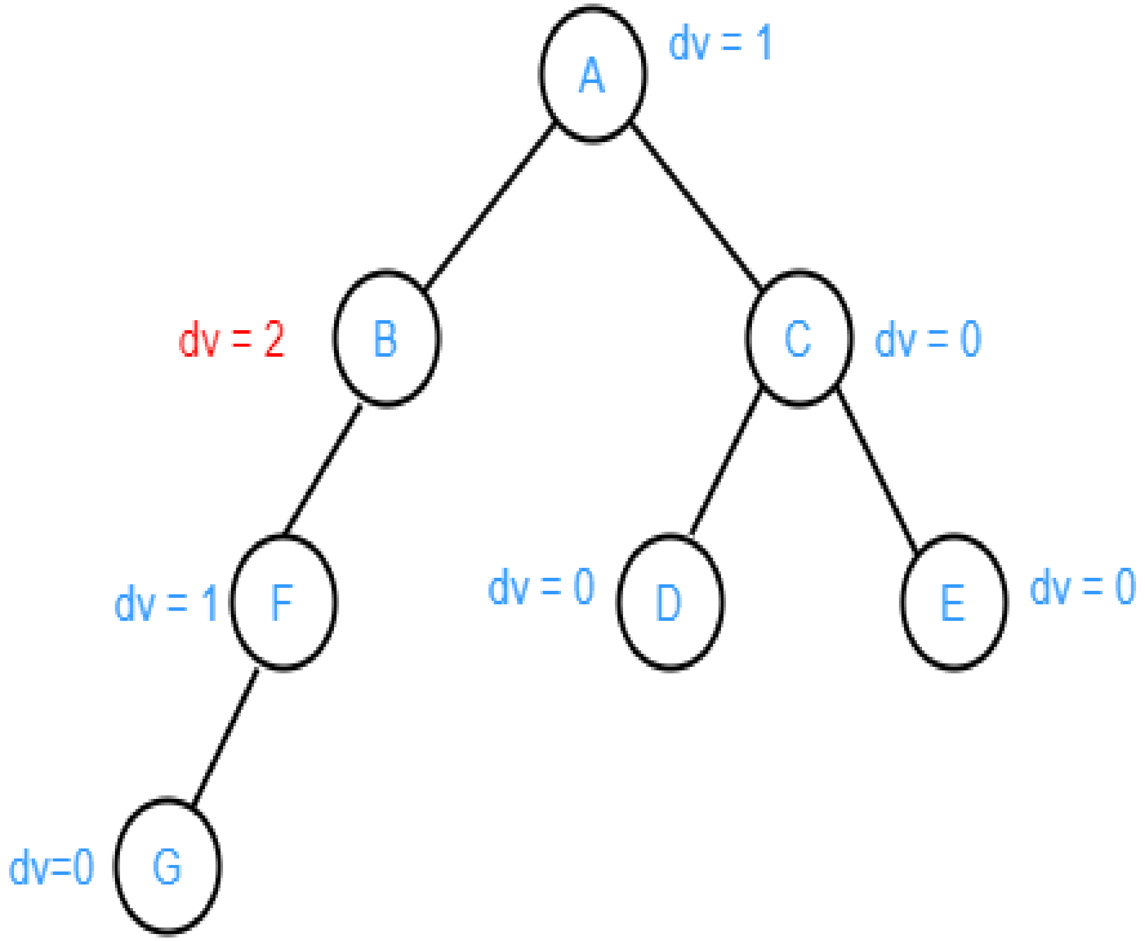


6. Balanced Binary Tree

It is a type of binary tree in which the difference between the height of the left and the right subtree for each node is either 0 or 1.

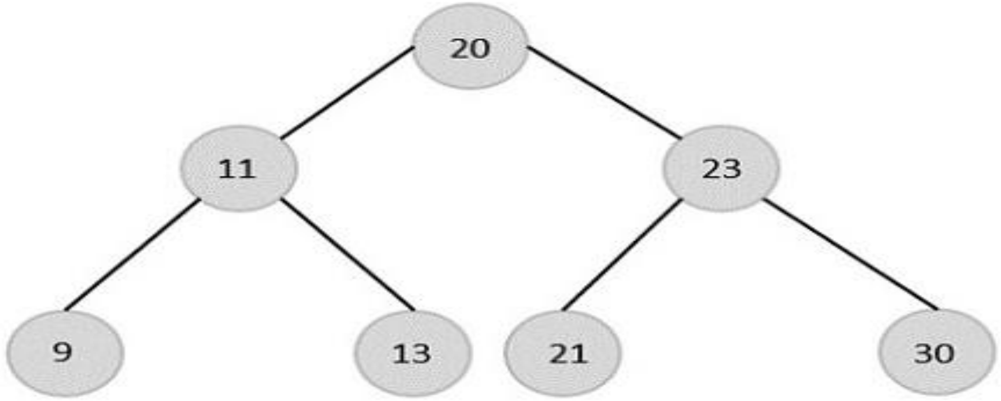


Balanced Binary Search Tree



Binary Search Trees

- Binary Search Trees possess all the properties of Binary Trees including some extra properties of their own
- The data in the Binary Search Trees (BST) is always stored in such a way that the values in the left subtree are always less than the values in the root node and the values in the right subtree are always greater than the values in the root node, i.e. left subtree < root node ≤ right subtree.



Binary Search Tree Data Structure

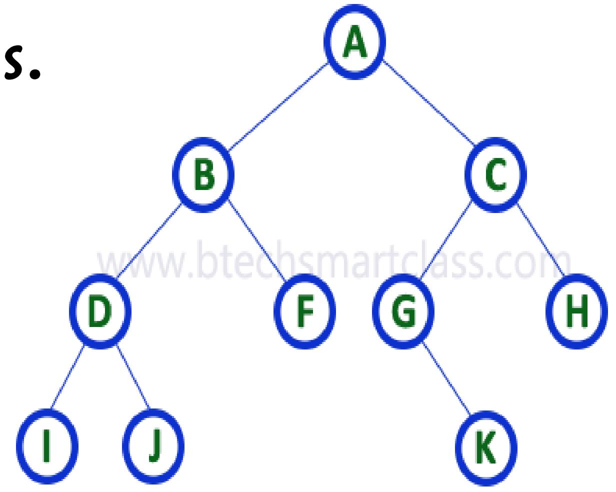
Binary Tree Traversals

- When we wanted to display a binary tree, we need to follow some order in which all the nodes of that binary tree must be displayed. In any binary tree, displaying order of nodes depends on the traversal method.

Displaying (or) visiting order of nodes in a binary tree is called as Binary Tree Traversal.

There are three types of binary tree traversals.

1. In - Order Traversal
2. Pre - Order Traversal
3. Post - Order Traversal

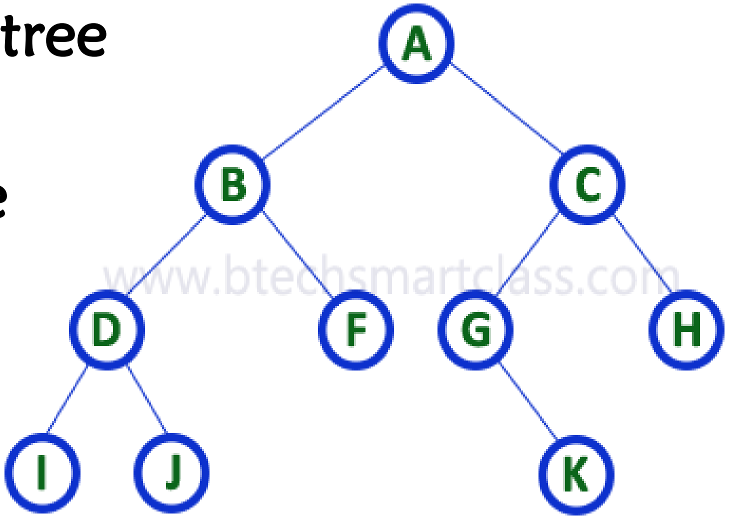


In - Order Traversal (leftChild - root - rightChild)

Traverse the left sub-tree first, and then traverse the root and the right sub-tree respectively. This procedure will be applied to each sub-tree of the tree recursively.

Inorder traversal

1. First, visit all the nodes in the left subtree
2. Then the root node
3. Visit all the nodes in the right subtree



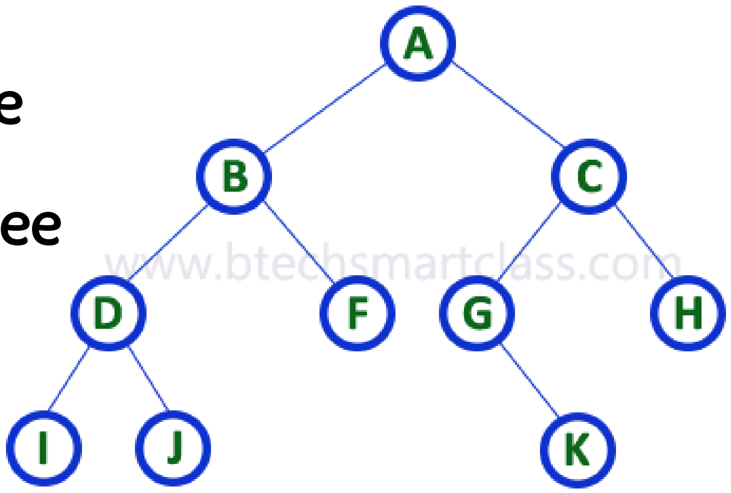
Output In-Order Traversal

I - D - J - B - F - A - G - K - C - H

Pre - Order Traversal (root - leftChild - rightChild)

Traverse the root first then traverse into the left sub-tree and right sub-tree respectively. This procedure will be applied to each sub-tree of the tree recursively.

1. Visit root node
2. Visit all the nodes in the left subtree
3. Visit all the nodes in the right subtree



Output Pre-Order Traversal
A-B-D-I-J-F-C-G-K-H

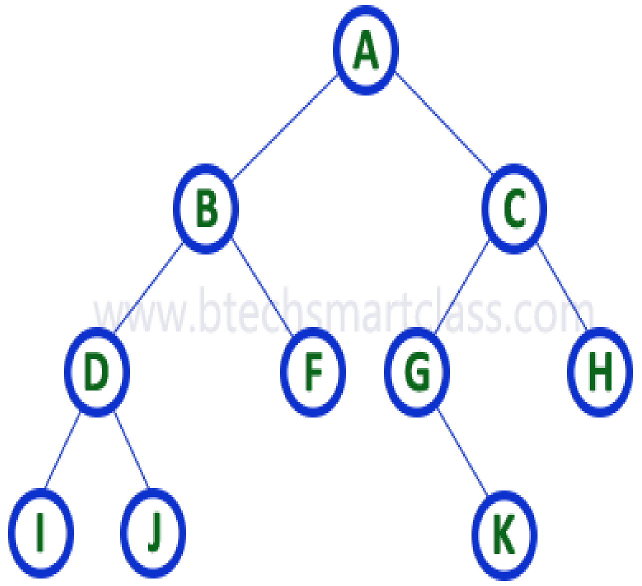
Post - Order Traversal (leftChild - rightChild - root)

Traverse the left sub-tree and then traverse the right sub-tree and root respectively. This procedure will be applied to each sub-tree of the tree recursively.

1.visit all the nodes in the left subtree

2.visit the root node

3.visit all the nodes in the right subtree



Output Post - Order Traversal

I - J - D - F - B - K - G - H - C - A

Operations on a Binary Search Tree

The following operations are performed on a binary search tree...

1. Search
2. Insertion
3. Deletion

In a binary search tree, the search operation is as follows...

Step 1 - Read the search element from the user.

Step 2 - Compare the search element with the value of root node in the tree.

Step 3 - If both are matched, then display "Given node is found!!!" and terminate the function

Step 4 - If both are not matched, then check whether search element is smaller or larger than that node value.

Operations on a Binary Search Tree

Step 5 - If search element is smaller, then continue the search process in left subtree.

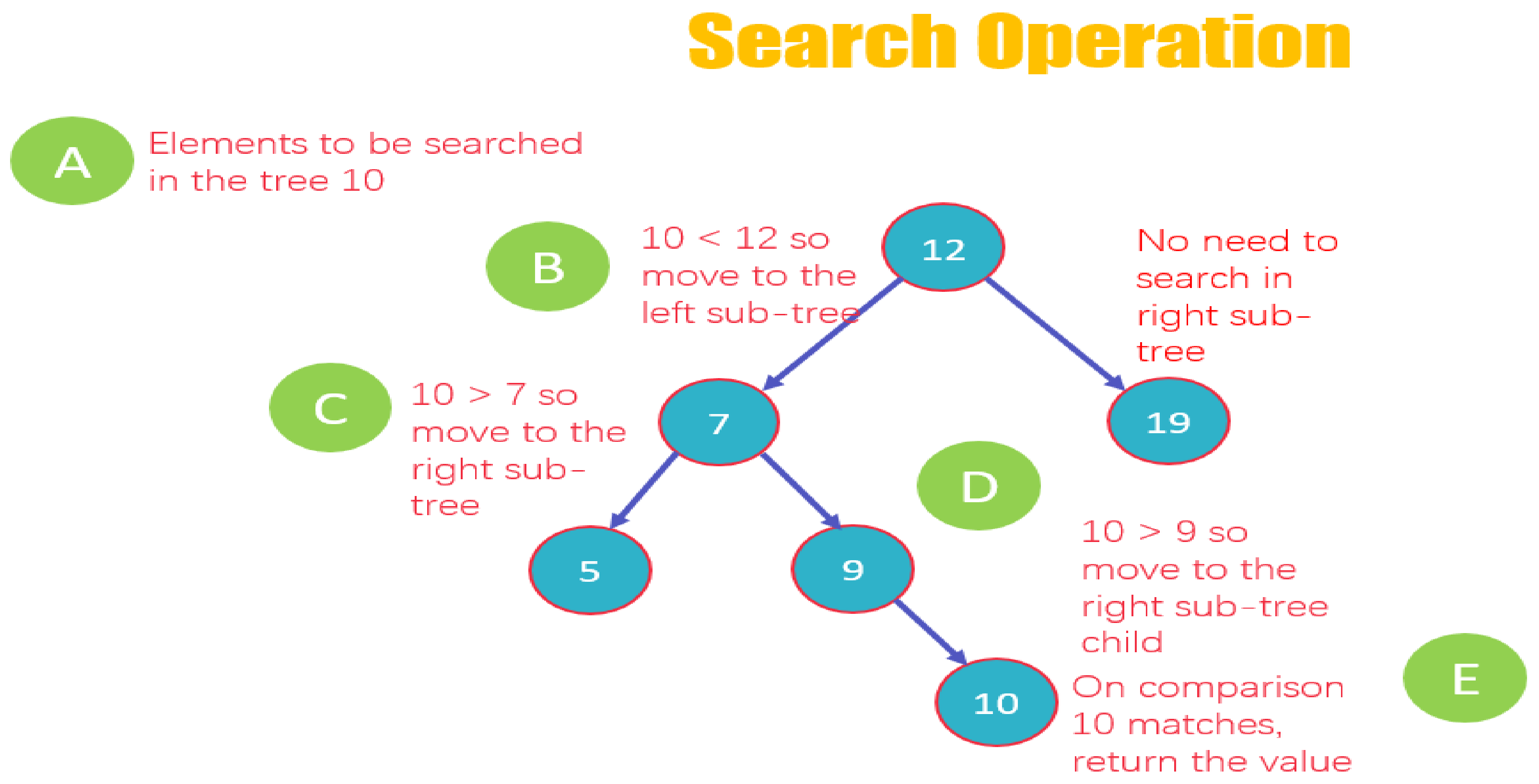
Step 6- If search element is larger, then continue the search process in right subtree.

Step 7 - Repeat the same until we find the exact element or until the search element is compared with the leaf node

Step 8 - If we reach to the node having the value equal to the search value then display "Element is found" and terminate the function. not found" and terminate the function.

Operations on a Binary Search Tree

- Step 9 - If we reach to the leaf node and if it is also not matched with the search element, then display "Element is not found" and terminate the function.


Search Operation
A
Elements to be searched in the tree 10
B
10 < 12 so move to the left sub-tree
12
No need to search in right sub-tree
C
10 > 7 so move to the right sub-tree
7
19
D
5
9
10 > 9 so move to the right sub-tree child
10
On comparison 10 matches, return the value
E

Insertion Operation in BST

In binary search tree, new node is always inserted as a leaf node. The insertion operation is performed as follows...

Step 1 - Create a newNode with given value and set its left and right to NULL.

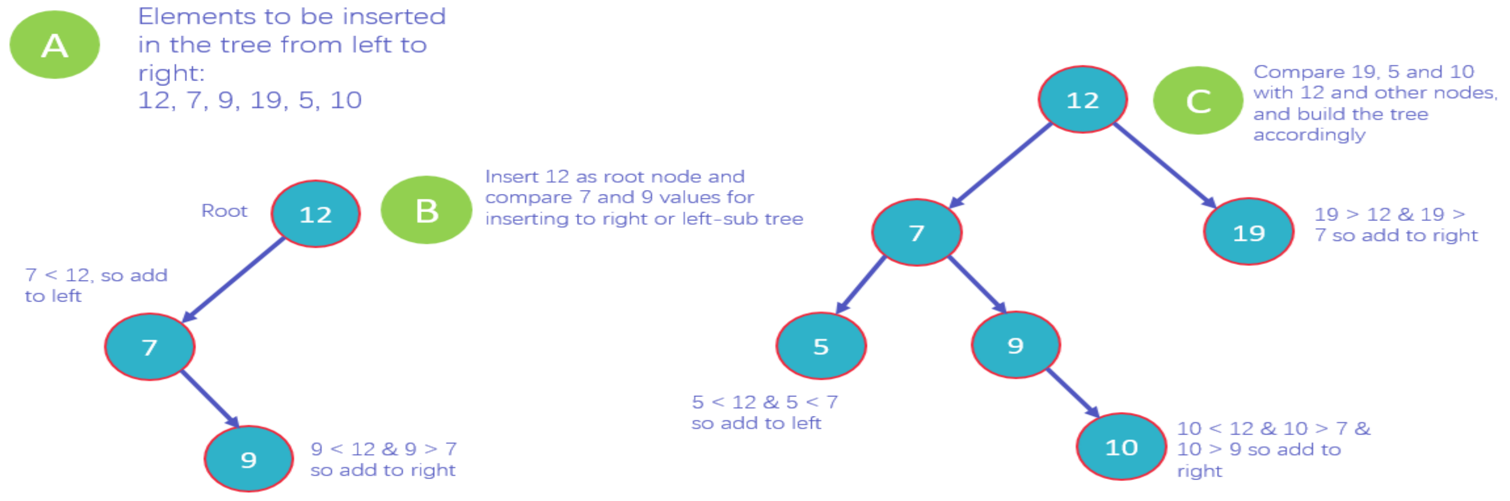
Step 2 - Check whether tree is Empty.

Step 3 - If the tree is Empty, then set root to newNode.

Step 4 - If the tree is Not Empty, then check whether the value of newNode is smaller or larger than the node (here it is root node).

Step 5 - If newNode is smaller than or equal to the node then move to its left child. If newNode is larger than the node then move to its right child.
Step 6- Repeat the above steps until we reach to the leaf node (i.e., reaches to NULL).
Step 7 - After reaching the leaf node, insert the newNode as left child if the newNode is smaller or equal to that leaf node or else insert it as right child.

Insert Operation


A
Elements to be inserted in the tree from left to right:
12, 7, 9, 19, 5, 10
B
Insert 12 as root node and compare 7 and 9 values for inserting to right or left-sub tree
Root
12
7 < 12, so add to left
7
9
9 < 12 & 9 > 7 so add to right
C
Compare 19, 5 and 10 with 12 and other nodes, and build the tree accordingly
12
7
19
19 > 12 & 19 > 7 so add to right
5
9
5 < 12 & 5 < 7 so add to left
10
10 < 12 & 10 > 7 & 10 > 9 so add to right

Deletion Operation in BST

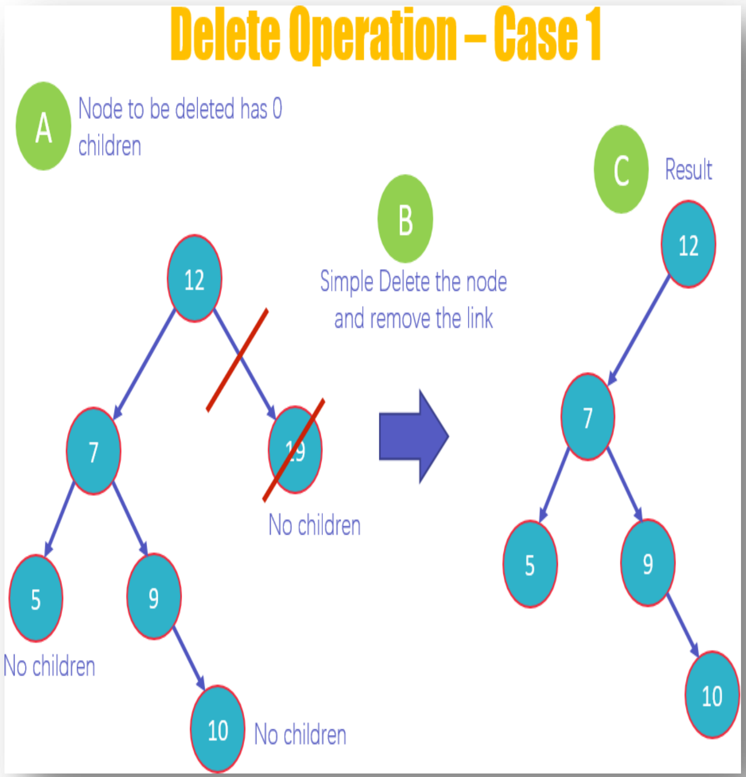
Deleting a node from Binary search tree includes following three cases...

- Case 1: Deleting a Leaf node (A node with no children)
- Case 2: Deleting a node with one child
- Case 3: Deleting a node with two children

Case 1: Deleting a leaf node

We use the following steps to delete a leaf node from BST...

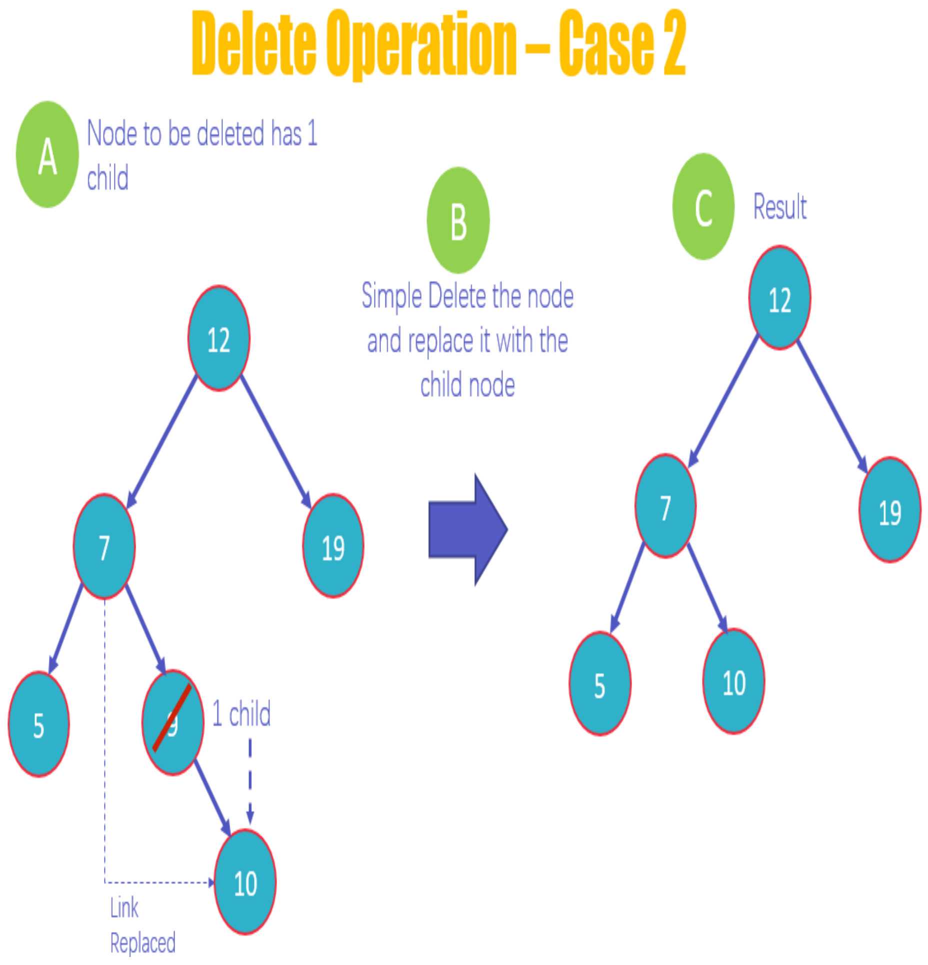
- Step 1 - Find the node to be deleted using search operation
- Step 2 - Delete the node using free function (If it is a leaf) and terminate the function.


Delete Operation – Case 1
A
Node to be deleted has 0 children
B
Simple Delete the node and remove the link
C
Result
12
7
19
5
9
10
No children
No children
No children
12
7
5
9
10

Case 2: Deleting a node with one child

We use the following steps to delete a node with one child from BST...

- Step 1 - Find the node to be deleted using search operation
- Step 2 - If it has only one child then create a link between its parent node and child node.
- Step 3 - Delete the node using free function and terminate the function.



Delete Operation – Case 2
A
Node to be deleted has 1 child
B
Simple Delete the node and replace it with the child node
C
Result
12
7
19
5
9
1 child
10
Link Replaced
12
7
19
5
10

Case 3: Deleting a node with two children

We use the following steps to delete a node with two children from BST...

- Step 1 - Find the node to be deleted using search operation
- Step 2 - If it has two children, then find the largest node in its left subtree (OR) the smallest node in its right subtree.
- Step 3 - Swap both deleting node and node which is found in the above step.
- Step 4 - Then check whether deleting node came to case 1 or case 2 or else goto step 2
- Step 5 - If it comes to case 1, then delete using case 1 logic.
- Step 6- If it comes to case 2, then delete using case 2 logic.
- Step 7 - Repeat the same process until the node is deleted from the tree.

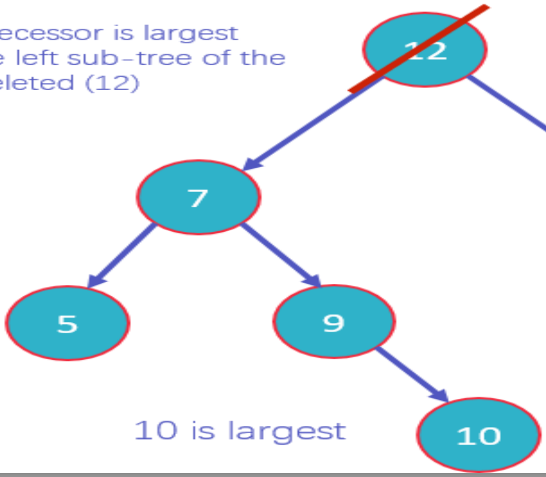
Delete Operation – Case 3 (a)

A

Node to be deleted has 2 child
Replace Situation: In Order Predecessor

In Order predecessor is largest element in the left sub-tree of the node to be deleted (12)

B



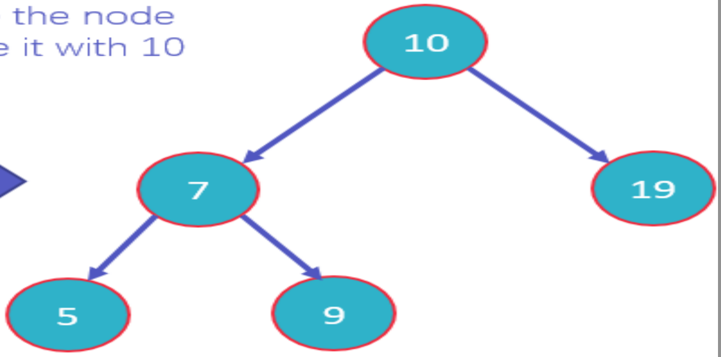
10 is largest

C

Simple Delete the node 12 and replace it with 10

D

Result



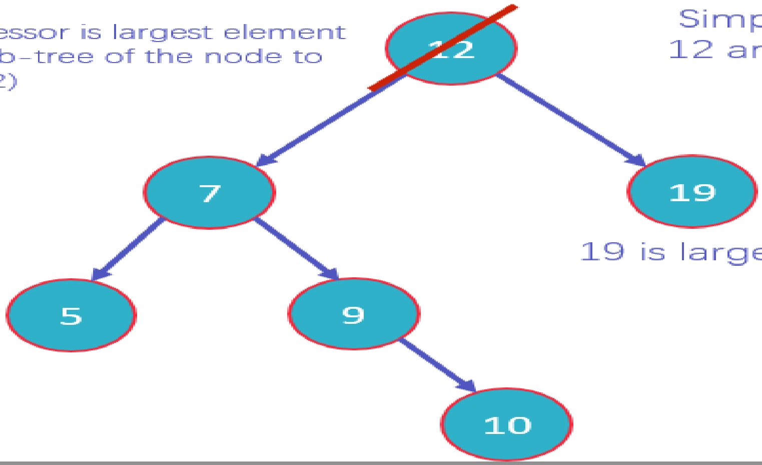
Delete Operation – Case 3 (b)

A

Node to be deleted has 2 child
Replace Situation: In Order Successor

In Order successor is largest element in the right sub-tree of the node to be deleted (12)

B



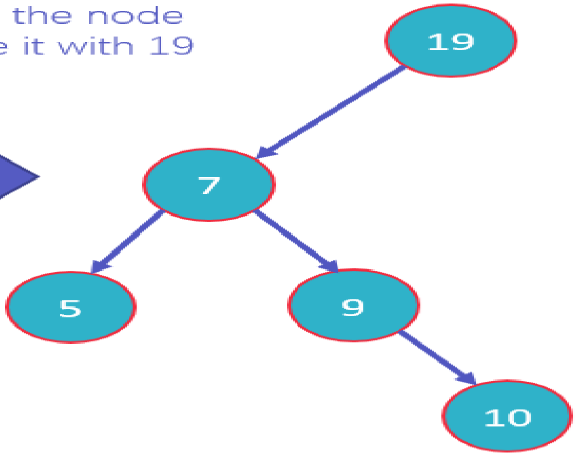
19 is largest

C

Simple Delete the node 12 and replace it with 19

D

Result



Applications of Tree:

File Systems: Each folder or directory is a node in the tree, and files are the leaves.

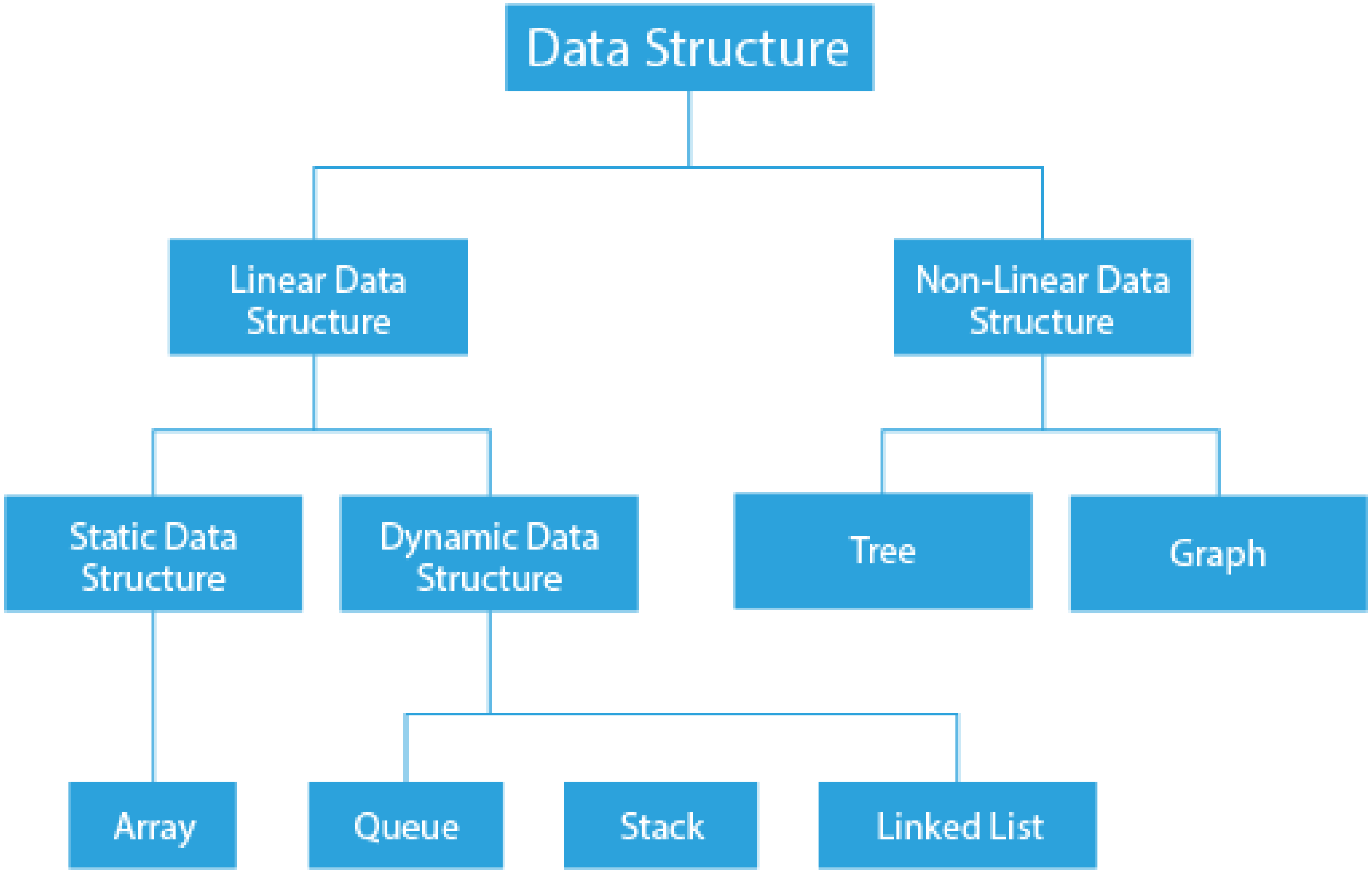
Database Indexing: Many databases use trees to index their data

Compiler Design: The syntax of programming languages is often defined using a tree structure called a parse tree. This is used by compilers to understand the structure of the code and generate machine code from it.

Artificial Intelligence: Decision trees are often used in artificial intelligence to make decisions based on a series of criteria.

Trees are used in several games like moves in chess.

Question ?


Data Structure
Linear Data Structure
Non-Linear Data Structure
Static Data Structure
Dynamic Data Structure
Tree
Graph
Array
Queue
Stack
Linked List