Modern Education Society's Wadia College of Engineering, Pune

210256: DATA STRUCTURES and ALGORITHM LABORATORY (2019 C0URSE)

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR:	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO: B11**

TITLE: Creation of Dictionary using BST.

AIM/PROBLEM STATEMENT: 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 for implementation.

OBJECTIVES:

1. To understand structure of binary search tree
2. To perform various objectives of dictionary using binary search tree

OUTCOMES:

1. To apply the binary search tree for dictionary operations
2. To analyze the

PRE-REQUISITES:

1. Knowledge of C++ programming
2. Basic knowledge of Dictionary ADT
3. Basic knowledge of searching in binary search tree.

THEORY:

Binary Search Tree, is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys lesser than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.

- The left and right subtree each must also be a binary search tree.
- There must be no duplicate nodes

The above properties of Binary Search Tree provide an ordering among keys so that the operations like search, minimum and maximum can be done fast. If there is no ordering, then we may have to compare every key to search a given key.

Searching a key:

To search a given key in Binary Search Tree, we first compare it with root, if the key is present at root, we return root. If key is greater than root's key, we recur for right subtree of root node. Otherwise, we recur for left subtree.

A simple implementation for the Dictionary ADT can be based on sorted or unsorted lists. When implementing the dictionary with an unsorted list, inserting a new record into the dictionary can be performed quickly by putting it at the end of the list. However, searching an unsorted list for a particular record requires Θ(n) time in the average case. For a large database, this is probably much too slow. Alternatively, the records can be stored in a sorted list. If the list is implemented using a linked list, then no speedup to the search operation will result from storing the records in sorted order. On the other hand, if we use a sorted array-based list to implement the dictionary, then binary search can be used to find a record in only Θ (log n) time. However, insertion will now require Θ(n) time on average because, once the proper location for the new record in the sorted list has been found, many records might be shifted to make room for the new record.

The way to organize a collection of records so that inserting records and searching for records can both be done quickly, is by using binary search tree (BST). The advantage of using the BST is that all major operations (insert, search, and remove) are Θ(log n) in the average case. Of course, if the tree is badly balanced, then the cost can be as bad as Θ(n)

QUESTIONS:

1. Discuss about various operations of binary search tree.

2. What is Dictionary ADT and its operations?