Problem 3393: Count Paths With the Given XOR Value

Problem Information

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

Problem Description

You are given a 2D integer array

grid

with size

m x n

. You are also given an integer

k

.

Your task is to calculate the number of paths you can take from the top-left cell

(0, 0)

to the bottom-right cell

(m - 1, n - 1)

satisfying the following

constraints

:

You can either move to the right or down. Formally, from the cell

(i, j)

you may move to the cell

(i, j + 1)

or to the cell

(i + 1, j)

if the target cell

exists

.

The

XOR

of all the numbers on the path must be

equal

to

k

.

Return the total number of such paths.

Since the answer can be very large, return the result

modulo

10

9

+ 7

.

Example 1:

Input:

grid = [[2, 1, 5], [7, 10, 0], [12, 6, 4]], k = 11

Output:

3

Explanation:

The 3 paths are:

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

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

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

Example 2:

Input:

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

Output:

5

Explanation:

The 5 paths are:

(0, 0) → (1, 0) → (2, 0) → (2, 1) → (2, 2) → (2, 3)

(0, 0) → (1, 0) → (1, 1) → (2, 1) → (2, 2) → (2, 3)

(0, 0) → (1, 0) → (1, 1) → (1, 2) → (1, 3) → (2, 3)

(0, 0) → (0, 1) → (1, 1) → (1, 2) → (2, 2) → (2, 3)

(0, 0) → (0, 1) → (0, 2) → (1, 2) → (2, 2) → (2, 3)

Example 3:

Input:

grid = [[1, 1, 1, 2], [3, 0, 3, 2], [3, 0, 2, 2]], k = 10

Output:

0

Constraints:

1 <= m == grid.length <= 300

1 <= n == grid[r].length <= 300

0 <= grid[r][c] < 16

0 <= k < 16

Code Snippets

C++:

```
class Solution {
public:
int countPathsWithXorValue(vector<vector<int>>& grid, int k) {

}
};
```

Java:

```
class Solution {
public int countPathsWithXorValue(int[][] grid, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function countPathsWithXorValue(grid: number[][], k: number): number {

};
```

C#:

```
public class Solution {
public int CountPathsWithXorValue(int[][] grid, int k) {

}
}
```

C:

```
int countPathsWithXorValue(int** grid, int gridSize, int* gridColSize, int k)
{

}
```

Go:

```
func countPathsWithXorValue(grid [][]int, k int) int {

}
```

Kotlin:

```
class Solution {
fun countPathsWithXorValue(grid: Array<IntArray>, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func countPathsWithXorValue(_ grid: [[Int]], _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_paths_with_xor_value(grid: Vec<Vec<i32>>, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} grid
# @param {Integer} k
# @return {Integer}
def count_paths_with_xor_value(grid, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $grid
* @param Integer $k
* @return Integer
*/
function countPathsWithXorValue($grid, $k) {

}
}
```

Dart:

```
class Solution {
int countPathsWithXorValue(List<List<int>> grid, int k) {

}
}
```

Scala:

```
object Solution {
def countPathsWithXorValue(grid: Array[Array[Int]], k: Int): Int = {

}
```

```
}
```

Elixir:

```
defmodule Solution do
@spec count_paths_with_xor_value(grid :: [[integer]], k :: integer) ::
integer
def count_paths_with_xor_value(grid, k) do

end
end
```

Erlang:

```
-spec count_paths_with_xor_value(Grid :: [[integer()]], K :: integer()) ->
integer().
count_paths_with_xor_value(Grid, K) ->
.
```

Racket:

```
(define/contract (count-paths-with-xor-value grid k)
(-> (listof (listof exact-integer?)) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Paths With the Given XOR Value
* 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 countPathsWithXorValue(vector<vector<int>>& grid, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Count Paths With the Given XOR Value
* 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 countPathsWithXorValue(int[][] grid, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Paths With the Given XOR Value
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 countPathsWithXorValue(self, grid: List[List[int]], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Paths With the Given XOR Value
* 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
* @param {number} k
* @return {number}
*/
var countPathsWithXorValue = function(grid, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Paths With the Given XOR Value
* 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 countPathsWithXorValue(grid: number[][], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Count Paths With the Given XOR Value
* 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 CountPathsWithXorValue(int[][] grid, int k) {

}
}
```

C Solution:

```
/*
* Problem: Count Paths With the Given XOR Value
* 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 countPathsWithXorValue(int** grid, int gridSize, int* gridColSize, int k)
{

}
```

Go Solution:

```
// Problem: Count Paths With the Given XOR Value
// 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 countPathsWithXorValue(grid [][]int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countPathsWithXorValue(grid: Array<IntArray>, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func countPathsWithXorValue(_ grid: [[Int]], _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Paths With the Given XOR Value
// 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 count_paths_with_xor_value(grid: Vec<Vec<i32>>, k: i32) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {Integer[][]} grid
# @param {Integer} k
# @return {Integer}
def count_paths_with_xor_value(grid, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $grid
* @param Integer $k
* @return Integer
*/
function countPathsWithXorValue($grid, $k) {

}
}
```

Dart Solution:

```
class Solution {
int countPathsWithXorValue(List<List<int>> grid, int k) {

}
}
```

Scala Solution:

```
object Solution {
def countPathsWithXorValue(grid: Array[Array[Int]], k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_paths_with_xor_value(grid :: [[integer]], k :: integer) ::
integer
def count_paths_with_xor_value(grid, k) do

end
end
```

Erlang Solution:

```
-spec count_paths_with_xor_value(Grid :: [[integer()]], K :: integer()) ->
integer().
count_paths_with_xor_value(Grid, K) ->
.
```

Racket Solution:

```
(define/contract (count-paths-with-xor-value grid k)
(-> (listof (listof exact-integer?)) exact-integer? exact-integer?)
)
```