Problem 3473: Sum of K Subarrays With Length at Least M

Problem Information

Difficulty: Medium
Acceptance Rate: 25.45%
Paid Only: No
Tags: Array, Dynamic Programming, Prefix Sum

Problem Description

You are given an integer array `nums` and two integers, `k` and `m`.

Return the **maximum** sum of `k` non-overlapping subarrays of `nums`, where each subarray has a length of **at least** `m`.

Example 1:

Input: nums = [1,2,-1,3,3,4], k = 2, m = 2

Output: 13

Explanation:

The optimal choice is:

* Subarray `nums[3..5]` with sum `3 + 3 + 4 = 10` (length is `3 >= m`). * Subarray `nums[0..1]` with sum `1 + 2 = 3` (length is `2 >= m`).

The total sum is `10 + 3 = 13`.

Example 2:

Input: nums = [-10,3,-1,-2], k = 4, m = 1

Output: -10

Explanation:

The optimal choice is choosing each element as a subarray. The output is `(-10) + 3 + (-1) + (-2) = -10`.

Constraints:

* `1 <= nums.length <= 2000` * `-104 <= nums[i] <= 104` * `1 <= k <= floor(nums.length / m)` * `1 <= m <= 3`

Code Snippets

C++:

```
class Solution {
public:
int maxSum(vector<int>& nums, int k, int m) {

}
};
```

Java:

```
class Solution {
public int maxSum(int[] nums, int k, int m) {

}
}
```

Python3:

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