Problem 3355: Zero Array Transformation I

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

]

.

For each

queries[i]

:

Select a

subset

of indices within the range

[l

i

, r

i

]

in

nums

.

Decrement the values at the selected indices by 1.

A

Zero Array

is an array where all elements are equal to 0.

Return

true

if it is

possible

to transform

nums

into a

Zero Array

after processing all the queries sequentially, otherwise return

false

.

Example 1:

Input:

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

Output:

true

Explanation:

For i = 0:

Select the subset of indices as

[0, 2]

and decrement the values at these indices by 1.

The array will become

[0, 0, 0]

, which is a Zero Array.

Example 2:

Input:

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

Output:

false

Explanation:

For i = 0:

Select the subset of indices as

[1, 2, 3]

and decrement the values at these indices by 1.

The array will become

[4, 2, 1, 0]

.

For i = 1:

Select the subset of indices as

[0, 1, 2]

and decrement the values at these indices by 1.

The array will become

[3, 1, 0, 0]

, which is not a Zero Array.

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:
```

```
bool isZeroArray(vector<int>& nums, vector<vector<int>>& queries) {

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} nums
* @param {number[][]} queries
* @return {boolean}
*/
var isZeroArray = function(nums, queries) {

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

```
func isZeroArray(nums []int, queries [][]int) bool {

}
```

Kotlin:

```
class Solution {
fun isZeroArray(nums: IntArray, queries: Array<IntArray>): Boolean {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
```

```
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (is-zero-array nums queries)
(-> (listof exact-integer?) (listof (listof exact-integer?)) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Zero Array Transformation I
* Difficulty: Medium
* 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 isZeroArray(vector<int>& nums, vector<vector<int>>& queries) {

}
};
```

Java Solution:

```
/**
* Problem: Zero Array Transformation I
* Difficulty: Medium
* 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 isZeroArray(int[] nums, int[][] queries) {

}
}
```

Python3 Solution:

```
"""
Problem: Zero Array Transformation I
Difficulty: Medium
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 isZeroArray(self, nums: List[int], queries: List[List[int]]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Zero Array Transformation I
* Difficulty: Medium
* 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[]} nums
* @param {number[][]} queries
* @return {boolean}
*/
var isZeroArray = function(nums, queries) {

};
```

TypeScript Solution:

```
/**
* Problem: Zero Array Transformation I
* Difficulty: Medium
* 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 isZeroArray(nums: number[], queries: number[][]): boolean {
```

```
};
```

C# Solution:

```
/*
* Problem: Zero Array Transformation I
* Difficulty: Medium
* 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 IsZeroArray(int[] nums, int[][] queries) {

}
}
```

C Solution:

```
/*
* Problem: Zero Array Transformation I
* Difficulty: Medium
* 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 isZeroArray(int* nums, int numsSize, int** queries, int queriesSize,
int* queriesColSize) {

}
```

Go Solution:

```
// Problem: Zero Array Transformation I
// Difficulty: Medium
// 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 isZeroArray(nums []int, queries [][]int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isZeroArray(nums: IntArray, queries: Array<IntArray>): Boolean {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Zero Array Transformation I
// Difficulty: Medium
// 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 is_zero_array(nums: Vec<i32>, queries: Vec<Vec<i32>>) -> bool {

}
```

```
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (is-zero-array nums queries)
(-> (listof exact-integer?) (listof (listof exact-integer?)) boolean?)
)
```