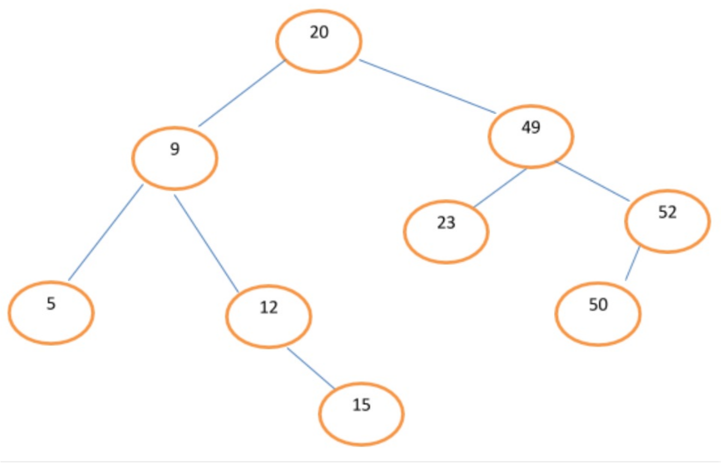
Find sum of all left leaves in a given Binary Tree

Given a Binary Tree, find sum of all left leaves in it. For example, sum of all left leaves in below Binary Tree is 5+23+50 = 78.



We strongly recommend to minimize your browser and try this yourself first.

The idea is to traverse the tree, starting from root. For every node, check if its left subtree is a leaf. If it is, then add it to the result.

Following is C++ implementation of above idea.

C++

```
// A C++ program to find sum of all left leaves
#include <iostream>
using namespace std;

/* A binary tree Node has key, pointer to left and right
   children */
struct Node
{
    int key;
    struct Node* left, *right;
};

/* Helper function that allocates a new node with the
   given data and NULL left and right pointer. */
Node *newNode(char k)
{
    Node *node = new Node;
    node->key = k;
    node->right = node->left = NULL;
    return node;
}

// A utility function to check if a given node is leaf or not
bool isLeaf(Node *node)
{
   if (node == NULL)
       return false;
   if (node->left == NULL && node->right == NULL)
       return true;
   return false;
}

// This function returns sum of all left leaves in a given
// binary tree
int leftLeavesSum(Node *root)
```

```
{
    // Initialize result
    int res = 0;

    // Update result if root is not NULL
    if (root != NULL)
    {
       // If left of root is NULL, then add key of
       // left child
       if (isLeaf(root->left))
            res += root->left->key;
       else // Else recur for left child of root
            res += leftLeavesSum(root->left);

       // Recur for right child of root and update res
       res += leftLeavesSum(root->right);
    }

    // return result
    return res;
}

/* Driver program to test above functions*/
int main()
{
    // Let us construct the Binary Tree shown in the
    // above figure
    struct Node *root         = newNode(20);
    root->left                = newNode(9);
    root->right               = newNode(49);
    root->right->left         = newNode(23);
    root->right->right        = newNode(52);
    root->right->right->left  = newNode(50);
    root->left->left          = newNode(5);
    root->left->right         = newNode(12);
    root->left->right->right  = newNode(12);
    cout << "Sum of left leaves is "
         << leftLeavesSum(root);
    return 0;
}
```

Java

```
// Java program to find sum of all left leaves
class Node
{
    int data;
    Node left, right;

    Node(int item)
    {
        data = item;
        left = right = null;
    }
}

class BinaryTree
{
    Node root;

    // A utility function to check if a given node is leaf or not
    boolean isLeaf(Node node)
    {
        if (node == null)
            return false;
        if (node.left == null && node.right == null)
            return true;
        return false;
    }

     // This function returns sum of all left leaves in a given
     // This function returns sum of all left leaves in a given
     // binary tree
    int leftLeavesSum(Node node)
    {
        // Initialize result
        int res = 0;

        // Update result if root is not NULL
        if (node != null)
        {
            // If left of root is NULL, then add key of
            // left child
            if (isLeaf(node.left))
                res += node.left.data;
            else // Else recur for left child of root
                res += leftLeavesSum(node.left);

            // Recur for right child of root and update res
            res += leftLeavesSum(node.right);
        }

        // return result
        return res;
    }

    // Driver program
    public static void main(String args[])
    {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(20);
        tree.root.left = new Node(9);
        tree.root.right = new Node(49);
        tree.root.left.right = new Node(12);
        tree.root.left.left = new Node(5);
        tree.root.right.left = new Node(23);
        tree.root.right.right = new Node(52);
        tree.root.left.right.right = new Node(12);
        tree.root.right.right.left = new Node(50);

        System.out.println("The sum of leaves is " +
                                    tree.leftLeavesSum(tree.root));
    }
}

// This code is contributed by Mayank Jaiswal
```

Python

```python
# Python program to find sum of all left leaves

# A Binary tree node
class Node:
    # Constructor to create a new Node
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

# A utility function to check if a given node is leaf or not
def isLeaf(node):
    if node is None:
        return False
    if node.left is None and node.right is None:
        return True
    return False

# This function return sum of all left leaves in a
# given binary tree
def leftLeavesSum(root):

    # Initialize result
    res = 0

    # Update result if root is not None
    if root is not None:

        # If left of root is None, then add key of
        # left child
        if isLeaf(root.left):
            res += root.left.key
        else:
            # Else recur for left child of root
            res += leftLeavesSum(root.left)

        # Recur for right child of root and update res
        res += leftLeavesSum(root.right)
    return res

# Driver program to test above function

# Let us constrcut the Binary Tree shown in the above function
root = Node(20)
root.left = Node(9)
root.right = Node(49)
root.right.left = Node(23)
root.right.right = Node(52)
root.right.right.left = Node(50)
root.left.left = Node(5)
root.left.right = Node(12)
root.left.right.right = Node(12)
print "Sum of left leaves is", leftLeavesSum(root)

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Output:

```
Sum of left leaves is 78
```

Time complexity of the above solution is O(n) where n is number of nodes in Binary Tree.

Following is Another Method to solve the above problem. This solution passes in a sum variable as an accumulator. When a left leaf is encountered, the leaf's data is added to sum. Time complexity of this method is also O(n). Thanks to Xin Tong (geeksforgeeks userid trent.tong) for suggesting this method.

C++

```
// A C++ program to find sum of all left leaves
#include <iostream>
using namespace std;

/* A binary tree Node has key, pointer to left and right
   children */
struct Node
{
    int key;
    struct Node* left, *right;
};

/* Helper function that allocates a new node with the
   given data and NULL left and right pointer. */
Node *newNode(char k)
{
    Node *node = new Node;
    node->key = k;
    node->right = node->left = NULL;
    return node;
}

/* Pass in a sum variable as an accumulator */
void leftLeavesSumRec(Node *root, bool isleft, int *sum)
{
    if (!root) return;

    // Check whether this node is a leaf node and is left.
    if (!root->left && !root->right && isleft)
        *sum += root->key;

    // Pass 1 for left and 0 for right
    leftLeavesSumRec(root->left,  1, sum);
    leftLeavesSumRec(root->right, 0, sum);
}

// A wrapper over above recursive function
int leftLeavesSum(Node *root)
{
    int sum = 0; //Initialize result

    // use the above recursive function to evaluate sum
    leftLeavesSumRec(root, 0, &sum);

    return sum;
}

/* Driver program to test above functions*/
int main()
{
    // Let us construct the Binary Tree shown in the
    // above figure
    int sum = 0;
    struct Node *root         = newNode(20);
    root->left                = newNode(9);
    root->right               = newNode(49);
    root->right->left         = newNode(23);
    root->right->right        = newNode(52);
    root->right->right->left  = newNode(50);
    root->left->left          = newNode(5);
    root->left->right         = newNode(12);
    root->left->right->right  = newNode(12);

    cout << "Sum of left leaves is " << leftLeavesSum(root) << endl;
    return 0;
}
```

Java

```java
// Java program to find sum of all left leaves
class Node
{
    int data;
    Node left, right;

    Node(int item) {
        data = item;
        left = right = null;
    }
}

// Passing sum as accumulator and implementing pass by reference
// of sum variable
class Sum
{
    int sum = 0;
}

class BinaryTree
{
    Node root;

    /* Pass in a sum variable as an accumulator */
    void leftLeavesSumRec(Node node, boolean isleft, Sum summ)
    {
        if (node == null)
            return;

        // Check whether this node is a leaf node and is left.
        if (node.left == null && node.right == null && isleft)
            summ.sum = summ.sum + node.data;

        // Pass true for left and false for right
        leftLeavesSumRec(node.left, true, summ);
        leftLeavesSumRec(node.right, false, summ);
    }

    // A wrapper over above recursive function
    int leftLeavesSum(Node node)
    {
        Sum suum = new Sum();

        // use the above recursive function to evaluate sum
        leftLeavesSumRec(node, false, suum);

        return suum.sum;
    }

    // Driver program
    public static void main(String args[])
    {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(20);
        tree.root.left = new Node(9);
        tree.root.right = new Node(49);
        tree.root.left.right = new Node(12);
        tree.root.left.left = new Node(5);
        tree.root.right.left = new Node(23);
        tree.root.right.right = new Node(52);
        tree.root.left.right.right = new Node(12);
        tree.root.right.right.left = new Node(50);

        System.out.println("The sum of leaves is " +
                                    tree.leftLeavesSum(tree.root));
    }
}

// This code is contributed by Mayank Jaiswal
```

Python

```
# Python program to find sum of all left leaves

# A binary tree node
class Node:

    # A constructor to create a new Node
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

def leftLeavesSumRec(root, isLeft, summ):
    if root is None:
        return

    # Check whether this node is a leaf node and is left
    if root.left is None and root.right is None and isLeft == True:
        summ[0] += root.key

    # Pass 1 for left and 0 for right
    leftLeavesSumRec(root.left, 1, summ)
    leftLeavesSumRec(root.right, 0, summ)


# A wrapper over above recursive function
def leftLeavesSum(root):
    summ = [0] # initialize result

    # Use the above recursive fucntion to evaluate sum
    leftLeavesSumRec(root, 0, summ)

    return summ[0]

# Driver program to test above function

# Let us construct the Binary Tree shown in the
# above figure
root = Node(20);
root.left= Node(9);
root.right   = Node(49);
root.right.left = Node(23);
root.right.right= Node(52);
root.right.right.left  = Node(50);
root.left.left  = Node(5);
root.left.right = Node(12);
root.left.right.right  = Node(12);

print "Sum of left leaves is", leftLeavesSum(root)

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Output:

```
Sum of left leaves is 78
```