Problem 3459: Length of Longest V-Shaped Diagonal Segment

Problem Information

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

Problem Description

You are given a 2D integer matrix

grid

of size

n x m

, where each element is either

0

,

1

, or

2

.

A

V-shaped diagonal segment

is defined as:

The segment starts with

1

.

The subsequent elements follow this infinite sequence:

2, 0, 2, 0, ...

.

The segment:

Starts

along

a diagonal direction (top-left to bottom-right, bottom-right to top-left, top-right to bottom-left, or bottom-left to top-right).

Continues the

sequence

in the same diagonal direction.

Makes

at most one clockwise 90-degree

turn

to another diagonal direction while

maintaining

the sequence.

Return the

length

of the

longest

V-shaped diagonal segment

. If no valid segment

exists

, return 0.

Example 1:

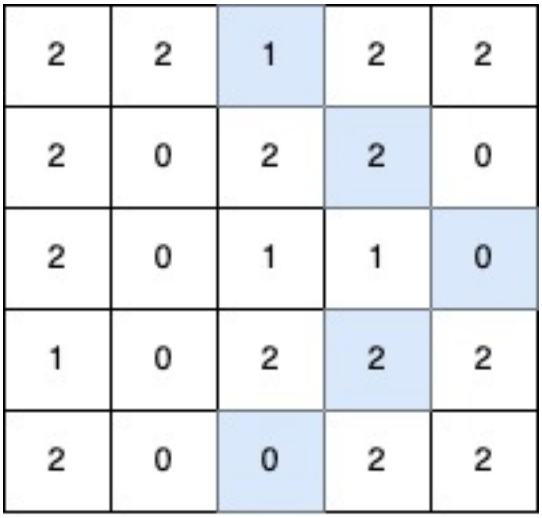
Input:

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

Output:

5

Explanation:



The longest V-shaped diagonal segment has a length of 5 and follows these coordinates:

(0,2) → (1,3) → (2,4)

, takes a

90-degree clockwise turn

at

(2,4)

, and continues as

(3,3) → (4,2)

.

Example 2:

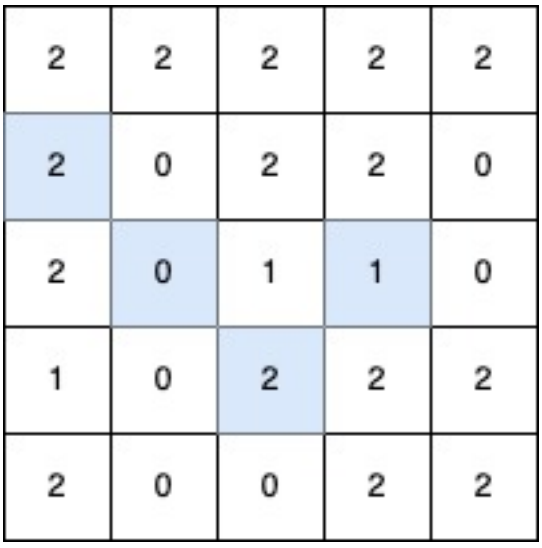
Input:

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

Output:

4

Explanation:



The longest V-shaped diagonal segment has a length of 4 and follows these coordinates:

(2,3) → (3,2)

, takes a

90-degree clockwise turn

at

(3,2)

, and continues as

(2,1) → (1,0)

.

Example 3:

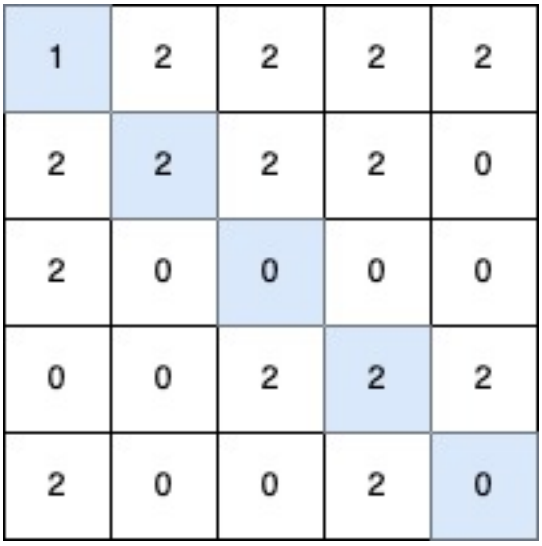
Input:

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

Output:

5

Explanation:



The longest V-shaped diagonal segment has a length of 5 and follows these coordinates:

(0,0) → (1,1) → (2,2) → (3,3) → (4,4)

.

Example 4:

Input:

grid = [[1]]

Output:

1

Explanation:

The longest V-shaped diagonal segment has a length of 1 and follows these coordinates:

(0,0)

.

Constraints:

n == grid.length

m == grid[i].length

1 <= n, m <= 500

grid[i][j]

is either

0

,

1

or

2

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (len-of-v-diagonal grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Length of Longest V-Shaped Diagonal Segment
* Difficulty: Hard
* 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 lenOfVDiagonal(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Length of Longest V-Shaped Diagonal Segment
* Difficulty: Hard
* 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 lenOfVDiagonal(int[][] grid) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Length of Longest V-Shaped Diagonal Segment
Difficulty: Hard
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 lenOfVDiagonal(self, grid: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Length of Longest V-Shaped Diagonal Segment
* Difficulty: Hard
* 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 lenOfVDiagonal = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Length of Longest V-Shaped Diagonal Segment
* Difficulty: Hard
* 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 lenOfVDiagonal(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Length of Longest V-Shaped Diagonal Segment
* Difficulty: Hard
* 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 LenOfVDiagonal(int[][] grid) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Length of Longest V-Shaped Diagonal Segment
* Difficulty: Hard
* 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 lenOfVDiagonal(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Length of Longest V-Shaped Diagonal Segment
// Difficulty: Hard
// 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 lenOfVDiagonal(grid [][]int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Length of Longest V-Shaped Diagonal Segment
// Difficulty: Hard
// 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 len_of_v_diagonal(grid: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (len-of-v-diagonal grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```