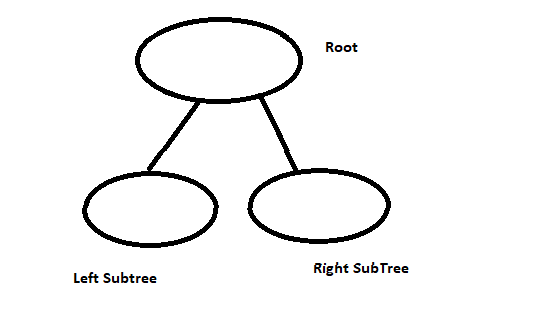
Data Structures

Tree

A Tree is used to represent data in a hierarchical format

Every node in a tree has 2 components (Data and References)

The top node of the tree is called the Root node and the 2 products under it are called "Left Subtree" and "Right Subtree".

Picture Representation of a tree:

Why do we need a Tree?

When we compare a Tree with other data structures, like arrays or a LinkedList, we need not have to mention the size of the tree, hence it is space efficient.

A linked list has big O(n) operation for insertion, deletion, and searching, whereas, with Trees, we do not have such a problem.

Tree Terminologies

"A" represents the Root node (which do not have a parent)

"Edge" is a link between Parent and a Chid (Ex: B to D)

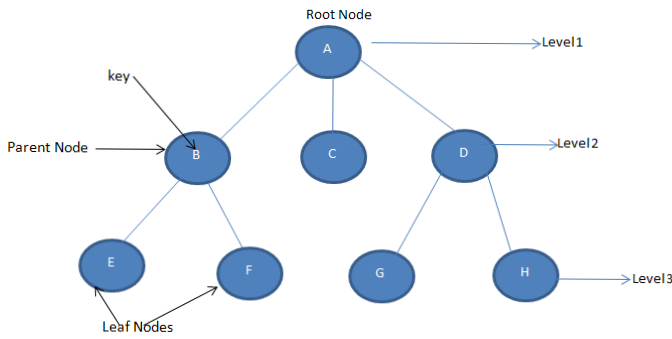
"Leaf" node with no children (Ex: D, E, F, G)

"Sibling" children of the same parent (Ex: D and E are Siblings, they both have Same parent B)

"Ancestor" parent, grandparent for a given node (Ex: D's ancestor is B and A)

"Depth of a node" length of the path from the root to that node (Ex: D's depth is 2)

"Height of a Node" height from a particular node to the deepest node (leaf node) (Ex: height of B is 1 (B to D))



Binary Tree

A tree is said to be a Binary tree if each node has zero, one or two children.

Types of Binary Trees:

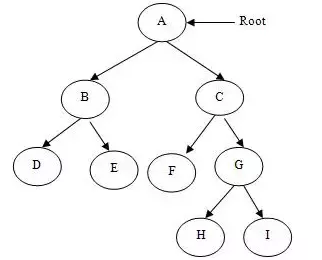
Strict Binary Tree

Full Binary Tree

Complete Binary Tree

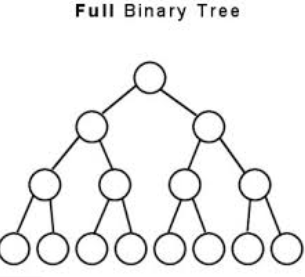
Strict BinaryTree

Each node has either two children or none.



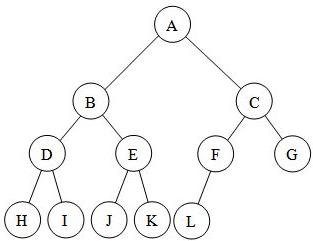
Full Binary Tree

Each Non-leaf node has two children and all the leaf nodes are at the same level.



Complete Binary Tree

If all the levels are completely filled, except the last level and the last level has all the keys as left as possible



Tree Representation can be implemented in two ways.

Using a Linkedlist

Using an Array

Please see code example in Data Structures Solution

References:

<https://www.c-sharpcorner.com/article/tree-data-structure/>