Sum of nodes in a Binary Search Tree with values from a given range

Last Updated : 20 Mar, 2023

Given a Binary Search Tree consisting of **N** nodes and two positive integers **L** and **R**, the task is to find the sum of values of all the nodes that lie in the range **[L, R]**.

Examples:

> ***Input:*** *L = 7, R = 15*
>
> ```
>          10
>         /  \
>       5     15
>     / \       \
>   3     7   18
> ```
>
> ***Output:*** *32*
>
> ***Explanation:***
>
> *The nodes in the given Tree that lies in the range [7, 15] are {7, 10, 15}. Therefore, the sum of nodes is 7 + 10 + 15 = 32.*
>
> ***Input:*** *L = 11, R = 15*
>
> ```
>          8
>         /  \
>       5     11
>     / \       \
>   3     6   20
> ```
>
> ***Output:*** *11*

Approach: The given problem can be solved by performing any Tree Traversal and calculating the sum of nodes which are lying in the range **[L, R]**.

Below is the implementation of the above approach:

C++

```
// C++ program for the above approach
#include <bits/stdc++.h>
using namespace std;

// Class for node of the Tree
class Node {
public:
    int val;
    Node *left, *right;
};

// Function to create a new BST node
Node* newNode(int item)
{
    Node* temp = new Node();
    temp->val = item;
    temp->left = temp->right = NULL;

    // Return the newly created node
    return temp;
}

// Stores the sum of all nodes
// lying in the range [L, R]
int sum = 0;

// Function to perform level order
// traversal on the Tree and
// calculate the required sum
int rangeSumBST(Node* root, int low,
                int high)
{
    // Base Case
    if (root == NULL)
        return 0;

    // Stores the nodes while
    // performing level order traversal
    queue<Node*> q;

    // Push the root node
    // into the queue
    q.push(root);

    // Iterate until queue is empty
    while (q.empty() == false) {

        // Stores the front
        // node of the queue
        Node* curr = q.front();
        q.pop();

        // If the value of the node
        // lies in the given range
        if (curr->val >= low
            && curr->val <= high) {

            // Add it to sum
            sum += curr->val;
        }

        // If the left child is
        // not NULL and exceeds low
        if (curr->left != NULL
            && curr->val > low)

            // Insert into queue
            q.push(curr->left);

        // If the right child is not
        // NULL and exceeds low
        if (curr->right != NULL
            && curr->val < high)

            // Insert into queue
            q.push(curr->right);
    }

    // Return the resultant sum
    return sum;
}

// Function to insert a new node
// into the Binary Search Tree
Node* insert(Node* node, int data)
{
    // Base Case
    if (node == NULL)
        return newNode(data);

    // If the data is less than the
    // value of the current node
    if (data <= node->val)

        // Recur for left subtree
        node->left = insert(node->left,
                            data);

    // Otherwise
    else

        // Recur for the right subtree
        node->right = insert(node->right,
                             data);

    // Return the node
    return node;
}

// Driver Code
int main()
{
    /* Let us create following BST
         10
        / \
       5   15
      / \     \
     3   7    18  */
    Node* root = NULL;
    root = insert(root, 10);
    insert(root, 5);
    insert(root, 15);
    insert(root, 3);
    insert(root, 7);
    insert(root, 18);

    int L = 7, R = 15;
    cout << rangeSumBST(root, L, R);

    return 0;
}
```

Java

```
// Java program for the above approach
import java.util.*;

public class GFG{

// Class for node of the Tree
static class Node
{
    int val;
    Node left, right;
};

// Function to create a new BST node
static Node newNode(int item)
{
    Node temp = new Node();
    temp.val = item;
    temp.left = temp.right = null;
```