Problem 2009: Minimum Number of Operations to Make Array Continuous

Problem Information

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

Problem Description

You are given an integer array

nums

. In one operation, you can replace

any

element in

nums

with

any

integer.

nums

is considered

continuous

if both of the following conditions are fulfilled:

All elements in

nums

are

unique

.

The difference between the

maximum

element and the

minimum

element in

nums

equals

nums.length - 1

.

For example,

nums = [4, 2, 5, 3]

is

continuous

, but

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

is

not continuous

.

Return

the

minimum

number of operations to make

nums

continuous

.

Example 1:

Input:

nums = [4,2,5,3]

Output:

0

Explanation:

nums is already continuous.

Example 2:

Input:

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

Output:

1

Explanation:

One possible solution is to change the last element to 4. The resulting array is [1,2,3,5,4], which is continuous.

Example 3:

Input:

nums = [1,10,100,1000]

Output:

3

Explanation:

One possible solution is to: - Change the second element to 2. - Change the third element to 3. - Change the fourth element to 4. The resulting array is [1,2,3,4], which is continuous.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (min-operations nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Operations to Make Array Continuous
* Difficulty: Hard
* Tags: array, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Operations to Make Array Continuous
* Difficulty: Hard
* Tags: array, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int minOperations(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Operations to Make Array Continuous
Difficulty: Hard
Tags: array, hash, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Number of Operations to Make Array Continuous
* Difficulty: Hard
* Tags: array, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Operations to Make Array Continuous
* Difficulty: Hard
* Tags: array, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function minOperations(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Operations to Make Array Continuous
* Difficulty: Hard
* Tags: array, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int MinOperations(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Operations to Make Array Continuous
* Difficulty: Hard
* Tags: array, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int minOperations(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Minimum Number of Operations to Make Array Continuous
// Difficulty: Hard
// Tags: array, hash, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map
```

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

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Number of Operations to Make Array Continuous
// Difficulty: Hard
// Tags: array, hash, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (min-operations nums)
(-> (listof exact-integer?) exact-integer?)
)
```