Problem 3362: Zero Array Transformation III

Problem Information

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

Problem Description

You are given an integer array

nums

of length

n

and a 2D array

queries

where

queries[i] = [l

i

, r

i

]

.

Each

queries[i]

represents the following action on

nums

:

Decrement the value at each index in the range

[l

i

, r

i

]

in

nums

by

at most

1.

The amount by which the value is decremented can be chosen

independently

for each index.

A

Zero Array

is an array with all its elements equal to 0.

Return the

maximum

number of elements that can be removed from

queries

, such that

nums

can still be converted to a

zero array

using the

remaining

queries. If it is not possible to convert

nums

to a

zero array

, return -1.

Example 1:

Input:

nums = [2,0,2], queries = [[0,2],[0,2],[1,1]]

Output:

1

Explanation:

After removing

queries[2]

,

nums

can still be converted to a zero array.

Using

queries[0]

, decrement

nums[0]

and

nums[2]

by 1 and

nums[1]

by 0.

Using

queries[1]

, decrement

nums[0]

and

nums[2]

by 1 and

nums[1]

by 0.

Example 2:

Input:

nums = [1,1,1,1], queries = [[1,3],[0,2],[1,3],[1,2]]

Output:

2

Explanation:

We can remove

queries[2]

and

queries[3]

.

Example 3:

Input:

nums = [1,2,3,4], queries = [[0,3]]

Output:

-1

Explanation:

nums

cannot be converted to a zero array even after using all the queries.

Constraints:

1 <= nums.length <= 10

5

0 <= nums[i] <= 10

5

1 <= queries.length <= 10

5

queries[i].length == 2

0 <= l

i

<= r

i

< nums.length

Code Snippets

C++:

```
class Solution {
public:
int maxRemoval(vector<int>& nums, vector<vector<int>>& queries) {

}
};
```

Java:

```
class Solution {
public int maxRemoval(int[] nums, int[][] queries) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} nums
* @param {number[][]} queries
* @return {number}
*/
```

```
var maxRemoval = function(nums, queries) {

};
```

TypeScript:

```
function maxRemoval(nums: number[], queries: number[][]): number {

};
```

C#:

```
public class Solution {
public int MaxRemoval(int[] nums, int[][] queries) {

}
}
```

C:

```
int maxRemoval(int* nums, int numsSize, int** queries, int queriesSize, int*
queriesColSize) {

}
```

Go:

```
func maxRemoval(nums []int, queries [][]int) int {

}
```

Kotlin:

```
class Solution {
fun maxRemoval(nums: IntArray, queries: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func maxRemoval(_ nums: [Int], _ queries: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_removal(nums: Vec<i32>, queries: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer[][]} queries
# @return {Integer}
def max_removal(nums, queries)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer[][] $queries
* @return Integer
*/
function maxRemoval($nums, $queries) {

}
}
```

Dart:

```
class Solution {
int maxRemoval(List<int> nums, List<List<int>> queries) {

}
```

```
}
```

Scala:

```
object Solution {
def maxRemoval(nums: Array[Int], queries: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_removal(nums :: [integer], queries :: [[integer]]) :: integer
def max_removal(nums, queries) do

end
end
```

Erlang:

```
-spec max_removal(Nums :: [integer()], Queries :: [[integer()]]) ->
integer().
max_removal(Nums, Queries) ->
.
```

Racket:

```
(define/contract (max-removal nums queries)
(-> (listof exact-integer?) (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Zero Array Transformation III
* Difficulty: Medium
* Tags: array, greedy, sort, queue, heap
```

```
*
* 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:
int maxRemoval(vector<int>& nums, vector<vector<int>>& queries) {

}
};
```

Java Solution:

```
/**
* Problem: Zero Array Transformation III
* Difficulty: Medium
* Tags: array, greedy, sort, queue, heap
*
* 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 int maxRemoval(int[] nums, int[][] queries) {

}
}
```

Python3 Solution:

```
"""
Problem: Zero Array Transformation III
Difficulty: Medium
Tags: array, greedy, sort, queue, heap

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Zero Array Transformation III
* Difficulty: Medium
* Tags: array, greedy, sort, queue, heap
*
* 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[]} nums
* @param {number[][]} queries
* @return {number}
*/
var maxRemoval = function(nums, queries) {

};
```

TypeScript Solution:

```
/**
* Problem: Zero Array Transformation III
* Difficulty: Medium
* Tags: array, greedy, sort, queue, heap
*
* 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 maxRemoval(nums: number[], queries: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Zero Array Transformation III
* Difficulty: Medium
* Tags: array, greedy, sort, queue, heap
*
* 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 int MaxRemoval(int[] nums, int[][] queries) {

}
}
```

C Solution:

```
/*
* Problem: Zero Array Transformation III
* Difficulty: Medium
* Tags: array, greedy, sort, queue, heap
*
* 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
```

```
*/

int maxRemoval(int* nums, int numsSize, int** queries, int queriesSize, int*
queriesColSize) {

}
```

Go Solution:

```
// Problem: Zero Array Transformation III
// Difficulty: Medium
// Tags: array, greedy, sort, queue, heap
//
// 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 maxRemoval(nums []int, queries [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxRemoval(nums: IntArray, queries: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxRemoval(_ nums: [Int], _ queries: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Zero Array Transformation III
// Difficulty: Medium
```

```
// Tags: array, greedy, sort, queue, heap
//
// 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 max_removal(nums: Vec<i32>, queries: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer[][]} queries
# @return {Integer}
def max_removal(nums, queries)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer[][] $queries
* @return Integer
*/
function maxRemoval($nums, $queries) {

}
}
```

Dart Solution:

```
class Solution {
int maxRemoval(List<int> nums, List<List<int>> queries) {

}
```

```
}
```

Scala Solution:

```
object Solution {
def maxRemoval(nums: Array[Int], queries: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_removal(nums :: [integer], queries :: [[integer]]) :: integer
def max_removal(nums, queries) do

end
end
```

Erlang Solution:

```
-spec max_removal(Nums :: [integer()], Queries :: [[integer()]]) ->
integer().
max_removal(Nums, Queries) ->
.
```

Racket Solution:

```
(define/contract (max-removal nums queries)
(-> (listof exact-integer?) (listof (listof exact-integer?)) exact-integer?)
)
```