GeeksforGeeks

A computer science portal for geeks

Placements Practice GATE CS IDE Q&A GeeksQuiz

Check if a binary tree is subtree of another binary tree | Set 2

Given two binary trees, check if the first tree is subtree of the second one. A subtree of a tree T is a tree S consisting of a node in T and all of its descendants in T.

The subtree corresponding to the root node is the entire tree; the subtree corresponding to any other node is called a proper subtree.

For example, in the following case, Tree1 is a subtree of Tree2.

```
        Tree1
          x 
        /    \
      a       b
       \
        c


        Tree2
              z
            /   \
          x      e
        /    \     \
      a       b      k
       \
        c
```

We have discussed a O(n^2) solution for this problem. In this post a O(n) solution is discussed. The idea is based on the fact that inorder and preorder/postorder uniquely identify a binary tree. Tree S is a subtree of T if both inorder and preorder traversals of S arew substrings of inorder and preorder traversals of T respectively.

Following are detailed steps.

1) Find inorder and preorder traversals of T, store them in two auxiliary arrays inT[] and preT[].

2) Find inorder and preorder traversals of S, store them in two auxiliary arrays inS[] and preS[].

3) If inS[] is a subarray of inT[] and preS[] is a subarray preT[], then S is a subtree of T. Else not.

We can also use postorder traversal in place of preorder in the above algorithm.

Let us consider the above example

```
Inorder and Preorder traversals of the big tree are.
inT[]   =  {a, c, x, b, z, e, k}
preT[]  =  {z, x, a, c, b, e, k}

Inorder and Preorder traversals of small tree are
inS[]  = {a, c, x, b}
preS[] = {x, a, c, b}

We can easily figure out that inS[] is a subarray of
inT[] and preS[] is a subarray of preT[].
```

EDIT

```
The above algorithm doesn't work for cases where a tree is present
in another tree, but not as a subtree. Consider the following example.

        Tree1
          x
        /    \
      a       b
     /
    c


        Tree2
          x
        /    \
      a       b
     /         \
    c            d

Inorder and Preorder traversals of the big tree or Tree2 are.

Inorder and Preorder traversals of small tree or Tree1 are

The Tree2 is not a subtree of Tree1, but inS[] and preS[] are
subarrays of inT[] and preT[] respectively.
```

The above algorithm can be extended to handle such cases by adding a special character whenever we encounter NULL in inorder and preorder traversals. Thanks to Shivam Goel for suggesting this extension.

Following is the implementation of above algorithm.

C

```
#include <iostream>
#include <cstring>
using namespace std;
#define MAX 100

// Structure of a tree node
struct Node
{
    char key;
    struct Node *left, *right;
};

// A utility function to create a new BST node
Node *newNode(char item)
{
    Node *temp =  new Node;
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}

// A utility function to store inorder traversal of tree rooted
// with root in an array arr[]. Note that i is passed as reference
void storeInorder(Node *root, char arr[], int &i)
{
    if (root == NULL)
    {
        arr[i++] = '$';
        return;
    }
    storeInorder(root->left, arr, i);
    arr[i++] = root->key;
    storeInorder(root->right, arr, i);
}

// A utility function to store preorder traversal of tree rooted
// with root in an array arr[]. Note that i is passed as reference
void storePreOrder(Node *root, char arr[], int &i)
{
    if (root == NULL)
    {
        arr[i++] = '$';
        return;
    }
    arr[i++] = root->key;
    storePreOrder(root->left, arr, i);
    storePreOrder(root->right, arr, i);
}

/* This function returns true if S is a subtree of T, otherwise false */
bool isSubtree(Node *T, Node *S)
{
    /* base cases */
    if (S == NULL)  return true;
    if (T == NULL)  return false;

    // Store Inorder traversals of T and S in inT[0..m-1]
    // and inS[0..n-1] respectively
    int m = 0, n = 0;
    char inT[MAX], inS[MAX];
    storeInorder(T, inT, m);
    storeInorder(S, inS, n);
    inT[m] = '\0', inS[n] = '\0';

    // If inS[] is not a substring of preS[], return false
    if (strstr(inT, inS) == NULL)
        return false;

    // Store Preorder traversals of T and S in inT[0..m-1]
    // and inS[0..n-1] respectively
    m = 0, n = 0;
    char preT[MAX], preS[MAX];
    storePreOrder(T, preT, m);
    storePreOrder(S, preS, n);
    preT[m] = '\0', preS[n] = '\0';

    // If inS[] is not a substring of preS[], return false
    // Else return true
    return (strstr(preT, preS) != NULL);
}

// Driver program to test above function
int main()
{
    Node *T = newNode('a');
    T->left = newNode('b');
    T->right = newNode('d');
    T->left->left = newNode('c');
    T->right->right = newNode('e');

    Node *S = newNode('a');
    S->left = newNode('b');
    S->left->left = newNode('c');
    S->right = newNode('d');

    if (isSubtree(T, S))
        cout << "Yes: S is a subtree of T";
    else
        cout << "No: S is NOT a subtree of T";

    return 0;
}
```

Run on IDE

Java

```
// Java program to check if binary tree is subtree of another binary tree
class Node {

    char data;
    Node left, right;

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

class Passing {

    int i;
    int m = 0;
    int n = 0;
}

class BinaryTree {

    static Node root;
    Passing p = new Passing();

    String strstr(String haystack, String needle) {
        if (haystack == null || needle == null) {
            return null;
        }
        int hLength = haystack.length();
        int nLength = needle.length();
        if (hLength < nLength) {
            return null;
        }
        if (nLength == 0) {
            return haystack;
        }
        for (int i = 0; i <= hLength - nLength; i++) {
            if (haystack.charAt(i) == needle.charAt(0)) {
                int j = 0;
                for (; j < nLength; j++) {
                    if (haystack.charAt(i + j) != needle.charAt(j)) {
                        break;
                    }
                }
                if (j == nLength) {
                    return haystack.substring(i);
                }
            }
        }
        return null;
    }

    // A utility function to store inorder traversal of tree rooted
    // with root in an array arr[]. Note that i is passed as reference
    void storeInorder(Node node, char arr[], Passing i) {
        if (node == null) {
            arr[i.i++] = '$';
            return;
        }
        storeInorder(node.left, arr, i);
        arr[i.i++] = node.data;
        storeInorder(node.right, arr, i);
    }

    // A utility function to store preorder traversal of tree rooted
    // with root in an array arr[]. Note that i is passed as reference
    void storePreOrder(Node node, char arr[], Passing i) {
        if (node == null) {
            arr[i.i++] = '$';
            return;
        }
        arr[i.i++] = node.data;
        storePreOrder(node.left, arr, i);
        storePreOrder(node.right, arr, i);
    }

    /* This function returns true if S is a subtree of T, otherwise false */
    boolean isSubtree(Node T, Node S) {
        /* base cases */
        if (S == null) {
            return true;
        }
        if (T == null) {
            return false;
        }

        // Store Inorder traversals of T and S in inT[0..m-1]
        // and inS[0..n-1] respectively
        char inT[] = new char[100];
        String op1 = String.valueOf(inT);
        char inS[] = new char[100];
        String op2 = String.valueOf(inS);
        storeInorder(T, inT, p);
        storeInorder(S, inS, p);
        inT[p.m] = '\0';
        inS[p.m] = '\0';

        // If inS[] is not a substring of preS[], return false
        if (strstr(op1, op2) != null) {
            return false;
        }

        // Store Preorder traversals of T and S in inT[0..m-1]
        // and inS[0..n-1] respectively
        p.m = 0;
        p.n = 0;
        char preT[] = new char[100];
        char preS[] = new char[100];
        String op3 = String.valueOf(preT);
        String op4 = String.valueOf(preS);
        storePreOrder(T, preT, p);
        storePreOrder(S, preS, p);
        preT[p.m] = '\0';
        preS[p.n] = '\0';

        // If inS[] is not a substring of preS[], return false
        // Else return true
        return (strstr(op3, op4) != null);
    }

    //Driver program to test above functions
    public static void main(String args[]) {
        BinaryTree tree = new BinaryTree();
        Node T = new Node('a');
        T.left = new Node('b');
        T.right = new Node('d');
        T.left.left = new Node('c');
        T.right.right = new Node('e');

        Node S = new Node('a');
        S.left = new Node('b');
        S.right = new Node('d');
        S.left.left = new Node('c');

        if (tree.isSubtree(T, S)) {
            System.out.println("Yes , S is a subtree of T");
        } else {
            System.out.println("No, S is not a subtree of T");
        }
    }
}

// This code is contributed by Mayank Jaiswal
```

Run on IDE

Output:

```
No: S is NOT a subtree of T
```

Time Complexity: Inorder and Preorder traversals of Binary Tree take O(n) time. The function strstr() can also be implemented in O(n) time using KMP string matching algorithm.

Auxiliary Space: O(n)

Thanks to **Ashwini Singh** for suggesting this method. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

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

107 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)

3.7 Average Difficulty : **3.7/5.0**
Based on **13** vote(s)

Add to TODO List
Mark as DONE

Like Share 22 people like this.

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

107 Comments **GeeksforGeeks** 1 **Login**

Recommend 6 Share Sort by Newest

Join the discussion…

Akshansh Singh • a month ago

There's a problem with this approach.

```
T1 )□□x □□□□T2)□□□□□x□□□ T3)□□□□x
□□□/□□\ □□□□□□□□□□/□\□□□□□□□□/□□\
□□a□□□b□□□□□□□□□c□□b□□□□□□a□□□b
□□□\□□□□□□□□□□□□/□□□□□□□□□/
□□□□c□□□□□□□□□□a□□□□□□□□□c
```

Here,
Inorder of T1 = Inorder of T2
Preorder of T1 = Preorder of T3

Make T2 and T3 subtree of any tree, let's say it tree P.

Now we have to check if T1 is a subtree of P.
Both inorder and preorder of T1 exists in P. So above algo will return true but T1 doesn't exist in P.

• Reply • Share ›

Raj Akshansh Singh • a month ago

Correct. But, I think the assumption in the question is that all nodes are unique. If that is not the assumption, the above solution is not accurate

• Reply • Share ›

Sith_Lord • 3 months ago

check out recursive O(n)

```
bool isSubTreeUtil(btnode *T,btnode *S,btnode *M)

{

if(S == NULL)

return true;

if(T == NULL && M == NULL)

return true;

if(T == NULL || M == NULL)

return false;

if(T->data == M->data){

return isSubTreeUtil(T->lchild,S,M->lchild) && isSubTreeUtil(T->rchild,S,M->rchild);
```

see more

1 • Reply • Share ›

Duke • 3 months ago

Because preorder with nullptr markers uniquely define a tree, I think using the inorder in the solution is redudant.

1 • Reply • Share ›

Raj ➦ Duke • a month ago

Correct. There is no need to use inorder traversal after calculating preorder sequence with some special character used to represent NULL.

• Reply • Share ›

Alex • 4 months ago

How can you prove this solution is correct?
There are some points to be mentioned.
1. preorder/postorder & inorder can uniquely define a tree.
2. if one tree's preorder and inorder traversal is a substring of another tree's preorder and inorder traversal, how do you prove it is a subtree?

2 • Reply • Share ›

Md Akbar Quraishi • 5 months ago

what happens when we add d as b's right child in tree1 and c as b's left child in tree2 else every thing same.......is tree1 subtree of tree2 or not...???

• Reply • Share ›

code down • 6 months ago

@GeeksforGeeks

As pointed out by **@Szilard Mandici**, it seems pre-order with null markers will work.

2 ^ | ⌄ • Reply • Share ›

Holden • 7 months ago

Can someone please share Java version of second solution?

^ | ⌄ • Reply • Share ›

TulsiRam • 8 months ago

Just do:
Find root of the "s" in tree "t", store it.
Now just do: isGivenTreesSame(s, pointer_to_s_in_t)

1 ^ | ⌄ • Reply • Share ›

Satish → TulsiRam • 3 months ago

what if there are multiple roots of 's' in 't' ?

^ | ⌄ • Reply • Share ›

Holden → TulsiRam • 7 months ago

Good one!

^ | ⌄ • Reply • Share ›

sasha • 8 months ago

without using reference variable
inT[m]='\0';
is not working properly. i mean it gives wrong output.

why is that?

^ | ⌄ • Reply • Share ›

sasha • 8 months ago

```
void storeInorder(Node *root, char arr[], int &i)
{
if (root == NULL)
{
arr[i++] = '$';
return;
}
```

what is happening in this part..actually i dont understand the point of passing reference variable .

plz explain.

• Reply • Share ›

jason • 8 months ago

i got a dumb question, is a tree a subtree of itself? If yes, then in-order traversal shouldn't work..

• Reply • Share ›

jason ➦ jason • 8 months ago

test case:

TreeA:

2

/ \

1 3

/

0

TreeB:

1

/ \

0 2

\

3

in order:

A: $0$1$2$3$

B: $0$1$2$3$

• Reply • Share ›

jason ➦ jason • 8 months ago

in TreeB, 3 is the right child of 2, text formatting is wrong in previous post

• Reply • Share ›

sasha • 8 months ago

Why is I passed as refernce in storeinorder n storepreorder function

• Reply • Share ›

Aakash Parashar ➦ sasha • 8 months ago

So that after it NULL can be placed. Think of it as to get the lenth of the array.

• Reply • Share ›

sasha ➦ Aakash Parashar • 8 months ago

i am not able to understand this.

coz refernce variable is used as an alias for a existing variable..right?

so why to use it so that null can be placed ?
can null not be placed without ref variable?

• Reply • Share ›

Akshay Thakare • 9 months ago
Code in C lang

https://ideone.com/O14Qg1

• Reply • Share ›

Abhishek Jaiswal • 9 months ago
@GeeksforGeeks

if we are storing positions of NULL pointers too.
we can uniquely identify each tree with any of the preorder traversals only, we need not to check for both preorder and inorder.
correct me if i am wrong.

http://www.geeksforgeeks.org/s...

2 • Reply • Share ›

Naveen Kumar Badgujar Abhishek Jaiswal • 9 months ago
no we need both the traversals...
see :http://code.geeksforgeeks.org/...

• Reply • Share ›

Abhishek Jaiswal Naveen Kumar Badgujar • 9 months ago
yes your output is correct.

But this problem can be solved using just the preorder traversal and storing the position of NULL values.
Read this post.
http://www.geeksforgeeks.org/s...

If you still have any doubt i can provide you working code using only preorder traversal and NULL pointers.

• Reply • Share ›

Naveen Kumar Badgujar Abhishek Jaiswal • 9 months ago
can u upload ur code

• Reply • Share ›

Naveen Kumar Badgujar Naveen Kumar Badgujar • 9 months ago
just run the code and see the op.

• Reply • Share ›

Naveen Kumar Badgujar ➦ Naveen Kumar Badgujar • 9 months ago

correct me if i am wrong...

^ | ˅ • Reply • Share ›

Gautham Kumaran ➦ Abhishek Jaiswal • 9 months ago

@geeksforgeeks
I can't think of any test case that fails, someone please confirm!

^ | ˅ • Reply • Share ›

Holden ➦ Gautham Kumaran • 7 months ago

Try these tree:

```
                    Node root3 = new Node(10);
                    root3.left = new Node(20);
                    root3.right = new Node(30);
                    root3.left.left = new Node(40);

                    Node root4 = new Node(10);
                    root4.left = new Node(40);
                    root4.right = new Node(30);
```

^ | ˅ • Reply • Share ›

SiSi • 9 months ago

using only one traversal for integer value trees in c++ can be implemented for

```
void chekisTree(struct Node *root,struct Node *root1,stack<node*> &s2)
{
std::stack<node*> s1;
while(1)
{
while(root && root1)
{
s1.push(root);
s2.push(root1);
root=root->left;
root1=root1->left;
}
root=s1.top();
root1=s2.top();
if(root->data!=root1->data)
break;
```

see more

• Reply • Share ›

Vishal Gupta • 9 months ago
Why is there need of taking Inorder and Preorder both, it can also be check by only one order?????
• Reply • Share ›

Tauseef Vishal Gupta • 9 months ago
No bro,in case of only one traversal we will not able to identify a binaty tree.Suppose you have inoder traversal array for tree1(Above example) then we will not be sure that child "c" is left or right child of "a".To make binary tree we need two atleast two traversals(inorder and preorder/postorder).
5 • Reply • Share ›

DS+Algo • 10 months ago
instead of creating different arrays for inorder and preorder, we can use just one.first use that for inorder then for preorder or vice versa
2 • Reply • Share ›

ksk • 10 months ago
use post order and inorder only instead of preorder
• Reply • Share ›

ss • 10 months ago
just need to be assured of my approach
please tell me if i m correct
say if we have two trees s and t
i want to find if s is subtree of t or not
what i can do is .. i will find the occurence of root(s) in t
wherever i find it return it
then just do isidentical on both trees from that root(s) in t starting node
this will take o(n) time and no extra space
• Reply • Share ›

Abhishek kr Yadav ss • 10 months ago
Yes,Your approach is right but complexity in the worst case is O(mn) m-number of nodes in s tree and n is number of node in T tree.
• Reply • Share ›

ss Abhishek kr Yadav • 10 months ago
why is it O(mn)??
• Reply • Share ›

Abhishek kr Yadav ↪ ss • 10 months ago

1
/

1
/
1
/

1
/
7
suppose this is your T tree and your s tree is
1
/
8
in this case -according to your approach
in this type of situation your worst case complexity will be O (mn).

• Reply • Share ›

lucy • 10 months ago

strstr() take O(n^2) for check substring is lies or not......
i think by KMP can be solve O(n).

• Reply • Share ›

guess • a year ago

I don't think it works. BST is a binary tree. Let BST_S = {10,4,16} and BST_T = {26,10,4,6,3,16}, so S is a subtree of T. But, Inorder S = {4,10,16} and Inorder T = {3,4,6,10,16,26} and S is not subarray of T. Preorder S = {10,4 ,16} and Preorder T= {26,10,4,3,6,16}. Again, S is not subarray of T. So, this don't work because BST_S is a subtree of BST_T..

• Reply • Share ›

guess ↪ guess • a year ago

My bad, misunderstood the question..

• Reply • Share ›

chanduthedev • a year ago

why do we need inorder and preorder to find the subtree. Just preorder is fine to rite? any issues in finding only preorder ? please comment

• Reply • Share ›

Manish M Berwani ↪ chanduthedev • 10 months ago

Well preorder/postorder alone are not enough to identify a unique tree. For eg...

Take big tree as

root = A
A->left = B
B->right = C

And small tree as
root = A
A->right = B

U will find preoder of big tree as ABC
small tree as AB
Still small tree is not subtree of big tree as they r not identical!

• Reply • Share ›

Sushil Kumar Maurya Manish M Berwani • 9 months ago

But by only using inorder we can find. The inorder of first tree is BCA and second tree is AB and AB does not belong to BCA in sequence so we can easily say that this is not sub tree. Please correct me if i am wrong.

• Reply • Share ›

SANTOSH KUMAR MISHRA • a year ago

can any one tell " what is the need of putting arr[i++] = '$' " ???

1 • Reply • Share ›

iamshobhit SANTOSH KUMAR MISHRA • a year ago

For the case which has been mentioned in edit section.Assign something for a null pointer.

1 • Reply • Share ›

SANTOSH KUMAR MISHRA iamshobhit • a year ago

Thanks

• Reply • Share ›

aaa • a year ago

Simple soltuion:
1) find same node value
2) check identical or not

http://ideone.com/fNVCQe

• Reply • Share ›

tintin.92 • a year ago

Having special character for NULLs in the traversal array uniquely identifies the tree as we can figure out all the leaf nodes. So we need to do substring match in any one we can figure out all the leaf nodes. So we need to do substring match in any one traversal only and we would be done.

Please correct me if I am wrong.

• Reply • Share ›

Venu Gopal • a year ago

Another O(n) approach. Code works. Please correct me if this is not good.

Below is my code which first searches for root node of subtree S in the parent tree T.
Then it uses recursive preorder traversal to match individual nodes in S with nodes in T.
This is same as checking for mirror image tree.
I have also handled the case of Edit section(extra element after S in T) in the above theory[check line 94 and 95 in code]
http://ideone.com/92HNjJ

1 • Reply • Share ›

Load more comments