2070. Number of ways to split array

2270. Number of Ways to Split Array

Solved

Medium | Topics | Companies | Hint

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.
```

A ideia é calcular, primeiramente, a soma total do array de entrada. Com a soma em mãos, é possível, com uma soma parcial, encontrar a soma do restante dos valores.

Ex: [10, 4, -8, 7] total = 13

soma parcial partindo do final do array:

[10, 4, -8, 7] Partial sum: 7

a soma dos elementos 10, 4, -8, pode ser encontrada usando:

total - Partial_sum (PS)

neste caso: 13 - 7 = 6 = 10 + 4 + (-8).

Usando essa lógica, basta realizar a verificação: total - Partial_sum >= Partial_sum

na próxima iteração: ↓

[10, 4, -8, 7] PS = -1

$$Total - PS = 13 - (-1) = 14$$

$14 > -1$
$Total - PS > PS$
} Portanto é um caso válido.

Análise

O algoritmo é $O(n)$ em tempo e $O(1)$ em espaço