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

Asked by Varun Bhatia

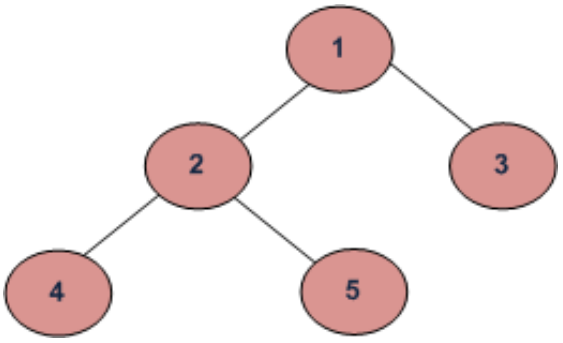
Here is the solution.

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

106 Comments Category: Trees

Related Questions:

- Expression Tree
- Construct all possible BSTs for keys 1 to N
- K'th smallest element in BST using O(1) Extra Space
- Count BST subtrees that lie in given range
- Count BST nodes that lie in a given range
- Data Structure for a single resource reservations
- How to handle duplicates in Binary Search Tree?
- Handshaking Lemma and Interesting Tree Properties

karteekrb • 10 days ago

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

rangeelaladkanashelliankheen • 10 days 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

Annonymous • a month 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 ›

lucy • 2 months ago

using vector...
http://code.geeksforgeeks.org/...

• Reply • Share ›

AB_kyusak ➚ lucy • 2 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 • a month 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 • 2 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 • a month 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 ›

dev21 • 2 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 • 2 months ago

https://ideone.com/FKpaxy

• Reply • Share ›

Avanish Singh • 2 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 • 2 months ago

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

• Reply • Share ›

Ananda kumar N • 2 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 ➚ Ananda kumar N • 2 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 • 2 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 • 2 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 • 22 days 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 • 3 months ago

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

• Reply • Share ›

Shivani Aggarwal • 3 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 • 3 months ago

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

• Reply • Share ›

Shivani Aggarwal ➚ Rishu Agrawal • 3 months ago

It's not working

• Reply • Share ›

Rishu Agrawal ➚ Shivani Aggarwal • 3 months ago

Segmentation fault after printing all the paths.

• Reply • Share ›

Shivani Aggarwal ➚ Rishu Agrawal • 3 months ago

I still have not been able to fix this.

• Reply • Share ›

Joey ➚ Shivani Aggarwal • 3 months ago

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

• Reply • Share ›

Shivani Aggarwal ➚ Joey • 3 months ago

works perfectly!
Thanks

• Reply • Share ›

guest ➚ Shivani Aggarwal • 2 months ago

Can somebody help me with my code

http://ideone.com/EN15ex

• Reply • Share ›

Shivani Aggarwal ➚ guest • 2 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 ›

Rishu Agrawal • 3 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

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. :)

• Reply • Share ›

Sai Bharath ➚ Rishu Agrawal • 2 months ago

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

• Reply • Share ›

Rishu Agrawal ➚ Sai Bharath • 2 months ago

yes exactly.

• Reply • Share ›

Gautham Kumaran • 4 months ago

Java Code
http://ideone.com/1ujk4v

• Reply • Share ›

Kams ➚ Gautham Kumaran • 15 days 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);
}
```

• Reply • Share ›

nikhil • 4 months ago

we 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);="" }="">
```

• Reply • Share ›

NITIN PANCHAL • 4 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

• Reply • Share ›

Aryan Parker • 5 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.

• Reply • Share ›

thevagabond85 • 5 months ago

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

• Reply • Share ›

Ashish Jaiswal • 6 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

• Reply • Share ›

Guest • 6 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

• Reply • Share ›

Jaguar • 6 months ago

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

• Reply • Share ›

A rai • 6 months ago

A simple recursive appproach
http://ideone.com/XEURRz

• Reply • Share ›

Vijay • 8 months ago

Won't this program give output like this.. 124.. 1245...12453??

• Reply • Share ›

SANKHYA ➚ Vijay • 8 months ago

NOPE, THE PARAMETER PATH LENGTH RECURS IN THE STACK WITH EVERY FUNCTION CALL IN THE ACTIVATION STACK. SO, A FUNCTION AT A GIVEN POINT OF TIME CAN ONLY INCREMENT THE PATH LENGTH OF THE FUNCTION PARAMETER IMMEDIATELY BELOW IN THE STACK OF ACTIVATION RECORDS.

YOU NEED TO RESEARCH IN THAT FIELD TO UNDERSTAND HOW THAT WORKS!

ITS A BIG TOPIC AND CANNOT BE COMMENTED

• Reply • Share ›

Ashish Jaiswal ➚ SANKHYA • 6 months ago

Bus yara....Bacche ki jaan lega kya :P ;)

• Reply • Share ›

prashan ➚ SANKHYA • 6 months ago

Easy man! :)

• Reply • Share ›

Guest • 8 months ago

```
void printpath(int *path,int len)
{
for(int i=0;i<len;i++) printf("%d",path[i]);="" printf("\n");="" }="">
```

• Reply • Share ›

Guest • 8 months ago

void printpath(int *path,int len)
{
for(int i = 0;i<len;i++) printf("%d",path[i]);="" printf("\n");="" }="">

• Reply • Share ›

Chirag • 8 months ago

Solution with recursion:
void printpath(int *path,int len)
{
for(int i=0;i<len;i++) {printf("%d="" ",path[i]);="" }printf("\n");}="" void="" move(struct=""
node*="" node,="" int="" index)="" {="" static="" int="" path[100];="" if(node="NULL)"
{="" return;="" }="" path[index++]="node->data;
move(node->left,index);
if(node->left==NULL && node->right==NULL)
printpath(path,index);
move(node->right,index);
}

1 • Reply • Share ›

Guest • 8 months ago

SOLUTION WITH RECURSION :

void printpath(int *path,int len)
{
for(int i=0;i<len;i++) {="" printf("%d="" ",path[i]);="" }="" printf("\n");="" }="" void="" move(struct="" node*="" node,="" int="" index)="" {="" static="" int="" path[100];="" if(node="NULL)" {return;}="" path[index++]="node->data;
move(node->left,index);
if(node->left==NULL || node->right==NULL)
printpath(path,index);
move(node->right,index);
}

Don't know why, spaces are convert into =""

• Reply • Share ›

Guest • 8 months ago

SOLUTION WITH RECURSION :

void printpath(int *path,int len)
{
for(int i=0;i<len;i++) {="" printf("%d="" ",path[i]);="" }="" printf("\n");="" }="" void="" move(struct="" node*="" node,="" int="" index)="" {="" static="" int="" path[100];=""

```
if(node="=NULL)" {return;}="" path[index++]="node-">data;
move(node->left,index);
if(node->left==NULL || node->right==NULL)
printpath(path,index);
move(node->right,index);
}
```

1 • Reply • Share ›

Abhishek • 9 months ago

JAVA solution with recursion:

```
public void printAllPaths() {
if(root == null)
return;
printAllPaths(root, new ArrayList<t>());
}

private void printAllPaths(Node<t> node, List<t> nodeList) {
nodeList.add(node.getData());
if(node.getLeft() == null && node.getRight() == null) {
print(nodeList);
}

if(node.getLeft() != null) {
printAllPaths(node.getLeft(), nodeList);
nodeList.remove(nodeList.size()-1);
}
if(node.getRight() != null) {
```

see more

3 • Reply • Share ›

Load more comments