

## 6366. Minimum Time to Visit a Cell In a Grid

My Submissions (/contest/weekly-contest-334/problems/minimum-time-to-visit-a-cell-in-a-grid/submissions/)

Back to Contest (/contest/weekly-contest-334/)

You are given a m x n matrix grid consisting of non-negative integers where grid [row] [col] represents the minimum time required to be able to visit the cell (row, col), which means you can visit the cell (row, col) only when the time you visit it is greater than or equal to grid[row][col] .

You are standing in the top-left cell of the matrix in the  $0^{th}$  second, and you must move to any adjacent cell in the four directions: up, down, left, and right. Each move you make takes 1 second.

Return the minimum time required in which you can visit the bottom-right cell of the matrix. If you cannot visit the bottomright cell, then return −1.

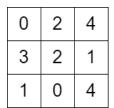
User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Hard

## Example 1:

0	1	3	2
5	1	2	5
4	3	8	6

```
Input: grid = [[0,1,3,2],[5,1,2,5],[4,3,8,6]]
Output: 7
Explanation: One of the paths that we can take is the following:
- at t = 0, we are on the cell (0,0).
- at t = 1, we move to the cell (0,1). It is possible because grid[0][1] <= 1.
- at t = 2, we move to the cell (1,1). It is possible because grid [1][1] <= 2.
- at t = 3, we move to the cell (1,2). It is possible because grid [1][2] <= 3.
- at t = 4, we move to the cell (1,1). It is possible because grid[1][1] \ll 4.
- at t = 5, we move to the cell (1,2). It is possible because grid[1][2] <= 5.
- at t = 6, we move to the cell (1,3). It is possible because grid [1][3] <= 6.
- at t = 7, we move to the cell (2,3). It is possible because grid [1][3] <= 7.
The final time is 7. It can be shown that it is the minimum time possible.
```

## Example 2:



Input: grid = [[0,2,4],[3,2,1],[1,0,4]]Output: -1 Explanation: There is no path from the top left to the bottom-right cell.

## Constraints:

- m == grid.length
- n == grid[i].length
- 2 <= m, n <= 1000
- $4 \le m * n \le 10^5$
- 0 <= grid[i][j] <= 10<sup>5</sup>
- grid[0][0] == 0







1 const initialize2DArray = (n, m) => [...Array(n)].map(() => Array(m).fill(Number.MAX\_SAFE\_INTEGER));

```
2 | const dx = [-1, 1, 0, 0], dy = [0, 0, -1, 1];
 4 \vee \text{const minimumTime} = (g) \Rightarrow \{
 5
         if (g[0][1] > 1 && g[1][0] > 1) return -1;
 6 •
         let pq = new MinPriorityQueue({
             compare: (x, y) => {
    if (x[0] != y[0]) return x[0] - y[0];
 7 •
 8
 9
                 if (x[1] != y[1]) return x[1] - y[1];
                 return x[2] - y[2];
10
11
         }), n = g.length, m = g[0].length, dis = initialize2DArray(n, m);
12
13
         pq.enqueue([0, 0, 0]);
14
         dis[0][0] = 0;
15 ▼
         while (pq.size()) {
             let [v, x, y] = pq.dequeue()
16
17
             if (x == n - 1 \& y == m - 1) return v;
18 •
             for (let k = 0; k < 4; k++) {
                  let nx = x + dx[k], ny = y + dy[k];
19
                  if (nx < 0 \mid | nx >= n \mid | ny < 0 \mid | ny >= m) continue;
20
21
                 let diff = g[nx][ny] - v;
22 •
                 if (diff < 0) {
23
                      diff = 0;
                 } else if (diff & 1) {
24 ▼
25
                      diff--;
26
27
                 let nv = v + 1 + diff;
28 •
                  if (dis[nx][ny] > nv) {
29
                      dis[nx][ny] = nv;
30
                      pq.enqueue([nv, nx, ny]);
31
                 }
32
             }
33
         }
34
         return -1;
35
    };
```

☐ Custom Testcase

Use Example Testcases

• Run ) (

**△** Submit

Submission Result: Accepted (/submissions/detail/905710966/) 2

More Details > (/submissions/detail/905710966/)

Share your acceptance!

Copyright © 2023 LeetCode

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)