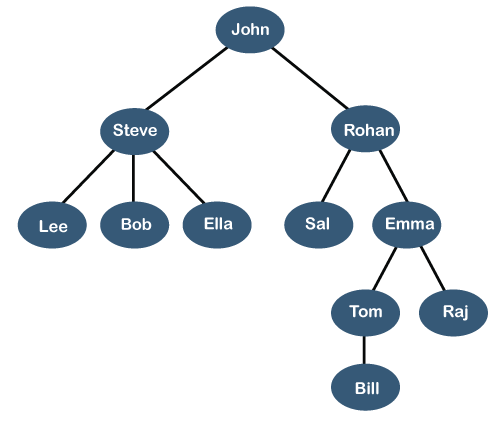
Tree Data Structure

* .

***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 particular logical structure is known as a ***Tree***. Its structure is similar to 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.

31.7M

732

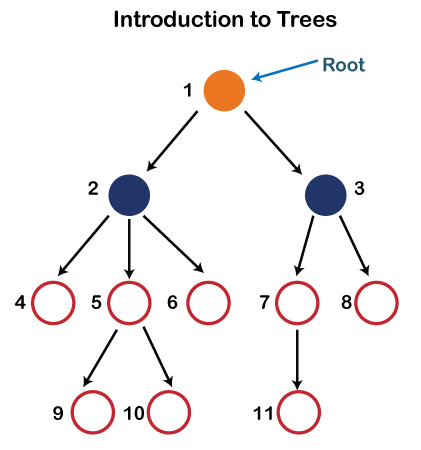
Features of Java - Javatpoint

**Let's understand some key points of the Tree data structure.**

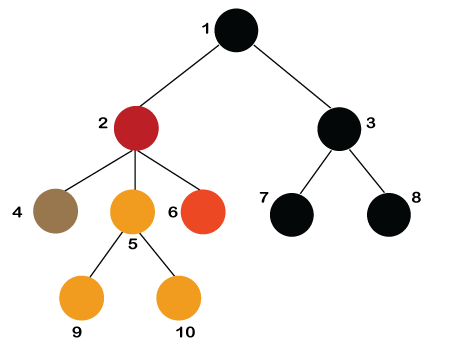
* A tree data structure is defined as a collection of objects or entities known as nodes that are linked together to represent or simulate hierarchy.
* A tree data structure is a non-linear data structure because it does not store in a sequential manner. It is a hierarchical structure as elements in 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.

**Some basic terms used in Tree data structure.**

Let's consider the tree structure, which is shown below:



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 atleast 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.
* .  
  
* **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 atleast one incoming link known as an edge. There would be one link for the parent-child relationship.
* **Depth of node x:** The depth of node x can be defined as the length of the path from the root to the node x. One edge contributes one-unit length in the path. So, the depth of node x can also be defined as the number of edges between the root node and the node x. The root node has 0 depth.
* **Height of node x:** The height of node x can be defined as the longest path from the node x to the leaf node.

Based on the properties of the Tree data structure, trees are classified into various categories.

Implementation of Tree

The tree data structure can be created by creating the nodes dynamically with the help of the pointers. The tree in the memory can be represented as shown below:



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.

In programming, the structure of a node can be defined as:

1. struct node
2. {
3. **int** data;
4. struct node \*left;
5. struct node \*right;
6. }

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.