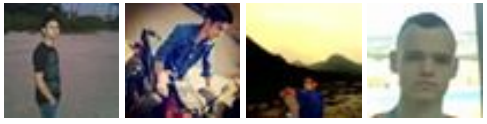
Check for Identical BSTs without building the trees

Given two arrays which represent a sequence of keys. Imagine we make a Binary Search Tree (BST) from each array. We need to tell whether two BSTs will be identical or not without actually constructing the tree.

Examples
For example, the input arrays are {2, 4, 3, 1} and {2, 1, 4, 3} will construct the same tree

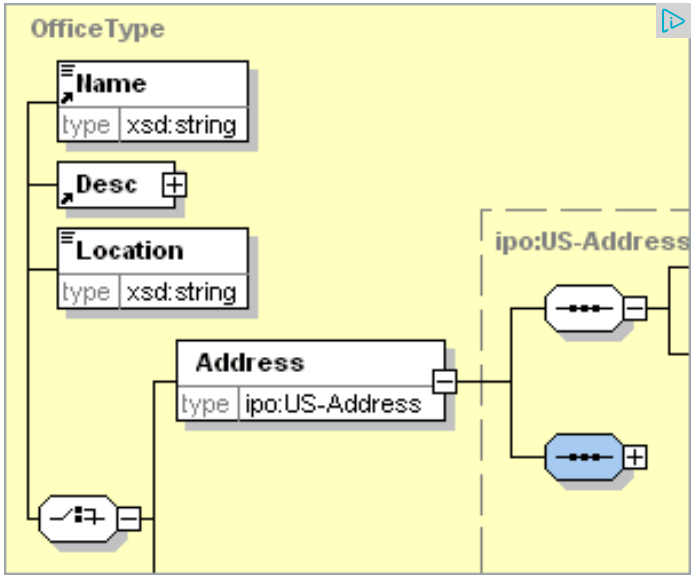
```
 Let the input arrays be a[] and b[]

Example 1:
a[] = {2, 4, 1, 3} will construct following tree.
   2
 /  \
1    4
    /
   3
b[] = {2, 4, 3, 1} will also also construct the same tree.
   2
 /  \
1    4
    /
   3
So the output is "True"

Example 2:
a[] = {8, 3, 6, 1, 4, 7, 10, 14, 13}
```

```
b[] = {8, 10, 14, 3, 6, 4, 1, 7, 13}

They both construct the same following BST, so output is "True"
           8
         /   \
       3       10
     /  \        \
    1     6       14
        /   \     /
       4     7   13
```

Solution:
According to BST property, elements of left subtree must be smaller and elements of right subtree must be greater than root.
Two arrays represent sane BST if for every element x, the elements in left and right subtrees of x appear after it in both arrays. And same is true for roots of left and right subtrees.
The idea is to check of if next smaller and greater elements are same in both arrays. Same properties are recursively checked for left and right subtrees. The idea looks simple, but implementation requires checking all conditions for all elements. Following is an interesting recursive implementation of the idea.

```
// A C program to check for Identical BSTs without building the trees
#include<stdio.h>
#include<limits.h>
#include<stdbool.h>

/* The main function that checks if two arrays a[] and b[] of size n c
  same BST. The two values 'min' and 'max' decide whether the call is
  left subtree or right subtree of a parent element. The indexes i1 an
  the indexes in (a[] and b[]) after which we search the left or right
  Initially, the call is made for INT_MIN and INT_MAX as 'min' and 'ma
  respectively, because root has no parent.
  i1 and i2 are just after the indexes of the parent element in a[] an
bool isSameBSTUtil(int a[], int b[], int n, int i1, int i2, int min, i
{
   int j, k;

   /* Search for a value satisfying the constraints of min and max in
      b[]. If the parent element is a leaf node then there must be som
      elements in a[] and b[] satisfying constraint. */
```



Popular Posts

```
    for (j=i1; j<n; j++)
        if (a[j]>min && a[j]<max)
            break;
    for (k=i2; k<n; k++)
        if (b[k]>min && b[k]<max)
            break;

    /* If the parent element is leaf in both arrays */
    if (j==n && k==n)
        return true;

    /* Return false if any of the following is true
       a) If the parent element is leaf in one array, but non-leaf in o
       b) The elements satisfying constraints are not same. We either s
          for left child or right child of the parent element (decinded
          and max values). The child found must be same in both arrays
    if (((j==n)^(k==n)) || a[j]!=b[k])
        return false;

    /* Make the current child as parent and recursively check for left
       subtrees of it. Note that we can also pass a[k] in place of a[j]
       are both are same */
    return isSameBSTUtil(a, b, n, j+1, k+1, a[j], max) &&  // Right Sub
           isSameBSTUtil(a, b, n, j+1, k+1, min, a[j]);    // Left Subt
}

// A wrapper over isSameBSTUtil()
bool isSameBST(int a[], int b[], int n)
{
    return isSameBSTUtil(a, b, n, 0, 0, INT_MIN, INT_MAX);
}

// Driver program to test above functions
int main()
{
    int a[] = {8, 3, 6, 1, 4, 7, 10, 14, 13};
    int b[] = {8, 10, 14, 3, 6, 4, 1, 7, 13};
    int n=sizeof(a)/sizeof(a[0]);
    printf("%s\n", isSameBST(a, b, n)?
                "BSTs are same":"BSTs not same");
    return 0;
}
```

Output:

```
BSTs are same
```

BSTs are same

This article is compiled by **Amit Jain**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

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

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 29 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 49 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 49 minutes 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) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

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

50 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Guest • 2 months ago

Another method I could think of is: (i is A[ ]'s index and j is B[ ]'s index)

* Start with i = 0 and j = 0. Check if the first element is same in both arrays (cc different structure

* Find the next smaller element in both the arrays and compare them. If they're structure

* Traverse the array again but now keep checking if the next larger element in different.

We're traversing the array twice so it's O(N). If you guys find any bugs, do con

Reply • Share ›

AlienOnEarth Guest • 3 days ago

will not work if preorder and postorder traversals are given for BST

Reply • Share ›

Guest • 4 months ago

I think the following should be an easier implementation:

bool IsIdenticalBST(int arr1[], int arr2[], int num_elems1, int num_elems2) {

```
if (num_elems1 != num_elems2) return false;

if (!num_elems1 && !num_elems2) return true;

if (arr1[0] == arr2[0]) return IsIdenticalBST(arr1+1, arr2+1, num_elems1-1, nu

int i = 1;

for (; i < num_elems2; i++) {

if (arr1[0] == arr2[i])

break;

}

if (!IsIdenticalBST(arr1, arr2+i, i, i) || !IsIdenticalBST(arr1+i, arr2, i, i)) return fal

return IsIdenticalBST(arr1+2*i+1, arr2+2*i+1, num_elems1-2*i-1, num_elems

}
```

• Reply • Share ›

anonymous • 5 months ago

One simple approach that i think could work.
As we know that, we have a unique BST, given the inorder traversal of BST, w
Sort both arrays, in O(n log n) + O(n log n) plus an O(n) traversal to check wh
not = O(n log n)

• Reply • Share ›

akki ➦ anonymous • 2 months ago

Consider the trees:
2
\

\
4

and

3
/ \
2 4

They would have the same inorder traversal but different structures.

Anonymous anonymous • 5 months ago

i think you solution will fail for below input..

a[] = {1,2,3}, b[]= {2,1,3}

after sorting both arrays will be same , but both trees are not same..

please , correct me if i am wrong...

Sumit Anonymous • 5 months ago

Tree for a will be (inorder travesal) :
2
/ \
1 3

Tree for b will be (preorder travesal) :
http://www.geeksforgeeks.org/c...
2
/ \
1 3

Both Tree appears to be same. Can you please tell us y sorting

anonymous Anonymous • 5 months ago

I think that, they should make the same tree. Only one tree pos
2
1 3

Tell me, how exactly is the array element order relevant?
Is the following correct?
Okay, given the array {3,1,2,4}, is it that first the node with data with '1' and then node with '2' and then '4'?
that is,

```
1.)      3
2.)      3
    1
3.)      3
    1
       2
4.)      3
    1        4
       2
```

If this is the way the BST is constructed from the array, then m

Anonymous anonymous • 5 months ago

yes, i think that is the way tree is constructed... it takes
creating the tree...
in that case your solution fails.

• Reply • Share ›

Anonymous ➚ Anonymous • 5 months ago

I think that, they should make the same tree. Only one tree pos
2
1 3

Tell me, how exactly is the array element order relevant?
Is the following correct?
Okay, given the array {3,1,2,4}, is it that first the node with data
with '1' and then node with '2' and then '4'?
that is,
1.) 3
2.) 3
1
3.) 3
1
2
4.) 3
1 4
2
If this is the way the BST is constructed from the array, then m

• Reply • Share ›

wasseypuriyan • 7 months ago

pure awesomeness \m/

4 • Reply • Share ›

rahul23 • 7 months ago

@AMIT

Please tell time complexity...is it n^2??

sumit dey • 8 months ago

The above code is not really going to work take for example :

Ex 1: Tree are T1={8}, T2={9} => it will return true.

I Think this logic has to be modified :

/* If the parent element is leaf in both arrays */

if (j==n && k==n) // needs to add whether a[j]==b[k].

return true;

dave sumit dey • 5 months ago

The check is done when j=0 and i=0, this will give the answer as false

if (((j==n)^(k==n)) || a[j]!=b[k])
return false;

Shradha Agrawal • 8 months ago

1.sort two sequences and check whether they are same or not.
2.if not , output false and stop.
3. else recursively check whether preorder of both is same or not.
Code for 3rd step is as:

```
int isIdentical(int in[],int seq1[],int seq2[] , int size)
{
int z_left , z_right;
if(seq1[0] != seq2[0])
return 0;
```

```
int loc = bsearch(in , seq1[0],0,size-1 );
if(loc != 0)
z_left = isIdentical(in,seq1 + 1, seq2 + 1 , loc);
else
z_left = 1;

if(loc != size-1)
z_right = isIdentical(in+loc+1,seq1 + loc + 1, seq2 + loc+ 1 , size-loc-1);
else
z_right = 1;
if(z_left && z_right)
return 0;
else
return 1;

}
```


bhavneet • 8 months ago

i have a o(n2) approach. Traverse the array1 and array2. Find minimum numb
number to the left of current number and find max number which is less then c
number in both the arrays. The min max for same number should be same in

141093 • 8 months ago

a[] = {8, 3, 6, 1, 4, 7, 10, 14, 13}
b[] = {8, 10, 14, 3, 6, 4, 1, 7, 13}

sorted array : 1, 3, 4, 6, 7, 8, 10, 13, 14
elements index in array a[]: 3, 1, 4, 2, 5, 0, 6, 8, 7
elements index in array b[]: 6, 3, 5, 4, 7, 0, 1, 8, 2

for a & b to represent same bts, occurance of indexes must be similar

(3>1 && 6>3) && (1<4 && 3<5) && (4>2 && 5>4).....

4 ^ | v • Reply • Share ›

Nitin Sharma 141093 • 6 months ago

Please will u explain it.....????????????

^ | v • Reply • Share ›

Anil • 8 months ago

You can use the stack based approach for finding the next greater and smalle
Complexity : O(n) for both space and time.

1 ^ | v • Reply • Share ›

Anil • 8 months ago

I think we can easily do it in O(n) using the stack approach for finding the next
and then simply comparing them . However, it will require O(n) space.

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

^ | v • Reply • Share ›

Chinnu Pavan • 9 months ago

bool isSame(int a[], int b[], int n).
{
int k=0, t=0;.
for(i=0;i<n;i++).
{.

for(j=0;j<n;j++).

{.

if(a[i]==b[j]).

{.

t=1;.

break;.

}.

see more

Reply • Share ›

gps • 9 months ago

Can't we just sort the elements and compare the arrays....because both the a identical ???

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

Reply • Share ›

dex gps • 9 months ago

Did you understand now ?
if(yes) // ignore this comment
exit(0);
else
{
int a[5]= {1,2,3,4,5}
int b[5]= {2,3,4,1,5}
Both arrays are same if sorted , but the roots of the trees to be constru
construction of tree.Think
}

Reply • Share ›

Just make one more check for the first element !

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

• Reply • Share ›

Manasa Kompella • 9 months ago

but where are we modifying min n max.

• Reply • Share ›

Ritesh Garg • 9 months ago

yes it is.....
worst case
1,2,3,4,5
1,2,3,4,5

• Reply • Share ›

prama12 • 9 months ago

I think- this is a little easier to understand.
The trees would be the same if
1. The root is the same
2. The left and right subtrees are both the same
[sourcecode language="JAVA"]
public boolean checkForIdenticalBST(ArrayList<Integer> arr1,ArrayList<Intege

if(arr1.size() != arr2.size())
return false ;

if(arr1.isEmpty() && arr2.isEmpty())
return true;

// the roots need to be the same

if(arr1.get(0)!= arr2.get(0))
return false;

ArrayList<Integer> left1 = new ArrayList<Integer>();
ArrayList<Integer> right1= new ArrayList<Integer>();

see more

• Reply • Share ›

Reflexion • 9 months ago

Why not just compare the sorted arrays.. that shall give the answer as well rig

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

• Reply • Share ›

dex ➚ Reflexion • 9 months ago

Did you understand now ?
if(yes) // ignore this comment
exit(0);
else
{
int a[5]= {1,2,3,4,5}
int b[5]= {2,3,4,1,5}
Both arrays are same if sorted , but the roots of the trees to be constru
construction of tree.Think
}

• Reply • Share ›

Pandian Raju • 9 months ago

You are wrong. Try to recurse through the steps. First 8 is chosen. Then 3 is c
the arrays. So, when 6 comes, the range will be 6,8 for the right. So, 10 will nc

Reply • Share ›

Vishal Goel • 10 months ago

space complexity is O(1). What is time complexity? Is it O(n^2)?

Reply • Share ›

Siddharth Rajpal • 10 months ago

Hey, I am not so sure if this approach is correct, for example:
If we have array1: 8,3,6,1,4,10,7,14,13.
array2: 8,3,6,1,4,7,10,14,13.
The two array represent the same BST tree, however the next greater elemen
second. Could you please solve my doubt?

Reply • Share ›

startre • 10 months ago

//Another simple Recursive solution

```
#include<stdio.h>

int is_Same_Tree(int *arr,int len1,int *brr,int len2)
{
     if(len1!=len2 || arr[0]!=brr[0])
     {
                  return 0;
     }
     if(len1==1 && arr[0]==brr[0])
                return 1;
     if(len1==1 && arr[0]!=brr[0])
                return 0;
     if(len1==0 || len2==0)
                return 1;
```

```
    int flag1,flag2;
```

see more

• Reply • Share ›

beginner • 10 months ago

wat is the INT_MIN and INT_MAX values here

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

1 • Reply • Share ›

Karthick • 10 months ago

```
a[] = {8, 3, 6, 1, 4, 7, 10, 14, 13}
b[] = {8, 10, 14, 13, 3, 6, 4, 1, 7}

These two are identical bsts but i guess ur implementation will give

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

1 • Reply • Share ›

GeeksforGeeks ➦ Karthick • 10 months ago

The implementation seems to be working fine for your example. Pleas

• Reply • Share ›

AMIT ➦ Karthick • 10 months ago

No,it is giving correct result..please check

• Reply • Share ›

Yeah.. Sorry I thought n represents last index!

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

Reply • Share ›

ministar • 10 months ago

```
O(nlogn)
def areEquals(a1,a2):
        l1=len(a1)
        l2=len(a2)
        if l1!=l2:
                return 0;
        if (l1==l2 and l1==0):
                return 1;
        if (a1[0]!=a2[0]):
                return 0;
        g1=[]
        g2=[]
        s1=[]
        s2=[]
        dict1={}
        dict2={}
        for i in range(1,l1):
                if(a1[i]>a1[0]):
```

see more

Reply • Share ›

Bala Sravan ministar • 10 months ago

Explain the algo please.....

Reply • Share ›

Xiaoge Yuan • 10 months ago

brilliant!

• Reply • Share ›

Santosh Kumar • 10 months ago

have posted iterative approach.

• Reply • Share ›

san4net • 10 months ago

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

package com.me.ds;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class SameBST<T> {

        /**<p>
         * Getting all the children of the BST, like the 0 index will
         * element like wise for each element in the array.
         *
         * </p>
         * @param inputArray
         * @return
         */
```

see more

• Reply • Share ›

anonymous • 10 months ago

i think there should be a problem in line

return isSameBSTUtil(a, b, n, j+1, k+1, a[j], max)&&// Left Subtree
isSameBSTUtil(a, b, n, j+1, k+1, min, a[j]); //Right Subtree

i think first call is for right subtree n second one is for left subtree .

admin explain if i m wrong plz.

Reply • Share ›

GeeksforGeeks anonymous • 10 months ago

Thanks for pointing this out. We have corrected the comments.

Reply • Share ›

gr81 • 10 months ago

is it possible, if I check the first element to both the array are same, i guess an
same BST.

can you elaborate the justification with another 2 array having different elemen

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

Reply • Share ›

Manish gr81 • 10 months ago

having the 1st element same is the necessary condition but not the su

Algo is recursive.we are checking for each element not just the root. Tl

If we could use the info of the previous found position of just max and r

Reply • Share ›

Time complexity for the given algo is O(n^2) can you explain ho

 • Reply • Share ›

AMIT gr81 • 10 months ago

Element at index 0 must be same

But if element at index 0 is same, this doesn't mean,both BSTs are eq

For example,try for

a[]={8,3,1,6}

b[]={8,1,3,6}

 • Reply • Share ›

AMIT AMIT • 10 months ago

By equal,i mean identical

 • Reply • Share ›

 Subscribe Add Disqus to your site