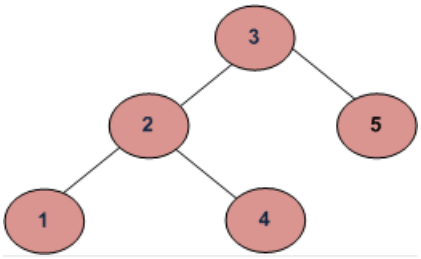
Get Level of a node in a Binary Tree

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.

C

```
#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;
}
```

Java

```
/* A tree node structure */
class Node
{
    int data;
    Node left, right;

    public Node(int d)
    {
        data = d;
        left = right = null;
    }
}

class BinaryTree
{

    Node root;

    /* Helper function for getLevel().  It returns level of the data
    if data is present in tree, otherwise returns 0.*/
    int getLevelUtil(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(Node node, int data)
    {
        return getLevelUtil(node, data, 1);
    }

    /* Driver function to test above functions */
    public static void main(String[] args)
    {
        BinaryTree tree = new BinaryTree();

        /* Constructing tree given in the above figure */
        tree.root = new Node(3);
        tree.root.left = new Node(2);
        tree.root.right = new Node(5);
        tree.root.left.left = new Node(1);
        tree.root.left.right = new Node(4);
        for (int x = 1; x <= 5; x++)
        {
            int level = tree.getLevel(tree.root, x);
            if (level != 0)
                System.out.println("Level of " + x + " is "
                        + tree.getLevel(tree.root, x));
            else
                System.out.println(x + " is not present in tree");
        }
    }
}

// This code has been contributed by Mayank Jaiswal(mayank_24)
```

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.