Problem 2145: Count the Hidden Sequences

Problem Information

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

Problem Description

You are given a

0-indexed

array of

n

integers

differences

, which describes the

differences

between each pair of

consecutive

integers of a

hidden

sequence of length

(n + 1)

. More formally, call the hidden sequence

hidden

, then we have that

differences[i] = hidden[i + 1] - hidden[i]

.

You are further given two integers

lower

and

upper

that describe the

inclusive

range of values

[lower, upper]

that the hidden sequence can contain.

For example, given

differences = [1, -3, 4]

,

lower = 1

,

upper = 6

, the hidden sequence is a sequence of length

4

whose elements are in between

1

and

6

(

inclusive

).

[3, 4, 1, 5]

and

[4, 5, 2, 6]

are possible hidden sequences.

[5, 6, 3, 7]

is not possible since it contains an element greater than

6

.

[1, 2, 3, 4]

is not possible since the differences are not correct.

Return

the number of

possible

hidden sequences there are.

If there are no possible sequences, return

0

.

Example 1:

Input:

differences = [1,-3,4], lower = 1, upper = 6

Output:

2

Explanation:

The possible hidden sequences are: - [3, 4, 1, 5] - [4, 5, 2, 6] Thus, we return 2.

Example 2:

Input:

differences = [3,-4,5,1,-2], lower = -4, upper = 5

Output:

4

Explanation:

The possible hidden sequences are: - [-3, 0, -4, 1, 2, 0] - [-2, 1, -3, 2, 3, 1] - [-1, 2, -2, 3, 4, 2] - [0, 3, -1, 4, 5, 3] Thus, we return 4.

Example 3:

Input:

differences = [4,-7,2], lower = 3, upper = 6

Output:

0

Explanation:

There are no possible hidden sequences. Thus, we return 0.

Constraints:

n == differences.length

1 <= n <= 10

5

-10

5

<= differences[i] <= 10

5

-10

5

<= lower <= upper <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int numberOfArrays(vector<int>& differences, int lower, int upper) {

}
};
```

Java:

```
class Solution {
public int numberOfArrays(int[] differences, int lower, int upper) {

}
}
```

Python3:

```
class Solution:
def numberOfArrays(self, differences: List[int], lower: int, upper: int) ->
int:
```

Python:

```
class Solution(object):
def numberOfArrays(self, differences, lower, upper):
"""
:type differences: List[int]
:type lower: int
:type upper: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} differences
* @param {number} lower
* @param {number} upper
* @return {number}
*/
var numberOfArrays = function(differences, lower, upper) {

};
```

TypeScript:

```
function numberOfArrays(differences: number[], lower: number, upper: number):
number {

};
```

C#:

```
public class Solution {
public int NumberOfArrays(int[] differences, int lower, int upper) {

}
}
```

C:

```
int numberOfArrays(int* differences, int differencesSize, int lower, int
upper) {

}
```

Go:

```
func numberOfArrays(differences []int, lower int, upper int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfArrays(differences: IntArray, lower: Int, upper: Int): Int {

}
}
```

Swift:

```
class Solution {
func numberOfArrays(_ differences: [Int], _ lower: Int, _ upper: Int) -> Int
{

}
}
```

Rust:

```
impl Solution {
pub fn number_of_arrays(differences: Vec<i32>, lower: i32, upper: i32) -> i32
{

}
}
```

Ruby:

```
# @param {Integer[]} differences
# @param {Integer} lower
# @param {Integer} upper
# @return {Integer}
def number_of_arrays(differences, lower, upper)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $differences
* @param Integer $lower
* @param Integer $upper
* @return Integer
```

```
*/
function numberOfArrays($differences, $lower, $upper) {

}
}
```

Dart:

```
class Solution {
int numberOfArrays(List<int> differences, int lower, int upper) {

}
}
```

Scala:

```
object Solution {
def numberOfArrays(differences: Array[Int], lower: Int, upper: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_arrays(differences :: [integer], lower :: integer, upper ::
integer) :: integer
def number_of_arrays(differences, lower, upper) do

end
end
```

Erlang:

```
-spec number_of_arrays(Differences :: [integer()], Lower :: integer(), Upper
:: integer()) -> integer().
number_of_arrays(Differences, Lower, Upper) ->
.
```

Racket:

```
(define/contract (number-of-arrays differences lower upper)
(-> (listof exact-integer?) exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count the Hidden Sequences
* Difficulty: Medium
* Tags: array
*
* 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 numberOfArrays(vector<int>& differences, int lower, int upper) {

}
};
```

Java Solution:

```
/**
* Problem: Count the Hidden Sequences
* Difficulty: Medium
* Tags: array
*
* 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 numberOfArrays(int[] differences, int lower, int upper) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Count the Hidden Sequences
Difficulty: Medium
Tags: array

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 numberOfArrays(self, differences: List[int], lower: int, upper: int) ->
int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberOfArrays(self, differences, lower, upper):
"""
:type differences: List[int]
:type lower: int
:type upper: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count the Hidden Sequences
* Difficulty: Medium
* Tags: array
*
* 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[]} differences
* @param {number} lower
* @param {number} upper
* @return {number}
*/
var numberOfArrays = function(differences, lower, upper) {

};
```

TypeScript Solution:

```
/**
* Problem: Count the Hidden Sequences
* Difficulty: Medium
* Tags: array
*
* 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 numberOfArrays(differences: number[], lower: number, upper: number):
number {

};
```

C# Solution:

```
/*
* Problem: Count the Hidden Sequences
* Difficulty: Medium
* Tags: array
*
* 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 NumberOfArrays(int[] differences, int lower, int upper) {

}
}
```

C Solution:

```
/*
* Problem: Count the Hidden Sequences
* Difficulty: Medium
* Tags: array
*
* 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 numberOfArrays(int* differences, int differencesSize, int lower, int
upper) {

}
```

Go Solution:

```
// Problem: Count the Hidden Sequences
// Difficulty: Medium
// Tags: array
//
// 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 numberOfArrays(differences []int, lower int, upper int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfArrays(differences: IntArray, lower: Int, upper: Int): Int {
```

```
}
}
```

Swift Solution:

```
class Solution {
func numberOfArrays(_ differences: [Int], _ lower: Int, _ upper: Int) -> Int
{

}
}
```

Rust Solution:

```
// Problem: Count the Hidden Sequences
// Difficulty: Medium
// Tags: array
//
// 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 number_of_arrays(differences: Vec<i32>, lower: i32, upper: i32) -> i32
{

}
}
```

Ruby Solution:

```
# @param {Integer[]} differences
# @param {Integer} lower
# @param {Integer} upper
# @return {Integer}
def number_of_arrays(differences, lower, upper)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $differences
* @param Integer $lower
* @param Integer $upper
* @return Integer
*/
function numberOfArrays($differences, $lower, $upper) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfArrays(List<int> differences, int lower, int upper) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfArrays(differences: Array[Int], lower: Int, upper: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_arrays(differences :: [integer], lower :: integer, upper ::
integer) :: integer
def number_of_arrays(differences, lower, upper) do

end
end
```

Erlang Solution:

```
-spec number_of_arrays(Differences :: [integer()], Lower :: integer(), Upper
:: integer()) -> integer().
number_of_arrays(Differences, Lower, Upper) ->
.
```

Racket Solution:

```
(define/contract (number-of-arrays differences lower upper)
(-> (listof exact-integer?) exact-integer? exact-integer? exact-integer?)
)
```