Sum of all the numbers that are formed from root to leaf paths

Given a binary tree, where every node value is a Digit from 1-9 .Find the sum of all the numbers which are formed from root to leaf paths.

For example consider the following Binary Tree.

```
                                              6
                                          /      \
                                        3          5
                                      /   \          \
                                     2     5          4
                                         /   \
                                        7     4
   There are 4 leaves, hence 4 root to leaf paths:
    Path                     Number
   6->3->2                    632
   6->3->5->7                6357
   6->3->5->4                6354
   6->5>4                     654
 Answer = 632 + 6357 + 6354 + 654 = 13997
```

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

The idea is to do a preorder traversal of the tree. In the preorder traversal, keep track of the value calculated till the current node, let this value be *val*. For every node, we update the *val* as *val*10* plus node's data.

C

```
// C program to find sum of all paths from root to leaves
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int data;
    struct node *left, *right;
};

// function to allocate new node with given data
struct node* newNode(int data)
{
    struct node* node = (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = node->right = NULL;
    return (node);
}

// Returns sum of all root to leaf paths. The first parameter is root
// of current subtree, the second parameter is value of the number formed
// by nodes from root to this node
int treePathsSumUtil(struct node *root, int val)
{
    // Base case
    if (root == NULL)  return 0;

    // Update val
    val = (val*10 + root->data);

    // if current node is leaf, return the current value of val
    if (root->left==NULL && root->right==NULL)
       return val;

    // recur sum of values for left and right subtree
    return treePathsSumUtil(root->left, val) +
           treePathsSumUtil(root->right, val);
}

// A wrapper function over treePathsSumUtil()
int treePathsSum(struct node *root)
{
    // Pass the initial value as 0 as there is nothing above root
    return treePathsSumUtil(root, 0);
}

// Driver function to test the above functions
int main()
{
    struct node *root = newNode(6);
    root->left        = newNode(3);
    root->right       = newNode(5);
    root->left->left  = newNode(2);
    root->left->right = newNode(5);
    root->right->right = newNode(4);
    root->left->right->left = newNode(7);
    root->left->right->right = newNode(4);
    printf("Sum of all paths is", treePathsSum(root));
    return 0;
}
```

Run on IDE

Java

```
// Java program to find sum of all numbers that are formed from root
// to leaf paths

// A binary tree node
```

```java
class Node {

    int data;
    Node left, right;

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

class BinaryTree {

    static Node root, prev;

    // Returns sum of all root to leaf paths. The first parameter is root
    // of current subtree, the second parameter is value of the number formed
    // by nodes from root to this node
    int treePathsSumUtil(Node node, int val) {

        // Base case
        if (node == null) {
            return 0;
        }

        // Update val
        val = (val * 10 + node.data);

        // if current node is leaf, return the current value of val
        if (node.left == null && node.right == null) {
            return val;
        }

        // recur sum of values for left and right subtree
        return treePathsSumUtil(node.left, val)
                + treePathsSumUtil(node.right, val);
    }

    // A wrapper function over treePathsSumUtil()
    int treePathsSum(Node node) {

        // Pass the initial value as 0 as there is nothing above root
        return treePathsSumUtil(node, 0);
    }

    // driver program to test above functions
    public static void main(String args[]) {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(6);
        tree.root.left = new Node(3);
        tree.root.right = new Node(5);
        tree.root.right.right = new Node(4);
        tree.root.left.left = new Node(2);
        tree.root.left.right = new Node(5);
        tree.root.left.right.right = new Node(4);
        tree.root.left.right.left = new Node(7);

        System.out.print("Sum of all paths is " + tree.treePathsSum(root));
    }
}

// This code has been contributed by Mayank Jaiswal
```

Run on IDE

Python

```python
# Python program to find sum of all paths from root to leaves

# A Binary tree node
class Node:

    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# Returs sums of all root to leaf paths. The first parameter is root
# of current subtree, the second parameter is value of the number
# formed by nodes from root to this node
def treePathsSumUtil(root, val):

    # Base Case
    if root is None:
        return 0

    # Update val
    val = (val*10 + root.data)

    # If current node is leaf, return the current value of val
    if root.left is None and root.right is None:
        return val

    # Recur sum of values for left and right subtree
    return (treePathsSumUtil(root.left, val) +
            treePathsSumUtil(root.right, val))

# A wrapper function over treePathSumUtil()
def treePathsSum(root):

    # Pass the initial value as 0 as ther is nothing above root
    return treePathsSumUtil(root, 0)

# Driver function to test above function
root = Node(6)
root.left = Node(3)
root.right = Node(5)
root.left.left = Node(2)
root.left.right = Node(5)
root.right.right = Node(4)
root.left.right.left = Node(7)
root.left.right.right = Node(4)
print "Sum of all paths is", treePathsSum(root)

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

Run on IDE

Output:

```
Sum of all paths is 13997
```

Time Complexity: The above code is a simple preorder traversal code which visits every exactly once. Therefore, the time complexity is O(n) where n is the number of nodes in the given binary tree.

This article is contributed by **Ramchand R**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

33 Comments Category: Trees

Related Posts:

- Check if removing an edge can divide a Binary Tree in two halves
- Check sum of Covered and Uncovered nodes of Binary Tree
- Lowest Common Ancestor in a Binary Tree | Set 2 (Using Parent Pointer)
- Construct a Binary Search Tree from given postorder
- BFS vs DFS for Binary Tree
- Maximum difference between node and its ancestor in Binary Tree
- Inorder Non-threaded Binary Tree Traversal without Recursion or Stack
- Check if leaf traversal of two Binary Trees is same?

(Login to Rate and Mark)

2.6 Average Difficulty : **2.6/5.0**
Based on **3** vote(s)

☐ Add to TODO List
☐ Mark as DONE

Like Share 11 people like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.