Experiment No 5

Title:

A Dictionary stores keywords & its meanings. Provide facility for adding new keywords, deleting keywords, updating values of any entry. Provide facility to display whole data sorted in ascending/ Descending order. Also find how many maximum comparisons may require for finding any keyword. Use Binary search tree and find the complexity for finding a keyword.

Pre-requisite:

- Knowledge of C++ programming
- Basic knowledge of BST operations

Objective:

- To analyze advanced data structure like BST
- To check the complexity for searching a keyword
- To understand practical implementation of BST operations

Input:

Keywords and its meanings

Outcome:

- BST of keywords and its meanings
- All given BST operations

Learning Outcome:

- Define BST using Object Oriented features.
- Analyze working of various operations on BST.

Theory:

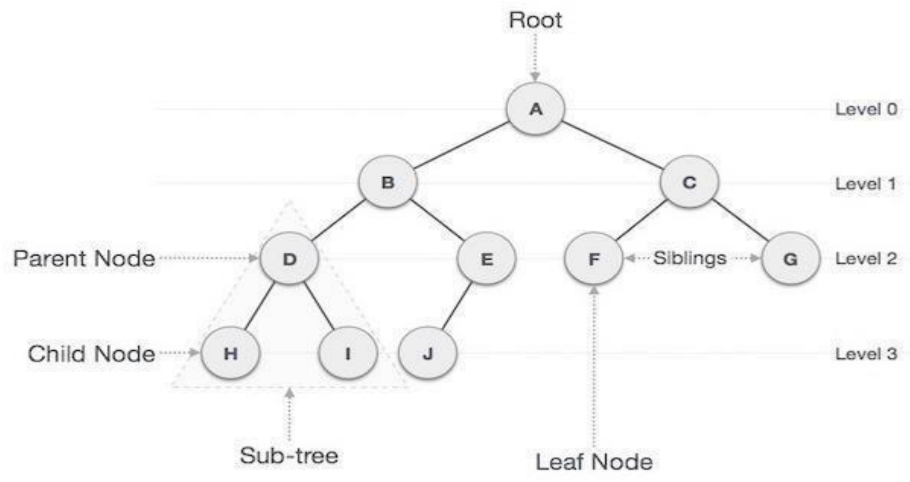
Tree

Tree represents the nodes connected by edges also a class of graphs that is acyclic is termed as trees. Let us now discuss an important class of graphs called trees and its associated terminology. Trees are useful in describing any structure that involves hierarchy. Familiar examples of such structures are family trees, the hierarchy of positions in an organization, and so on.

Binary Tree

A binary tree is made of nodes, where each node contains a "left" reference, a "right" reference, and a data element. The topmost node in the tree is called the root.

Every node (excluding a root) in a tree is connected by a directed edge from exactly one other node. This node is called a parent. On the other hand, each node can be connected to arbitrary number of nodes, called children. Nodes with no children are called leaves, or external nodes. Nodes which are not leaves are called internal nodes. Nodes with the same parent are called siblings.



Insert Operation

The very first insertion creates the tree. Afterwards, 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.

Traversals

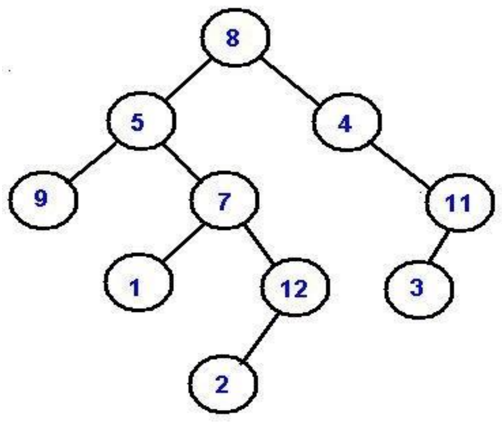
A traversal is a process that visits all the nodes in the tree. Since a tree is a nonlinear data structure, there is no unique traversal. We will consider several traversal algorithms with we group in the following two kinds

- depth-first traversal
- breadth-first traversal

There are three different types of depth-first traversals, :

- PreOrder traversal - visit the parent first and then left and right children;
- InOrder traversal - visit the left child, then the parent and the right child;
- PostOrder traversal - visit left child, then the right child and then the parent;

There is only one kind of breadth-first traversal--the level order traversal. This traversal visits nodes by levels from top to bottom and from left to right. As an example consider the following tree and its four traversals:



PreOrder - 8, 5, 9, 7, 1, 12, 2, 4, 11, 3
InOrder - 9, 5, 1, 7, 2, 12, 8, 4, 3, 11
PostOrder - 9, 1, 2, 12, 7, 5, 3, 11, 4, 8
LevelOrder - 8, 5, 4, 9, 7, 11, 1, 12, 3, 2

Algorithm:

-

Algorithm to insert a node :

Step 1 - Search for the node whose child node is to be inserted. This is a node at some level i, and a node is to be inserted at the level i +1 as either its left child or right child. This is the node after which the insertion is to be made.

Step 2 - Link a new node to the node that becomes its parent node, that is, either the Lchild or the Rchild.

Algorithm to traverse a tree :

- **Inorder traversal**

Until all nodes are traversed −

Step 1 − Recursively traverse left subtree.
Step 2 − Visit root node.

Step 3 − Recursively traverse right subtree.

- **Preorder**

Until all nodes are traversed
– Step 1 − Visit root node.

Step 2 − Recursively traverse left subtree.
Step 3 − Recursively traverse right subtree.

- **Postorder**

Until all nodes are traversed −

Step 1 − Recursively traverse left subtree.
Step 2 − Recursively traverse right subtree.
Step 3 − Visit root node.

Algorithm to copy one tree into another tree :

Step 1 – if (Root == Null)
Then return Null

Step 2 −Tmp = new TreeNode

Step 3 – Tmp->Lchild = TreeCopy(Root->Lchild); Step 4 – Tmp->Rchild = TreeCopy(Root->Rchild); Step 5 – Tmp-Data =

Then return

- S**oftware required:** g++ / gcc compiler- / 64 bit fedora.

Outcome:

Learn object oriented programming features.

Understand & implement different operations on tree & binary tree.

Conclusion: Thus we have studied the implementation of various Binary tree operations

Question Bank:

1. **What is Binary Search Tree?**
2. **What are the criteria to search an element in the BST?**
3. **Construct a BST from the elements 5,2,8,4,1,9,7. Show preorder, inorder, postorder traversal for the same.**