GeeksforGeeks
A computer science portal for geeks

GeeksQuiz

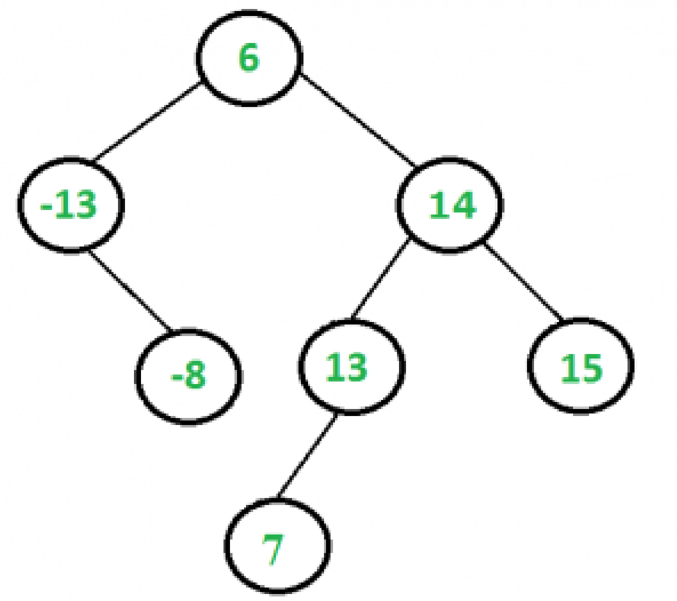
Login

Home Algorithms DS GATE Interview Corner Q&A C C++ Java Books Contribute Ask a Q About

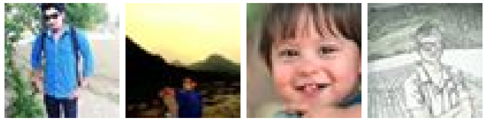
Array Bit Magic C/C++ Articles GFacts Linked List MCQ Misc Output String Tree Graph

Find if there is a triplet in a Balanced BST that adds to zero

Given a Balanced Binary Search Tree (BST), write a function isTripletPresent() that returns true if there is a triplet in given BST with sum equals to 0, otherwise returns false. Expected time complexity is O(n^2) and only O(Logn) extra space can be used. You can modify given Binary Search Tree. Note that height of a Balanced BST is always O(Logn)
For example, isTripletPresent() should return true for following BST because there is a triplet with sum 0, the triplet is {-13, 6, 7}.



The Brute Force Solution is to consider each triplet in BST and check whether the sum adds upto zero. The time complexity of this solution will be O(n^3).



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

A **Better Solution** is to create an auxiliary array and store Inorder traversal of BST in the array. The array will be sorted as Inorder traversal of BST always produces sorted data. Once we have the Inorder traversal, we can use method 2 of this post to find the triplet with sum equals to 0. This solution works in O(n^2) time, but requires O(n) auxiliary space.

Following is the solution that works in O(n^2) time and uses O(Logn) extra space:
1) Convert given BST to Doubly Linked List (DLL)
2) Now iterate through every node of DLL and if the key of node is negative, then find a pair in DLL with sum equal to key of current node multiplied by -1. To find the pair, we can use the approach used in hasArrayTwoCandidates() in method 1 of this post.

```
// A C++ program to check if there is a triplet with sum equal to 0 in
// a given BST
#include<stdio.h>

// A BST node has key, and left and right pointers
struct node
{
    int key;
    struct node *left;
    struct node *right;
};

// A function to convert given BST to Doubly Linked List. left pointer
// as previous pointer and right pointer is used as next pointer. The
// sets *head to point to first and *tail to point to last node of con
void convertBSTtoDLL(node* root, node** head, node** tail)
{
    // Base case
    if (root == NULL)
        return;

    // First convert the left subtree
    if (root->left)
        convertBSTtoDLL(root->left, head, tail);

    // Then change left of current root as last node of left subtree
    root->left = *tail;

    // If tail is not NULL, then set right of tail as root, else curre
    // node is head
    if (*tail)
        (*tail)->right = root;
    else
```

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

```
        *head = root;

    // Update tail
    *tail = root;

    // Finally, convert right subtree
    if (root->right)
        convertBSTtoDLL(root->right, head, tail);
}

// This function returns true if there is pair in DLL with sum equal
// to given sum. The algorithm is similar to hasArrayTwoCandidates()
// in method 1 of http://tinyurl.com/dy6palr
bool isPresentInDLL(node* head, node* tail, int sum)
{
    while (head != tail)
    {
        int curr = head->key + tail->key;
        if (curr == sum)
            return true;
        else if (curr > sum)
            tail = tail->left;
        else
            head = head->right;
    }
    return false;
}

// The main function that returns true if there is a 0 sum triplet in
// BST otherwise returns false
bool isTripletPresent(node *root)
{
    // Check if the given  BST is empty
    if (root == NULL)
       return false;

    // Convert given BST to doubly linked list.  head and tail store t
    // pointers to first and last nodes in DLLL
    node* head = NULL;
    node* tail = NULL;
    convertBSTtoDLL(root, &head, &tail);

    // Now iterate through every node and find if there is a pair with
    // equal to -1 * heaf->key where head is current node
    while ((head->right != tail) && (head->key < 0))
    {
        // If there is a pair with sum equal to  -1*head->key, then re
        // true else move forward
        if (isPresentInDLL(head->right, tail, -1*head->key))
            return true;
        else
            head = head->right;
    }

    // If we reach here, then there was no 0 sum triplet
    return false;
}

// A utility function to create a new BST node with key as given num
node* newNode(int num)
{
    node* temp = new node;
    temp->key = num;
    temp->left = temp->right = NULL;
    return temp;
}

// A utility function to insert a given key to BST
node* insert(node* root, int key)
{
    if (root == NULL)
       return newNode(key);
    if (root->key > key)
       root->left = insert(root->left, key);
    else
       root->right = insert(root->right, key);
    return root;
}

// Driver program to test above functions
int main()
{
    node* root = NULL;
    root = insert(root, 6);
    root = insert(root, -13);
    root = insert(root, 14);
    root = insert(root, -8);
    root = insert(root, 15);
    root = insert(root, 13);
    root = insert(root, 7);
    if (isTripletPresent(root))
        printf("Present");
    else
        printf("Not Present");
```

Subscribe

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

```
    return 0;
}
```

Output:

```
Present
```

Note that the above solution modifies given BST.

Time Complexity: Time taken to convert BST to DLL is O(n) and time taken to find triplet in DLL is O(n^2).

Auxiliary Space: The auxiliary space is needed only for function call stack in recursive function convertBSTtoDLL(). Since given tree is balanced (height is O(Logn)), the number of functions in call stack will never be more than O(Logn).

We can also find triplet in same time and extra space without modifying the tree. See next post. The code discussed there can be used to find triplet also.

This article is compiled by Ashish Anand and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Explore New
Frontiers in Data.
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

30 Tweet 3 1

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

20 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion…

Anand • a month ago

We can also do this by implementing a hash table with all the contents of the I checkifSum(int sum) which checks if there are two numbers that add up to su table and checks for every entry n if there is an entry (sum-x)...so for every en O(n) for the function itself.

Now, to solve this problem, we loop thru the hash table and for every entry m, m). Since this function runs in O(n), we can implement this solution in n*n ie C

Any thoughts/comments

^ | v • Reply • Share ›

Mojo • a month ago

Mojo • a month ago

When there are three zeroes in a BST, this will not return true.

Reply • Share ›

Prama • 9 months ago

@GeeksForGeeks

For the better solution; why will the complexity be O(n^2)? The inorder tree tra

the array will be another O(n) and this will lead to O(n)+O(n) ~O(n) .. it isn't th

array - which is what will make it O(n^2). Please explain.

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

Reply • Share ›

12rad • 9 months ago

@GeeksForGeeks

The better solution :

A Better Solution is to create an auxiliary array and store Inorder traversal of B

sorted as Inorder traversal of BST always produces sorted data. Once we ha

method 2 of this post to find the triplet with sum equals to 0. This solution wor

auxiliary space.

Why will this be O(n^2)? Tree traversal will be O(n) and then traversing the arr

not that for each tree node, we're traversing the array.

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

Reply • Share ›

Sreenivas Doosa ➜ 12rad • 6 months ago

@12rad:

It will be O(n^2) only.. Finding a triplet sum equal to given sum takes O

find just a pair sum equal to given sum takes O(n) .. for finding triplet, added to another pair in the array.. hence it will be O(n^2).

Reply • Share ›

Ganesh • 9 months ago

```
Recursive solution using inorder and reverse inorder.

Time complexity : I guess its O(nlogn)
Space complexity: O(logn) due to recursion call stack

package com.ganesh;

import com.ganesh.Node;

public class TreePair {

        static int INT_MIN = -32767;

        // Do an inorder traversal to print a tree
    public static void printTree(Node root) {
        if (root==null) return;
        printTree(root.small);
```

see more

Reply • Share ›

Ganesh • 9 months ago

Following is a Recursive solution that performs inorder and reverse inorder; w summing to the given target. I think the time complexity in worst case is O(nlo recursion call stack) is O(logn).

Checks are first made to see if duplicate values may form a triplet before sear This makes the code little more efficient.

This makes the code little more efficient.

```
package com.ganesh;

import com.ganesh.Node;

public class TreePair {

        static int INT_MIN = -32767;

        // Do an inorder traversal to print a tree
    public static void printTree(Node root) {
        if (root==null) return;
```

see more

Reply • Share ›

abhishek08aug • 11 months ago

Intelligent :D

Reply • Share ›

Vimal • a year ago

Can you please explain the brute force method ?
I am finding it hard to get it.

Reply • Share ›

GeeksFollower • a year ago

@GeeksForGeeks

Your program runs great EXCEPT ONE CASE:
when tree is having only single node and it is negative. slight modification will s

Reply • Share ›

Aaman • a year ago

Better approach will be do inorder and reverse in order simultaneously,now ad
negative of this sum in tree in logn times..so N+NlogN

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

⌃ | ⌄ • Reply • Share ›

Geeker • a year ago

@GeeksforGeeks Can you please explain why have you converted the Balanc
searching in Balanced BST would hav kept the time complexity at O(nlogn) ar

⌃ | ⌄ • Reply • Share ›

Die_hard dhoni_fan → Geeker • a year ago

@Geeker If there is no parent pointer in BST we cannot go to the parer
Since it will be the next node.

⌃ | ⌄ • Reply • Share ›

viki • a year ago

How can you convert DLL back into the original BST ?

⌃ | ⌄ • Reply • Share ›

GeeksforGeeks → viki • a year ago

Following post may be helpful

http://www.geeksforgeeks.org/i...

⌃ | ⌄ • Reply • Share ›

Aayush • a year ago

Hi,

Can be done in n^2logn by considering two number(say a and b) frm BST and logn search for BST.????

Reply • Share ›

Aayush ➜ Aayush • a year ago

But there may be problem how we can select two node from BST??

Reply • Share ›

sandeep • a year ago

That is not a BST

Reply • Share ›

GeeksforGeeks ➜ sandeep • a year ago

Thanks for pointing this out. We have updated the diagram.

Reply • Share ›

Cleon Barrett • a year ago

Is it C/ C++ alone you use for your solutions? If not can you present an alterna
Java/scheme/Python, I know the time complexity may differ a little based on th

Reply • Share ›

Subscribe   Add Disqus to your site