

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

Asked by Varun Bhatia

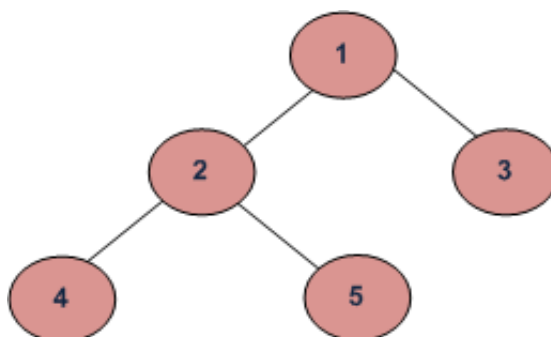
Here is the solution.

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);
}

/* 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;
}
```

[Run on IDE](#)

References:

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

116 Comments Category: [Trees](#)

Related Posts:

- [Check if a given array can represent Preorder Traversal of Binary Search Tree](#)
- [Mirror of n-ary Tree](#)
- [Succinct Encoding of Binary Tree](#)
- [Construct Binary Tree from given Parent Array representation](#)
- [Symmetric Tree \(Mirror Image of itself\)](#)
- [Find Minimum Depth of a Binary Tree](#)
- [Maximum Path Sum in a Binary Tree](#)
- [Expression Tree](#)

0

Average Difficulty : **0/5.0**
No votes yet.

(Login to Rate)

Like Share 14 people like this. Be the first of your friends.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.116 Comments **GeeksforGeeks** Login▼ Recommend 4 Share

Sort by Newest▼



Join the discussion...

**Riya** • 25 days ago

Where is stack getting formed??? Can't identify..plsss help

^ | v • Reply • Share ›

**Sa** → Riya • 7 days agono stack is formed in the above program, the node's data is stored in the array.
'pathlen' variable stores it at proper index.

^ | v • Reply • Share ›

**uchiha_itachi** • 2 months ago

here is another way, I came up with and it doesn't need any extra space or stack or queue.

i) recursively go to the leaf, print the leftest path.

ii) while returning from recursive call swap the left and right subtree.

iii) travel to the left subtree(which was initially right tree) and do the same process.

iv) after that swap the subtrees again.

1

/\

2 3

/\

4 5 go to 4 print the leftest path that is 1,2,4

from 2 swap the left and right subtree

1

/\

2 3

/\

5 4 go to 5 and print the leftest path 1,2,5

[see more](#)

1 ^ | v • Reply • Share ›

**arun kumar** • 2 months ago`int myarray [100];``int i = 0;``void rootToleafPath(BSTNode* root)``{``if(root == Null)``return ;``myarray[i] = root->data;``i++;``rootToleafPath(root->left);``rootToleafPath(root->right);``if(root->left == Null && root->right == Null)`

[see more](#)

^ | v • Reply • Share ›

**TechieByChoice** • 2 months ago

This code will not print the paths correctly. Once leaf node is found, it is printing the array. That's fine. But after that it should pop that leaf node from path, otherwise that node will be printed even if it is not in the path..

Correct modification of that is -->>>>

`bool printPathsRecur(struct node* node)``{``if (node==NULL) return false;``/* append this node to the path array */``path[pathlen] = node->val;``pathlen++;``/* it's a leaf, so print the path that led to here */`

if (node->left==NULL && node->right==NULL)

[see more](#)

^ | v • Reply • Share ›



vivek ➔ TechieByChoice • 2 months ago

nope it works fine the array in the call stack will keep on updating with each recursive call, and when it shifts from the left to right subtree, control shifts back to the state of array what it was on approach at the root node, you do not need to decrement the index;

1 ^ | v • Reply • Share ›



Vikram Prasad ➔ TechieByChoice • 2 months ago

exactly my concern too

^ | v • Reply • Share ›



kartteekrb • 3 months ago

<http://code.geeksforgeeks.org/...>

^ | v • Reply • Share ›



rangeelaladkanashelliankheen • 3 months ago

if we remove ..if (node==NULL) return; line from the function void printPathsRecur(struct node* node, int path[], int pathLen)will it work in all condition or not....pls guys and GALS answer plzzzzzzzz...thanks in advance....plss

^ | v • Reply • Share ›



Anonymous • 4 months ago

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

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

```
struct node
```

```
{
```

```
int data;
```

```
struct node* left;
```

```
struct node* right;
```

```
};
```

```
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);
```

```

    ...
}

```

[see more](#)

^ | v • Reply • Share ›



lucy • 4 months ago

using vector...

<http://code.geeksforgeeks.org/...>

^ | v • Reply • Share ›



AB_kyusak → lucy • 4 months ago

why u have passed the vector as a reference to the recursive function ?

i have done the C++ implementation using vectors also <http://ideone.com/uP9MZU>

^ | v • Reply • Share ›



lucy → AB_kyusak • 4 months ago

i think your code not work in case

```
root=CreateNewNode(1);
```

```
root->left=CreateNewNode(2);
```

```
root->left->right=CreateNewNode(3);
```

output should 1 2 3

^ | v • Reply • Share ›



radek • 5 months ago

I tried to replace the use of array with linked list..

it is only displaying the first path..

and incompletely displaying the other paths..

can anyone help on this..

<http://code.geeksforgeeks.org/...>

^ | v • Reply • Share ›



Narendra → radek • 4 months ago

I have corrected your code.

<http://code.geeksforgeeks.org/...>

mistakes you made

1) when passing local pointer which you want to modify in other function either you pass as a reference or double pointer

2) you need to delete last node in the list not the entire list

^ | v • Reply • Share ›



vivek → Narendra • 2 months ago



could you explain why deleting of the last node is necessary?

^ | v • Reply • Share ›

**dev21** • 5 months ago

If the node->data is distinct and can be treated as vertex like in graphs. Implement preorder traversal along with backtracking. Modified above code printPathsrecur() function. <https://ideone.com/HVg4oM>

^ | v • Reply • Share ›

**blank space** • 5 months ago<https://ideone.com/FKpaxy>

^ | v • Reply • Share ›

**Avanish Singh** • 5 months ago

We can use a Deque to implement this. It will take $O(n)$ time and $O(n)$ space. We keep on adding the nodes (visited during pre-order traversal). When a leaf is encountered, we will deque from the front, print it and enqueue it again at the back.

Algorithm:

1. Enqueue the root at the back of deque.
2. while(the deque is not empty)
 1. Pick the last element from deque.
 2. If it is a leaf, deque all the elements from the front, print them and enqueue them at the back in the same order. Remove the last element from queue (leaf).
 3. If it is not a leaf, enqueue its (i) left child (ii) right child at the back of deque.

Here's the java code: <http://ideone.com/3WfXUa>

^ | v • Reply • Share ›

**Shantanu** → Avanish Singh • 4 months agoagain and again enqueue makes algorithm run in $O(n^2)$

^ | v • Reply • Share ›

**vk** → Shantanu • 3 months agonope it won't take $O(n^2)$ time @Shantanu ..see review your answer

^ | v • Reply • Share ›

**Ananda kumar N** • 5 months ago

```
void printpath(struct node *r,struct stack * s){
    if(r==NULL){
        return;
    }
}
```



```

else{
push(r->data,s);
printpath(r->l,s);
printpath(r->r,s);
printstack(s);
printf("\n");
pop(s);
}
}

```

^ | v • Reply • Share ›



Deepak → Ananda kumar N • 5 months ago

```

void printpath(struct node *r,struct stack * s){
if(r==NULL){
return;
}
else{
push(r->data,s);
printpath(r->l,s);
printpath(r->r,s);
if(r->l == null && r->r == null) {
printstack(s);
printf("\n");
}
pop(s);
}
}

```

^ | v • Reply • Share ›



jitinmaher • 5 months ago

What is the significance of "pathlen" in the recursive call as one of the parameter ?
How is "pathlen" different in each stack since it has global scope!!!

^ | v • Reply • Share ›



Ajcoo • 5 months ago

Once it goes into the statement
if (node->left==NULL && node->right==NULL)

it prints the array but then how does it return from this call there is no return statement????

^ | v • Reply • Share ›



Varun Sagar → Ajcoo • 3 months ago

once it prints the array which is inside the if block it goes to the end of the function

and then goes back to the calling scope, we don't need a return statement here as it will return anyhow

^ | v • Reply • Share ›



sanjeev • 5 months ago

simple -

<http://ideone.com/21Cttx>

^ | v • Reply • Share ›



Shivani Aggarwal • 5 months ago

I have implemented the iterative algorithm, but there is some issue with the termination condition

Can somebody help with that

link : <http://ideone.com/t9dX3m>

^ | v • Reply • Share ›



Rishu Agrawal → Shivani Aggarwal • 5 months ago

have you tried debugger ?

use this - <https://dbgr.cc//cpp>

^ | v • Reply • Share ›



Shivani Aggarwal → Rishu Agrawal • 5 months ago

It's not working

^ | v • Reply • Share ›



Rishu Agrawal → Shivani Aggarwal • 5 months ago

Segmentation fault after printing all the paths.

^ | v • Reply • Share ›



Shivani Aggarwal → Rishu Agrawal • 5 months ago

I still have not been able to fix this.

^ | v • Reply • Share ›



Joey → Shivani Aggarwal • 5 months ago

corrected code...<http://ideone.com/bojG8G>

^ | v • Reply • Share ›



Shivani Aggarwal → Joey • 5 months ago

works perfectly!

Thanks

^ | v • Reply • Share ›



guest → Shivani Aggarwal • 4 months ago

Can somebody help me with my code

<http://ideone.com/EN15ex>

^ | v • Reply • Share ›



Shivani Aggarwal → guest • 4 months ago

<http://ideone.com/1wovIR>

It works now. There was a segmentation fault.

After printing all paths, it was executing `s.pop()` when the stack was empty.

^ | v • Reply • Share ›



Rishu Agrawal • 5 months ago

A good and efficient iterative algorithm :

Use a stack and do a depth first traversal of the tree. Whenever you find a leaf node, just print the stack. I'll try to explain the algorithm.

1. Start with the root. Push root node to stack.
2. while(stack not empty) // i'll take top=top_of_stack
3. mark top as visited
4. if(top.left_node exists AND not_visited)
5. push(top.left_node)
6. else if(top.right_node exists AND not_visited)
7. push(top.right_node)
8. else //Leaf node
9. print stack
10. pop

What I am doing here is, I traverse to each leaf node then print the path, then backtrack to find other leaf nodes.

Source Quora. :)

1 ^ | v • Reply • Share ›



Sai Bharath → Rishu Agrawal • 5 months ago

This is DFS right? Indeed it's another way of printing all paths

^ | v • Reply • Share ›



Rishu Agrawal → Sai Bharath • 5 months ago

yes exactly.

1 ^ | v • Reply • Share ›

**Gautham Kumaran** • 6 months ago

Java Code

<http://ideone.com/1ujk4v>

^ | v • Reply • Share ›

**Kams** → Gautham Kumaran • 3 months ago

Instead of a plain array, you can use a list.

```

void allpaths1(Node n)
{
    if (n == null)
        return;

    list.add(n.data);
    if (n.left == null & n.right == null)
    {
        for (Integer i: list)
        {
            System.out.print(i + " ");
        }
        System.out.println();
    }
    else
    {
        allpaths1(n.left);
        allpaths1(n.right);
    }
    list.remove(list.size()-1);
}

```

^ | v • Reply • Share ›

**Holden** → Kams • 2 months ago

Your code does not work ...

^ | v • Reply • Share ›

**nikhil** • 7 months agowe can also do it by modifying preorder [algorithm](#). here ar is arraylist.

```

public static void preorder(Node root){
    if(root==null){
        return;
    }
    // System.out.println(root.getData());
    // ...
}

```

```

ar.add(root);
preorder(root.getLeft());
preorder(root.getRight());
if(root.getLeft()==null && root.getRight()==null){
int i=0;
for( i=0;i<ar.size();i++){ system.out.println(ar.get(i).getdata());="" }=""
system.out.println("="" break="" ");="" }="" ar.remove(ar.size()-1);="" }="">

```

^ | v • Reply • Share ›



NITIN PANCHAL • 7 months ago

we can instead search for a leaf node and print the path
 //program to est the actual implementation of the delete function in the tree

```
#include<iostream>
```

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

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

```
#include<queue>
```

```
using namespace std ;
```

```
struct tree
```

```
{
```

```
int data ;
```

```
tree *left;
```

[see more](#)

^ | v • Reply • Share ›



Aryan Parker • 7 months ago

What is the time complexity of the above code.I think it would be $O(N^2)$, in case of a skewed tree.

^ | v • Reply • Share ›



thevagabond85 • 8 months ago

Possible duplicate <http://www.geeksforgeeks.org/g...>

^ | v • Reply • Share ›



Ashish Jaiswal • 8 months ago

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

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

```
typedef struct node
{
    int data;
    struct node*left;
    struct node*right;
}Node;
```

```
typedef struct stack
{
    int data;
    struct stack*next;
}Stack;
```

```
Node*createnode(int d)
{
    Node*newnode=(Node*)malloc(sizeof(Node));
```

[see more](#)[^](#) | [v](#) • [Reply](#) • [Share](#) ›**Guest** • 8 months ago

Stack implementation using C


```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
    int data;
    struct node*left;
    struct node*right;
}Node;
```

```
typedef struct stack
{
    int data;
    struct stack*next;
}Stack;
```


```
Node*createnode(int d)
{
```

[see more](#)[^](#) | [v](#) • [Reply](#) • [Share](#) ›**Jaguar** • 9 months ago

The path[] array is a local variable here for each stack element in the recursion. Isn't it?



^ | v • Reply • Share ›






Arai • 9 months ago

A simple recursive approach

<http://ideone.com/XEURRz>

^ | v • Reply • Share ›

Load more comments

 [Subscribe](#)  [Add Disqus to your site](#)  [Privacy](#)

[@geeksforgeeks](#), [Some rights reserved](#)[Contact Us!](#)[About Us!](#)[Advertise with us!](#)