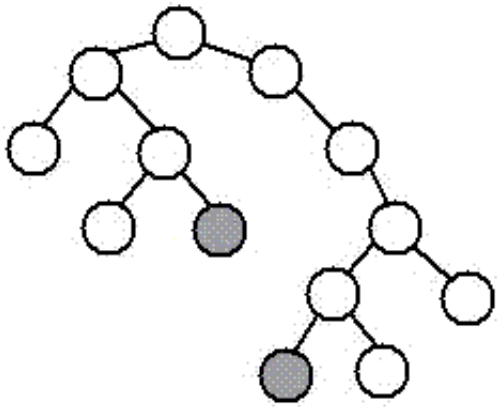
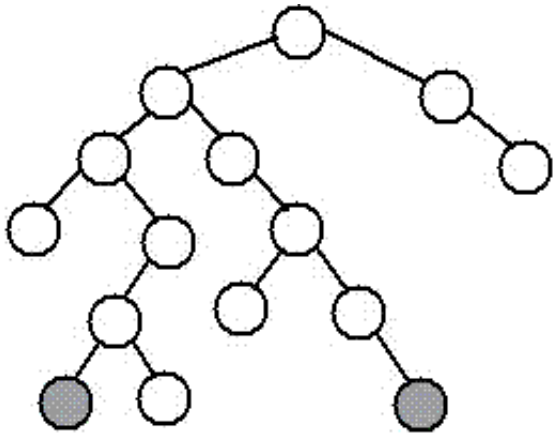
Diameter of a Binary Tree

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:

```
#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;
};

/* 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 == 0)
     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

Time Complexity: O(n^2)

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).

```
/*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

Time Complexity: O(n)

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.

263 Comments Category: Trees

Related Posts:

- Find Count of Single Valued Subtrees
- Check if a given array can represent Preorder Traversal of Binary Search Tree
- Mirror of n-ary Tree
- Succinct Encoding of Binary Tree
- Construct Binary Tree from given Parent Array representation
- Symmetric Tree (Mirror Image of itself)
- Find Minimum Depth of a Binary Tree
- Maximum Path Sum in a Binary Tree

3.5 Average Difficulty : **3.5/5.0**
Based on **2** vote(s)

(Login to Rate)

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

Manoj • a month ago

@GeeksforGeeks Can we proceed like doing a dfs and updating two variables highest_depth and second_highest_depth of nodes then finally adding these two variables

• Reply • Share ›

Ramendu Shandilya • a month ago

Sweet and short :)

```
public int findDiameter(BSTNode root){
                if(root == null)
return 0;
                int lh = findDiameter(root.getLeftChild());
                int rh = findDiameter(root.getRightChild());
                int dia = lh+rh+1;
                if(oldd < dia)
                {
                        oldd = dia;
                }
                return (lh > rh ? lh+1 : rh+1);
        }
```

PS: oldd is a global variable

• Reply • Share ›

Flower girl • 2 months ago

cant get the optimized solution..

• Reply • Share ›

Holden • 2 months ago

I have written the 2nd method in Java, but it returns incorrect result.
http://ideone.com/6Ba0me

Does anybody know what is wrong? It returns '1', instead of '4', which is the diameter.

• Reply • Share ›

Ashutosh Sharma ➚ Holden • 12 days ago

You are passing height as value, not as a reference which is causing the problem. In this code, the height is passed through as a pointer variable.

• Reply • Share ›

Satish Kumar • 3 months ago

simplest code ever

http://code.geeksforgeeks.org/...

4 ▲ | ▼ • Reply • Share ›

Sachin Bansal ➦ Satish Kumar • 2 months ago

really cool thanks....man

▲ | ▼ • Reply • Share ›

Praveen Yadav • 3 months ago

Another short solution with use of reference:

```
#include <iostream>

using namespace std;

typedef struct node{

int data;

node *left;

node *right;

}node;

node *newNode(int new_data)

{

node *new_node=new node;
```

see more

▲ | ▼ • Reply • Share ›

codemonk • 3 months ago

how to print nodes on this path of diameter.....

3 ▲ | ▼ • Reply • Share ›

Siddharth Gupta • 3 months ago

Not sure if this is entirely correct. Consider a case where only two nodes exists -> in that case there can be only one leaf node and hence the diameter should come out to be 0. In the code above its coming out to be 2.

▲ | ▼ • Reply • Share ›

grokbrokkoli ➦ Siddharth Gupta • 3 months ago

It is because they defined diameter incorrectly. The correct definition is "A longest path or route between any two nodes in a tree". With such a definition, the answer is 2

is 2.

1 ▲ | ▼ • Reply • Share ›

LedZep • 3 months ago

Their program is broken. It also prints wrong output in case of following tree:

```
struct node *root = newNode(1);
root->right = newNode(3);
root->right->left = newNode(4);
root->right->left->left = newNode(5);
root->right->left->right = newNode(6);
```

▲ | ▼ • Reply • Share ›

Shikha Gupta ➦ LedZep • 3 months ago

Yes u r right output comes as diameter is 4 whereas it should be 3.

▲ | ▼ • Reply • Share ›

Hari ➦ Shikha Gupta • 3 months ago

the reason for that is leaf node has a height of 1 instead of 0

▲ | ▼ • Reply • Share ›

Divyansh Agarwal • 3 months ago

Also, shouldn't the height function return -1 for when root is NULL?

▲ | ▼ • Reply • Share ›

Shubham Mohanka ➦ Divyansh Agarwal • 3 months ago

actually here height represents number of nodes in the path form a node to the leaf and not number of edges.Hence it should return 0.Hope this clears your doubt

▲ | ▼ • Reply • Share ›

Divyansh Agarwal • 3 months ago

If the question is to find the path having the maximum no of nodes in a binary tree, will it always be the diameter of the tree? I remember seeing a question like that

▲ | ▼ • Reply • Share ›

Abhishek Agarwal ➦ Divyansh Agarwal • 3 months ago

That is the very definition of diameter.

1 ▲ | ▼ • Reply • Share ›

rishi • 3 months ago

```
# include <stdlib.h>

# include<iostream>

using namespace std;
```

```
using namespace std;

typedef struct tree {int data;struct tree *left,*right;};

tree* newNode(int data){

tree* node = new struct tree;

node->data = data;

node->left = NULL;

node->right = NULL;

return(node);

}
```

see more

• Reply • Share ›

holdnet • 3 months ago

simply use this:

```
int diameter(node* tree,int* max1) {

if(tree==NULL)

return 0;

int l=diameter(tree->left,max1);

int r=diameter(tree->right,max1);

if(l+r+1>=*max1)

*max1=l+r+1;

return max(l,r)+1;

}
```

• Reply • Share ›

Paras • 3 months ago

http://ideone.com/VqEW4A Simpler Optimized Solution

http://ideone.com/vaFW4A Simpler Optimized Solution

• Reply • Share ›

roboT • 4 months ago

what to do when a root node has more than two child nodes. how the address gets accessed

• Reply • Share ›

VIJAY • 4 months ago

we can also do inverse, i.e we can keep a pointer to max diameter and return height from the function.

• Reply • Share ›

Chirag Bansal • 4 months ago

what would be the answer if tree has only a root and left child..!!the solution shows 2.??

• Reply • Share ›

fateh Chirag Bansal • 4 months ago

this program will fail according to your condition . by definition diameter is number of nodes between 2 leaves . So a tree having only root and left child must show diameter zero.
please confirm

• Reply • Share ›

Chirag Bansal fateh • 4 months ago

yes....thats wat i wanted to point out..here!! :)

• Reply • Share ›

fateh Chirag Bansal • 4 months ago

https://ideone.com/9z6QIP
SEE THIS i changed some logic and it work wonders

• Reply • Share ›

prantik • 4 months ago

A simple O(n) solution

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

struct node{
int data;
struct node *left;
struct node *right;
};
```

```
int diameter(struct node *root,int *maxlength)
{
if(root==NULL)
return 0;
int left=0;
int right=0;
left = diameter(root->left,maxlength);
right=diameter(root->right,maxlength);
```

see more

• Reply • Share ›

Aniket Ghag • 4 months ago

It has been mentioned that "The diameter of a tree (sometimes called the width) is the number of nodes on the longest path between TWO LEAVES in the tree".
According to that,
1. If a tree has only head node (so only one leaf as well), it should give ans 0. But above code is giving 1.
2. If a tree has head and only left child (no right child), it should give ans 0 as only one leaf node. But above code is giving 2.
Can anyone please explain?? Or is there a need to adjust the problem statement?

1 • Reply • Share ›

Milind Gokhale ➚ Aniket Ghag • 4 months ago

I think for 1. When there is only one node (tree with only 1 node) that node itself becomes the leaf node so the answer for the path from leaf node to itself will be 1. So 1 is correct.
2. I think the answer here as well is correct because since there is no other leaf in the tree, the answer of the diameter becomes identical to the height of the tree.

• Reply • Share ›

gsd • 4 months ago

why we need extra diameter function? when we can simply do this ques similarly by getting height of left and right subtrees and taking 1+height(left tree)+height(right tree).
what is need of
// max(lheight + rheight + 1, max(ldiameter, rdiameter));

return lheight+rheight+1 // only this works fine

plz check https://ideone.com/9WSMxr
correct me if wrong

• Reply • Share ›

blank space ➚ gsd • 4 months ago

if you go by using this approach. this may give wrong answers in some cases. Just see the photo 2 above. this approach will give answer 8 but the correct answer is 9 and that is not through the main root of the tree, so you have to consider the left and right height of each subtree which is what is being done by diameter function

• Reply • Share ›

Karan Kapoor • 4 months ago

We can do this by double BFS as well.. That will be O(n) as well..

1 • Reply • Share ›

garima Choudhary • 5 months ago

int ldiameter = diameter(tree->left);
int rdiameter = diameter(tree->right)
Can anyone explain these lines..hiw they are getting diameter ??

• Reply • Share ›

Lehar ➚ garima Choudhary • 5 months ago

Diameter is a recursive function. So ldia and rdia are being calculated recursively.

• Reply • Share ›

Rohit Sinha • 5 months ago

For full tree it is giving 1 more than the actual answer please help.

• Reply • Share ›

Hitesh Saini ➚ Rohit Sinha • 5 months ago

no it is giving the right answer for full tree also.
share your code if doubt is still not clear.

• Reply • Share ›

Arpit Kashyap • 5 months ago

it doing coding in java..in place of pass by reference *diameter can use static diameter

1 • Reply • Share ›

Utsav Vyas • 5 months ago

Is solution given by geeksforgeeks is correct?
because this is given wrong answer in
1
2
3
5
4 6
7
8

• Reply • Share ›

Shivani • 5 months ago

Is it right ?

```
int diameter(struct btree * root, int *d)
{
if(!root)
return 0;
int l,r;
l=diameter(root->left,d);
r=diameter(root->right,d);

if((l+r)>*d)
*d=l+r;

if(l>r)
return 1+l;
else
return 1+r;
}
```

• Reply • Share ›

fateh ➚ Shivani • 4 months ago

instead of *d=l+r;
it should be *d=l +r +1;
and finally extract d and print on console rather than value returned by diameter function

• Reply • Share ›

fateh ➚ fateh • 4 months ago

https://ideone.com/9z6QIP check this

• Reply • Share ›

Eknoor • 5 months ago

```
int diameter2(node *r, int &d)

{

if(!r)

return 0;

int lheight = diameter2(r->lchild,d);
```

```
int rheight = diameter2(r->rchild,d);

d = max(rheight+lheight+1,d);

return max(lheight,rheight)+1;

}
```

• Reply • Share ›

Ananda kumar N • 5 months ago

Is diameter path should goes root or not for Binary tree?

• Reply • Share ›

anurag das • 5 months ago

i think another good solution would be running a DFS from root to find node which is farthest located and then from that node again run DFS to search node which is farthest located from this node which would give two end nodes with farthest distance . And hence the diameter.

1 • Reply • Share ›

uchiha_itachi anurag das • a month ago

So, if you are treating the binary tree as graph then only you can do a BFS from the farthest node right ?

• Reply • Share ›

Gautham Kumaran anurag das • 5 months ago

how do you run DFS from the farthest node in a normal binary tree with left and right pointers, and no parent pointer?

am I missing something obvious?

• Reply • Share ›

anurag das Gautham Kumaran • 4 months ago

Since when do you need parent pointer for running a DFS. I am giving my solution with BFS as a reply to @hitesh_saini . Please see that and if you have more trouble then u are free to ask.

• Reply • Share ›

Think!!! anurag das • 5 months ago

So, how would you handle the case when diameter doesn't include the root?
In that case, you have to do DFS from each node (n x n) !

• Reply • Share ›

anurag das Think!!! • 5 months ago

actually this part already gets included . U start from root to get the farthest

point right . This is the only job of root . Now from this point u can search for farthest point which may or may not include root . So it doesnt matter whether diameter has the root or not . For better understanding there is spoj question for this -
http://www.spoj.com/problems/P...
try this for better understanding .
its O(n) solution.

1 • Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy