Problem 3128: Right Triangles

Problem Information

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

Problem Description

You are given a 2D boolean matrix

grid

.

A collection of 3 elements of

grid

is a

right triangle

if one of its elements is in the

same row

with another element and in the

same column

with the third element. The 3 elements may

not

be next to each other.

Return an integer that is the number of

right triangles

that can be made with 3 elements of

grid

such that

all

of them have a value of 1.

Example 1:

0

1

0

0

1

1

0

1

0

0

1

0

0

1

1

0

1

0

0

1

0

0

1

1

0

1

0

Input:

grid = [[0,1,0],[0,1,1],[0,1,0]]

Output:

2

Explanation:

There are two right triangles with elements of the value 1. Notice that the blue ones do

not

form a right triangle because the 3 elements are in the same column.

Example 2:

1

0

0

0

0

1

0

1

1

0

0

0

Input:

grid = [[1,0,0,0],[0,1,0,1],[1,0,0,0]]

Output:

0

Explanation:

There are no right triangles with elements of the value 1. Notice that the blue ones do

not

form a right triangle.

Example 3:

1

0

1

1

0

0

1

0

0

1

0

1

1

0

0

1

0

0

Input:

grid = [[1,0,1],[1,0,0],[1,0,0]]

Output:

2

Explanation:

There are two right triangles with elements of the value 1.

Constraints:

1 <= grid.length <= 1000

1 <= grid[i].length <= 1000

0 <= grid[i][j] <= 1

Code Snippets

C++:

```
class Solution {
public:
```

```
long long numberOfRightTriangles(vector<vector<int>>& grid) {

}
};
```

Java:

```
class Solution {
public long numberOfRightTriangles(int[][] grid) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long NumberOfRightTriangles(int[][] grid) {

}
}
```

C:

```
long long numberOfRightTriangles(int** grid, int gridSize, int* gridColSize)
{

}
```

Go:

```
func numberOfRightTriangles(grid [][]int) int64 {

}
```

Kotlin:

```
class Solution {
fun numberOfRightTriangles(grid: Array<IntArray>): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn number_of_right_triangles(grid: Vec<Vec<i32>>) -> i64 {

}
```

```
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (number-of-right-triangles grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Right Triangles
* Difficulty: Medium
* Tags: array, math, hash
*
* 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:
long long numberOfRightTriangles(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Right Triangles
* Difficulty: Medium
* Tags: array, math, hash
*
* 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 long numberOfRightTriangles(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Right Triangles
Difficulty: Medium
Tags: array, math, hash

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Right Triangles
* Difficulty: Medium
* Tags: array, math, hash
*
* 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[][]} grid
* @return {number}
*/
var numberOfRightTriangles = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Right Triangles
* Difficulty: Medium
* Tags: array, math, hash
*
* 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 numberOfRightTriangles(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Right Triangles
* Difficulty: Medium
* Tags: array, math, hash
*
```

```
* 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 long NumberOfRightTriangles(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Right Triangles
* Difficulty: Medium
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

long long numberOfRightTriangles(int** grid, int gridSize, int* gridColSize)
{

}
```

Go Solution:

```
// Problem: Right Triangles
// Difficulty: Medium
// Tags: array, math, hash
//
// 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 numberOfRightTriangles(grid [][]int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfRightTriangles(grid: Array<IntArray>): Long {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Right Triangles
// Difficulty: Medium
// Tags: array, math, hash
//
// 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 number_of_right_triangles(grid: Vec<Vec<i32>>) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (number-of-right-triangles grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```