Problem 845: Longest Mountain in Array

Problem Information

Difficulty: Medium
Acceptance Rate: 41.45%
Paid Only: No
Tags: Array, Two Pointers, Dynamic Programming, Enumeration

Problem Description

You may recall that an array `arr` is a **mountain array** if and only if:

* `arr.length >= 3` * There exists some index `i` (**0-indexed**) with `0 < i < arr.length - 1` such that: * `arr[0] < arr[1] < ... < arr[i - 1] < arr[i]` * `arr[i] > arr[i + 1] > ... > arr[arr.length - 1]`

Given an integer array `arr`, return _the length of the longest subarray, which is a mountain_. Return `0` if there is no mountain subarray.

Example 1:

Input: arr = [2,1,4,7,3,2,5] **Output:** 5 **Explanation:** The largest mountain is [1,4,7,3,2] which has length 5.

Example 2:

Input: arr = [2,2,2] **Output:** 0 **Explanation:** There is no mountain.

Constraints:

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

Follow up:

* Can you solve it using only one pass? * Can you solve it in `O(1)` space?

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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