1. You are given a map of a building, and your task is to count the number of its rooms. The size of the map is n * m squares, and each square is either floor or wall. You can walk left, right, up, and down through the floor squares.

   Input
   The first input line has two integers n and m: the height and width of the map.
   Then there are n lines of m characters describing the map. Each character is either 1 (the floor) or # (the wall).
   Output
   Print one integer: the number of rooms.

   Input:

   5 8
   # # # # # # # #
   # 1 1 # 1 1 1 #
   # # # # 1 # 1 #
   # 1 1 # 1 1 1 #
   # # # # # # # #

   Output:
   3

2. You are given a map of a labyrinth, and your task is to find a path from start to end. You can walk left, right, up and down.

   Input
   The first input line has two integers n and m: the height and width of the map.
   Then there are n lines of m characters describing the labyrinth. Each character is 1 (floor), # (wall), a (start), or b (end). There is exactly one a and one b in the input.

   Output
   First print "yes" if there is a path, and "no" otherwise.
   If there is a path, print the length of the shortest such path and its description as a string consisting of characters L (left), R (right), U (up), and D (down). You can print any valid solution.

   Example
   Input:

   5 8
   # # # # # # # #
   # 1 a # 1 1 1 #
   # 1 # # 1 # b #
   # 1 1 1 1 1 1 #
   # # # # # # # #

Output:
YES
9
LDDRRRRRU

3. You have to complete n courses. There are m requirements of the form "course a has to be completed before course b". Your task is to find an order in which you can complete the courses.

   Input
   The first input line has two integers n and m: the number of courses and requirements. The courses are numbered 1,2,...,n.
   After this, there are m lines describing the requirements(or prerequisites). Each line has two integers a and b: course a has to be completed before course b.

   Output
   Print an order in which you can complete the courses. You can print any valid order that includes all the courses.
   If there are no solutions, print "IMPOSSIBLE".

   Input:

   5 3

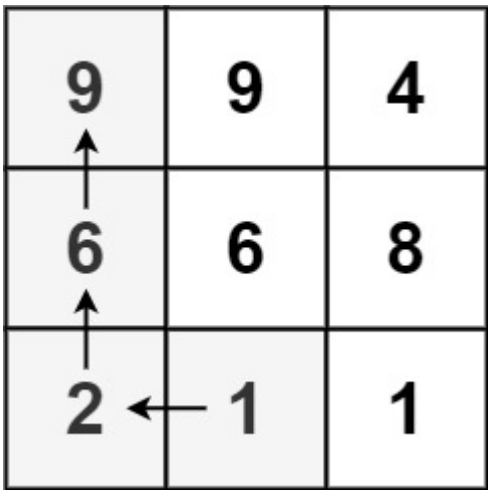
   1 2
   3 1
   4 5

   Output:

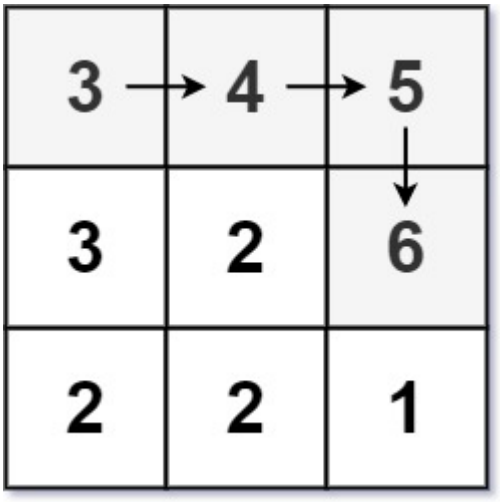
   3 4 1 5 2

4. Given an m x n integers matrix, return the length of the longest increasing path in matrix.

   From each cell, you can either move in four directions: left, right, up, or down. You may not move diagonally or move outside the boundary (i.e., wrap-around is not allowed).



Input: matrix = [[9,9,4],[6,6,8],[2,1,1]]
Output: 4
Explanation: The longest increasing path is [1, 2, 6, 9].

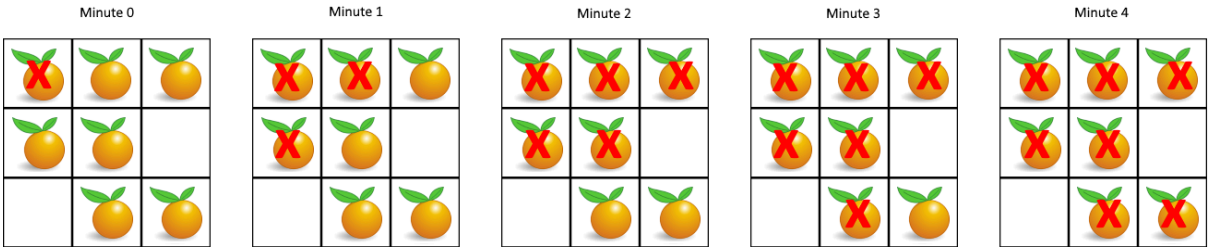


Input: matrix = [[3,4,5],[3,2,6],[2,2,1]]
Output: 4
Explanation: The longest increasing path is [3, 4, 5, 6]. Moving diagonally is not allowed.

5. You are given an m x n grid where each cell can have one of three values:

   0 representing an empty cell,
   1 representing a fresh orange, or
   2 representing a rotten orange.
   Every minute, any fresh orange that is 4-directionally adjacent to a rotten orange becomes rotten.

   Return the minimum number of minutes that must elapse until no cell has a fresh orange. If this is impossible, return -1.



Input: grid = [[2,1,1],[1,1,0],[0,1,1]]
Output: 4

Input: grid = [[2,1,1],[0,1,1],[1,0,1]]
Output: -1
Explanation: The orange in the bottom left corner (row 2, column 0) is never rotten, because rotting only happens 4-directionally.

Input: grid = [[0,2]]
Output: 0
Explanation: Since there are already no fresh oranges at minute 0, the answer is just 0.

6. You are given an m x n integer matrix grid, where you can move from a cell to any adjacent cell in all 4 directions.

   Return the number of strictly increasing paths in the grid such that you can start from any cell and end at any cell. Since the answer may be very large, return it modulo 10^9 + 7.

   Two paths are considered different if they do not have exactly the same sequence of visited cells.

   Example 1:

1	1
3	4

Input: grid = [[1,1],[3,4]]
Output: 8
Explanation: The strictly increasing paths are:
- Paths with length 1: [1], [1], [3], [4].
- Paths with length 2: [1 -> 3], [1 -> 4], [3 -> 4].
- Paths with length 3: [1 -> 3 -> 4].
The total number of paths is 4 + 3 + 1 = 8.

Example 2:

Input: grid = [[1],[2]]
Output: 3
Explanation: The strictly increasing paths are:
- Paths with length 1: [1], [2].
- Paths with length 2: [1 -> 2].
The total number of paths is 2 + 1 = 3.