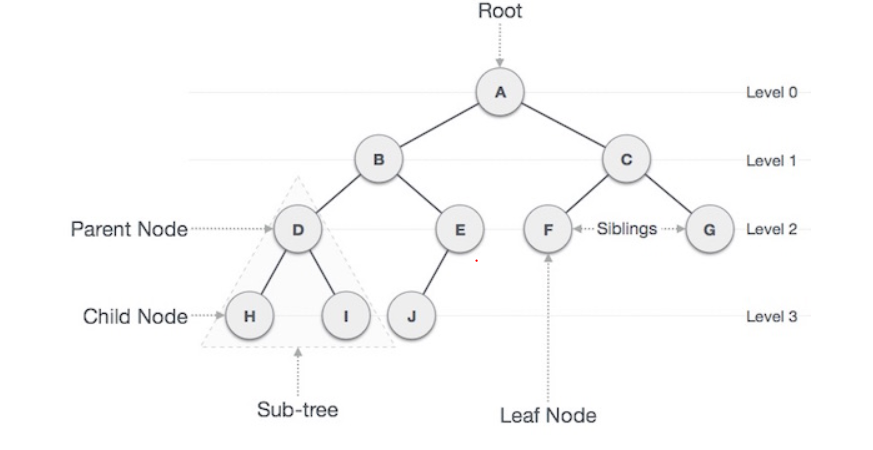
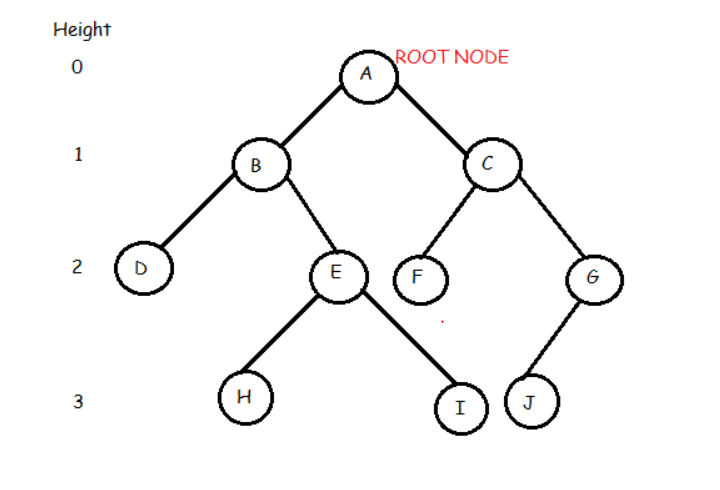
**Trees:**

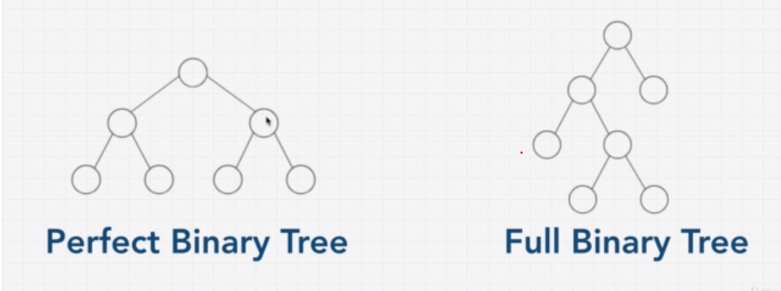


* **Path** − Path refers to the sequence of nodes along the edges of a tree.
* **Root** − The node at the top of the tree is called root. There is only one root per tree and one path from the root node to any node.
* **Parent** − Any node except the root node has one edge upward to a node called parent.
* **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 in a tree with height n is 2^n -1

In a Binary Tree with N nodes, minimum possible height or minimum number of levels is  Log2(N+1)

***A Binary Tree with L leaves has at least   Log2L + 1   levels***

***In Binary tree where every node has 0 or 2 children, number of leaf nodes is always one more than nodes with two children.***

**Types of Binary Tree**

**Full Binary Tree:**

A Binary Tree is a full binary tree if every node has 0 or 2 children.

18

/ \

15 30

/ \ / \

40 50 100 40

18

/ \

15 20

/ \

40 50

/ \

30 50

18

/ \

40 30

/ \

100 40

***In a Full Binary Tree, number of leaf nodes is the number of internal nodes plus 1***   
       L = I + 1   
Where L = Number of leaf nodes, I = Number of internal nodes

**Complete Binary Tree:** A Binary Tree is a complete Binary Tree if all the levels are completely filled except possibly the last level and the last level has all keys as left as possible

18

/ \

15 30

/ \ / \

40 50 100 40

18

/ \

15 30

/ \ / \

40 50 100 40

/ \ /

8 7 9

**Perfect Binary Tree** A Binary tree is a Perfect Binary Tree in which all the internal nodes have two children and all leaf nodes are at the same level.   
The following are the examples of Perfect Binary Trees.

18

/ \

15 30

/ \ / \

40 50 100 40

18

/ \

15 30

**A degenerate (or pathological) tree**A Tree where every internal node has one child. Such trees are performance-wise same as linked list.

10

/

20

\

30

\

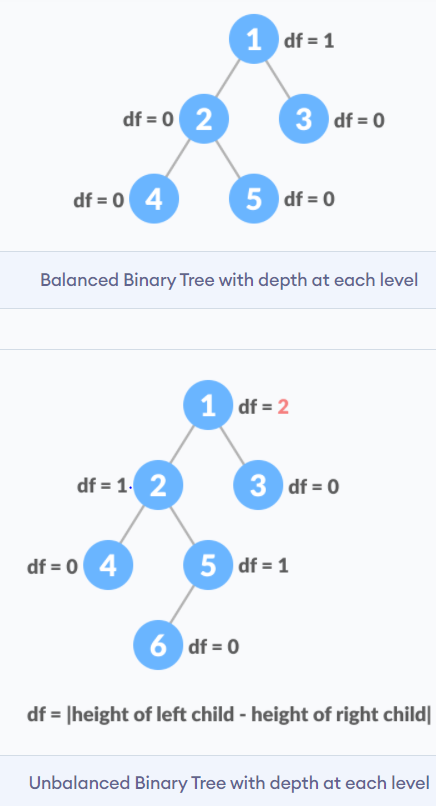
40

**Balaanced Binary tree**

A balanced binary tree, also referred to as a height-balanced binary tree, is defined as a binary tree in which the height of the left and right subtree of any node differ by not more than 1.

To learn more about the height of a tree/node, visit [Tree Data Structure](https://www.programiz.com/dsa/trees).Following are the conditions for a height-balanced binary tree:

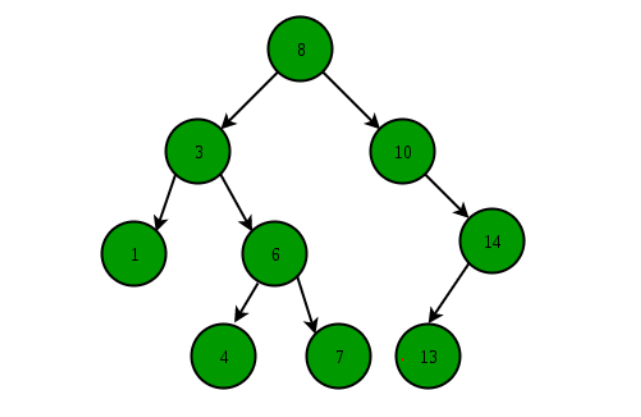
1. difference between the left and the right subtree for any node is not more than one
2. the left subtree is balanced
3. the right subtree is balanced



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:**

