**Name: Veeransh Shah**

**Reg No.: 221070063**

**BINARY SEARCH TREE**

**Aim:** To learn what a binary search tree(BST) is and to perform insertion and checking if a binary tree is a BST or not. To do preorder traversal using an iterative way using stack. To do inorder traversal recursively

**Theory:**

Definition : i) It is a binary tree i.e each parent node can have maximally two children nodes.

ii) At each node, the left child data is less than the parent node data and right child data is greater than the parent node data.

Note : A printing of inorder traversal for BST results in a sorted (ascending)  printing of data.

iii) Preorder traversal is defined as a type of tree traversal that follows the Root-Left-Right policy where:

* The root node of the subtree is visited first.
* Then the left subtree  is traversed.
* At last, the right subtree is traversed.

iv)Inorder traversal is defined as a type of tree traversal technique which follows the Left-Root-Right pattern, such that:

* The left subtree is traversed first
* Then the root node for that subtree is traversed
* Finally, the right subtree is traversed

**Algorithm:**

i) Insertion in a BST :

For inserting in a BST there are two ways, iterative and recursive.

By Iterative Way: We will make use of 2 pointers. A current pointer (p) and a follower pointer (q).

//Initialization.

At every node we will check whether it should go left or right.

At start our p is pointing to the root and q is pointing to null.

//Looking for the parent node for given input data

Using a while loop with condition p!=NULL.

At every node i.e p we will check p->data with the input data.

If it is less than p->data then q = p and p=p->left.

If it is greater than p->data then q = p and p=p->right.

//Looking for its position according to data of its parent node.

When p becomes NULL, the loop breaks.

Then we will again check for q->data with input data.

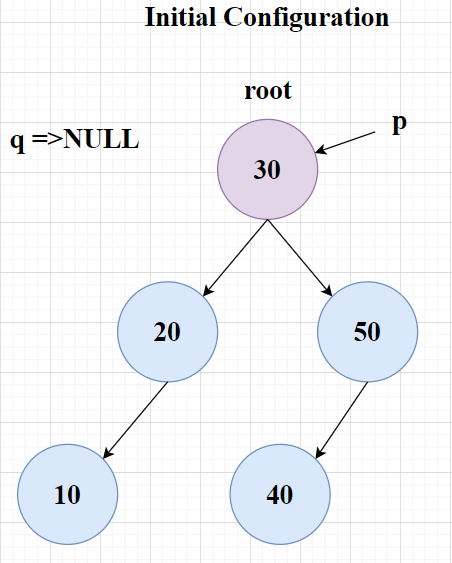
If it is less than q->data then q->left = new node(data);

It it is greater than q->data then q->right = new node(data);

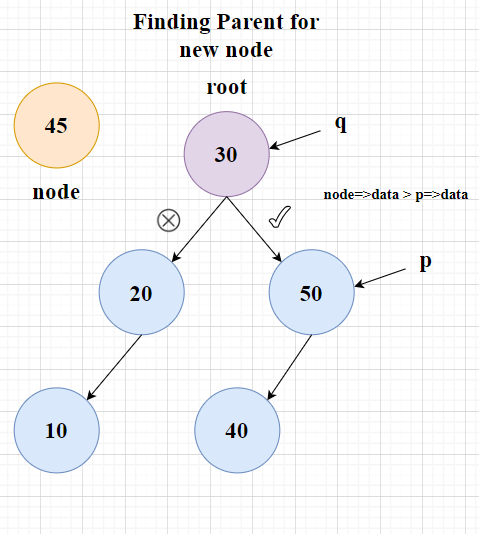
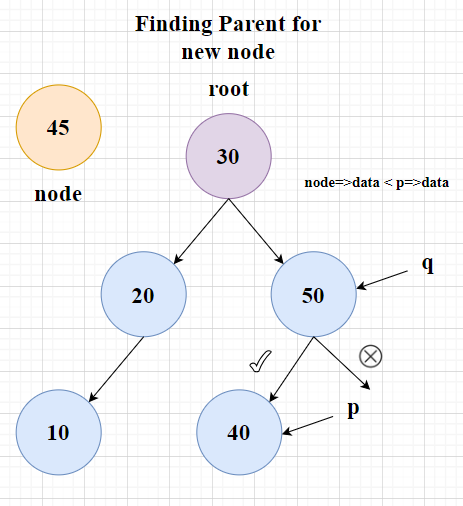
**Explanation/Example :**

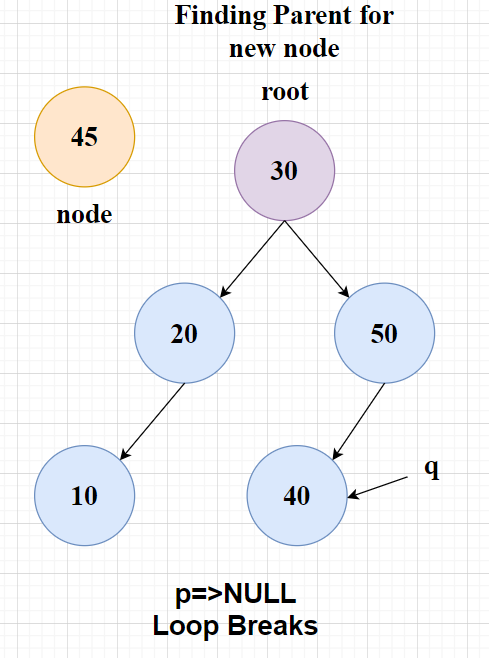
i) Iterative way: We want to add a node with data = 45.

Step 1 : Initialization



Step 2 : Looking for the parent node for the given input data



Step 3 : Looking for its position according to data of its parent node.

Since the node data = 45 and it is greater than q->data i.e 40.

Therefore it will become a right child node 40.

Note : When the first node is added to a BST then we directly add the node to our bst and return from there.

**Algorithm :**

ii) Checking a BT is a BST or not :

//We make a function to check bst as isBST and call it recursively at each node

     // Base case: an empty tree is a BST

if node is null:

return true

endif

         // Check if the node is leaf node

         if node->left is null and node->right is null

return true

          Endif

//Check if the left node is present

//Check if the left node’s value is more than the node

if node->data<node->left->data

return false

//Check if the right node is present

//Check if the right node’s value is less than the node

if node->data>node->right->data

return false

//Check recursively if the left and right node is bst or not

return isBst(node->left) and isBst(node->right)

// Example usage:

// Construct a sample BST

root = TreeNode(2)

root.left = TreeNode(1)

root.right = TreeNode(3)

// Check if it's a BST

result = isBST(root, null)

print(result)  // Output: true

**Example :**

1)

// Construct a sample BST

root->value =2

root->left = 1

root->right =3

// Check if it's a BST

result = isBST(root)

print(result)  // Output: true

2)

// Construct a sample BST

root->value =2

root->left = 3

root->right =1

// Check if it's a BST

result = isBST(root)

print(result)  // Output: false

**Algorithm :** iii) To do preorder traversal using stack

1. Initialize a stack and push the root node.
2. Enter the while loop having condition !st.isEmpty()
3. Pop the node, print the data and then push the right child of the temporary node and then push the left child of the temporary node respectively.

**Example :** Consider a BST,

root = 20

root->left = 10

root->left->left = 5

root->left->right = 15

root->right = 30

root->right->left = 25

root->right->right = 35

i) We will start from the root, and push the root into the stack.

ii) Popping the top element from stack i.e 20 and pushing the root and left child respectively. Stack={30,10}

iii) Again popping the stack top element i.e 10, print and push the right and left child respectively. Stack={30,15,5}

iv) Print 5 , Stack = {30,15}

v) Print 15 , stack = {30}

vi) Print 30, Stack = {35,25}

vii) Print 25, Stack = {35}

viii) Print 35, Stack = {}

Since, stack has now become empty, therefore exit this while loop.

**Algorithm :** iv) To do inorder traversal recursively

1. Traverse the left subtree
2. Perform the action on the current node
3. Traverse the right subtree

**Example :** Consider a bst as given in the above example.

Therefore, printing will occur in the following fashion 5,10,15,20,25,30,35.

CONCLUSION:

We learnt about Binary trees and BST and learnt about insertion in a binary tree and checked whether the given binary tree is a BST.We did inorder traversal normally and preorder traversal in iterative method using stacks.

ScreenShot:

