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;
};

/* Helper function for getLevel().  It returns level of the data if data is
   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);
    root->left->left = newNode(1);
    root->left->right = newNode(4);

    for (x = 1; x <=5; x++)
    {
      int level = getLevel(root, x);
      if (level)
        printf(" Level of %d is %d\n", x, getLevel(root, x));
      else
        printf(" %d is not present in tree \n", 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.

87 Comments Category: Trees

Related Posts:

- Check if removing an edge can divide a Binary Tree in two halves
- Check sum of Covered and Uncovered nodes of Binary Tree
- 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?

(Login to Rate and Mark)

1.7 Average Difficulty : **1.7/5.0**
Based on **11** vote(s)

Add to TODO List

Mark as DONE

Like Share 3 people like this.

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

@geeksforgeeks, Some rights reserved Contact Us! About Us! Advertise with us!