EXPERIMENT :5(B)

Aim:

Beginning with an empty binary search tree, Construct binary searchtree 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
iii.Minimum data value found inthe tree
iv. Change a tree so that the roles of the left and right pointers are swapped atevery
node
v.Search a value

Theory –

Tree represents the nodes connected by edges. **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.

Binary Search Tree Representation
Binary Search tree exhibits a special behavior. A node's left child must have a value less than its parent's value and the node's right child must have a value greater than its parent value.

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties −
•The left sub-tree of a node has a key less than or equal to its parent node's key.
•The right sub-tree of a node has a key greater than or equal to its parent node's key.
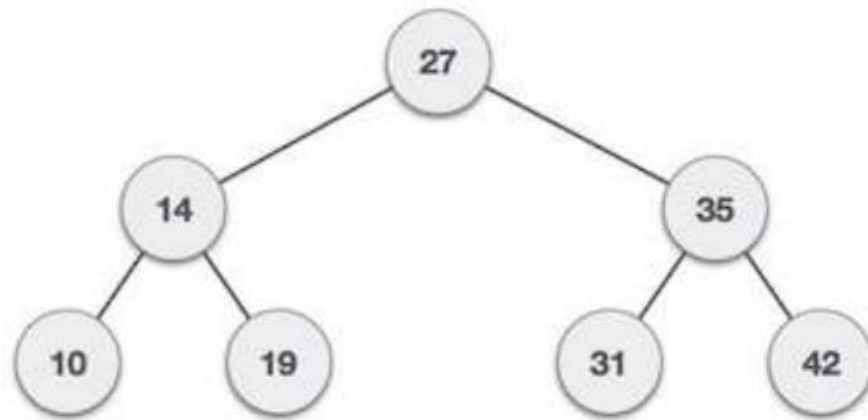Thus, BST divides all its sub-trees into two segments;the left sub-tree and the right sub-tree and can be defined as –

$$left\_subtree \ (keys) \ \leq \ node \ (key) \ \leq \ right\_subtree \ (keys)$$

Tree Node

> **Following Structure is used for Node creation**
>
> > Struct node {
> > Int data ;
> > Struct node *leftChild;
> > Struct node *rightChild;
> > };

Fig: Binary Search TreeBST Basic

Operations

> The basic operations that can be performed on a binary search tree data structure, are the following −
>
> • **Insert**− Inserts an element in a tree/create a tree.
> •**Search**− Searches an element in a tree.
> •**Traversal**− A traversal is a systematic way to visit all nodes of T -Inorder, Preprder, Postorder,
> > a. pre-order: Root, Left, Right
> > > Parent comes before children; overall root first
> > B.post-order: Left, Right, Root
> > > Parent comes after children; overall root last
> > c. In Order: In-order: Left, Root, Right,

Insert Operation: Algorithm

```
If root is NULL
    then create root node
return


If root exists then
    compare the data with node.data

    while until insertion position is located

       If data is greater than node.data
          goto right subtree
       else
          goto left subtree
```

Search Operation:
Algorithm

```
If root.data is equal to search.data
    return root
else
    while data not found

       If data is greater than node.data
          goto right subtree
       else
          goto left subtree

       If data found
          return node

    endwhile
```

Tree Traversal

In order traversal algorithm

```
Until all nodes are traversed –

Step 1 – Recursively traverse left subtree.

Step 2 – Visit root node.

Step 3 – Recursively traverse right subtree.
```

Pre order traversal Algorithm

```
Until all nodes are traversed -
Step 1 - Visit root node.
Step 2 - Recursively traverse left subtree.
Step 3 - Recursively traverse right subtree.
```

Post order traversal Algorithm

```
Until all nodes are traversed -
Step 1 - Recursively traverse left subtree.
Step 2 - Recursively traverse right subtree.
Step 3 - Visit root node.
```

Deleting in a BST

case 1: delete a node with zero child-
    if x is left of its parent, set
    parent(x).left = nullelse set
    parent(x).right = null
case 2: delete a node with one child
    link parent(x)
to the child of xcase
3: delete a node with
2 children
Replace inorder successor to deleted node position

C++ CODE :-

Conclusion: we have implemented binary search tree by inserting the values in the order given