Given a binary tree, print out all of its root-to-leaf paths one per line.

Asked by Varun Bhatia

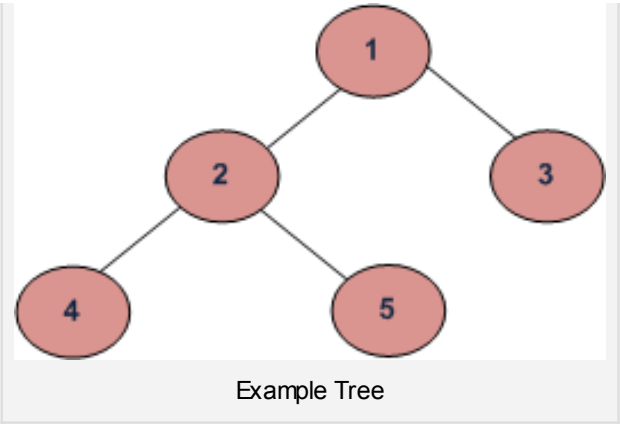
Here is the solution.

Algorithm:

```
initialize: pathlen = 0, path[1000]
/*1000 is some max limit for paths, it can change*/

/*printPathsRecur traverses nodes of tree in preorder */
printPathsRecur(tree, path[], pathlen)
   1) If node is not NULL then
         a) push data to path array:
               path[pathlen] = node->data.
         b) increment pathlen
               pathlen++
   2) If node is a leaf node then print the path array.
   3) Else
        a) Call printPathsRecur for left subtree
                printPathsRecur(node->left, path, pathLen)
        b) Call printPathsRecur for right subtree.
               printPathsRecur(node->right, path, pathLen)
```

Example:



Example Tree

Output for the above example will be

```
1 2 4
1 2 5
1 3
```

Implementation:

```
/*program to print all of its root-to-leaf paths for a tree*/
#include <stdio.h>
#include <stdlib.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;
};

void printArray(int [], int);
void printPathsRecur(struct node*, int [], int);
struct node* newNode(int );
void printPaths(struct node*);

/* Given a binary tree, print out all of its root-to-leaf
   paths, one per line. Uses a recursive helper to do the work.*/
void printPaths(struct node* node)
{
  int path[1000];
  printPathsRecur(node, path, 0);
```


Popular Posts

```
}

/* Recursive helper function -- given a node, and an array containing
 the path from the root node up to but not including this node,
 print out all the root-leaf paths. */
void printPathsRecur(struct node* node, int path[], int pathLen)
{
  if (node==NULL) return;

  /* append this node to the path array */
  path[pathLen] = node->data;
  pathLen++;

  /* it's a leaf, so print the path that led to here */
  if (node->left==NULL && node->right==NULL)
  {
    printArray(path, pathLen);
  }
  else
  {
  /* otherwise try both subtrees */
    printPathsRecur(node->left, path, pathLen);
    printPathsRecur(node->right, path, pathLen);
  }
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode(int data)
{
  struct node* node = (struct node*)
                      malloc(sizeof(struct node));
  node->data = data;
  node->left = NULL;
  node->right = NULL;

  return(node);
}

/* Utility that prints out an array on a line */
void printArray(int ints[], int len)
{
  int i;
  for (i=0; i<len; i++) {
    printf("%d ", ints[i]);
  }
  printf("\n");
```

```
}

/* Driver program to test mirror() */
int main()
{
  struct node *root = newNode(1);
  root->left        = newNode(2);
  root->right       = newNode(3);
  root->left->left  = newNode(4);
  root->left->right = newNode(5);

  /* Print all root-to-leaf paths of the input tree */
  printPaths(root);

  getchar();
  return 0;
}
```

References:

http://cslibrary.stanford.edu/110/BinaryTrees.html

Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)

Subscribe

Recent Comments

karthik it should have been max_wrap= max_wrap -...

Maximum circular subarray sum · 1 minute ago

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

Backtracking | Set 7 (Sudoku) · 45 minutes ago

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

Flipkart Interview | Set 6 · 1 hour ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 1 hour 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

- 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

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

26 Comments GeeksforGeeks

Sort by Newest

Tony • a month ago

```
public void allPathSum(BNode root,Stack<bnode> s){

if (root == null){
return;

}
s.add(root);
if (root.left == null && root.right == null){
int sum =0;
for (BNode node : s){
System.out.print(node.data+" + ");
sum =sum+node.data;
}
System.out.print(" = "+(sum)+" \n");
```

```
}

allPathSum(root.left,s);

allPathSum(root.right,s);
s.pop();

}
```

Reply • Share ›

Marsha Donna • 2 months ago

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node
{
char data;
struct node *left;
struct node *right;
};

struct node* insert(char item)
{
struct node *temp=(struct node *)malloc(sizeof(struct node));
temp->data=item;
temp->left=NULL;
temp->right=NULL;
return temp;
}
```

see more

Reply • Share ›

cruser11232 • 2 months ago

void PrintPathSub(node * root, stack * stk)
{
if(root== NULL)
return;

push(stk,root->data);
PrintPathSub(root->left,stk);

PrintPathSub(root->right,stk);

if(root->left==NULL && root->right==NULL)
{
print(stk);
}
pop(stk);
}

  • Reply • Share ›

Kush Pandey • 9 months ago

A recursive implementation of above problem

```
 void printpath()
{
    printReculen(root,0)
}
void printReculen(node *root,int length)
{
    int i=1,path[50];
    if(root==0)
    {
        while(i<length)
       {
```

```
        printf("%d",path[i]);
       }
        printf("\n");
        return
    }
    path[length]=root->data;
    length++;
    printReculen(root->left,length);
    printReculen(root->right,length);
}
```

Reply • Share ›

shek8034 • 11 months ago

This post is duplicate of
http://www.geeksforgeeks.org/g...

Reply • Share ›

Sasuke • 11 months ago

Here is my code for the same using stack

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

typedef struct tNode
{
    int data ;
    struct tNode *left ;
    struct tNode *right ;
} tNode ;

typedef struct sNode
```

```
{
    int data;
    struct sNode *next ;
} sNode ;
```

see more

• Reply • Share ›

huntur • 11 months ago

```
//iterative solution
void roottoleaf(struct node *root)
{
struct node *t;
int a[MAX],i,j,flag=0;
i=0;
t=root;
while(1)
{
while(t->left)
{
push(t);
a[i]=t->data;
i++;
t=t->left;
}
while(t->right==NULL)
{
a[i]=t->data;
for(j=0;jright;
i++;
}
```

```
}
}
}
```

Reply • Share ›

abhishek08aug • a year ago

C++ code: extended from my last post on: http://www.geeksforgeeks.org/g...

```
 #include <iostream>
#include <stdlib.h>
using namespace std;

class tree_node {
  private:
    int data;
    tree_node * left;
    tree_node * right;
  public:
    tree_node() {
      left=NULL;
      right=NULL;
    }
    void set_data(int data) {
      this->data=data;
```

see more

Reply • Share ›

abhishek07july ➦ abhishek08aug • 9 months ago

inelligent ;)

very tricky code...

Reply • Share ›

Nishant Kumar • a year ago

We can also use Queue in place of array for less auxiliary space

```
 void printPath(tree* root,node* top){
    if(root==NULL)
    return;

    enQueue(top,root->data);

    if(root->left == NULL && root->right == NULL){
        printQueue(top);
        printf("\n");
        deQueue(top);
    }
    else{
        printPath(root->left,top);
        printPath(root->right,top);
        deQueue(top);
    }
}
```

  • Reply • Share ›

stupid • a year ago

```
void printarr(int a[],int level){
     if(level<0){
              printf("\n");
              return;
     }
     printarr(a,level-1);
     printf("%d ",a[level]);
```

```
        printf("%d  ",a[level]);
}
void findroottoleafpath(struct node * root,int a[],int level){
        if(root == NULL){
                return;
        }
        a[level]=root->data;
        level++;
        if(root->left == NULL && root->right == NULL){
                        printarr(a,level-1);
        }
        findroottoleafpath(root->left,a,level);
        findroottoleafpath(root->right,a,level);
}
```


Nikin • a year ago

```
void printArray(int path[], int pathLen)
{
for(int i=0;i<pathLen;i++)
cout<<path[i];
}

void printPathsRecur(node *sr, int path[], int pathLen)
{
if(sr == NULL) return;
path[pathLen++] = sr->data;
if(sr->left == NULL && sr->right == NULL)
{
```

```
printArray(path, pathLen);
return;
}
printPathsRecur(sr->left);
```

see more

Reply • Share ›

Sandeep Jain • a year ago

Please take a closer look at the program. It works fine. pathLen doesn&#039t r ancestors of two leaf nodes are in paths of both nodes.

Reply • Share ›

Sandeep Bc • a year ago

Hi.

The pathlen variable is NOT reset when a leaf node is reached , This needs to a leaf node is reached.

if (node->left==NULL && node->right==NULL) {.
printArray(path, pathLen);
pathlen = 0;.

Reply • Share ›

trilok sharma • a year ago

/*
6
5 2 8 9 6 11
8 2 9 5 11 6
*/

#include
#include

```
#include
using namespace std;

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

node* Newnode(int data)
```

see more

• Reply • Share ›

sandeep • a year ago

Hi.

The pathlen variable is NOT reset when a leaf node is reached ,This needs to leaf node is reached.

```
if (node->left==NULL && node->right==NULL) {
printArray(path, pathLen);
pathlen = 0;
```

• Reply • Share ›

mrn • a year ago

```
/* Paste your code here (You may delete these lines if not writing co
void root_to_leaf(Node *n,vector<int> &v)
{
        if(n->l==NULL && n->r==NULL)
        {
```

```
                for(vector<int>::iterator it=v.begin();it!=v.end();it
                                cout<<*it<<" ";
                cout<<n->v;
                cout<<endl;
        return;
        }
        v.push_back(n->v);
        if(n->l)
        root_to_leaf(n->l,v);
        if(n->r)
        root_to_leaf(n->r,v);
        v.pop_back();
return;
}
```


Sahil • 2 years ago

There is no need to allocate any memory to print the paths.

```
void printallpaths(struct node* node)
{
     if(node==NULL){return;}

//Insert a newline character when the node is leaf

     if(node->left==NULL && node->right==NULL)
     {
        printf("\n");
     }
```

```
else
{
    //For left subtree path
```

see more

• Reply • Share ›

Sahil ➦ Sahil • 2 years ago

A small correction

```
void printallpaths(struct node* node)
{
     if(node==NULL){return;}

//Insert a newline character when the node is leaf

     if(node->left==NULL && node->right==NULL)
     {
        printf("%d ",node->data);
        printf("\n");
     }

     else
     {
         //For left subtree path
```

see more

• Reply • Share ›

yahoo ➦ Sahil • 2 years ago

it doesnt work for more than 2 levels.. you have to store it in a a

Reply • Share ›

jane ➔ Sahil • 2 years ago

no it wont work for mre than 2 level trees as
15
/ \
10 7
/ \ \
6 4 9

it will not print the path 15->10>4 for leaf node 4 instead it will p

Reply • Share ›

Raja • 3 years ago

will this work? or am i missing any cases?
// create a global stock "st".

```
root-to-leaf(root)
{

if (root == null ) return root;

st.push(root->data);
if( root->left == null && root->right == null ) {
print(st); st.pop();
}
else{
root-to-leaf(root->left);
root-to-leaf(root-> right);
}
}
```

Reply • Share ›

Ramakrishna ➦ Raja • a year ago

I don't think I have understood this..

For the given example, after printing 1,2,4 and 1,2,5 - the stack now co
left and right sub trees are null, we print the stack i.e 1,2 and 3 - where

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

  • Reply • Share ›

harish verma • 3 years ago

**Thanks a lot for sharing your knowledge. I am quite satisfied with the ar
and
i like very much the stacking of node->right recursive call.**

**Thanks and regards,
Harish Verma**

  • Reply • Share ›

g33k • 3 years ago

```
PrintPath( Node root, String path){
        if(root == null)
                return;
        if(root->left==NULL && root->right ==NULL){
                print path + " " + root.info;
                return;
        }

        PrintPath(root->left, path + " " + root.info);
        PrintPath(root->right, path + " " + root.info);
```

```
}
```

driver function:

```
        PrintPath(root, "");
```

Reply • Share ›

Gandalf g33k • 2 months ago

//From Leaf To Root without extra space.

boolean rootToLeaf(Node root){

if(root == null) return false;

if(!root.left && !root.right) return true;

if(rootToLeaf(root.left)) System.out.println(root..left.data);

if(rootToLeaf(root.right))System.out.println(root.right.data);

return true;

}

1 Reply • Share ›