Problem 1214: Two Sum BSTs

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Given the roots of two binary search trees,

root1

and

root2

, return

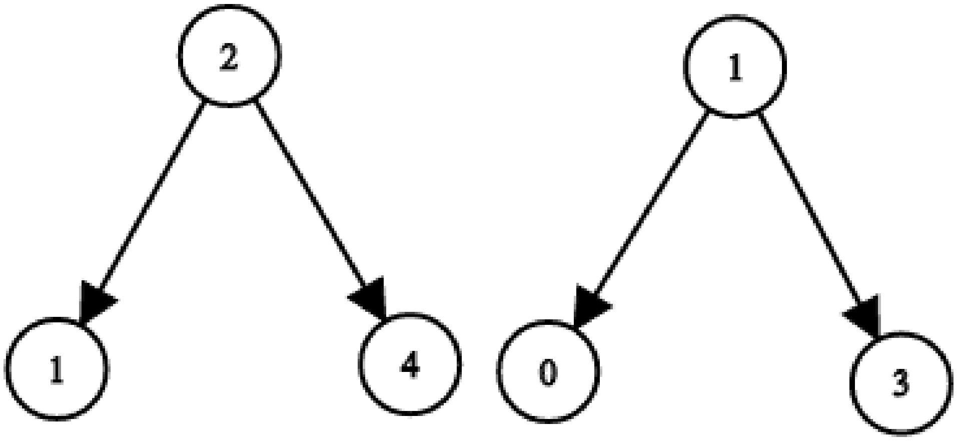
true

if and only if there is a node in the first tree and a node in the second tree whose values sum up to a given integer

target

.

Example 1:



Input:

root1 = [2,1,4], root2 = [1,0,3], target = 5

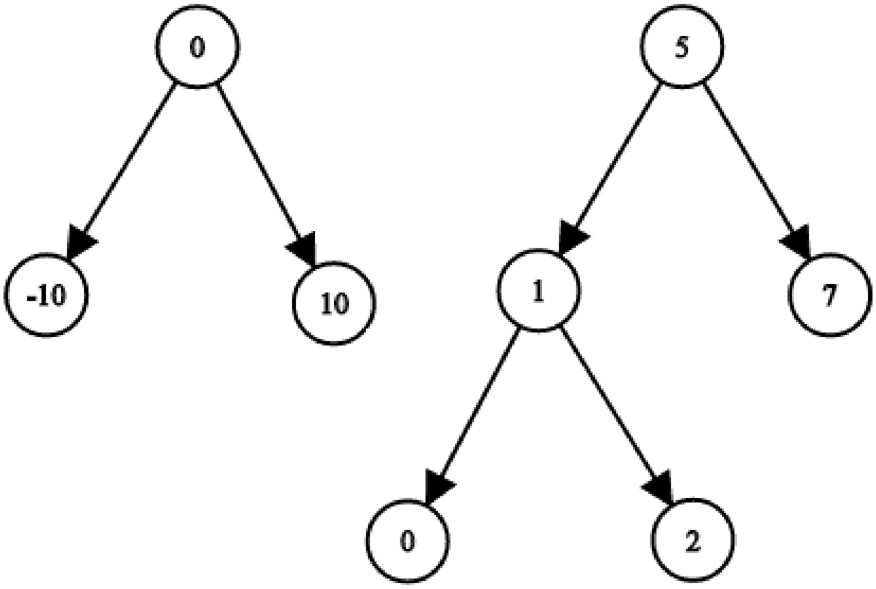
Output:

true

Explanation:

2 and 3 sum up to 5.

Example 2:



Input:

root1 = [0,-10,10], root2 = [5,1,7,0,2], target = 18

Output:

false

Constraints:

The number of nodes in each tree is in the range

[1, 5000]

.

-10

9

<= Node.val, target <= 10

9

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
```

```
public:
bool twoSumBSTs(TreeNode* root1, TreeNode* root2, int target) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public boolean twoSumBSTs(TreeNode root1, TreeNode root2, int target) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def twoSumBSTs(self, root1: Optional[TreeNode], root2: Optional[TreeNode],
target: int) -> bool:
```

Python:

```
# Definition for a binary tree node.
# class TreeNode(object):
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution(object):
def twoSumBSTs(self, root1, root2, target):
"""
:type root1: Optional[TreeNode]
:type root2: Optional[TreeNode]
:type target: int
:rtype: bool
"""
```

JavaScript:

```
/**
* Definition for a binary tree node.
* function TreeNode(val, left, right) {
* this.val = (val===undefined ? 0 : val)
* this.left = (left===undefined ? null : left)
* this.right = (right===undefined ? null : right)
* }
*/
/**
* @param {TreeNode} root1
* @param {TreeNode} root2
* @param {number} target
* @return {boolean}
*/
var twoSumBSTs = function(root1, root2, target) {

};
```

TypeScript:

```
/**
* Definition for a binary tree node.
* class TreeNode {
* val: number
* left: TreeNode | null
* right: TreeNode | null
* constructor(val?: number, left?: TreeNode | null, right?: TreeNode | null)
{
* this.val = (val===undefined ? 0 : val)
* this.left = (left===undefined ? null : left)
* this.right = (right===undefined ? null : right)
* }
* }
*/

function twoSumBSTs(root1: TreeNode | null, root2: TreeNode | null, target:
number): boolean {

};
```

C#:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* public int val;
* public TreeNode left;
* public TreeNode right;
* public TreeNode(int val=0, TreeNode left=null, TreeNode right=null) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
public class Solution {
public bool TwoSumBSTs(TreeNode root1, TreeNode root2, int target) {

}
}
```

C:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* struct TreeNode *left;
* struct TreeNode *right;
* };
*/
bool twoSumBSTs(struct TreeNode* root1, struct TreeNode* root2, int target) {

}
```

Go:

```
/**
* Definition for a binary tree node.
* type TreeNode struct {
* Val int
* Left *TreeNode
* Right *TreeNode
* }
*/
func twoSumBSTs(root1 *TreeNode, root2 *TreeNode, target int) bool {

}
```

Kotlin:

```
/**
* Example:
* var ti = TreeNode(5)
* var v = ti.`val`
* Definition for a binary tree node.
* class TreeNode(var `val`: Int) {
* var left: TreeNode? = null
* var right: TreeNode? = null
* }
*/
class Solution {
fun twoSumBSTs(root1: TreeNode?, root2: TreeNode?, target: Int): Boolean {

}
}
```

Swift:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* public var val: Int
* public var left: TreeNode?
* public var right: TreeNode?
* public init() { self.val = 0; self.left = nil; self.right = nil; }
* public init(_ val: Int) { self.val = val; self.left = nil; self.right =
nil; }
* public init(_ val: Int, _ left: TreeNode?, _ right: TreeNode?) {
* self.val = val
* self.left = left
* self.right = right
* }
* }
*/
class Solution {
func twoSumBSTs(_ root1: TreeNode?, _ root2: TreeNode?, _ target: Int) ->
Bool {

}
}
```

Rust:

```
// Definition for a binary tree node.
// #[derive(Debug, PartialEq, Eq)]
// pub struct TreeNode {
// pub val: i32,
// pub left: Option<Rc<RefCell<TreeNode>>>,
// pub right: Option<Rc<RefCell<TreeNode>>>,
// }
//
// impl TreeNode {
// #[inline]
// pub fn new(val: i32) -> Self {
// TreeNode {
// val,
// left: None,
// right: None
// }
// }
// }
```

```
use std::rc::Rc;
use std::cell::RefCell;
impl Solution {
pub fn two_sum_bs_ts(root1: Option<Rc<RefCell<TreeNode>>>, root2:
Option<Rc<RefCell<TreeNode>>>, target: i32) -> bool {

}
}
```

Ruby:

```
# Definition for a binary tree node.
# class TreeNode
# attr_accessor :val, :left, :right
# def initialize(val = 0, left = nil, right = nil)
# @val = val
# @left = left
# @right = right
# end
# end
# @param {TreeNode} root1
# @param {TreeNode} root2
# @param {Integer} target
# @return {Boolean}
def two_sum_bs_ts(root1, root2, target)

end
```

PHP:

```
/**
* Definition for a binary tree node.
* class TreeNode {
* public $val = null;
* public $left = null;
* public $right = null;
* function __construct($val = 0, $left = null, $right = null) {
* $this->val = $val;
* $this->left = $left;
* $this->right = $right;
* }
* }
```

```
*/
class Solution {

/**
* @param TreeNode $root1
* @param TreeNode $root2
* @param Integer $target
* @return Boolean
*/
function twoSumBSTs($root1, $root2, $target) {

}
}
```

Dart:

```
/**
* Definition for a binary tree node.
* class TreeNode {
* int val;
* TreeNode? left;
* TreeNode? right;
* TreeNode([this.val = 0, this.left, this.right]);
* }
*/
class Solution {
bool twoSumBSTs(TreeNode? root1, TreeNode? root2, int target) {

}
}
```

Scala:

```
/**
* Definition for a binary tree node.
* class TreeNode(_value: Int = 0, _left: TreeNode = null, _right: TreeNode = null) {
* var value: Int = _value
* var left: TreeNode = _left
* var right: TreeNode = _right
* }
*/
```

```
object Solution {
def twoSumBSTs(root1: TreeNode, root2: TreeNode, target: Int): Boolean = {

}
}
```

Elixir:

```
# Definition for a binary tree node.
#
# defmodule TreeNode do
# @type t :: %__MODULE__{
# val: integer,
# left: TreeNode.t() | nil,
# right: TreeNode.t() | nil
# }
# defstruct val: 0, left: nil, right: nil
# end

defmodule Solution do
@spec two_sum_bs_ts(root1 :: TreeNode.t | nil, root2 :: TreeNode.t | nil,
target :: integer) :: boolean
def two_sum_bs_ts(root1, root2, target) do

end
end
```

Erlang:

```
%% Definition for a binary tree node.
%%
%% -record(tree_node, {val = 0 :: integer(),
%% left = null :: 'null' | #tree_node{},
%% right = null :: 'null' | #tree_node{}}).

-spec two_sum_bs_ts(Root1 :: #tree_node{} | null, Root2 :: #tree_node{} |
null, Target :: integer()) -> boolean().
two_sum_bs_ts(Root1, Root2, Target) ->
.
```

Racket:

```
; Definition for a binary tree node.
#|

; val : integer?
; left : (or/c tree-node? #f)
; right : (or/c tree-node? #f)
(struct tree-node
(val left right) #:mutable #:transparent)

; constructor
(define (make-tree-node [val 0])
(tree-node val #f #f))

|#

(define/contract (two-sum-bs-ts root1 root2 target)
(-> (or/c tree-node? #f) (or/c tree-node? #f) exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Two Sum BSTs
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {
// TODO: Implement optimized solution
return 0;
}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {
// TODO: Implement optimized solution
return 0;
}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {
// TODO: Implement optimized solution
return 0;
}
* };
*/
class Solution {
public:
bool twoSumBSTs(TreeNode* root1, TreeNode* root2, int target) {

}
};
```

Java Solution:

```
/**
* Problem: Two Sum BSTs
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {
// TODO: Implement optimized solution
return 0;
}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public boolean twoSumBSTs(TreeNode root1, TreeNode root2, int target) {

}
}
```

Python3 Solution:

```
"""
Problem: Two Sum BSTs
Difficulty: Medium
Tags: array, tree, search, stack

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def twoSumBSTs(self, root1: Optional[TreeNode], root2: Optional[TreeNode],
target: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
# Definition for a binary tree node.
# class TreeNode(object):
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution(object):
def twoSumBSTs(self, root1, root2, target):
"""
:type root1: Optional[TreeNode]
:type root2: Optional[TreeNode]
:type target: int
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Two Sum BSTs
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* function TreeNode(val, left, right) {
* this.val = (val===undefined ? 0 : val)
* this.left = (left===undefined ? null : left)
* this.right = (right===undefined ? null : right)
* }
*/
/**
* @param {TreeNode} root1
* @param {TreeNode} root2
* @param {number} target
* @return {boolean}
*/
var twoSumBSTs = function(root1, root2, target) {
};
```

TypeScript Solution:

```
/**
* Problem: Two Sum BSTs
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* class TreeNode {
* val: number
* left: TreeNode | null
* right: TreeNode | null
* constructor(val?: number, left?: TreeNode | null, right?: TreeNode | null)
{
* this.val = (val===undefined ? 0 : val)
* this.left = (left===undefined ? null : left)
* this.right = (right===undefined ? null : right)
* }
* }
*/

function twoSumBSTs(root1: TreeNode | null, root2: TreeNode | null, target:
number): boolean {

};
```

C# Solution:

```
/*
* Problem: Two Sum BSTs
* Difficulty: Medium
* Tags: array, tree, search, stack
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* public class TreeNode {
* public int val;
* public TreeNode left;
* public TreeNode right;
* public TreeNode(int val=0, TreeNode left=null, TreeNode right=null) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
public class Solution {
public bool TwoSumBSTs(TreeNode root1, TreeNode root2, int target) {

}
}
```

C Solution:

```
/*
* Problem: Two Sum BSTs
* Difficulty: Medium
* Tags: array, tree, search, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* struct TreeNode *left;
* struct TreeNode *right;
* };
*/
bool twoSumBSTs(struct TreeNode* root1, struct TreeNode* root2, int target) {

}
```

Go Solution:

```
// Problem: Two Sum BSTs
// Difficulty: Medium
// Tags: array, tree, search, stack
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

/**
* Definition for a binary tree node.
* type TreeNode struct {
* Val int
* Left *TreeNode
* Right *TreeNode
* }
*/
func twoSumBSTs(root1 *TreeNode, root2 *TreeNode, target int) bool {

}
```

Kotlin Solution:

```
/**
* Example:
* var ti = TreeNode(5)
* var v = ti.`val`
* Definition for a binary tree node.
* class TreeNode(var `val`: Int) {
* var left: TreeNode? = null
* var right: TreeNode? = null
* }
```

```
*/
class Solution {
fun twoSumBSTs(root1: TreeNode?, root2: TreeNode?, target: Int): Boolean {

}
}
```

Swift Solution:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* public var val: Int
* public var left: TreeNode?
* public var right: TreeNode?
* public init() { self.val = 0; self.left = nil; self.right = nil; }
* public init(_ val: Int) { self.val = val; self.left = nil; self.right =
nil; }
* public init(_ val: Int, _ left: TreeNode?, _ right: TreeNode?) {
* self.val = val
* self.left = left
* self.right = right
* }
* }
*/
class Solution {
func twoSumBSTs(_ root1: TreeNode?, _ root2: TreeNode?, _ target: Int) ->
Bool {

}
}
```

Rust Solution:

```
// Problem: Two Sum BSTs
// Difficulty: Medium
// Tags: array, tree, search, stack
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height
// Definition for a binary tree node.
// #[derive(Debug, PartialEq, Eq)]
// pub struct TreeNode {
// pub val: i32,
// pub left: Option<Rc<RefCell<TreeNode>>>,
// pub right: Option<Rc<RefCell<TreeNode>>>,
// }
//
// impl TreeNode {
// #[inline]
// pub fn new(val: i32) -> Self {
// TreeNode {
// val,
// left: None,
// right: None
// }
// }
// }
use std::rc::Rc;
use std::cell::RefCell;
impl Solution {
pub fn two_sum_bs_ts(root1: Option<Rc<RefCell<TreeNode>>>, root2:
Option<Rc<RefCell<TreeNode>>>, target: i32) -> bool {

}
}
```

Ruby Solution:

```
# Definition for a binary tree node.
# class TreeNode
# attr_accessor :val, :left, :right
# def initialize(val = 0, left = nil, right = nil)
# @val = val
# @left = left
# @right = right
# end
# end
# @param {TreeNode} root1
# @param {TreeNode} root2
# @param {Integer} target
# @return {Boolean}
def two_sum_bs_ts(root1, root2, target)

end
```

PHP Solution:

```
/**
* Definition for a binary tree node.
* class TreeNode {
* public $val = null;
* public $left = null;
* public $right = null;
* function __construct($val = 0, $left = null, $right = null) {
* $this->val = $val;
* $this->left = $left;
* $this->right = $right;
* }
* }
*/
class Solution {

/**
* @param TreeNode $root1
* @param TreeNode $root2
* @param Integer $target
* @return Boolean
*/
function twoSumBSTs($root1, $root2, $target) {

}
}
```

Dart Solution:

```
/**
* Definition for a binary tree node.
* class TreeNode {
* int val;
* TreeNode? left;
* TreeNode? right;
* TreeNode([this.val = 0, this.left, this.right]);
* }
*/
class Solution {
bool twoSumBSTs(TreeNode? root1, TreeNode? root2, int target) {

}
}
```

Scala Solution:

```
/**
* Definition for a binary tree node.
* class TreeNode(_value: Int = 0, _left: TreeNode = null, _right: TreeNode = null) {
* var value: Int = _value
* var left: TreeNode = _left
* var right: TreeNode = _right
* }
*/
object Solution {
def twoSumBSTs(root1: TreeNode, root2: TreeNode, target: Int): Boolean = {

}
}
```

Elixir Solution:

```
# Definition for a binary tree node.
#
# defmodule TreeNode do
# @type t :: %__MODULE__{
# val: integer,
# left: TreeNode.t() | nil,
# right: TreeNode.t() | nil
# }
# defstruct val: 0, left: nil, right: nil
# end

defmodule Solution do
```

```
@spec two_sum_bs_ts(root1 :: TreeNode.t | nil, root2 :: TreeNode.t | nil,
target :: integer) :: boolean
def two_sum_bs_ts(root1, root2, target) do

end
end
```

Erlang Solution:

```
%% Definition for a binary tree node.
%%
%% -record(tree_node, {val = 0 :: integer(),
%% left = null :: 'null' | #tree_node{},
%% right = null :: 'null' | #tree_node{}}).

-spec two_sum_bs_ts(Root1 :: #tree_node{} | null, Root2 :: #tree_node{} |
null, Target :: integer()) -> boolean().
two_sum_bs_ts(Root1, Root2, Target) ->
.
```

Racket Solution:

```
; Definition for a binary tree node.
#|

; val : integer?
; left : (or/c tree-node? #f)
; right : (or/c tree-node? #f)
(struct tree-node
(val left right) #:mutable #:transparent)

; constructor
(define (make-tree-node [val 0])
(tree-node val #f #f))

|#

(define/contract (two-sum-bs-ts root1 root2 target)
(-> (or/c tree-node? #f) (or/c tree-node? #f) exact-integer? boolean?)
)
```