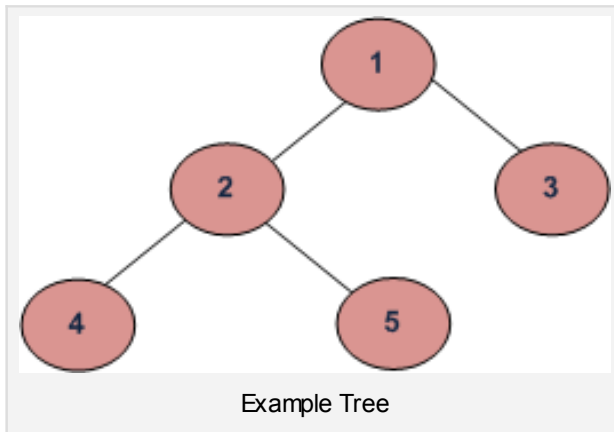


## Reverse Level Order Traversal

We have discussed [level order traversal](#) of a post in previous post. The idea is to print last level first, then second last level, and so on. Like Level order traversal, every level is printed from left to right.



Reverse Level order traversal of the above tree is “4 5 2 3 1”.

Both methods for [normal level order traversal](#) can be easily modified to do reverse level order traversal.

### METHOD 1 (Recursive function to print a given level)

We can easily modify the method 1 of the normal [level order traversal](#). In method 1, we have a method printGivenLevel() which prints a given level number. The only thing we need to change is, instead of calling printGivenLevel() from first level to last level, we call it from last level to first level.

```
// A recursive C program to print REVERSE level order traversal
```



```

#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left and right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

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

/* Function to print REVERSE level order traversal a tree*/
void reverseLevelOrder(struct node* root)
{
    int h = height(root);
    int i;
    for (i=h; i>=1; i--) //THE ONLY LINE DIFFERENT FROM NORMAL LEVEL ORDER
        printGivenLevel(root, i);
}

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

/* 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

```



## 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

```

{
    /* 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(4);
    root->left->right = newNode(5);

    printf("Level Order traversal of binary tree is \n");
    reverseLevelOrder(root);

    return 0;
}

```

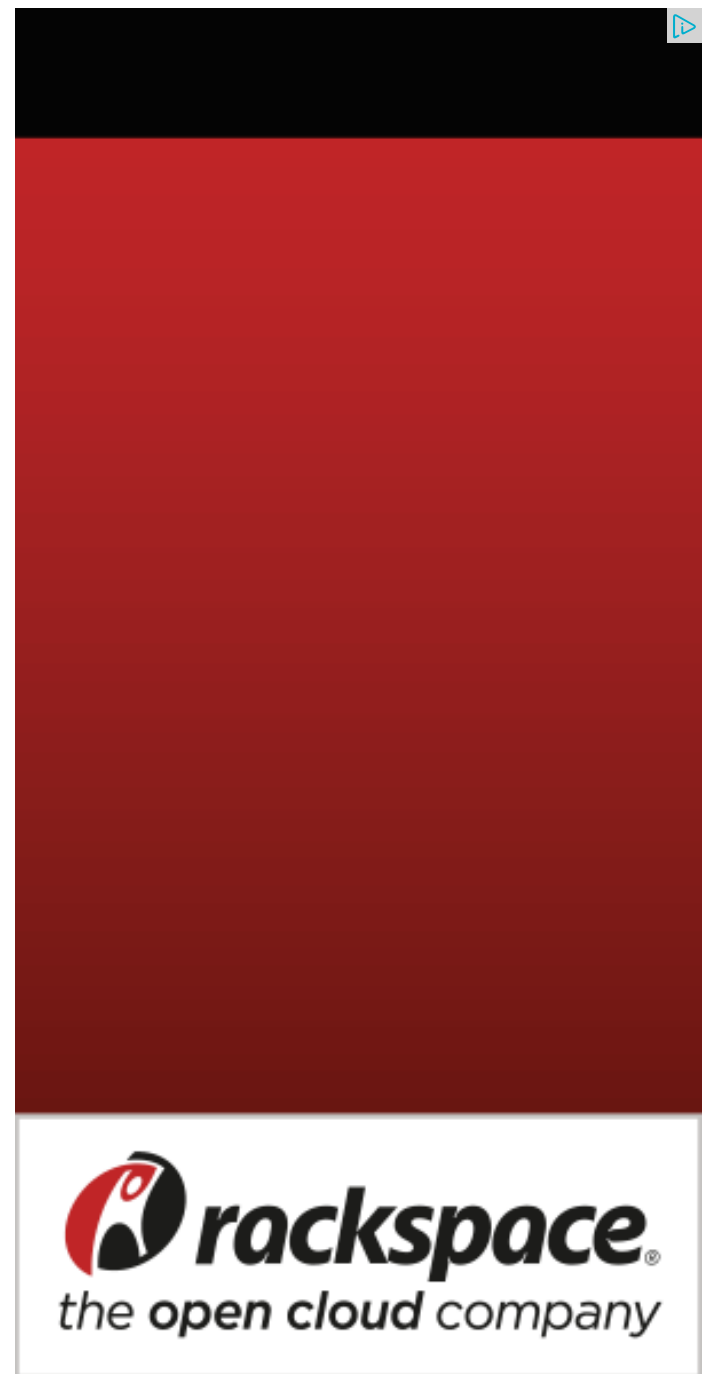
Output:

```

Level Order traversal of binary tree is
4 5 2 3 1

```

**Time Complexity:** The worst case time complexity of this method is  $O(n^2)$ . For a skewed tree,



printGivenLevel() takes  $O(n)$  time where  $n$  is the number of nodes in the skewed tree. So time complexity of printLevelOrder() is  $O(n) + O(n-1) + O(n-2) + \dots + O(1)$  which is  $O(n^2)$ .

## METHOD 2 (Using Queue and Stack)

The method 2 of [normal level order traversal](#) can also be easily modified to print level order traversal in reverse order. The idea is to use a stack to get the reverse level order. If we do normal level order traversal and instead of printing a node, push the node to a stack and then print contents of stack, we get "5 4 3 2 1" for above example tree, but output should be "4 5 2 3 1". So to get the correct sequence (left to right at every level), we process children of a node in reverse order, we first push the right subtree to stack, then left subtree.

```
// A C++ program to print REVERSE level order traversal using stack and queue
// This approach is adopted from following link
// http://tech-queries.blogspot.in/2008/12/level-order-tree-traversal-
#include <iostream>
#include <stack>
#include <queue>
using namespace std;

/* A binary tree node has data, pointer to left and right children */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Given a binary tree, print its nodes in reverse level order */
void reverseLevelOrder(struct node* root)
{
    stack<node*> S;
    queue<node*> Q;
    Q.push(root);

    // Do something like normal level order traversal order. Following
    // differences with normal level order traversal
    // 1) Instead of printing a node, we push the node to stack
    // 2) Right subtree is visited before left subtree
    while (Q.empty() == false)
    {
        /* Dequeue node and make it root */
        root = Q.front();
        Q.pop();

        // Push right child first, then left child
        if (root->right) S.push(root->right);
        if (root->left) S.push(root->left);

        // Print the node
        cout << root->data << " ";
    }
}
```

## Recent Comments

affizerv Your example has two 4s on row 3, that's why it...

[Backtracking | Set 7 \(Sudoku\)](#) · 32 minutes ago

**RVM** Can someone please elaborate this Qs from above...

[Flipkart Interview | Set 6](#) · 52 minutes ago

**Vishal Gupta** I talked about as an Interviewer in general,...

[Software Engineering Lab, Samsung Interview | Set 2](#) · 52 minutes 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 · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

AdChoices 

[► Binary Tree](#)

[► Java Tree](#)

[► Java to C++](#)

[► C++ Reverse](#)[► Java Reverse](#)[► AI Level](#)[► Tree Root](#)[► Level 1 Data](#)[► Level II](#)

```
S.push(root);

/* Enqueue right child */
if (root->right)
    Q.push(root->right); // NOTE: RIGHT CHILD IS ENQUEUED BEFORE LEFT CHILD

/* Enqueue left child */
if (root->left)
    Q.push(root->left);
}

// Now pop all items from stack one by one and print them
while (S.empty() == false)
{
    root = S.top();
    cout << root->data << " ";
    S.pop();
}
}
```

```
/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
```

```
node* newNode(int data)
```

```
{
    node* temp = new node;
    temp->data = data;
    temp->left = NULL;
    temp->right = NULL;
```

```
    return (temp);
}
```

```
/* 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(4);
    root->left->right = newNode(5);
    root->right->left = newNode(6);
    root->right->right = newNode(7);
```

```
    cout << "Level Order traversal of binary tree is \n";
    reverseLevelOrder(root);
```

```
    return 0;
```

```
}
```

*Output:*

Level Order traversal of binary tree is

4 5 6 7 2 3 1

*Time Complexity:*  $O(n)$  where  $n$  is number of nodes in the binary tree.

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



Better Than Hadoop.

HPCC Systems is Big Data Processing and Analytics  
Open Source. Proven. Trusted.

 LexisNexis® [Learn More](#) 

## 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](#)

- Check if a given Binary Tree is height balanced like a Red-Black Tree



44



5



1

Writing code in comment? Please use [ideone.com](https://ideone.com) and share the link here.

**19 Comments****GeeksforGeeks**

Sort by Newest ▼



Join the discussion...

**sourabh** • a month ago

Hi,

I am trying to do a reverse level order in this way :

```
void btree::reverseLevelOrder( node * leaf)
```

```
{
```

```
if(leaf !=NULL)
```

```
{
```

```
std::stack<node *=""> S;
```

```
S.push(leaf);
```

```
while(!S.empty())
```

```
{
```

```
node * current = S.top();
```

[see more](#)

^ | v • Reply • Share ›



**pasha shaik** • 2 months ago

this is most elegant way to print reverse level order

```
void function reverseLevelOrder(root)
{
    if(root->left!=null)
        enqueue(root->left)
    if(root->right!=null)
        enqueue(root->right)
    while(queue is notEmpty){
        reverseLevelOrder(dequeue());
    }
    print root->data;
}
```

^ | v • Reply • Share ›



**Anish Thomas** • 2 months ago

In the second approach, we can use a De-queue instead of both stack and qu

^ | v • Reply • Share ›



**newgeek** • 4 months ago

This is just a postorder traversal of the tree i think.

^ | v • Reply • Share ›



**underflow** → newgeek • 2 months ago

No, it isn't. Inadequate example. If the tree is [A [B C D] [E [F G]]],  
the output expected is, C D F G B E A. The post-order would be, C D E

^ | v • Reply • Share ›





**underflow** → underflow • 2 months ago

Typo: tree should be [A [B C D] [E F G]]

^ | v • Reply • Share ›



**Sriharsha g.r.v** • 6 months ago

```
void reverseLevelOrder(struct node* root)
{
    if(root)
    {
        reverseLevelOrder(root->left);

        reverseLevelOrder(root->right);
        printf("%d ",root->data);
    }
}
```

cant we use this...it works!!!!!!!

1 ^ | v • Reply • Share ›



**Nishant Kumar** • 8 months ago

We can also achieve same by using one array.

First we will insert right child then left->child and when we are done with insert

```
// n is total no of node in tree
void printReversLevelOrder(tree* root,int n){
    int index = 0;
    int count = 0;
    tree** array = (tree**)calloc(n,sizeof(tree*));
    array[0] = root;
    count++;
    tree* curr = NULL;
    while(index<n){ curr="array[index++];" if(curr-="">right)
        array[count++]=curr->right;
```

```

        if(curr->left)
            array[count++]=curr->left;
    }

    while(index>0){
        curr = array[--index];
        printf("%d ",curr->data);
    }
    free(array);
}

```

^ | v • Reply • Share ›



**Nikhil Agrawal** • 11 months ago

Method 1 is awesome because one just need to reverse the for loop of level or for Reverse level order traversal.

^ | v • Reply • Share ›



**Anonymous** • 11 months ago

Reverse Level Order traversal using 2 stacks:

```

public void ReverselevelOrderQueue(Node root)
{
    if(root==null)
        return;
    Stack s=new Stack();
    Stack s2=new Stack();

    s.add(root);
    s2.add(root);

    while(!s.isEmpty())

```

```

{
    Node tt=(Node) s.pop();

    if(tt.left!=null)
    {

```

[see more](#)

^ | v • Reply • Share ›



**priyanka** → Anonymous • 9 months ago

will it work for this tree

```
root=newnode(8);
```

```
root->left=newnode(5);
```

```
root->right=newnode(3);
```

```
root->left->left=newnode(2);
```

```
root->left->right=newnode(8);
```

```
root->right->right=newnode(7)
```

reply me please i m not able to find answer for this

^ | v • Reply • Share ›



**abhishek08aug** • 11 months ago

Intelligent :D

^ | v • Reply • Share ›



**Biren Barodia** • a year ago

cant we use a double ended Q and dequeue from the other end after we exam

^ | v • Reply • Share ›



**rahul** • a year ago

A modified version of the question:

print reverse level order with each level separated by a newline.

^ | v • Reply • Snare ›



**Manoj** • a year ago

it will not work for input

1

2 3

4 5 6 7

8 9 10

^ | v • Reply • Share ›



**Manoj** → Manoj • a year ago

It works fine :P

Didn't read question properly

^ | v • Reply • Share ›



**Aditya** • a year ago

[sourcecode language="Python"]

```
def ReverseLevalOrder(self,node):
```

```
q = Queue.Queue()
```

```
stack = []
```

```
q.put(node)
```

```
stack.append(node)
```

```
while q:
```

```
try:
```

```
temp = q.get_nowait()
```

```
except Exception:
```

```
break
```

```
if temp.left is not None:
```

```
q.put(temp.left)
```

```
stack.append(temp.left)
```

```
if temp.right is not None:
```

```
q.put(temp.right)
```

```
q.put(temp.right)
stack.append(temp.right)
```

```
while stack:
    print(stack.pop().key)
```

^ | v • Reply • Share ›



**aashishh** • a year ago

It can also be done using Doubly Queue. Push the right child before the left child. For any node to add to queue, start moving from tail to head printing the node's data.

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

^ | v • Reply • Share ›



**aashishh** → aashishh • a year ago

Code using a doubly linked list

```
typedef struct node
{
    int data;
    struct node* left;
    struct node* right;
}node;

typedef struct queue_node
{
    node* n;
    struct queue_node* prev;
    struct queue_node* next;
```

```
}queue_node;
```

```
class queue
```

[see more](#)

^ | v • Reply • Share ›



Subscribe



Add Disqus to your site

@geeksforgeeks, **Some rights reserved**

[Contact Us!](#)

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team