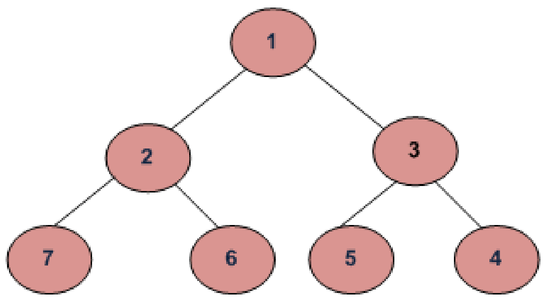
Level order traversal in spiral form

Write a function to print spiral order traversal of a tree. For below tree, function should print 1, 2, 3, 4, 5, 6, 7.



Method 1 (Recursive)

This problem can bee seen as an extension of the level order traversal post.

To print the nodes in spiral order, nodes at different levels should be printed in alternating order. An additional Boolean variable *ltr* is used to change printing order of levels. If *ltr* is 1 then printGivenLevel() prints nodes from left to right else from right to left. Value of *ltr* is flipped in each iteration to change the order.

Function to print level order traversal of tree

```
printSpiral(tree)
  bool ltr = 0;
  for d = 1 to height(tree)
     printGivenLevel(tree, d, ltr);
     ltr ~= ltr /*flip ltr*/
```

Function to print all nodes at a given level

```
printGivenLevel(tree, level, ltr)
if tree is NULL then return;
if level is 1, then
    print(tree->data);
else if level greater than 1, then
    if(ltr)
        printGivenLevel(tree->left, level-1, ltr);
        printGivenLevel(tree->right, level-1, ltr);
    else
        printGivenLevel(tree->right, level-1, ltr);
        printGivenLevel(tree->left, level-1, ltr);
```

Following is C implementation of above algorithm.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.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;
};

/* Function protoypes */
void printGivenLevel(struct node* root, int level, int ltr);
int height(struct node* node);
struct node* newNode(int data);

/* Function to print spiral traversal of a tree*/
void printSpiral(struct node* root)
{
    int h = height(root);
    int i;

    /*ltr -> Left to Right. If this variable is set,
      then the given level is traverseed from left to right. */
    bool ltr = false;
    for(i=1; i<=h; i++)
    {
        printGivenLevel(root, i, ltr);

        /*Revert ltr to traverse next level in oppposite order*/
        ltr = !ltr;
    }
}

/* Print nodes at a given level */
void printGivenLevel(struct node* root, int level, int ltr)
{
    if(root == NULL)
        return;
    if(level == 1)
        printf("%d ", root->data);
    else if (level > 1)
    {
        if(ltr)
        {
            printGivenLevel(root->left, level-1, ltr);
            printGivenLevel(root->right, level-1, ltr);
        }
        else
        {
            printGivenLevel(root->right, level-1, ltr);
            printGivenLevel(root->left, level-1, ltr);
        }
    }
}

/* Compute the "height" of a tree -- the number of
    nodes along the longest path from the root node
    down to the farthest leaf node.*/
int height(struct node* node)
{
    if (node==NULL)
        return 0;
    else
    {
        /* compute the height of each subtree */
        int lheight = height(node->left);
        int rheight = height(node->right);

        /* use the larger one */
        if (lheight > rheight)
            return(lheight+1);
        else return(rheight+1);
    }
}

/* 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);
}

/* Driver program to test above functions*/
int main()
{
    struct node *root = newNode(1);
    root->left        = newNode(2);
    root->right       = newNode(3);
    root->left->left  = newNode(7);
    root->left->right = newNode(6);
    root->right->left  = newNode(5);
    root->right->right = newNode(4);
    printf("Spiral Order traversal of binary tree is \n");
    printSpiral(root);

    return 0;
}
```

Run on IDE

Output:

```
Spiral Order traversal of binary tree is
1 2 3 4 5 6 7
```

Time Complexity: Worst case time complexity of the above method is **O(n^2)**. Worst case occurs in case of skewed trees.

Method 2 (Iterative)

We can print spiral order traversal in **O(n) time** and O(n) extra space. The idea is to use two stacks. We can use one stack for printing from left to right and other stack for printing from right to left. In every iteration, we have nodes of one level in one of the stacks. We print the nodes, and push nodes of next level in other stack.

```
// C++ implementation of a O(n) time method for spiral order traversal
#include <iostream>
#include <stack>
using namespace std;

// Binary Tree node
struct node
{
    int data;
    struct node *left, *right;
};

void printSpiral(struct node *root)
{
    if (root == NULL)  return;   // NULL check

    // Create two stacks to store alternate levels
    stack<struct node*> s1;  // For levels to be printed from right to left
    stack<struct node*> s2;  // For levels to be printed from left to right

    // Push first level to first stack 's1'
    s1.push(root);

    // Keep ptinting while any of the stacks has some nodes
    while (!s1.empty() || !s2.empty())
    {
        // Print nodes of current level from s1 and push nodes of
        // next level to s2
        while (!s1.empty())
        {
            struct node *temp = s1.top();
            s1.pop();
            cout << temp->data << " ";

            // Note that is right is pushed before left
            if (temp->right)
                s2.push(temp->right);
            if (temp->left)
                s2.push(temp->left);
        }

        // Print nodes of current level from s2 and push nodes of
        // next level to s1
        while (!s2.empty())
        {
            struct node *temp = s2.top();
            s2.pop();
            cout << temp->data << " ";

            // Note that is left is pushed before right
            if (temp->left)
                s1.push(temp->left);
            if (temp->right)
                s1.push(temp->right);
        }
    }
}

// A utility functiont to create a new node
```

```
struct node* newNode(int data)
{
    struct node* node = new struct node;
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return(node);
}

int main()
{
    struct node *root = newNode(1);
    root->left        = newNode(2);
    root->right       = newNode(3);
    root->left->left  = newNode(7);
    root->left->right = newNode(6);
    root->right->left  = newNode(5);
    root->right->right = newNode(4);
    cout << "Spiral Order traversal of binary tree is \n";
    printSpiral(root);

    return 0;
}
```

Run on IDE

Output:

```
Spiral Order traversal of binary tree is
1 2 3 4 5 6 7
```

Please write comments if you find any bug in the above program/algorithm; or if you want to share more information about spiral traversal.

169 Comments Category: Queue Trees

Related Posts:

- Minimum time required to rot all oranges
- How to efficiently implement k Queues in a single array?
- An Interesting Method to Generate Binary Numbers from 1 to n
- Iterative Method to find Height of Binary Tree
- Construct Complete Binary Tree from its Linked List Representation
- Find the first circular tour that visits all petrol pumps
- Implement Stack using Queues
- Find the largest multiple of 3

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

(Login to Rate)

Like Share 13 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.

169 Comments **GeeksforGeeks** 1 **Login**

Recommend 3 **Share** **Sort by Newest**

Join the discussion…

d_geeks • a month ago
For n-ary tree, iterative code

http://ideone.com/QWCq0Q
• Reply • Share ›

Harshit Gupta • a month ago
In the iterative solution, there are nested while loops. But the complexity is still O(n). Can anyone point out which fact I'm missing?
1 • Reply • Share ›

Gaurav Arora Harshit Gupta • a month ago
That every node will be processed exactly once.
• Reply • Share ›

Jayesh • 2 months ago
Java Implementation

http://javabypatel.blogspot.in...
• Reply • Share ›

Victor • 2 months ago
```
private static void spiralOrderTraversal(Node root) {
System.out.println("Spiral order traversal");
if(root == null)
return;
Stack<node> stack1 = new Stack<node>();
Stack<node> stack2 = new Stack<node>();
boolean levelOrderflag = true;
```

```
stack1.push(root);
while(!stack1.isEmpty()) {
Node temp = stack1.pop();
System.out.print(temp.data + "\t");
if(levelOrderflag) {
if(temp.right != null)
stack2.push(temp.right);
if(temp.left != null)
stack2.push(temp.left);
} else {
```

see more

• Reply • Share ›

Gaurav Arora • 3 months ago

I was asked in my interview to modify the given tree so that it now has the nodes in ZigZag order

For eg.
1
2 3
4 5 6 7

should be converted to
1
3 2
4 5 6 7

Not to mention
you cant change the data in the nodes,just the pointers.

• Reply • Share ›

Gaurav Ambast Gaurav Arora • a month ago

```
package zigzag;

import java.util.*;

public class Zigzag {

public void generateZigzag(Node root){

if(root == null)

return;
```

```
Stack st1 = new Stack();

Stack st2 = new Stack();

Queue q = new LinkedList();

swapInternal(root);

if(root.getLeft() != null){
```

see more

• Reply • Share ›

Mandeep Beniwal Gaurav Arora • 2 months ago
How about changing the child pointers of each required level?
For eg. save 3's left and right children as temp1 and temp2..
Then make 4 and 5 as 3's left and right children... Then make temp1 and temp2 as 2's left and right children..
Finally, switch 1's left and right pointers in a similar way..
Correct me if I'm wrong.

• Reply • Share ›

Gaurav Arora Mandeep Beniwal • 2 months ago
Yeah, that's quite what we have to do.

But coming up with the exact algorithm is the main task here.
How do you intend to traverse down to node 4 and 5 to make them the sub trees of node 3?

I was required to implement an O(n) solution.
Tell me if you want to have a look at what I did :)

1 • Reply • Share ›

Mandeep Beniwal Gaurav Arora • 2 months ago
Yes plz :)

• Reply • Share ›

Gaurav Arora Mandeep Beniwal • 2 months ago
http://ideone.com/5GvXLc

This was the document we created during the interview so I wrote down in words what I intended to do and then gave a function that implements the idea

To check out the entire running program see this

http://ideone.com/0CL0KU

1 • Reply • Share ›

Andro • 3 months ago

Solution using only one queue for level order traversal:

/*

* spiral_level_order.cpp

*

* Created on: 02-Aug-2015

* Author: Andro

*/

#include "stdio.h"

#include <queue>

using namespace std;

#ifndef NULL

see more

• Reply • Share ›

Shubham Gupta • 4 months ago

Implemented using BFS with queue and stack for printing right to left :)

Time Complexity : O(n)
Space Complexity : O(n)

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

• Reply • Share ›

Akshay Monga Shubham Gupta • 2 months ago

how s that O(n) ? its O(n square)

• Reply • Share ›

Prajwal Muralidhara Akshay Monga • 2 months ago

You are visiting each and every node only once although you have 2 nested while loops. Hence its O(n) and not O(n^2)

• Reply • Share ›

Sathiyaseelan • 4 months ago

What about BFS with queue ? Will do the same in O(n) right? Correct me if I'm wrong

1 • Reply • Share ›

Shubham Gupta Sathiyaseelan • 4 months ago

Implemented using BFS with queue and stack for printing right to left :)

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

• Reply • Share ›

vikky_codder Sathiyaseelan • 4 months ago

That will do simple level order traversal, not in spiral form.

• Reply • Share ›

Shashi Jey • 4 months ago

https://ideone.com/SJPDuH

• Reply • Share ›

sasha • 4 months ago

Can u explain why stack is defined as
Stack< struct node* > s1;

• Reply • Share ›

Arun Mittal sasha • 4 months ago

because stack store element having type (node) means it store the address of node .we push the addresss of node in stack rather than value of node

• Reply • Share ›

Gangadhar • 4 months ago

```
void printSpiral(struct node* root)

{

int h = height(root);

int i;

/*ltr -> Left to Right. If this variable is set,

then the given level is traverseed from left to right. */

for(i=1; i<=h; i++)

{

if(i%2==0)

printLeftToRightGivenLevel(root, i);
```

else

see more

• Reply • Share ›

Naveen Kumar Badgujar • 4 months ago

elegant c++ code using a stack and a queue: i have used c++ stl for stack and queue

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

1 • Reply • Share ›

arpit • 4 months ago

can we use deque and then similarly perform operation like once on left right second time right left
??

• Reply • Share ›

Dipankar Bhardwaj • 4 months ago

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

• Reply • Share ›

Prince Bharti • 5 months ago

java solution using a queue and stack.

explanation: traverse level by level (using queue) and for each even level just push the nodes into stack and at the end of level print from stack (it would be in reverse) and at each odd level simply print from left to right.

class sol{

void spiral(node root){

if(root==null)

return;

Queue<node> q=new LinkedList<node>();

Stack s=new Stack();

boolean switchlevel=false;

see more

• Reply • Share ›

Jordan Sarraf • 7 months ago

Using Stacks and Queues (Java Implementation):

```
class Tree {
// Spiral Level Order Traversal
void printSpiral() {
Queue<node> q = new LinkedList<node>();
Queue<node> tempQ = new LinkedList<node>();
Stack<node> tempS = new Stack<node>();
boolean forward = false;

//--------
q.add(node);
q.add(null);
while(!q.isEmpty()) {
Node temp = q.remove();
if(temp != null) {
if(forward == true) {
// Adds in Queue
```

see more

• Reply • Share ›

NITIN PANCHAL • 7 months ago

We can use the following algorithm also
Take two stacks /push the root on first stack
pop the element . print it and right first and then left on another stack ,
now from second stack pop the first element and print it and push its left and right in the first stack .keep on doing this process till both the stacks donot becm eempty
Thanks

1 • Reply • Share ›

Aryan Parker • 7 months ago

Using a stack and a queue-
http://ideone.com/Im0PIQ
Only two lines of the method two are modified

• Reply • Share ›

arpit ➦ Aryan Parker • 5 months ago

its not working for more than 3 levels

• Reply • Share ›

Saurabh ➦ Aryan Parker • 6 months ago

I don't think this solution will work when there are more than 3 levels. Please check

• Reply • Share ›

Vineeth Reddy • 7 months ago

We can either use one stack, or one stack and one queue. Store first level in stack and the second one in queue. And continue the pattern for the coming levels.

• Reply • Share ›

arpit Vineeth Reddy • 5 months ago

it wont work likewise i guess do u have a working code ??

• Reply • Share ›

Anonymous • 8 months ago

We can do it using only one stack. We push left first and right second at even level and right first and left second at odd level.

1 • Reply • Share ›

Ashish Jaiswal • 8 months ago

http://codepad.org/V9MbOMyk

• 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 list
{
Node*data;
struct list*next;
}List;

typedef struct stack
{
List*top;
```

see more

• Reply • Share ›

Jun • 8 months ago

http://ideone.com/wUFnHc

• Reply • Share ›

Techie Me • 8 months ago

Thank you for the wonderful explanation... Here is the Java version of the code and some detailed analysis of running time. http://techieme.in/spiral-trav...

• Reply • Share ›

neelabhsingh • a year ago

http://ideone.com/WjvSAu

• Reply • Share ›

Naveen Khatri • a year ago

http://ideone.com/UjTzjd

• Reply • Share ›

GeeksforGeeks Mod • a year ago

@all, thanks for pointing out the typo in recursive function. We have corrected it.

1 • Reply • Share ›

Guest GeeksforGeeks • 9 months ago

Don't think that boolean flag is not at all required . It's redundant. We can extract same info using odd/even test on level itself.

```
void printGivenLevelSpiral2(struct node* root, int level){

if(root != NULL){

if(level == 1){

printf(" %d ",root->data);

} else if(level > 1){

if(level%2){

printGivenLevelSpiral2(root->left, level-1);

printGivenLevelSpiral2(root->right, level-1);

}else {
```

see more

2 • Reply • Share ›

Aditya Goel • a year ago

@GeeksforGeeks

Pls note the typo. In algorithm(recursive) it is written if(rtl). It should be if(ltr)

1 ^ | ˅ • Reply • Share ›

Guest • a year ago

GeeksforGeeks

Pls note the typo. In algorithm(recursive) it is written
if(rtl). It should be if(ltr)

^ | ˅ • Reply • Share ›

Demon • a year ago

In the recursive algorithm (logic part):
2nd function printGivenLevel(tree, level, ltr)
in line no 5: if(rtl) it should be if(ltr)

1 ^ | ˅ • Reply • Share ›

vipinkaushal • a year ago

solution using only one deque

http://ideone.com/r6ly5b

2 ^ | ˅ • Reply • Share ›

The Internet ➦ vipinkaushal • 9 months ago

Hey, vipinkaushal! Thank you for sharing your solution. I like your general approach of using a single queue and saving the number of nodes on each level for an inner loop that then traverses just one level. However, I think one can write it a little shorter, without losing readability, so I have forked your solution here: http://ideone.com/eW9M08. Take a look and maybe you like it better. I was also pondering if I should use member function pointers to choose between push_back/push_front and pop_front/pop_back as appropriate, but it turned out looking pretty arcane, so I abandoned it.

2 ^ | ˅ • Reply • Share ›

surbhijain93 ➦ The Internet • 7 months ago

Great code!

^ | ˅ • Reply • Share ›

The Internet ➦ The Internet • 9 months ago

Oh, you are also getting runtime errors dereferencing null pointers. I fixed that.

^ | ˅ • Reply • Share ›

Pranav Kumar Jha • a year ago

We can use a stack and a queue to push nodes alternatively in them
Here's an implementation :

http://ideone.com/uQSLQR

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

@geeksforgeeks, Some rights reserved Contact Us! About Us! Advertise with us!