Problem 2245: Maximum Trailing Zeros in a Cornered Path

Problem Information

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

Problem Description

You are given a 2D integer array

grid

of size

m x n

, where each cell contains a positive integer.

A

cornered path

is defined as a set of adjacent cells with

at most

one turn. More specifically, the path should exclusively move either

horizontally

or

vertically

up to the turn (if there is one), without returning to a previously visited cell. After the turn, the path will then move exclusively in the

alternate

direction: move vertically if it moved horizontally, and vice versa, also without returning to a previously visited cell.

The

product

of a path is defined as the product of all the values in the path.

Return

the

maximum

number of

trailing zeros

in the product of a cornered path found in

grid

.

Note:

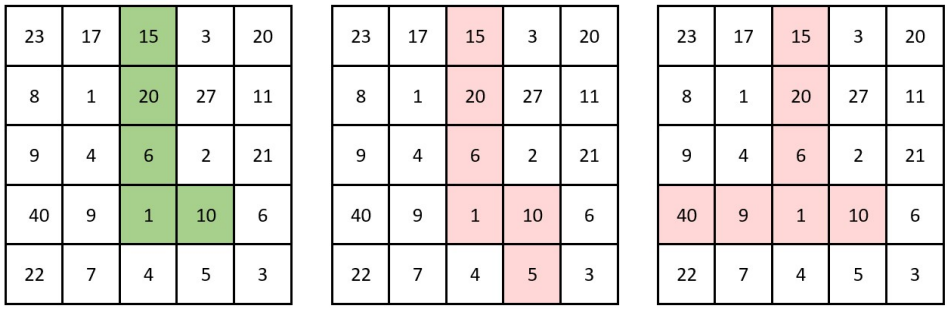
Horizontal

movement means moving in either the left or right direction.

Vertical

movement means moving in either the up or down direction.

Example 1:



Input:

grid = [[23,17,15,3,20],[8,1,20,27,11],[9,4,6,2,21],[40,9,1,10,6],[22,7,4,5,3]]

Output:

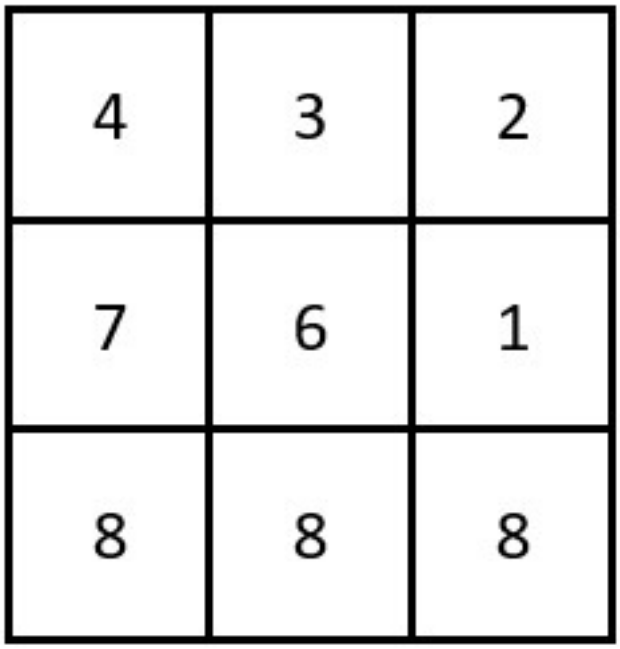
3

Explanation:

The grid on the left shows a valid cornered path. It has a product of 15 * 20 * 6 * 1 * 10 = 18000 which has 3 trailing zeros. It can be shown that this is the maximum trailing zeros in the product of a cornered path.

The grid in the middle is not a cornered path as it has more than one turn. The grid on the right is not a cornered path as it requires a return to a previously visited cell.

Example 2:



Input:

grid = [[4,3,2],[7,6,1],[8,8,8]]

Output:

0

Explanation:

The grid is shown in the figure above. There are no cornered paths in the grid that result in a product with a trailing zero.

Constraints:

m == grid.length

n == grid[i].length

1 <= m, n <= 10

5

1 <= m * n <= 10

5

1 <= grid[i][j] <= 1000

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[][]} grid
```

```
 * @return {number}
 */
var maxTrailingZeros = function(grid) {

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (max-trailing-zeros grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Trailing Zeros in a Cornered Path
* Difficulty: Medium
* Tags: array
*
* 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 maxTrailingZeros(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Trailing Zeros in a Cornered Path
* Difficulty: Medium
* Tags: array
*
* 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 maxTrailingZeros(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Trailing Zeros in a Cornered Path
Difficulty: Medium
Tags: array

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

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Trailing Zeros in a Cornered Path
* Difficulty: Medium
* Tags: array
*
* 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 maxTrailingZeros = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Trailing Zeros in a Cornered Path
* Difficulty: Medium
* Tags: array
*
* 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 maxTrailingZeros(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Trailing Zeros in a Cornered Path
* Difficulty: Medium
* Tags: array
*
* 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 MaxTrailingZeros(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Trailing Zeros in a Cornered Path
* Difficulty: Medium
* Tags: array
*
* 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 maxTrailingZeros(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Maximum Trailing Zeros in a Cornered Path
// Difficulty: Medium
// Tags: array
//
// 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 maxTrailingZeros(grid [][]int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Trailing Zeros in a Cornered Path
// Difficulty: Medium
// Tags: array
//
// 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 max_trailing_zeros(grid: Vec<Vec<i32>>) -> i32 {

}
```

```
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (max-trailing-zeros grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```