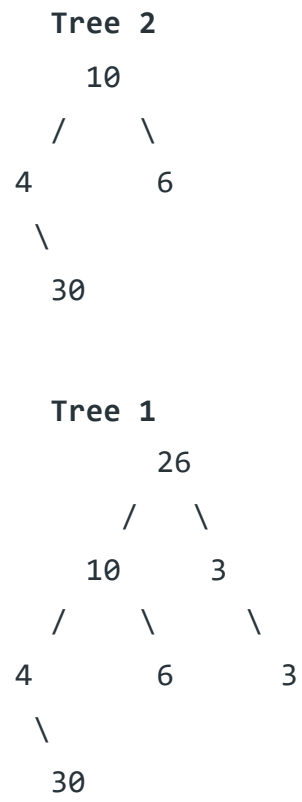
Check if a binary tree is subtree of another binary tree | Set 1

Difficulty Level : Medium • Last Updated : 17 Jun, 2022

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, tree S is a subtree of tree T.



Following is the implementation for this.

C++

```
// C++ program to check if binary tree
// is subtree of another binary tree
#include<bits/stdc++.h>
using namespace std;

/* A binary tree node has data,
left child and right child */
class node
{
    public:
    int data;
    node* left;
    node* right;
};

/* A utility function to check
whether trees with roots as root1 and
root2 are identical or not */
bool areIdentical(node * root1, node *root2)
{
    /* base cases */
    if (root1 == NULL && root2 == NULL)
        return true;

    if (root1 == NULL || root2 == NULL
```

```
    same and data of left and right
    subtrees are also same */
    return (root1->data == root2->data &&
            areIdentical(root1->left, root2->left) &&
            areIdentical(root1->right, root2->right) );
}


/* 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;

    /* Check the tree with root as current node */
    if (areIdentical(T, S))
        return true;

    /* If the tree with root as current
    node doesn't match then try left
    and right subtrees one by one */
    return isSubtree(T->left, S) ||
        isSubtree(T->right, S);
}


/* Helper function that allocates
a new node with the given data
and NULL left and right pointers. */
node* newNode(int data)
{
    node* Node = new node();
    Node->data = data;
    Node->left = NULL;
    Node->right = NULL;
    return(Node);
}

/* Driver code*/
int main()
{
    // TREE 1
    /* Construct the following tree
            26
            / \
```

```
    \
        30
    */
    node *T = newNode(26);
```


```
    T->left->right     = newNode(6);

    // TREE 2
    /* Construct the following tree
        10
        / \
    4 6
    \
        30
    */
    node *S = newNode(10);
    S->right     = newNode(6);
    S->left     = newNode(4);
    S->left->right = newNode(30);


    if (isSubtree(T, S))
        cout << "Tree 2 is subtree of Tree 1";
    else
        cout << "Tree 2 is not a subtree of Tree 1";

    return 0;
}

// This code is contributed by rathbhupendra
```

C

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

/* A binary tree node has data, left child and right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};
```

```
{
    /* base cases */
    if (root1 == NULL && root2 == NULL)
        return true;

    if (root1 == NULL || root2 == NULL)
        return false;

    /* Check if the data of both roots is same and data of left and right
       subtrees are also same */
    return (root1->data == root2->data   &&
            areIdentical(root1->left, root2->left) &&
            areIdentical(root1->right, root2->right) );
}


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

    if (T == NULL)
        return false;

    /* Check the tree with root as current node */
    if (areIdentical(T, S))
        return true;

    /* If the tree with root as current node doesn't match then
       try left and right subtrees one by one */
    return isSubtree(T->left, S) ||
           isSubtree(T->right, S);
}


/* 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);
}

/* Driver program to test above functi
```

```
    /* Construct the following tree
              26
            /   \
          10     3
        /    \     \
      4      6      3
       \
        30
    */
    struct node *T        = newNode(26);
    T->right              = newNode(3);
    T->right->right       = newNode(3);
    T->left               = newNode(10);
    T->left->left         = newNode(4);
    T->left->left->right  = newNode(30);
    T->left->right        = newNode(6);

    // TREE 2
    /* Construct the following tree
          10
        /    \
      4      6
       \
        30
    */
    struct node *S    = newNode(10);
    S->right          = newNode(6);
    S->left           = newNode(4);
    S->left->right    = newNode(30);


    if (isSubtree(T, S))
        printf("Tree 2 is subtree of Tree 1");
    else
        printf("Tree 2 is not a subtree of Tree 1");

    getchar();
    return 0;
}
```

Java

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

/ A binary tree node
lass Node
{
    int data;
```

```
    {
        data = item;
        left = right = nextRight = null;
    }
}

class BinaryTree
{
    Node root1,root2;

    /* A utility function to check whether trees with roots as root1 and
       root2 are identical or not */
    boolean areIdentical(Node root1, Node root2)
    {

        /* base cases */
        if (root1 == null && root2 == null)
            return true;

        if (root1 == null || root2 == null)
            return false;

        /* Check if the data of both roots is same and data of left and right
           subtrees are also same */
        return (root1.data == root2.data
                && areIdentical(root1.left, root2.left)
                && areIdentical(root1.right, root2.right));
    }

    /* 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;

        /* Check the tree with root as current node */
        if (areIdentical(T, S))
            return true;

        /* If the tree with root as current node doesn't match then
           try left and right subtrees one by one */
        return isSubtree(T.left, S)
                || isSubtree(T.right, S);
    }

    public static void main(String ar
```

```
        // TREE 1
        /* Construct the following tree
              26
             /   \
            10     3
           /    \     \
          4      6      3
           \
            30  */

        tree.root1 = new Node(26);
        tree.root1.right = new Node(3);
        tree.root1.right.right = new Node(3);
        tree.root1.left = new Node(10);
        tree.root1.left.left = new Node(4);
        tree.root1.left.left.right = new Node(30);
        tree.root1.left.right = new Node(6);

        // TREE 2
        /* Construct the following tree
           10
         /    \
         4      6
          \
          30  */

        tree.root2 = new Node(10);
        tree.root2.right = new Node(6);
        tree.root2.left = new Node(4);
        tree.root2.left.right = new Node(30);

        if (tree.isSubtree(tree.root1, tree.root2))
            System.out.println("Tree 2 is subtree of Tree 1 ");
        else
            System.out.println("Tree 2 is not a subtree of Tree 1");
    }
}

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

Python3

```
# Python program to check binary tree is a subtree of
# another tree

# A binary tree node
class Node:
```

```
        self.left = None
        self.right = None

# A utility function to check whether trees with roots
# as root 1 and root2 are indetical or not
def areIdentical(root1, root2):

    # Base Case
    if root1 is None and root2 is None:
        return True
    if root1 is None or root2 is None:
        return False

    # Check fi the data of both roots is same and data of
    # left and right subtrees are also same
    return (root1.data == root2.data and
            areIdentical(root1.left , root2.left)and
            areIdentical(root1.right, root2.right)
            )

# This function returns True if S is a subtree of T,
# otherwise False
def isSubtree(T, S):

    # Base Case
    if S is None:
        return True

    if T is None:
        return False

    # Check the tree with root as current node
    if (areIdentical(T, S)):
        return True

    # IF the tree with root as current node doesn't match
    # then try left and right subtree one by one
    return isSubtree(T.left, S) or isSubtree(T.right, S)


# Driver program to test above function

""" TREE 1
     Construct the following tree
               26
             /   \
           10     3
         /    \     \
        4      6      3
         \
```

```
T = Node(26)
T.right = Node(3)
T.right.right  = Node(3)
T.left = Node(10)
T.left.left = Node(4)
T.left.left.right = Node(30)
T.left.right = Node(6)

""" TREE 2
     Construct the following tree
          10
        /    \
      4       6
       \
        30
    """
S = Node(10)
S.right = Node(6)
S.left = Node(4)
S.left.right = Node(30)

if isSubtree(T, S):
    print ("Tree 2 is subtree of Tree 1")
else :
    print ("Tree 2 is not a subtree of Tree 1")

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

C#

```
// C# program to check if binary tree
// is subtree of another binary tree
using System;

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

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

public class BinaryTree
```

```
/* A utility function to check whether
    trees with roots as root1 and
    root2 are identical or not */
bool areIdentical(Node root1, Node root2)
{

    /* base cases */
    if (root1 == null && root2 == null)
        return true;

    if (root1 == null || root2 == null)
        return false;

    /* Check if the data of both roots is
    same and data of left and right
    subtrees are also same */
    return (root1.data == root2.data
            && areIdentical(root1.left, root2.left)
            && areIdentical(root1.right, root2.right));
}

/* 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;

    /* Check the tree with root as current node */
    if (areIdentical(T, S))
        return true;

    /* If the tree with root as current
      node doesn't match then try left
      and right subtrees one by one */
    return isSubtree(T.left, S)
            || isSubtree(T.right, S);
}

// Driver code
public static void Main()
{
    BinaryTree tree = new BinaryTree();

    // TREE 1
    /* Construct the following tre
```

```
        / \ \
        4 6 3
        \
            30 */

        tree.root1 = new Node(26);
        tree.root1.right = new Node(3);
        tree.root1.right.right = new Node(3);
        tree.root1.left = new Node(10);
        tree.root1.left.left = new Node(4);
        tree.root1.left.left.right = new Node(30);
        tree.root1.left.right = new Node(6);

        // TREE 2
        /* Construct the following tree
        10
        / \
        4 6
        \
        30 */

        tree.root2 = new Node(10);
        tree.root2.right = new Node(6);
        tree.root2.left = new Node(4);
        tree.root2.left.right = new Node(30);

        if (tree.isSubtree(tree.root1, tree.root2))
            Console.WriteLine("Tree 2 is subtree of Tree 1 ");
        else
            Console.WriteLine("Tree 2 is not a subtree of Tree 1");
    }
}

/* This code is contributed by Rajput-Ji*/
```

Javascript

```
<script>

// JavaScript program to check if binary tree
// is subtree of another binary tree

// A binary tree node
class Node {
    constructor(val) {
        this.data = val;
        this.left = null;
        this.right = null;
```

```
var  root1,root2;

    /* A utility function to check whether
       trees with roots as root1 and
       root2 are identical or not */
    function areIdentical(root1,  root2)
    {

        /* base cases */
        if (root1 == null && root2 == null)
            return true;

        if (root1 == null || root2 == null)
            return false;

        /* Check if the data of both roots
           is same and data of left and right
           subtrees are also same */
        return (root1.data == root2.data
                && areIdentical(root1.left, root2.left)
                && areIdentical(root1.right, root2.right));
    }

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

        if (T == null)
            return false;

        /* Check the tree with root as current node */
        if (areIdentical(T, S))
            return true;

        /* If the tree with root as
           current node doesn't match then
           try left and right subtrees one by one */
        return isSubtree(T.left, S)
                || isSubtree(T.right, S);
    }


        // TREE 1
        /* Construct the following tree
              26
```

```
          4      6      3
           \
            30  */

        root1 = new Node(26);
        root1.right = new Node(3);
        root1.right.right = new Node(3);
        root1.left = new Node(10);
        root1.left.left = new Node(4);
        root1.left.left.right = new Node(30);
        root1.left.right = new Node(6);

        // TREE 2
        /* Construct the following tree
           10
         /    \
         4      6
          \
          30  */

        root2 = new Node(10);
        root2.right = new Node(6);
        root2.left = new Node(4);
        root2.left.right = new Node(30);

        if (isSubtree(root1, root2))
            document.write("Tree 2 is subtree of Tree 1 ");
        else
            document.write("Tree 2 is not a subtree of Tree 1");


// This code is contributed by todaysgaurav

</script>
```

Output:

```
Tree 2 is subtree of Tree 1
```

Time Complexity: Time worst-case complexity of above solution is O(mn) where m and n are number of nodes in given two trees.

Auxiliary space: O(n)

We can solve the above problem in O(n) time. Please refer Check if a binary tree is subtree of another binary tree | Set 2 for solution.

Previous

Next

RECOMMENDED ARTICLES

Page : 1 2 3

01 Check if a binary tree is subtree of another binary tree | Set 2
26, Jul 14

02 Check if a Binary tree is Subtree of another Binary tree | Set 3
24, Feb 22

03 Check if a binary tree is subtree of another binary tree using preorder traversal : Iterative
08, Apr 20

04 Check if the given Binary Tree have a Subtree with equal no of 1's and 0's
22, Feb 19

05 Finding if a node X is present in subtree of another node Y or vice versa for Q queries
10, Feb 22

06 Count of nodes in given N-ary tree such that their subtree is a Binary Tree
26, Nov 21

07 Duplicate subtree in Binary Tree | SET 2
30, Aug 19

08 Find the largest BST subtree in a given Binary Tree | Set 3
17, Feb 12

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

Improved By : SrivathsanAravamudan, Rajput-Ji, rathbhupendra, todaysgaurav, volumezero9786, amartyaghoshgfg, germanshephered48, sumitgumber28, hardikkoriintern

Article Tags : Adobe, Amazon, Cavisson System, MakeMyTrip, Microsoft, Morgan Stanley, OYO Rooms, SAP Labs, Tree

Practice Tags : Morgan Stanley, Amazon, Microsoft, OYO Rooms, MakeMyTrip, Adobe, SAP Labs, Cavisson System, Tree

Improve Article | Report Issue