Find the maximum sum leaf to root path in a Binary Tree

Given a Binary Tree, find the maximum sum path from a leaf to root. For example, in the following tree, there are three leaf to root paths 8->-2->10, -4->-2->10 and 7->10. The sums of these three paths are 16, 4 and 17 respectively. The maximum of them is 17 and the path for maximum is 7->10.

```
              10
            /     \
          -2       7
        /   \
      8     -4
```

Solution

1) First find the leaf node that is on the maximum sum path. In the following code getTargetLeaf() does this by assigning the result to *target_leaf_ref.
2) Once we have the target leaf node, we can print the maximum sum path by traversing the tree. In the following code, printPath() does this.

The main function is maxSumPath() that uses above two functions to get the complete solution.

```
#include<stdio.h>
#include<limits.h>

/* A tree node structure */
struct node
{
    int data;
    struct node *left;
```


Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

```
    struct node *right;
};

// A utility function that prints all nodes on the path from root to t
bool printPath (struct node *root, struct node *target_leaf)
{
    // base case
    if (root == NULL)
        return false;

    // return true if this node is the target_leaf or target leaf is p
    // one of its descendants
    if (root == target_leaf || printPath(root->left, target_leaf) ||
            printPath(root->right, target_leaf) )
    {
        printf("%d ", root->data);
        return true;
    }

    return false;
}

// This function Sets the target_leaf_ref to refer the leaf node of th
// path sum.  Also, returns the max_sum using max_sum_ref
void getTargetLeaf (struct node *node, int *max_sum_ref, int curr_sum,
                    struct node **target_leaf_ref)
{
    if (node == NULL)
        return;

    // Update current sum to hold sum of nodes on path from root to th
    curr_sum = curr_sum + node->data;

    // If this is a leaf node and path to this node has maximum sum so
    // then make this node target_leaf
    if (node->left == NULL && node->right == NULL)
    {
        if (curr_sum > *max_sum_ref)
        {
            *max_sum_ref = curr_sum;
            *target_leaf_ref = node;
        }
    }

    // If this is not a leaf node, then recur down to find the target_
    getTargetLeaf (node->left, max_sum_ref, curr_sum, target_leaf_ref)
    getTargetLeaf (node->right, max_sum_ref, curr_sum, target_leaf_ref
```


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

```
}

// Returns the maximum sum and prints the nodes on max sum path
int maxSumPath (struct node *node)
{
    // base case
    if (node == NULL)
        return 0;

    struct node *target_leaf;
    int max_sum = INT_MIN;

    // find the target leaf and maximum sum
    getTargetLeaf (node, &max_sum, 0, &target_leaf);

    // print the path from root to the target leaf
    printPath (node, target_leaf);

    return max_sum;  // return maximum sum
}

/* Utility function to create a new Binary Tree node */
struct node* newNode (int data)
{
    struct node *temp = new struct node;
    temp->data = data;
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}

/* Driver function to test above functions */
int main()
{
    struct node *root = NULL;

    /* Constructing tree given in the above figure */
    root = newNode(10);
    root->left = newNode(-2);
    root->right = newNode(7);
    root->left->left = newNode(8);
    root->left->right = newNode(-4);

    printf ("Following are the nodes on the maximum sum path \n");
    int sum = maxSumPath(root);
    printf ("\nSum of the nodes is %d ", sum);
```

```
    getchar();
    return 0;
}
```

Output:

```
Following are the nodes on the maximum sum path
7 10
Sum of the nodes is 17
```

Time Complexity: Time complexity of the above solution is O(n) as it involves tree traversal two times.

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)

Recent Comments

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

Backtracking | Set 7 (Sudoku) · 37 minutes ago

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

Flipkart Interview | Set 6 · 57 minutes ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 57 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) {...

Given a linked list, reverse alternate nodes and append at the end · 3 hours ago

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

Find depth of the deepest odd level leaf node · 3 hours ago

- 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

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

44 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

isidorouk • 3 months ago

Can some one help me to understand what is wrong with this method, which s

```
private long GetMax(int row, int column, Pyramid pyramid)

{

if (row == 0) return pyramid[row, column];

long value = pyramid[row, column];

if (row != 0)

{

long left = pyramid[row - 1, column];

long right = pyramid[row - 1, column + 1];
```

long highest = Math.Max(left, right);

int nextColumn = highest == left ? column : column + 1;

value += GetTotalAbove(row – 1, nextColumn, pyramid);

}

return value;

}

• Reply • Share ›

Akhil • 11 months ago

To print the maximum sum :
An O(n) postorder simple code

```
 #include<stdio.h>
#include<stdlib.h>
struct tree
{
    int info;
    struct tree *l;
    struct tree *r;
};
typedef struct tree *Tree;

Tree newNode(int num)
{
    Tree temp = (Tree)malloc(sizeof(struct tree));
    temp->info = num;
    temp->l = NULL;
```

see more

1 • Reply • Share ›

Guest → Akhil • 8 months ago

And in Haskell:

```
data Tree a =
Empty
| Node (Tree a) a (Tree a)
deriving (Eq,Show)

maxRootLeaf :: Tree Int -> Int
maxRootLeaf Empty = 0
maxRootLeaf (Node l x r) = x + (if maxRootLeaf (r) > maxRootLeaf
then maxRootLeaf (r)
else maxRootLeaf (l))
```

• Reply • Share ›

NNavneet • 11 months ago

"The sums of these three paths are 16, 8 and 17 respectively."
It should be
" The sums of these three paths are 16, 4 and 17 respectively." In the first para

```
/* Paste your code here (You may delete these lines if not writing co
```

• Reply • Share ›

GeeksforGeeks → NNavneet • 11 months ago

Thanks for pointing this out. We have corrected it. Keep it up!

• Reply • Share ›

ysunil040 • 11 months ago

```
 int maxSum(Tree root,int toPrint)
{
        if (root == NULL)
                return INT_MIN;
        if (root->left == NULL && root->right == NULL)
        {
                if (toPrint)
                        printf("%d\n",root->data);
                return root->data;
        }
        int lsum = maxSum(root->left,0);
        int rsum = maxSum(root->right,0);

        if (toPrint)
        {
                printf("%d ",root->data);
                int sum = root->data;
                if (lsum >= rsum)
```

see more

• Reply • Share ›

ysunil040 • 11 months ago

```
 int maxSum(Tree root,int toPrint)
{
        if (root == NULL)
                return INT_MIN;
        if (root->left == NULL && root->right == NULL)
        {
                if (toPrint)
                        printf("%d\n",root->data);
                return root->data;
```

```
        }
        int lsum = maxSum(root->left,0);
        int rsum = maxSum(root->right,0);

        if (toPrint)
        {
                printf("%d ",root->data);
                int sum = root->data;
                if (lsum >= rsum)
```

see more

• Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

• Reply • Share ›

ishanu • a year ago

//i dont understand why people are writing so complicated codes.this can be d

```
#include <iostream>
//to find the maximum distance between root and a leaf
using namespace std;

struct bst
{
        int data;
        bst *lchild;
        bst *rchild;
};
bst* root=NULL;
```

```
bst* total=NULL;
bst* newNode(int num)
{
        bst*node= new bst;
```

see more

Reply • Share ›

ishanu ishanu • a year ago

a minor change to the above code

```
 #include <iostream>
//to find the maximum distance between root and a leaf
using namespace std;

struct bst
{
        int data;
        bst *lchild;
        bst *rchild;
};
bst* root=NULL;

bst* newNode(int num)
{
        bst*node= new bst;
        node->data=num;
```

see more

Reply • Share ›

ishanu ishanu • a year ago

a minor change to the above code

```
 #include <iostream>
//to find the maximum distance between root and a leaf
using namespace std;

struct bst
{
        int data;
        bst *lchild;
        bst *rchild;
};
bst* root=NULL;
bst* total=NULL;
bst* newNode(int num)
{
        bst*node= new bst;
        node->data=num;
```

see more

• Reply • Share ›

sirisha • a year ago

```
 main()
{
struct node *root=NULL;
root=newnode(10);
root->left=newnode(-2);
root->right=newnode(7);
root->left->left=newnode(8);
root->left->right=newnode(-4);
print_max_sum_path(root);
```

```
struct node* newnode(int n)
{
struct node *nu;
nu=(struct node*)malloc(sizeof(struct node));
nu->data=n;
nu->left=NULL;
nu->right=NULL;
return nu;
```

see more

• Reply • Share ›

Karthik • a year ago

How about simple postorder traversal?

[sourcecode language="C++"]
class Node
{
public:
Node* left;
Node* right;
int data;
Node(int data, Node*right = NULL, Node* left = NULL):data(data),right(right),lef
};
int postorderSum(Node* root, vector<int> &list)
{
if(!root)
return 0;

vector<int> rightList, leftList;
int rightSum = postorderSum(root->right, rightList);
int leftSum = postorderSum(root->left, leftList);

see more

• Reply • Share ›

Nikhil • 2 years ago

[sourcecode language="C++"]
#include<iostream>
using namespace std;
#include<stdio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *left;
struct node *right;
};
typedef struct node *nodeptr;

nodeptr head=NULL;
nodeptr maketree(int x)
{
nodeptr p;
p=(nodeptr)malloc(sizeof(struct node));

see more

• Reply • Share ›

RJ • 2 years ago

```
 #include <iostream>
#include<string>
#include<cstdlib>
using namespace std;
```

```
struct node
{
    int data;
    struct node *left;
    struct node *right;
};
void PrintMaxSum(struct node* , int , int&  , string , string&);
void Print(struct node* root)
{
        int sum = 0;
        int maxsum = 0;
        string path ="";
        string maxpath="";
```

see more

• Reply • Share ›

Kundan • 2 years ago

Sir, The explanation is good. But if the solution for the problem is little bit in an learner to understand in a crystal clear way. apologies if my idea doesnt fit her

```
/* Paste your code here (You may delete these lines if not writing c
```

• Reply • Share ›

RameshSuthan • 2 years ago

int maxSumtoLeaf(tnodeptr node,int sum,char *decisionTree,int nodePos)
{
if(node==NULL)
{
return sum;
}

```
int nodesum = sum + node->val;
int leftSum = maxSumtoLeaf(node->left,nodesum,decisionTree,left(nodePos)
int rightSum = maxSumtoLeaf(node->right,nodesum,decisionTree,right(nodeF

decisionTree[nodePos]= (leftSum > rightSum)? 'L' : 'R';
int max = ( leftSum > rightSum ) ? leftSum : rightSum ;

return max;
}

void printMaxSumPath(tnodeptr node,char* decisionTree,int index)
{
```

see more

• Reply • Share ›

RameshSuthan • 2 years ago

```
 /* Paste your code here (You may delete these lines if not writing c
int maxSumtoLeaf(tnodeptr node,int sum,char *decisionTree,int nodePos
{
        if(node==NULL)
        {
                return sum;
        }

        int nodesum = sum + node->val;
        int leftSum  = maxSumtoLeaf(node->left,nodesum,decisionTree,le
        int rightSum =  maxSumtoLeaf(node->right,nodesum,decisionTree,

        decisionTree[nodePos]= (leftSum > rightSum)? 'L' : 'R';
        int max =  ( leftSum > rightSum ) ? leftSum : rightSum ;
```

```
        return max;
}
```

see more

Reply • Share ›

Robin • 2 years ago

Thanks

Reply • Share ›

Ramesh Suthan • 2 years ago

```
 int maxSumtoLeaf(tnodeptr node,int sum,char *decisionTree,int nodePos
{
        if(node==NULL)
        {
                return sum;
        }

        int nodesum = sum + node->val;
        int leftSum  = maxSumtoLeaf(node->left,nodesum,decisionTree,le
        int rightSum =  maxSumtoLeaf(node->right,nodesum,decisionTree,

        decisionTree[nodePos]= (leftSum > rightSum)? 'L' : 'R';
        int max =  ( leftSum > rightSum ) ? leftSum : rightSum ;

        return max;
}
```

see more

Reply • Share ›

Shipra Agrawal • 2 years ago

I too have a recursive method which also prints the path from leaf to root.

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#include<stdlib.h>
void createbst(int,struct tree**);
int maxsump(struct tree*);
int max(int,int);
struct tree{
                int val;
                struct tree* left;
                struct tree* right;
                };
void main()
{
        clrscr();
```

see more

• Reply • Share ›

Sreenivas ➦ Shipra Agrawal • 4 months ago

This is simple and good :)

• Reply • Share ›

Anil arya • 2 years ago

/* Paste your code here (You may delete these lines if not writing code) */

[/ void max_sum(struct node *root,int *max)

{

```
if(root==NULL)
return ;

if(root->left==NULL&&root->right==NULL)
{
if(root->data>*max)
*max=root->data;

}

if(root->left)
root->left->data+=root->data;

if(root->right)
root->right->data+=root->data;

max_sum(root->left,max);
max_sum(root->right,max);

}
]
```

• Reply • Share ›

xerox ➦ Anil arya • 2 years ago

this code changes the tree structure and doesn't prints path

```
/* Paste your code here (You may delete these lines if not wri
```

• Reply • Share ›

Algogeek • 2 years ago

we can print from root to leaf by little modification in the print_path() function.

```
int array[20]={0};
int path_print(struct node* root,struct node* targetleaf)
{
    static int i=0;
    if(root==NULL)
        return 0;
    if((root==targetleaf)||path_print(root->left,targetleaf)||path_pr
    {
        array[i++]= root->data;
        return i;
    }
return 0;
}
```

In the main function, we can print the array elements from i-1 to 0. This will be

• Reply • Share ›

red → Algogeek • 2 years ago

Yeah nice . :)

```
/* Paste your code here (You may delete these lines if not wr
```

• Reply • Share ›

Arpit • 2 years ago

```
 /* Paste your code here (You may delete these lines if not writing c
#include<stdio.h>
#include<limits.h>

struct node
```

```
{
        int data;
        struct node *left;
        struct node *right;
}*root;

struct node *newNode(int key)
{
        struct node*tmp=(struct node*)malloc(sizeof(struct node));
        tmp->left=NULL;
        tmp->right=NULL;
        tmp->data=key;
        return tmp;
```

see more

• Reply • Share ›

bobby • 2 years ago

I found a recursive solution that uses extra memory to track and print the path.
http://www.codingissue.com/Que...

• Reply • Share ›

ani • 2 years ago

i need code to draw parse tree

• Reply • Share ›

Gautam • 2 years ago

```
 #include<stdio.h>
#include<stdlib.h>
#include<limits.h>

/* A binary tree node has data, pointer to left child
```

```
   and a pointer to right child */
struct node
{
   int data;
   struct node* left;
   struct node* right;
};

 //static int maxsum=INT_MIN;
 int maxpath[1000];
 int maxsumlen=0;
/* Prototypes for funtions needed in printPaths() */
void printPathsRecur(struct node* node, int path[], int pathLen);
```

see more

• Reply • Share ›

Sambasiva • 2 years ago

http://effprog.blogspot.com/2011/06/print-path-of-minimum-sum-in-binary.html

• Reply • Share ›

Daddy • 2 years ago

```
 #include<stdio.h>
#include<conio.h>
#include<malloc.h>

struct node{
       int data;
       struct node* left;
       struct node* right;
       };
int maxsum,k;
```

```
int d[10];
struct node* insert(struct node* ,int );
struct node* newnode(int );
void preorder(struct node*);
void printmaxpath(struct node* );
void checkallpaths(struct node*,int [],int);
void maxpath(struct node*,int [],int);
void printpath(int [],int);
```

see more

• Reply • Share ›

akshayjohri • 2 years ago

```
 /* Paste your code here (You may delete these lines if not writing c
int max=0;
char *maxdir="";
char *curdir="";
void push(char ch){
curdir[++top]=ch;
}
char pop(){
return curdir[top--];
}

void maxpath(Node root,int val,char dir){
    if(root==NULL) return 0;
    root->val= root->val + val;
    push(dir);
    if(root->val>max){
       max=root->val;
       strcpy(maxdir,curdir);
```

see more

• Reply • Share ›

Agniswar • 2 years ago

Here is the recursive function i have written..

```
 /* Paste your code here (You may delete these lines if not writing c

int maxSumPath(node *root)
{
    if(root==NULL)
                  return 0;
    else
    {
        int ls=maxSumPath(root->left);
        int rs=maxSumPath(root->right);

        int max=(ls>rs)? ls: rs;
        return root->data + max;
    }
}
```

see more

• Reply • Share ›

rohit ➦ Agniswar • 2 years ago

> would you like to share the printing code

• Reply • Share ›

gaurav ➦ Agniswar • 2 years ago

• Reply • Share ›

dumbcoder ➦ gaurav • 2 years ago

```
#include
#include
#include

/* A tree node structure */
struct Node
{
Node(int dat) {
data = dat;
left = NULL;
right = NULL;
maxSum = INT_MIN;
}

~Node() {
if (left != NULL) delete left;
if (right != NULL) delete right;
}
```

see more

• Reply • Share ›

kartik ➦ Agniswar • 2 years ago

@Agniswar: thanks for suggesting a new method. This seems good. C
well. We will add it to the original post.

• Reply • Share ›

Nitin Gupta • 2 years ago

I have another method which work recursively....

take
a static variable called Current_value it will contain the value so far which have

a current_sum variable which will hold current sum.

a max_sum variable which will hold maximum sum connected with current_va

we are going to find maximum sum from left side of the root then proceed righ

example:
initially Current_sum = max_sum= current_value=0
Say you have tree like

```
                10
              /      \
            -2        7
          /   \
        8      -4
```

see more

• Reply • Share ›

avinash • 2 years ago

```
I think it is printing the path from leaf to root.
```

• Reply • Share ›

GeeksforGeeks → avinash • 2 years ago

@avinash: Thanks for pointing this out. We have changed the title of th

• Reply • Share ›

g33k • 2 years ago

I have a recursive algorithm, which I believe is correct.

```
int maxPath(Node root){

if(root == null) return 0;
//find sum path of left tree and right tree
int maxLeft = maxPath(root->left);
int maxRight = maxPath(root->right);

//max can be either of the two
if(maxLeft > maxRight){
retunrn max(maxLeft, root, root + maxLeft);
else
return max(maxRIght, root, root+maxRight);

}
```

I will modify the code to get the path or someone can do. It should be easy.

• Reply • Share ›

lla → g33k • 2 years ago

When you are taking max(maxLeft,root,root+maxleft) u may or may no but the question requires to display a complete path from root to leaf if return max(maxRight,maxLeft) and store whichever node returns max

• Reply • Share ›

g33k → lla • 2 years ago

I think you are right. Just remove the root from max funtcion.

```
/* Paste your code here (You may delete these lines if
```

• Reply • Share ›