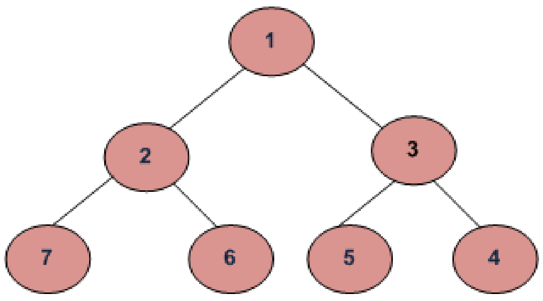
Level order traversal in spiral form

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.

```c
// C program for recursive level order traversal in spiral form
#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 opposite 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

Java

```
// Java program for recursive level order traversal in spiral form

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
class Node
{
    int data;
    Node left, right;

    public Node(int d)
    {
        data = d;
        left = right = null;
```

```java
    }
}

class BinaryTree
{
    Node root;

    // Function to print the spiral traversal of tree
    void printSpiral(Node node)
    {
        int h = height(node);
        int i;

        /* ltr -> left to right. If this variable is set then the
           given label is transversed from left to right */
        boolean ltr = false;
        for (i = 1; i <= h; i++)
        {
            printGivenLevel(node, i, ltr);

            /*Revert ltr to traverse next level in opposite order*/
            ltr = !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(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);
        }
    }

    /* Print nodes at a given level */
    void printGivenLevel(Node node, int level, boolean ltr)
    {
        if (node == null)
            return;
        if (level == 1)
            System.out.print(node.data + " ");
        else if (level > 1)
        {
            if (ltr != false)
            {
                printGivenLevel(node.left, level - 1, ltr);
                printGivenLevel(node.right, level - 1, ltr);
            }
            else
            {
                printGivenLevel(node.right, level - 1, ltr);
                printGivenLevel(node.left, level - 1, ltr);
            }
        }
    }
    /* Driver program to test the above functions */
    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(7);
        tree.root.left.right = new Node(6);
        tree.root.right.left = new Node(5);
        tree.root.right.right = new Node(4);
        System.out.println("Spiral order traversal of Binary Tree is ");
        tree.printSpiral(tree.root);
    }
}

// This code has been contributed by Mayank Jaiswal(mayank_24)
```

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 function 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

Java

```java
// Java implementation of an O(n) approach of level order
// traversal in spiral form

import java.util.*;

// A Binary Tree node
class Node
{
    int data;
    Node left, right;

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

class BinaryTree
{

    static Node root;

    void printSpiral(Node node)
    {
        if (node == null)
            return;   // NULL check

        // Create two stacks to store alternate levels
        Stack<Node> s1 = new Stack<Node>();// For levels to be printed from righ
        Stack<Node> s2 = new Stack<Node>();// For levels to be printed from left

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

        // 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())
            {
                Node temp = s1.peek();
                s1.pop();
                System.out.print(temp.data + " ");

                // Note that is right is pushed before left
                if (temp.right != null)
                    s2.push(temp.right);

                if (temp.left != null)
                    s2.push(temp.left);

            }

            // Print nodes of current level from s2 and push nodes of
            // next level to s1
            while (!s2.empty())
            {
                Node temp = s2.peek();
                s2.pop();
                System.out.print(temp.data + " ");

                // Note that is left is pushed before right
                if (temp.left != null)
                    s1.push(temp.left);
                if (temp.right != null)
                    s1.push(temp.right);
            }
        }
    }

    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(7);
        tree.root.left.right = new Node(6);
        tree.root.right.left = new Node(5);
        tree.root.right.right = new Node(4);
        System.out.println("Spiral Order traversal of Binary Tree is ");
        tree.printSpiral(root);
    }
}

// This code has been contributed by Mayank Jaiswal(mayank_24)
```

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.

207 Comments Category: Queue Trees

Related Posts:

- Implement a stack using single queue
- 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

(Login to Rate and Mark)

2.7 Average Difficulty : **2.7/5.0**
Based on **59** vote(s)

🟩 Add to TODO List

🟩 Mark as DONE

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

207 Comments GeeksforGeeks 1 Login

Recommend 4 Share Sort by Newest

Join the discussion…

wilson · 3 days ago

using one deque we can solve this

Reply · Share ›

Apar Madan · 4 days ago

Solution using queue and stack (BFS approach):
http://ideone.com/OKEp8O

Reply · Share ›

Aomine · 11 days ago

https://ideone.com/pP2xdx
Modification of level order traversal easy to learn simple solution :)

Reply · Share ›

Rakesh · 13 days ago

I tried it with one queue and one stack(Code written in C# and tried to make Node generic where T)

```
public void SpiralLevelTraversal(Node<t> root)
{
Queue<node<t>> queue = new Queue<node<t>>();
Stack<node<t>> stack = new Stack<node<t>>();
stack.Push(root);
SpiralLevelRecursion(queue, stack);
}

private void SpiralLevelRecursion(Queue<node<t>> queue, Stack<node<t>> stack)
{
if (queue.Count == 0 && stack.Count == 0) return;
Console.WriteLine();

while (queue.Count > 0)
{
Node<t> node = queue.Dequeue();
```

see more

Reply · Share ›

Musarrat_123 · 15 days ago

Iterative code (according to Narsimha Karumanchi book)

```
void spiralOrder(Node *root)
{
        if(!root) return;

        stack<node*> curr,next;  //curr = current level, next = next level
        bool ltr = false;       // ltr = left to right

        curr.push(root);

        while(curr.size() != 0)
        {
                root = curr.top();
                curr.pop();
                cout<<root->data<<" ";

                if(ltr)
```

see more

^ | ˅ · Reply · Share ›

Ritvik Raj · a month ago

C++soln using dequeue
http://ideone.com/47kh0g

1 ^ | ˅ · Reply · Share ›

Abhilash · 2 months ago

Anyone know how to do this prob with single datastructure?

^ | ˅ · Reply · Share ›

Ritvik Raj → Abhilash · a month ago

using dequeue:
http://ideone.com/47kh0g

1 ^ | ˅ · Reply · Share ›

learner · 2 months ago

http://code.geeksforgeeks.org/...
please check this answer is coming different

^ | ˅ · Reply · Share ›

Duvvuri Ram Kiran · 2 months ago

Tried with
out using stacks

public int getTreeHeight(TreeNode ptr) {

if (ptr != null) {

int leftHeight = getTreeHeight(ptr.getLeftChild());

int rightHeight = getTreeHeight(ptr.getRightChild());

return Math.max(leftHeight, rightHeight) + 1;

} else

return 0;

}

public void printGivenLevel(TreeNode ptr, int level) {

see more

Reply · Share ›

Mohammad Rafi · 2 months ago

Python solution using recursion: https://ideone.com/z1fJ3O

Reply · Share ›

m_asif · 4 months ago

Java O(N), using List: http://code.geeksforgeeks.org/...

Reply · Share ›

.NetGeek · 4 months ago

C# Implementation (Method - 1): http://ideone.com/O5ikdf

1 · Reply · Share ›

.NetGeek · 4 months ago

@GeeksForGeeks: Instead of below statement in Method - 1:

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

We can rely on the value of "i" to pass true or false using ternary operator as below:

i % 2 ? true : false i.e. printGivenLevel(root, i, i % 2 ? true : false);

Reply · Share ›

yesh_02 · 4 months ago

https://ideone.com/u5Cx2E
C++ solution with a stack and a queue.

Reply · Share ›

Rashmi Mishra · 5 months ago
C++ solution using deque. The idea is to use delimeter (NULL) to find out the change in levels.
http://code.geeksforgeeks.org/...
Reply · Share ›

Solazy · 6 months ago
http://ideone.com/byjntr
C++ sol using 2 stacks,easy,concise
Reply · Share ›

Gagandeep Wadhwa ➦ Solazy · 5 months ago
@Solazy what if we keep only the node data rather than storing the whole node?
it will be more memory efficient
Reply · Share ›

Siya ➦ Gagandeep Wadhwa · 5 months ago
From data you cant reach its left and right child. For that we need whole node.
Reply · Share ›

Solazy ➦ Gagandeep Wadhwa · 5 months ago
I am storing pointer to node, not d node, n data or pointer both are of 4 bytes..
Reply · Share ›

Anshul ➦ Solazy · 2 months ago
if root is NULL , you have to add this if(root == NULL) return;
Reply · Share ›

Solazy ➦ Anshul · 2 months ago
Thnks for pointing that out
..
Reply · Share ›

Vardaan Sangar · 6 months ago
http://code.geeksforgeeks.org/...

the zigzagorder function using one stack only ie O(n) space O(n)Time
Refer link for above function
Reply · Share ›

Mahesh Chikkanna · 6 months ago
In method2 is s2.empty() need not check in first while loop right ? since the last while loop will empty it all the time?

Reply • Share ›

Marimuthu Mahalingam • 6 months ago

It can be done using two stacks.

private void zigzag(Node node){

Stack<node> stack_1=new Stack<>();

Stack<node> stack_2=new Stack<>();

stack_1.push(node);

while(!stack_1.isEmpty()||!stack_2.isEmpty()){

Node temp;

while(!stack_1.isEmpty()){

temp=stack_1.pop();

System.out.print(temp+" ");

if(temp.right!=null)

see more

Reply • Share ›

abc def • 7 months ago

This can also be done using 2 queues...

2 Reply • Share ›

arusha goyal • 7 months ago

We can implement using queue and stack . Please let me know if something is wrong in this

void level_spiral(struct node *root)
{
struct queue *q ;
struct stack *s;
int count ;
int ltr = 0;
struct node *current, *temp;
current = root;
enqueue(q, current);
while (!empty(q))
{
count = q->size;
if(count ==0)
break;

```
while (count >0)
```

see more

Reply · Share ›

Jatin · 8 months ago

```
class NodeWrapper<t> {
private Node<t> node;
private int level;
public NodeWrapper(Node<t> node, int level) {
super();
this.node = node;
this.level = level;
}
public String toString() {
return node.toString();
}
}
```

Reply · Share ›

Jatin · 8 months ago

www.coder2design.com

Java version using Queue and Stack..

```
---------------------------------------------------
public void levelSpriralOrderTraversal(BinaryTree<integer> bt1) {
List<nodewrapper<integer>> queue = new ArrayList<nodewrapper<integer>>();
queue.add(new NodeWrapper(bt1.getRoot(), 0));
Stack<nodewrapper<integer>> stack = new Stack<nodewrapper<integer>>();
while (!queue.isEmpty()) {
NodeWrapper<integer> top = queue.remove(0);
if (top.level % 2 == 0 && stack.size() != 0) {
System.out.println("Stack: "+stack.pop().node);
} else {
System.out.println("Queue: " + top.node);
}
NodeWrapper<integer> lNode = new NodeWrapper<integer>(
top.node.getLeftChild(), top.level + 1);
NodeWrapper<integer> rNode = new NodeWrapper<integer>(
```

see more

Reply · Share ›

Subhadip Samanta · 8 months ago

Here is my C code for level order traversal
http://code.geeksforgeeks.org/...

Reply · Share ›

d_geeks · 9 months ago

For n-ary tree, iterative code

http://ideone.com/QWCq0Q

Reply · Share ›

Goku d_geeks · 7 months ago

Thanks for the code. Much appreciated :)

Reply · Share ›

Harshit Gupta · 10 months 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 · 9 months ago

That every node will be processed exactly once.

1 · Reply · Share ›

Jayesh · a year ago

Java Implementation

http://javabypatel.blogspot.in...

Reply · Share ›

Victor · a year 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 {
if(temp.left != null)
```

see more

Reply • Share ›

Gaurav Arora • a year 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 ›

Anand Nahar → Gaurav Arora • 5 months ago

Here can we use "find mirror of given tree" logic...but instead of swapping nodes at all levels, swap the nodes only when the level is even?

Reply • Share ›

Anand Nahar → Anand Nahar • 5 months ago

Never mind...this won't work...got the problem in it.

Reply • Share ›

Vardaan Sangar → Gaurav Arora • 6 months ago

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

soln code to your question.

Reply • Share ›

Vardaan Sangar → Gaurav Arora • 6 months ago

what will u do if even 4,5,6,7 also hve children

Reply • Share ›

Gaurav Arora → Vardaan Sangar • 5 months ago

you have to change the sequence of nodes in alternate level.
1
2 3
4 5 6 7
/\ /

9 10 11

transforms to
1
3 2
4 5 6 7
.... \ /\
..... 11 10 9

Reply • Share ›

Gaurav Ambast Gaurav Arora • 9 months 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 • 10 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 • 10 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 • 10 months ago

Yes plz :)

^ | ˅ • Reply • Share ›

Gaurav Arora ➦ Mandeep Beniwal • 10 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 ›

Vardaan Sangar ➦ Gaurav Arora • 6 months ago

@Gaurav Arora
what u will do if 4,5,6,7 also have childrens?

^ | ˅ • Reply • Share ›

Andro • a year 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 • a year 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 ›