Problem 1671: Minimum Number of Removals to Make Mountain Array

Problem Information

Difficulty: Hard
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

nums

, return

the

minimum

number of elements to remove to make

nums

a

mountain array

.

Example 1:

Input:

nums = [1,3,1]

Output:

0

Explanation:

The array itself is a mountain array so we do not need to remove any elements.

Example 2:

Input:

nums = [2,1,1,5,6,2,3,1]

Output:

3

Explanation:

One solution is to remove the elements at indices 0, 1, and 5, making the array nums = [1,5,6,3,1].

Constraints:

3 <= nums.length <= 1000

1 <= nums[i] <= 10

9

It is guaranteed that you can make a mountain array out of

nums

.

Code Snippets

C++:

```
class Solution {
public:
int minimumMountainRemovals(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int minimumMountainRemovals(int[] nums) {

}
}
```

Python3:

```
class Solution:
def minimumMountainRemovals(self, nums: List[int]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minimumMountainRemovals(nums: number[]): number {

};
```

C#:

```
}
```

```
public class Solution {
public int MinimumMountainRemovals(int[] nums) {

}
}
```

C:

```
int minimumMountainRemovals(int* nums, int numsSize) {

}
```

Go:

```
func minimumMountainRemovals(nums []int) int {

}
```

Kotlin:

```
class Solution {
fun minimumMountainRemovals(nums: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func minimumMountainRemovals(_ nums: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_mountain_removals(nums: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer}
def minimum_mountain_removals(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function minimumMountainRemovals($nums) {

}
}
```

Dart:

```
class Solution {
int minimumMountainRemovals(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def minimumMountainRemovals(nums: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_mountain_removals(nums :: [integer]) :: integer
def minimum_mountain_removals(nums) do

end
end
```

Erlang:

```
-spec minimum_mountain_removals(Nums :: [integer()]) -> integer().
minimum_mountain_removals(Nums) ->
.
```

Racket:

```
(define/contract (minimum-mountain-removals nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Removals to Make Mountain Array
* Difficulty: Hard
* Tags: array, dp, greedy, search
*
* 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 minimumMountainRemovals(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Removals to Make Mountain Array
* Difficulty: Hard
* Tags: array, dp, greedy, search
*
* 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 minimumMountainRemovals(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Removals to Make Mountain Array
Difficulty: Hard
Tags: array, dp, greedy, search

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 minimumMountainRemovals(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Number of Removals to Make Mountain Array
* Difficulty: Hard
* Tags: array, dp, greedy, search
*
* 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[]} nums
* @return {number}
*/
var minimumMountainRemovals = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Removals to Make Mountain Array
* Difficulty: Hard
* Tags: array, dp, greedy, search
*
* 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 minimumMountainRemovals(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Removals to Make Mountain Array
* Difficulty: Hard
* Tags: array, dp, greedy, search
*
* 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 MinimumMountainRemovals(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Removals to Make Mountain Array
* Difficulty: Hard
* Tags: array, dp, greedy, search
*
* 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 minimumMountainRemovals(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Minimum Number of Removals to Make Mountain Array
// Difficulty: Hard
// Tags: array, dp, greedy, search
//
// 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 minimumMountainRemovals(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minimumMountainRemovals(nums: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func minimumMountainRemovals(_ nums: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Number of Removals to Make Mountain Array
// Difficulty: Hard
// Tags: array, dp, greedy, search
//
// 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 minimum_mountain_removals(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def minimum_mountain_removals(nums)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer[] $nums
* @return Integer
*/
function minimumMountainRemovals($nums) {

}
}
```

Dart Solution:

```
class Solution {
int minimumMountainRemovals(List<int> nums) {

}
}
```

Scala Solution:

```
object Solution {
def minimumMountainRemovals(nums: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_mountain_removals(nums :: [integer]) :: integer
def minimum_mountain_removals(nums) do

end
end
```

Erlang Solution:

```
-spec minimum_mountain_removals(Nums :: [integer()]) -> integer().
minimum_mountain_removals(Nums) ->
.
```

Racket Solution:

```
(define/contract (minimum-mountain-removals nums)
(-> (listof exact-integer?) exact-integer?)
)
```