Problem 3665: Twisted Mirror Path Count

Problem Information

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

Problem Description

Given an

m x n

binary grid

grid

where:

grid[i][j] == 0

represents an empty cell, and

grid[i][j] == 1

represents a mirror.

A robot starts at the top-left corner of the grid

(0, 0)

and wants to reach the bottom-right corner

(m - 1, n - 1)

. It can move only

right

or

down

. If the robot attempts to move into a mirror cell, it is

reflected

before entering that cell:

If it tries to move

right

into a mirror, it is turned

down

and moved into the cell directly below the mirror.

If it tries to move

down

into a mirror, it is turned

right

and moved into the cell directly to the right of the mirror.

If this reflection would cause the robot to move outside the

grid

boundaries, the path is considered invalid and should not be counted.

Return the number of unique valid paths from

(0, 0)

to

(m - 1, n - 1)

.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

Note

: If a reflection moves the robot into a mirror cell, the robot is immediately reflected again based on the direction it used to enter that mirror: if it entered while moving right, it will be turned down; if it entered while moving down, it will be turned right. This process will continue until either the last cell is reached, the robot moves out of bounds or the robot moves to a non-mirror cell.

Example 1:

Input:

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

Output:

5

Explanation:

Number

Full Path

1

(0, 0) → (0, 1) [M] → (1, 1) → (1, 2) [M] → (2, 2)

2

(0, 0) → (0, 1) [M] → (1, 1) → (2, 1) → (2, 2)

3

(0, 0) → (1, 0) → (1, 1) → (1, 2) [M] → (2, 2)

4

(0, 0) → (1, 0) → (1, 1) → (2, 1) → (2, 2)

5

(0, 0) → (1, 0) → (2, 0) [M] → (2, 1) → (2, 2)

[M]

indicates the robot attempted to enter a mirror cell and instead reflected.

Example 2:

Input:

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

Output:

2

Explanation:

Number

Full Path

1

(0, 0) → (0, 1) → (1, 1)

2

(0, 0) → (1, 0) → (1, 1)

Example 3:

Input:

grid =

[[0,1,1],[1,1,0]]

Output:

1

Explanation:

Number

Full Path

1

(0, 0) → (0, 1) [M] → (1, 1) [M] → (1, 2)

(0, 0) → (1, 0) [M] → (1, 1) [M] → (2, 1)

goes out of bounds, so it is invalid.

Constraints:

m == grid.length

n == grid[i].length

2 <= m, n <= 500

grid[i][j]

is either

0

or

1

.

grid[0][0] == grid[m - 1][n - 1] == 0

Code Snippets

C++:

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

}
};
```

Java:

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

```
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (unique-paths grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Twisted Mirror Path Count
* Difficulty: Medium
* Tags: array, dp
*
* 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 uniquePaths(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Twisted Mirror Path Count
* Difficulty: Medium
* Tags: array, dp
*
* 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 uniquePaths(int[][] grid) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Twisted Mirror Path Count
Difficulty: Medium
Tags: array, dp

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Twisted Mirror Path Count
* Difficulty: Medium
* Tags: array, dp
*
* 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 uniquePaths = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Twisted Mirror Path Count
* Difficulty: Medium
* Tags: array, dp
*
* 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 uniquePaths(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Twisted Mirror Path Count
* Difficulty: Medium
* Tags: array, dp
*
* 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 UniquePaths(int[][] grid) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Twisted Mirror Path Count
* Difficulty: Medium
* Tags: array, dp
*
* 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 uniquePaths(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Twisted Mirror Path Count
// Difficulty: Medium
// Tags: array, dp
//
// 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 uniquePaths(grid [][]int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Twisted Mirror Path Count
// Difficulty: Medium
// Tags: array, dp
//
// 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 unique_paths(grid: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (unique-paths grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```