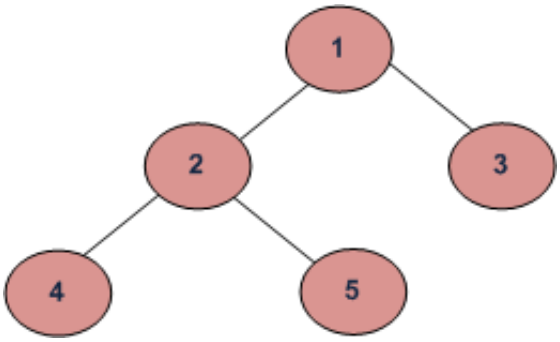
Level Order Tree Traversal

Level order traversal of a tree is breadth first traversal for the tree.



Example Tree

Level order traversal of the above tree is 1 2 3 4 5

METHOD 1 (Use function to print a given level)

Algorithm:

There are basically two functions in this method. One is to print all nodes at a given level (printGivenLevel), and other is to print level order traversal of the tree (printLevelorder). printLevelorder makes use of printGivenLevel to print nodes at all levels one by one starting from root.

```
/*Function to print level order traversal of tree*/
printLevelorder(tree)
for d = 1 to height(tree)
   printGivenLevel(tree, d);

/*Function to print all nodes at a given level*/
printGivenLevel(tree, level)
if tree is NULL then return;
if level is 1, then
    print(tree->data);
else if level greater than 1, then
    printGivenLevel(tree->left, level-1);
    printGivenLevel(tree->right, level-1);
```

Implementation:

```c
#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;
};

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

/* Function to print level order traversal a tree*/
void printLevelOrder(struct node* root)
{
  int h = height(root);
  int i;
  for(i=1; i<=h; i++)
    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
   {
     /* 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");
  printLevelOrder(root);

  getchar();
  return 0;
}
```

Run on IDE

Time Complexity: O(n^2) in worst case. 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) + .. + O(1) which is O(n^2).

METHOD 2 (Use Queue)

Algorithm:

For each node, first the node is visited and then it's child nodes are put in a FIFO queue.

```
printLevelorder(tree)
1) Create an empty queue q
2) temp_node = root /*start from root*/
3) Loop while temp_node is not NULL
    a) print temp_node->data.
    b) Enqueue temp_node's children (first left then right children) to q
    c) Dequeue a node from q and assign it's value to temp_node
```

Implementation:

Here is a simple implementation of the above algorithm. Queue is implemented using an array with maximum size of 500. We can implement queue as linked list also.

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

```
#define MAX_Q_SIZE 500

/* 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;
};

/* frunction prototypes */
struct node** createQueue(int *, int *);
void enQueue(struct node **, int *, struct node *);
struct node *deQueue(struct node **, int *);

/* Given a binary tree, print its nodes in level order
   using array for implementing queue */
void printLevelOrder(struct node* root)
{
  int rear, front;
  struct node **queue = createQueue(&front, &rear);
  struct node *temp_node = root;

  while(temp_node)
  {
    printf("%d ", temp_node->data);

    /*Enqueue left child */
    if(temp_node->left)
      enQueue(queue, &rear, temp_node->left);

    /*Enqueue right child */
    if(temp_node->right)
      enQueue(queue, &rear, temp_node->right);

    /*Dequeue node and make it temp_node*/
    temp_node = deQueue(queue, &front);
  }
}

/*UTILITY FUNCTIONS*/
struct node** createQueue(int *front, int *rear)
{
  struct node **queue =
   (struct node **)malloc(sizeof(struct node*)*MAX_Q_SIZE);

  *front = *rear = 0;
  return queue;
}

void enQueue(struct node **queue, int *rear, struct node *new_node)
{
  queue[*rear] = new_node;
  (*rear)++;
}

struct node *deQueue(struct node **queue, int *front)
{
  (*front)++;
  return queue[*front - 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");
  printLevelOrder(root);

  getchar();
  return 0;
}
```

Run on IDE

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

References:

http://en.wikipedia.org/wiki/Breadth-first_traversal

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

184 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 15 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.

184 Comments GeeksforGeeks 1 Login

Recommend 4 Share Sort by Newest

Join the discussion…

shiv • 3 days ago
please explain how height() funtion works ?
• Reply • Share ›

superman • 13 days ago
@GeeksforGeeks @Mr. Lazy What will be the best way to print the node values level wise in second method where we are using queue?
• Reply • Share ›

KS • a month ago
won't a simple BFS do the trick?
• Reply • Share ›

Darshan Washimkar KS • 5 days ago
That's what the queue method is.
• Reply • Share ›

Rahul K Kaushik • a month ago
Can anyone tell me where is this going wrong ?

```
void LevelOrder(node * root)
{
static int x = 0;
if(x == 0)
cout<<root->data<<" ";
x = 1;
if(root ==NULL)
return;
```

```
if(root->left !=NULL)
cout<<root->left->data<<" ";
if(root->right !=NULL)
cout<<root->right->data<<" ";
LevelOrder(root->left);
LevelOrder(root->right);
}
```

• Reply • Share ›

Debasis Untouchable ➚ Rahul K Kaushik • a month ago

You are just trying to print the children of a node here, not all in the level. Try to understand the algorithm.

• Reply • Share ›

amit • a month ago

can you provide level order traversal of generic [n-ary] trees

1 • Reply • Share ›

Jeff ➚ amit • a month ago

Just do the Breadth First Traversal

1 • Reply • Share ›

Jayesh • 2 months ago

Java Implementation

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

• Reply • Share ›

Dishant Maheshwari • 2 months ago

solution using queue library in c++
http://ideone.com/aaSXhY

2 • Reply • Share ›

nikhil ➚ Dishant Maheshwari • 2 months ago

Nice solution Dishant..

• Reply • Share ›

polo • 3 months ago

```
void TraverseByLayerIterative()
{
Node* cur = root;
std::deque<node*> q;

while (cur != nullptr)
```

```
{
std::cout << cur->data << " -> ";
if (cur->left != nullptr) q.push_back(cur->left);
if (cur->right != nullptr) q.push_back(cur->right);
if (q.empty()) break;
cur = q.front();
q.pop_front();
}
}
```

• Reply • Share ›

Sumit Singh Deode • 3 months ago

Can anybody explain me role of double pointer

```
struct node **queue = createQueue(&front, &rear);
```

• Reply • Share ›

centaursg • 3 months ago

Why the pointer indirection is needed here node** ?

```
struct node** createQueue(int *front, int *rear)
{
```

• Reply • Share ›

nish • 3 months ago

simple iterative implementation using queue in c++.
http://ideone.com/9CMjlb

• Reply • Share ›

Nikhil Shaw • 4 months ago

implementation using queue in simpler way ...

```
//level order traversing
#include<stdio.h>
#include<stdlib.h>

struct bNode
{
int data;
struct bNode *left;
struct bNode *right;
};

struct qNode
{
```

```
struct bNode* ptr;
struct qNode* next;
};
```

see more

• Reply • Share ›

NIKHIL SINGH • 4 months ago

simpler implementation using queue as linklist.....

http://ideone.com/Wc6gbp

• Reply • Share ›

mak • 4 months ago

i think without using o(n) space we can do it using one extra * next pointer in structure like

```
#include <iostream>

#include<stdlib.h>

using namespace std;

struct node{

int data;

struct node *left;

struct node *right;

struct node *next;

};

struct node *newnode(int d){
```

see more

• Reply • Share ›

blank space • 5 months ago

https://ideone.com/o2YxeO

• Reply • Share ›

hmm • 5 months ago

Can we do it in O(N) time and O(1) space by Connect nodes at same level using constant extra space

• Reply • Share ›

Rohan Dhaka ➚ hmm • 4 months ago

Then it won't be a tree. It will become graph.

• Reply • Share ›

Aditya Verma ➚ hmm • 5 months ago

actually that will add up to O(n) extra space.. so no..

1 • Reply • Share ›

Avata

This comment was deleted.

Mr. Lazy ➚ Guest • 5 months ago

It will fail as the size of tree increases ...Wrong Idea ...see your code here
http://ideone.com/5Wouw6
It gives output 4 2 5 1 3 6 7 8 9 10 11 12 13 14 1

However the correct output is 4 2 5 1 3 10 11 6 7 8 9 12 13 14 15

1 • Reply • Share ›

Pankaj Kushwaha ➚ Mr. Lazy • 5 months ago

you are correct , algo is not working for big input, thanks for pointing , I should have tested it with long input than post here, thanks for pointing the mistake...

1 • Reply • Share ›

Mr. Lazy ➚ Pankaj Kushwaha • 5 months ago

Happy to help :)

• Reply • Share ›

shiv ➚ Mr. Lazy • 3 days ago

can you please explain height function in above code ?

• Reply • Share ›

Mr. Lazy ➚ shiv • 2 days ago

It simply returns the height of the tree. If you don't understand it's working than go through this article
http://www.geeksforgeeks.org/w...

• Reply • Share ›

rohan • 5 months ago

There must be a check in deQueue function to return NULL, when the queue becomes empty(so as to terminate the while loop).
The above code luckily works as malloc is returning a 0 initialised block! But this might not

The above code luckily works as malloc is returning a 0 initialised block! But this might not be the case everytime!!

read

http://stackoverflow.com/quest...

1 • Reply • Share ›

Prince Bharti • 5 months ago

java solution

```
void levelorder_traversal(node root){
if(root==null){
return;
}
Queue<node> q=new LinkedList<node>();

while(true){
System.out.print(" "+root.data);
if(root.left!=null)
q.add(root.left);
if(root.right!=null)
q.add(root.right);
if(q.isEmpty()){
break;
}
root=q.remove();
}

}
```

• Reply • Share ›

pk • 5 months ago

```
void printLevelOrder(struct node* root)

{

node* temp = root;
deque<node*> d;

while (temp){
cout << temp->data;

if (temp->left != NULL)
d.push_back(temp->left);

if (temp->right != NULL)
```

```
d.push_back(temp->right);

if (d.empty())
return;

temp = d.front();
d.pop_front();
}
}
```

• Reply • Share ›

piyush32 • 5 months ago

```
void _levelorder(struct treeNode* root)

{

if (root==NULL ) return;

if (root->left!=NULL ) printf(" %d ", root->left->data);

if (root->right!=NULL ) printf(" %d ", root->right->data);

_levelorder(root->left);

_levelorder(root->right);

}

void levelorder(struct treeNode* root)

{

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

_levelorder(root);

}
```

• Reply • Share ›

Arpit Kashyap ➚ piyush32 • 5 months ago

it is wrong ...bcz it will do deep left first..This code will only work it left and right sub tree are on same level

1 • Reply • Share ›

Pankaj Kushwaha ➚ piyush32 • 5 months ago

I also though same...but for big input this will not work...

• Reply • Share ›

Pankaj Boola • 6 months ago

Optimisation in method 2:
At a particular time, there will be at max 5 nodes in queue(tricky), so we can use circular queue of 5 nodes instead of using static queue.

• Reply • Share ›

Siya ➚ Pankaj Boola • 5 months ago

what if the size of tree is more? Think of a complete tree with 10 levels you will get your answer.

• Reply • Share ›

Pankaj Boola ➚ Siya • 5 months ago

Yes, I got it. Thanks :)

• Reply • Share ›

Kapil Dalal • 6 months ago

IN METHOD 1 (Use function to print a given level)

if we RETURN after printing node's data when level is 1...
then we can reduce our time to half...
Because of not going unnecessarily until we find a null node.

• Reply • Share ›

Krishana ➚ Kapil Dalal • 6 months ago

I think we can't do in this way. May be I'm wrong so post your code.

• Reply • Share ›

Kapil Dalal ➚ Krishana • 6 months ago

https://ideone.com/fork/AsQof6

i think its giving right result,but if u find smthng odd abt dis..
thn plz rply..

• Reply • Share ›

Krishana ➚ Kapil Dalal • 6 months ago

This is okk bro. I get this in diffrent way. But this is optimization but u can't say this will reduce the time to half.

• Reply • Share ›

Kapil Dalal ➚ Krishana • 6 months ago

yes u r right..

• Reply • Share ›

Danish Dot Java • 7 months ago

simple Java code for level order traversal

```
public void levelOrderTraversal(Node n)
{
ArrayList<node> list = new ArrayList<>();
list.add(n);
while(true)
{
if(n.left!=null)
list.add(n.left);

if(n.right!=null)
list.add(n.right);

System.out.println(n.data);
list.remove(0);

if(list.isEmpty()==false)
n = list.get(0);
else
break;
}
}
```

1 ▲ | ▼ • Reply • Share ›

Holden ➦ Danish Dot Java • 6 months ago

why you used ArrayList? why you didn't use queue?

▲ | ▼ • Reply • Share ›

thevagabond85 • 7 months ago

I have tried to use STL for method 2 but it's giving Run Time Error:
Point the error????

```
void levelOrderUsingQueue(Node root)
{

Node temp;
queue<node> q;
if(!root) return;
q.push(root);
while(!q.empty() )
{
temp = q.front();
q.pop();
cout<<temp->data<<" ";
```

```
if(temp->left);
q.push(temp->left);
if(temp->right)
q.push(temp->right);
}

}
```

• Reply • Share ›

Krishana ➚ thevagabond85 • 6 months ago

Remove the semicolon(;) after
if(temp->left)

1 • Reply • Share ›

thevagabond85 ➚ Krishana • 6 months ago

ty

• Reply • Share ›

Mission Peace • 7 months ago

Variation of above qs
https://www.youtube.com/watch?...

• Reply • Share ›

ankita • 7 months ago

For method 1, What would be the average complexity or best case complexity?

• Reply • Share ›

surbhijain93 • 7 months ago

using queues in c++
https://ideone.com/DSDOeq

• Reply • Share ›

Load more comments