Write Code to Determine if Two Trees are Identical

Two trees are identical when they have same data and arrangement of data is also same.

To identify if two trees are identical, we need to traverse both trees simultaneously, and while traversing we need to compare data and children of the trees.

Algorithm:

```
sameTree(tree1, tree2)
1. If both trees are empty then return 1.
2. Else If both trees are non -empty
     (a) Check data of the root nodes (tree1->data ==  tree2->data)
     (b) Check left subtrees recursively  i.e., call sameTree(
          tree1->left_subtree, tree2->left_subtree)
     (c) Check right subtrees recursively  i.e., call sameTree(
          tree1->right_subtree, tree2->right_subtree)
     (d) If a,b and c are true then return 1.
3  Else return 0 (one is empty and other is not)
```

C/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;
};

/* 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);
}

/* Given two trees, return true if they are
 structurally identical */
int identicalTrees(struct node* a, struct node* b)
{
    /*1. both empty */
    if (a==NULL && b==NULL)
        return 1;

    /* 2. both non-empty -> compare them */
    if (a!=NULL && b!=NULL)
    {
        return
        (
            a->data == b->data &&
            identicalTrees(a->left, b->left) &&
            identicalTrees(a->right, b->right)
        );
    }

    /* 3. one empty, one not -> false */
    return 0;
}

/* Driver program to test identicalTrees function*/
int main()
{
    struct node *root1 = newNode(1);
    struct node *root2 = newNode(1);
    root1->left = newNode(2);
    root1->right = newNode(3);
    root1->left->left  = newNode(4);
    root1->left->right = newNode(5);

    root2->left = newNode(2);
    root2->right = newNode(3);
    root2->left->left = newNode(4);
    root2->left->right = newNode(5);

    if(identicalTrees(root1, root2))
        printf("Both tree are identical.");
    else
        printf("Trees are not identical.");

    getchar();
  return 0;
}
```

Run on IDE

Java

```
// Java program to see if two trees are identical

// A binary tree node
class Node {
    int data;
    Node left, right;

    Node(int item) {
        data = item;
        left = right = null;
    }
}

class BinaryTree {

    static Node root1, root2;

    /* Given two trees, return true if they are
     structurally identical */
    boolean identicalTrees(Node a, Node b) {

        /*1. both empty */
        if (a == null && b == null) {
            return true;
        }

        /* 2. both non-empty -> compare them */
        if (a != null && b != null) {
            return (a.data == b.data
                    && identicalTrees(a.left, b.left)
                    && identicalTrees(a.right, b.right));
        }

        /* 3. one empty, one not -> false */
        return false;
    }

    /* Driver program to test mirror() */
    public static void main(String[] args) {

        BinaryTree tree = new BinaryTree();

        tree.root1 = new Node(1);
        tree.root1.left = new Node(2);
        tree.root1.right = new Node(3);
        tree.root1.left.left = new Node(4);
        tree.root1.left.right = new Node(5);

        tree.root2 = new Node(1);
        tree.root2.left = new Node(2);
        tree.root2.right = new Node(3);
        tree.root2.left.left = new Node(4);
        tree.root2.left.right = new Node(5);

        if (tree.identicalTrees(root1, root2)) {
            System.out.println("Both trees are identical");
        } else {
            System.out.println("Trees are not identical");
        }

    }
}
```

Run on IDE

Python

```
# Python program to determine if two trees are identical

# A binary tree node has data, pointer to left child
# and a pointer to right child
class Node:
    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None


# Given two trees, return true if they are structurally
# identical
def identicalTrees(a, b):

    # 1. Both empty
    if a is None and b is None:
        return True

    # 2. Both non-empty -> Compare them
    if a is not None and b is not None:
        return ((a.data == b.data) and
                identicalTrees(a.left, b.left)and
                identicalTrees(a.right, b.right))

    # 3. one empty, one not -- false
    return False

# Driver program to test identicalTress function
root1 = Node(1)
root2 = Node(1)
root1.left = Node(2)
root1.right = Node(3)
root1.left.left = Node(4)
root1.left.right = Node(5)

root2.left = Node(2)
root2.right = Node(3)
root2.left.left = Node(4)
root2.left.right = Node(5)

if identicalTrees(root1, root2):
    print "Both trees are identical"
else:
    print "Trees are not identical"

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

Run on IDE

Time Complexity:

Complexity of the identicalTree() will be according to the tree with lesser number of nodes. Let number of nodes in two trees be m and n then complexity of sameTree() is O(m) where m < n.

Spend less time debugging, more time coding.
Get Visual Studio Code FREE
Microsoft

102 Comments Category: Trees Tags: Tree Traveral , 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)

1.5 Average Difficulty : **1.5/5.0**
Based on **27** vote(s)

Add to TODO List

Mark as DONE

Like Share 8 people like this.

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

102 Comments GeeksforGeeks 1 Login

Recommend 2 Share Sort by Newest

Join the discussion…

Satish Kumar • 2 months ago

```
int check(bnode* t1,bnode*t2)

{

if(t1==NULL&&t2==NULL)

return 1;

else if(t1==NULL&&t2!=NULL)

return 0;

else if(t1!=NULL&&t2==NULL)

return 0;

if(t1->data!=t2->data)

return 0;

return (check(t1->lc,t2->lc)==0) ? 0:check(t1->rc,t2->rc);

}
```

• Reply • Share ›

Vardaan Sangar • 3 months ago

This can be the code.

```
int AreStructurallySameTree(struct node* root,struct node* rooot)

{

if(root==NULL && rooot==NULL)

return 1;

if((root==NULL && rooot!=NULL) || (root!=NULL && rooot==NULL))

return 0;

if(root->data!=rooot->data)

return 0;

int a=AreStructurallySameTree(root->left,rooot->left);

if(a==0)
```

return 0;

else

return(AreStructurallySameTree(root->right,rooot->right));

}

^ | ˅ • Reply • Share ›

Karan • 4 months ago

int identicalTrees(struct node* root1, struct node* root2)

{

if(root1==NULL && root2==NULL)

return 1;

else if(root1->data != root2->data)

return 0;

else if(!(identicalTrees(root1->left,root2->left)))

return 0;

else if(!(identicalTrees(root1->right,root2->right)))

return 0;

else

return 1;

}

^ | ˅ • Reply • Share ›

Tonmoy Rakshit • 7 months ago

Can any1 explain the recursive operation in detail?

^ | ˅ • Reply • Share ›

Jayesh • 7 months ago

Java Implementation

http://javabypatel.blogspot.in...

^ | ˅ • Reply • Share ›

crazyforstudy it • 8 months ago

So. It is a top down approach......correct me if I am wrong

• Reply • Share ›

Ashish • 8 months ago

There is no reference to sameTree() used in Time Complexity analysis.It should be identicalTrees().

• Reply • Share ›

Pingu • 9 months ago

return
(
a->data == b->data &&
identicalTrees(a->left, b->left) &&
identicalTrees(a->right, b->right)
);

if all these conditions evaluate to true then it returns 1

is this the logic ?

• Reply • Share ›

Hitesh Saini Pingu • 8 months ago

yes

• Reply • Share ›

Pingu • 9 months ago

if inorder traversal of 2 trees is same does that mean they both are identical?

• Reply • Share ›

Aayush Pingu • 8 months ago

Nope, they might not be identical Pingu

• Reply • Share ›

Waqas Hamid • 9 months ago

If one of trees contains only the root and the other has multiple nodes run time error occurs.....Checked on Ideone...

• Reply • Share ›

Dipankar Bhardwaj • 9 months ago

Simple and clean http://code.geeksforgeeks.org/...

1 • Reply • Share ›

Lokesh • 9 months ago

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

• Reply • Share ›

sunil • 10 months ago
if i do pre/post/inorder travesal of the two tree and stored them into two different array.Is that stiil work. I know that going to take more space, but is it true to do so.
1 • Reply • Share ›

Anand Barnwal sunil • 9 months ago
Suppose the two trees are:
B <- A -> NULL (here A is root, left child is B and right child is NULL)
And
NULL<-A->B (here A is root, left child is NULL and right child is B)
The preorder traversal of both trees will give "AB". But the trees are not identical.
Hope, it clears your doubt.
4 • Reply • Share ›

Shravan Durvasula Anand Barnwal • 23 days ago
InOrder / PreOrder and PostOrder all the three would not respectively be the same for both the trees mentioned in the example.

However, in order to determine if trees are identical this way would need three traversals on each tree. But the method mentioned in the article, we can get away with just one.
• Reply • Share ›

Vardaan Sangar Anand Barnwal • 3 months ago
But your example doesnt work in inorder. So can we use inorder traversal to check whether the two BST trees are identical.
1 • Reply • Share ›

pk • 10 months ago
```
bool compare(struct node* node1, struct node* node2)
{

if (node1 == NULL && node2 == NULL)
return true;

if (node1 == NULL || node2 == NULL)
return false;

if (node1->data == node2->data)
return compare(node1->left, node2->left) && compare(node1->right, node2->right);

return false;
}
```

}

3 ^ | v • Reply • Share ›

dhiru • 10 months ago

can we check the identicality of the tree by comparing the preorder of two trees?

^ | v • Reply • Share ›

Hitesh Saini ➦ dhiru • 9 months ago

yes we can but only if both the trees have same number of nodes.
if the number of nodes are different then it will not work.
take a example make two trees of different length and in larger tree preserve the state of nodes of small tree (means both the tree are identical) but when you right preorder of both tree you will find yhem different .
Hope it clears your doubt .

^ | v • Reply • Share ›

dhiru ➦ Hitesh Saini • 9 months ago

but for two trees to be identical they should have same number of nodes

^ | v • Reply • Share ›

Hitesh Saini ➦ dhiru • 8 months ago

consider two trees
A
/ \
B C
and

A
/
B
/
C

preorder traversal of both the tree are same but they are not identical.
in my previous comment same number of nodes means same arrangement of nodes.

^ | v • Reply • Share ›

Vardaan Sangar ➦ Hitesh Saini • 3 months ago

Inorder traversal is the only traversal which can check whether Binary Search trees are identical? Can you counter.

^ | v • Reply • Share ›

Mission Peace • a year ago

Checkout my channel on youtube on binary trees
https://www.youtube.com/channe...

• Reply • Share ›

Anonymous • a year ago

Code in Java :

```
int compare(Node rt1 , Node rt2){
if(rt1 == null && rt2 == null)
return 1;
if(rt1 == null || rt2 == null)
return 0;
if(rt1.data == rt2.data){
return Math.min(compare(rt1.left,rt2.left),compare(rt1.right,rt2.right));
}else{
return 0;
}

}
```

1 • Reply • Share ›

Holden Anonymous • 10 months ago

You can define it 'boolean' with 'true' and 'false' :)

• Reply • Share ›

natasha • a year ago

```
int TreeFuncLib::IdenticalTree(struct node* tree1, struct node* tree2)

{

if(tree1 == NULL && tree2 == NULL)

return 1;

else if(tree1 == NULL && tree2 != NULL)

return(0);

else if(tree2 == NULL && tree1 != NULL)

return(0);

else

{
```

return((tree1->data == tree2->data)&&(IdenticalTree(tree1->left,tree2->left))&&
(IdenticalTree(tree1->right,tree2->right)));

}

}

1 • Reply • Share ›

Jerry Goyal • a year ago

```
int it(struct node* r1, struct node* r2)

{
        if(!r1&&!r2) return 1;

        if(!r1||!r2) return 0;

        return (r1->data==r2->data && it(r1->left,r2->left)
         && it(r1->right,r2->right) );
}
```

1 • Reply • Share ›

Jerry Goyal • a year ago

instead, use a global variable flag to check

```
int flag=0;
identicalTrees(struct node* a, struct node* b)
{
   if(a==NULL&&b==NULL){
        return;
   }
if(a==NULL || b==NULL){
        flag=1;
     return;
   }
   if(a->data!=b->data){
                flag=1;
           return;
        }
        identicalTrees(a->left,b->left);
        identicalTrees(a->right,b->right);
}

if flag=1 i.e.not identical
```

• Reply • Share ›

LNR Jerry Goyal • 10 months ago
What if either of the trees is NULL?
Won't
if(a->data!=b->data)
statement lead to an error?
• Reply • Share ›

Jerry Goyal LNR • 10 months ago
thanks for pointing out. updated the code.
• Reply • Share ›

surbhijain93 • a year ago
int i(struct node* n1,struct node* n2)

{

if(n1==NULL && n2==NULL)

return 1;

if((n1==NULL)||(n2==NULL))

return 0;

return

(n1->data==n2->data &&

i(n1->left,n2->left) &&

i(n1->right,n2->right));

}
• Reply • Share ›

trend_setter • a year ago
what is the difference between return 0, return 1 and return -1, How does it differs from exit?
• Reply • Share ›

AllergicToBitches trend_setter • a year ago
exit from anywhere on your code will terminate execution immediately.

return from main() is equivalent to exit() function. The program terminates immediately execution with exit status set as the value passed to return. Status 0 means the program succeded. Status different from 0 means the program exited due to error or abnornally.

return in an inner function (not main) will terminate immediately the execution of the specific function returning the given result to the calling function.

1 • Reply • Share ›

Rahul trend_setter • a year ago

return is used to return program control to the calling function. Exit is used to exit the code

• Reply • Share ›

Anand Barnwal Rahul • 10 months ago

exit(STATUS) is a system call i.e. used to terminate the current process.

• Reply • Share ›

trend_setter Rahul • a year ago

Thanks for the reply. What does return 0 and return 1 means?

• Reply • Share ›

Holden trend_setter • 10 months ago

true and false :)

• Reply • Share ›

trend_setter • a year ago

sometimes we put struct node* node and other times struct node *node, what is the difference ?

• Reply • Share ›

Hari trend_setter • a year ago

no difference.. it will always be interpreted as a pointer of type (struct node)

1 • Reply • Share ›

Ajitesh Mandal • a year ago

http://ideone.com/WsJylq

• Reply • Share ›

surbhijain93 • a year ago

```
bool compare(nodeptr node1,nodeptr node2)
{
if(node1==NULL&& node2==NULL)
return 1;
else if(node1!=NULL&& node2==NULL||node1==NULL&&node2!=NULL)
return 0;
else if(node1->data==node2->data)
{
printf("%d \n",node1->data);
return(compare(node1->left,node2->left)&&compare(node1->right,node2->right));
}
else
return 0;
}
```

• Reply • Share ›

Rajeev surbhijain93 • 10 months ago

in your second if condition you neednt check for all four conditions. Just an node1 == NULL || node2 == NULL would do as you are already checking the case where both are null before it

1 • Reply • Share ›

Saurabh Gupta • 2 years ago

Inorder traversal of both the trees simultaneously can also be an approach to the problem. Here is my code, do comment in case of any flaws or improvements.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int num;
node *left, *right;
};
node *root1 = NULL;
node *root2 = NULL;
node *newnode(int n)
{
node *ptr = (node *)malloc(sizeof(node));
ptr ->num = n;
ptr ->left = NULL;
ptr ->right = NULL;
```

see more

• Reply • Share ›

neelabhsingh Saurabh Gupta • a year ago

I think your following code will not run, Suppose root1==NULL, root1!=NULL, so your first condition fails. Now in your second condition root1->num will give you runtime error, please carefully check your code. Add if(root!=NULL && root2!=NULL) then your code is fine.

condition1:
if(root1 == NULL && root2 == NULL)
return;
condition2:
if(root1 ->num == root2 ->num)
{
identical(root1 ->left, root2 ->left);
identical(root1 ->right, root2 ->right);
}

4 • Reply • Share ›

Sumit Thakur • 2 years ago

Calculate in-order\pre-order\post-order for both trees and store them in two arrays. Now compare both arrays. Is it a good idea ?

4 • Reply • Share ›

Pranav Kumar Jha → Sumit Thakur • a year ago

See for yourself!
tree1: 1
/ \
2 3

tree2: 2
\
1
\
3

Both trees have same inorder traversals: 2 1 3
And they are definitely not identical!

11 • Reply • Share ›

Vardaan Sangar → Pranav Kumar Jha • 3 months ago

If we wanna check this identical nature in case of BST than we can use orders like Inorder

• Reply • Share ›

Holden → Pranav Kumar Jha • 10 months ago

Nice example :)

• Reply • Share ›

Load more comments