Problem 845: Longest Mountain in Array

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

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 <= 10

4

0 <= arr[i] <= 10

4

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:
```

Python:

```
class Solution(object):
def longestMountain(self, arr):
"""
:type arr: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} arr
* @return {number}
*/
var longestMountain = function(arr) {

};
```

TypeScript:

```
function longestMountain(arr: number[]): number {

};
```

C#:

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

}
}
```

C:

```
int longestMountain(int* arr, int arrSize) {

}
```

Go:

```
func longestMountain(arr []int) int {

}
```

Kotlin:

```
class Solution {
fun longestMountain(arr: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func longestMountain(_ arr: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn longest_mountain(arr: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @return {Integer}
def longest_mountain(arr)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer
*/
function longestMountain($arr) {

}
}
```

Dart:

```
class Solution {
int longestMountain(List<int> arr) {

}
}
```

Scala:

```
object Solution {
def longestMountain(arr: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec longest_mountain(arr :: [integer]) :: integer
def longest_mountain(arr) do

end
end
```

Erlang:

```
-spec longest_mountain(Arr :: [integer()]) -> integer().
longest_mountain(Arr) ->
.
```

Racket:

```
(define/contract (longest-mountain arr)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Mountain in Array
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Longest Mountain in Array
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

```
}
}
```

Python3 Solution:

```
"""
Problem: Longest Mountain in Array
Difficulty: Medium
Tags: array, dp

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def longestMountain(self, arr: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def longestMountain(self, arr):
"""
:type arr: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Longest Mountain in Array
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

```
/**
* @param {number[]} arr
* @return {number}
*/
var longestMountain = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Mountain in Array
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function longestMountain(arr: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Longest Mountain in Array
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int LongestMountain(int[] arr) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Longest Mountain in Array
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int longestMountain(int* arr, int arrSize) {

}
```

Go Solution:

```
// Problem: Longest Mountain in Array
// Difficulty: Medium
// Tags: array, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func longestMountain(arr []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun longestMountain(arr: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func longestMountain(_ arr: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Longest Mountain in Array
// Difficulty: Medium
// Tags: array, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn longest_mountain(arr: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @return {Integer}
def longest_mountain(arr)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer
*/
function longestMountain($arr) {

}
}
```

Dart Solution:

```
class Solution {
int longestMountain(List<int> arr) {

}
}
```

Scala Solution:

```
object Solution {
def longestMountain(arr: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec longest_mountain(arr :: [integer]) :: integer
def longest_mountain(arr) do

end
end
```

Erlang Solution:

```
-spec longest_mountain(Arr :: [integer()]) -> integer().
longest_mountain(Arr) ->
.
```

Racket Solution:

```
(define/contract (longest-mountain arr)
(-> (listof exact-integer?) exact-integer?)
)
```