Problem 2270: Number of Ways to Split Array

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

of length

n

.

nums

contains a

valid split

at index

i

if the following are true:

The sum of the first

i + 1

elements is

greater than or equal to

the sum of the last

n - i - 1

elements.

There is

at least one

element to the right of

i

. That is,

0 <= i < n - 1

.

Return

the number of

valid splits

in

nums

.

Example 1:

Input:

nums = [10,4,-8,7]

Output:

2

Explanation:

There are three ways of splitting nums into two non-empty parts: - Split nums at index 0. Then, the first part is [10], and its sum is 10. The second part is [4,-8,7], and its sum is 3. Since 10 >= 3, i = 0 is a valid split. - Split nums at index 1. Then, the first part is [10,4], and its sum is 14. The second part is [-8,7], and its sum is -1. Since 14 >= -1, i = 1 is a valid split. - Split nums at index 2. Then, the first part is [10,4,-8], and its sum is 6. The second part is [7], and its sum is 7. Since 6 < 7, i = 2 is not a valid split. Thus, the number of valid splits in nums is 2.

Example 2:

Input:

nums = [2,3,1,0]

Output:

2

Explanation:

There are two valid splits in nums: - Split nums at index 1. Then, the first part is [2,3], and its sum is 5. The second part is [1,0], and its sum is 1. Since 5 >= 1, i = 1 is a valid split. - Split nums at index 2. Then, the first part is [2,3,1], and its sum is 6. The second part is [0], and its sum is 0. Since 6 >= 0, i = 2 is a valid split.

Constraints:

2 <= nums.length <= 10

5

-10

5

<= nums[i] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int waysToSplitArray(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int waysToSplitArray(int[] nums) {

}
}
```

Python3:

```
class Solution:
def waysToSplitArray(self, nums: List[int]) -> int:
```

Python:

```
class Solution(object):
def waysToSplitArray(self, nums):
```

```
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @return {number}
*/
var waysToSplitArray = function(nums) {

};
```

TypeScript:

```
function waysToSplitArray(nums: number[]): number {

};
```

C#:

```
public class Solution {
public int WaysToSplitArray(int[] nums) {

}
}
```

C:

```
int waysToSplitArray(int* nums, int numsSize) {

}
```

Go:

```
func waysToSplitArray(nums []int) int {

}
```

Kotlin:

```
class Solution {
fun waysToSplitArray(nums: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func waysToSplitArray(_ nums: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn ways_to_split_array(nums: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer}
def ways_to_split_array(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function waysToSplitArray($nums) {

}
}
```

Dart:

```
class Solution {
int waysToSplitArray(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def waysToSplitArray(nums: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec ways_to_split_array(nums :: [integer]) :: integer
def ways_to_split_array(nums) do

end
end
```

Erlang:

```
-spec ways_to_split_array(Nums :: [integer()]) -> integer().
ways_to_split_array(Nums) ->
.
```

Racket:

```
(define/contract (ways-to-split-array nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Ways to Split Array
* 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 waysToSplitArray(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Ways to Split Array
* 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 waysToSplitArray(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Ways to Split Array
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 waysToSplitArray(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def waysToSplitArray(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Ways to Split Array
* 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[]} nums
* @return {number}
*/
var waysToSplitArray = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Ways to Split Array
* 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 waysToSplitArray(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Number of Ways to Split Array
* 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 WaysToSplitArray(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Number of Ways to Split Array
* 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 waysToSplitArray(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Number of Ways to Split Array
// 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 waysToSplitArray(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun waysToSplitArray(nums: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func waysToSplitArray(_ nums: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Ways to Split Array
// 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 ways_to_split_array(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def ways_to_split_array(nums)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function waysToSplitArray($nums) {

}
}
```

Dart Solution:

```
class Solution {
int waysToSplitArray(List<int> nums) {

}
}
```

Scala Solution:

```
object Solution {
def waysToSplitArray(nums: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec ways_to_split_array(nums :: [integer]) :: integer
def ways_to_split_array(nums) do

end
end
```

Erlang Solution:

```
-spec ways_to_split_array(Nums :: [integer()]) -> integer().
ways_to_split_array(Nums) ->
.
```

Racket Solution:

```
(define/contract (ways-to-split-array nums)
(-> (listof exact-integer?) exact-integer?)
)
```