GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

Login

Home Algorithms DS GATE Interview Corner Q&A C C++ Java Books Contribute Ask a Q About

Array Bit Magic C/C++ Articles GFacts Linked List MCQ Misc Output String Tree Graph

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.

Google™ Custom Search

GeeksforGeeks

52,731 people like GeeksforGeeks.

Interview Experiences

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

Pattern Searching

Divide & Conquer

Mathematical Algorithms

Recursion

Geometric Algorithms

```
// 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 for
// 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);
}
```

Popular Posts

All permutations of a given string

Memory Layout of C Programs

Understanding “extern” keyword in C

Median of two sorted arrays

Tree traversal without recursion and without stack!

Structure Member Alignment, Padding and Data Packing

Intersection point of two Linked Lists

Lowest Common Ancestor in a BST.

Check if a binary tree is BST or not

Sorted Linked List to Balanced BST

```
// Driver function to test the above functions
int main()
{
    struct node *root = newNode(6);
    root->left        = newNode(3);
    root->right       = newNode(5);
    root->right->right= newNode(7);
    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 %d", treePathsSum(root));
    return 0;
}
```

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

Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)
- Print Right View of a Binary Tree
- Red-Black Tree | Set 3 (Delete)
- Construct a tree from Inorder and Level order traversals
- Print all nodes at distance k from a given node
- Print a Binary Tree in Vertical Order | Set 1
- Interval Tree
- Check if a given Binary Tree is height balanced like a Red-Black Tree

12 Tweet 3 1

Writing code in comment? Please use **ideone.com** and share the link here.

18 Comments **GeeksforGeeks**

Sort by Newest ▾

Join the discussion

695 Subscribe

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 27 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 47 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 47 minutes ago

@meya Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Join the discussion…

Ronak Hingar · 3 months ago

Awesome code!!!

1 ⌃ | ⌄ • Reply • Share ›

Mário Barbosa · 3 months ago

Python

http://pastebin.com/EKHi7WTP

⌃ | ⌄ • Reply • Share ›

Sarthak · 3 months ago

This code seem Incorrect. You are summing the numbers at each node. How when you encounter a leaf.

call the below code with num=0 and ans=0. I have tested this code and submi

```
void sum(TreeNode *root, int num, int *ans){
if(root == NULL){
return;
}

num = num*10+ root->val;
if(!root->left && !root->right){

*ans += num;
return;
}
sum(root->left,num,ans);
sum(root->right,num,ans);
}
```

1 ⌃ | ⌄ • Reply • Share ›

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

Kartik ➦ Sarthak • 3 months ago

Thanks for sharing your thought. The code looks fine to me. Could you code doesn't work.

^ | ˅ • Reply • Share ›

Sarthak ➦ Kartik • 3 months ago

My mistake. The code given in this page is correct.

^ | ˅ • Reply • Share ›

Aniket Thakur • 3 months ago

There appears to be typing error in the example provided . Answer must be 63 code with output --> http://opensourceforgeeks.blog...

^ | ˅ • Reply • Share ›

GeeksforGeeks ➦ Aniket Thakur • 3 months ago

Thanks for pointing this out. We have corrected typo.

^ | ˅ • Reply • Share ›

Gaurav pruthi • 3 months ago

//The program is to to sum the numbers created from root to leaf traversal of r

```
#include<iostream>
#include<cmath>

using namespace std;

typedef struct treeStruct{
int element;
struct treeStruct *left;
struct treeStruct *right;
}treeStruct;
```

```
struct treeStruct* newNode(int data)
{
struct treeStruct *newElement = new(struct treeStruct);
newElement->left = NULL;
newElement->right = NULL;
newElement->element = data;
```

see more

⌃ | ⌄ • Reply • Share ›

Lathe • 3 months ago

Here is the one in Python:

```
class Tree:

def __init__(self, value):
self.value = value
self.left = None
self.right = None

def ListRootLeaf(self, col, ls=[]):
ls.append(self.value)

if self.left:
self.left.ListRootLeaf(col, ls)
if self.right:
self.right.ListRootLeaf(col, ls)

if not (self.left or self.right):
col.append(tuple(ls))
```

see more

⌃ | ⌄ • Reply • Share ›

SBUser • 3 months ago

```
#include <stdio.h>

#include <iostream>

#include <queue>

#include <stack>

#include <utility>

using namespace std;

typedef struct tree {

int val;
```

see more

⌃ | ⌄ • Reply • Share ›

SBUser • 3 months ago

Iterative version using stack:

```
int sumofall2(TREE *node) {

std::stack <std::pair <tree="" *,int=""> > tstack;

tstack.push(std::pair<tree *,int=""> (node, node->val));
```

```
int sum = 0;

std::vector<int> final;

// printf("%d, %d", ((TREE *)tstack.top().first)->val, tstack.top().second);

while (!tstack.empty()) {

TREE *temp = (TREE *)tstack.top().first;

int tval = (int)tstack.top().second;

tstack.pop();
```

see more

• Reply • Share ›

dhrumil • 3 months ago

will this code work fine if the tree has a root node and only a single left node bu

• Reply • Share ›

Gaurav pruthi dhrumil • 3 months ago

I dont think so

• Reply • Share ›

Kartik Gaurav pruthi • 3 months ago

seems to be working for all cases. could u provide an example

• Reply • Share ›

Sreenivas • 3 months ago

```
// SumRootToLeaf.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"
```

```
#include <iostream>

using namespace std;

int g_TotalSum = 0;

struct Node

{

int data;

Node *left;

Node *right;
```

see more

^ | ⌄ • Reply • Share ›

Dikshith Gokhale • 3 months ago

Brilliant:):) I would have done using the print path method from root to leaf and global variable.. This is definitely better..
Thanks for sharing..

2 ^ | ⌄ • Reply • Share ›

Santosh Kumar • 3 months ago

```
package com.NewsCast;

import java.util.ArrayList;

import java.util.List;

public class BSTtest {

static List<treenode> leafList = new ArrayList<treenode>(4);
static List<treenode> leafList = new ArrayList<treenode>(4);

class TreeNode {

int val;

TreeNode left, right;

TreeNode(TreeNode t1, TreeNode t2, int i) {

this.left = t1;

this.right = t2;
```

see more

^ | ⌄ • Reply • Share ›

Rahman • 3 months ago

Awesome code..Power of recusion :)

1 ^ | ⌄ • Reply • Share ›

✉ Subscribe Ⓓ Add Disqus to your site