Problem 3689: Maximum Total Subarray Value I

Problem Information

Difficulty: Medium
Acceptance Rate: 60.99%
Paid Only: No
Tags: Array, Greedy

Problem Description

You are given an integer array `nums` of length `n` and an integer `k`.

You need to choose **exactly** `k` non-empty subarrays `nums[l..r]` of `nums`. Subarrays may overlap, and the exact same subarray (same `l` and `r`) **can** be chosen more than once.

The **value** of a subarray `nums[l..r]` is defined as: `max(nums[l..r]) - min(nums[l..r])`.

The **total value** is the sum of the **values** of all chosen subarrays.

Return the **maximum** possible total value you can achieve.

Example 1:

Input: nums = [1,3,2], k = 2

Output: 4

Explanation:

One optimal approach is:

* Choose `nums[0..1] = [1, 3]`. The maximum is 3 and the minimum is 1, giving a value of `3 - 1 = 2`. * Choose `nums[0..2] = [1, 3, 2]`. The maximum is still 3 and the minimum is still 1, so the value is also `3 - 1 = 2`.

Adding these gives `2 + 2 = 4`.

Example 2:

Input: nums = [4,2,5,1], k = 3

Output: 12

Explanation:

One optimal approach is:

* Choose `nums[0..3] = [4, 2, 5, 1]`. The maximum is 5 and the minimum is 1, giving a value of `5 - 1 = 4`. * Choose `nums[0..3] = [4, 2, 5, 1]`. The maximum is 5 and the minimum is 1, so the value is also `4`. * Choose `nums[2..3] = [5, 1]`. The maximum is 5 and the minimum is 1, so the value is again `4`.

Adding these gives `4 + 4 + 4 = 12`.

Constraints:

* `1 <= n == nums.length <= 5 * 10■■■■■■■4` * `0 <= nums[i] <= 109` * `1 <= k <= 105`

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public long maxTotalValue(int[] nums, int k) {
}
}
```

Python3:

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