GeeksforGeeks

A computer science portal for geeks

Placements Practice GATE CS IDE Q&A
GeeksQuiz

Closest leaf to a given node in Binary Tree

Given a Binary Tree and a node **x** in it, find distance of the closest leaf to **x** in Binary Tree. If given node itself is a leaf, then distance is 0.

Examples:

```
Input: Root of below tree
       And x = pointer to node 13
          10
       /     \
     12       13
             /
           14
Output 1
Distance 1. Closest leaf is 14.


Input: Root of below tree
       And x = pointer to node 13
          10
       /     \
     12       13
           /     \
         14       15
        /   \     /  \
       21   22   23   24
       /\   /\   /\   /\
      1 2  3 4  5 6  7  8

Output 2
Closest leaf is 12 through 10.
```

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

The idea is to first traverse the subtree rooted with give node and find the closest leaf in this subtree. Store this distance. Now traverse tree starting from root. If given node x is in left subtree of root, then find the closest leaf in right subtree, else find the closest left in left subtree. Below is C++ implementation of this idea.

```
/* Find closest leaf to the given node x in a tree */
#include<bits/stdc++.h>
using namespace std;

// A Tree node
struct Node
{
    int key;
    struct Node* left, *right;
};

// Utility function to create a new node
Node* newNode(int key)
{
    Node* temp = new Node;
    temp->key = key;
    temp->left = temp->right = NULL;
    return (temp);
}

// This function finds closest leaf to root.  This distance
// is stored at *minDist.
void findLeafDown(Node *root, int lev, int *minDist)
{
    // base case
    if (root == NULL)
        return ;

    // If this is a leaf node, then check if it is closer
    // than the closest so far
    if (root->left == NULL && root->right == NULL)
    {
        if (lev < (*minDist))
            *minDist = lev;
        return;
    }

    // Recur for left and right subtrees
    findLeafDown(root->left, lev+1, minDist);
    findLeafDown(root->right, lev+1, minDist);
}

// This function finds if there is closer leaf to x through
// parent node.
int findThroughParent(Node * root, Node *x, int *minDist)
{
    // Base cases
    if (root == NULL) return -1;
    if (root == x) return 0;

    // Search x in left subtree of root
    int l = findThroughParent(root->left, x,  minDist);

    // If left subtree has x
    if (l != -1)
    {
        // Find closest leaf in right subtree
        findLeafDown(root->right, l+2, minDist);
        return l+1;
    }

    // Search x in right subtree of root
    int r = findThroughParent(root->right, x, minDist);
    // If right subtree has x
    if (r != -1)
    {
        // Find closest leaf in left subtree
        findLeafDown(root->left, r+2, minDist);
        return r+1;
    }

    return -1;
}

// Returns minimum distance of a leaf from given node x
int minimumDistance(Node *root, Node *x)
{
    // Initialize result (minimum distance from a leaf)
    int minDist = INT_MAX;

    // Find closest leaf down to x
    findLeafDown(x, 0, &minDist);

    // See if there is a closer leaf through parent
    findThroughParent(root, x, &minDist);

    return minDist;
}

// Driver program
int main ()
{
    // Let us create Binary Tree shown in above example
    Node *root  = newNode(1);
    root->left  = newNode(12);
    root->right = newNode(13);

    root->right->left   = newNode(14);
    root->right->right  = newNode(15);

    root->right->left->left   = newNode(21);
    root->right->left->right  = newNode(22);
    root->right->right->left  = newNode(23);
    root->right->right->right = newNode(24);

    root->right->left->left->left  = newNode(1);
    root->right->left->left->right = newNode(2);
    root->right->left->right->left  = newNode(3);
    root->right->left->right->right = newNode(4);
    root->right->right->left->left  = newNode(5);
    root->right->right->left->right = newNode(6);
    root->right->right->right->left = newNode(7);
    root->right->right->right->right = newNode(8);

    Node *x = root->right;

    cout << "The closest leaf to the node with value "
         << x->key << " is at a distance of "
         << minimumDistance(root, x) << endl;

    return 0;
}
```

Run on IDE

Output:

```
The closest leaf to the node with value 13 is at a distance of 2
```

Time Complexity of this above solution is O(n) as it does at most two traversals of given Binary Tree.

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

15 Comments Category: Trees

Related Posts:

- Lowest Common Ancestor in a Binary Tree | Set 2 (Using Parent Pointer)
- Construct a Binary Search Tree from given postorder
- BFS vs DFS for Binary Tree
- Maximum difference between node and its ancestor in Binary Tree
- Inorder Non-threaded Binary Tree Traversal without Recursion or Stack
- Check if leaf traversal of two Binary Trees is same?
- Locking and Unlocking of Resources arranged in the form of n-ary Tree
- Find all possible binary trees with given Inorder Traversal

(Login to Rate and Mark)

3.6 Average Difficulty : **3.6/5.0**
Based on **11** vote(s)

□ Add to TODO List

□ Mark as DONE

Like Share 34 people like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.