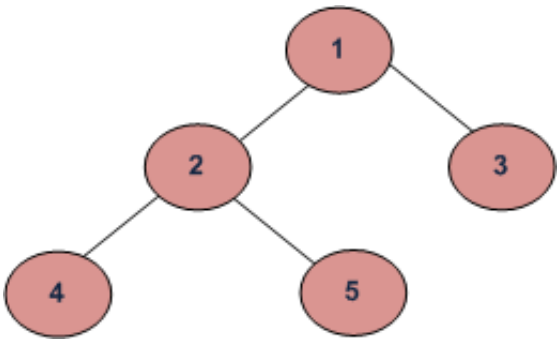
Write a Program to Find the Maximum Depth or Height of a Tree

Maximum depth or height of the below tree is 3.



Example Tree

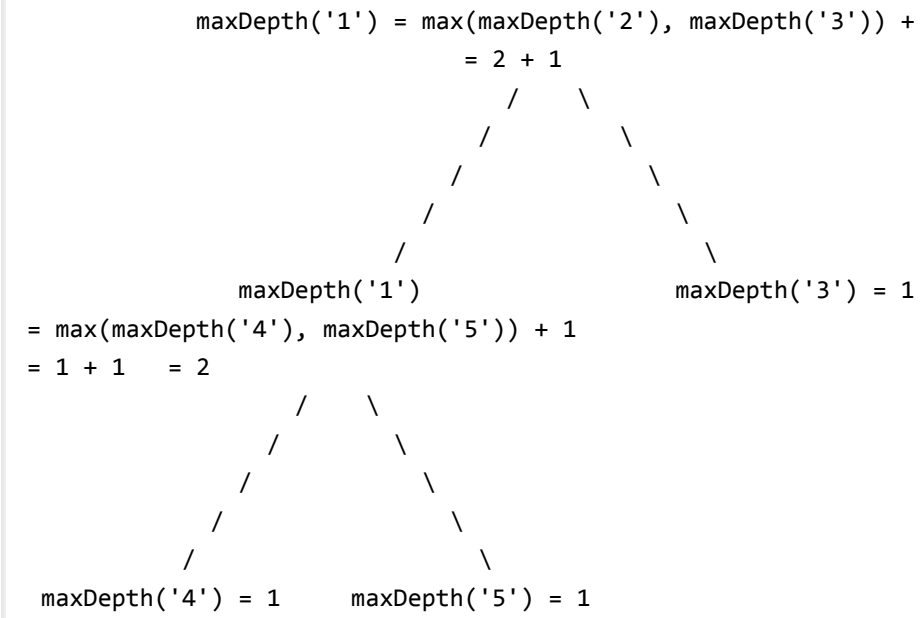
Recursively calculate height of left and right subtrees of a node and assign height to the node as max of the heights of two children plus 1. See below pseudo code and program for details.

Algorithm:

```
 maxDepth()
1. If tree is empty then return 0
2. Else
     (a) Get the max depth of left subtree recursively  i.e.,
          call maxDepth( tree->left-subtree)
     (a) Get the max depth of right subtree recursively  i.e.,
          call maxDepth( tree->right-subtree)
     (c) Get the max of max depths of left and right
          subtrees and add 1 to it for the current node.
         max_depth = max(max dept of left subtree,
                         max depth of right subtree)
                         + 1
     (d) Return max_depth
```

See the below diagram for more clarity about execution of the recursive function maxDepth() for above

example tree.



Implementation:

C

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


/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Compute the "maxDepth" of a tree -- the number of
    nodes along the longest path from the root node
    down to the farthest leaf node.*/
int maxDepth(struct node* node)
{
   if (node==NULL)
       return 0;
   else
   {
       /* compute the depth of each subtree */
       int lDepth = maxDepth(node->left);
       int rDepth = maxDepth(node->right);
```

```
       /* use the larger one */
       if (lDepth > rDepth)
           return(lDepth+1);
       else return(rDepth+1);
   }
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode(int data)
{
    struct node* node = (struct node*)
                                malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return(node);
}

int main()
{
    struct node *root = newNode(1);

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

    printf("Hight of tree is %d", maxDepth(root));

    getchar();
    return 0;
}
```

Java

```
// Java program to find height of tree

// A binary tree node
class Node {

    int data;
    Node left, right;

    Node(int item) {
        data = item;
        left = right = null;
    }
```

```
}

class BinaryTree {

    static Node root;

    /* Compute the "maxDepth" of a tree -- the number of
     nodes along the longest path from the root node
     down to the farthest leaf node.*/
    int maxDepth(Node node) {
        if (node == null) {
            return 0;
        } else {

            /* compute the depth of each subtree */
            int lDepth = maxDepth(node.left);
            int rDepth = maxDepth(node.right);

            /* use the larger one */
            if (lDepth > rDepth) {
                return (lDepth + 1);
            } else {
                return (rDepth + 1);
            }
        }
    }

    /* Driver program to test mirror() */
    public static void main(String[] args) {

        BinaryTree tree = new BinaryTree();

        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);

        System.out.println("Height of tree is : " + tree.maxDepth(root));


    }
}
```

Python

```
# Python program to find the maximum depth of tree
```

```
# A binary tree node
class Node:

    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# Compute the "maxDepth" of a tree -- the number of nodes
# along the longest path from the root node down to the
# farthest leaf node
def maxDepth(node):
    if node is None:
        return 0 ;

    else :

        # Compute the depth of each subtree
        lDepth = maxDepth(node.left)
        rDepth = maxDepth(node.right)

        # Use the larger one
        if (lDepth > rDepth):
            return lDepth+1
        else:
            return rDepth+1


# Driver program to test above function
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)


print "Height of tree is %d" %(maxDepth(root))

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Time Complexity: O(n) (Please see our post Tree Traversal for details)

References:

http://cslibrary.stanford.edu/110/BinaryTrees.html

159 Comments Category: Trees Tags: Height of a Tree, Tree Traveral, 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?

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