Problem 978: Longest Turbulent Subarray

Problem Information

Difficulty: Medium
Acceptance Rate: 48.51%
Paid Only: No
Tags: Array, Dynamic Programming, Sliding Window

Problem Description

Given an integer array `arr`, return _the length of a maximum size turbulent subarray of_ `arr`.

A subarray is **turbulent** if the comparison sign flips between each adjacent pair of elements in the subarray.

More formally, a subarray `[arr[i], arr[i + 1], ..., arr[j]]` of `arr` is said to be turbulent if and only if:

* For `i <= k < j`: * `arr[k] > arr[k + 1]` when `k` is odd, and * `arr[k] < arr[k + 1]` when `k` is even. * Or, for `i <= k < j`: * `arr[k] > arr[k + 1]` when `k` is even, and * `arr[k] < arr[k + 1]` when `k` is odd.

Example 1:

Input: arr = [9,4,2,10,7,8,8,1,9] **Output:** 5 **Explanation:** arr[1] > arr[2] < arr[3] > arr[4] < arr[5]

Example 2:

Input: arr = [4,8,12,16] **Output:** 2

Example 3:

Input: arr = [100] **Output:** 1

Constraints:

* `1 <= arr.length <= 4 * 104` * `0 <= arr[i] <= 109`

Code Snippets

C++:

```
class Solution {
public:
int maxTurbulenceSize(vector<int>& arr) {

}
};
```

Java:

```
class Solution {
public int maxTurbulenceSize(int[] arr) {

}
}
```

Python3:

```
class Solution:
def maxTurbulenceSize(self, arr: List[int]) -> int:
```