Problem 1351: Count Negative Numbers in a Sorted Matrix

Problem Information

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

Problem Description

Given a

m x n

matrix

grid

which is sorted in non-increasing order both row-wise and column-wise, return

the number of

negative

numbers in

grid

.

Example 1:

Input:

grid = [[4,3,2,-1],[3,2,1,-1],[1,1,-1,-2],[-1,-1,-2,-3]]

Output:

8

Explanation:

There are 8 negatives number in the matrix.

Example 2:

Input:

grid = [[3,2],[1,0]]

Output:

0

Constraints:

m == grid.length

n == grid[i].length

1 <= m, n <= 100

-100 <= grid[i][j] <= 100

Follow up:

Could you find an

O(n + m)

solution?

Code Snippets

C++:

```
class Solution {
public:
int countNegatives(vector<vector<int>>& grid) {

}
};
```

Java:

```
class Solution {
public int countNegatives(int[][] grid) {

}
}
```

Python3:

```
class Solution:
def countNegatives(self, grid: List[List[int]]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function countNegatives(grid: number[][]): number {

};
```

C#:

```
public class Solution {
public int CountNegatives(int[][] grid) {

}
}
```

C:

```
int countNegatives(int** grid, int gridSize, int* gridColSize) {

}
```

Go:

```
func countNegatives(grid [][]int) int {

}
```

Kotlin:

```
class Solution {
fun countNegatives(grid: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func countNegatives(_ grid: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_negatives(grid: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} grid
# @return {Integer}
def count_negatives(grid)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $grid
* @return Integer
*/
function countNegatives($grid) {

}
}
```

Dart:

```
class Solution {
int countNegatives(List<List<int>> grid) {

}
}
```

Scala:

```
object Solution {
def countNegatives(grid: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_negatives(grid :: [[integer]]) :: integer
def count_negatives(grid) do

end
end
```

Erlang:

```
-spec count_negatives(Grid :: [[integer()]]) -> integer().
count_negatives(Grid) ->
.
```

Racket:

```
(define/contract (count-negatives grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Negative Numbers in a Sorted Matrix
* Difficulty: Easy
* Tags: array, sort, search
*
* 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 countNegatives(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Count Negative Numbers in a Sorted Matrix
* Difficulty: Easy
* Tags: array, sort, search
*
* 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 countNegatives(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Negative Numbers in a Sorted Matrix
Difficulty: Easy
Tags: array, sort, search

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

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Count Negative Numbers in a Sorted Matrix
* Difficulty: Easy
* Tags: array, sort, search
*
* 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[][]} grid
* @return {number}
*/
var countNegatives = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Negative Numbers in a Sorted Matrix
* Difficulty: Easy
* Tags: array, sort, search
*
* 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 countNegatives(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Count Negative Numbers in a Sorted Matrix
* Difficulty: Easy
* Tags: array, sort, search
*
* 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 CountNegatives(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Count Negative Numbers in a Sorted Matrix
* Difficulty: Easy
* Tags: array, sort, search
*
* 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 countNegatives(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Count Negative Numbers in a Sorted Matrix
// Difficulty: Easy
// Tags: array, sort, search
//
// 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 countNegatives(grid [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countNegatives(grid: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func countNegatives(_ grid: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Negative Numbers in a Sorted Matrix
// Difficulty: Easy
// Tags: array, sort, search
//
// 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 count_negatives(grid: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} grid
# @return {Integer}
def count_negatives(grid)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $grid
* @return Integer
*/
function countNegatives($grid) {

}
}
```

Dart Solution:

```
class Solution {
int countNegatives(List<List<int>> grid) {

}
}
```

Scala Solution:

```
object Solution {
def countNegatives(grid: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_negatives(grid :: [[integer]]) :: integer
def count_negatives(grid) do

end
end
```

Erlang Solution:

```
-spec count_negatives(Grid :: [[integer()]]) -> integer().
count_negatives(Grid) ->
.
```

Racket Solution:

```
(define/contract (count-negatives grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```