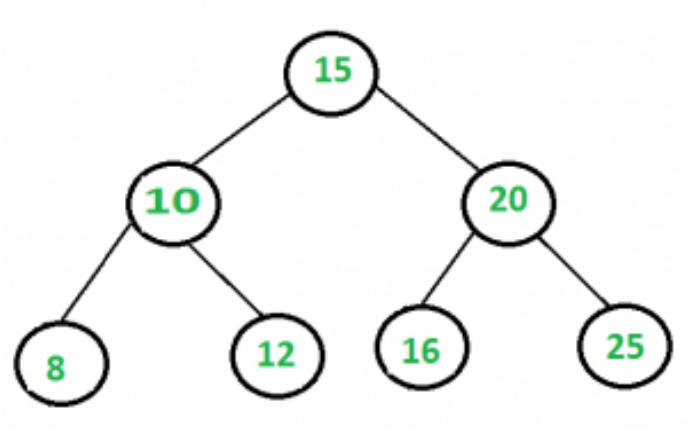
# Find a pair with given sum in a Balanced BST

Given a Balanced Binary Search Tree and a target sum, write a function that returns true if there is a pair with sum equals to target sum, otherwise return false. Expected time complexity is O(n) and only O(Logn) extra space can be used. Any modification to Binary Search Tree is not allowed. Note that height of a Balanced BST is always O(Logn).




This problem is mainly extension of the previous post. Here we are not allowed to modify the BST.

The **Brute Force Solution** is to consider each pair in BST and check whether the sum equals to X. The time complexity of this solution will be O(n^2).

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 pair in O(n) time (See this for details). This solution works in O(n) time, but requires O(n) auxiliary space.



A **space optimized solution** is discussed in previous post. The idea was to first in-place convert BST to Doubly Linked List (DLL), then find pair in sorted DLL in O(n) time. This solution takes O(n) time and O(Logn) extra space, but it modifies the given BST.

The **solution discussed below takes O(n) time, O(Logn) space and doesn't modify BST**. The idea is same as finding the pair in sorted array (See method 1 of this for details). We traverse BST in Normal Inorder and Reverse Inorder simultaneously. In reverse inorder, we start from the rightmost node which is the maximum value node. In normal inorder, we start from the left most node which is minimum value node. We add sum of current nodes in both traversals and compare this sum with given target sum. If the sum is same as target sum, we return true. If the sum is more than target sum, we move to next node in reverse inorder traversal, otherwise we move to next node in normal inorder traversal. If any of the traversals is finished without finding a pair, we return false. Following is C++ implementation of this approach.

```
/* In a balanced binary search tree isPairPresent two element which su
   a given value time O(n) space O(logn) */
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 100

// A BST node
struct node
{
    int val;
    struct node *left, *right;
};

// Stack type
struct Stack
{
    int size;
    int top;
    struct node* *array;
};

// A utility function to create a stack of given size
struct Stack* createStack(int size)
{
    struct Stack* stack =
        (struct Stack*) malloc(sizeof(struct Stack));
    stack->size = size;
    stack->top = -1;
    stack->array =
```



## Popular Posts

```
        (struct node**) malloc(stack->size * sizeof(struct node*));
    return stack;
}

// BASIC OPERATIONS OF STACK
int isFull(struct Stack* stack)
{   return stack->top - 1 == stack->size;  }

int isEmpty(struct Stack* stack)
{   return stack->top == -1;   }

void push(struct Stack* stack, struct node* node)
{
    if (isFull(stack))
        return;
    stack->array[++stack->top] = node;
}

struct node* pop(struct Stack* stack)
{
    if (isEmpty(stack))
        return NULL;
    return stack->array[stack->top--];
}

// Returns true if a pair with target sum exists in BST, otherwise fal
bool isPairPresent(struct node *root, int target)
{
    // Create two stacks. s1 is used for normal inorder traversal
    // and s2 is used for reverse inorder traversal
    struct Stack* s1 = createStack(MAX_SIZE);
    struct Stack* s2 = createStack(MAX_SIZE);

    // Note the sizes of stacks is MAX_SIZE, we can find the tree size
    // fix stack size as O(Logn) for balanced trees like AVL and Red B
    // tree. We have used MAX_SIZE to keep the code simple

    // done1, val1 and curr1 are used for normal inorder traversal usi
    // done2, val2 and curr2 are used for reverse inorder traversal us
    bool done1 = false, done2 = false;
    int val1 = 0, val2 = 0;
    struct node *curr1 = root, *curr2 = root;

    // The loop will break when we either find a pair or one of the tw
    // traversals is complete
    while (1)
    {
```

```
        // Find next node in normal Inorder traversal. See following p
        // http://www.geeksforgeeks.org/inorder-tree-traversal-without
        while (done1 == false)
        {
            if (curr1 != NULL)
            {
                push(s1, curr1);
                curr1 = curr1->left;
            }
            else
            {
                if (isEmpty(s1))
                    done1 = 1;
                else
                {
                    curr1 = pop(s1);
                    val1 = curr1->val;
                    curr1 = curr1->right;
                    done1 = 1;
                }
            }
        }

        // Find next node in REVERSE Inorder traversal. The only
        // difference between above and below loop is, in below loop
        // right subtree is traversed before left subtree
        while (done2 == false)
        {
            if (curr2 != NULL)
            {
                push(s2, curr2);
                curr2 = curr2->right;
            }
            else
            {
                if (isEmpty(s2))
                    done2 = 1;
                else
                {
                    curr2 = pop(s2);
                    val2 = curr2->val;
                    curr2 = curr2->left;
                    done2 = 1;
                }
            }
        }
```

Subscribe

## Recent Comments

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

Backtracking | Set 7 (Sudoku) · 32 minutes ago

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

Flipkart Interview | Set 6 · 52 minutes ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 52 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

```
        // If we find a pair, then print the pair and return. The firs
        // condition makes sure that two same values are not added
        if ((val1 != val2) && (val1 + val2) == target)
        {
            printf("\n Pair Found: %d + %d = %d\n", val1, val2, target
            return true;
        }

        // If sum of current values is smaller, then move to next node
        // normal inorder traversal
        else if ((val1 + val2) < target)
            done1 = false;

        // If sum of current values is greater, then move to next node
        // reverse inorder traversal
        else if ((val1 + val2) > target)
            done2 = false;

        // If any of the inorder traversals is over, then there is no
        // so return false
        if (val1 >= val2)
            return false;
    }
}

// A utility function to create BST node
struct node * NewNode(int val)
{
    struct node *tmp = (struct node *)malloc(sizeof(struct node));
    tmp->val = val;
    tmp->right = tmp->left =NULL;
    return tmp;
}

// Driver program to test above functions
int main()
{
    /*
                  15
                /    \
              10      20
             / \     /  \
            8  12   16  25    */
    struct node *root =  NewNode(15);
    root->left = NewNode(10);
    root->right = NewNode(20);
    root->left->left = NewNode(8);
```

```
    root->left->right = NewNode(12);
    root->right->left = NewNode(16);
    root->right->right = NewNode(25);

    int target = 33;
    if (isPairPresent(root, target) == false)
        printf("\n No such values are found\n");

    getchar();
    return 0;
}
```

Output:

```
Pair Found: 8 + 25 = 33
```

This article is compiled by Kumar 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.



## 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



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

## 47 Comments GeeksforGeeks

Sort by Newest



**AlienOnEarth** • 3 days ago

Another o(n) Approach. Same as finding pair sum in a sorted array. C source

```
#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

bool isSumSet = false;

// Structure of a BST Node

struct node

{
```

int data;

struct node *left;

struct node *right;

see more

• Reply • Share ›

**Lohith Ravi** • 4 days ago

I have done a Iterative method or processing InOrder and reverseInOrder

http://ideone.com/p33W9c

• Reply • Share ›

**sukisukimo** • a month ago

/*

* Modified version of the above algorithm, where is the pair resides on the left/r

*/

boolean findPairOfSumBestSol(BinaryNode head, int sum)

{

int leftVal = 0, rightVal = 0;

BinaryNode left = head;

BinaryNode right = head.right;

boolean isLeft, isRight, still;

isLeft = isRight = still = true;

Stack s1 = new Stack(10);

see more

• Reply • Share ›

**sukisukimo** • a month ago

This algorithm will not work if the pair are in the left subtree(or right subtree) fr

• Reply • Share ›

**Jayanth** • 3 months ago

Can someone explain how is the space O(logn)..??

Isnt it O(n)...??

• Reply • Share ›

**Sumit Monga** ➦ Jayanth • 3 months ago

height of tree is always 0(logn) as it is a balanced bst so maximum de
elements we go from one level to another. hope its clear now

1 • Reply • Share ›

**Newgeek** • 4 months ago

i think we can have a O(nlogn) solution for this without any space.

Since it is a BST, for each node we can search for (k - node->data) in the tree

• Reply • Share ›

**OP Coder** ➦ Newgeek • 3 months ago

In today's world, space is not a problem, but speed is. So always try to

1 • Reply • Share ›

**amit** • 7 months ago

This code returns only single pair value if we have multiple pair value then it wi
to update the code

```
// condition makes sure that two same values are not added
if ((val1 != val2) && (val1 + val2) == target)
{
printf("\n Pair Found: %d + %d = %d\n", val1, val2, target);
boo1= false;
boo2 = false;
}
```

let me know for any more issue.

  • Reply • Share ›

**hary** • 7 months ago

Can someone please point if either of the stack would ever go empty before th
could not think of a scenario and so please help understanding the same. In ca
request please update the code to avoid the confusion as it adds extra conditio

  • Reply • Share ›

**Ganesh** • 9 months ago

```
 package com.ganesh;

import com.ganesh.Node;

public class TreePair {

        // Do an inorder traversal to print a tree
    public static void printTree(Node root) {
        if (root==null) return;
        printTree(root.small);
        System.out.print(Integer.toString(root.data) + " ");
        printTree(root.large);
    }

        public static void findsumPair(int sum, Node root, Node curr)
```

```
    {
        if(root == null && curr == null) return;
```

see more

• Reply • Share ›

**Kuldeep Tiwari** • 9 months ago

Hi Geeksforgeeks,
Below is a working recursive solution for this, which uses parent pointers in nc

/*Main method.
It starts with min node (left most child of root) and max node (right most child
sum, return this pair, else if their sum < desired sum, replace left with inorder
function. If sum of node values > desired sum, replace right with inorder prede
of class &#039Pair&#039 which is described later.
*/
private static Pair getPair(Node left, Node right, Integer sum ).
{
if ( left == null || right == null ).
return null;
Integer leftNum = left.getData(), rightNum = right.getData();.

if ( leftNum + rightNum == sum ).
return new Pair(leftNum, rightNum);.
else if (leftNum + rightNum < sum ).

see more

• Reply • Share ›

**Prabhu** • 9 months ago

Here is a concise solution...

```
void pushallleft(node * n, stack & stk)
{
while(n)
{
stk.push(n);
n = n->left;
}
}

void pushallright(node * n, stack & stk)
{
while(n)
{
stk.push(n);
n = n->right;
}
}
```

see more

• Reply • Share ›

**Abhay** • 10 months ago

Hi,
There is a bug in the code.
Try on following input

```
/*
                15
              /    \
            10      17
           / \     /  \
          8  12   16  25    */
```

```
struct node *root = NewNode(15);
root->left = NewNode(10);
root->right = NewNode(17);
root->left->left = NewNode(8);
root->left->right = NewNode(12);
root->right->left = NewNode(16);
root->right->right = NewNode(25);
```

see more

• Reply • Share ›

**GeeksforGeeks** ➚ Abhay • 10 months ago

@Abhay: Thanks for pointing this out. We have updated the condition.

• Reply • Share ›

**Abhay** ➚ GeeksforGeeks • 10 months ago

:)
you can even replace
if ((curr1==NULL && isEmpty(s1)) || ((curr2==NULL) && isEmp

with
if(val1>=val2)
return false;
as before any traversal ends val1 will have to pass through val2

• Reply • Share ›

**GeeksforGeeks** ➚ Abhay • 10 months ago

Ok thanks Abhay, we have updated the condition.

• Reply • Share ›

**Jayanth** ➚ GeeksforGeeks • 3 months ago

lsnt it O(n)...???



**Amit Singh** GeeksforGeeks • 8 months ago

This code returns only one pair , what if there exists mu
/sum.

Example : target=14, pairs (9,5) & (8,6)

6
2 8
1 4 7 9
3 5



**Akhil** • 10 months ago

A concise C++ code

```
[sourcecode language="C++"]
#include<stdio.h>
#include<stdlib.h>
#include<stack>
using namespace std;
struct tree
{
int info;
struct tree *l;
struct tree *r;
};
typedef struct tree *Tree;

Tree insert(int num)
{
```

```
Tree root = (Tree)malloc(sizeof(struct tree));
root->info = num;
```

see more

• Reply • Share ›

**Akhil** • 10 months ago

A Concise Code.

```
 #include<stdio.h>
#include<stdlib.h>
#include<stack>
using namespace std;
struct tree
{
    int info;
    struct tree *l;
    struct tree *r;
};
typedef struct tree *Tree;

Tree insert(int num)
{
    Tree root = (Tree)malloc(sizeof(struct tree));
    root->info = num;
```

see more

• Reply • Share ›

**Sandeep Jain** • 11 months ago

Thanks for pointing this out. We have updated the code to handle this conditio
val2) before printing the solution.

**abhishek08aug** • 11 months ago

Intelligent :D



**Akhilesh Saini** • a year ago

can you please tell me the approach...?



**Akhilesh Saini** • a year ago

can you please tell me the approach...?



**Vimal** • a year ago

What would be brute force solution for this ?
How do you find each pair in an BST ?



**Asif Eqbal** • a year ago

Hi I think in above program since you are traversing both inorder and reverse i
fail in scenario such that if target value is 50 then it will give soln as 25+25=50
Pair Present". In general if we given value of target as double of any node valu

Modified code:

```
#include
#include
#include
struct node{
int num;
node *left;
```

```
};
node *root;
int buildtree(int);
int inorder(node *);
int findsum(node *,int);
int main()
```

see more

Reply • Share ›

**Sarvanan Boopathy** • a year ago

Hi, I think we can solve this in Inorder Traversal way as below. Please correct

```
class FindIfSumExists{
	public boolean ifSumExists(Tree root, Tree node, int sum){
		if(root == null || node == null) return false;
		boolean b = false;
		if(node == root){
			b = findOther(root.left,node.data - sum);
			b = findOther(root.right,node.data -sum) | b;

			if(b) return true;
		}
		if(findOther(root,node.data - sum) return true;
		return ifSumExists(root,node.left,sum) || ifSumExists(
	}

	public boolean findOther(Tree root, int sum){
	if(root == null) return false;
```

see more

Reply • Share ›

**Sarvanan Boopathy** ➚ Sarvanan Boopathy • a year ago

Sorry, the above code had a bug... this is the correct version

```
package trees;

class FindIfSumExists{
public boolean ifSumExists(Tree root, Tree node, int sum){
if(root == null || node == null) return false;
boolean b = false;
if(node == root){
b = findOther(root.left,sum - node.i,node);
b = findOther(root.right,sum - node.i,node) | b;

if(b) return true;
}
if(findOther(root,sum - node.i,node)) return true;
return ifSumExists(root,node.left,sum) || ifSumExists(root,node.right,su
}

public boolean findOther(Tree root, int sum,Tree node){
```

see more

^ | v • Reply • Share ›

**Sarvanan Boopathy** • a year ago

Hi, I think the simpler solution would be of O(n). When we traverse the tree in i
complement of this value exists in the tree. Below is the Java code.

Please correct me if I am wrong.

```
class FindIfSumExists{
public boolean ifSumExists(Tree root, Tree node, int sum){
if(root == null || node == null) return false;
```

```
boolean b = false;
if(node == root){
b = findOther(root.left,node.data - sum);
b = findOther(root.right,node.data -sum) | b;

if(b) return true;
}
if(findOther(root,node.data - sum) return true;
return ifSumExists(root,node.left,sum) || ifSumExists(root,node.right,sum);
}
```

see more

Reply • Share ›

**Rushabh Shah** • a year ago

I think there is a bug in the code. What if you are searching for the sum 40. Bo 20 and print as sum found whereas this is not the case. Right?

Reply • Share ›

**Gupt** • a year ago

A Hashset based solution

do in/pre/post/level order traversal

```
{
compliment = sum - currnode->data;

if (hashset.contains(compliment))
{
return true or alternatively print order;
}
```

}

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



**sk** • a year ago

But how do you solve, if given pair lies in either left side or only in right side of t
I think , given geeksforgeeks solution is incorrect. Correct me if i am wrong.



**GeeksforGeeks** ➦ sk • a year ago

@sk: Could you please let us know a test case for which it didn't work
seems to be working for all. See following main() function for example.

```
 int main()
{
    /*
                    15
                  /    \
                10      20
               / \     /  \
              8  12   16  25    */
    struct node *root =  NewNode(15);
    root->left = NewNode(10);
    root->right = NewNode(20);
    root->left->left = NewNode(8);
    root->left->right = NewNode(12);
    root->right->left = NewNode(16);
    root->right->right = NewNode(25);
```

Reply • Share ›

**hunter** GeeksforGeeks • 11 months ago

here i didn't get one thing....first we are going upto right==NULL cur2=cur2->left then after completion.cur2 is NULL,THEN we a we are in same node then how program will execute

Reply • Share ›

**GeeksforGeeks** hunter • 11 months ago

Please see following post for better understanding.

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

Reply • Share ›

**hunter** GeeksforGeeks • 11 months ago

thanks,i didn't see the code properly.....

Reply • Share ›

**Soumya Sengupta** GeeksforGeeks • a year ago

fucntion returns true for 16(8 and 8)...
we should have a condition if(val1!=val2)....after which the com

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

Reply • Share ›

**GeeksforGeeks** Soumya Sengupta • 11 months ago

Thanks for pointing this out. We have updated code to h

Reply • Share ›

**Amit** • a year ago

```
#include<stdio.h>
#include<conio.h>
int min = 0;
int max =0;
int count =0;
struct node
{
    int val;
    struct node *left, *right;
};
struct node * NewNode(int val)
{
    struct node *tmp = (struct node *)malloc(sizeof(struct node));
    tmp->val = val;
    tmp->right = tmp->left =NULL;
    return tmp;
}
```

see more

• Reply • Share ›

**Amit** • a year ago

```
 /* Paste your code here (You may delete these lines if not writing c
#include<stdio.h>
#include<conio.h>
int min = 0;
int max =0;
int count =0;
struct node
{
    int val;
```

```
};
struct node * NewNode(int val)
{
    struct node *tmp = (struct node *)malloc(sizeof(struct node));
    tmp->val = val;
    tmp->right = tmp->left =NULL;
    return tmp;
}
```

see more

• Reply • Share ›

**Amit** Amit • a year ago

```
 #include<stdio.h>
#include<conio.h>
int min = 0;
int max =0;
int flag =1;
struct node
{
    int val;
    struct node *left, *right;
};
struct node * NewNode(int val)
{
    struct node *tmp = (struct node *)malloc(sizeof(struct node
    tmp->val = val;
    tmp->right = tmp->left =NULL;
    return tmp;
}
void isPairPresent(struct node * root1,int target,struct node *
```

see more

• Reply • Share ›

**12rad** Amit • 9 months ago

This looks like a better solution. Thanks.

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

• Reply • Share ›

**GeeksforGeeks** • a year ago

Thanks for sharing your thoughts. Could you please provide more details. It wo
code or detailed algorithm

• Reply • Share ›

**Sandeep Bc** • a year ago

this is a very unoptimozed way of doing it, This can be done with a simple inor
based on the min and max values and adjusting it like BST check,

• Reply • Share ›

**viki** • a year ago

Can this problem be solved using recursive inorder traversal ?

• Reply • Share ›

**Ammy** • a year ago

Can't we somehow modify morris traversal to make it o(n)time and o(1) space

• Reply • Share ›

Subscribe Add Disqus to your site