Problem 3468: Find the Number of Copy Arrays

Problem Information

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

Problem Description

You are given an array

original

of length

n

and a 2D array

bounds

of length

n x 2

, where

bounds[i] = [u

i

, v

i

]

.

You need to find the number of

possible

arrays

copy

of length

n

such that:

(copy[i] - copy[i - 1]) == (original[i] - original[i - 1])

for

1 <= i <= n - 1

.

u

i

<= copy[i] <= v

i

for

0 <= i <= n - 1

.

Return the number of such arrays.

Example 1:

Input:

original = [1,2,3,4], bounds = [[1,2],[2,3],[3,4],[4,5]]

Output:

2

Explanation:

The possible arrays are:

[1, 2, 3, 4]

[2, 3, 4, 5]

Example 2:

Input:

original = [1,2,3,4], bounds = [[1,10],[2,9],[3,8],[4,7]]

Output:

4

Explanation:

The possible arrays are:

[1, 2, 3, 4]

[2, 3, 4, 5]

[3, 4, 5, 6]

[4, 5, 6, 7]

Example 3:

Input:

original = [1,2,1,2], bounds = [[1,1],[2,3],[3,3],[2,3]]

Output:

0

Explanation:

No array is possible.

Constraints:

2 <= n == original.length <= 10

5

1 <= original[i] <= 10

9

bounds.length == n

bounds[i].length == 2

1 <= bounds[i][0] <= bounds[i][1] <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int countArrays(vector<int>& original, vector<vector<int>>& bounds) {

}
};
```

Java:

```
class Solution {
public int countArrays(int[] original, int[][] bounds) {

}
}
```

Python3:

```
class Solution:
def countArrays(self, original: List[int], bounds: List[List[int]]) -> int:
```

Python:

```
class Solution(object):
def countArrays(self, original, bounds):
"""
:type original: List[int]
:type bounds: List[List[int]]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} original
* @param {number[][]} bounds
* @return {number}
*/
var countArrays = function(original, bounds) {

};
```

TypeScript:

```
function countArrays(original: number[], bounds: number[][]): number {

};
```

C#:

```
public class Solution {
public int CountArrays(int[] original, int[][] bounds) {

}
}
```

C:

```
int countArrays(int* original, int originalSize, int** bounds, int
boundsSize, int* boundsColSize) {

}
```

Go:

```
func countArrays(original []int, bounds [][]int) int {

}
```

Kotlin:

```
class Solution {
fun countArrays(original: IntArray, bounds: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func countArrays(_ original: [Int], _ bounds: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_arrays(original: Vec<i32>, bounds: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} original
# @param {Integer[][]} bounds
# @return {Integer}
def count_arrays(original, bounds)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $original
* @param Integer[][] $bounds
* @return Integer
*/
function countArrays($original, $bounds) {

}
}
```

Dart:

```
class Solution {
int countArrays(List<int> original, List<List<int>> bounds) {

}
}
```

Scala:

```
object Solution {
def countArrays(original: Array[Int], bounds: Array[Array[Int]]): Int = {
```

```
}
}
```

Elixir:

```
defmodule Solution do
@spec count_arrays(original :: [integer], bounds :: [[integer]]) :: integer
def count_arrays(original, bounds) do

end
end
```

Erlang:

```
-spec count_arrays(Original :: [integer()], Bounds :: [[integer()]]) ->
integer().
count_arrays(Original, Bounds) ->
.
```

Racket:

```
(define/contract (count-arrays original bounds)
(-> (listof exact-integer?) (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Number of Copy Arrays
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
class Solution {
public:
int countArrays(vector<int>& original, vector<vector<int>>& bounds) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Number of Copy Arrays
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int countArrays(int[] original, int[][] bounds) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Number of Copy Arrays
Difficulty: Medium
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def countArrays(self, original: List[int], bounds: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countArrays(self, original, bounds):
"""
:type original: List[int]
:type bounds: List[List[int]]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Number of Copy Arrays
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} original
* @param {number[][]} bounds
* @return {number}
*/
var countArrays = function(original, bounds) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Number of Copy Arrays
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
function countArrays(original: number[], bounds: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Find the Number of Copy Arrays
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int CountArrays(int[] original, int[][] bounds) {

}
}
```

C Solution:

```
/*
* Problem: Find the Number of Copy Arrays
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int countArrays(int* original, int originalSize, int** bounds, int
boundsSize, int* boundsColSize) {

}
```

Go Solution:

```
// Problem: Find the Number of Copy Arrays
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func countArrays(original []int, bounds [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countArrays(original: IntArray, bounds: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func countArrays(_ original: [Int], _ bounds: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Number of Copy Arrays
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn count_arrays(original: Vec<i32>, bounds: Vec<Vec<i32>>) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {Integer[]} original
# @param {Integer[][]} bounds
# @return {Integer}
def count_arrays(original, bounds)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $original
* @param Integer[][] $bounds
* @return Integer
*/
function countArrays($original, $bounds) {

}
}
```

Dart Solution:

```
class Solution {
int countArrays(List<int> original, List<List<int>> bounds) {

}
}
```

Scala Solution:

```
object Solution {
def countArrays(original: Array[Int], bounds: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_arrays(original :: [integer], bounds :: [[integer]]) :: integer
def count_arrays(original, bounds) do

end
end
```

Erlang Solution:

```
-spec count_arrays(Original :: [integer()], Bounds :: [[integer()]]) ->
integer().
count_arrays(Original, Bounds) ->
.
```

Racket Solution:

```
(define/contract (count-arrays original bounds)
(-> (listof exact-integer?) (listof (listof exact-integer?)) exact-integer?)
)
```