Find Count of Single Valued Subtrees

Given a binary tree, write a program to count the number of Single Valued Subtrees. A Single Valued Subtree is one in which all the nodes have same value. Expected time complexity is O(n).

Example:

```
Input: root of below tree
              5
             / \
            1   5
           / \   \
          5   5   5
Output: 4
There are 4 subtrees with single values.


Input: root of below tree
              5
             / \
            4   5
           / \   \
          4   4   5
Output: 5
There are five subtrees with single values.
```

We strongly recommend you to minimize your browser and try this yourself first.

A **Simple Solution** is to traverse the tree. For every traversed node, check if all values under this node are same or not. If same, then increment count. Time complexity of this solution is $O(n^2)$.

An **Efficient Solution** is to traverse the tree in bottom up manner. For every subtree visited, return true if subtree rooted under it is single valued and increment count. So the idea is to use count as a reference parameter in recursive calls and use returned values to find out if left and right subtrees are single valued or not.

Below is the implementation of above idea.

C++

```
// C++ program to find count of single valued subtrees
#include<bits/stdc++.h>
using namespace std;

// A Tree node
struct Node
{
    int data;
    struct Node* left, *right;
};

// Utility function to create a new node
Node* newNode(int data)
{
    Node* temp = new Node;
    temp->data = data;
    temp->left = temp->right = NULL;
    return (temp);
}

// This function increments count by number of single
// valued subtrees under root. It returns true if subtree
// under root is Singly, else false.
bool countSingleRec(Node* root, int &count)
{
    // Return false to indicate NULL
    if (root == NULL)
       return true;

    // Recursively count in left and right subtrees also
    bool left = countSingleRec(root->left, count);
    bool right = countSingleRec(root->right, count);

    // If any of the subtrees is not singly, then this
    // cannot be singly.
    if (left == false || right == false)
       return false;

    // If left subtree is singly and non-empty, but data
    // doesn't match
    if (root->left && root->data != root->left->data)
        return false;

    // Same for right subtree
    if (root->right && root->data != root->right->data)
        return false;

    // If none of the above conditions is true, then
    // tree rooted under root is single valued, increment
    // count and return true.
    count++;
    return true;
}

// This function mainly calls countSingleRec()
// after initializing count as 0
int countSingle(Node* root)
{
    // Initialize result
```

```
    int count = 0;

    // Recursive function to count
    countSingleRec(root, count);

    return count;
}

// Driver program to test
int main()
{
    /* Let us construct the below tree
            5
          /   \
        4      5
      /  \      \
     4    4      5 */
    Node* root         = newNode(5);
    root->left         = newNode(4);
    root->right        = newNode(5);
    root->left->left   = newNode(4);
    root->left->right  = newNode(4);
    root->right->right = newNode(5);

    cout << "Count of Single Valued Subtrees is "
         << countSingle(root);
    return 0;
}
```

Run on IDE

Java

```
// Java program to find count of single valued subtrees

/* 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 Count {

    public int count;
}

class BinaryTree {

    Node root;
    Count ct = new Count();

    // This function increments count by number of single
    // valued subtrees under root. It returns true if subtree
    // under root is Singly, else false.
    boolean countSingleRec(Node node, Count c) {
```

```
        // Return false to indicate NULL
        if (node == null) {
            return true;
        }

        // Recursively count in left and right subtrees also
        boolean left = countSingleRec(node.left, c);
        boolean right = countSingleRec(node.right, c);

        // If any of the subtrees is not singly, then this
        // cannot be singly.
        if (left == false || right == false) {
            return false;
        }

        // If left subtree is singly and non-empty, but data
        // doesn't match
        if (node.left != null && node.data != node.left.data) {
            return false;
        }

        // Same for right subtree
        if (node.right != null && node.data != node.right.data) {
            return false;

        }

        // If none of the above conditions is true, then
        // tree rooted under root is single valued, increment
        // count and return true.
        c.count++;
        return true;
    }

    // This function mainly calls countSingleRec()
    // after initializing count as 0
    int countSingle() {
        return countSingle(root);
    }

    int countSingle(Node node) {

        // Recursive function to count
          countSingleRec(node, ct);
        return ct.count;
    }

    public static void main(String args[]) {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(5);
        tree.root.left = new Node(4);
        tree.root.right = new Node(5);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(4);
        tree.root.right.right = new Node(5);

        System.out.println("The count of single valued sub trees is : "
                                            + tree.countSingle());
    }
}

// This code has been contributed by Mayank Jaiswal
```

Run on IDE

Python

```
# Python program to find the count of single valued subtrees

# Node Structure
class Node:
    # Utility function to create a new node
    def __init__(self ,data):
        self.data = data
        self.left = None
        self.right = None


# This function increments count by number of single
# valued subtrees under root. It returns true if subtree
# under root is Singly, else false.
def countSingleRec(root , count):
    # Return False to indicate None
    if root is None :
        return True

    # Recursively count in left and right subtress also
    left = countSingleRec(root.left , count)
    right = countSingleRec(root.right , count)

    # If any of the subtress is not singly, then this
    # cannot be singly
    if left == False or right  == False :
        return False

    # If left subtree is singly and non-empty , but data
    # doesn't match
    if root.left and root.data != root.left.data:
        return False

    # same for right subtree
    if root.right and root.data != root.right.data:
        return False

    # If none of the above conditions is True, then
    # tree rooted under root is single valued,increment
    # count and return true
    count[0] += 1
    return True


# This function mainly calss countSingleRec()
# after initializing count as 0
def countSingle(root):
    # initialize result
    count = [0]

    # Recursive function to count
    countSingleRec(root , count)

    return count[0]


# Driver program to test

"""Let us contruct the below tree
           5
```

```
          /   \
        4       5
       /  \      \
      4    4      5
"""
root = Node(5)
root.left = Node(4)
root.right = Node(5)
root.left.left = Node(4)
root.left.right = Node(4)
root.right.right = Node(5)
countSingle(root)
print "Count of Single Valued Subtress is" , countSingle(root)

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

Run on IDE

Output:

```
Count of Single Valued Subtrees is 5
```

Time complexity of this solution is O(n) where n is number of nodes in given binary tree.

Thanks to Gaurav Ahirwar fir suggesting above solution.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

23 Comments Category: Trees

Related Posts:

- 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?
- Closest leaf to a given node in Binary Tree

(Login to Rate and Mark)

2.7 Average Difficulty : **2.7/5.0**
Based on **14** vote(s)

Add to TODO List
Mark as DONE

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

23 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

wes chen • 10 days ago
This is postorder solution.
• Reply • Share ›

Utkarsh Agrawal • a month ago
Minor error in countSingleRec() function..
Comment not coherent with code:

// Return false to indicate NULL
if (root == NULL)
return true;
• Reply • Share ›

RAJ • a month ago
Very simple solution to understand -

public int singleValuedSubTrees(Node root) {

if(root == null) return 0;

if(root.left == null && root.right == null) return 1;

int left = singleValuedSubTrees(root.left);

int right = singleValuedSubTrees(root.right);

if(root.left != null && root.data != root.left.data) return (left+right);

if(root.right != null && root.data != root.right.data) return (left+right);

return left + right + 1;

}

Basically, propagate the count of single valued subtrees from left and right and add 1 to it whenever the value of the root matches its left and right child node data.

• Reply • Share ›

AllergicToBitches • 2 months ago

http://ideone.com/yavIUq

• Reply • Share ›

Rohit singh • 2 months ago

c program to count single valued subtree using static variable
http://code.geeksforgeeks.org/...

• Reply • Share ›

Mohit Kumra • 3 months ago

Could be done with the use of Static variable limiting the number of classes to 2!!

// Java program to find count of single valued subtrees

//Class Node

class Node {

int info;

Node left, right;

public Node(int item) {

info = item;

left = right = null;

}

}

see more

• Reply • Share ›

coder12489 • 4 months ago

Ruby Code

class TreeNode

attr_accessor :value, :left, :right

def initialize value

@value = value

end

end

class CountOfSingleValuedSubtree

def count_single_valued_subtrees root

if root.nil?

return [0, true]

see more

• Reply • Share ›

Suraj Adhikari • 4 months ago

Hiring SDE-2s for Amazon. If interested send resume to adhikari.suraj@gmail.com

• Reply • Share ›

Avata

This comment was deleted.

Krishna Kumar Guest • 4 months ago

public class SingleValuedSubTree {
public static class Tree {
public static class Node {
Node left;
Node right;
int value;

public Node(int value) {

```
this.value = value;
}
}

public Node root;
private int numOfSingleValuedSubTree = 0;

public int getNumOfSingleValuedSubtree(Node root) {
calculateNumOfSingleValuedSubTree(root);
return numOfSingleValuedSubTree;
}
```

see more

• Reply • Share ›

Yt R • 4 months ago

The time complexity of tree traversal is O(n),but why the time complexity of simple solution above is O(n2)?

1 • Reply • Share ›

Krishna Kumar ➦ Yt R • 4 months ago

in the simple solution : its a top down approach starting from root to the leaves. Tree Traversal is O(N) but for the simple solution, for each node you visit you need to read values of all the nodes in the sub tree rooted at the visited node, hence it becomes the operation of N^2.

2 • Reply • Share ›

Yt R ➦ Krishna Kumar • 4 months ago

Got it,thank you :-)

• Reply • Share ›

abhishek • 5 months ago

why below is not considered as subtree, it also has single values :

5
\
5
\
5

• Reply • Share ›

Fattepur Mahesh ➦ abhishek • 5 months ago

But the left left is not having same value as root.

Condition that:

1. If a root node has one children then, node and its children should have same value

2. If a root node has two children then, node and its children should have same value

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

1 • Reply • Share ›

abhishek ➚ Fattepur Mahesh • 5 months ago

okay, got it. But this condition should be mentioned above.

• Reply • Share ›

Krishna Kumar ➚ abhishek • 5 months ago

because of the different left child's value

• Reply • Share ›

abhishek ➚ Krishna Kumar • 5 months ago

I didn't get you, consider only this subtree :

5
\
5
\
5

from the given example. It has all single values right ?

• Reply • Share ›

Krishna Kumar ➚ abhishek • 5 months ago

see the left child's value its 4 so it becomes a tree rooted at 5 , left child having value as 4 and right child having value as 5 so it is not a single valued subtree

• Reply • Share ›

Md Imran Ali • 5 months ago

look the below solution.

// C++ program to find count of single valued subtrees

#include<bits stdc++.h="">

using namespace std;

// A Tree node

```
struct Node

{

int data;

struct Node* left, *right;

};

// Utility function to create a new node
```

see more

• Reply • Share ›

Koustav Chatterjee • 5 months ago

Why the output of the above code is 5 ? It should be 2 according to me. Can anyone help me out ?

• Reply • Share ›

Krishna Kumar Koustav Chatterjee • 5 months ago

All the leaf nodes make a Single valued Subtree so for above we have 3 such nodes (4,4,5). then a subtree rooted at 4 and left & right children as 4 makes 1 single valued subtree. Also we have a subtree rooted at 5 and left child null & right child as 5. so total as 5

• Reply • Share ›

Ravi Yadav Krishna Kumar • 4 months ago

Thx krishna

1 • Reply • Share ›

Mayank Paneri • 5 months ago

Authors should come up with new questions this type of question has been asked so many times that i have lost the count...

• Reply • Share ›