Deepest left leaf node in a binary tree

Given a Binary Tree, find the deepest leaf node that is left child of its parent. For example, consider the following tree. The deepest left leaf node is the node with value 9.

```
       1
     /   \
    2     3
  /      /  \
 4      5    6
        \     \
         7     8
        /       \
       9         10
```

We strongly recommend you to minimize the browser and try this yourself first.

The idea is to recursively traverse the given binary tree and while traversing, maintain “level” which will store the current node’s level in the tree. If current node is left leaf, then check if its level is more than the level of deepest left leaf seen so far. If level is more then update the result. If current node is not leaf, then recursively find maximum depth in left and right subtrees, and return maximum of the two depths. Thanks to Coder011 for suggesting this approach.

```
// A C++ program to find the deepest left leaf in a given binary tree
#include <stdio.h>
#include <iostream>
using namespace std;

struct Node
{
```

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

```
    int val;
    struct Node *left, *right;
};

Node *newNode(int data)
{
    Node *temp = new Node;
    temp->val = data;
    temp->left = temp->right =  NULL;
    return temp;
}

// A utility function to find deepest leaf node.
// lvl:  level of current node.
// maxlvl: pointer to the deepest left leaf node found so far
// isLeft: A bool indicate that this node is left child of its parent
// resPtr: Pointer to the result
void deepestLeftLeafUtil(Node *root, int lvl, int *maxlvl,
                         bool isLeft, Node **resPtr)
{
    // Base case
    if (root == NULL)
        return;

    // Update result if this node is left leaf and its level is more
    // than the maxl level of the current result
    if (isLeft && !root->left && !root->right && lvl > *maxlvl)
    {
        *resPtr = root;
        *maxlvl = lvl;
        return;
    }

    // Recur for left and right subtrees
    deepestLeftLeafUtil(root->left, lvl+1, maxlvl, true, resPtr);
    deepestLeftLeafUtil(root->right, lvl+1, maxlvl, false, resPtr);
}

// A wrapper over deepestLeftLeafUtil().
Node* deepestLeftLeaf(Node *root)
{
    int maxlevel = 0;
    Node *result = NULL;
    deepestLeftLeafUtil(root, 0, &maxlevel, false, &result);
    return result;
}
```

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

```
// Driver program to test above function
int main()
{
    Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->right->left = newNode(5);
    root->right->right = newNode(6);
    root->right->left->right = newNode(7);
    root->right->right->right = newNode(8);
    root->right->left->right->left = newNode(9);
    root->right->right->right->right = newNode(10);

    Node *result = deepestLeftLeaf(root);
    if (result)
        cout << "The deepest left child is " << result->val;
    else
        cout << "There is no left leaf in the given tree";

    return 0;
}
```

Output:

```
The deepest left child is 9
```

Time Complexity: The function does a simple traversal of the tree, so the complexity is O(n).

This article is contributed by **Abhay Rathi**. 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

695 Subscribe

Recent Comments

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

Backtracking | Set 7 (Sudoku) · 27 minutes ago

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

Flipkart Interview | Set 6 · 47 minutes ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 47 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,

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

25 Tweet 3 2

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

24 Comments GeeksforGeeks

Sort by Newest

Join the discussion

HarryPotter • a day ago

Why are we checking (!root->left && !root->right) in (isLeft && !root->left && !
in deepestLeftLeafUtil

^ | ⌄ • Reply • Share ›

neelabhsingh • 2 months ago

@GeeksforGeeks. Suppose in the second last level Node value 8 has left chil
10. Please clearly it.

1 ^ | ⌄ • Reply • Share ›

codey.modey • 3 months ago

Do you know the biggest bug in your code is you didn't typedef your struct to N
functions

^ | ⌄ • Reply • Share ›

shrek • 3 months ago

Try this solution, this will give us the deepest node length

```
int length(Node node){
if(node==null)
return 0;
int a = length(node.leftChild)+1;
int b = length(node.rightChild)+1;
return a>b ? a:b;
}
```

^ | ⌄ • Reply • Share ›

amit • 4 months ago

what about traversing levelorder and returning first node of last level

^ | ⌄ • Reply • Share ›

in...

Find depth of the deepest odd level leaf node · 2 hours ago

Jonathan Chen ➦ amit • a month ago

The result has to be a left child. The first node of the last level could be

^ | ⌄ • Reply • Share ›

Anirudh Mathad ➦ amit • 3 months ago

if your tree ends like ...
1
2 3
10
10 , you will end up returning 10 and not 2

^ | ⌄ • Reply • Share ›

Aniket Thakur • 4 months ago

Java Code with output : http://opensourceforgeeks.blog...

1 ^ | ⌄ • Reply • Share ›

Guest • 5 months ago

package reva.geeksforgeeks.solution;

import reva.java.practice.TreeNode;

public class Deepest_Left_LeafNode {

static TreeNode result = null;

static int flevel = Integer.MIN_VALUE;

public static void main(String[] args) {

TreeNode root = new TreeNode(1);

root.leftchild = new TreeNode(2);

```
root.rightchild = new TreeNode(3);

root.leftchild.leftchild = new TreeNode(4);

root.leftchild.rightchild = new TreeNode(5);
```

see more

Reply • Share ›

Satyendra Kumar Singh • 5 months ago

Another simple solution....

```
void DeepestLeft(Node * root,int level,int *maxsofar,Node **leftnode)
{
if(root==NULL)
return;
if(*maxsofar < level && root->left)
{
*maxsofar=level;
*leftnode= root->left;
}
DeepestLeft(root->left,level+1,maxsofar,leftnode);
DeepestLeft(root->right,level+1,maxsofar,leftnode);
}
```

2 • Reply • Share ›

Diego Giagio • 5 months ago

C++ iterative version: http://ideone.com/Tc7YvU

1 • Reply • Share ›

Diego Giagio • 5 months ago

C++ recursive version: http://ideone.com/fbQcjj

Reply • Share ›

Setu • 5 months ago

Simple Java Implementaion

class DeepsestLeftNode{
static int maxLevel =0;
static Node node =null;

private static void printLeftUtil(Node root, boolean left,int level)
{

if(root==null)
return;

if(left & level>maxLevel){
maxLevel=level;
node=root;
}

printLeftUtil(root.left , true, level+1);
printLeftUtil(root.right, false, level+1);

see more

Reply • Share ›

pavansrinivas • 5 months ago

iterative soln in java (using level order)

Time :O(n)
Space :O(n)

```
`void deepestLeft(){
        int fin_ans = -1;
         Queue<node> Q = new LinkedList<>();
            Q.add(root);
            while (!Q.isEmpty())
            {
                temp = Q.remove();
                if (temp.left!=null){
                         Q.add(temp.left);
                         fin_ans = temp.left.i;
                }
               if (temp.right!=null){
                 Q.add(temp.right);
               }
               }
                System.out.print(fin_ans);
        }
```

1 ˄ | ˅ • Reply • Share ›

neelabhsingh ➦ pavansrinivas • 2 months ago

@pavansrinivas: Your solution will work fine but for 6 2 5 3 10 7 9 8 11 root) will return 12 which is same level as 8, 12 and 8 are both left chil comes before 12. So it should return 8 instead of 12.

˄ | ˅ • Reply • Share ›

pavansrinivas ➦ neelabhsingh • a month ago

got it..this might work..i am just using 2 variables which indicat for one time...

`void deepestLeft(){
Node temp = root;
int fin_ans = -1;

```
int pr = -1,cl = -1;
Queue<node> Q = new LinkedList<>();
Q.add(root);
Q.add(null);
while (!Q.isEmpty())
{
temp = Q.remove();

if(temp==null){

if(!Q.isEmpty()){
Q.add(null);
}
```

see more

^ | ⌄ • Reply • Share ›

Dixit • 5 months ago

To make the code work in java I defined 'maxLevel' and 'result' as static variat

^ | ⌄ • Reply • Share ›

samsammy ➦ Dixit • 5 months ago

No, you can use instance variable also.

```
package abc;

public class DeepestLeftLeaf {

Node node;
int maxHeight=0;
public static void main(String[] args) {
Node root = new Node(2);
root.left = new Node(1);
root.right = new Node(3);
root.left.left=new Node(4);
root.left.right=new Node(5);
root.left.right.left=new Node(6);
root.right.left=new Node(7);
root.right.left.right=new Node(8);
root.right.left.right.left=new Node(9);
DeepestLeftLeaf d= new DeepestLeftLeaf();
```

see more

^ | ˅ • Reply • Share ›

deep • 5 months ago

i think time complexity in o(n^2).Could you please clarify how time complexly is

^ | ˅ • Reply • Share ›

Kartik ➜ deep • 5 months ago

The time complexity seems to be O(n). It's a simple recursive tree trav
T(n-k-1) + C.

^ | ˅ • Reply • Share ›

acharyayogesh • 5 months ago

Another solution using iterative method :

Scan the tree level by level and if node has left child, push the left child data in
get the answer.

```
#include <iostream>
#include <stack>
#include <queue>

using namespace std;

class Node
{
```

```
{
public :
int data;
Node *left, *right;

Node (int d)
{
```

see more

2 • Reply • Share ›

Nitin Pallindrome → acharyayogesh • 5 months ago

simple solution that is what comes into my mind ...thanx yogesh

• Reply • Share ›

acharyayogesh → Nitin Pallindrome • 5 months ago

thanks for ur feedback.

I think there is no need for stack.
Just use a variable deep.

```
#include <iostream>
//#include <stack>
#include <queue>

using namespace std;

class Node
{
public :
int data;
Node *left, *right;

Node (int d)
```

```
{
```

see more

Reply • Share ›

kunal gupta acharyayogesh • 3 months ago

HI
I dont think there is any need for " bool isleft ",
left leg of a tree called first always.

Reply • Share ›

Subscribe   Add Disqus to your site