Problem 3728: Stable Subarrays With Equal Boundary and Interior Sum

Problem Information

Difficulty: Medium
Acceptance Rate: 25.16%
Paid Only: No
Tags: Array, Hash Table, Prefix Sum

Problem Description

You are given an integer array `capacity`.

A subarray `capacity[l..r]` is considered **stable** if:

* Its length is **at least** 3. * The **first** and **last** elements are each equal to the **sum** of all elements **strictly between** them (i.e., `capacity[l] = capacity[r] = capacity[l + 1] + capacity[l + 2] + ... + capacity[r - 1]`).

Return an integer denoting the number of **stable subarrays**.

Example 1:

Input: capacity = [9,3,3,3,9]

Output: 2

Explanation:

* `[9,3,3,3,9]` is stable because the first and last elements are both 9, and the sum of the elements strictly between them is `3 + 3 + 3 = 9`. * `[3,3,3]` is stable because the first and last elements are both 3, and the sum of the elements strictly between them is 3.

Example 2:

Input: capacity = [1,2,3,4,5]

Output: 0

Explanation:

No subarray of length at least 3 has equal first and last elements, so the answer is 0.

Example 3:

Input: capacity = [-4,4,0,0,-8,-4]

Output: 1

Explanation:

`[-4,4,0,0,-8,-4]` is stable because the first and last elements are both -4, and the sum of the elements strictly between them is `4 + 0 + 0 + (-8) = -4`

Constraints:

* `3 <= capacity.length <= 105` * `-109 <= capacity[i] <= 109`

Code Snippets

C++:

```
class Solution {
public:
long long countStableSubarrays(vector<int>& capacity) {

}
};
```

Java:

```
class Solution {
public long countStableSubarrays(int[] capacity) {
```

```
}
}
```

Python3:

```
class Solution:
def countStableSubarrays(self, capacity: List[int]) -> int:
```