Problem 1102: Path With Maximum Minimum Value

Problem Information

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

Problem Description

Given an

m x n

integer matrix

grid

, return

the maximum

score

of a path starting at

(0, 0)

and ending at

(m - 1, n - 1)

moving in the 4 cardinal directions.

The

score

of a path is the minimum value in that path.

For example, the score of the path

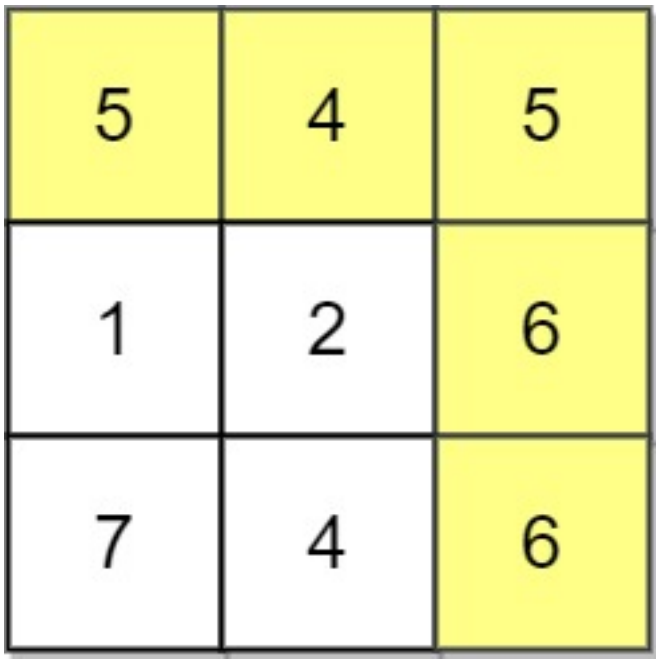
$8 \to 4 \to 5 \to 9$

is

4

.

Example 1:



Input:

grid = [[5,4,5],[1,2,6],[7,4,6]]

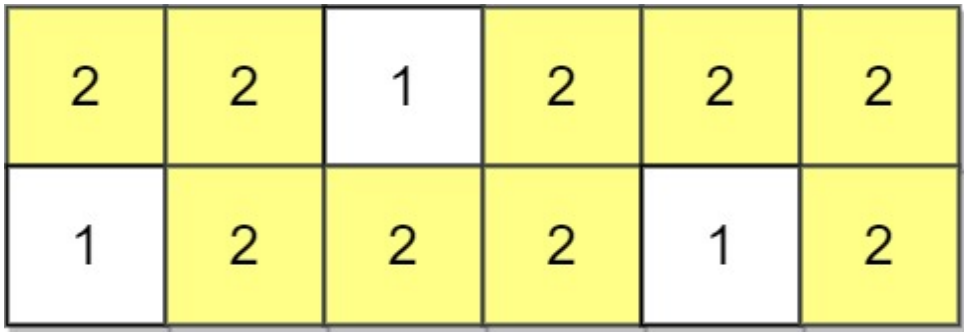
Output:

4

Explanation:

The path with the maximum score is highlighted in yellow.

Example 2:



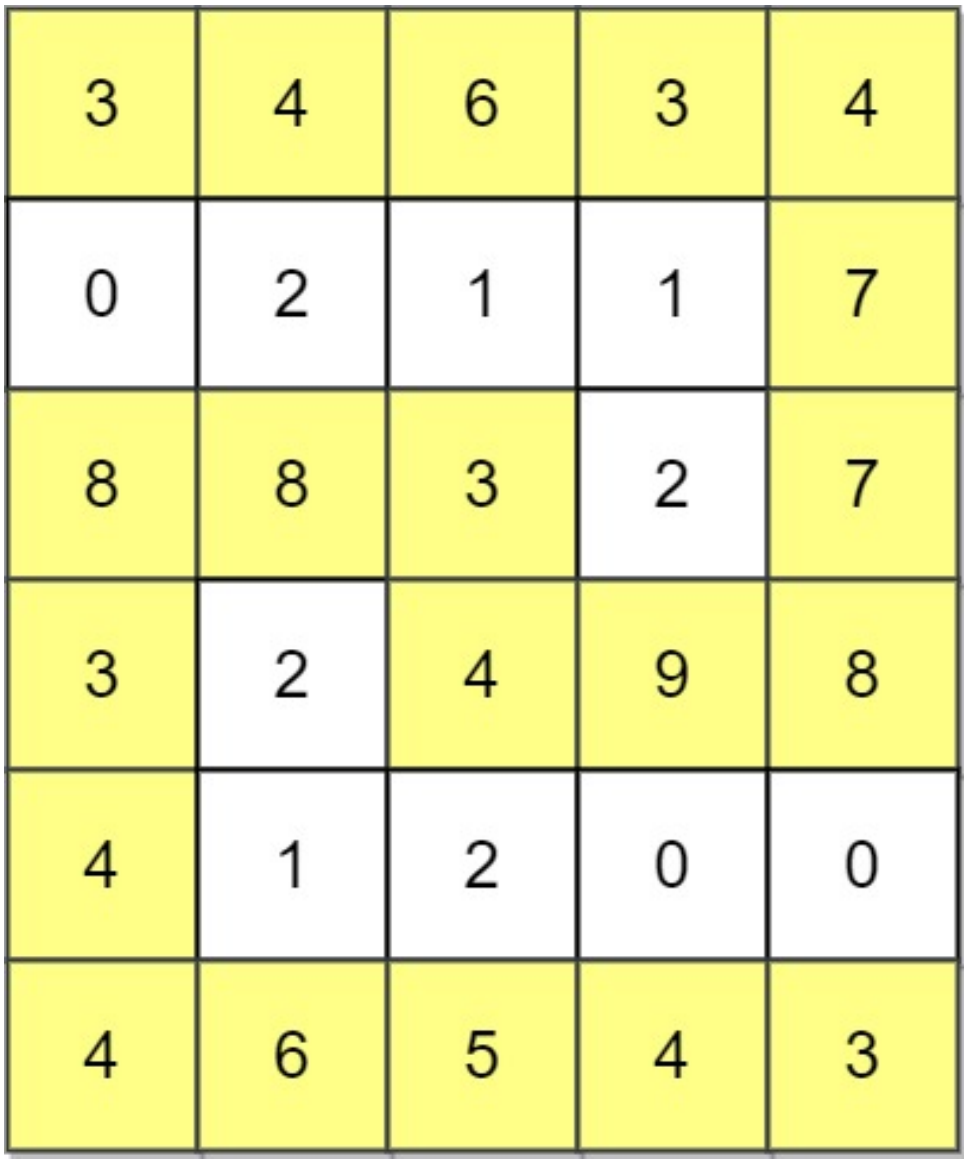
Input:

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

Output:

2

Example 3:



Input:

grid = [[3,4,6,3,4],[0,2,1,1,7],[8,8,3,2,7],[3,2,4,9,8],[4,1,2,0,0],[4,6,5,4,3]]

Output:

3

Constraints:

m == grid.length

n == grid[i].length

1 <= m, n <= 100

0 <= grid[i][j] <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (maximum-minimum-path grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Path With Maximum Minimum Value
* Difficulty: Medium
* Tags: array, graph, search, queue, heap
*
* 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 maximumMinimumPath(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Path With Maximum Minimum Value
* Difficulty: Medium
* Tags: array, graph, search, queue, heap
*
* 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 maximumMinimumPath(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Path With Maximum Minimum Value
Difficulty: Medium
Tags: array, graph, search, queue, heap

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

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Path With Maximum Minimum Value
* Difficulty: Medium
* Tags: array, graph, search, queue, heap
*
* 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 maximumMinimumPath = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Path With Maximum Minimum Value
* Difficulty: Medium
* Tags: array, graph, search, queue, heap
*
* 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 maximumMinimumPath(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Path With Maximum Minimum Value
* Difficulty: Medium
* Tags: array, graph, search, queue, heap
*
* 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 MaximumMinimumPath(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Path With Maximum Minimum Value
* Difficulty: Medium
* Tags: array, graph, search, queue, heap
*
* 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 maximumMinimumPath(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Path With Maximum Minimum Value
// Difficulty: Medium
// Tags: array, graph, search, queue, heap
//
// 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 maximumMinimumPath(grid [][]int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Path With Maximum Minimum Value
// Difficulty: Medium
// Tags: array, graph, search, queue, heap
//
// 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 maximum_minimum_path(grid: Vec<Vec<i32>>) -> i32 {

}
```

```
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (maximum-minimum-path grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```