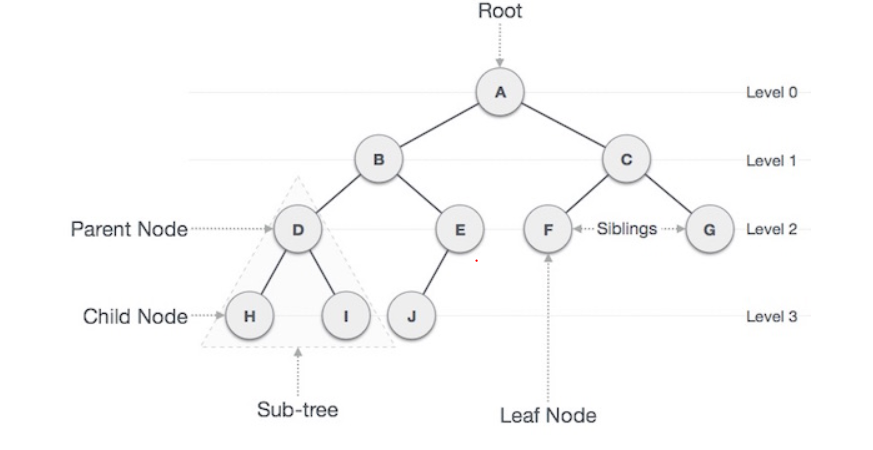
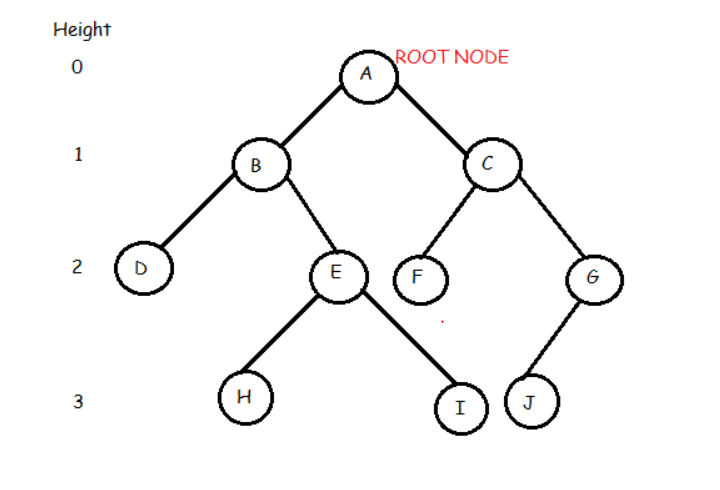
**Trees:**

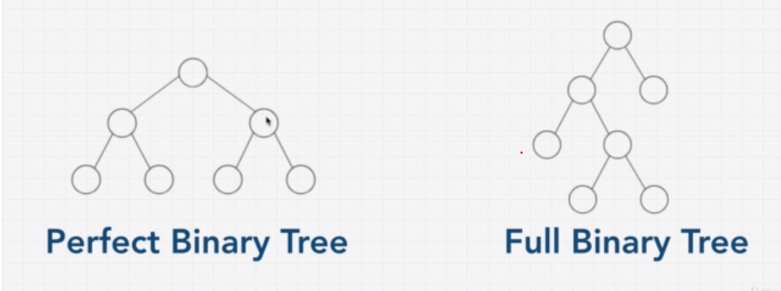


* **Path** − Path refers to the sequence of nodes along the edges of a tree.
* **Root** − The node at the top of the tree is called root. There is only one root per tree and one path from the root node to any node.
* **Parent** − Any node except the root node has one edge upward to a node called parent.
* **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.

**Binary Trees:**

A binary tree is a hierarchical data structure in which each node has at most two children generally referred as left child and right child.





**Perfect binary tree:**

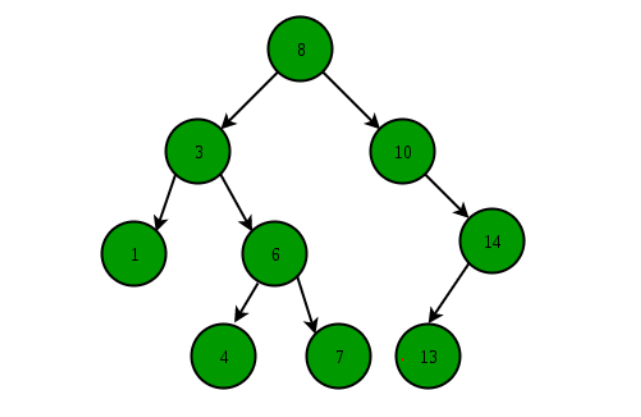
|  |  |  |  |
| --- | --- | --- | --- |
| Level | No of nodes |  |  |
| 0 | 1 | 2^0 |  |
| 1 | 2 | 2^1 |  |
| 2 | 4 | 2^2 |  |
| 3 | 8 | 2^3 |  |
| 4 | 16 | 2^4 |  |
| 5 | 32 | 2^5 |  |
| n | … | 2^n |  |

No of nodes = 2^n -1 where n is height of tree

Log (nodes) = height /steps

BIG O is O(Log N)

**Binary Search Tree:**



* The left subtree of a node contains only nodes with keys lesser than the node’s key.
* The right subtree of a node contains only nodes with keys greater than the node’s key.
* The left and right subtree each must also be a binary search tree.

**Unbalanced Binary Search tree:**

