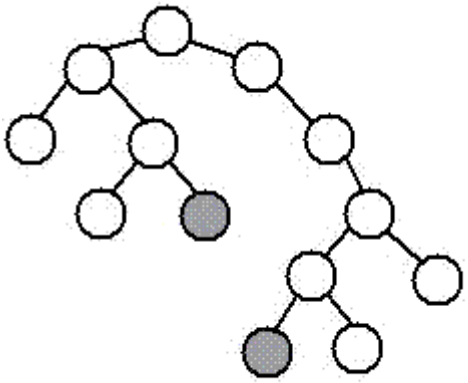
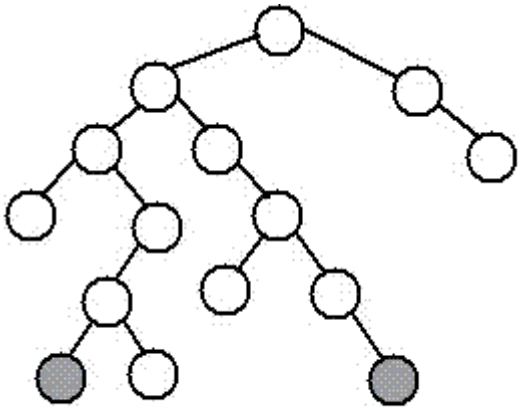
Diameter of a Binary Tree

The diameter of a tree (sometimes called the width) is the number of nodes on the longest path between two leaves in the tree. The diagram below shows two trees each with diameter nine, the leaves that form the ends of a longest path are shaded (note that there is more than one path in each tree of length nine, but no path longer than nine nodes).



diameter, 9 nodes, through root



diameter, 9 nodes, NOT through root

The diameter of a tree T is the largest of the following quantities:

* the diameter of T's left subtree

* the diameter of T's right subtree

* the longest path between leaves that goes through the root of T (this can be computed from the heights of the subtrees of T)

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, *right;
};

/* function to create a new node of tree and returns pointer */
struct node* newNode(int data);

/* returns max of two integers */
```

```
int max(int a, int b);

/* function to Compute height of a tree. */
int height(struct node* node);

/* Function to get diameter of a binary tree */
int diameter(struct node * tree)
{
   /* base case where tree is empty */
   if (tree == NULL)
     return 0;

  /* get the height of left and right sub-trees */
  int lheight = height(tree->left);
  int rheight = height(tree->right);

  /* get the diameter of left and right sub-trees */
  int ldiameter = diameter(tree->left);
  int rdiameter = diameter(tree->right);

  /* Return max of following three
   1) Diameter of left subtree
   2) Diameter of right subtree
   3) Height of left subtree + height of right subtree + 1 */
  return max(lheight + rheight + 1, max(ldiameter, rdiameter));
}

/* UTILITY FUNCTIONS TO TEST diameter() FUNCTION */

/*  The function Compute the "height" of a tree. Height is the
    number f nodes along the longest path from the root node
    down to the farthest leaf node.*/
int height(struct node* node)
{
   /* base case tree is empty */
   if(node == NULL)
       return 0;

   /* If tree is not empty then height = 1 + max of left
      height and right heights */
   return 1 + max(height(node->left), height(node->right));
}

/* 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);
}

/* returns maximum of two integers */
int max(int a, int b)
{
  return (a >= b)? a: b;
}

/* Driver program to test above functions*/
int main()
{

  /* Constructed binary tree is
            1
          /   \
        2      3
      /  \
    4     5
  */
  struct node *root = newNode(1);
```

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

  printf("Diameter of the given binary tree is %d\n", diameter(root));

  getchar();
  return 0;
}
```

Run on IDE

Java

```
// Recursive optimized Java program to find the diameter of a
// Binary Tree

/* Class containing left and right child of current
 node and key value*/
class Node
{
    int data;
    Node left, right;

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

/* Class to print the Diameter */
class BinaryTree
{
    Node root;

    /* Method to calculate the diameter and return it to main */
    int diameter(Node root)
    {
        /* base case if tree is empty */
        if (root == null)
            return 0;

        /* get the height of left and right sub trees */
        int lheight = height(root.left);
        int rheight = height(root.right);

        /* get the diameter of left and right subtrees */
        int ldiameter = diameter(root.left);
        int rdiameter = diameter(root.right);

        /* Return max of following three
          1) Diameter of left subtree
         2) Diameter of right subtree
         3) Height of left subtree + height of right subtree + 1 */
        return Math.max(lheight + rheight + 1,
                        Math.max(ldiameter, rdiameter));

    }

    /* A wrapper over diameter(Node root) */
    int diameter()
    {
        return diameter(root);
    }

    /*The function Compute the "height" of a tree. Height is the
      number f nodes along the longest path from the root node
      down to the farthest leaf node.*/
    static int height(Node node)
```

```
    {
        /* base case tree is empty */
        if (node == null)
            return 0;

        /* If tree is not empty then height = 1 + max of left
           height and right heights */
        return (1 + Math.max(height(node.left), height(node.right)));
    }

    public static void main(String args[])
    {
        /* creating a binary tree and entering the nodes */
        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("The diameter of given binary tree is : "
                           + tree.diameter());
    }
}
```

Run on IDE

Python

```
# Python program to find the diameter of binary 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


"""
The function Compute the "height" of a tree. Height is the
number f nodes along the longest path from the root node
down to the farthest leaf node.
"""
def height(node):

    # Base Case : Tree is empty
    if node is None:
        return 0 ;

    # If tree is not empty then height = 1 + max of left
    # height and right heights
    return 1 + max(height(node.left) ,height(node.right))

# Function to get the diamtere of a binary tree
def diameter(root):

    # Base Case when tree is empty
    if root is None:
        return 0;

    # Get the height of left and right sub-trees
    lheight = height(root.left)
    rheight = height(root.right)

    # Get the diameter of left and irgh sub-trees
    ldiameter = diameter(root.left)
    rdiameter = diameter(root.right)
```

```
    # Return max of the following tree:
    # 1) Diameter of left subtree
    # 2) Diameter of right subtree
    # 3) Height of left subtree + height of right subtree +1
    return max(lheight + rheight + 1, max(ldiameter, rdiameter))


# Driver program to test above functions

"""
Constructed binary tree is
            1
          /   \
        2      3
      /  \
    4     5
"""

root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
print "Diameter of given binary tree is %d" %(diameter(root))

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

Run on IDE

Time Complexity: O(n^2)

Output:

```
Diameter of the given binary tree is 4
```

Optimized implementation: The above implementation can be optimized by calculating the height in the same recursion rather than calling a height() separately. Thanks to Amar for suggesting this optimized version. This optimization reduces time complexity to O(n).

C

```
/*The second parameter is to store the height of tree.
   Initially, we need to pass a pointer to a location with value
   as 0. So, function should be used as follows:

   int height = 0;
   struct node *root = SomeFunctionToMakeTree();
   int diameter = diameterOpt(root, &height); */
int diameterOpt(struct node *root, int* height)
{
  /* lh --> Height of left subtree
     rh --> Height of right subtree */
  int lh = 0, rh = 0;

  /* ldiameter  --> diameter of left subtree
     rdiameter  --> Diameter of right subtree */
  int ldiameter = 0, rdiameter = 0;

  if(root == NULL)
  {
```

```
    *height = 0;
     return 0; /* diameter is also 0 */
  }

  /* Get the heights of left and right subtrees in lh and rh
    And store the returned values in ldiameter and ldiameter */
  ldiameter = diameterOpt(root->left, &lh);
  rdiameter = diameterOpt(root->right, &rh);

  /* Height of current node is max of heights of left and
     right subtrees plus 1*/
  *height = max(lh, rh) + 1;

  return max(lh + rh + 1, max(ldiameter, rdiameter));
}
```

Run on IDE

Java

```
// Recursive Java program to find the diameter of a
// Binary Tree

/* Class containing left and right child of current
 node and key value*/
class Node
{
    int data;
    Node left, right;

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

// A utility class to pass heigh object
class Height
{
    int h;
}

/* Class to print the Diameter */
class BinaryTree
{
    Node root;

    /* define height =0 globally and  call diameterOpt(root,height)
       from main */
    int diameterOpt(Node root, Height height)
    {
        /* lh --> Height of left subtree
           rh --> Height of right subtree */
        Height lh = new Height(), rh = new Height();

        if (root == null)
        {
            height.h = 0;
            return 0; /* diameter is also 0 */
        }

        /* ldiameter  --> diameter of left subtree
           rdiameter  --> Diameter of right subtree */
        /* Get the heights of left and right subtrees in lh and rh
         And store the returned values in ldiameter and ldiameter */
        lh.h++;     rh.h++;
        int ldiameter = diameterOpt(root.left, lh);
        int rdiameter = diameterOpt(root.right, rh);

        /* Height of current node is max of heights of left and
```

```
         right subtrees plus 1*/
        height.h = Math.max(lh.h, rh.h) + 1;

        return Math.max(lh.h + rh.h + 1, Math.max(ldiameter, rdiameter));
    }

    /* A wrapper over diameter(Node root) */
    int diameter()
    {
        Height height = new Height();
        return diameterOpt(root, height);
    }

    /*The function Compute the "height" of a tree. Height is the
      number f nodes along the longest path from the root node
      down to the farthest leaf node.*/
    static int height(Node node)
    {
        /* base case tree is empty */
        if (node == null)
            return 0;

        /* If tree is not empty then height = 1 + max of left
           height and right heights */
        return (1 + Math.max(height(node.left), height(node.right)));
    }

    public static void main(String args[])
    {
        /* creating a binary tree and entering the nodes */
        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("The diameter of given binary tree is : "
                           + tree.diameter());
    }
}
```

Run on IDE

Time Complexity: O(n)
Output:

```
4
```

Diameter of an N-ary tree

References:
http://www.cs.duke.edu/courses/spring00/cps100/assign/trees/diameter.html

Please write comments if you find any of the above codes/algorithms incorrect, or find other ways to solve the same problem.

GATE CS Corner Company Wise Coding Practice

Trees

Recommended Posts:

How to determine if a binary tree is height-balanced?
Inorder Tree Traversal without Recursion
Level Order Tree Traversal
Write a Program to Find the Maximum Depth or Height of a Tree
Maximum width of a binary tree

(Login to Rate and Mark)

3.2 Average Difficulty : **3.2/5.0**
Based on **249** vote(s)

Add to TODO List
Mark as DONE

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

Share this post!