# 845. Longest Mountain in Array

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 <= 10<sup>4</sup>`
- `0 <= arr[i] <= 10<sup>4</sup>`

**Follow up:**

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

```
if(A.size() == 0)
            return 0;

        int maxLen = 0;
        for(int i=1; i<A.size()-1; i++)
        {
            if(A[i] > A[i+1] && A[i] > A[i-1])
```

```
            {
                int left = i-1;
                int right = i+1;

                while(left > 0 && A[left-1] < A[left])
                    left--;
                while(right < A.size()-1 && A[right+1] < A[right])
                    right++;
                maxLen = max(maxLen, right-left+1);
            }
        }
        return maxLen;
    }
```