Two nodes of a BST are swapped, correct the BST

Two of the nodes of a Binary Search Tree (BST) are swapped. Fix (or correct) the BST.

```
Input Tree:
         10
        /  \
       5    8
      / \
     2   20

In the above tree, nodes 20 and 8 must be swapped to fix the tree.
Following is the output tree
         10
        /  \
       5    20
      / \
     2   8
```

The inorder traversal of a BST produces a sorted array. So a **simple method** is to store inorder traversal of the input tree in an auxiliary array. Sort the auxiliary array. Finally, insert the auxiilary array elements back to the BST, keeping the structure of the BST same. Time complexity of this method is O(nLogn) and auxiliary space needed is O(n).

We can solve this in O(n) time and with a single traversal of the given BST. Since inorder traversal of BST is always a sorted array, the problem can be reduced to a problem where two elements of a sorted array are swapped. There are two cases that we need to handle:

Google™ Custom Search

GeeksforGeeks

52,731 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

1. The swapped nodes are not adjacent in the inorder traversal of the BST.

```
For example, Nodes 5 and 25 are swapped in {3 5 7 8 10 15 20 25}.
The inorder traversal of the given tree is 3 25 7 8 10 15 20 5
```

If we observe carefully, during inorder traversal, we find node 7 is smaller than the previous visited node 25. Here save the context of node 25 (previous node). Again, we find that node 5 is smaller than the previous node 20. This time, we save the context of node 5 (current node). Finally swap the two node's values.

2. The swapped nodes are adjacent in the inorder traversal of BST.

```
 For example, Nodes 7 and 8 are swapped in {3 5 7 8 10 15 20 25}.
 The inorder traversal of the given tree is 3 5 8 7 10 15 20 25
```

Unlike case #1, here only one point exists where a node value is smaller than previous node value. e.g. node 7 is smaller than node 8.

How to Solve? *We will maintain three pointers, first, middle and last. When we find the first point where current node value is smaller than previous node value, we update the first with the previous node & middle with the current node. When we find the second point where current node value is smaller than previous node value, we update the last with the current node. In case #2, we will never find the second point. So, last pointer will not be updated. After processing, if the last node value is null, then two swapped nodes of BST are adjacent*.

Following is C implementation of the given code.

```
// Two nodes in the BST's swapped, correct the BST.
#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, *right;
};

// A utility function to swap two integers
void swap( int* a, int* b )
```

Popular Posts
All permutations of a given string
Memory Layout of C Programs
Understanding "extern" keyword in C
Median of two sorted arrays
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

```
{
    int t = *a;
    *a = *b;
    *b = t;
}

/* 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);
}

// This function does inorder traversal to find out the two swapped no
// It sets three pointers, first, middle and last.  If the swapped nod
// adjacent to each other, then first and middle contain the resultant
// Else, first and last contain the resultant nodes
void correctBSTUtil( struct node* root, struct node** first,
                     struct node** middle, struct node** last,
                     struct node** prev )
{
    if( root )
    {
        // Recur for the left subtree
        correctBSTUtil( root->left, first, middle, last, prev );

        // If this node is smaller than the previous node, it's violat
        // the BST rule.
        if (*prev && root->data < (*prev)->data)
        {
            // If this is first violation, mark these two nodes as
            // 'first' and 'middle'
            if ( !*first )
            {
                *first = *prev;
                *middle = root;
            }

            // If this is second violation, mark this node as last
            else
                *last = root;
        }
        // Mark this node as previous
        *prev = root;

        // Recur for the right subtree
        correctBSTUtil( root->right, first, middle, last, prev );
    }
}

// A function to fix a given BST where two nodes are swapped.  This
// function uses correctBSTUtil() to find out two nodes and swaps the
// nodes to fix the BST
void correctBST( struct node* root )
{
    // Initialize pointers needed for correctBSTUtil()
    struct node *first, *middle, *last, *prev;
    first = middle = last = prev = NULL;

    // Set the poiters to find out two nodes
    correctBSTUtil( root, &first, &middle, &last, &prev );

    // Fix (or correct) the tree
    if( first && last )
        swap( &(first->data), &(last->data) );
    else if( first && middle ) // Adjacent nodes swapped
        swap( &(first->data), &(middle->data) );

    // else nodes have not been swapped, passed tree is really BST.
}

/* A utility function to print Inoder traversal */
void printInorder(struct node* node)
{
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%d ", node->data);
    printInorder(node->right);
}

/* Driver program to test above functions*/
int main()
{
    /*    6
        /  \
       10    2
      / \   / \
     1   3 7   12
```

Recent Comments

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

Backtracking | Set 7 (Sudoku) · 35 minutes ago

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

Flipkart Interview | Set 6 · 55 minutes ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 55 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 · 3 hours ago

```
      10 and 2 are swapped
     */

    struct node *root = newNode(6);
    root->left        = newNode(10);
    root->right       = newNode(2);
    root->left->left  = newNode(1);
    root->left->right = newNode(3);
    root->right->right = newNode(12);
    root->right->left = newNode(7);

    printf("Inorder Traversal of the original tree \n");
    printInorder(root);

    correctBST(root);

    printf("\nInorder Traversal of the fixed tree \n");
    printInorder(root);

    return 0;
}
```

Output:

```
Inorder Traversal of the original tree
1 10 3 6 7 2 12
Inorder Traversal of the fixed tree
1 2 3 6 7 10 12
```

Time Complexity: O(n)

See this for different test cases of the above code.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Better Than Hadoop.
HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.
LexisNexis
Learn More

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

2 Tweet 0 0

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

34 Comments **GeeksforGeeks**

Sort by Newest

Join the discussion

AlienOnEarth • 3 days ago

@Geeksforgeeks:

Another simpler solution:

1.) use isBST routine to check whether the given tree is BST.

2.) if so, do not fix tree.

3.) if not BST, initialise 2 pointers first and last to nodes to be fixed.

4.) fix BST and print.

C program for the above algorithm is as below:

// A program to check if a given binary tree is complete or not

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

see more

Reply • Share ›

abubakar • 2 months ago

struct node *root = newNode(15);
root->left = newNode(6);
root->right = newNode(20);
root->left->left = newNode(10);
root->left->right = newNode(7);
root->left->left->left = newNode(1);

root->left->left->right = newNode(3);
it does not work for this input

Reply • Share ›

abubakar abubakar • 2 months ago

Input is wrong

Reply • Share ›

M C S KRISHNA • 4 months ago

O(n) solution:

```
void CorrectBSTMine (struct node * root)
{
static struct node *temp1 = NULL;
static struct node *temp2 = NULL;
int temp;

if (root != NULL && (temp1 == NULL || temp2 == NULL) ) {
if (root->left != NULL && root->left->data > root->data) {

if (temp1 == NULL) {
temp1 = root->left;
}
else {
temp2 = root->left;
}
}
```

see more

Reply • Share ›

rahul tibrewal • 6 months ago

what if the first element of the inorder traversal, or the leftmost element has to
is correct, am i wrong?

^ | ˅ • Reply • Share ›

mrn • 8 months ago

```
void correctSwappedTree(Node *root,Node *&first,Node *&second,Node *&pr

if(root==NULL)

return;

correctSwappedTree(root->l,first,second,prev);

if(prev!=NULL && root->v < prev->v)

{

if(!first)

{first=prev;second=root;}

else

second=root;

}

prev=root;

correctSwappedTree(root->r,first,second,prev);

}
```

^ | ˅ • Reply • Share ›

ubiquitous • 9 months ago

create a binary tree with the data 10, 5 ,7,6,8,9. Now you swap 7,9. Check the answer. Thanks.

Reply • Share ›

Sumit Monga • 9 months ago

Sorry no need to use (count<2) in both else if statements.

Reply • Share ›

Sumit Monga • 9 months ago

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

struct node
{

int data;.

struct node *left, *right;.
};

void correctBSTUtil( struct node * root, int min, int max).
{
if (root==NULL)
return;
static int count;.
static struct node * temp;.
if(root->left)
```

see more

Reply • Share ›

illuminati • 9 months ago

simple method but stucked in implementation... :P simple method but stucked in implementation... :P

^ | v • Reply • Share ›

Ishita • 9 months ago

We don't need three pointers first, middle and last. We can just do with two.

```
if ( !*first )
            {
                *first = *prev;
                *last = root;
            }

            else
                *last = root;
```

Now just swap data at first and last.

1 ^ | v • Reply • Share ›

Someone • 10 months ago

Can someone give me a working java solution to this? How to use double poin

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

^ | v • Reply • Share ›

Bond • 11 months ago

I think we can save ourselves from recursion using the iterative in-order traver
would be lot easier to handle. Just run a loop and maintain three pointers as su
I agree it would not allow you to swap the pointers, but it would be easier to sw

^ | v • Reply • Share ›

abhishek08aug • 11 months ago

Intelligent :D

Reply • Share ›

ABHINAV abhishek08aug • 11 months ago

Please tell me if you find my method a little faulty

It may help me

My Comment is 2 comments below

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

Reply • Share ›

apsc • a year ago

in the simple method after sorting the auxiliary array, how do you insert elemei

Reply • Share ›

Kunal Arora apsc • 22 days ago

http://www.geeksforgeeks.org/b...

Refer to above question you will get the answer to your question...

Reply • Share ›

ABHINAV • a year ago

I think for this not for travelling whole tree just mark the positions when we find defaulty positions

please comment on this
/*
node *ptr1=null,*ptr 2=null;
function tree(node)

{
leftnode=node->left;
rightnode=node->right;

if(node->left==null && node->right==null)
return;

if(leftnode->info>node->info || rightnode->infoinfo)
{
if(*ptr1!=null)
*ptr1=node;
else if(*ptr2!=null)

see more

^ | v • Reply • Share ›

ABHINAV ➦ ABHINAV • a year ago

*if(leftnode->info>node->info || rightnode-><infoinfo)

means if we find default node in the BST

^ | v • Reply • Share ›

Amit • a year ago

```
 /* Paste your code here (You may delete these lines if not writing c
#include <stdio.h>
#include <stdlib.h>
#include<limits.h>
struct node
{
    int data;
    struct node *left, *right;
};
struct node * first = NULL;
struct node * second = NULL;
void swap( int* a, int* b )
{
    int t = *a;
    *a = *b;
    *b = t;
}
struct node* newNode(int data)
```

see more

· Reply · Share ›

Amit ➦ Amit · a year ago

void correctBST(struct node * root, int min, int max){
if(root== NULL){
return;
}
correctBST(root->left,min,root->data-1);
if(root->data data > max){
if(first == NULL){
first = root;
}
else{
second = root;
}
}
correctBST(root->right,root->data+1,max);
}

· Reply · Share ›

srinichal · a year ago

```
void Tree::BSTNormal(TNode* node)
{
        if(node==NULL) return;
        int max = node->val;
        int min = node->val;
        GetMax(node->left,&max);
        GetMin(node->right,&min);
        if(min < max)
                swap(node,max,min);
        else if(node->val <= max)
                swap(node,node->val,max);
        else if(node->val >= min)
                swap(node,node->val,min);
        BSTNormal(node->left);
        BSTNormal(node->right);
}
```

^ | ⌄ • Reply • Share ›

firefist • 2 years ago

i thought i should try swapping the pointers instead of data.
please let me know if it has any erros.

```
static void Main(){
            BinaryTree obj = new BinaryTree();
            Node beta = null;
            Node alpha = null;
            obj.FindSwappedNodes(obj.Root, ref alpha, ref beta);
            Node alphaParent = null;
            Node betaParent = null;
            bool isAlphaFound = false;
            bool isBetaFound = false;
            obj.Root = obj.SwapBSTNodes(obj.Root, alpha, beta, ref is

}
public void FindSwappedNodes(Node root,ref Node alpha,ref Node beta)
        {
```

see more

^ | ⌄ • Reply • Share ›

latha • 2 years ago

/*
The solution for this one is after obtaining the inorder traversal of bst then use
increment first upto u get a greater element than that one, and decrement last
swap both the elements.
Eg:
3 25 6 8 10 5 28
/ \last
first and store the first add in some variable let us say first1.
compare 3 with 25 3 less so inc after inc first
3 25 6 8 10 5 28
/ \
first last
now compare 25 with 6 greater so stop here now compare 28 with 5 using a t
3 25 6 8 10 5 28
/ \
first last
now compare 5 with 10 5 is less and now stop and swap first and last.
after swapping
3 5 6 8 10 25 28
thank u:)
*/

^ | ⌄ • Reply • Share ›

latha ➦ latha • 2 years ago

/*sry first nd last are not pointed correctly ..
in first case first points to 3 and last points to 28.
in second case first points to 25 and last points to 28.
in third case first points to 25 and last points to 5.
now then swap..
thank u:)
*/

• Reply • Share ›

vasavi • 2 years ago

nice......but......it is not a good program.....u have to try again

• Reply • Share ›

Anant Upadhyay • 2 years ago

good solution

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

• Reply • Share ›

vasavi ➦ Anant Upadhyay • 2 years ago

fine

• Reply • Share ›

Karthick • 2 years ago

Hi,
I think the following approach can be used if the nodes as such have to be swa

1) First, find the two misplaced nodes along with their parents(n1,p1,n2,p2) Th
modifying the code for checking if the given tree is a BST.
2) Swap the children of the nodes n1 and n2.
3) Make p1 point to n2 and p2 point to n1.
There is one small catch in the third step - if n1 is the parent of n2, it has to ha

Thanks,
Karthick

• Reply • Share ›

Karthick Karthick • 2 years ago

The following code can also be used to find the two misplaced nodes.
This is a slight modification of checking if the given tree is a BST.
[sourcecode language="C++"]

```
void findMisplacedNodes(node *root,int low,int high,node *&n1,node *&
if(!root)
return;
findMisplacedNodes(root->left,low,root->data,n1,n2);
if(root->datadata>high){
if(!n1)
n1=root;
else
n2=root;
}
findMisplacedNodes(root->right,root->data,high,n1,n2);
return;
}
```

[sourcecode]

If anything is wrong, please comment.

Thanks,

^ | ⌄ • Reply • Share ›

Karthick ➦ Karthick • 2 years ago

Sorry, I tried the sourcecode tag with c++ and it is not working. Reposting the code.

```
void findMisplacedNodes(node *root,int low,int high,node
        if(!root)
                return;
        findMisplacedNodes(root->left,low,root->data,n1,
        if(root->data<low || root->data>high){
                if(!n1)
                        n1=root;
                else
                        n2=root;
        }
        findMisplacedNodes(root->right,root->data,high,r
        return;
}
```

Thanks,

^ | ⌄ • Reply • Share ›

vasavi ➦ Karthick • 2 years ago

It is Somewhat complicated so U have write a program. u...k........good luck

^ | ⌄ • Reply • Share ›

Aashish ➦ Karthick • 2 years ago

Your approach doesn't work for:
10,5,15,3.
Swap nodes 5 and 3.

• Reply • Share ›

Karthick ➦ Aashish • 2 years ago

Yes. Thanks for pointing that out. :)

• Reply • Share ›

Subscribe   Add Disqus to your site