GeeksforGeeks
A computer science portal for geeks

Login

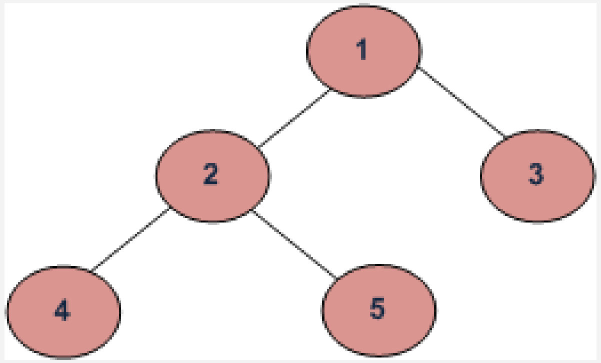
GeeksQuiz

Home Algorithms DS GATE Interview Corner Q&A C C++ Java Books Contribute Ask a Q About
Array Bit Magic C/C++ Articles GFacts Linked List MCQ Misc Output String Tree Graph

Iterative Method to find Height of Binary Tree

There are two conventions to define height of Binary Tree
1) Number of nodes on longest path from root to the deepest node.
2) Number of edges on longest path from root to the deepest node.

In this post, the first convention is followed. For example, height of the below tree is 3.



Example Tree

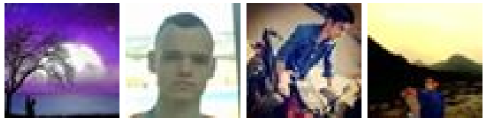
Recursive method to find height of Binary Tree is discussed here. How to find height without recursion? We can use level order traversal to find height without recursion. The idea is to traverse level by level. Whenever move down to a level, increment height by 1 (height is initialized as 0). Count number of nodes at each level, stop traversing when count of nodes at next level is 0.
Following is detailed algorithm to find level order traversal using queue.

```
Create a queue.
```

Google™ Custom Search

GeeksforGeeks

52,731 people like GeeksforGeeks.



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

```
Push root into the queue.
height = 0
Loop
        nodeCount = size of queue

        // If number of nodes at this level is 0, return height
        if nodeCount is 0
                return Height;
        else
                increase Height

        // Remove nodes of this level and add nodes of
        // next level
        while (nodeCount > 0)
                pop node from front
                push its children to queue
                decrease nodeCount
       // At this point, queue has nodes of next level
```

Following is C++ implementation of above algorithm.

```
/* Program to find height of the tree by Iterative Method */
#include <iostream>
#include <queue>
using namespace std;

// A Binary Tree Node
struct node
{
    struct node *left;
    int data;
    struct node *right;
};

// Iterative method to find height of Bianry Tree
int treeHeight(node *root)
{
    // Base Case
    if (root == NULL)
        return 0;
```

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

```
    // Create an empty queue for level order tarversal
    queue<node *> q;

    // Enqueue Root and initialize height
    q.push(root);
    int height = 0;

    while (1)
    {
        // nodeCount (queue size) indicates number of nodes
        // at current lelvel.
        int nodeCount = q.size();
        if (nodeCount == 0)
            return height;

        height++;

        // Dequeue all nodes of current level and Enqueue all
        // nodes of next level
        while (nodeCount > 0)
        {
            node *node = q.front();
            q.pop();
            if (node->left != NULL)
                q.push(node->left);
            if (node->right != NULL)
                q.push(node->right);
            nodeCount--;
        }
    }
}
```

```
// Utility function to create a new tree node
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()
{
    // Let us create binary tree shown in above diagram
    node *root = newNode(1);
```

```
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    cout << "Height of tree is " << treeHeight(root);
    return 0;
}
```

Output:

```
Height of tree is 3
```

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

This article is contributed by **Rahul Kumar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

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)

Recent Comments

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

Backtracking | Set 7 (Sudoku) · 31 minutes ago

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

Flipkart Interview | Set 6 · 51 minutes ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 51 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

- 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

28 Tweet 4 2

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

25 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion…

GOPI GOPINATH • a month ago

Here is the implementation of iterative method to find height of a binary tree wi

```
#include<iostream>

#include<stdio.h>

#include<stdlib.h>

struct Treenode

{

int data;

struct Treenode * left;

struct Treenode *right;
};

struct Treenode* newnode(int data)
```

see more

⌃ | ⌄ • Reply • Share ›

Guest • a month ago

Here is the solution for finding the height (or depth) of a binary tree without rec

http://ideone.com/e.js/ndP4PS

⌃ | ⌄ • Reply • Share ›

isha • 6 months ago

as you have discussed here that we find the height of a tree by the Number of deepest node then according this what should be the height of a tree 2 or 3 fo

1 ⌃ | ⌄ • Reply • Share ›

anonymous ➦ isha • 5 months ago

The usual convention says that the height of such a tree should be 2. T the height.

The only problem with this is that, when you write the recursive functio number of edges, you would have to give the base case as

if(!root)

return -1;

That is, if we count it as the number of edges, then both, a tree with on tree as -1.

⌃ | ⌄ • Reply • Share ›

Nitin Sharma • 6 months ago

/*HEIGHT OF TREE WITHOUT LEVEL ORDER TRAVERSAL*/

#include<stdlib.h>

```
#include<stdio.h>

typedef struct node
{
int value;
struct node *left,*right;
}node;

node* newnode(int n)
{
node *tmp;

tmp = (node*)calloc(1,sizeof(node));

if(tmp==NULL)
{
```

see more

^ | ⌄ • Reply • Share ›

Patil • 7 months ago

Here is C implementation.

```
int treeHeight(mynode *root)
{
if(root == NULL)
return 0;
mynode *queue[20];
int height,front,rear;
height=0;
front = 0;
rear = 1;
queue[rear] = root;
```

```
while(1)
{
int nodeCount = (rear-front);
if(nodeCount == 0)
return height;
else
```

see more

2 ⌃ | ⌄ • Reply • Share ›

12rad • 9 months ago

Java Implementation:

```
        public static int getHeightOFtree_Iterative(Node root){

                Deque<Node> a = new LinkedList<Node>();
                int height = 0;

                int nodesinCurrentLevel =0;

                if(root == null){
                        return height;
                }

                a.add(root);
                height ++;

                nodesinCurrentLevel++;
                int nodeinNextLevel = 0;
```

see more

⌃ | ⌄ • Reply • Share ›

ankur jain • 9 months ago

[sourcecode language="Cpp"]

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<vector>
#include<set>
#include<map>
#include<string>

#define input freopen("input.txt","r",stdin)
#define output freopen("out.txt","w",stdout)
//a=a+b-(b=a);

using namespace std;
/*
struct tree
{
int data;
```

see more

^ | ˅ • Reply • Share ›

Akshay Jindal • 9 months ago

Here's the c implementation tested for the above tree
My approach uses a stack based Iterative inorder traversal
In my approach a node will have 2 extra fields
1.parent(to traverse upwards)
2.visited

visited--->Here's what it means
1.node->visited=0 ---> It means that the node has been unvisited yet

2.node->visited=1 ---> It means that the node has been visited but its left and r
the node into the stack)
3.node->visited=2 ---> It means that the node has been visited and its left chilc
child from the stack)
4.node->visited=3 ---> it means that the node has been visited and its left and
right child from the stack)

Works perfectly well but quite a long one, suggest some optimization for this r

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

see more

^ | ⌄ • Reply • Share ›

Akshay Jindal ➦ Akshay Jindal • 9 months ago

The above code is the for traversal. Here comes the main part i.e. calc
slight modification in the section starting from line 13

```
 if(p->visited==0)
    {
     while(p->left!=NULL)
     {
      p->visited=1;
      top=push(p);
      p=p->left;
     }
     p->visited=1;push(p);
     if(max<top)
       max=top;
}//close of if
```

• Reply • Share ›

Coder • 10 months ago

```
 public void HeightOfTree(struct node *root)
{
        struct Queue *Q = createQueue();
        int level = 1;

        if(!root)
          return;

        Enqueue(Q,root);
        Enqueue(Q,NULL);

        while(!IsEmpty(Q))
        {
                root = Dequeue(Q);

                // Indicates level completion.
                if(root == NULL)
                {
```

see more

^ | ⌄ • Reply • Share ›

noobie ➦ Coder • 10 months ago

level must be initiated with value 0 bcoz u r incrementing it after the co up displaying +1 levels.

^ | ⌄ • Reply • Share ›

kush • 11 months ago

```
 int height(tree *root)
{
```

```
int max=-1;
tree *arr[10000];int top=-1,hr[10000],h=0;
while(1)
{
        while(root)
        {
                ++top;
                arr[top]=root;
                root=root->left;
                hr[top]=++h;
        }
        tree *temp=arr[top];
        while(!(temp->right))
        {
                temp=arr[top];
                h=hr[top];
```

see more

^ | ⌄ • Reply • Share ›

Nitin Sharma ➦ kush • 6 months ago

I think your algorithm will go in infinite loop.....lets see this example
1

1->left =2
1->right=3
2->left=4
2->right=5

now your algorithm will go in infinite loop in switching from 2 to 5 and 5

^ | ⌄ • Reply • Share ›

```
int height(tree *root)
{
int max=-1;
tree *arr[10000];int top=-1,hr[10000],h=0;
while(1)
{
while(root)
{
++top;
arr[top]=root;
root=root->left;
hr[top]=++h;
}
tree *temp=arr[top];
while(!(temp->right))
{
temp=arr[top];
h=hr[top];
if(maxright;

}
return max;

}
```

⌃ | ⌄ • Reply • Share ›

AMIT • 11 months ago

If we just want to find height,we can do any other traversal like iterative inorder stack node,so with same time complexity,space complexity can be reduced tc

⌃ | ⌄ • Reply • Share ›

MANISH ➦ AMIT • 11 months ago

Hi Amit,
Isn't if you do iterative inorder traversal, then your time complexity will b

^ | ⌄ • Reply • Share ›

AMIT ➦ MANISH • 11 months ago

yes,time complexity of both level order traversal and inorder tra
but the space complexity of level order traversal is o(n) while in
consider it as a balanced Binary tree)

^ | ⌄ • Reply • Share ›

Nikhil Agrawal • 11 months ago

```
 public void iterativeHeight(Node root)
{
    int height=0;
    Node t=new Node(-1);
    if(root==null)
        System.out.println("Height="+height);

    Queue<Node> s=new LinkedList<>();
    s.add(root);
    s.add(t);

    while(!s.isEmpty())
    {
        Node tt=(Node) s.remove();

        if(tt.value==-1)
        {
            height++;
```

see more

• Reply • Share ›

Chandra Sekhar Nayak • 11 months ago

```
int heightItr(node *root).
{
if(root == NULL).

return 0;.

int level = 0;.
queue <node *> q;.
q.push(root);.
q.push(NULL);.

while (! q.empty() )
{.

root = q.front() ; q.pop();.

if(root).

{.
```

see more

⌃ | ⌄ • Reply • Share ›

Devarshi • 11 months ago

why dont we simply to the DFS.

⌃ | ⌄ • Reply • Share ›

Anon_001 ➦ Devarshi • 11 months ago

Because topic is to solve iteratively .

⌃ | ⌄ • Reply • Share ›

Shashank ➦ Anon_001 • 11 months ago

you mean dfs can't be implemented iteratively ?

FYI we can ;)

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

Reply • Share ›

AMIT ➦ Shashank • 11 months ago

Exactly..its better to perform iterative inorder or preorde
space complexity

Reply • Share ›

Devarshi ➦ Anon_001 • 11 months ago

ohh!!...thanks.

Reply • Share ›

Subscribe   Add Disqus to your site