

GeeksforGeeks

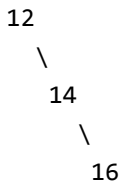
A computer science portal for geeks

Placements Practice GATE CS IDE Q&A
GeeksQuiz

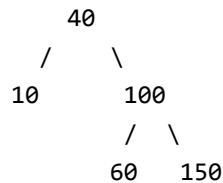
Check if a given Binary Tree is height balanced like a Red-Black Tree

In a **Red-Black Tree**, the maximum height of a node is at most twice the minimum height (The **four Red-Black tree properties** make sure this is always followed). Given a Binary Search Tree, we need to check for following property.

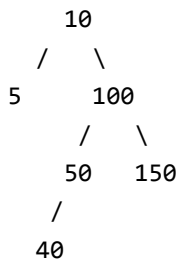
For every node, length of the longest leaf to node path has not more than twice the nodes on shortest path from node to leaf.



Cannot be a Red-Black Tree
with any color assignment
Max height of 12 is 1
Min height of 12 is 3



It can be Red-Black Tree



It can also be Red-Black Tree

Expected time complexity is $O(n)$. The tree should be traversed at-most once in the solution.

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

For every node, we need to get the maximum and minimum heights and compare them. The idea is to traverse the tree and for every node check if it's balanced. We need to write a recursive function that returns three things, a boolean value to indicate the tree is balanced or not, minimum height and maximum height. To return multiple values, we can either use a structure or pass variables by reference. We have passed maxh and minh

by reference so that the values can be used in parent calls.

```

/* Program to check if a given Binary Tree is balanced like a Red-Black Tree */
#include <iostream>
using namespace std;

struct Node
{
    int key;
    Node *left, *right;
};

/* utility that allocates a new Node with the given key */
Node* newNode(int key)
{
    Node* node = new Node;
    node->key = key;
    node->left = node->right = NULL;
    return (node);
}

// Returns true if the Binary tree is balanced like a Red-Black
// tree. This function also sets value in maxh and minh (passed by
// reference). maxh and minh are set as maximum and minimum heights of root.
bool isBalancedUtil(Node *root, int &maxh, int &minh)
{
    // Base case
    if (root == NULL)
    {
        maxh = minh = 0;
        return true;
    }

    int lmxh, lmnh; // To store max and min heights of left subtree
    int rmhx, rmnh; // To store max and min heights of right subtree

    // Check if left subtree is balanced, also set lmxh and lmnh
    if (isBalancedUtil(root->left, lmxh, lmnh) == false)
        return false;

    // Check if right subtree is balanced, also set rmhx and rmnh
    if (isBalancedUtil(root->right, rmhx, rmnh) == false)
        return false;

    // Set the max and min heights of this node for the parent call
    maxh = max(lmxh, rmhx) + 1;
    minh = min(lmnh, rmnh) + 1;

    // See if this node is balanced
    if (maxh <= 2*minh)

```

```
        return true;

    return false;
}

// A wrapper over isBalancedUtil()
bool isBalanced(Node *root)
{
    int maxh, minh;
    return isBalancedUtil(root, maxh, minh);
}

/* Driver program to test above functions*/
int main()
{
    Node * root = newNode(10);
    root->left = newNode(5);
    root->right = newNode(100);
    root->right->left = newNode(50);
    root->right->right = newNode(150);
    root->right->left->left = newNode(40);
    isBalanced(root)? cout << "Balanced" : cout << "Not Balanced";

    return 0;
}
```

Output:

Balanced

Time Complexity: Time Complexity of above code is $O(n)$ as the code does a simple tree traversal.

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



15 Comments Category: [Trees](#)

Related Posts:

- [Check if removing an edge can divide a Binary Tree in two halves](#)
- [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?](#)

([Login](#) to Rate and Mark)

3.2 Average Difficulty : **3.2/5.0**
Based on **4** vote(s)

☐

Add to TODO List

☐

Mark as DONE

[Like](#) [Share](#) 7 people like this.

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

@geeksforgeeks, [Some rights reserved](#)

[Contact Us!](#)

[About Us!](#)

[Advertise with us!](#)