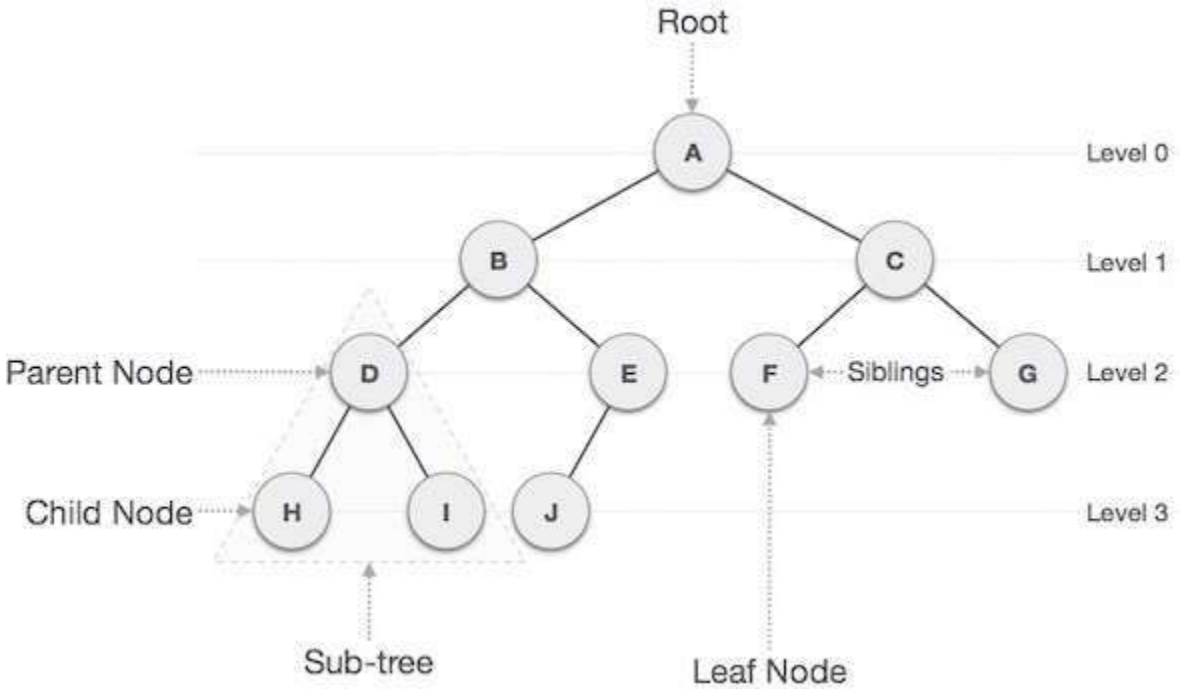
Tree-Binary Tree

Trees

- A tree is a finite set of one or more nodes such that:
- There is a specially designated node called the root.
- The remaining nodes are partitioned into n>=0 disjoint sets T1, ..., Tn, where each of these sets is a tree.
- We call T1, ..., Tn the subtrees of the root.

Tree Structure



Basic Terminologies

- **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.
- **Sibling:** The nodes with common parent are called siblings.

Basic Terminologies

- **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.

Binary Tree

- A binary tree is a finite set of nodes that is either empty or consists of a root and two disjoint binary trees called *the left subtree* and *the right subtree*.

A Tree node contains following parts.

1. Data
2. Pointer to left child
3. Pointer to right child

Tree Representation:

```
struct node
{
  int data;
  struct node *left;
  struct node *right;
};
```

Types of Binary Trees

- ✓ Full Binary Tree: A Binary Tree is full if every node has 0 or 2 children.
- ✓ Complete Binary Tree: A complete binary tree is full binary tree in which all leaves are at the same depth.
- ✓ Left and right skewed Trees:
- ➢ Left skewed tree: Tree in which node is attached as a left child of parent node.
- ➢ Right skewed tree: Tree in which node is attached as a right child of parent node.

Applications of Trees

1. Manipulate hierarchical data.
2. Make information easy to search (tree traversal).
3. Manipulate sorted lists of data.
4. As a workflow for compositing digital images for visual effects.
5. Router algorithms
6. Form of a multi-stage decision-making