GeeksforGeeks

A computer science portal for geeks
Register | Login

- Home
- Links
- Q&A
- Interview Corner
- Ask a question

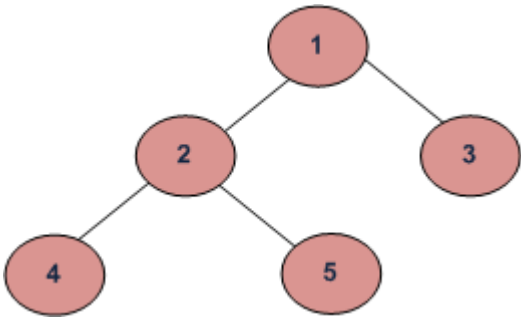
- Feedback
- Contribute
- About us

Subscribe
Arrays
Articles
Bit Magic
C/C++ Puzzles
GFacts
Linked Lists
MCQ
Misc
Output
Strings
Trees

Level Order Tree Traversal

November 7, 2009

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:

```
#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;
}
```

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;
}
```

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.

28 comments so far

1. *Avinash* says:
   February 12, 2012 at 8:29 AM

```
/* Paste your code here (You may delete these lines if not writing code) */
void printlevelorder(struct node *tree)
{
    struct queue *Q=createQueue();
    If (!tree) return NULL;
    Enqueue(Q,root)
    while(!IsEmptyQueue(Q))
    {
        struct node *temp=DeQueue(Q);
        printf("%d",Q->data);
        If temp->left EnQueue(Q, temp->left);
        If temp->right EnQueue(Q, temp->right);
    }

    DeleteQueue(Q);
}
```

   Reply

2. *PsychoCoder* says:
   February 8, 2012 at 1:12 PM

   Same thins as method 2. But dynamic
   Same complexity : O(n)

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

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

typedef struct list {
  node *data ;
  struct list *next;
}list;

typedef struct queue {
  int count ;
  struct list *front ;
  struct list *rear;
}queue ;

void createQueue (queue **head) {
  *head = (queue *) malloc (sizeof(queue)) ;
  (*head)->count = 0 ;
  (*head)->front = NULL ;
  (*head)->rear = NULL ;
}

list* newList (node* data) {
  list *head ;
  head = (list *) malloc (sizeof(list)) ;
  head->next = NULL ;
  head->data = data ;
  return head ;
}

queue* enqueue (queue *head, node* data) {
  list *temp = newList (data) ;
  if (head->front == NULL) {
    head->front = temp ;
    head->rear = temp ;
    head->count ++ ;
  }
  else {
    head->rear->next = temp ;
    head->rear = head->rear->next ;
    head->count ++ ;
  }
```

```
  return head;
}

list* dequeue (queue **head) {
  list *temp ;
  if ((*head)->count == 0)
    return NULL ;
  if ((*head)->front != NULL) {
    temp = (*head)->front ;
    if ( (*head)->front == (*head)->rear ) {
      (*head)->front = NULL ;
      (*head)->rear = NULL ;
      (*head)->count -- ;
    }
    else
      (*head)->front = (*head)->front->next ;
      (*head)->count -- ;
  }
  return temp ;
}

int isEmpty (queue *q) {
  return (q->count == 0);
}

node* newnode (int data) {
  node *head ;
  head = (node *) malloc (sizeof(node)) ;
  head->data = data ;
  head->left = NULL ;
  head->right = NULL ;
  return head ;
}

void printLevelBFS (node *tree) {
  queue *q ;
  list *temp ;
  /* Initialize the queue */
  createQueue (&q) ;

  enqueue (q,tree) ;
  /* Use INT_MAX as delimiter */
  enqueue (q,newnode(INT_MAX)) ;

  while ( !isEmpty (q) ) {
    temp = dequeue (&q) ;

    /* Check whether it is a delimiter or not */
    if ( temp->data->data != INT_MAX ) {
      /* If not then extreme right of this level
         is not reached till now. Enqueue its
         child */
      printf ( "%d ", temp->data->data ) ;
      if (temp->data->left)
```

```
        q = enqueue (q, temp->data->left) ;

      if (temp->data->right)
        q = enqueue (q, temp->data->right) ;
    } else {
      printf ( "\n" ) ;
      /* If a delimiter is reached, then set the
         delimiter of the next level. Because
         extreme right is reached for this level */
      if ( !isEmpty(q) )
        enqueue (q,newnode(INT_MAX)) ;
    }
  }

  while (!isEmpty (q))
    dequeue (&q) ;
  free (q) ;
}

int main() {

  node *root        = newnode(10);
  root->left        = newnode(8);
  root->right       = newnode(2);
  root->left->left  = newnode(3);
  root->right->left = newnode(6) ;
  root->right->right = newnode(7) ;

  printLevelBFS (root ) ;
  free (root);
  return 1 ;
}
```

Reply

- *PsychoCoder* says:
  February 8, 2012 at 1:38 PM

  There is a small mistake in this fucntion . I have Updated it. Here it is :

```
list* dequeue (queue **head) {
  list *temp ;
  if ((*head)->count == 0)
    return NULL ;
  if ((*head)->front != NULL) {
    temp = (*head)->front ;
    if ( (*head)->front == (*head)->rear ) {
      (*head)->front = NULL ;
      (*head)->rear = NULL ;
      (*head)->count -- ;
    }
    else {
      (*head)->front = (*head)->front->next ;
      (*head)->count -- ;
    }
```

```
        }
        return temp ;
    }
```

Reply

3. *Manish_Dawar* says:
October 31, 2011 at 7:02 PM

your code wont work if we have a tree have a root only because there is nothing in the queue and we are still dequeueing it..

Reply

- *kartik* says:
October 31, 2011 at 9:39 PM

Take a closer look at the program. It works for this case also. Also see this run

Reply

- *Manish_Dawar* says:
November 1, 2011 at 12:37 AM

i tried it on gcc. Not working.. So, i jst add on a condition jst before dequeue i.e if front and rear are equal break.. Correct me, if i m wrng..

Reply

- *kartik* says:
November 1, 2011 at 3:57 PM

@Manish_Dawar: could you post the error message that you got.

Reply

4. *sachin* says:
September 20, 2011 at 10:59 AM

my solution is..
#include
typedef struct node
{
int value;
struct node *right;
struct node *left;
}mynode;
mynode *root;
add_node(int value);
void levelOrderTraversal(mynode *root);
int main(int argc, char* argv[]) {
root = NULL;
add_node(5);
add_node(1);
add_node(-20);
add_node(100);
add_node(23);
add_node(67);

```
add_node(13);
printf("\n\n\nLEVEL ORDER TRAVERSAL\n\n");
levelOrderTraversal(root);
system("pause");
return 0;
}
add_node(int value)
{
mynode *prev, *cur, *temp;
temp = (mynode *) malloc(sizeof(mynode));
temp->value = value;
temp->right = NULL;
temp->left = NULL;
if(root==NULL) {
printf("\nCreating the root..\n");
root = temp;
return;
}
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(valuevalue)?cur->left:cur->right; }
if(value value)
prev->left=temp;
else
prev->right=temp;
}
// Level order traversal..
void levelOrderTraversal(mynode *root)
{
mynode *queue[100] = {(mynode *)0};
// Important to initialize!
int size = 0;
int queue_pointer = 0;
while(root)
{ printf("[%d] ", root->value);
if(root->left) { queue[size++] = root->left; }
if(root->right) { queue[size++] = root->right; }
root = queue[queue_pointer++];
}
}
```

Reply

5. *Decompiler* says:
August 23, 2011 at 12:02 AM

What about storing each node data in multimap and printing them at last ?

```
void level_order( struct node* root, int l )
```

```
{
    int m;
    if ( root == NULL )
        return;
    m = root -> data;
    M.insert ( pair<int,int>(l,m) );
    l++;
    level_order(root->left, l);
    level_order(root->right, l);
}
```

Reply

6. *satya* says:
   May 28, 2011 at 4:59 PM

   @geeksforgeeks,,,Can you guys please make your post error free ..m getting 100 of error but just cut & paste in ideone..
   don't belive click here https://ideone.com/txVW8

   like prog.cpp:4: error: stray ‘\302’ in program
   prog.cpp:4: error: stray ‘\240’ in program
   prog.cpp:9: error: stray ‘\302’ in program
   prog.cpp:9: error: stray ‘\240’ in program
   prog.cpp:9: error: stray ‘\302’ in program
   prog.cpp:9: error: stray ‘\240’ in program
   prog.cpp:9: error: stray ‘\302’ in program
   prog.cpp:9: error: stray ‘\240’ in program
   & so on

   Reply

   - *GeeksforGeeks* says:
     May 28, 2011 at 6:24 PM

     @satya: We tried to run both the given programs using ideone. They both worked fine for us. See the following links.

     https://ideone.com/XEpQw

     https://ideone.com/2PkJ9

     To avoid selecting extra characters when copying, double click anywhere on the code segment and copy the code.

     Reply

     - *Pragya* says:
       September 20, 2011 at 11:36 AM

       @Satya- Even me too encountered with same error many times but now i got the reason behind it. Actually, whenever u copy and paste source code in your .c file some characters are not read properly by ur editor. Mostly , u can see this when u will copy and paste any printf() then the color of characters written inside " " is pink in linux, but sometimes when u copy and paste the color remains black, it means it is not yet read properly... So delete double quotes (" ") and type by your own keyboard the quotes only and then u will find the

color changes to pink ...so now no error will come... I hope u got it... If not got then do let me know please...

Reply

7. *mike* says:
April 12, 2011 at 9:03 PM

Hey guys!
so what is the time complexity of the first code?
it it (O)n or (O)n^2 ???

Reply

- *Sandeep* says:
April 16, 2011 at 11:21 PM

@mike:
We have updated the post and added explaination for the time complexity of method 1.

Reply

8. *rohith* says:
October 4, 2010 at 1:31 PM

in the first solution in the height function, with out counting how will the height of left an right subtree caculated??

Reply

- *kartik* says:
October 9, 2010 at 3:06 PM

@rohith
I am not able to understand your question . Please elaborate.

Reply

- *Pragya* says:
April 17, 2011 at 7:46 PM

@Rohith- In height function of the first solution,u can see the inner if-else statement there you can find that height of left subtree(lheight) and height of right subtree (rheight) is increment as per condition, and thus able to count height.

Reply

9. *Lakshmi* says:
September 28, 2010 at 10:33 PM

What is the time complexity of the first method?

Reply

- *kartik* says:
October 9, 2010 at 3:09 PM

I think time complexity is O(n^2). We can see this for a skewed tree.

Reply

10. *Anshul* says:
August 26, 2010 at 11:24 AM

will this give this output:

1
2 3
4 5

i.e by printing at each level in seperate line. How this can be done?

also this line: /*Dequeue node and make it temp_node*/
temp_node = deQueue(queue, &front);

will only print a Node while we are filling left and right again and again?

Reply

11. *gauravs* says:
August 10, 2010 at 10:25 PM

What will be the time complexity of recursive solution of level order traversal ?

Reply

12. *Shrijeet* says:
April 5, 2010 at 6:40 AM

```
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);
  }
}
```

There is an error in code, note the correction needed.
Thanks!

Reply

- *GeeksforGeeks* says:

April 5, 2010 at 1:57 PM

@Shrijeet: Thanks for pointing this out. We have made the suggested changes.

Reply

13. *Rosy* says:
November 16, 2009 at 11:44 PM

Thanks..Yeah It will work perfectly..After writing the comments only, I realized that but did n't know how to delete my comments..:)..But thanks for the explanation..

Its really a great site..:)

Reply

14. *geeksforgeeks* says:
November 16, 2009 at 10:51 PM

@Rosy: The first solution will work as printGivenLevel() prints the data when *level* becomes 1. For i = 1 (i.e. root) there won't be any recursive calls as *level* is 1 so root node will be printed. For other levels, passed *level* will be more than 1, traversal will start from root and *level* will be decremented as we move down the tree recursively. When we reach the required level, *level* will become 1 and nodes will be printed. You can try the code with different trees.

Reply

15. *Rosy* says:
November 16, 2009 at 7:16 PM

Hi,

I do n't think the first solution is going to work..

```
Inside printLevelOrder(struct node* root)
{
  ....
  printGivenLevel(root, i); //here its starting from i=0
}
```

And Inside

```
printGivenLevel(struct node* root, int level)
{
  ...
  if(level >1)
   //calling recursively by passing the left and right child
   // as the first argument and (level-1) as the 2nd argument.
   //Here level starts from zero..so how the revursive calls
   //are going to happen..
}
```

Please comment if my understanding is incorrect..

Reply

16. *geeksforgeeks* says:
November 12, 2009 at 10:15 PM

@Shikha: Thanks for pointing out this. There was a typo. We have corrected it. Keep it up!!

Reply

17. *Shikha* says:
November 12, 2009 at 5:02 PM

"Time Complexity: O(n)
Space Complexity: O(n)
Space complexity will be equal to height of the tree and for a skewed tree height can be O(n)."

Space complexity will be equal to number of nodes in the tree. 'n' refers to number of nodes in tree.

Reply

Comment

Name (*Required*) Email (*Required*) Website URI

Your Comment (Writing code? please paste your code between sourcecode tags)

```
[sourcecode language="C"]
/* Paste your code here (You may delete these
lines if not writing code) */
[/sourcecode]
```

Have Your Say

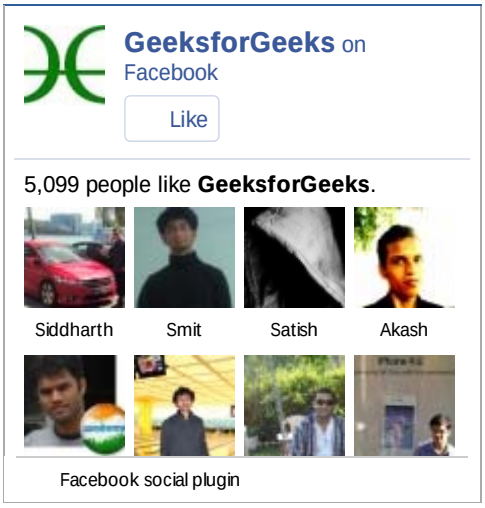
Notify me of followup comments via e-mail

Subscribe without commenting
E-Mail: Subscribe

- Search

- # Popular Tags

  - GATE
  - Java
  - Dynamic Programming
  - Divide & Conquer
  - Backtracking
  - Pattern Searching
  - Operating Systems
  - Recursion



- # Forum Latest Discussion

  - Amazon Interview Question for Software Engineer/Developer about Trees
    Last Post By: rageguy
    Inside: Interview Questions
  - How to find out the position of a msb in a number?
    Last Post By: nishu
    Inside: Interview Questions
  - Amazon Interview Question for Software Engineer/Developer (Fresher) about Algori [...]
    Last Post By: PsychoCoder
    Inside: Interview Questions
  - Regarding Heap for Dynamic memory allocation
    Last Post By: geek4u
    Inside: Interview Questions
  - Amazon Interview Question for Software Engineer/Developer about Trees
    Last Post By: Nages
    Inside: Interview Questions
  - samsung interview question.....
    Last Post By: tanmaydude
    Inside: Interview Questions
  - Yahoo Interview Question for Software Engineer/Developer (0 - 2 Years) about Alg [...]
    Last Post By: anji.swe
    Inside: Interview Questions
  - trees
    Last Post By: kartik
    Inside: Trees specific questions

- # Popular Posts

  - Sorted Linked List to Balanced BST
  - Tree traversal without recursion and without stack!
  - All permutations of a given string
  - Memory Layout of C Programs
  - Longest substring without repeating characters
  - 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
- Understanding “extern” keyword in C
- Median of two sorted arrays

Forum Categories

- Interview Questions
- C/C++ Programming Questions
- Algorithms
- Trees specific questions
- Linked List specific questions
- Multiple Choice Questions
- Object oriented queries
- GPuzzles
- Operating Systems
- Miscellaneous
- Java specific Questions
- Perl specific Questions

Add Interview Questions

If you have attended an interview recently or have good interview questions to share with the fellow geeks and get the best solution, please add your questions to Add Interview Questions page

Recent Comments

- Rahul sharma on Print a given matrix in spiral form
- shashank paliwal on Given an array arr[], find the maximum j – i such that arr[j] > arr[i]
- shashank paliwal on Given an array arr[], find the maximum j – i such that arr[j] > arr[i]
- Sandeep Vasani on Find k-th smallest element in BST (Order Statistics in BST)
- kartik on Minimum number of jumps to reach end
- GeeksforGeeks on Minimum number of jumps to reach end
- kv391 on Minimum number of jumps to reach end
- anji.swe on Find whether a given number is a power of 4 or not