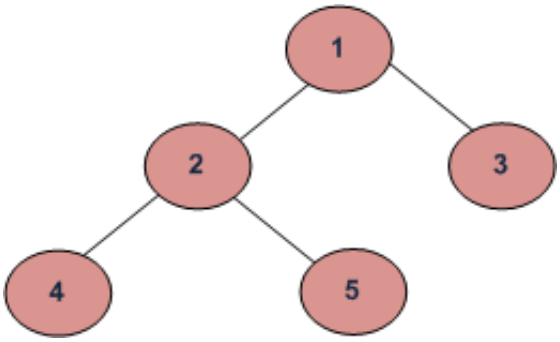
Program to count leaf nodes in a binary tree

A node is a leaf node if both left and right child nodes of it are NULL.

Here is an algorithm to get the leaf node count.

```
getLeafCount(node)
1) If node is NULL then return 0.
2) Else If left and right child nodes are NULL return 1.
3) Else recursively calculate leaf count of the tree using below formula.
    Leaf count of a tree = Leaf count of left subtree +
                               Leaf count of right subtree
```



Example Tree

Leaf count for the above tree is 3.

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;
};
```

```c
/* Function to get the count of leaf nodes in a binary tree*/
unsigned int getLeafCount(struct node* node)
{
  if(node == NULL)
    return 0;
  if(node->left == NULL && node->right==NULL)
    return 1;
  else
    return getLeafCount(node->left)+
           getLeafCount(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);
}

/*Driver program to test above functions*/
int main()
{
  /*create a tree*/
  struct node *root = newNode(1);
  root->left        = newNode(2);
  root->right       = newNode(3);
  root->left->left  = newNode(4);
  root->left->right = newNode(5);

  /*get leaf count of the above created tree*/
  printf("Leaf count of the tree is %d", getLeafCount(root));

  getchar();
  return 0;
}
```

Run on IDE

Time & Space Complexities: Since this program is similar to traversal of tree, time and space complexities will be same as Tree traversal (Please see our Tree Traversal post for details)

Please write comments if you find any bug in the above programs/algorithms or other ways to solve the same problem.

81 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

2 Average Difficulty : **2/5.0**
Based on **1** vote(s)

(Login to Rate)

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

81 Comments GeeksforGeeks 1 Login

Recommend 1 Share Sort by Newest

Join the discussion…

Himanshu Mishra • 2 months ago

```
int noOfLeafNode(struct node *node) {
        if(node == NULL)
                return 0;
        else {
                if(node->left == NULL && node->right == NULL)
                        return 1;
                return (noOfLeafNode(node->left) + noOfLeafNode(node->right));
        }
}
```

1 • Reply • Share ›

arun kumar • 2 months ago

int c = 0;

void LeafNodeCount(BSTNode* root)

{

if(root == Null)

```
return ;

else

{

LeafNodeCount(root->left);

LeafNodeCount(root->right);

if(root->left == Null && root->right == Null)

c++;

}

}
```

• Reply • Share ›

Sathiyaseelan • 4 months ago

Iterative way of calculating the same.

```
//Using the fact " # of leaf nodes = #internal nodes +1 "
i = 0
queue<node> nodes;
nodes.insert(root);
while(!nodes.empty()){
node u = nodes.front();
if(u->left && u->right){
i++;
nodes.insert(u->left);
nodes.insert(u->right);
}
}
return i+1;
```

• Reply • Share ›

Holden ➦ Sathiyaseelan • 3 months ago

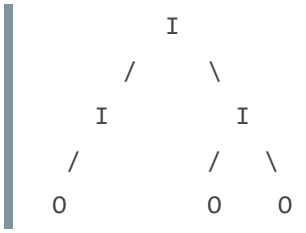
You code is correct, but this formula is not correct:

> `" # of leaf nodes = #internal nodes +1 "`

the correct definition is:

> `" # of leaf nodes = #nodes with 2 children + 1 "`

check in this tree:



It has 3 internal nodes and 3 leaves :)

• Reply • Share ›

sanchi ➚ Holden • 3 months ago

@Holden root will not be counted as an internal node.

• Reply • Share ›

Holden ➚ sanchi • 3 months ago

I think you are not correct. The correct definition for internal node is:
Internal node = every non-leaf node
(root is also internal, when it is not a leaf)

• Reply • Share ›

Sathiyaseelan ➚ Holden • 3 months ago

@Holden Internal node for K-ary tree is the node that has K childrens except root.
So for binary tree, Internal node means nodes that are having two childrens except root.

• Reply • Share ›

Jaguar • 4 months ago

Do an inorder / preorder / postorder traversal and if we encounter any node, increase a static variable count.

code : http://pastebin.com/Z9fvp0Hr

• Reply • Share ›

dude • 4 months ago

binary tree animation here

http://animatedarena.com/Jex/f...

• Reply • Share ›

Holden ➚ dude • 3 months ago

It needs username and password

• Reply • Share ›

Ashish Singh • 5 months ago

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

• Reply • Share ›

blank space • 5 months ago

https://ideone.com/gdkAK5

• Reply • Share ›

hey1234 • 5 months ago

sorry ceiling not floor

• Reply • Share ›

hey1234 • 5 months ago

I think that number of leaf nodes can be calculated easily .They are from floor(count of nodes/2) to n.

• Reply • Share ›

chomu chaturvedanta(chakuwala) • 5 months ago

pls tell guys and GALS.....its an emergency...my interview is just in 5 mins plzzzzzzz help guys and GALS

• Reply • Share ›

chomu chaturvedanta(chakuwala) • 5 months ago

will this also work.....

unsigned int getLeafCount(struct node* node)

{

if(node == NULL)

return 1;

else

return getLeafCount(node->left)+

getLeafCount(node->right);

}

and our ans will be.....printf("Leaf count of the tree is %d", getLeafCount(root)/2);

• Reply • Share ›

Jabir Ansari chomu chaturvedanta(chakuwala) • 5 months ago

1
/ \
2 3
\ \
4 5
in above example there is only two leaf node but you function return (6/2)= 3 which is wrong

• Reply • Share ›

chomu chaturvedanta(chakuwala) Jabir Ansari • 5 months ago
thank you Jabir Sir.....

• Reply • Share ›

Pankaj Kushwaha • 5 months ago
we can use global variable also:

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node* left;

struct node* right;

};

int max = 0;

void getleafcount(struct node* root)
```

see more

• Reply • Share ›

Rajnish Kr Jha • 5 months ago
simple solution!!

http://ideone.com/OiR3WM

• Reply • Share ›

shrag • 6 months ago
why the function contains unsigned int? any special need of using unsigned?

• Reply • Share ›

Anup Rai shrag • 5 months ago

There is no need but the count will always be greater then 0 so using unsigned is convention that is used to show numbers like counts, memory locations showing that these numbers will always be non -ve.

• Reply • Share ›

Utkarsh Mishra • 9 months ago

```
int countleaf(node* root)
{
int left,right,sum;
if(root==NULL)
return 0;
else
{
left=countleaf(root->llink);
right=countleaf(root->rlink);
sum =left+right;
if(sum==0)
return sum+1;
else return sum;
}
}
```

• Reply • Share ›

Rishi Verma • 10 months ago

Java Code.

http://ideone.com/QTLq7r

• Reply • Share ›

Praveen Boodagoli • a year ago

Java code to find the number of leaf nodes, O(N) complexity

```
public int getCountOfLeafNodes() {
return getCountOfLeafNodes(root);
}

private int getCountOfLeafNodes(TreeNode root) {
if (root == null)
return 0;
if (root.left == null && root.right == null) {
return 1;
}
```

```
return getCountOfLeafNodes(root.left) + getCountOfLeafNodes(root.right);
}
```

2 • Reply • Share ›

abc • a year ago

is this fine ??

```
int count(node *root){
static int c=0;
if(root==NULL){
return 0;
}

count(root->left);
count(root->right);

if(root->left==NULL&&root->right==NULL){
c++;

}
return c;

}
```

15 • Reply • Share ›

Holden abc • 3 months ago

It is not working!

```
public static int countLeaf2(Node node){
        if(node == null){
                return 0;
        }
        int count = 0;
        countLeaf2(node.left);
        countLeaf2(node.right);
        if(node.left == null && node.right == null){
                count++;
        }
        return count;
}
```

• Reply • Share ›

Ashish Jaiswal abc • 8 months ago

Avoid static variable...either make it global or try the above approach...

1 • Reply • Share ›

Atul Maini ➚ Ashish Jaiswal • 7 months ago
why static variables should not be used? any specific reason?
• Reply • Share ›

Ashish Jaiswal ➚ Atul Maini • 7 months ago
because it eats up unnecessary space.
• Reply • Share ›

thevagabond85 ➚ Ashish Jaiswal • 8 months ago
even global vars are discouraged. Best is to pass by reference.
1 • Reply • Share ›

Neha ➚ abc • 9 months ago
Using your logic, why isnt this working? https://ideone.com/AsI4Jm
• Reply • Share ›

Pankaj Kushwaha ➚ Neha • 5 months ago
its working fine , your tree is 1,2,3 ,4,5, 10 ,15 (you have 2 extra node compare to Q posted) , leaf count is 4 , which is fine..
• Reply • Share ›

anonymous ➚ Neha • 9 months ago
ur code is bit confusing.. why u did this
if(size1==0)
{
printf("yes");
}
else
printf("no");
by the way above code is correct
• Reply • Share ›

Sukrit Kumar Shanu ➚ abc • 10 months ago
absolutely
• Reply • Share ›

coder.girl ➚ abc • a year ago
i wrote the same :P
1 • Reply • Share ›

Anon ➚ abc • a year ago
Looks Good

• Reply • Share ›

Deepesh Panjabi • a year ago

http://ideone.com/pdjk1Y

• Reply • Share ›

groomnestle • 2 years ago

Traverse the tree in any order (in-order, pre-order or post-order) and count++ for nodes with no children.

2 • Reply • Share ›

munai • 2 years ago

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int data;
struct node *left,*right;
};
typedef struct node node;
struct list
{
node *a;
struct list *next;
};
typedef struct list list;
list *head,*cur_node,*prev_node;
node *NewNode(int val)
{
node *temp=(node *)malloc(sizeof(node));
```

see more

4 • Reply • Share ›

Nikhil Agrawal • 2 years ago

Below code is simple iterative version for finding number of leaf nodes using queue. The key concept is add all element of a particular level and then add null to queue. For finding number of leaf nodes just count number nodes between last null in the queue and second last null in the queue.

[sourcecode language="Java" 1="void" 2="numberofleafs(Node" 3="root)" 4="{" 5="int" max="value;" 6="int" value="0;" 7="if(root==null)" 8="{" 9="System.out.println("Number" 10="of" 11="leaf" 12="nodes=0");" 13="return;" 14="}" 15="else" 16="{" 17="Node" temp="null;" 18="Queue<Node>" q="new" 19="LinkedList<>();" 20="q.add(root);"

21="q.add(temp);" 22="while(!q.isEmpty())" 23="{" 24="Node" t="q.remove();" 25="if(t!=null)" 26="{" 27="while(t!=null)" 28="{" 29="value++;" 30="if(t.left!=null)" 31="q.add(t.left);" 32="if(t.right!=null)" 33="q.add(t.right);" 34="}" 35="q.add(temp);" 36="if(value>max)" 37="}" 38="else" 39="if(t==null)" 40="{" 41="continue;" 42="}" 43="}" 44="System.out.println("Number" 45="of" 46="leaf" 47="Nodes="+max);" 48="}" 49="}" 50="[/sourcecode"]

▲ | ▼ • Reply • Share ›

Nikhil Agrawal ➦ Nikhil Agrawal • 2 years ago

```
 public void numberofleafs(Node root)
{
      int max=-1;
        int value=0;
    if(root==null)
    {
        System.out.println("Number of leaf nodes=0");
        return;
    }
    else
    {
        Node temp=null;
        Queue<Node> q=new LinkedList<>();
        q.add(root);
        q.add(temp);

        while(!q.isEmpty())
        {
```

see more

▲ | ▼ • Reply • Share ›

Nikhil Agrawal ➦ Nikhil Agrawal • 2 years ago

Sorry this solution will NOT work for following tree:

1
/ \
2 3
/ \
4 6
/ \
5 7
\
8

1 ▲ | ▼ • Reply • Share ›

abhikumar18 Nikhil Agrawal • 2 years ago

i think nikhil it will work...
c code...

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
typedef struct Node
{
struct Node* left;
int data;
struct Node* right;
}tNode;
tNode* memory_Alloc(int item)
{
tNode* ptr=NULL;
ptr=(tNode*)malloc(sizeof(tNode));
ptr->left=NULL;
ptr->data=item;
```

see more

• Reply • Share ›

srk abhikumar18 • a year ago

@abhi: why have you put this line in the function calculate_Leaf_Nodes???

```
if(root->left==NULL || root->right==NULL)
{
return 1;
}
```

if a tree is such that it has one child and that child again has multiple children then it will give wrong answer...please correct me if i am wrong.

• Reply • Share ›

abhishek08aug • 3 years ago

C++ code:

```
 #include <iostream>
#include <stdlib.h>
using namespace std;


class tree_node {
```

```
  private:
    int data;
    tree_node * left;
    tree_node * right;
  public:
    tree_node() {
      left=NULL;
      right=NULL;
    }
    void set_data(int data) {
      this->data=data;
```

see more

2 • Reply • Share ›

vignesh • 3 years ago

For counting the leaf nodes, we can use the level order algorithm itself by passing the height as the maxheight-1. Correct me if am wrong.

• Reply • Share ›

vignesh vignesh • 3 years ago

This would work only in case of a balanced tree. It won't work for other scenarios.

2 • Reply • Share ›

Holden vignesh • 3 months ago

How come it works only for balanced trees?

• Reply • Share ›

sankarshan • 3 years ago

```
 void countleaf(struct node* root,int *count)
{
     if(root){
              countleaf(root->left,count);
              if(root->left==NULL && root->right==NULL)
              (*count)++;
              countleaf(root->right,count);
     }
}
int main(void){
int count=0;
/*build tree*/
 countleaf(root,&count);
printf("no of leaves: %d",count);
return 0;
```

```
}
```

1 • Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy