Experiment No. 8

Aim:

Implementation of Binary Search Tree (BST) and its operations.

Objective:

The objective of this experiment is to implement and understand various operations on a Binary Search Tree (BST)**.**

1. To construct a BST by inserting nodes in the given order.
2. To perform insertion of a new node into the BST.
3. To determine the number of nodes in the longest path from the root (height of the tree).
4. To find the minimum data value present in the BST.
5. To implement the search operation to locate a given value in the BST.

Problem Statement:

Beginning with an empty binary search tree, Construct a binary search tree by inserting the values in the order given. After constructing a binary tree

i. Insert new node
ii. Find number of nodes in longest path from root
iii. Minimum data value found in the tree
iv. Change a tree so that the roles of the left and right pointers are swapped at every node
v. Search a value.

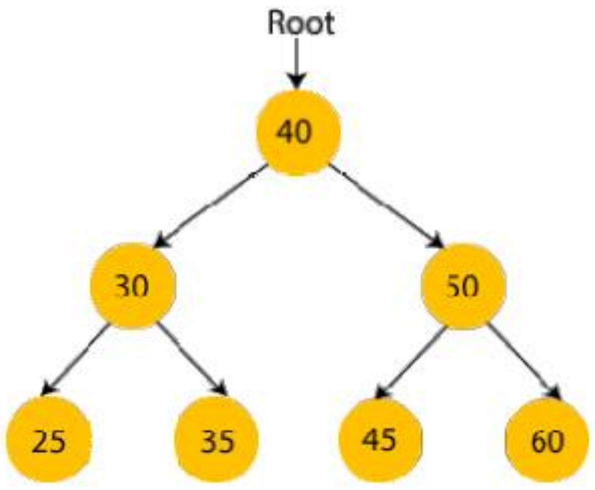
Theory:

A binary tree is composed of nodes connected by edges. Some binary tree is either empty or consists of a single root element with two distinct binary tree child elements known as the left subtree and the right subtree of a node. As the name binary suggests, a node in a binary tree has a maximum of children.

An important special kind of binary tree is the binary search tree (BST). In a

BST, each node stores some information including a unique key value, and perhaps some associated data. A binary tree is a BST iff, for every node n in the tree:

- All keys in n's left subtree are less than the key in n, and
- all keys in n's right subtree are greater than the key in n.



Binary Search Tree Operations:

Algorithms:

Algorithm to insert a node in the binary tree:

```
1.  Create a new BST node and assign values to it.
2.  insert(node, key)
    i) If root == NULL,
    return the new node to the calling function.
    ii) if root=>data < key
    call the insert function with root=>right and assign
    the return value in root=>right. root->right =
    insert(root=>right,key)
    iii) if root=>data > key
    call the insert function with root->left and assign the
    return value in root=>left. root=>left =
    insert(root=>left,key)
3.  Finally, return the original root pointer to the calling function.
```

Searching a Node in BST:

If the value is below the root, we can say for sure that the value is not in the right sub tree; we need to only search in the left sub tree and if the value is above the root, we can say for sure that the value is not in the left sub tree; we need to only search in the right sub tree.

Algorithm:

```
1. If root == NULL
2. return NULL;
3. If number == root->data
4. return root->data;
5. If number < root->data
6. return search(root->left)
7. If number > root->data
8. return search(root->right)
```

Algorithm to find the minimum value in binary tree:

1. Define Node class which has three attributes namely: data, left and right. Here, left represents the left child of the node and right represents the right child of the node.
2. When a node is created, data will pass to data attribute of node and both left and right will be set to null.
3. Define another class which has an attribute root.
   A) Root represent root node of the tree and initialize it to null.

Mirror() Operation

The Mirror() operation finds the mirror of the tree that will interchange all left and right subtrees in a linked binary tree.

Algorithm:

(1) Call Mirror for left->subtree i.e., Mirror(left->subtree)

(2) Call Mirror for right->subtree i.e., Mirror(right-subtree)

(3) Swap left and right subtrees.

```
temp = left->subtree
left->subtree = right->subtree
right->subtree = temp
```

Example:

Sample Input / Output

Input Order of Insertions: 50, 30, 70, 20, 40, 60, 80

Operations:

- Insert new node → 25
- Height of tree → 3
- Minimum value → 20
- Mirror tree → Inorder traversal before mirror: 20 25 30 40 50 60 70 80 After mirror: 80 70 60 50 40 30 25 20
- Search 60 → Found

Input:

Note: Write input you are giving to your program

Output:

Note: Write output of your program

Conclusion:

Thus, a Binary Search Tree was constructed and operations like insertion, finding height, minimum value, mirroring, and searching were successfully implemented.