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

Asked by Varun Bhatia

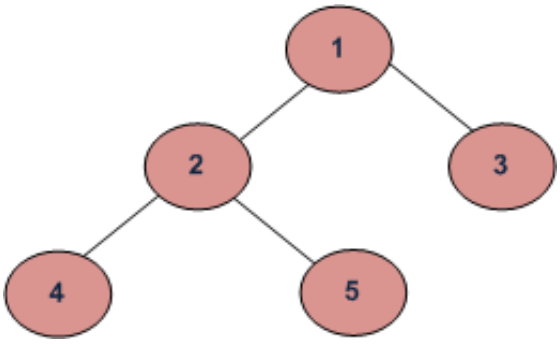
Here is the solution.

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:

C

```
/*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

Java

```
// Java program to print all root to leaf paths

// A binary tree node
```

```
class Node {

    int data;
    Node left, right;

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

class BinaryTree {

    static Node root;

    /* 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(Node node) {
        int path[] = new int[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(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);
        }
    }

    /* Utility that prints out an array on a line */
    void printArray(int ints[], int len) {
        int i;
        for (i = 0; i < len; i++) {
            System.out.print(ints[i] + " ");
        }
        System.out.println("");
    }

    /* Driver program to test mirror() */
    public static void main(String[] args) {

        BinaryTree tree = new BinaryTree();

        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);
```

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

    }
}
```

Run on IDE

References:

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

128 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 Average Difficulty : **2/5.0**
Based on **19** vote(s)

Add to TODO List

Mark as DONE

Like Share 15 people like this.

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

128 Comments GeeksforGeeks 1 Login

Recommend 4 Share Sort by Newest

Join the discussion…

Akash Gupta • 25 days ago
Why so serious?? Simple java code:

http://code.geeksforgeeks.org/...

1 • Reply • Share ›

Pk Akash Gupta • 21 days ago
Add a check for empty tree

• Reply • Share ›

Kataria Deepak Akash Gupta • 22 days ago
Good one..!!

• Reply • Share ›

Krishna • 2 months ago
What is the time complexity of this program?

• Reply • Share ›

Vaibhav Bajpai Mod Krishna • 2 months ago
The algorithm requires to visit each node for pre-order, hence time for that would be O(V+E).

Now, as and when a leaf node is found,it prints the corresponding branch of the binary tree which is stored in the array(using the printArray() function). That means the printArray() function will be called the times equal to the number of leaves in the binary tree.

If we talk about the worst case scenario, a binary tree will have the number of leaf nodes in the order of O(V/2), and the number of paths from root to leaves will be equal to the number of leaves. And whenever a leaf node is found, the corresponding branch from root to that leaf node is printed.

Now, in the worst case, the max length of the branch can be equal to the height of the binary tree which is O(log V). Hence extra time will be = no of leaf nodes *

the binary tree which is O(log V). Hence extra time will be = no of leaf nodes length of the branch = O(V/2 * log V).

Therefore, total time complexity can be written as : traversal time + print branch time
: O(V+E) + O(V/2 * log V)
: O(V+E+V logV)
: O(E+V logV)

Please correct if I am wrong.

• Reply • Share ›

Satish Kumar • 2 months ago

here is c++ solution

void print(bnode* t,int a[],int top)

{

a[top++]=t->data;

if(t->lc==NULL&&t->rc==NULL)

{

for(int i=0;i<top;i++) cout<<a[i]<<"="" ";="" cout<<endl;="" return="" ;="" }="" print(t-="">lc,a,top);

print(t->rc,a,top);

• Reply • Share ›

rockeblaze • 3 months ago

Is the time complexity O(n*j)? where j = max number of nodes in a path?

• Reply • Share ›

shish_009 • 4 months ago

A simple recursive method in java using just a string

public void print_all_path(Node head,String s){
if (head!=null)
s=s+" "+head.data;

if (head.left!=null)
print_all_path(head.left,s);

if (head.right!=null)
print_all_path(head.right,s);

```
if(head.right==null & head.left==null)
System.out.println(s);

}
public void print_all_path_header(Node head){
String s="";
print_all_path(head,s);
}
```

• Reply • Share ›

Jatin • 4 months ago

www.coder2design.com

Java version

```
public void printTreeVerticals(BinaryTree<integer> bt) {
List<node<integer>> list = new ArrayList<node<integer>>();
traverse(bt.getRoot(), list);
}

private void traverse(Node<integer> node, List<node<integer>> list ) {
if (node == null) {
return;
}
list.add(node);
traverse(node.getLeftChild(), list);
traverse(node.getRightChild(), list);
if (node.getLeftChild() == null && node.getRightChild() == null) {
for (Node<integer> i : list) {
System.out.print("" + i + ", ");
}
}
list.remove(node);
}
```

• Reply • Share ›

Riya • 5 months ago

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

• Reply • Share ›

Sa ➦ Riya • 5 months ago

no stack is formed in the above program, the node's data is stored in the array. 'pathlen' variable stores it at proper index

pathlist variable stores it at proper index.

uchiha_itachi • 6 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

Anshul Garg ➚ uchiha_itachi • 3 months ago

In root_to_path function, in line 1 i.e. the if condition

incorrect : if(node->left == NULL || node == NULL),
here node == NULL has to be checked before node->left because if node == NULL is true then node->left would give a segmentation fault.
correct: if(node == NULL || node->left == NULL)

arun kumar • 7 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

• Reply • Share ›

TechieByChoice • 7 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

1 • Reply • Share ›

vivek ➚ TechieByChoice • 6 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 • Reply • Share ›

Vikram Prasad ➚ TechieByChoice • 7 months ago

exactly my concern too

1 • Reply • Share ›

karteekrb • 7 months ago

http://code.geeksforgeeks.org/...

• Reply • Share ›

rangeelaladkanashelliankheen • 7 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 plzzzzzzz...thanks in advance....plss

• Reply • Share ›

Annonymous • 8 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

• Reply • Share ›

Gourav Goswami ➦ Annonymous • 4 months ago

gud solution

• Reply • Share ›

lucy • 9 months ago

using vector...

http://code.geeksforgeeks.org/...

• Reply • Share ›

AB_kyusak ➚ lucy • 9 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

• Reply • Share ›

lucy ➚ AB_kyusak • 8 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

• Reply • Share ›

radek • 9 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/...

• Reply • Share ›

Narendra ➚ radek • 9 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

• Reply • Share ›

vivek ➚ Narendra • 6 months ago

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

• Reply • Share ›

dev21 • 9 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

• Reply • Share ›

blank space • 9 months ago

https://ideone.com/FKpaxy

• Reply • Share ›

Avanish Singh • 9 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 enque it again at the back.

Algorithm:
1. Enque 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 enque them at the back in the same order. Remove the last element from queue (leaf).
3. If it is not a leaf, enque its (i) left child (ii) right child at the back of deque.

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

• Reply • Share ›

Shantanu ➚ Avanish Singh • 9 months ago

again and again enqueue makes algorithm run in O(n^2)

• Reply • Share ›

vk ➚ Shantanu • 7 months ago

nope it won't take O(n^2) time @Shantanu ..see review your answer

• Reply • Share ›

Ananda kumar N • 9 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);
}
}
```

• Reply • Share ›

Deepak_1 ➚ Ananda kumar N • 9 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-->s == null) {
printstack(s);
printf("\n");
}
pop(s);
}
}
```

• Reply • Share ›

jitinmaher • 9 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!!!

• Reply • Share ›

Ajcoo • 9 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????

• Reply • Share ›

Varun Sagar ➚ Ajcoo • 8 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

• Reply • Share ›

sanjeev • 10 months ago

simple -
http://ideone.com/21Ctzx

• Reply • Share ›

Shivani Aggarwal • 10 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

• Reply • Share ›

Rishu Agrawal Shivani Aggarwal • 10 months ago

have you tried debugger ?
use this - https://dbgr.cc/l/cpp

• Reply • Share ›

Shivani Aggarwal Rishu Agrawal • 10 months ago

It's not working

• Reply • Share ›

Rishu Agrawal Shivani Aggarwal • 10 months ago

Segmentation fault after printing all the paths.

• Reply • Share ›

Shivani Aggarwal Rishu Agrawal • 10 months ago

I still have not been able to fix this.

• Reply • Share ›

Joey Shivani Aggarwal • 10 months ago

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

• Reply • Share ›

Shivani Aggarwal Joey • 10 months ago

works perfectly!
Thanks

• Reply • Share ›

guest Shivani Aggarwal • 9 months ago

Can somebody help me with my code

http://ideone.com/EN15ex

• Reply • Share ›

Shivani Aggarwal guest • 9 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.

• Reply • Share ›

videet Shivani Aggarwal • 4 months ago

This code wont work if there are duplicate values

• Reply • Share ›

Rishu Agrawal • 10 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 • Reply • Share ›

Sai Bharath Rishu Agrawal • 9 months ago

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

• Reply • Share ›

Load more comments