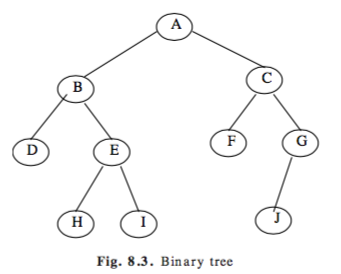
**1. What do you mean by binary tree? Describe the properties of Binary search tree. Explain the binary search tree with example.**

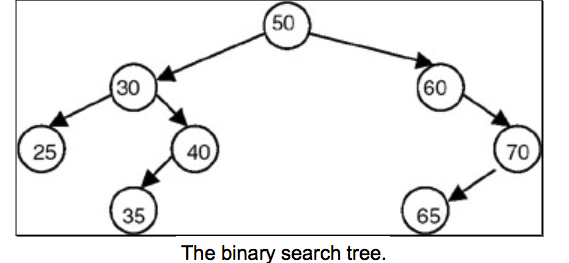
A binary tree is a tree, in which no node can have children more than two. Following is a figure of a binary tree. Here A is called a root of a tree, B is a left sub tree and C is right sub tree. B and C are called siblings and children of A. D,H,I,F,J are called leaf of the tree.

****

A binary tree is a finite set of elements that are either empty or portioned into three disjoint subsets. The first subset contains single elements called root. The other two subsets are themselves binary trees called the left and right sub-trees of the original tree. A left or right sub-tree can be empty. Each element of a binary tree is called a node of the tree.

A binary search tree (BST) is a binary tree that is either empty or in which every node contains a key (value) and satisfies the following conditions:

* All keys in the left sub-tree of the root are smaller than the key in the root node.
* All keys in the right sub-tree of the root are greater than the key in the root node.
* The left and right sub-trees of the root are again binary search trees.



The properties of binary tree are:

* The first subset contains a single element called the root of the tree.
* The other two subsets are themselves binary trees, called left and right subtree of the original tree.

**2. Write recursive algorithms for constructing BST and its traversals. Illustrate them with an example.**

**3. What is Binary Searching. Compare and Contrast between Binary searching and Binary tree searching.**

Binary searching is a fast search algorithm with run-time complexity of Ο(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly the data collection should be in sorted form.

Binary search searches a particular item by comparing the middle most item of the collection. If match occurs then index of item is returned. If middle item is greater than item then item is searched in sub-array to the right of the middle item otherwise item is search in sub-array to the left of the middle item. This process continues on sub-array as well until the size of sub array reduces to zero.

A binary tree is made of nodes, where each node contains a "left" pointer, a "right" pointer, and a data element. The "root" pointer points to the topmost node in the tree. The left and right pointers recursively point to smaller "subtrees" on either side. A null pointer represents a binary tree with no elements -- the empty tree. The formal recursive definition is: a binary tree is either empty (represented by a null pointer), or is made of a single node, where the left and right pointers (recursive definition ahead) each point to a binary tree.

A binary search tree (BST) or "ordered binary tree" is a type of binary tree where the nodes are arranged in order: for each node, all elements in its left subtree are less to the node (<), and all the elements in its right subtree are greater than the node (>). The tree shown above is a binary search tree -- the "root" node is a 5, and its left subtree nodes (1, 3, 4) are < 5, and its right subtree nodes (6, 9) are > 5. Recursively, each of the subtrees must also obey the binary search tree constraint: in the (1, 3, 4) subtree, the 3 is the root, the 1 < 3 and 4 > 3.

**4. Explain the characteristics of Huffman’s algorithm and its application.**

Huffman algorithm is a method for building an extended binary tree with a minimum weighted path length from a set of given weights. This is a method for the construction of minimum redundancy codes. This algorithm is applicable to many forms of data transmission.

The characteristics of Huffman's algorithms are:

* Huffaman algorithm is a method of building an extended binary tree with a minimum weighted part length from a set of given weight.
* This is a method for the construction of minimum redundancy codes.
* Applicable to many forms of data transmission.
* Its value and and the previously calculated sum of the tree are used to form the new node which in turn becomes their parent.

**5. Write the steps involved in deleting a node in a Binary Search tree.**

* Start
* If a node to be deleted is leaf node at a left side just remove it and assign null to its parent left pointer.
* If a node to be deleted is a leaf node at a right side just remove it and assign null to its parent right pointer.
* If a node to be delete has a one child then connect it’s child pointer to parent’s pointer the remove it from the tree.
* If a node to be deleted has two children then replace the node to be deleted either by
  + Right most node of its left sub tree.
  + Left most node of its right sub tree.
* End

**WAP to construct a binary tree.(Already provided some part of the code now complete it with searching, insertion and deletion)**

[**https://github.com/destro014/DSA\_Assignment/tree/master/assignment\_07**](https://github.com/destro014/DSA_Assignment/tree/master/assignment_07)