Problem 3495: Minimum Operations to Make Array Elements Zero

Problem Information

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

Problem Description

You are given a 2D array

queries

, where

queries[i]

is of the form

[l, r]

. Each

queries[i]

defines an array of integers

nums

consisting of elements ranging from

l

to

r

, both

inclusive

.

In one operation, you can:

Select two integers

a

and

b

from the array.

Replace them with

floor(a / 4)

and

floor(b / 4)

.

Your task is to determine the

minimum

number of operations required to reduce all elements of the array to zero for each query. Return the sum of the results for all queries.

Example 1:

Input:

queries = [[1,2],[2,4]]

Output:

3

Explanation:

For

queries[0]

:

The initial array is

nums = [1, 2]

.

In the first operation, select

nums[0]

and

nums[1]

. The array becomes

[0, 0]

.

The minimum number of operations required is 1.

For

queries[1]

:

The initial array is

nums = [2, 3, 4]

.

In the first operation, select

nums[0]

and

nums[2]

. The array becomes

[0, 3, 1]

.

In the second operation, select

nums[1]

and

nums[2]

. The array becomes

[0, 0, 0]

.

The minimum number of operations required is 2.

The output is

1 + 2 = 3

.

Example 2:

Input:

queries = [[2,6]]

Output:

4

Explanation:

For

queries[0]

:

The initial array is

nums = [2, 3, 4, 5, 6]

.

In the first operation, select

nums[0]

and

nums[3]

. The array becomes

[0, 3, 4, 1, 6]

.

In the second operation, select

nums[2]

and

nums[4]

. The array becomes

[0, 3, 1, 1, 1]

.

In the third operation, select

nums[1]

and

nums[2]

. The array becomes

[0, 0, 0, 1, 1]

.

In the fourth operation, select

nums[3]

and

nums[4]

. The array becomes

[0, 0, 0, 0, 0]

.

The minimum number of operations required is 4.

The output is 4.

Constraints:

1 <= queries.length <= 10

5

queries[i].length == 2

queries[i] == [l, r]

1 <= l < r <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public long minOperations(int[][] queries) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long MinOperations(int[][] queries) {
}
}
```

C:

```
long long minOperations(int** queries, int queriesSize, int* queriesColSize)
{

}
```

Go:

```
func minOperations(queries [][]int) int64 {

}
```

Kotlin:

```
class Solution {
fun minOperations(queries: Array<IntArray>): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn min_operations(queries: Vec<Vec<i32>>) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Minimum Operations to Make Array Elements Zero
* Difficulty: Hard
* Tags: array, math
*
* 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:
long long minOperations(vector<vector<int>>& queries) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Operations to Make Array Elements Zero
* Difficulty: Hard
* Tags: array, math
*
* 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 long minOperations(int[][] queries) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Operations to Make Array Elements Zero
Difficulty: Hard
Tags: array, math

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Operations to Make Array Elements Zero
* Difficulty: Hard
* Tags: array, math
*
* 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[][]} queries
* @return {number}
*/
var minOperations = function(queries) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Operations to Make Array Elements Zero
* Difficulty: Hard
* Tags: array, math
*
* 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 minOperations(queries: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Operations to Make Array Elements Zero
* Difficulty: Hard
* Tags: array, math
*
* 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 long MinOperations(int[][] queries) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Operations to Make Array Elements Zero
* Difficulty: Hard
* Tags: array, math
*
* 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
*/

long long minOperations(int** queries, int queriesSize, int* queriesColSize)
{

}
```

Go Solution:

```
// Problem: Minimum Operations to Make Array Elements Zero
// Difficulty: Hard
// Tags: array, math
//
// 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 minOperations(queries [][]int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun minOperations(queries: Array<IntArray>): Long {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Operations to Make Array Elements Zero
// Difficulty: Hard
// Tags: array, math
//
// 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 min_operations(queries: Vec<Vec<i32>>) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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