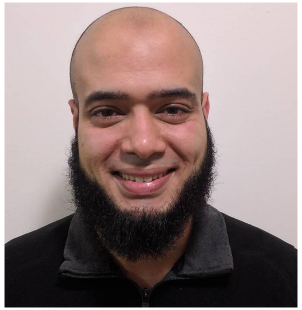
Algorithms

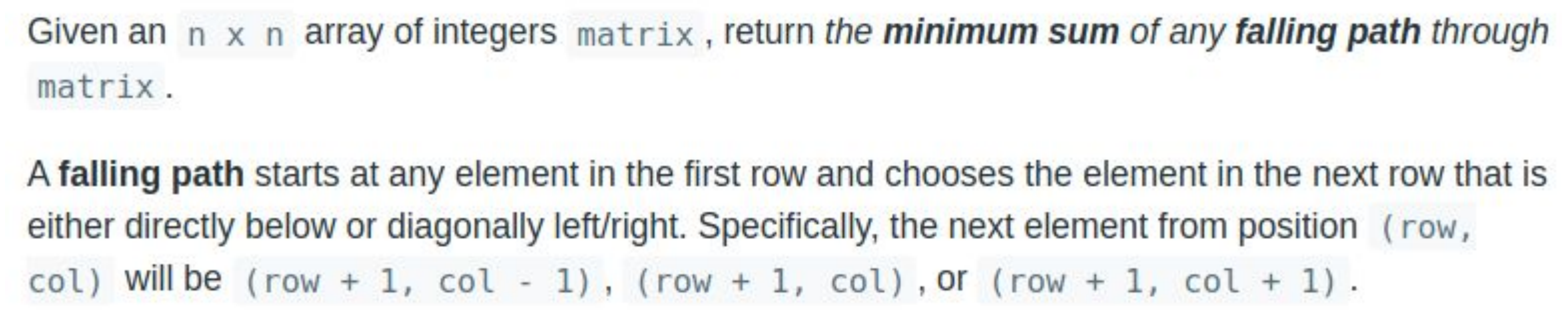
DP Homework 1

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

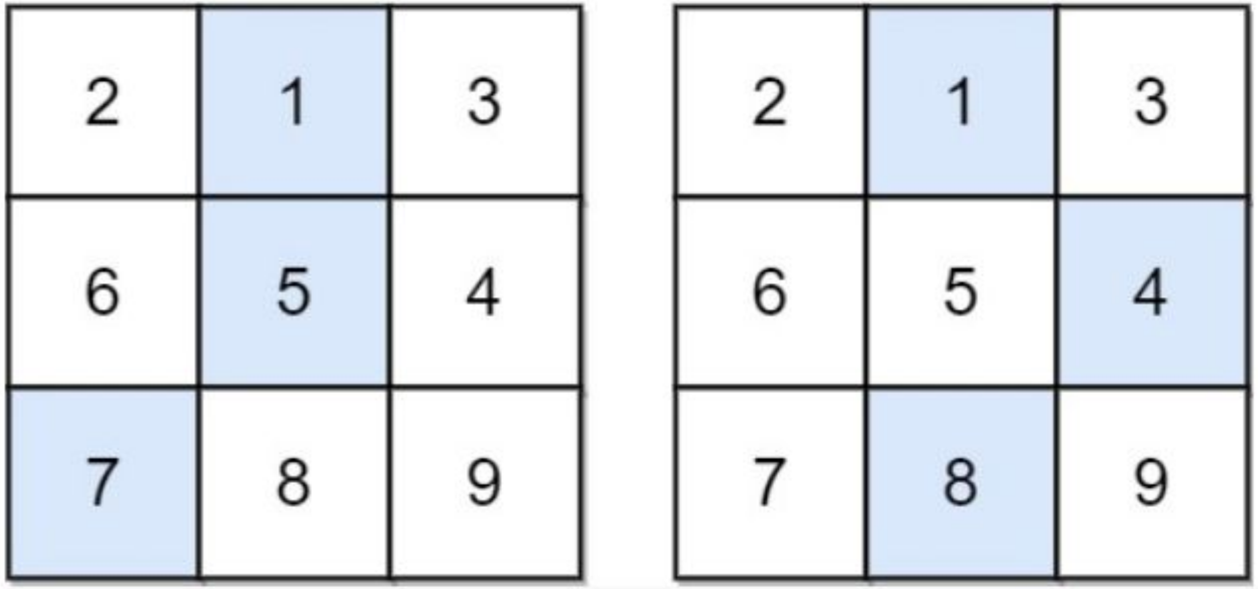


Problem #1: LeetCode 931 - Minimum Falling Path Sum



Given an `n x n` array of integers `matrix`, return *the **minimum sum** of any **falling path** through* `matrix`.

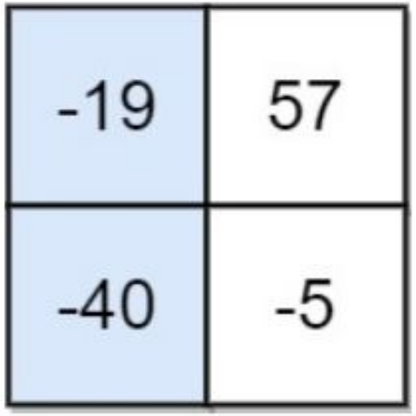
A **falling path** starts at any element in the first row and chooses the element in the next row that is either directly below or diagonally left/right. Specifically, the next element from position `(row, col)` will be `(row + 1, col - 1)`, `(row + 1, col)`, or `(row + 1, col + 1)`.



Input: `matrix = [[2,1,3],[6,5,4],[7,8,9]]`

Output: `13`

Explanation: There are two falling paths with a minimum sum as shown.

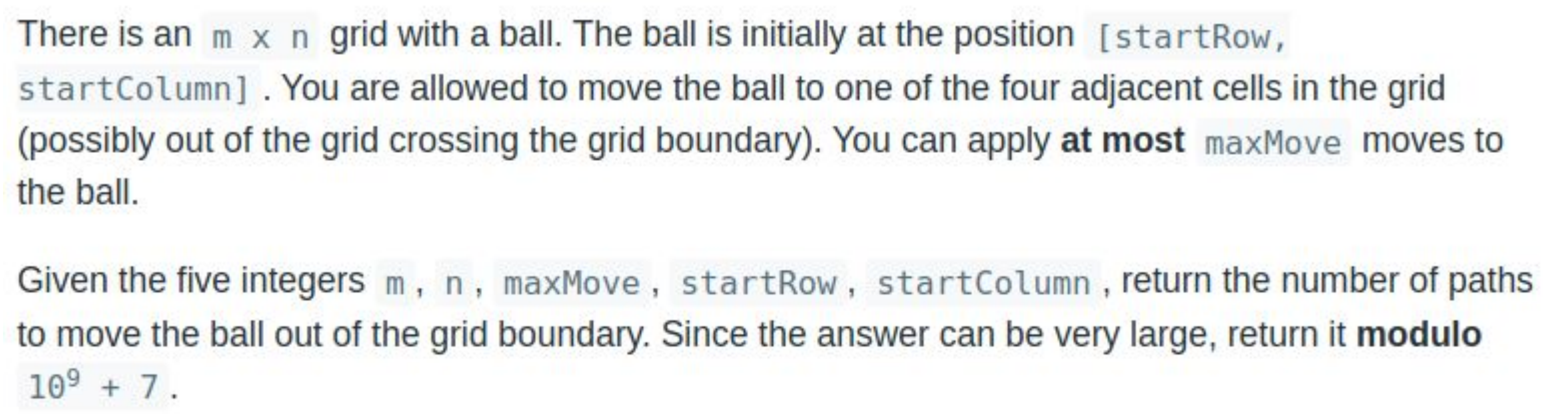


```
Input: matrix = [[-19,57],[-40,-5]]
Output: -59
Explanation: The falling path with a minimum sum is shown.
```

Constraints:

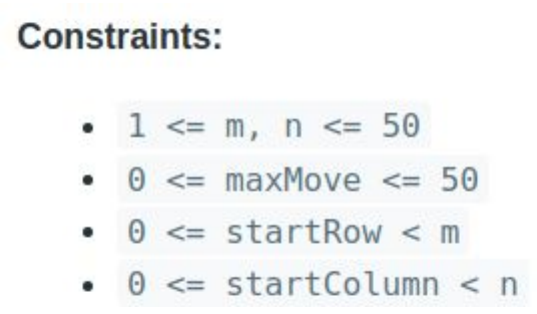
- `n == matrix.length == matrix[i].length`
- `1 <= n <= 100`
- `-100 <= matrix[i][j] <= 100`

Problem #2: LeetCode 576 - Out of Boundary Paths



There is an `m x n` grid with a ball. The ball is initially at the position `[startRow, startColumn]`. You are allowed to move the ball to one of the four adjacent cells in the grid (possibly out of the grid crossing the grid boundary). You can apply **at most** `maxMove` moves to the ball.

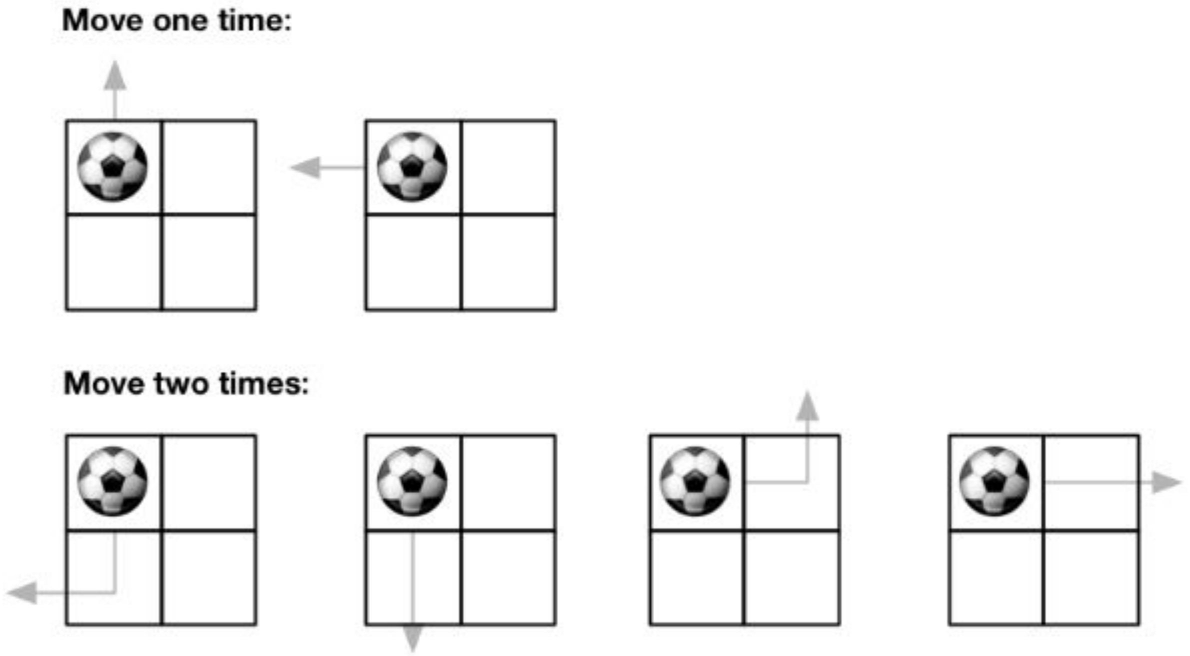
Given the five integers `m`, `n`, `maxMove`, `startRow`, `startColumn`, return the number of paths to move the ball out of the grid boundary. Since the answer can be very large, return it **modulo** $10^9 + 7$.



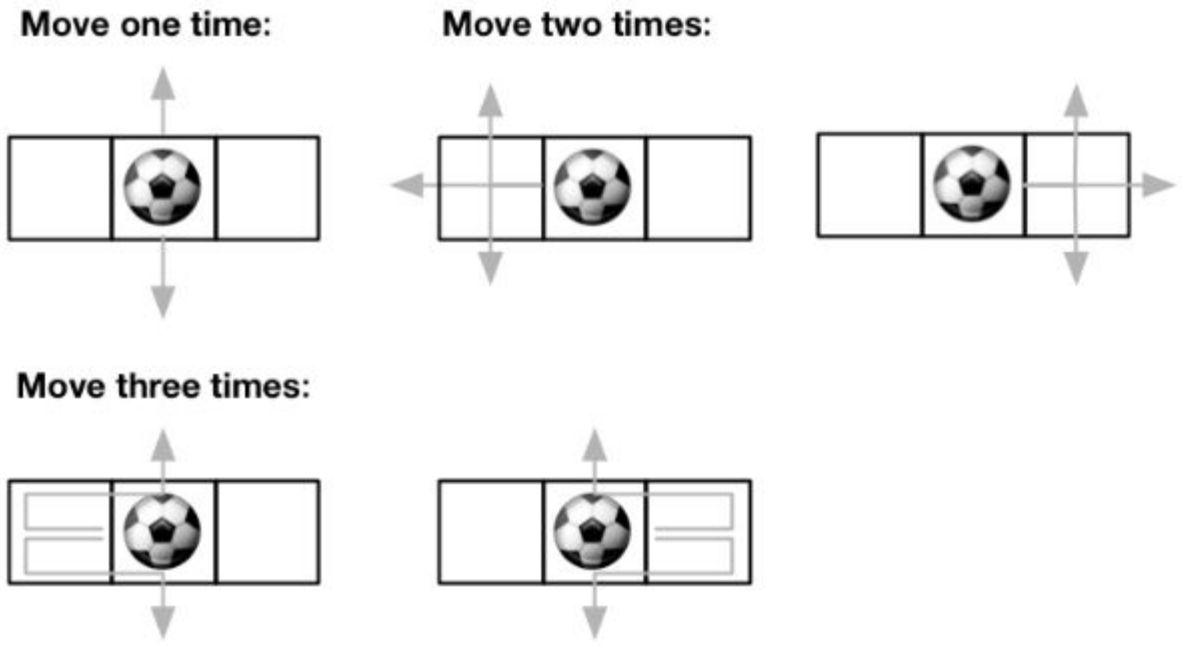
Constraints:

- `1 <= m, n <= 50`
- `0 <= maxMove <= 50`
- `0 <= startRow < m`
- `0 <= startColumn < n`

Example 1:

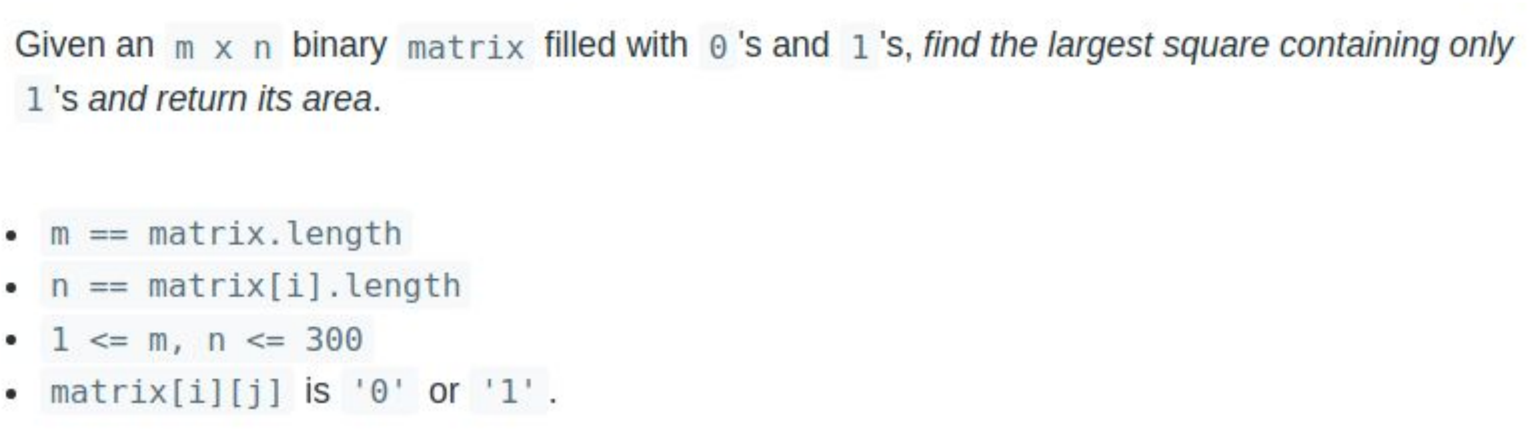


```
Input: m = 2, n = 2, maxMove = 2, startRow = 0, startColumn = 0
Output: 6
```



```
Input: m = 1, n = 3, maxMove = 3, startRow = 0, startColumn = 1
Output: 12
```

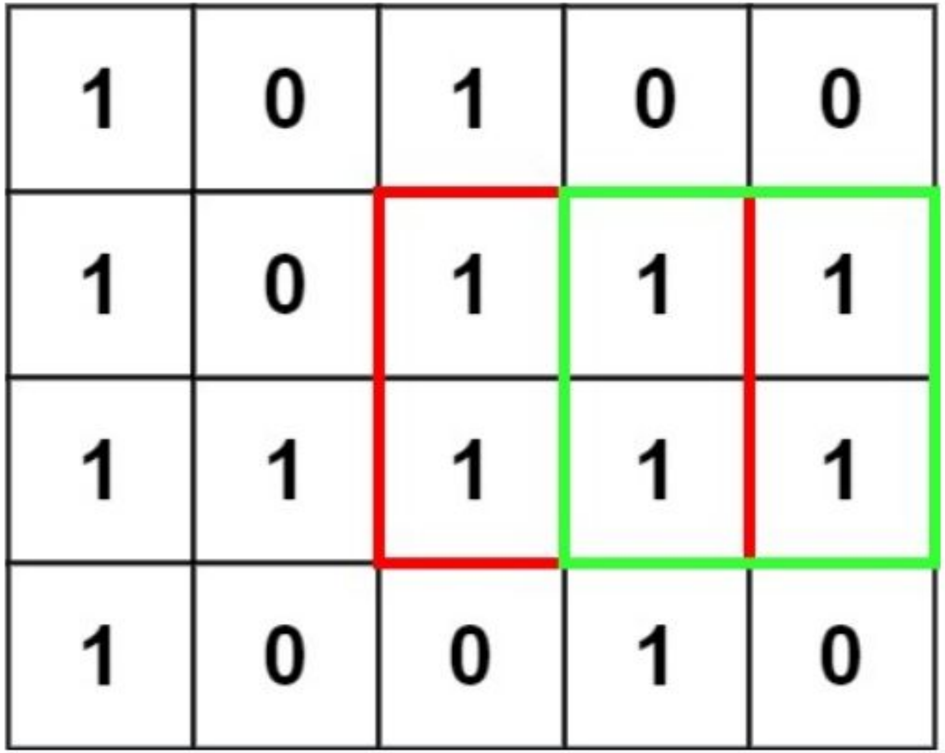
Problem #3: LeetCode 221 - Maximal Square



Given an `m x n` binary `matrix` filled with `0`'s and `1`'s, *find the largest square containing only* `1`*'s and return its area.*

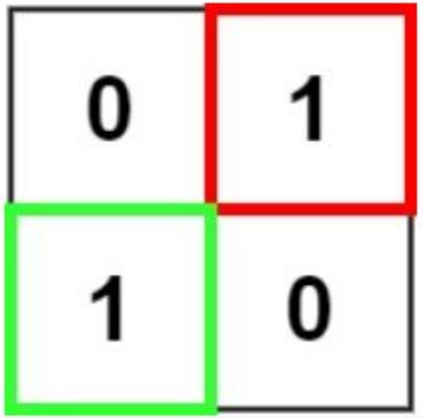
- `m == matrix.length`
- `n == matrix[i].length`
- `1 <= m, n <= 300`
- `matrix[i][j]` is `'0'` or `'1'`.

Example 1:



```
Input: matrix = [["1","0","1","0","0"],["1","0","1","1","1"],
["1","1","1","1","1"],["1","0","0","1","0"]]
Output: 4
```

Example 2:



```
Input: matrix = [["0","1"],["1","0"]]
Output: 1
```

In the next page a hint
Try first
Then come back and see the hint

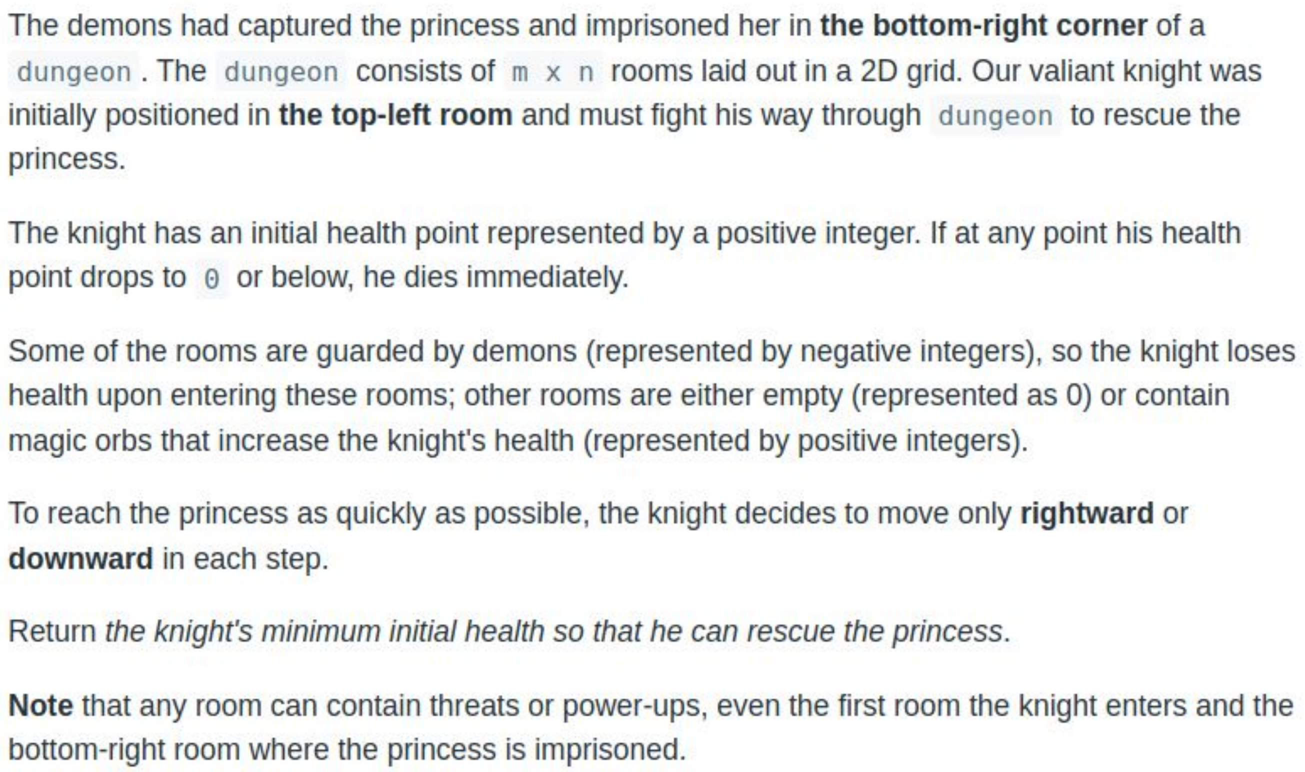
Example 3:

```
Input: matrix = [["0"]]
Output: 0
```

Hint

Let: max_square_side(r, c): is the side of the largest square containing only 1's and ending at (r, c)

Problem #4: LeetCode 174 - Dungeon Game



The demons had captured the princess and imprisoned her in **the bottom-right corner** of a `dungeon`. The `dungeon` consists of `m x n` rooms laid out in a 2D grid. Our valiant knight was initially positioned in **the top-left room** and must fight his way through `dungeon` to rescue the princess.

The knight has an initial health point represented by a positive integer. If at any point his health point drops to `0` or below, he dies immediately.

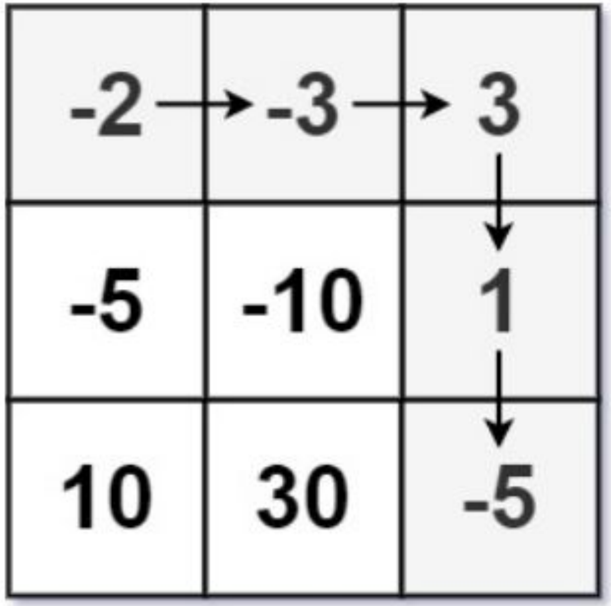
Some of the rooms are guarded by demons (represented by negative integers), so the knight loses health upon entering these rooms; other rooms are either empty (represented as 0) or contain magic orbs that increase the knight's health (represented by positive integers).

To reach the princess as quickly as possible, the knight decides to move only **rightward** or **downward** in each step.

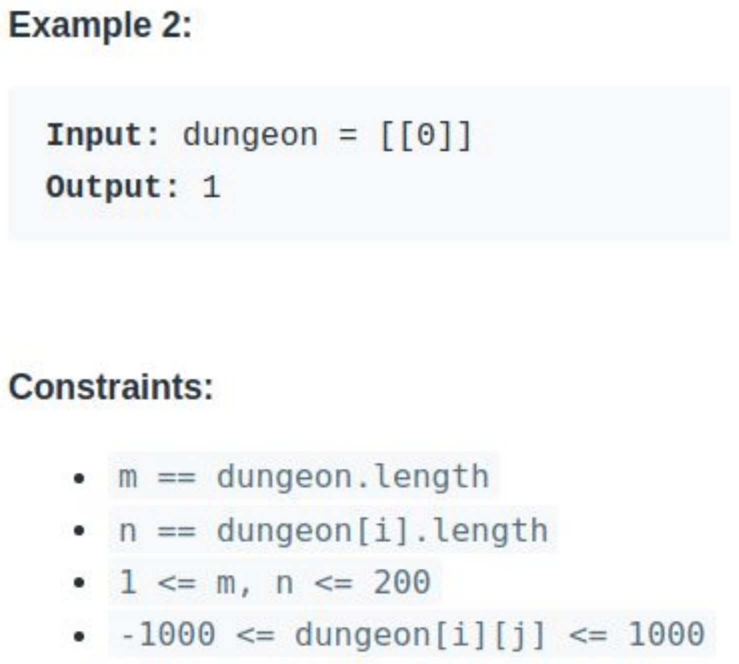
Return *the knight's minimum initial health so that he can rescue the princess.*

Note that any room can contain threats or power-ups, even the first room the knight enters and the bottom-right room where the princess is imprisoned.

Example 1:



```
Input: dungeon = [[-2,-3,3],[-5,-10,1],[10,30,-5]]
Output: 7
Explanation: The initial health of the knight must be at least 7 if he
follows the optimal path: RIGHT-> RIGHT -> DOWN -> DOWN.
```



Example 2:

```
Input: dungeon = [[0]]
Output: 1
```

Constraints:

- `m == dungeon.length`
- `n == dungeon[i].length`
- `1 <= m, n <= 200`
- `-1000 <= dungeon[i][j] <= 1000`

In the next page a hint
Try first
Then come back and see the hint

Hint

- It is actually a classical grid problem!
- The hard part of this problem is to think slowly in a logical way
- The problem asks for the minHealth, so let's try to define the DP directly based on it
- dp(r, c) = the knight's minimum initial health
    - starting from (r, c)
    - so that he can rescue the princess at bottom right
- Develop the base case
- Develop the result of the sub-calls
    - Think logically, with examples, how can you compute the state from sub-states
- Code is simple

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."