Problem 3652: Best Time to Buy and Sell Stock using Strategy

Problem Information

Difficulty: Medium
Acceptance Rate: 46.87%
Paid Only: No
Tags: Array, Sliding Window, Prefix Sum

Problem Description

You are given two integer arrays `prices` and `strategy`, where:

* `prices[i]` is the price of a given stock on the `ith` day. * `strategy[i]` represents a trading action on the `ith` day, where: * `-1` indicates buying one unit of the stock. * `0` indicates holding the stock. * `1` indicates selling one unit of the stock.

You are also given an **even** integer `k`, and may perform **at most one** modification to `strategy`. A modification consists of:

* Selecting exactly `k` **consecutive** elements in `strategy`. * Set the **first** `k / 2` elements to `0` (hold). * Set the **last** `k / 2` elements to `1` (sell).

The **profit** is defined as the **sum** of `strategy[i] * prices[i]` across all days.

Return the **maximum** possible profit you can achieve.

Note: There are no constraints on budget or stock ownership, so all buy and sell operations are feasible regardless of past actions.

Example 1:

Input: prices = [4,2,8], strategy = [-1,0,1], k = 2

Output: 10

Explanation:

Modification | Strategy | Profit Calculation | Profit ---|---|---|--- Original | [-1, 0, 1] | (-1 × 4) + (0 × 2) + (1 × 8) = -4 + 0 + 8 | 4 Modify [0, 1] | [0, 1, 1] | (0 × 4) + (1 × 2) + (1 × 8) = 0 + 2 + 8 | 10 Modify [1, 2] | [-1, 0, 1] | (-1 × 4) + (0 × 2) + (1 × 8) = -4 + 0 + 8 | 4 Thus, the maximum possible profit is 10, which is achieved by modifying the subarray `[0, 1]`■■■■■■■.

Example 2:

Input: prices = [5,4,3], strategy = [1,1,0], k = 2

Output: 9

Explanation:

Modification | Strategy | Profit Calculation | Profit ---|---|---|--- Original | [1, 1, 0] | (1 × 5) + (1 × 4) + (0 × 3) = 5 + 4 + 0 | 9 Modify [0, 1] | [0, 1, 0] | (0 × 5) + (1 × 4) + (0 × 3) = 0 + 4 + 0 | 4 Modify [1, 2] | [1, 0, 1] | (1 × 5) + (0 × 4) + (1 × 3) = 5 + 0 + 3 | 8 Thus, the maximum possible profit is 9, which is achieved without any modification.

Constraints:

* `2 <= prices.length == strategy.length <= 105` * `1 <= prices[i] <= 105` * `-1 <= strategy[i] <= 1` * `2 <= k <= prices.length` * `k` is even

Code Snippets

C++:

```
class Solution {
public:
long long maxProfit(vector<int>& prices, vector<int>& strategy, int k) {

}
};
```

Java:

```
class Solution {
public long maxProfit(int[] prices, int[] strategy, int k) {

}
}
```

Python3:

```
class Solution:
def maxProfit(self, prices: List[int], strategy: List[int], k: int) -> int:
```