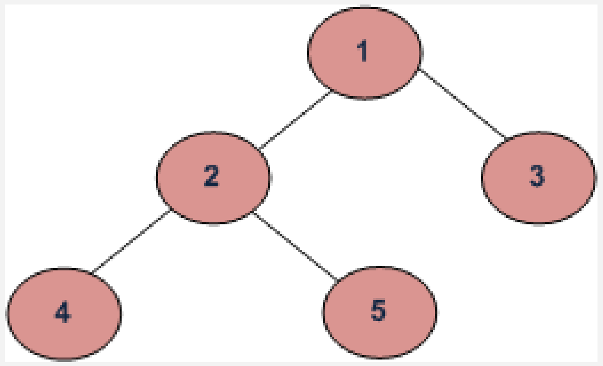
Program to count leaf nodes in a binary tree

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:

Google™ Custom Search

GeeksforGeeks

52,731 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```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;
};

/* 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);
```

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

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

  getchar();
  return 0;
}
```

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.

Better Than Hadoop.
HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.
LexisNexis
Learn More

Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)
- Print Right View of a Binary Tree
- Red-Black Tree | Set 3 (Delete)
- Construct a tree from Inorder and Level order traversals
- Print all nodes at distance k from a given node
- Print a Binary Tree in Vertical Order | Set 1

- Interval Tree
- Check if a given Binary Tree is height balanced like a Red-Black Tree

7 Tweet 0 2

695 Subscribe

Writing code in comment? Please use **ideone.com** and share the link here.

36 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

groomnestle · 5 months ago

Traverse the tree in any order (in-order, pre-order or post-order) and count++

Reply · Share ›

munai · 7 months 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;
};
```

Recent Comments

karthik it should have been max_wrap= max_wrap -...

Maximum circular subarray sum · 1 minute ago

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 45 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 1 hour ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 1 hour ago

@meya Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 3 hours ago

typedef struct list list;
list *head,*cur_node,*prev_node;
node *NewNode(int val)
{
node *temp=(node *)malloc(sizeof(node));

see more

3 ^ | ⌄ • Reply • Share ›

Nikhil Agrawal • 11 months ago

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

[sourcecode language="Java" 1="void" 2="numberofleafs(Node" 3="root)" 4="
value="0;" 7="if(root==null)" 8="{" 9="System.out.println("Number" 10="of" 11=
14="}" 15="else" 16="{" 17="Node" temp="null;" 18="Queue<Node>" q="new"
21="q.add(temp);" 22="while(!q.isEmpty())" 23="{" 24="Node" t="q.remove();"
27="while(t!=null)" 28="{" 29="value++;" 30="if(t.left!=null)" 31="q.add(t.left);" 3
34="}" 35="q.add(temp);" 36="if(value>max)" 37="}" 38="else" 39="if(t==null)"
44="System.out.println("Number" 45="of" 46="leaf" 47="Nodes="+max);" 48="

^ | ⌄ • Reply • Share ›

Nikhil Agrawal ➦ Nikhil Agrawal • 11 months 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 • 11 months ago

Sorry this solution will NOT work for following tree:

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

1 • Reply • Share ›

abhikumar18 Nikhil Agrawal • 10 months 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 ›

abhishek08aug • a year 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

1 ^ | ⌄ • Reply • Share ›

vignesh • 2 years ago

For counting the leaf nodes, we can use the level order algorithm itself by pas
Correct me if am wrong.

^ | ⌄ • Reply • Share ›

vignesh ➦ vignesh • 2 years ago

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

^ | ⌄ • Reply • Share ›

sankarshan • 2 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;
}
```

^ | v • Reply • Share ›

beginner • 2 years ago

what would be the best way to check if all the leaf nodes are at the same level

```
 /* Paste your code here (You may delete these lines if not writing c
```

^ | v • Reply • Share ›

dam • 3 years ago

Can someone provide me a snippet showing how to show only the nodes whi

^ | v • Reply • Share ›

dam ➦ dam • 3 years ago

In this case (the picture above) to show 2.

^ | v • Reply • Share ›

kartik ➦ dam • 3 years ago

```
 int print(struct node* node)
{
  if(node == NULL)
    return 0;
  if(node->left == NULL && node->right==NULL)
    return 1;
  if (print(node->left) && print(node->right))
    printf(" %d ", node->data);
  return 0;
}
```

sagar2693 ➦ kartik • 11 months ago

@kartik your code falters when a node has only one lea
what would be the most obvious way for dat???

```
/* Paste your code here (You may delete these li
```

Reply • Share ›

guest123 • 3 years ago

Hi All,

What will be the method to find the number of leaf nodes in a tree if depth is al

Thanks!

Reply • Share ›

Venki ➦ guest123 • 3 years ago

@guest123, In a strictly binary tree, usually depth starts at 0, and at lev
next level. If the depth is d, the tree will have *2^d* leaf nodes, and *2^(d+*
nodes overall in the tree.

Reply • Share ›

Ankit Sablok ➦ Venki • a year ago

your logic only holds for a complete binary tree not for general c
given the depth of the tree lets say 6, there can be different nun
leaves on the 5th level, 4th, 3rd and so on and hence this calcu

```
/* Paste your code here (You may delete these lines if
```

Reply • Share ›

Reema Khandelwal • 3 years ago

```
int CalLeafNodes(node *root)
{
 int count=0,h1=0,h2=0;
 if(root->llink==NULL && root->rlink==NULL)
 {
  return count++;
 }
else
 {
  h1=CalLeafNodes(root->llink);
  h2=CalLeafNodes(root->rlink);
 }
return (h1+h2);
}
```

Reply • Share ›

paul • 3 years ago

Could you help in how may i count the nodes that do not have grachild? I cant

Reply • Share ›

kartik paul • 3 years ago

following code should work.

```
unsigned int getCount(struct node* node)
{
  if (node == NULL)
    return 0;
  if (node->left == NULL && node->right==NULL)
```

```
    return 1;
  if (node->left == NULL && node->right==NULL &&
      node->left->left == NULL && node->right->left==NULL &&
      node->left->right == NULL && node->right->right==NULL
     )
   return 1;
  else
    return getCount(node->left)+
           getCount(node->right);
}
```

^ | v • Reply • Share ›

Kishan Gohil • 3 years ago

The code mentioned in the original post will only count the number of levels in the diagram is actually 5, not 3. The level count however is 3. If you want to co including the root and all child elements, and all child elements of these childre tiniest change will make the biggest difference.

The change is that in the final return statement where the count of the left is a a 1 each time you also count it's parent.

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

see more

^ | ⌄ • Reply • Share ›

kartik ➦ Kishan Gohil • 3 years ago

Your code counts total nodes in a Binary Tree (not leaf nodes). The lea
only. The 3 leaf nodes are nodes with values as 4, 5 and 3.

^ | ⌄ • Reply • Share ›

srock ➦ kartik • 3 years ago

I think the code put up by Kishan Gohil doesn't count the total n
check for

```
if(node->left == NULL && node->right==NULL)
   return 1;
```

to count the total number of node. Any comments.

^ | ⌄ • Reply • Share ›

kartik ➦ srock • 3 years ago

The line mentioned by u is fine. The problem is with the

```
return 1 + getLeafCount(node->left) + getLeafCour
```

Following is the correct modified line

```
return getLeafCount(node->left) + getLeafCount(nc
```

Reply • Share ›

srock kartik • 3 years ago
Makes sense.
Thanks again!
Reply • Share ›

kartik kartik • 3 years ago
@srock
The following line is not necessary to count total nodes
harm the count logic as the size of a leaf node is 1. So t
of nodes in a binary tree works with or without following

```
if(node->left == NULL && node->right==NULL)
   return 1;
```

Reply • Share ›

srock kartik • 3 years ago
Hi Kartik,
why would you check if the child nodes are empty to co
nodes (not just leaf)?

why is this code required if someone wanted to count th
tree (not just leaf node). [the below code is from Kishan

```
if(node->left == NULL && node->right==NULL)
   return 1;
```

My question is because for Kishan Gohil's post you mer count the total number of nodes.

Just wanted to make sure if I was incorrect in understar

Thanks.

Reply • Share ›

jack • 3 years ago

write a fn that returns total no. of nodes in a binary tree.

```
int GetTotal(NODE* tree)
{
  int NL, NR;
  if(tree==NULL)
    return 0;
  NL = GetTotal(tree->left);
  NR = GetTotal(tree->right);
  return NL+NR+1;
}
```

Reply • Share ›

Venki • 4 years ago

There is another way. We can make use of level order traversing. Use a queu
one after another. While dequeuing if both the left and right nodes are NULL it

1. If root is only node return 1, otherwise enqueue the root
2. Repeat till queue is not empty
3. If current->left && current->right are NULL, it is leaf
4. If current->left node is not NULL enqueue the node
5. If current->right node is not NULL enqueue the node
6. dequeue next node from queue

The worst case queue size is maximum number of leaf nodes which depends

^ | ⌄ • Reply • Share ›

Venki ➦ Venki • 4 years ago

Code provided in Qt framework http://qt.nokia.com/

```
#include <QtCore/QCoreApplication>
#include <QQueue>

// A binary tree aggregate data structure
struct TreeNode
{
    int data;
    struct TreeNode* left;
    struct TreeNode* right;
};

// Node pointer
typedef struct TreeNode* TreeNodePointer;

/* Algorithm
```

see more

^ | ⌄ • Reply • Share ›

Venki ➦ Venki • 4 years ago

Code provided using Qt framework. Sorry for incorrect phrasin

^ | ⌄ • Reply • Share ›

Bandicoot • 4 years ago

Can't we just use a global variable "count" and do an inorder traversal like this:

```
void inorder(node *node)
{
   if(node == NULL)
       return;
   if(node->left == NULL && node->right == NULL)
       count++;
   inorder(node->left);
   inorder(node->right);
}
```

^ | ⌄ • Reply • Share ›

GeeksforGeeks ➦ Bandicoot • 4 years ago

@Bandicoot: The code is fine. It gives the correct output. But, global va
considered as bad practice.

http://en.wikipedia.org/wiki/G...

^ | ⌄ • Reply • Share ›

Neeraj Mangal ➦ GeeksforGeeks • 4 years ago

We can use static variable to print the number of leafs in above

```
int  print_number_of_leaf(struct node *tree_node)
{
  static number_of_leaf = 0;
  if (tree_node == NULL){
    return 0;
  }else {
    if (tree_node->left == NULL && tree_node->right == N
      number_of_leaf++;
    }
    print_number_of_leaf(tree_node->left);
    print_number_of_leaf(tree_node->right);
  }
  return number_of_leaf;
}
```

call this from main as

printf("number of leaf nodes [%d]", print_number_of_leaf(root));

^ | ⌄ • Reply • Share ›

kartik ➦ Neeraj Mangal • 4 years ago

The use of local static variables should also be avoided
number of times a function is being called. The reason i
effects of a function call to other subsequent function ca

Like, if I call your function print_number_of_leaf() 2 time
give correct results first time, but it will not for the next c

^ | ⌄ • Reply • Share ›

Subscribe   Add Disqus to your site