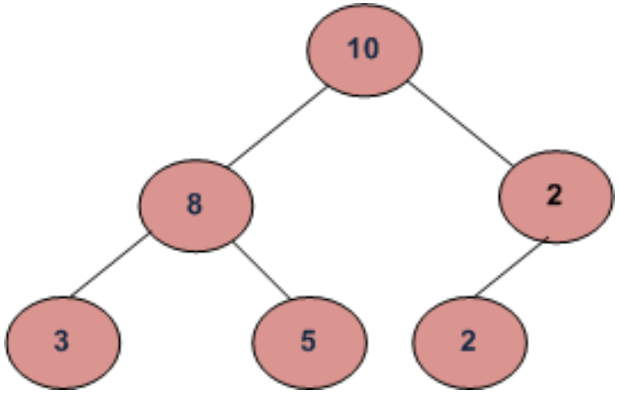
Given a binary tree, print all root-to-leaf paths

For the below example tree, all root-to-leaf paths are:
10 –> 8 –> 3
10 –> 8 –> 5
10 –> 2 –> 2



Algorithm:
Use a path array path[] to store current root to leaf path. Traverse from root to all leaves in top-down fashion. While traversing, store data of all nodes in current path in array path[]. When we reach a leaf node, print the path array.

```
#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;
};

/* Prototypes for funtions needed in printPaths() */
void printPathsRecur(struct node* node, int path[], int pathLen);
void printArray(int ints[], int len);

/*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);
  }
}


/* UTILITY FUNCTIONS */
/* Utility that prints out an array on a line. */
void printArray(int ints[], int len)
{
  int i;
```


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

```
  for (i=0; i<len; i++)
  {
    printf("%d ", ints[i]);
  }
  printf("\n");
}

/* utility 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);
}

/* Driver program to test above functions*/
int main()
{

  /* Constructed binary tree is
            10
          /   \
        8      2
      /  \    /
    3     5  2
  */
  struct node *root = newnode(10);
  root->left        = newnode(8);
  root->right       = newnode(2);
  root->left->left  = newnode(3);
  root->left->right = newnode(5);
  root->right->left = newnode(2);

  printPaths(root);

  getchar();
  return 0;
}
```

Time Complexity: O(n)

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

Please write comments if you find any bug in above codes/algorithms, or find other ways to solve the same problem.

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)
- 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

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...
Backtracking | Set 7 (Sudoku) · 43 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

Neha I think that is what it should return as, in...
Find depth of the deepest odd level leaf node · 3 hours ago

- 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.

40 Comments GeeksforGeeks

Sort by Newest

Mahda • 5 days ago

Hi geeks, what must I do if I want to add the averages from each path?

Reply • Share ›

prashant jha • 2 months ago

use recursion and vector for path here is my code
http://ideone.com/idlpnx

Reply • Share ›

Sabitaa Bhabhi • 2 months ago

why we are not using dynamic array

```
void printPaths(struct node* node)
{
int path[]={0};
printPathsRecur(node, path, 0);
}
```

Reply • Share ›

Babajj • 2 months ago

Just use a stack having a print function.

-Push current node is not null
-Print stack if its a leaf node
-Recurse left
-Recurse right
-Pop stack

Reply • Share ›

shravan • 7 months ago

Each of the path[] array may contain O(h) elements, where h is the height of tl
so process these elements only the algorithm takes O(n*h) time.
how we can say the algorithm runs in O(n)?
:-(

Reply • Share ›

Jonathan shravan • 25 days ago

Each node is traversed only once throughout the tree.

O(n*h) would occur only when nodes traversed multiple times.

Reply • Share ›

Vivek • 7 months ago

hi geeksforgeeks .. why waste so much space?
we know that the length of any path wont be greater than the height of the tree
so we can initialize the array "path" with length = height of tree instead of any p
here's the optimized solution below

1 Reply • Share ›

Vivek • 7 months ago

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

```
#include <stdlib.h>

struct node

{

        int data;

        struct node *left;

        struct node *right;

};

void root_leaf(struct node *root, int *arr, int e)
```

see more

1 ▲ | ▼ • Reply • Share ›

kumar • 7 months ago

Same question has been asked on FB's interview, with constraint that use iter

▲ | ▼ • Reply • Share ›

Rahul Mahale • 9 months ago

void PrintArr(int *arr,int len)
{
static int pathNo=0;
int i;

printf("\nPath %d",++pathNo);

for(i=0;i<len;i++) {="" printf("="" %d="" ",arr[i]);="" }="" return;="" }="" void="" pr

```
pathLen++;

if(root->left==NULL && root->right==NULL)
{
PrintArr(pathArr,pathLen);
return;
}
else
{
PrintR2LPaths(root->left,pathArr,pathLen);
PrintR2LPaths(root->right,pathArr,pathLen);
}
}
```


Rahul Mahale • 9 months ago

```
void PrintArr(int *arr,int len)
{
        static int pathNo=0;
        int i;

        printf("\nPath %d",++pathNo);

        for(i=0;i<len;i++)
        {
                printf(" %d ",arr[i]);
        }

        return;
}
```

```
void PrintR2LPaths(BST *root,int pathArr[],int pathLen)
{
```

see more

1 ^ | v • Reply • Share ›

shek8034 • 11 months ago

Simple and working code

```
 /* void printPaths(node* root, int len)
{
  static int path[1000];
  if(root == NULL)
        return;
  path[len++] = root->data;
  if (root->left==NULL && root->right==NULL)
  {
    int i;
        for(i=0;i<len;i++)
                printf("%d ",path[i]);
        printf("\n");
  }
  else
  {
    /* otherwise try both subtrees */
    printPaths(root->left,len);
    printPaths(root->right,len);
  }
} */
```

^ | v • Reply • Share ›

Nikhil Agrawal • 11 months ago

Iterative version for printing all possible paths from root to leafs:

```
 public static void printAllPathToLeafNonRecursive(Node root)
{
    if (root == null)
    {
        return;
    }

    Queue<Object> q = new LinkedList<Object>();
    q.add(root);
    q.add(root.value + "");

    while(!q.isEmpty()){

        Node head = (Node) q.remove();
        String headPath = (String) q.remove();
```

see more

▲ | ▼ • Reply • Share ›

pavansrinivas ➦ Nikhil Agrawal • 7 months ago

nyc soln...

▲ | ▼ • Reply • Share ›

yo-gi • 2 years ago

void printAllPaths(struct node* root, int *arr, int len)
{
if(root == NULL)
return;

```
arr[len++] = root->data;

if((root->left == NULL) && (root->right==NULL))
{
int i=0;
for(i=0; i left, arr, len);
printAllPaths(root->right, arr, len);
}
```

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


hemant • 2 years ago

Correct me if I am wrong.

Assume a BST and insert 10,5,18,15,12
Now consider it to be a BT and run the logic given in the original post.
I guess it will give a crash because it will try to access the right of node with da
Hence a slight modification required.

Guys, please share your comments

```
int Roofleaf[6]= {0};// instead of 6 we can take height of the tree

void print_array(int size)
{
int i;
if(size == -1)
return;

for(i=0; idata;
```

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

see more

• Reply • Share ›

hemant ➚ hemant • 2 years ago

Not again.....I wonder why i always end up pasting the half code thoug

anyways my point was we should add conditions like

if(root->left)
printf_root_to_leaf(root->left,i+1)

if(root->right)
printf_root_to_leaf(root->right,i+1)

• Reply • Share ›

GeeksforGeeks ➚ hemant • 2 years ago

@hemant: Please paste your code between sourcecode tags.
code and keep the sourcecode tags.

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

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

• Reply • Share ›

hemant ➚ GeeksforGeeks • 2 years ago

```
 void print_array(int size)
{
    int i;
        for(i=0; i<=size; i++)
     printf("%d ",Roofleaf[i]);
```

```
     printf("\n");
}

void printf_root_to_leaf(struct node* root, int i
{
   Roofleaf[i]= root->data;

    if((root->left == NULL) && (root->right == NU
    {
      print_array(i);
      return;
    }
```

see more

• Reply • Share ›

hemant ➚ hemant • 2 years ago

I wonder why i always end up pasting the half code though i paste it co

anyways my point was we should add conditions like

if(root->left)
printf_root_to_leaf(root->left,i+1)

if(root->right)
printf_root_to_leaf(root->right,i+1)

• Reply • Share ›

hemant • 2 years ago

Assume a BST and insert 10,5,18,15,12
Now consider it to be a BT and run the logic given in the original post.
I guess it will give a crash because it will try to access the right of node with da
Hence a slight modification required.

Guys, please share your comments

int Roofleaf[6]= {0};// instead of 6 we can take height of the tree

void print_array(int size)
{
int i;
if(size == -1)
return;

for(i=0; idata;

if((root->left == NULL) && (root->right == NULL))
{

see more

• Reply • Share ›

krishna • 2 years ago

Instead of fixing the path size to const, its better if we take height of the tree as

• Reply • Share ›

FireFox • 2 years ago

```
void printAllpaths(Node root, int[] a, int i){
       If(node == null)
              return;
      A[i] = node.data;
      printAllpaths(root.left, a,i+1);
      printAllpaths(root.right,a,i+1);
      printTheArray(a,i);
}
```

• Reply • Share ›

krishna FireFox • 2 years ago

This logic wont work, because it also prints all the inter mediate paths.

• Reply • Share ›

munna krishna • 2 years ago

just find all the leaves of the tree, and for each leaf backtrack to

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

• Reply • Share ›

neeraj singh • 3 years ago

[sourcecode language="java"]public static void printAllRoot2LeafPaths(Node n
if(null == n){
return;
}
arr[index] = n.value;

//is leaf node
if (null == n.left && n.right == null) {
for(int i : arr){
if(i==0){
break;
}
System.out.print(i + ",");
}
System.out.println();
return;
}

printAllRoot2LeafPaths(n.left, arr, index + 1);
printAllRoot2LeafPaths(n.right, arr, index + 1);

}

• Reply • Share ›

Ehsan Ahmadi • 3 years ago

this is my concept
this code show **all** path in the tree
for example :
/* Constructed binary tree is

```
          10
        /
      8      2
    /      /
  3     5  2
*/
```

output is
[10,8],[10,8,3],[10,8,5],[10,2],[10,2,2]

```
 #include &lt;iostream&gt;
#include
#include
```

see more

• Reply • Share ›

Shwetha • 3 years ago

Hi...can anyone help me with a program to implement insertion in a binary tree
Binary search tree. For BST I know to do...but want the logic on how to keep ir

Kindly help

• Reply • Share ›

Sandeep ➚ Shwetha • 3 years ago

@Shwetha: Could you provide an input array and binary tree that you w
array?

• Reply • Share ›

Shwetha ➚ Sandeep • 3 years ago

@Sandeep: Say i have an array
int arr[]={50,20,13,45,67,55,23}
I am expecting a balanced binary tree like dis:

```
      50
     /  \
    20   13
   / \   / \
  45 67 55 23
```

For BST we check for whether the value is more or less and de
Here just keep on putting one after another based on whichevel
Would appreciate help regarding this

• Reply • Share ›

Rehan ➚ Shwetha • 2 years ago

@Shwetha: You could use the following trick.
If a node is an index 'i' then its children can be '2*i+1' an

• Reply • Share ›

Punam • 3 years ago

Hi,

I have a binary tree that looks like this

```
      50
     /  \
    20  60
   / \  / \
 10  30 55  80
          /  \
         70   100
        /  \    \
       65  75   120
      /
     62
```

Somehow when m trying the above logic for this tree, its failing.
It prints till 62 and then fails.
Not sure what is going wrong.
Looking forward for help of any kind.

 • Reply • Share ›

Sandeep Punam • 3 years ago

@Punam: Please post your code here.

 • Reply • Share ›

Jameel • 4 years ago

[sourcecode language="java"]
public void printPath(Node node, String str){

if(node == null){
return;
}

```
System.out.println(str +" - "+ node.data);
return;
}

str += " - "+ node.data;
printPath(node.left, str);
printPath(node.right, str);
}
```

• Reply • Share ›

ghadeer ➦ Jameel • 2 years ago

thanks for this program

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

• Reply • Share ›

manoj gupta • 4 years ago

Better and Simpler approach is

```
 void printPaths(struct node* node, int len)
{
  static int path[1000];
  int pathlen=len;

  if(NULL == node)
        return;

  path[pathlen] = node->data;



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

```
  {
    printArray(path, pathlen);
  }
  else
  {
    /* otherwise try both subtrees */
    printPaths(node->left,pathlen +1 );
    printPaths(node->right,pathlen + 1);
  }

}
```


Gauri • 4 years ago

```
 void printPathshimanshu(struct node* node, int len)
{
  static int path[1000];
  int pathlen=len;

  if(NULL == node)
        return;

  path[pathlen] = node->data;
  pathlen++;

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

```
    /* otherwise try both subtrees */
    printPathshimanshu(node->left,pathlen);
    printPathshimanshu(node->right,pathlen);
  }

}
```

This could solve the problem.

Himanshu Aggarwal • 4 years ago

Hi,

We can simplify the above function. There is no need to use a second-level re

This can be done by declaring the two variables path and pathlen as 'static' ins

The function would then become as:

```
 void printPaths(struct node* node)
{
  static int path[1000];
  static int pathlen=0;

  if(NULL == node) return;
  path[pathlen] = node->data;
  pathlen++;

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

Sandeep ➚ Himanshu Aggarwal • 4 years ago

If we make pathlen a static variable then pathlen always gets incremen
example.

```
          10
         /   \
       8      2
     /  \    /
   3     5  2
```

The output of your code for above tree is:

10 8 3
10 8 3 5
10 8 3 5 2 2

But, the desired output is:

10 8 3
10 8 5
10 2 2

 • Reply • Share ›

Himanshu Aggarwal ➚ Sandeep • 4 years ago

@Sandeep, Thanks for pointing out the mistake. I have correct

Please find the corrected version below.

```
 void printPaths(struct node* node)
{
  static int path[1000];
  static int pathlen=0;
```

```
if(NULL == node) return;
path[pathlen] = node->data;
pathlen++;

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

see more

Reply • Share ›

Subscribe Add Disqus to your site