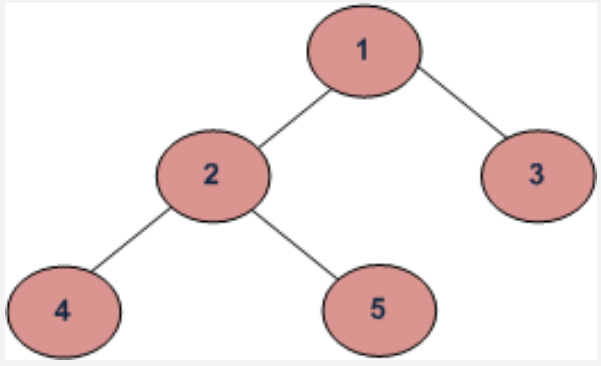
Write a C Program to Find the Maximum Depth or Height of a Tree

Maximum depth or height of the below tree is 3.



Example Tree

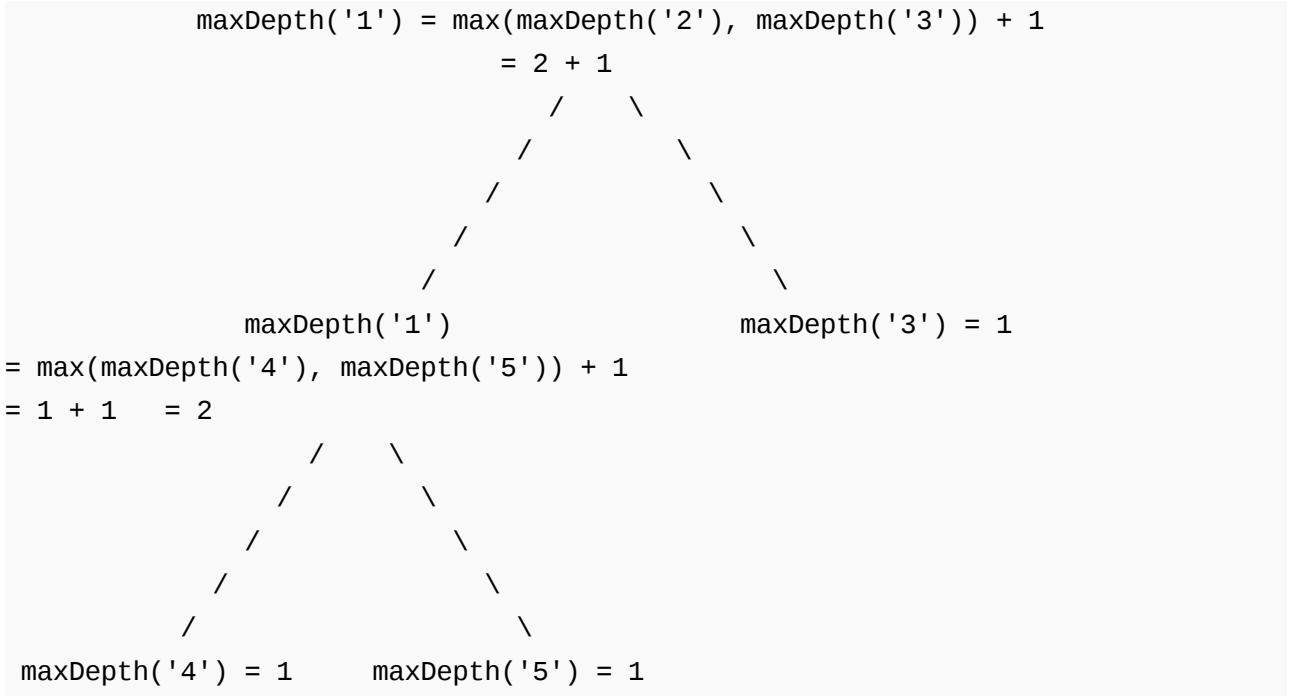
Recursively calculate height of left and right subtrees of a node and assign height to the node as max of the heights of two children plus 1. See below pseudo code and program for details.

Algorithm:

```
 maxDepth()
1. If tree is empty then return 0
2. Else
     (a) Get the max depth of left subtree recursively  i.e.,
          call maxDepth( tree->left-subtree)
     (a) Get the max depth of right subtree recursively  i.e.,
```


```
         call maxDepth( tree->right-subtree)
    (c) Get the max of max depths of left and right
         subtrees and add 1 to it for the current node.
        max_depth = max(max dept of left subtree,
                            max depth of right subtree)
                            + 1
    (d) Return max_depth
```

See the below diagram for more clarity about execution of the recursive function maxDepth() for above example tree.



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
{
```


Popular Posts

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

/* Compute the "maxDepth" of a tree -- the number of
    nodes along the longest path from the root node
    down to the farthest leaf node.*/
int maxDepth(struct node* node)
{
   if (node==NULL)
       return 0;
   else
   {
       /* compute the depth of each subtree */
       int lDepth = maxDepth(node->left);
       int rDepth = maxDepth(node->right);

       /* use the larger one */
       if (lDepth > rDepth)
           return(lDepth+1);
       else return(rDepth+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);
}

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("Hight of tree is %d", maxDepth(root));

    getchar();
    return 0;
}
```

Time Complexity: O(n) (Please see our post Tree Traversal for details)

References:
http://cslibrary.stanford.edu/110/BinaryTrees.html

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

Recent Comments

karthik it should have been max_wrap= max_wrap -...
Maximum circular subarray sum · 1 minute ago

affiszerv Your example has two 4s on row 3, that's why it...
Backtracking | Set 7 (Sudoku) · 45 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

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

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

38 Comments GeeksforGeeks

Sort by Newest

an • 5 days ago

Why not have a counter variable count the number of nodes in a tree whenev

when the depth has to be found, convert the count to binary - the position of le

If the number of nodes is 37, binary value is 00100101 - 6th place from right is tree is 6.

Marsha Donna • 2 months ago

```
int height(struct node *root)
{
if(root==NULL)
return 0;
return 1+max(height(root->left),height(root->right));

}
```


RANDOMIZER • 5 months ago

```
int maxDepth(treeptr s){
if(s==NULL)
```

if(s==NULL)
return 0;
return(maxDepth(s->left)>maxDepth(s->right)?maxDepth(s->left)+1:maxDepl
}

3 ^ | v • Reply • Share ›

Kaladhar • 5 months ago

int maxDepth(TreeNode *root){

if(root==NULL)return 0;

return max(maxDepth(root->left)+1,maxDepth(root->right)+1);

}

1 ^ | v • Reply • Share ›

Musaddique Hossain • 8 months ago

```
struct Node{

        int val;

        struct Node *left, *right;

};

typedef struct Node node;

node* newNode(int val){

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

        temp->val = val;
```

```
        temp->left = NULL;
```

see more

• Reply • Share ›

Rahul Mahale • 9 months ago

```
 int maxDepth(BST *root)
{
        int ldepth=0,rdepth=0,maxdepth=0;

        if(root==NULL)
          return(0);

        ldepth=maxDepth(root->left);
        rdepth=maxDepth(root->right);

        maxdepth=MAX(ldepth,rdepth);
        return(maxdepth+1);

}
```

4 • Reply • Share ›

PK's • 9 months ago

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

#include<stdio.h>
#include<iostream>

struct node
{

```
{
int data;
struct node* left;
struct node* right;
};
void findHieght(struct node*,int);
int depth;
int main()
{
struct node* root = new node();
root->data = 0;
```

see more

Reply • Share ›

hak23 • 11 months ago

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

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

int max(a,b)
{
        return(a>b?a:b);
}

int height(struct node* root)
{
        if(root == NULL)
                return 0;
```

```
        return(max(height(root->left),height(root->right))+1);
```

see more

• Reply • Share ›

initialcoder • 11 months ago

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

typedef struct NodeTag{
        char SYMBOL;
        struct NodeTag * LLINK;
        struct NodeTag * RLINK;
} TreeNode;


int getHeight(TreeNode * root){

        if(root == NULL)
                return 0;
        int lHeight =0;
        int rHeight = 0;

        lHeight = getHeight(root->LLINK);
```

see more

• Reply • Share ›

abhishek08aug • a year ago

C++ code: extended from the one I posted in: http://www.geeksforgeeks.org/w

```
#include<iostream>
using namespace std;

class tree_node {
  private:
    int data;
    tree_node * left;
    tree_node * right;
  public:
    tree_node() {
      left=NULL;
      right=NULL;
    }
    void set_data(int data) {
      this->data=data;
```

see more

• Reply • Share ›

Ankit Sablok • a year ago

Well as is written correctly, the height of the tree is essentially the height of the
just a single node in the tree or for the case of leaf nodes present in the tree, tl
case but this program returns 1, which is false, so one could refrain adding 1 a
output is maxDepth(rootNode) - 1, to compensate for the 1 that you add to finc

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

• Reply • Share ›

neha2210 Ankit Sablok • a year ago

Yes, or we could just make the base case set to return as -1, rather th

1 ^ | v • Reply • Share ›

bikram ➦ neha2210 • a year ago

```
/* Paste your code here (You may delete these lines if r
```

^ | v • Reply • Share ›

neha2210 • a year ago

@geeksforgeeks
Isn't this program returning 1 for a tree with only root node while it should be 0
example should be 2 and not 3?
Please correct me if I am wrong.

^ | v • Reply • Share ›

diepakk ➦ neha2210 • a year ago

@neha2210 I think height for a tree with only one node should be 1 anc
me.

^ | v • Reply • Share ›

Sasuke ➦ neha2210 • a year ago

ya..i also think the same .Where should the code be changed to get th

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

^ | v • Reply • Share ›

ksharp ➦ Sasuke • 4 months ago

@neha2210 Yes you are right. See the below wiki link:
http://en.wikipedia.org/wiki/T...

^ | v • Reply • Share ›

Nikin • a year ago

```
int height(node *sr)
{
if(sr == NULL) return 0;
return max(height(sr->left), height(sr->right)) + 1;
}
```

^ | v • Reply • Share ›

Ravindar Reddy Lenkala • a year ago

thanks a lot.....

^ | v • Reply • Share ›

Cracker • 2 years ago

http://algods-cracker.blogspot...

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

^ | v • Reply • Share ›

nonlinearly • 2 years ago

In the diagram in the node 2 (second layer) there is an error. Not maxDepth('1'

Keep the good work...Bravo

^ | v • Reply • Share ›

Vinoth • 2 years ago

This should work

```
private int depth(Node node)
        {
                if(node == null)
                {
                        return 0;
                }

                return max (depth(node.left) +1,depth(node.right) +1)

        }
```

  • Reply • Share ›

Swati • 3 years ago

Hi
can u pls xplain how the func maxdepth is calcutaing the height...i mn whr is d undersatnd the base condition...suppose the node is not null it will move to left node...n dis will keep happening till it reaches the end dat is till it becomes null the height calculated..

  • Reply • Share ›

neha2210 ➦ Swati • a year ago

You yourself have given half the answer.. "it reaches the end dat is till i null it returns zero value. which is later taken as 0+1 for height of leaf n

1   • Reply • Share ›

abhishek ➦ Swati • 2 years ago

I am also not able to understand about the height increase.
When the base case become zero and returning zero than how is it gc opinion

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

Reply • Share ›

sankalp • 3 years ago

The function for calculating max. depth can be converted into a little utility !

```
int depth(tree* r,int d)
{
   if(r==NULL)
     return d;
   return max(depth(r->l,d+1),depth(r->r,d+1));
}
```

I think that this should work fine.. Correct me if I am wrong!

Reply • Share ›

sankalp ➚ sankalp • 3 years ago

The arguments to function call should be:

depth(root,0);

Reply • Share ›

varaprasad • 3 years ago

Shouldn't the function "maxDepth" be returning -1 instead of 0 if the root is NU

Thanks

Reply • Share ›

Sandeep ➚ varaprasad • 3 years ago

@varaprasad: There are two different conventions in different text book convetion considers height as number of edges in the path from root to height as number of nodes in thh path from root to the farthest leaf. We in above implementation.

^ | v • Reply • Share ›

Dhanya • 4 years ago

The height of the above tree is not 3 but 2 .Please check

^ | v • Reply • Share ›

A • 4 years ago

This was helpful. Thanks

^ | v • Reply • Share ›

Aditya • 4 years ago

Beautifully done!

^ | v • Reply • Share ›

Sandeep • 4 years ago

@neet: Level order traversal can be used to get the height iteratively. Please s
http://geeksforgeeks.org/?p=26....

You need to make following modifications to level order traversal code:
Every queue element will be a structure having two things: (i) node pointer (ii)

```
struct queue_element
{
  struct node *node;
  int level
}
```

Initialize max_level as 0.

When you enqueue an element, enqueued level will be parent's level + 1.

When you dequeue an element, you need to compare element's level with the

see more

• Reply • Share ›

Neet • 4 years ago

can you give a iterative version?

• Reply • Share ›

geeksforgeeks • 4 years ago

@nida: The function works. In each recursive step mas depth is calculated as right_child_depth) + 1.

We have added a diagram for showing the recursive execution of maxDepth()

• Reply • Share ›

nida • 4 years ago

how is the frst code calculating maxdepth??
i mean dere is no counter or somethng so how are we calculating dat??

1 • Reply • Share ›

abhishek08aug nida • a year ago

nida that is the beauty of recursion. You keep on building your solution return values.

• Reply • Share ›

Raghav nida • 3 years ago

ohh cmmon nida...it will return the maximum value of i.e d+1 will give tl

• Reply • Share ›

Subscribe Add Disqus to your site