Tree :

A tree is a non-linear and hierarchical data structure which consists of collection of nodes such that each node stores a value and the reference to the other node.

What is node?

Each element in the tree data structure is called node, where each node contains a left reference and a right reference. Each node is connected using an edge to represent the hierarchical connections.

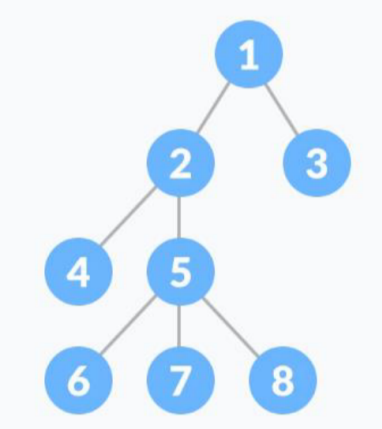
Binary tree:

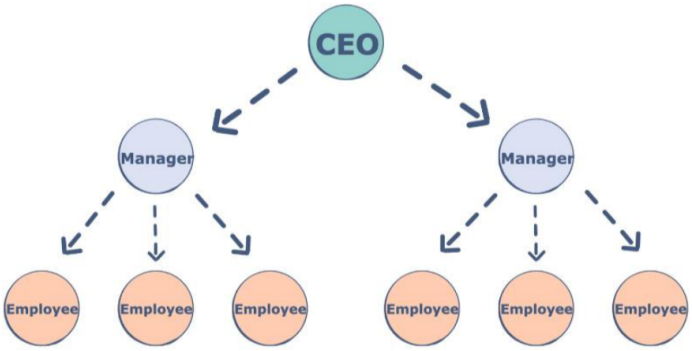
Binary tree is defined as a tree data structure with at most 2 children. Each element in a binary tree can have only 2 children, and we typically name them as left and right node.

Following are the important terms with respect to tree.

- **Path** − Path refers to the sequence of nodes along the edges of a tree.
- **Root** − The node at the top of the tree is called root. There is only one root per tree and one path from the root node to any node.
- **Parent** − Any node except the root node has one edge upward to a node called parent.
- **Child** − The node below a given node connected by its edge downward is called its child node.
- **Leaf** − The node which does not have any child node is called the leaf node.
- **Subtree** − Subtree represents the descendants of a node.
- **Visiting** − Visiting refers to checking the value of a node when control is on the node.
- **Traversing** − Traversing means passing through nodes in a specific order.
- **Levels** − Level of a node represents the generation of a node. If the root node is at level 0, then its next child node is at level 1, its grandchild is at level 2, and so on.
- **keys** − Key represents a value of a node based on which a search operation is to be carried out for a node.

Hierarchy of tree data structure:





1.Root node:

root node is a node which do not have any incoming edges or it is the top most node.(it is the starting point of the tree).

2.Parent node:

node with at least one outgoing edge is called

3.Child node:

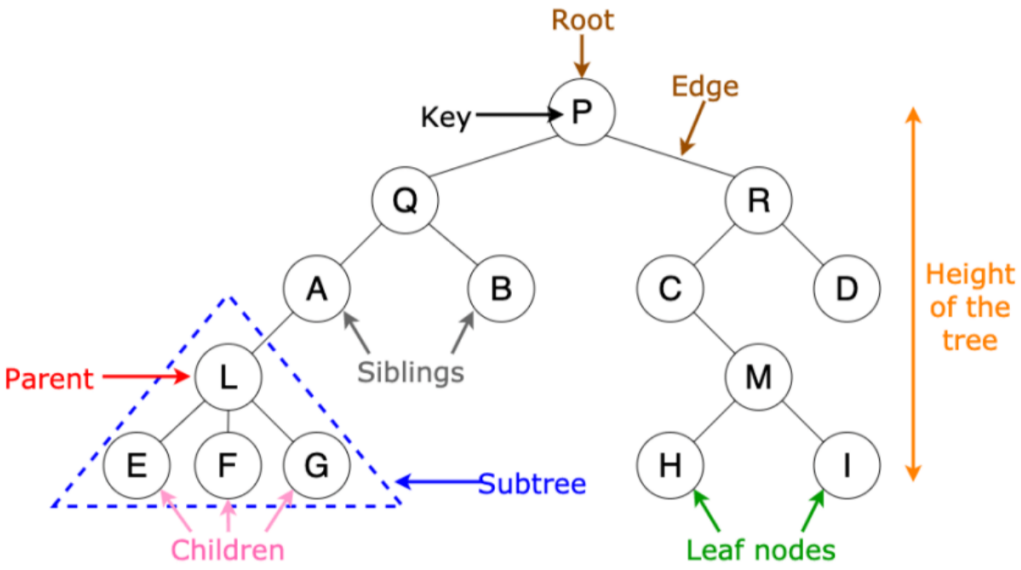
a node with at least one incoming is called child node(from parent node).

4.Internal node:

a node with at least one child node is called internal node.

5.Sibling nodes:

when two nodes have the same parent then the two nodes are said to be siblings.



Height of a tree:

The highest number of nodes from the root node to the deepest leaf is called the height of a tree.

Depth of a tree:

The highest number of nodes from the tree's node to the root node is called the depth of a tree.

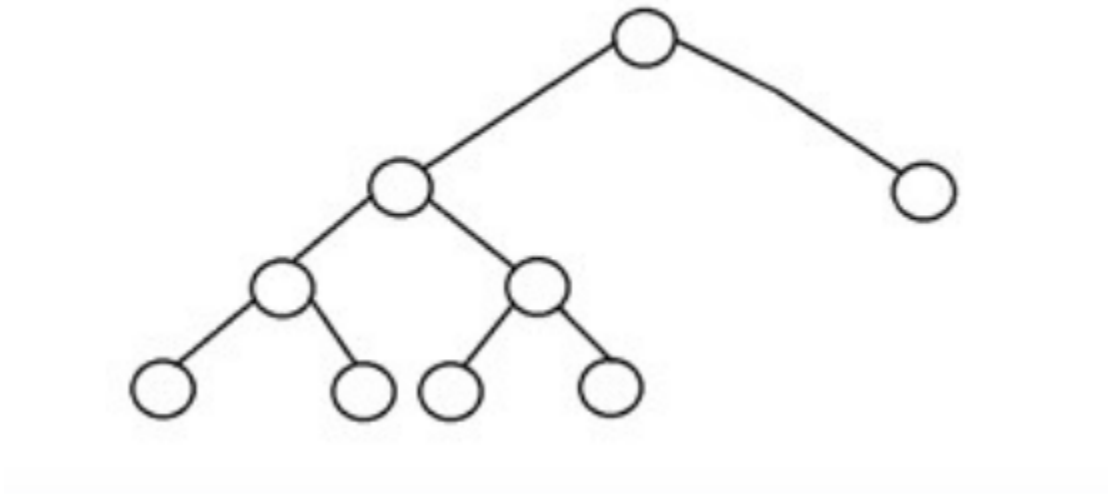
Components of a tree:

A binary tree has three elements associated to it. They are:

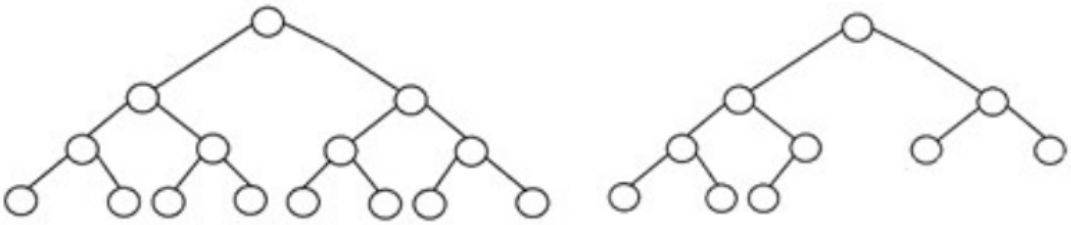
1. data elements
2. pointer to the left node
3. pointer to the right node.

Types of binary tree:

1. Full binary tree:
    a. It is a special kind of binary tree that has two children or zero children.
    b. In other words, each node in a full binary tree should have two nodes except the external node.
    c.



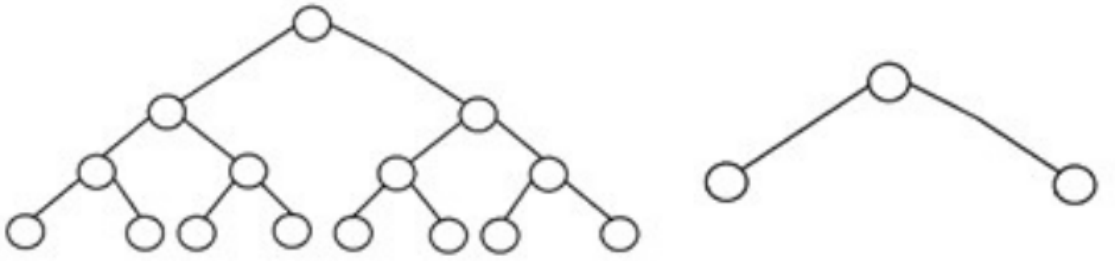
2. Complete binary tree:
    a. In this kind of binary tree all the nodes except the external node should be filled completely and in the leaf node, every node should possibly reside on the left side.
    b.



3. perfect binary tree:

   a. In a binary tree, when all the internal nodes have strictly two nodes and the height of every external node should be equal to every external node.

   b.



Basic operations in a binary tree:

1. inserting an element
2. removing an element
3. searching for an element
4. traversing through the element

Auxiliary operations in a binary tree:

1.finding the height of the binary tree

2.finding the level of the tree

3.finding the size of the tree

In order to traverse the tree, there are three ways:

1. pre-order:

a. during the traversal, we will visit the root, traverse through the left subtree and finally we will traverse through the right subtree.

b. algorithm for pre-order:

   i. *Visit the root.*

   ii. *Traverse the left subtree, i.e., call Preorder(left->subtree)*

   iii. *Traverse the right subtree, i.e., call Preorder(right->subtree)*

c. in pre- order, we will get the copy of the tree in same order.

2. in-order

   a. during the traversal, we will call the left subtree, visit the root and finally, we will call the right left subtree.

   b. algorithm for in-order:

      i. *Traverse the left subtree, i.e., call Inorder(left->subtree)*

      ii. *Visit the root.*

      iii. *Traverse the right subtree, i.e., call Inorder(right->subtree)*

   c. in order gives nodes in non-decreasing order

3. post-order:

   a. during the traversal, we will traverse through the left subtree, followed by the right subtree and finally we will visit the root.

   b. algorithm for post-order:

      i. *Traverse the left subtree, i.e., call Postorder(left->subtree)*

      ii. *Traverse the right subtree, i.e., call Postorder(right->subtree)*

      iii. *Visit the root*

   c. post order traversal is used to delete the tree. It is also useful in order to get the post expression of the expression tree.

Code using linked list and recursion:

```
// class to create nodes
class Node {
int key;
Node left, right;

public Node(int item) {
key = item;
left = right = null;
}
}

class BinaryTree {
Node root;

// Traverse tree
public void traverseTree(Node node) {
if (node != null) {
traverseTree(node.left);
System.out.print(" " + node.key);
traverseTree(node.right);
}
}

public static void main(String[] args) {

// create an object of BinaryTree
BinaryTree tree = new BinaryTree();

// create nodes of the tree
tree.root = new Node(1);
tree.root.left = new Node(2);
tree.root.right = new Node(3);
tree.root.left.left = new Node(4);

System.out.print("\\nBinary Tree: ");
tree.traverseTree(tree.root);

}
}
```

Binary search tree:

A binary search tree is binary tree with the following properties:

1. the left subtree of the node should contain keys whose value is less than the node
2. the right subtree of the node should contain keys whose value is more than the node
3. the left and the right subtree should be binary search tree.