How to determine if a binary tree is height-balanced?

Google™ Custom Search

A tree where no leaf is much farther away from the root than any other leaf. Different balancing schemes allow different definitions of “much farther” and different amounts of work to keep them balanced.

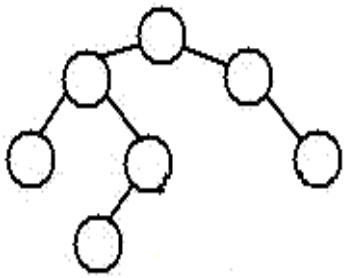
Consider a height-balancing scheme where following conditions should be checked to determine if a binary tree is balanced.
An empty tree is height-balanced. A non-empty binary tree T is balanced if:
1) Left subtree of T is balanced
2) Right subtree of T is balanced
3) The difference between heights of left subtree and right subtree is not more than 1.

The above height-balancing scheme is used in AVL trees. The diagram below shows two trees, one of them is height-balanced and other is not. The second tree is not height-balanced because height of left subtree is 2 more than height of right subtree.

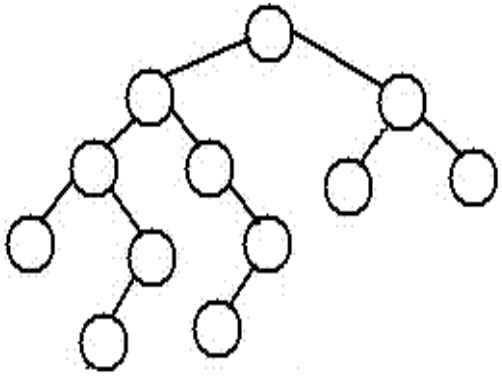
GeeksforGeeks

52,731 people like GeeksforGeeks.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com



A height-balanced Tree



Not a height-balanced tree

To check if a tree is height-balanced, get the height of left and right subtrees. Return true if difference between heights is not more than 1 and left and right subtrees are balanced, otherwise return false.

```
/* program to check if a tree is height-balanced or not */
#include<stdio.h>
#include<stdlib.h>
#define bool int

/* 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;
};

/* Returns the height of a binary tree */
int height(struct node* node);

/* Returns true if binary tree with root as root is height-balanced */
bool isBalanced(struct node *root)
{
   int lh; /* for height of left subtree */
   int rh; /* for height of right subtree */

   /* If tree is empty then return true */
```


ITT Tech - Official Site

itt-tech.edu

Tech-Oriented Degree Programs. Education for the Future.

Popular Posts

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
    if(root == NULL)
     return 1;

    /* Get the height of left and right sub trees */
    lh = height(root->left);
    rh = height(root->right);

    if( abs(lh-rh) <= 1 &&
        isBalanced(root->left) &&
        isBalanced(root->right))
      return 1;

    /* If we reach here then tree is not height-balanced */
    return 0;
}

/* UTILITY FUNCTIONS TO TEST isBalanced() FUNCTION */

/* returns maximum of two integers */
int max(int a, int b)
{
  return (a >= b)? a: b;
}

/*  The function Compute the "height" of a tree. Height is the
    number of 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;
```


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

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
    return(node);
}

int main()
{
    struct node *root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    root->left->left->left = newNode(8);

    if(isBalanced(root))
      printf("Tree is balanced");
    else
      printf("Tree is not balanced");

    getchar();
    return 0;
}
```

Time Complexity: O(n^2) Worst case occurs in case of skewed tree.

Optimized implementation: Above implementation can be optimized by calculating the height in the same recursion rather than calling a height() function separately. Thanks to Amar for suggesting this optimized version. This optimization reduces time complexity to O(n).

```
/* program to check if a tree is height-balanced or not */
#include<stdio.h>
#include<stdlib.h>
#define bool int

/* 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;
};

/* The function returns true if root is balanced else false
   The second parameter is to store the height of tree.
   Initially, we need to pass a pointer to a location with value
```

TRY MANAGED CLOUD
rackspace
the open cloud company

695 Subscribe

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)

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
   as 0. We can also write a wrapper over this function */
bool isBalanced(struct node *root, int* height)
{
  /* lh --> Height of left subtree
     rh --> Height of right subtree */
  int lh = 0, rh = 0;

  /* l will be true if left subtree is balanced
    and r will be true if right subtree is balanced */
  int l = 0, r = 0;

  if(root == NULL)
  {
    *height = 0;
     return 1;
  }

  /* Get the heights of left and right subtrees in lh and rh
    And store the returned values in l and r */
  l = isBalanced(root->left, &lh);
  r = isBalanced(root->right,&rh);

  /* Height of current node is max of heights of left and
     right subtrees plus 1*/
  *height = (lh > rh? lh: rh) + 1;

  /* If difference between heights of left and right
     subtrees is more than 2 then this node is not balanced
     so return 0 */
  if((lh - rh >= 2) || (rh - lh >= 2))
    return 0;

  /* If this node is balanced and left and right subtrees
    are balanced then return true */
  else return l&&r;
}


/* UTILITY FUNCTIONS TO TEST isBalanced() FUNCTION */

/* 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;
```

{...

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

AdChoices

▶ Binary Tree

▶ Java Tree

▶ XML Tree Viewer

AdChoices

▶ Red Black Tree

▶ JavaScript Tree

▶ Tree Structure

AdChoices

▶ Root Tree

▶ Tree Trees

▶ In the Tree

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
    node->left = NULL;
    node->right = NULL;

    return(node);
}

int main()
{
  int height = 0;

  /* Constructed binary tree is
             1
           /   \
         2       3
       /  \     /
     4     5  6
    /
   7
  */
  struct node *root = newNode(1);
  root->left = newNode(2);
  root->right = newNode(3);
  root->left->left = newNode(4);
  root->left->right = newNode(5);
  root->right->left = newNode(6);
  root->left->left->left = newNode(7);

  if(isBalanced(root, &height))
    printf("Tree is balanced");
  else
    printf("Tree is not balanced");

  getchar();
  return 0;
}
```

Time Complexity: O(n)

Please write comments if you find any of the above codes/algorithms incorrect, or find other ways to solve the same problem.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

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

13 Tweet 1 2

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

63 Comments GeeksforGeeks

Sort by Newest

Join the discussion

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Mohamad Al Mustapha • 2 months ago

Check this iterative algorithm solving the same problem.

http://cs-and-design.blogspot....

1 ▲ | ▼ • Reply • Share ›

supershal • 2 months ago

```
if( Integer.MaxInteger == isHeightBalanced( root, 0) ) {

return false;

}else{

return true;

}

public int isHeightBalanced(Node root, int height){

if(root == null){

return height;

}

int lHeight = isHeightBalanced( root.left, height + 1);

int rHeight = isHeightBalanced( root.left, height + 1);

if( abs(lHeight - rHeight) > 1) {

return Integer.MAXInteger;

}
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

}

return max(lHeight , rHeight) ;

}

• Reply • Share ›

phantom • 2 months ago

max = max_depth(root) #function to find max_depth ---> takes O(n)
min = min_depth(root) # function to find min_depth ---> takes O(n)
if (max - min) <= boundary:

return true
else:
return false

• Reply • Share ›

deepak • 6 months ago

Is the following tree height balanced:

1

/ \

6 2

/ \ / \

7 11 3 4

/ \ \ /

8 10 12 5

\

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

9

struct node *root = newNode(1);

see more

Reply • Share ›

Jasprit • 7 months ago

```
int isBalanced(struct node *root, int *height)
{
if (root == NULL) {*height = 0; return 1; }
int lh = 0, rh = 0;
if (isBalanced(root->left, &lh) && isBalanced(root->right, &rh))
{
*height = max(lh, rh) +1;
if ((abs(lh - rh) >= 2)) return 0;
return 1;
}
return 0;
}
```

Reply • Share ›

pavansrinivas • 7 months ago

Similar to "Check if Leaves are at same level"..

Reply • Share ›

Aditya Ambashtha • 10 months ago

We can check balance while finding height of the tree..

supply the address of an int whose value is initialized as 1 in the beginning..If t balanced, else unbalanced...

```
int findHeightAndBalance(struct node *root,int *ans)
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
{
    if (root==NULL)
        return 0;

    int lef,rig;

    lef=findHeightAndBalance(root->left,ans);
    rig=findHeightAndBalance(root->right,ans);

    if (abso(lef,rig)<=1)
        *ans=(*ans)&1;
    else
        *ans=(*ans)&0;

    return (max(lef,rig)+1);
}
```

• Reply • Share ›

Saurabh Gupta • 10 months ago

We can also calculate is_balanced by making small.
modification in max_depth code.

concept:check is_balanced at every node when calculating max_depth.

int is_balanced=1; // true.
int max_depth2(struct node* root){ // for is_balanced calculation.
if(root==NULL)return 0;.
int h1= max_depth2(root->left);.
int h2=max_depth2(root->right);.
if(abs(h1-h2)>1)is_balanced=0;.
return max(h1, h2)+1;.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

}
comment if you find this incorrect.

ultimate_coder • 11 months ago

+1 optimisation to method 1's space complexity.

```
 if( abs(lh-rh) <= 1 &&
       isBalanced(root->left) &&
       isBalanced(root->right))
     return 1;
This should be replaced by

return ( (abs(lh-rh) <= 1) &&  isBalanced(root->left) &&  isBalanced(r

As u can see it is a "tail recursion", so it reduces size of stack fra
```

initialcoder • 11 months ago

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

typedef struct NodeTag{
        char SYMBOL;
        struct NodeTag * LLINK;
        struct NodeTag * RLINK;
} TreeNode;

int getHeight(TreeNode * root){

        if(root == NULL)
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
            return 0;
        int lHeight =0;
        int rHeight = 0;

        lHeight = getHeight(root->LLINK);
        rHeight = getHeight(root->RLINK);
```

see more

Reply • Share ›

Guangjie Huang • 11 months ago

//C code

typedef struct NodeTag{.
char SYMBOL;
struct NodeTag * LLINK;.
struct NodeTag * RLINK;.
} TreeNode;

int getHeight(TreeNode * root){.

if(root == NULL).
return 0;
int lHeight =0;.
int rHeight = 0;.

lHeight = getHeight(root->LLINK);.
rHeight = getHeight(root->RLINK);.

if(lHeight > rHeight).

see more

Reply • Share ›

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

prity • 11 months ago

Hi, can some one explain the second approach. Thanks.

• Reply • Share ›

Priyanka K • a year ago

```
 //the function returns height of subtree whichever is greatest
//if subtrees are balanced otherwise return -1
int ifHeightBal(node *root){
    if(!root)
             return 0;
    int l=ifHeightBal(root->left);
    int r=ifHeightBal(root->right);
    if(abs(l-r)>1 || l==-1 || r==-1)
           return -1;
    return l>r?l+1:r+1;
}
```

• 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;
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

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

see more

1 ^ | v • Reply • Share ›

Nishant Kumar • a year ago

It returns -1 if not balanced.

```
int isHeightBalance(tree* root){
       if(root==NULL)
              return 0;
       int left = isHeightBalance(root->left);
       int right = isHeightBalance(root->right);

       if(abs(right-left)<2 && left!=-1 && right!=-1)
              return 1 + max(left,right);
       else
              return -1;

}
```

^ | v • Reply • Share ›

Gaurav ➦ Nishant Kumar • 5 months ago

your code will never return -1. Take a simple example:

1

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

/
2
/
3

• Reply • Share ›

Mika Gaurav • 5 months ago

For root node '1', it would return -1 as abs(right-left) = 2

• Reply • Share ›

varun jain • a year ago

```
#include
#include
#include
int h=0,lmax=0;
struct tree
{
char data;
struct tree *left;
struct tree *right;
};
int buildtree(struct tree **,int );
int balance(struct tree *,int);
int max(int,int);
int main()
{struct tree *root;
root=NULL;
int arr[8]={50,54,32,33,31,56,34,53};
for(int i=0;i<8;i++)
```

see more

• Reply • Share ›

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Shankar • a year ago

A
/ \
B C
\ / \
E F G
/ \
H I

Is the above a balanced binary tree? If not, wont the algorithm fail in this case?

Each of the subtrees are balanced and the diff between left subtree height anc

^ | v • Reply • Share ›

Nishant Kumar ➦ Shankar • a year ago

Yes, It is a balanced tree.

^ | v • Reply • Share ›

Shankar ➦ Shankar • a year ago

Sorry, i meant this tree.

....A
.../..\
..B....C
..\.../..\
..E..F....G
......../ \
.......H...I

1 ^ | v • Reply • Share ›

Shankar ➦ Shankar • a year ago

H & I are under G, F & G under C, E under B, B&C under A.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

mrn • a year ago

```
 /* Paste your code here (You may delete these lines if not writing c
int avl_check(Node *n,bool *check)
{
        if(n==NULL) return 0;
        int lh=avl_check(n->l,check);
        int lr=avl_check(n->r,check);
        if(abs(lh-lr) > 1)
        {
                *check=false;
                return 0;
        }
        return max(lh,lr)+1;
}
```


aygul • a year ago

Here is a c# version. Tried to optimize the given solution...

[sourcecode language="C#"]
bool IsBalanced(BTNode root)
{
int h = 0;
retrun IsBalanced(root, ref h);
}

bool IsBalanced(BTNode root, ref int h)
{
if (root == null) return true;

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
int lHeight = 0;
if (!IsBalanced(root.Left, ref lHeight))
return false;

int rHeight = 0;
if (!IsBalanced(root.Right, ref rHeight))
return false;

if (lHeight - rHeight >= 2 || rHeight - lHeight >= 2)
return false;

h = (lHeight > rHeight ? lHeight : rHeight) + 1;
return true;
}
```

▲ | ▼ • Reply • Share ›

Porus • a year ago

```
/* Paste your code here (You may delete these lines if not writing code) */
/*A little more Optimization..*/
bool isBalancedTree(Tree *t , int *height)

int lh = 0 , rh = 0;
if(!t) return TRUE;
if(isBalancedTree(t->left , &lh) , isBalancedTree(t->right , &rh))
{
*height = max(lh,rh) +1;
if(Abs(lh-rh) < 2)
return TRUE;
else
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

return FALSE;
}
else
return FALSE;
}

Hey Geeks , Please correct if im wrong...

• Reply • Share ›

Porus ➚ Porus • a year ago

Correction in the above code ::

Should be as below..

if(isBalancedTree(t->left , &lh) && isBalancedTree(t->right , &rh))

• Reply • Share ›

sreeram • a year ago

```
int isbalanced(struct node *root)
{
if(root == NULL)
return 0;

int b1=isbalanced(root->left);
int b2=isbalanceed(root->right);

if( b1 != -1 && b2 != -1 && abs(b1-b2) <= 1)
return 1+max(b1,b2);
else
return -1;
} //returns the height of the tree if balanced else returns -1
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Reply • Share ›

aaddd sreeram • a year ago

I think when root==null should return -1 since the empty tree height is -

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

Reply • Share ›

Udit Bhatia • a year ago

```
int isHeightBalanced(NODE *root){

        if(root==NULL){
                return 0;
        }

        int LH=0,RH=0;

        LH=isHeightBalanced(root->left);
        RH=isHeightBalanced(root->right);

        if((LH==-1) || (RH==-1)){
                return -1; //not a height balanced.
        }

        if( abs(LH-RH) <= 1){

                // is height balanced uptill now
                return ((LH > RH) ? (LH + 1) : (RH + 1));
        }else{
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
                return -1;
        }
}
```

Reply • Share ›

ramana • a year ago

```
 int maxheight( struct stree *p)
{
     if(p==NULL)
    return 0;
    else
    m=max(1+maxheight(p->left),1+maxheight(p->right));
    return m;
}

int minheight(struct stree *p)
{
    if(p==NULL)
    return 0;
    else
    m=min(1+minheight(p->left),1+minheight(p->right));

    return m;
}
```

sub max-min and verify it is 1 or 0

Reply • Share ›

avgul ramana • a year ago

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Nice approach! It is still O(n).

 • Reply • Share ›

godrej ramana • a year ago

can ny one explain about this complexity

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

 • Reply • Share ›

ramana • a year ago

```
int maxheight( struct stree *p)
{
if(p==NULL)
return 0;
else
m=max(1+maxheight(p->left),1+maxheight(p->right));
return m;
}

int minheight(struct stree *p)
{
if(p==NULL)
return 0;
else
m=min(1+minheight(p->left),1+minheight(p->right));

return m;
}
```

substract maxheight-minheight and check it if is 1 or 0

this works..... :)

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

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

• Reply • Share ›

code_player • 2 years ago

Actually I am not getting why time complexity of first approach is O(n^2). in the
lr)<=1 doesn&#039t satisfy then next expressions of && won&#039t be evalua

[sourcecode language="C"]
/* Paste your code here (You may delete these lines if not writing code) */

• Reply • Share ›

suja • 2 years ago

this is confusing for a clear idea go to link
http://www.mytechinterviews.co...

• Reply • Share ›

Abhinav • 2 years ago

The time complexity calculation for the first algorithm in this post seems flawe

I am providing my calculation and it would be great if you guys can take a look

Complexity Calculation
==========
The O(n^2) argument misses the fact that the number of nodes at every level i
height algorithm is going to vary at each level. Also, it is the balanced tree that
because the algorithm will find a much earlier exit in the case of unbalanced tr
in order to determine the balance.

The correct way to calculate this would be: lets say T is the time taken and we
tree,
T = summation[(2^h - 2) * n/(2^h - 1)]
where n = number of nodes in tree,

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

h = height of the node,
h ranges from 2 to lg(n).

The second term n/(2^h - 1) is the number of nodes at height h. The first term

see more

• Reply • Share ›

vj • 2 years ago

just find the max height from root and min height from root.If the difference is l
balanced tree..else not.

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

• Reply • Share ›

Dexters • 2 years ago

How is maxdepth for the tree is 0?

http://www.mytechinterviews.co...
http://stackoverflow.com/quest...

• Reply • Share ›

vinay polisetti • 2 years ago

Can someone explain me the usage of the height variable in method2 ?

• Reply • Share ›

mohit • 2 years ago

```
/* Paste your code here (You may delete these lines if not writing c
int blncd_tree(Node  n)
{
        if(n==NULL) return 0;
        int l=0,r=0;
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
        l=blncd_tree(n->l);
        r=blncd_tree(n->r);
        if(l==-2 || r==-2)
                 return -2;
        if(abs(l-r) <=1)
                return l>r?l+1:r+1;
        else
                return -2;
}
```

Reply • Share ›

master fuji • 2 years ago

Hi GeeksforGeeks,

im happy with ur site, one suggestion is to hide the solutions for people, who w

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

Reply • Share ›

GeeksforGeeks ➦ master fuji • 2 years ago

@master fuji: Thanks for the feedback. Hiding solutions is a nice idea. Keep visiting us!!

Reply • Share ›

sreenivas putta • 3 years ago

findout the maximum depth of the tree and minimum depth of the tree and find maxdepth-mindepth<=1 , then tree is balanced. the more optimized version fo

Reply • Share ›

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

GeeksforGeeks ➚ sreenivas putta • 3 years ago

@sreenivas putta & ayan_2587:

Consider the following tree. max and min dpeth of the tree are 0. But th
right subtrees of A are not balanced.
.....A
...../...\
.....B....C
..../.....\
...D.....E
../.....\
.F.....G

  • Reply • Share ›

Dexters ➚ GeeksforGeeks • 2 years ago

How is maxdepth for the tree is 0?

http://www.mytechinterviews.co...
http://stackoverflow.com/quest...

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

  • Reply • Share ›

ayan_2587 ➚ sreenivas putta • 3 years ago

Dude..that is what I have posted in my solution below !!!

  • Reply • Share ›

ayan_2587 • 3 years ago

My solution to the post :-
Do a simple Depth First Traversal of the given tree in question.
Maintain two variables, min & max. While doing the depth traversal once you r
level(level of the leaf node ie. height) of the leaf node with the min & max varial

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

level(level of the leaf node ie. height) of the leaf node with the min & max variable
min, else if it is greater than max, store it in max. At the end of the traversal, if
more than one then, the tree is not balanced.

Please let me know if there is any bug in this solution
Thanks :)

• Reply • Share ›

krishna ayan_2587 • 2 years ago

As i understand depth of the node is the count of nodes from the root t

Assuming above defnition is legal, Your logic doesnt make much sens
min depths should differ by one.

• Reply • Share ›

gaurav • 3 years ago

Could anyone explain, how the time complexity of the Optimized Algorithm is O

• Reply • Share ›

kartik gaurav • 3 years ago

@gaurav: The optimized solution just does tree traversal of the tree an
is O(n). See this post for time complexity of tree traversals.

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

@geeksforgeeks, **Some rights reserved** **Contact Us!** Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com