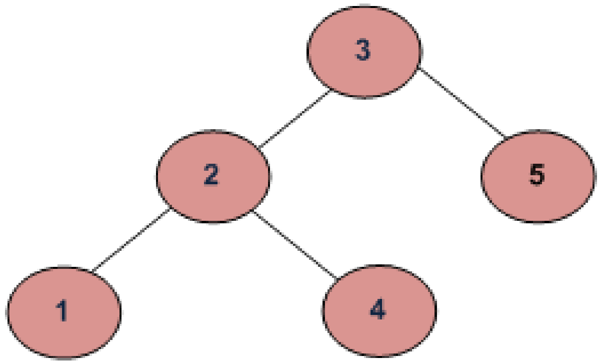
Get Level of a node in a Binary Tree

Given a Binary Tree and a key, write a function that returns level of the key.

For example, consider the following tree. If the input key is 3, then your function should return 1. If the input key is 4, then your function should return 3. And for key which is not present in key, then your function should return 0.



Thanks to prandeey for suggesting the following solution.

The idea is to start from the root and level as 1. If the key matches with root's data, return level. Else recursively call for left and right subtrees with level as level + 1.

```
#include<stdio.h>

/* A tree node structure */
struct node
{
    int data;
    struct node *left;
    struct node *right;
```

GeeksforGeeks
52,731 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

```
};

/* Helper function for getLevel().  It returns level of the data if da
   present in tree, otherwise returns 0.*/
int getLevelUtil(struct node *node, int data, int level)
{
    if (node == NULL)
        return 0;

    if (node->data == data)
        return level;

    int downlevel = getLevelUtil(node->left, data, level+1);
    if (downlevel != 0)
        return downlevel;

    downlevel = getLevelUtil(node->right, data, level+1);
    return downlevel;
}

/* Returns level of given data value */
int getLevel(struct node *node, int data)
{
    return getLevelUtil(node,data,1);
}

/* Utility function to create a new Binary Tree node */
struct node* newNode(int data)
{
    struct node *temp = new struct node;
    temp->data = data;
    temp->left = NULL;
    temp->right = NULL;

    return temp;
}

/* Driver function to test above functions */
int main()
{
    struct node *root = new struct node;
    int x;

    /* Constructing tree given in the above figure */
    root = newNode(3);
    root->left = newNode(2);
    root->right = newNode(5);
```

Geometric Algorithms

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

```
    root->left->left = newNode(1);
    root->left->right = newNode(4);

    for (x = 1; x <=5; x++)
        printf(" Level of %d is %d\n", x, getLevel(root, x));

    getchar();
    return 0;
}
```

Output:

```
Level of 1 is 3
Level of 2 is 2
Level of 3 is 1
Level of 4 is 3
Level of 5 is 2
```

Time Complexity: O(n) where n is the number of nodes in the given Binary Tree.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

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

3 Tweet 0 0

Writing code in comment? Please use **ideone.com** and share the link here.

39 Comments **GeeksforGeeks**

Sort by Newest

Join the discussion…

Uma Trika • 4 months ago

void getLevelUtil(struct node *node, int h, int *level, int key)
{
if(node == NULL)
return ;

if(node->data == key)
{

695 Subscribe

Recent Comments

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

Backtracking | Set 7 (Sudoku) · 43 minutes ago

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

Flipkart Interview | Set 6 · 1 hour ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 1 hour 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 · 3 hours ago

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

Find depth of the deepest odd level leaf node · 3 hours ago

```
*level = h ;
}

getLevelUtil(node->left, h+1, level, key);
getLevelUtil(node->right, h+1, level, key);
}
```

Reply • Share ›

hhh • 5 months ago

I think it is O(n*logn) - getLevelUntil is O(logn) and when we run it for each eler
will be O(n*logn)

1 • Reply • Share ›

Pranav hhh • 4 months ago

Its O(n) because if you look closely you will find that getLevelUtil() first
tree and finally right sub tree.
So every node is visited exactly once.

Reply • Share ›

DarkProtocol • 7 months ago

How is this T(n)= O(n)... it basically does searching and which should be O(lo

Reply • Share ›

cool_dude • 7 months ago

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


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

struct node *create_node(int num)
{
struct node *temp=(struct node*)malloc(sizeof(struct node ));
temp->left=NULL;
temp->right=NULL;
temp->data=num;
```

see more

Reply • Share ›

cool_dude • 7 months ago

#include<stdio.h>

#include<stdlib.h>

struct node

{

struct node *left;

struct node *right;

int data;

};

struct node *create_node(int num)

{

struct node *temp=(struct node*)malloc(sizeof(struct node));

see more

^ | ⌄ • Reply • Share ›

Vandal • 8 months ago

/* A much simpler Implementation It returns 0 if key not found*/

```
int keyLevel(int key, tNode* n, int count)
{

if(n == NULL)
return 0;

if( n->data == key)
return count;
else
{
return max(keyLevel(key, n->left, count+1), keyLevel(key, n->right, count+1));

}

}
```

4 ^ | ⌄ • Reply • Share ›

Sunil • 10 months ago

Why is a bitwise or used instead of logical or in the following line?
return getLevelUtil (node->left, data, level+1) |
getLevelUtil (node->right, data, level+1);

I compiled the program using 'logical or' for input=4 with the same binary tree,

^ | ⌄ • Reply • Share ›

GeeksforGeeks ➦ Sunil • 10 months ago

@Sunil: We have updated the code to avoid confusion.

The bitwise or was used to return integer value, rather than boolean va
works for trees with all distinct keys, but may fail for same key in left ar
Following is old code for record.

```
 #include<stdio.h>

/* A tree node structure */
struct node
{
  int data;
  struct node *left;
  struct node *right;
};

/*
   Helper function for getLevel().  It returns level of the dat
```

see more

Reply • Share ›

Silent • 11 months ago
why this is not giving correct result??

/* int level(struct tree *root,int k,int count)
{
if(root&&root->info==k)
return count;
if(root)
{
return level(root->left,k,count+1);

```
return level(root->right,k,count+1);
}
}
*/
```

Reply • Share ›

Linuxwc Silent • 11 months ago

In addition, you finish the activation (at the latest) after calling the left su

The line

return level(root->right,k,count+1);

is never executed since it is located after an unconditional return-stater

Reply • Share ›

shek8034 Linuxwc • 11 months ago

@Linuxwc : Yes, you are right.
The correct return statement at the end should be:-
return level(root->left,k,count+1)||level(root->right,k,count+1);

Reply • Share ›

Silent shek8034 • 11 months ago

yes i got it..

Reply • Share ›

shek8034 Silent • 11 months ago

Add this line in the beginning :
if (root == NULL)
return 0; // for empty tree

1 • Reply • Share ›

Wah Wah kya bat hai !!!! main toh sach main dar gaya !!!

^ | ⌄ • Reply • Share ›

kk • 11 months ago

kanha se aaye ho bhai....

^ | ⌄ • Reply • Share ›

vineet • a year ago

I think there is a simpler way out.
We find height(h1) of tree rooted at the target node(the node, for which we hav
height(h2) of the original tree.
The difference(h2 - h1 + 1) should be the answer I think. Please correct if I am
extra check for the case where the target node doesn't exist in the tree.

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

^ | ⌄ • Reply • Share ›

Linuxwc ➦ vineet • a year ago

It works if all paths have the same length. For example, perfect binary t

^ | ⌄ • Reply • Share ›

Hanish Bansal • a year ago

I think we might do the question this way in O(log n) time :
Follow the search procedure of a tree and keep adding 1 to the (static int level
If we reach NULL, return (level=0);
We will find the level of the node in O(log n) time this way instead of O(n) as m
Please correct me in case i am wrong...

^ | ⌄ • Reply • Share ›

Hanish ➦ Hanish Bansal • a year ago

I was completely thinking out of the line. I considered it as a BST.

Reply • Share ›

Sayak • 2 years ago

int f(node * T, int k)
{
if (!T) return 0;
if (T->val == k) return 1;
int left = f(T->left, k);
int right = f(T->right, k);

if (left == right) return 0;
else return (left>right)? (left + 1): (right + 1);
}

Reply • Share ›

PsychoCoder • 2 years ago

Here is the BFS implementation of the tree. If you find any node return the leve
space is required!! But still another approach.

```
 #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;
```

see more

^ | ˅ • Reply • Share ›

Hemil • 2 years ago

What if we solved this using BFS.

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

^ | ˅ • Reply • Share ›

Agniswar • 3 years ago

Hi,this is my solution-

int get_level(node *root,int k)
{
static int level=1;
if(root==NULL)
return 0;
if(root->data==k)
return level;
else if(root->left->data==k || root->right->data==k)
return (++level);
else if(get_level(root->left,k) || get_level(root->right,k))
return (++level);
else
return level;
}

Reply • Share ›

Linuxwc ➦ Agniswar • a year ago

If level is static, it should be zero when nothing has been found and init
See the code below for details.

```
 int get_level(node *root,int k)
{
//static int level=1;
static int level=0;
if(root==NULL)
return 0;
if(root->data==k)
// return level;
    return (level=1); // Because this is static,
// initialize here
// else if(root->lelt->data==k || root->right->data==k)
// return (++level);
// Remove those lines since the pointer may be null
// and the lines are redundant
else if(get_level(root->left,k) || get_level(root->right,k))
return (++level);
else
// return level;
return 0; // Do not use level if value not found
}
```

Reply • Share ›

Sayak ➦ Agniswar • 2 years ago

dude check mine as well,

```
int f(node * T, int k)
{
if (!T) return 0;
if (T->val == k) return 1;
int left = f(T->left, k);
int right = f(T->right, k);

if (left == right) return 0;
else return (left>right)? (left + 1): (right + 1);
}
```

• Reply • Share ›

Harshit • 3 years ago

```
 int getlevel(treeptr p,int x)
{


    if(p==NULL)
    return 0;

    if(p->info==x)
    return 1;

    int d;

    if(d=getlevel(p->left,x))
    return 1+d;

    if(d=getlevel(p->right,x))
    return 1+d;
    else
}
```

^ | ⌄ • Reply • Share ›

anantha89 • 3 years ago

Hi All,

I guess we can avoid traversing the entire tree by modifying like this,

```
int getNodeL(Tnode *root, int key, int level) {
    int leftlevel = 0, rightlevel = 0;

    if (root == NULL)
        return 0;

    if (key == root->data)
        return level;

    leftlevel = getNodeL(root->left, key, level + 1);

    if (leftlevel == 0)
        rightlevel = getNodeL(root->right, key, level + 1);

    return leftlevel + rightlevel;
}
```

^ | ⌄ • Reply • Share ›

mrn • 3 years ago

i donno y this code is not working :(

```
int level(int v)
```

```
{
        map<struct node *,int > q;
        struct node *n;
        int l=0;
        q.insert(make_pair(root,l+1));
        map<struct  node *,int>::iterator it;
        for(it=q.begin();it!=q.end();l++)
        {
                if(!it->first)
                        break;
                n=it->first->r;
                printf("it:%d\n",n->v);
                if(it->first->v==v)
                {
                        return it->second;
```

see more

^ | ⌄ • Reply • Share ›

Linuxwc ➦ mrn • a year ago

You increment the level every time you get a node from the list, even th
level. Moreover, remove your test lines that may point to contents of a r
more details.

```
 int getLevel(int v)
{
    map<struct node *,int > q;
    struct node *n;
    int l=0;
    q.insert(make_pair(root,l+1));
    map<struct  node *,int>::iterator it;
    for(it=q.begin();it!=q.end();) // remove l++ since
    // l must be increased only when new level starts
    // in this breadth first -list
    {
        if(!it->first)
            break;
    //          n=it->first->r;
```

see more

^ | ⌄ • Reply • Share ›

Trouble • 3 years ago

```
 int getLevel(node* root, int key, int level){
        if(!root) return -1;
        if(root->val == key) return level;
        int ld, rd;
        ld = getLevel(root->l, key, level+1);
        rd = getLevel(root->r, key, level+1);
        return ld>rd?ld:rd;
}
```

^ | ⌄ • Reply • Share ›

hari6988 • 3 years ago

Hey,just a thought...

return getLevelUtil (node->left, data, level+1) |
getLevelUtil (node->right, data, level+1);

This works only if all the data values of nodes in the tree are distinct ... IF there
with the same data value, then | operator will add them up n display the added

^ | ⌄ • Reply • Share ›

Vijay ➦ hari6988 • 2 years ago

If elements are guaranteed to be distinct in the tree, we can check thro
need to pass the address of the node to be searched

```
/* Paste your code here (You may delete these lines if not wr:
```

Reply • Share ›

manishj • 3 years ago

Iterative approach :

```
 int find_level(node *root, int key)
{
        int depth = 0;
        node *temp = root;
        while(temp != NULL)
        {
                if(temp->data == key)
                        return depth;
                else if(temp->data < key)
                {
                        depth++;
                        temp = temp->right;
                }
                else
                {
                        depth++;
                        temp = temp->left;
                }

        }
        return depth;
}
```

1 ⌃ | ⌄ • Reply • Share ›

aditya ➦ manishj • 7 months ago

this is not a bst

⌃ | ⌄ • Reply • Share ›

RAW ➦ manishj • 3 years ago

I think finally depth should not be returned otherwise if the node is not p
tree... return -1 instead.

⌃ | ⌄ • Reply • Share ›

Hill ➦ manishj • 3 years ago

this works for BST only. But, the problem is to find for binary tree

⌃ | ⌄ • Reply • Share ›

manishj ➦ Hill • 3 years ago

```
 int levelorder_traversal(btree *root,int key)
{
        queue<btree*> q;

        btree dummy ;

        q.push(root);
        q.push(&dummy);
    int level = 0;
        while(!q.empty())
        {
                btree *node = q.front();
                q.pop();
                if(node == &dummy)
                {
                        level++;
                        if(!q.empty())
                                q.push(&dummy);
```

see more

Reply • Share ›

mrn ➦ manishj • 3 years ago

why is it not working :(

```
 int level(int v)
{
        map<struct node *,int > q;
        struct node *n;
        int l=0;
        q.insert(make_pair(root,l+1));
        map<struct  node *,int>::iterator it;
        for(it=q.begin();it!=q.end();l++)
        {
                if(!it->first)
                        break;
                n=it->first->r;
                printf("it:%d\n",n->v);
                if(it->first->v==v)
                {
                        return it->second;
```

see more

Reply • Share ›