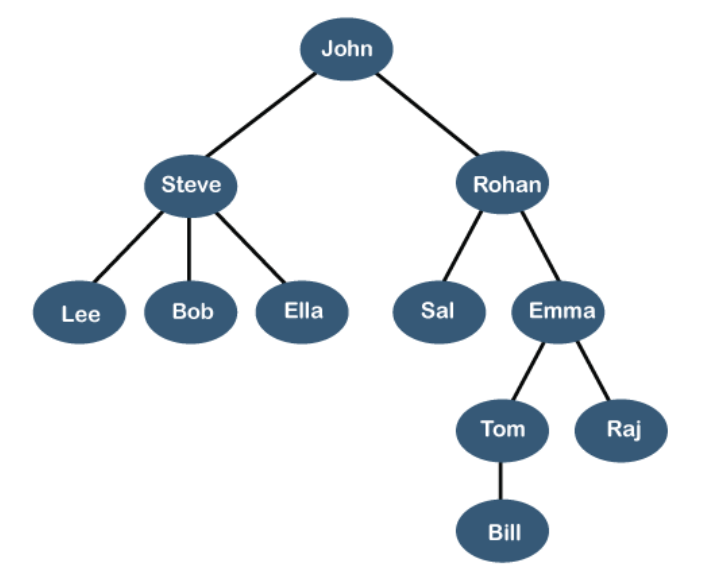
# https://www.javatpoint.com/tree

# Why tree data structure is used?

A tree is also one of the data structures that represent hierarchical data.

Suppose we want to show the employees and their positions in the hierarchical form then it can be represented as shown below:

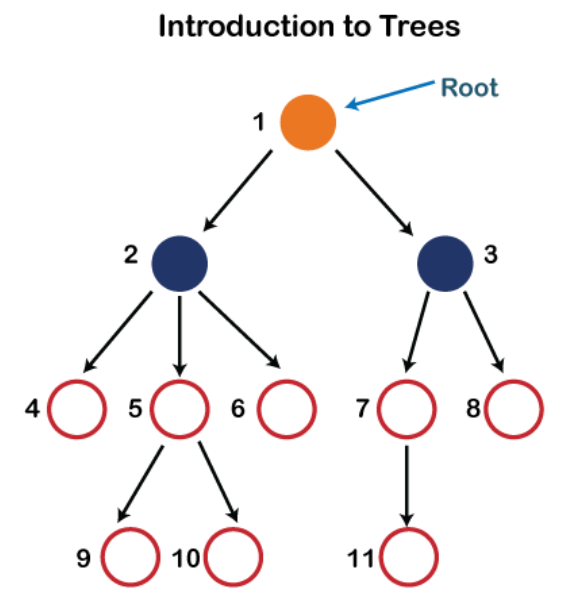


The above tree shows the organization hierarchy of some company. In the above structure, john is the CEO of the company, and John has two direct reports named as Steve and Rohan. Steve has three direct reports named Lee, Bob, Ella where Steve is a manager. Bob has two direct reports named Sal and Emma. Emma has two direct reports named Tom and Raj. Tom has one direct report named Bill. This logical structure is known as a Tree. Its structure is like the real tree, so it is named a Tree. In this structure, the root is at the top, and its branches are moving in a downward direction. Therefore, we can say that the Tree data structure is an efficient way of storing the data in a hierarchical way.

# Important points about tree data structure?

* Tree is collection “**nodes”** that are linked together to represent or simulate hierarchy.
* A tree data structure is a non-linear data structure because it does not store in a sequential manner. It is a hierarchical structure as elements in a Tree are arranged in multiple levels.
* In the Tree data structure, the topmost node is known as a root node. Each node contains some data, and data can be of any type. In the above tree structure, the node contains the name of the employee, so the type of data would be a string.
* Each node contains some data and the link or reference of other nodes that can be called children.

# Basic Terms used in tree data structure?



In the above structure, each node is labeled with some number. Each arrow shown in the above figure is known as a link between the two nodes.

**Root**: The root node is the topmost node in the tree hierarchy. In other words, the root node is the one that doesn't have any parent. In the above structure, node numbered 1 is the root node of the tree. If a node is directly linked to some other node, it would be called a parent-child relationship.

**Child node**: If the node is a descendant of any node, then the node is known as a child node.

**Parent**: If the node contains any sub-node, then that node is said to be the parent of that sub-node.

**Sibling**: The nodes that have the same parent are known as siblings.

**Leaf Node**: The node of the tree, which doesn't have any child node, is called a leaf node. A leaf node is the bottom-most node of the tree. There can be any number of leaf nodes present in a general tree. Leaf nodes can also be called external nodes.

**Internal nodes**: A node has at least one child node known as an internal.

**Ancestor node**: An ancestor of a node is any predecessor node on a path from the root to that node. The root node doesn't have any ancestors. In the tree shown in the above image, nodes 1, 2, and 5 are the ancestors of node 10.

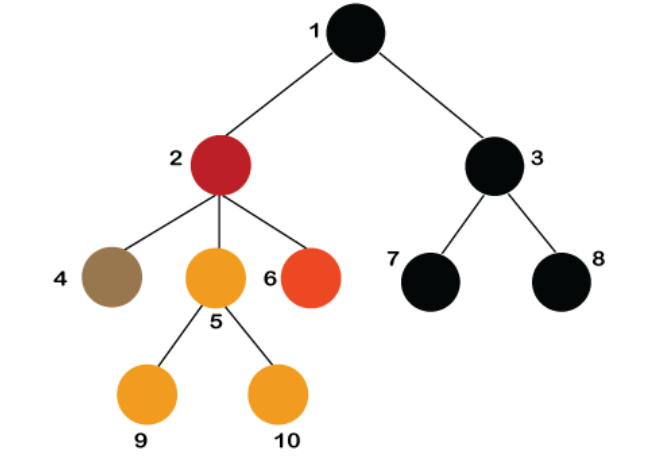
**Descendant**: The immediate successor of the given node is known as a descendant of a node. In the above figure, 10 is the descendant of node 5.

# Properties of Tree data structure?

**Recursive data structure:**

The tree is also known as a **recursive data structure**. A tree can be defined as recursively because the distinguished node in a tree data structure is known as a **root node**.

The root node of the tree contains a link to all the roots of its subtrees. The left subtree is shown in the yellow color in the below figure, and the right subtree is shown in the red color. The left subtree can be further split into subtrees shown in three different colors. Recursion means reducing something in a self-similar manner. So, this recursive property of the tree data structure is implemented in various applications.



**Number of edges**:

If there are n nodes, then there would n-1 edges. Each arrow in the structure represents the link or path. Each node, except the root node, will have at least one incoming link known as an edge. There would be one link for the parent-child relationship.

**Depth of node x:**

number of edges between the root node and the node x.

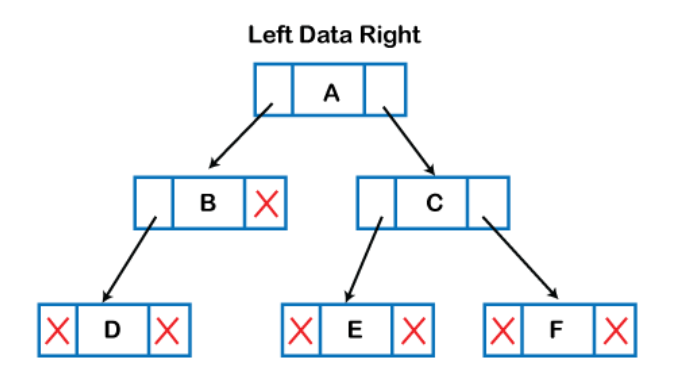
length of the path from the root to the node x.

The root node has 0 depth.

**Height of node x:**

longest path from the node x to the leaf node.

# Implementation of the tree.



The above figure shows the representation of the tree data structure in the memory. In the above structure, the node contains three fields. The second field stores the data; the first field stores the address of the left child, and the third field stores the address of the right child.

The above structure can only be defined for the binary trees because the binary tree can have utmost two children, and generic trees can have more than two children. The structure of the node for generic trees would be different as compared to the binary tree.

# Applications of trees

**Storing naturally hierarchical data**: Trees are used to store the data in the hierarchical structure. For example, the file system. The file system stored on the disc drive, the file and folder are in the form of the naturally hierarchical data and stored in the form of trees.

**Organize data**: It is used to organize data for efficient insertion, deletion and searching. For example, a binary tree has a logN time for searching an element.

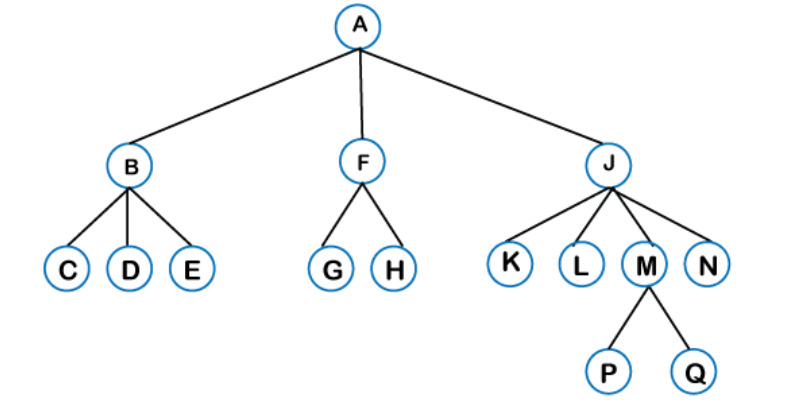
**B-Tree and B+Tree**: B-Tree and B+Tree are the tree data structures used to implement indexing in databases.

**Trie**: It is a special kind of tree that is used to store the dictionary. It is a fast and efficient way for dynamic spell checking.

**Routing table**: The tree data structure is also used to store the data in routing tables in the routers.

# Types of tree data structures:

**General tree**: The general tree is one of the types of tree data structure. In the general tree, a node can have either 0 or maximum n number of nodes. There is no restriction imposed on the degree of the node (the number of nodes that a node can contain). The topmost node in a general tree is known as a root node. The children of the parent node are known as subtrees.

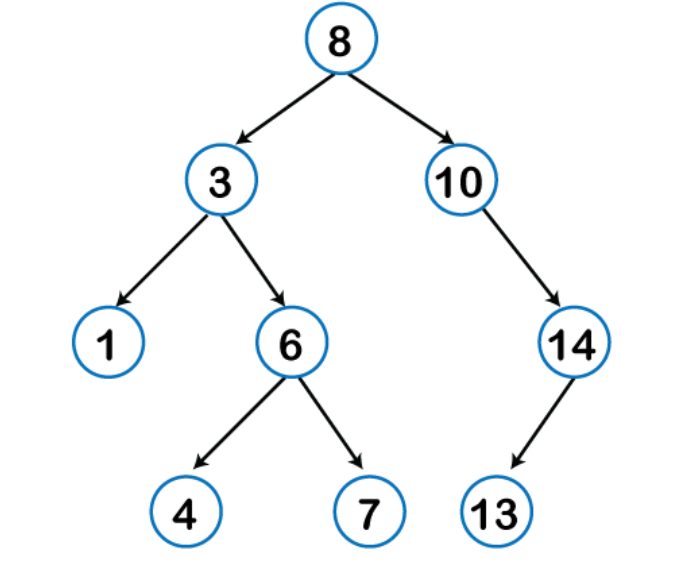


There can be n number of subtrees in a general tree. In the general tree, the subtrees are unordered as the nodes in the subtree cannot be ordered.

There can be n number of subtrees in a general tree. In the general tree, the subtrees are unordered as the nodes in the subtree cannot be ordered.

Every non-empty tree has a downward edge, and these edges are connected to the nodes known as child nodes. The root node is labeled with level 0. The nodes that have the same parent are known as siblings

**Binary tree**: Here, binary name itself suggests two numbers, i.e., 0 and 1. In a binary tree, each node in a tree can have utmost two child nodes. Here, utmost means whether the node has 0 nodes, 1 node or 2 nodes.



**Binary Search Tree:**

* Binary Search tree is type of binary tree
* In binary tree the nodes are arranged in a specific order. This is also called ordered binary tree.
* In a binary search tree, the value of all the nodes in the left sub-tree is less than the value of the root.
* In a binary search tree, the value of all the nodes in the left sub-tree is less than the value of the root.
* Similarly, value of all the nodes in the right sub-tree is greater than or equal to the value of the root.
* This rule will be recursively applied to all the left and right sub-trees of the root.

