Problem 3738: Longest Non-Decreasing Subarray After Replacing at Most One Element

Problem Information

Difficulty: Medium
Acceptance Rate: 19.85%
Paid Only: No
Tags: Array, Dynamic Programming

Problem Description

You are given an integer array `nums`.

You are allowed to replace **at most** one element in the array with any other integer value of your choice.

Return the length of the **longest non-decreasing subarray** that can be obtained after performing at most one replacement.

An array is said to be **non-decreasing** if each element is greater than or equal to its previous one (if it exists).

Example 1:

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

Output: 4

Explanation:

Replacing `nums[3] = 1` with 3 gives the array [1, 2, 3, 3, 2].

The longest non-decreasing subarray is [1, 2, 3, 3], which has a length of 4.

Example 2:

Input: nums = [2,2,2,2,2]

Output: 5

Explanation:

All elements in `nums` are equal, so it is already non-decreasing and the entire `nums` forms a subarray of length 5.

Constraints:

* `1 <= nums.length <= 105` * `-109 <= nums[i] <= 109`

Code Snippets

C++:

```
class Solution {
public:
int longestSubarray(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int longestSubarray(int[] nums) {

}
}
```

Python3:

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