Problem 750: Number Of Corner Rectangles

Problem Information

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

Problem Description

Given an

m x n

integer matrix

grid

where each entry is only

0

or

1

, return

the number of

corner rectangles

.

A

corner rectangle

is four distinct

1

's on the grid that forms an axis-aligned rectangle. Note that only the corners need to have the value

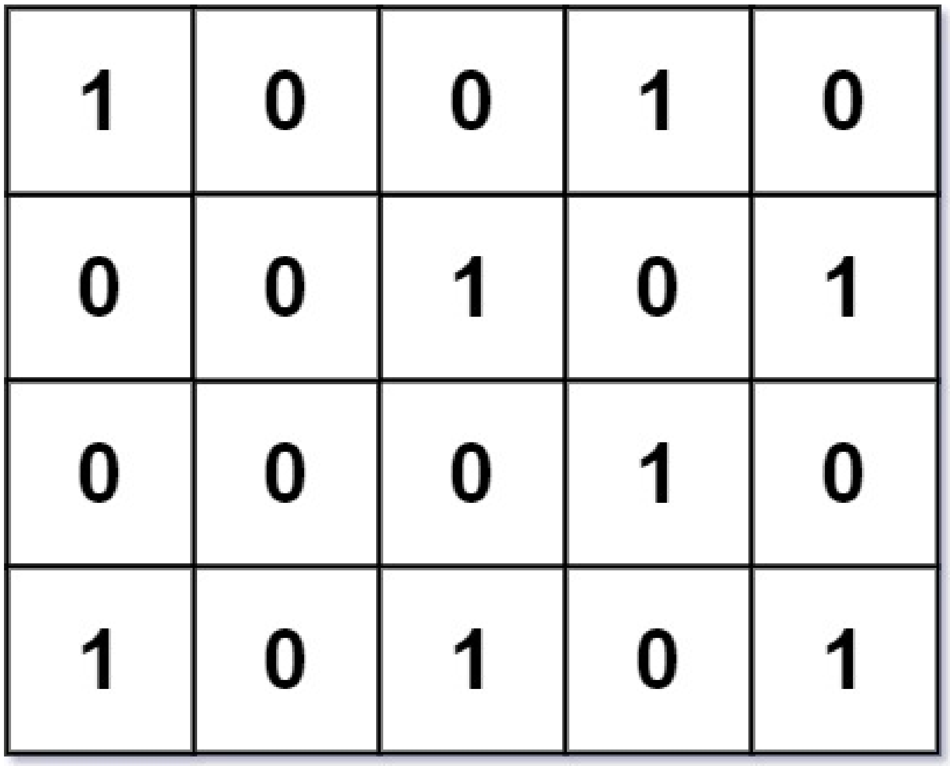
1

. Also, all four

1

's used must be distinct.

Example 1:



1	0	0	1	0
0	0	1	0	1
0	0	0	1	0
1	0	1	0	1

Input:

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

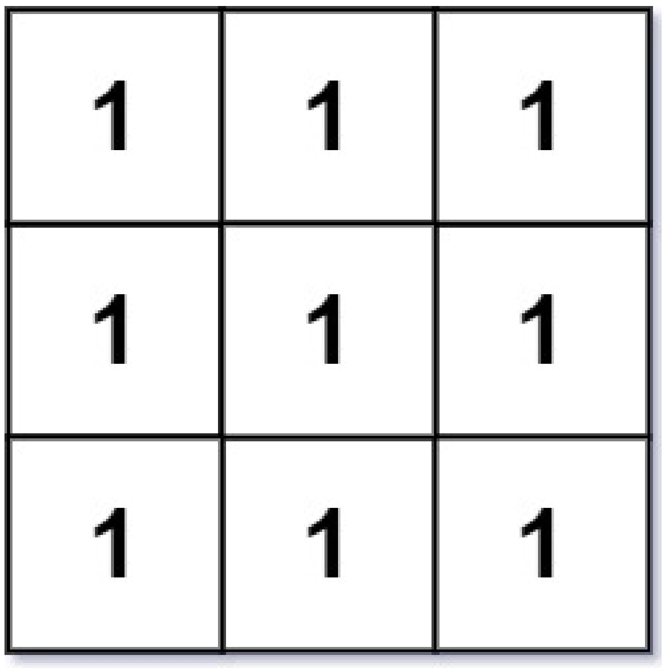
Output:

1

Explanation:

There is only one corner rectangle, with corners grid[1][2], grid[1][4], grid[3][2], grid[3][4].

Example 2:



Input:

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

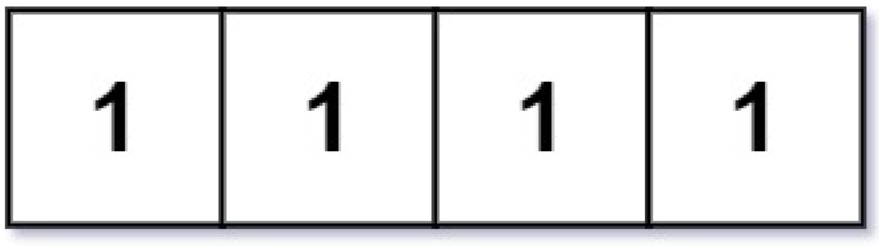
Output:

9

Explanation:

There are four 2x2 rectangles, four 2x3 and 3x2 rectangles, and one 3x3 rectangle.

Example 3:



Input:

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

Output:

0

Explanation:

Rectangles must have four distinct corners.

Constraints:

m == grid.length

n == grid[i].length

1 <= m, n <= 200

grid[i][j]

is either

0

or

1

.

The number of

1

's in the grid is in the range

[1, 6000]

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

```
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Number Of Corner Rectangles
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Number Of Corner Rectangles
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int countCornerRectangles(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Number Of Corner Rectangles
Difficulty: Medium
Tags: array, dp, math
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def countCornerRectangles(self, grid: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number Of Corner Rectangles
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Number Of Corner Rectangles
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function countCornerRectangles(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Number Of Corner Rectangles
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int CountCornerRectangles(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Number Of Corner Rectangles
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

int countCornerRectangles(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Number Of Corner Rectangles
// Difficulty: Medium
// Tags: array, dp, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func countCornerRectangles(grid [][]int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number Of Corner Rectangles
// Difficulty: Medium
// Tags: array, dp, math
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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