LAB Manual

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**Experiment No.07**

**A.1 Aim:**

**To understand and implement concept of non-linear data structure Tree.**

**A.2 Prerequisite:**

Prior knowledge of introduction to data structure.

**A.3 Outcome:**

**After successful completion of this experiment students will be able to:**

After successful completion of this experiment students will be able to:

1. Understand the concept of data structures
2. Know the use and implementation of different types trees

**A.4 Theory:**

Data structure is logical or mathematical organization of data; it describes how to store the data and access data from memory. Actually in our programming data stored in main memory(RAM) and to develop efficient software or firmware we need to care about memory. To efficiently manage we required data structure.

There are two different types of data structure:

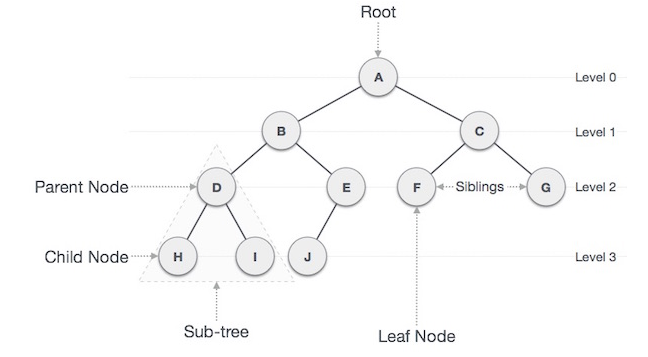
**Linear Data Structure**: In linear data structure data elements stored in sequential manner. Stack, Queue and Linked List are the types of linear data structure.

**Non Linear Data Structure**: In Non-Linear data structure data elements are not stored in the sequence manner. Tree and Graph are the type of non-linear data structure.

**TREES:**

Tree represents the nodes connected by edges. We will discuss binary tree or binary search tree specifically.

Binary Tree is a special data structure used for data storage purposes. A binary tree has a special condition that each node can have a maximum of two children. A binary tree has the benefits of both an ordered array and a linked list as search is as quick as in a sorted array and insertion or deletion operation are as fast as in linked list.



**Important Terms**

Following are the important terms with respect to tree.

**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.

**A.5 Procedure/Algorithm:**

**A.5.1 TASK 1: Write programs for the following questions:**

**Q1**. Write a program in C/C++ to implement Insertion in a Binary search tree.

**A.5.1 TASK 2:**

Save and close the file and name it as **DSA\_Experiment07\_YourName.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No. | Name: |
| Class : | Batch : |
| Date of Experiment: | Date of Submission |
| Grade : |  |

**B.1 Answers of Task to be written by student:**

***B.1.1 Code***

***B.1.2 Output***

**B.2 Observations and learning:**

***(Students are expected to comment on the output obtained with clear observations and learning for each task/ sub part assigned)***

**B.3 Conclusion:**

*(****Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.2)***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***