Problem 941: Valid Mountain Array

Problem Information

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

Problem Description

Given an array of integers

arr

, return

true

if and only if it is a valid mountain array

.

Recall that arr is a mountain array if and only if:

arr.length >= 3

There exists some

i

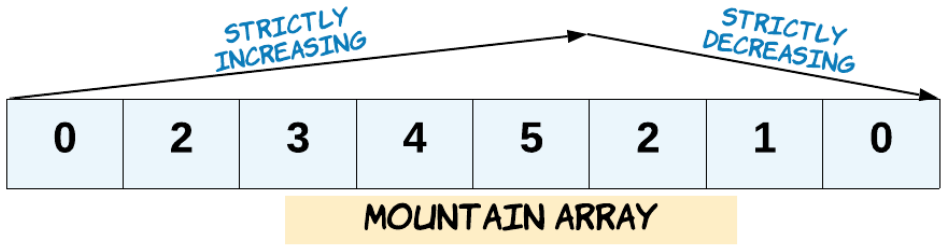
with

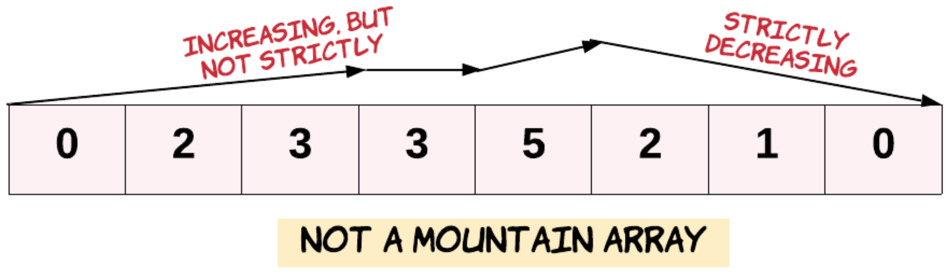
0 < i < arr.length - 1

such that:

arr[0] < arr[1] < ... < arr[i - 1] < arr[i]

arr[i] > arr[i + 1] > ... > arr[arr.length - 1]





Example 1:

Input:

arr = [2,1]

Output:

false

Example 2:

Input:

arr = [3,5,5]

Output:

false

Example 3:

Input:

arr = [0,3,2,1]

Output:

true

Constraints:

1 <= arr.length <= 10

4

0 <= arr[i] <= 10

4

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public boolean validMountainArray(int[] arr) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function validMountainArray(arr: number[]): boolean {

};
```

C#:

```
public class Solution {
public bool ValidMountainArray(int[] arr) {

}
}
```

C:

```



bool validMountainArray(int* arr, int arrSize){

}
```

Go:

```
func validMountainArray(arr []int) bool {

}
```

Kotlin:

```
class Solution {
fun validMountainArray(arr: IntArray): Boolean {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Scala:

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

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Valid Mountain Array
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Valid Mountain Array
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public boolean validMountainArray(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Valid Mountain Array
Difficulty: Easy
Tags: array

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Valid Mountain Array
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Valid Mountain Array
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function validMountainArray(arr: number[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Valid Mountain Array
* Difficulty: Easy
* Tags: array
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public bool ValidMountainArray(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Valid Mountain Array
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/



bool validMountainArray(int* arr, int arrSize){

}
```

Go Solution:

```
// Problem: Valid Mountain Array
// Difficulty: Easy
// Tags: array
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func validMountainArray(arr []int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun validMountainArray(arr: IntArray): Boolean {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Valid Mountain Array
// Difficulty: Easy
// Tags: array
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Scala Solution:

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

}
}
```