

Problem 437: Path Sum III

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given the

root

of a binary tree and an integer

targetSum

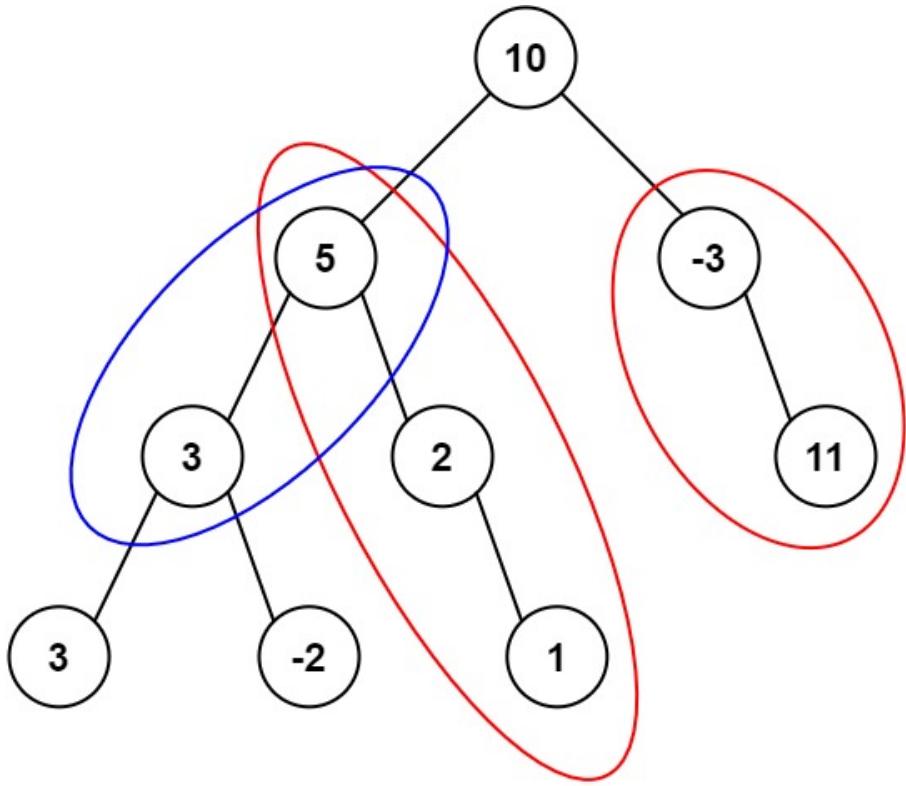
, return

the number of paths where the sum of the values along the path equals

targetSum

The path does not need to start or end at the root or a leaf, but it must go downwards (i.e., traveling only from parent nodes to child nodes).

Example 1:



Input:

```
root = [10,5,-3,3,2,null,11,3,-2,null,1], targetSum = 8
```

Output:

3

Explanation:

The paths that sum to 8 are shown.

Example 2:

Input:

```
root = [5,4,8,11,null,13,4,7,2,null,null,5,1], targetSum = 22
```

Output:

3

Constraints:

The number of nodes in the tree is in the range

[0, 1000]

.

-10

9

$\leq \text{Node.val} \leq 10$

9

$-1000 \leq \text{targetSum} \leq 1000$

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:
    int pathSum(TreeNode* root, int targetSum) {
```

```
}
```

```
};
```

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 int pathSum(TreeNode root, int targetSum) {  
  
    }  
}
```

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 pathSum(self, root: Optional[TreeNode], targetSum: int) -> int:
```

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 pathSum(self, root, targetSum):
        """
:type root: Optional[TreeNode]
:type targetSum: int
:rtype: int
"""

```

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} root
 * @param {number} targetSum
 * @return {number}
 */
var pathSum = function(root, targetSum) {

};

```

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 pathSum(root: TreeNode | null, targetSum: number): number {
}

```

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 int PathSum(TreeNode root, int targetSum) {
    }
}

```

C:

```

    /**
     * Definition for a binary tree node.
     * struct TreeNode {
     *     int val;
     *     struct TreeNode *left;
     *     struct TreeNode *right;
     * };
     */
int pathSum(struct TreeNode* root, int targetSum) {

```

```
}
```

Go:

```
/**  
 * Definition for a binary tree node.  
 * type TreeNode struct {  
 *     Val int  
 *     Left *TreeNode  
 *     Right *TreeNode  
 * }  
 */  
func pathSum(root *TreeNode, targetSum int) int {  
  
}
```

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 pathSum(root: TreeNode?, targetSum: Int): Int {  
 *         ...  
 *     }  
 * }
```

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 pathSum(_ root: TreeNode?, _ targetSum: Int) -> Int {

}
}

```

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 path_sum(root: Option<Rc<RefCell<TreeNode>>,
                 target_sum: i32) -> i32
  {

```

```
}
```

```
}
```

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} root
# @param {Integer} target_sum
# @return {Integer}
def path_sum(root, target_sum)

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 $root
 * @param Integer $targetSum
```

```
* @return Integer
*/
function pathSum($root, $targetSum) {

}
}
```

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 {
int pathSum(TreeNode? root, int targetSum) {
}

}
```

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 pathSum(root: TreeNode, targetSum: Int): Int = {

}
```

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 path_sum(root :: TreeNode.t() | nil, target_sum :: integer) :: integer
def path_sum(root, target_sum) 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 path_sum(Root :: #tree_node{} | null, TargetSum :: integer()) ->
integer().
path_sum(Root, TargetSum) ->
.
.
```

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 (path-sum root targetSum)
  (-> (or/c tree-node? #f) exact-integer? exact-integer?))
)

```

Solutions

C++ Solution:

```

/*
 * Problem: Path Sum III
 * Difficulty: Medium
 * Tags: tree, search
 *
 * Approach: DFS or BFS traversal
 * Time Complexity: O(n) where n is number of nodes
 * 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
 *     }
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {
 *         // TODO: Implement optimized solution
 *     }
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
 *     right(right) {
 *         // TODO: Implement optimized solution
 *     }
 */

```

```

        return 0;
    }
* };
*/
class Solution {
public:
int pathSum(TreeNode* root, int targetSum) {

}
};


```

Java Solution:

```

/**
 * Problem: Path Sum III
 * Difficulty: Medium
 * Tags: tree, search
 *
 * Approach: DFS or BFS traversal
 * Time Complexity: O(n) where n is number of nodes
 * 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 int pathSum(TreeNode root, int targetSum) {  
  
    }  
}
```

Python3 Solution:

```
"""  
  
Problem: Path Sum III  
Difficulty: Medium  
Tags: tree, search  
  
Approach: DFS or BFS traversal  
Time Complexity: O(n) where n is number of nodes  
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 pathSum(self, root: Optional[TreeNode], targetSum: int) -> int:  
        # 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 pathSum(self, root, targetSum):  
        """  
        :type root: Optional[TreeNode]
```

```
:type targetSum: int
:rtype: int
"""

```

JavaScript Solution:

```
/**
 * Problem: Path Sum III
 * Difficulty: Medium
 * Tags: tree, search
 *
 * Approach: DFS or BFS traversal
 * Time Complexity: O(n) where n is number of nodes
 * 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} root
 * @param {number} targetSum
 * @return {number}
 */
var pathSum = function(root, targetSum) {

};


```

TypeScript Solution:

```
/**
 * Problem: Path Sum III
 * Difficulty: Medium
 * Tags: tree, search
 *
 * Approach: DFS or BFS traversal

```

```

* Time Complexity: O(n) where n is number of nodes
* 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 pathSum(root: TreeNode | null, targetSum: number): number {
}

```

C# Solution:

```

/*
 * Problem: Path Sum III
 * Difficulty: Medium
 * Tags: tree, search
 *
 * Approach: DFS or BFS traversal
 * Time Complexity: O(n) where n is number of nodes
 * 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 int PathSum(TreeNode root, int targetSum) {

}
}

```

C Solution:

```

/*
* Problem: Path Sum III
* Difficulty: Medium
* Tags: tree, search
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* 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;
* };
*
int pathSum(struct TreeNode* root, int targetSum) {

}

```

Go Solution:

```

// Problem: Path Sum III
// Difficulty: Medium

```

```

// Tags: tree, search
//
// Approach: DFS or BFS traversal
// Time Complexity: O(n) where n is number of nodes
// 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 pathSum(root *TreeNode, targetSum int) int {

}

```

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 pathSum(root: TreeNode?, targetSum: Int): Int {
        if (root == null) return 0
        val sum = root.`val` + targetSum
        if (sum == 0) return 1
        return pathSum(root.left, sum) + pathSum(root.right, sum)
    }
}

```

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 pathSum(_ root: TreeNode?, _ targetSum: Int) -> Int {
}

}
}

```

Rust Solution:

```

// Problem: Path Sum III
// Difficulty: Medium
// Tags: tree, search
//
// Approach: DFS or BFS traversal
// Time Complexity: O(n) where n is number of nodes
// 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 path_sum(root: Option<Rc<RefCell<TreeNode>>>, target_sum: i32) -> i32
    {
        }

    }
}

```

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} root
# @param {Integer} target_sum
# @return {Integer}
def path_sum(root, target_sum)

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 $root
     * @param Integer $targetSum
     * @return Integer
     */
    function pathSum($root, $targetSum) {

    }
}

```

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 {
int pathSum(TreeNode? root, int targetSum) {

}
}

```

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 pathSum(root: TreeNode, targetSum: Int): Int = {
        }
    }
}

```

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 path_sum(TreeNode.t() | nil, integer()) :: integer()
def path_sum(root, target_sum) 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 path_sum(Root :: #tree_node{} | null, TargetSum :: integer()) ->
integer().

```

```
path_sum(Root, TargetSum) ->
.
```

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 (path-sum root targetSum)
  (-> (or/c tree-node? #f) exact-integer? exact-integer?))
)
```