Lecture

NOTE: FOR FURTHER DETAILS AND MORE COMPREHENSIVE STUDY, PLEASE SEE RECOMMENDED BOOKS OR INTERNET.

Binary Search Tree

A ***Binary Search Tree (BST)*** is a tree in which all the nodes follow the below-mentioned properties:

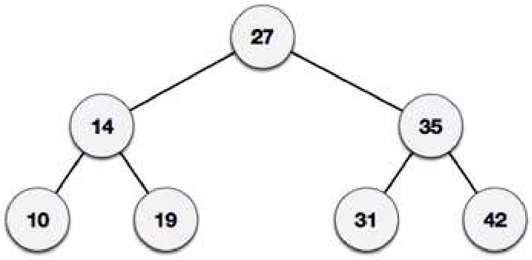
- *The value of the key of the left sub-tree is less than the value of its parent (root) node's key.*
- *The value of the key of the right sub-tree is greater than or equal to the value of its parent (root) node's key.*

Thus, BST divides all its sub-trees into two segments; the left sub-tree and the right sub-tree and can be defined as:

Left < parent ≤ right

Representation

BST is a collection of nodes arranged in a way where they maintain BST properties. Each node has a key and an associated value. While searching, the desired key is compared to the keys in BST and if found, the associated value is retrieved.
Following is a pictorial representation of BST:



We observe that the root node key (27) has all less-valued keys on the left sub-tree and the higher valued keys on the right sub-tree.

Basic Operations

Following are the basic operations of a BST:
Search − Searches an element in a BST.
Insert − Inserts an element in a BST.
Delete – Deletes an element in a BST.
Traverse – Accessing all the elements in a BST

Search Operation

Whenever an element is to be searched, start searching from the root node. Then if the data is less than the key value, search for the element in the left subtree. Otherwise, search for the element in the right subtree.

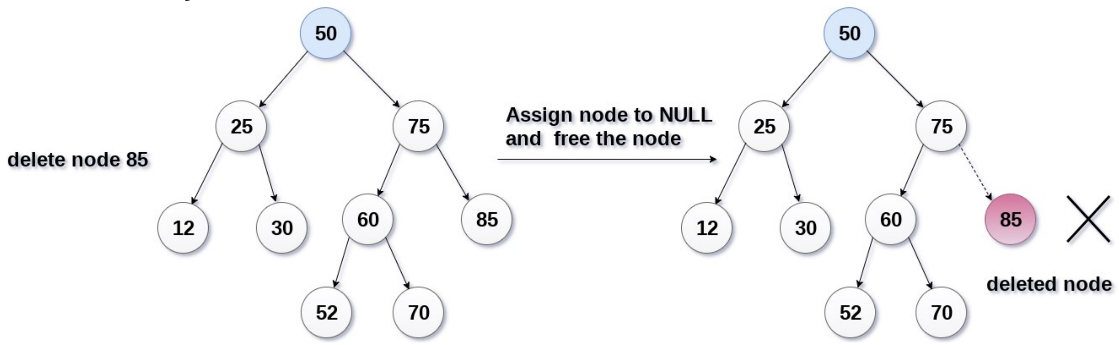
Insert Operation

Whenever an element is to be inserted, first locate its proper location. Start searching from the root node, then if the data is less than the key value, search for the empty location in the left subtree and insert the data. Otherwise, search for the empty location in the right subtree and insert the data.

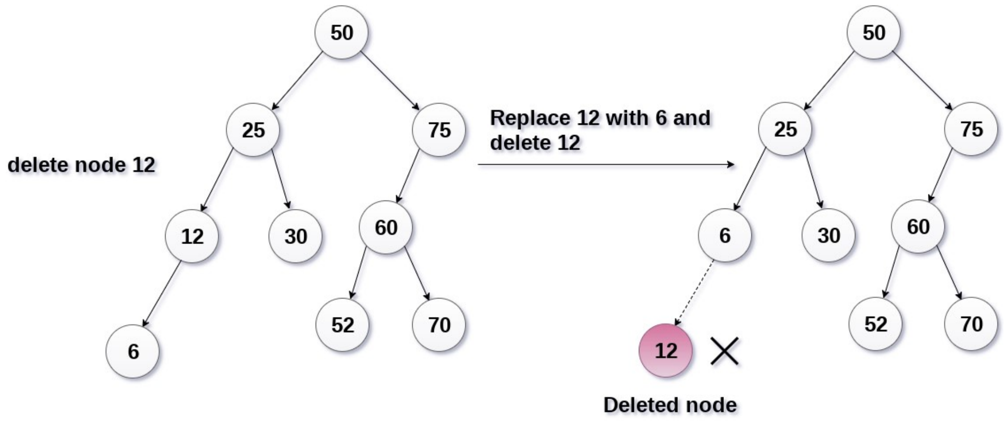
Delete Operation

Whenever an element is to be Deleted, first locate the node to be deleted. Start searching from the root, if the node data is less than the key value, search for the empty location in the left subtree. Otherwise, search in the right subtree. After this there are three cases:

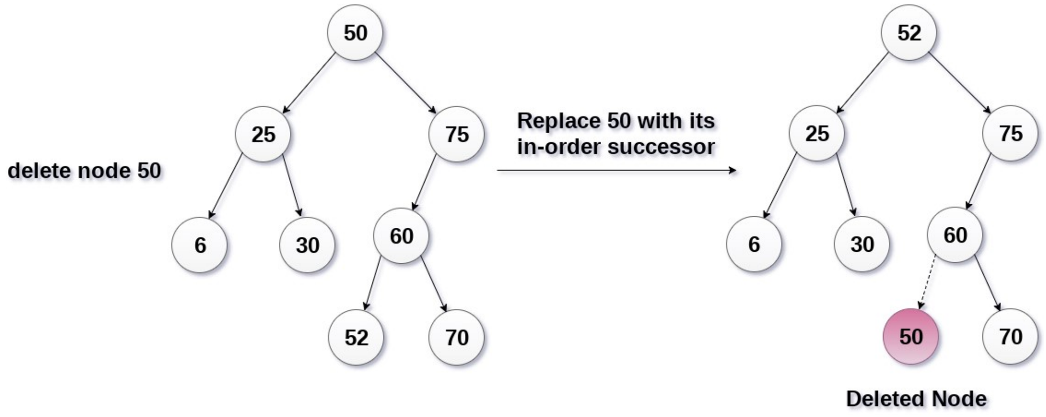
Node is a Leaf Node:



Node has 1 Child:



Node has 2 Childs:



Traverse Operation

Traversal is a process to visit all the nodes of a tree and may print their values too. Because, all nodes are connected via edges (links) we always start from the root (head) node. That is, we cannot randomly access a node in a tree. There are three ways which we use to traverse a tree:

- ***In-order Traversal***
- ***Pre-order Traversal***
- ***Post-order Traversal***

Generally, we traverse a tree to search or locate a given item or key in the tree or to print all the values it contains. Here the Order represents the order of the ***Parent Node***.