6081. Minimum Obstacle Removal to Reach Corner

My Submissions (/contest/weekly-contest-295/problems/minimum-obstacle-removal-to-reach-corner/submissions/)

Back to Contest (/contest/weekly-contest-295/)

You are given a **0-indexed** 2D integer array `grid` of size `m x n`. Each cell has one of two values:

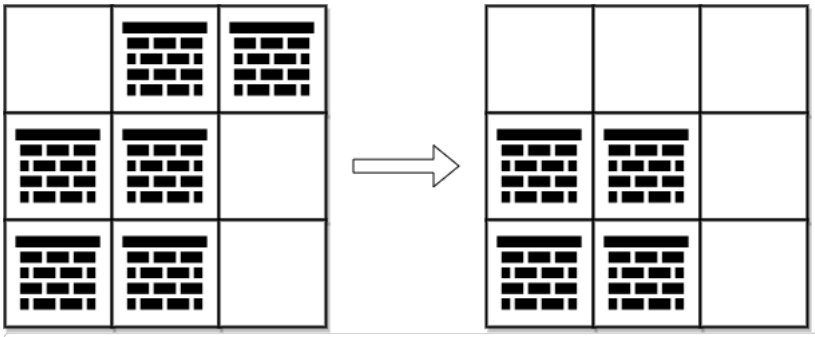
- `0` represents an **empty** cell,
- `1` represents an **obstacle** that may be removed.

You can move up, down, left, or right from and to an empty cell.

Return *the **minimum** number of **obstacles** to **remove** so you can move from the upper left corner* `(0, 0)` *to the lower right corner* `(m - 1, n - 1)`.

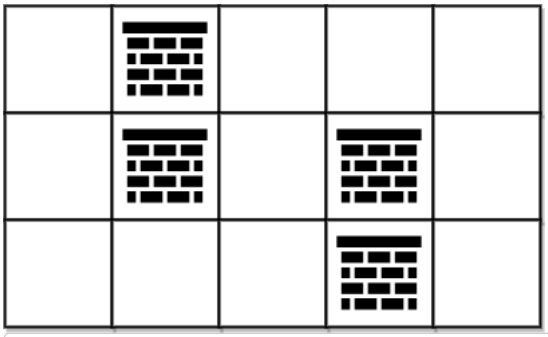
User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Hard

Example 1:



```
Input: grid = [[0,1,1],[1,1,0],[1,1,0]]
Output: 2
Explanation: We can remove the obstacles at (0, 1) and (0, 2) to create a path from (0, 0) to (2, 2).
It can be shown that we need to remove at least 2 obstacles, so we return 2.
Note that there may be other ways to remove 2 obstacles to create a path.
```

Example 2:



```
Input: grid = [[0,1,0,0,0],[0,1,0,1,0],[0,0,0,1,0]]
Output: 0
Explanation: We can move from (0, 0) to (2, 4) without removing any obstacles, so we return 0.
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 10^5`
- `2 <= m * n <= 10^5`
- `grid[i][j]` is either `0` **or** `1`.
- `grid[0][0] == grid[m - 1][n - 1] == 0`

JavaScript

```
const initialize2DArray = (n, m) => { let d = []; for (let i = 0; i < n; i++) { let t =
Array(m).fill(Number.MAX_SAFE_INTEGER); d.push(t); } return d; };

const minimumObstacles = (g) => minDis(g);

const dx = [-1, 1, 0, 0], dy = [0, 0, -1, 1];
const minDis = (g) => {
    if (g[0][0]) return -1;
    let n = g.length, m = g[0].length, allow = 0, forbid = 1;
    let q = [[0, 0]], dis = initialize2DArray(n, m);
    dis[0][0] = 0;
    while (q.length) {
        let [x, y] = q.shift();
        for (let k = 0; k < 4; k++) {
            let nx = x + dx[k], ny = y + dy[k];
            if (nx < 0 || nx >= n || ny < 0 || ny >= m) continue;
            if (g[nx][ny] == forbid) {
                if (dis[nx][ny] > dis[x][y] + 1) {
                    dis[nx][ny] = dis[x][y] + 1;
                    q.push([nx, ny]);
                }
            } else {
                if (dis[nx][ny] > dis[x][y]) {
                    dis[nx][ny] = dis[x][y];
                    q.unshift([nx, ny]);
                }
            }
        }
    }
    return dis[n - 1][m - 1];
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/709588995/) | More Details > (/submissions/detail/709588995/)

Share your acceptance!