KJ's Educational Institutes

K J College Of Engineering & Management Research, Pune.

Department of E & TC

CLASS: S. E. (E &TC) **SUBJECT:-DSA**

Ex. No: 4 **Date:**

AIM : Binary Search Tree Operations
An online directory system uses a BST to keep names in a sorted manner and support fast searching.

Create a binary search tree and implement recursive traversals (inorder, preorder, postorder) and search for a specific name in the directory.

OBJECTIVES

i) To implement logic for constructing binary search tree.

ii) To implement following primitive operations-

Create , search, recursive traversals .

THEORY

Tree:

Tree is a data structure in the form of a finite set of elements that is either empty or is partitioned into one or more disjoint subsets of which one subset is special contains a single element called as root of the tree. The two other subsets are themselves binary trees called as children of root node. The root is not child of any node.

Binary Tree:

A binary tree is a finite set of nodes which is either empty or consists of root and two disjoint binary trees called the left sub-tree and right sub-tree.

Binary search tree:

Binary search tree is binary tree which have all the values less than the root will lie in left sub tree & having values greater than the root lie in right sub tree.

In computer science a binary search tree (BST) is a node based binary tree data structure which has the following properties:

- The left sub-tree of a node contains only nodes with keys less than the node's key.
- The right sub-tree of a node contains only nodes with keys greater than the node's key.
- Both the left and right sub-trees must also be binary search trees

Traversal:

Traversal is a systematic way of retrieving information form tree in such a way that no node will be left unvisited or no node will be visited twice or more.
There are different techniques of traversals:

1) Pre-order
2) Post-order

3) In-order
4) Depth first search
5) Breadth first

Each of these methods start processing from root of tree.

1) **Pre-order**: In this data on the root node will be printed first then we move on the left sub-tree and go on printing the data till we reach to the leftmost node. Print the data at that node and then move to the right sub-tree.

2) **In-order** : In this traversal first we go towards the leftmost node to print data on that node then traversing left sub-tree then print root node then traverse right subtree.

3) **Post-order:** In post order traversal we follow the LRD principle i.e. move to the leftmost node check if right sub-tree is there or not if not then print the leftmost node, if right sub tree is there move towards rightmost node.

Algorithm for search:

```
Algorithm search (tree node,key)
  1) If tree is empty

     Return NULL.
  2) If key = root of key

       Return root.
  3) Else If key<root  of key

       Return (search( root of left child, key))
  4) Else

         Return (search( root of right child, key))
  5) If key is not found

       Return NULL
       End search.
```

Algorithm for create :

Modify above search algorithms in such way that if key is found into BST return NULL otherwise return last visited leaf node.

```
Algorithm create(tree node,key,data )
1) Tree ptr, temp;

   Temp= search(tree node, key);
2) If temp= NULL return

3) Allocate memory for ptr

4) Ptr of key = key

    Ptr of data =data
   Ptr of left child =ptr of right child=NULL.
5) If tree is not empty
```

```
        If key < temp of key
                Temp of left child = ptr return tree node
        Else temp of right child =ptr return tree node
    Else node =ptr return ptr
              End create.
```

Input:

Data in the form of numbers.

Output:

Output is the BST and primitive operations on BST.

Application:

- priority queues
- associative arrays

CONCLUSION:-