


## 2257. Count Unguarded Cells in the Grid

Solved 

Medium

Topics

Companies

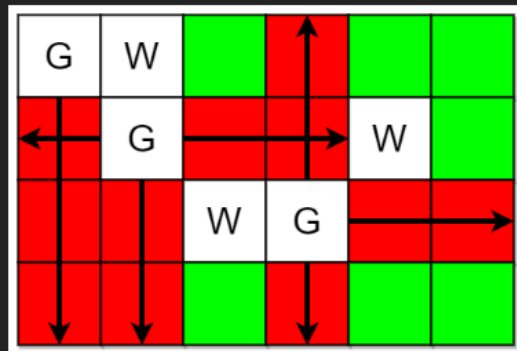
Hint

You are given two integers  $m$  and  $n$  representing a **0-indexed**  $m \times n$  grid. You are also given two 2D integer arrays `guards` and `walls` where `guards[i] = [rowi, coli]` and `walls[j] = [rowj, colj]` represent the positions of the  $i^{\text{th}}$  guard and  $j^{\text{th}}$  wall respectively.

A guard can see **every** cell in the four cardinal directions (north, east, south, or west) starting from their position unless **obstructed** by a wall or another guard. A cell is **guarded** if there is **at least** one guard that can see it.

Return *the number of unoccupied cells that are **not** guarded*.

**Example 1:**



**Input:** `m = 4, n = 6, guards = [[0,0],[1,1],[2,3]], walls = [[0,1],[2,2],[1,4]]`

**Output:** 7

**Explanation:** The guarded and unguarded cells are shown in red and green respectively in the above diagram.

There are a total of 7 unguarded cells, so we return 7.

**Example 2:**

	W	
W	G	W
	W	

**Input:** `m = 3, n = 3, guards = [[1,1]], walls = [[0,1],[1,0],[2,1],[1,2]]`

**Output:** 4

**Explanation:** The unguarded cells are shown in green in the above diagram.

There are a total of 4 unguarded cells, so we return 4.

### Constraints:

- $1 \leq m, n \leq 10^5$
- $2 \leq m * n \leq 10^5$
- $1 \leq \text{guards.length}, \text{walls.length} \leq 5 * 10^4$
- $2 \leq \text{guards.length} + \text{walls.length} \leq m * n$
- $\text{guards}[i].\text{length} == \text{walls}[j].\text{length} == 2$
- $0 \leq \text{row}_i, \text{row}_j < m$
- $0 \leq \text{col}_i, \text{col}_j < n$
- All the positions in `guards` and `walls` are **unique**.

## Python:

class Solution:

def countUnguarded(self, m: int, n: int, guards: List[List[int]], walls: List[List[int]]) -> int:

# Initialize grid with zeros

```

g = [[0] * n for _ in range(m)] # g - 2D grid/matrix

# Mark guards and walls as 2
for x, y in guards: # gx, gy - guard's position
    g[x][y] = 2
for x, y in walls:
    g[x][y] = 2

# Directions: up, right, down, left
dirs = [(-1, 0), (0, 1), (1, 0), (0, -1)]

# Process each guard's line of sight
for gx, gy in guards:
    for dx, dy in dirs: # dx, dy - movement direction offsets
        x, y = gx, gy
        while True:
            x += dx
            y += dy
            # Check cells in current direction until hitting boundary or obstacle
            if x < 0 or x >= m or y < 0 or y >= n or g[x][y] == 2:
                break
            g[x][y] = 1

# Count unguarded cells (cells with value 0)
return sum(row.count(0) for row in g)

```

## JavaScript:

```

// JavaScript

var countUnguarded = function(m, n, guards, walls) {
    // Initialize grid with zeros
    const g = Array(m).fill().map(() => Array(n).fill(0));

    // Mark guards and walls as 2
    for (const [x, y] of guards) {
        g[x][y] = 2;
    }
    for (const [x, y] of walls) {
        g[x][y] = 2;
    }

    // Directions: up, right, down, left
    const dirs = [[-1, 0], [0, 1], [1, 0], [0, -1]];

```

```

// Process each guard's line of sight
for (const [gx, gy] of guards) {
  for (const [dx, dy] of dirs) {
    let x = gx;
    let y = gy;
    while (true) {
      x += dx;
      y += dy;
      // Check cells in current direction until hitting boundary or obstacle
      if (x < 0 || x >= m || y < 0 || y >= n || g[x][y] === 2) {
        break;
      }
      g[x][y] = 1;
    }
  }
}

// Count unguarded cells (cells with value 0)
return g.reduce((sum, row) => sum + row.filter(cell => cell === 0).length, 0);
};

```

## Java:

```

class Solution {
  public int countUnguarded(int m, int n, int[][] guards, int[][] walls) {
    // Initialize grid with zeros
    int[][] g = new int[m][n];

    // Mark guards and walls as 2
    for (int[] e : guards) {
      g[e[0]][e[1]] = 2;
    }
    for (int[] e : walls) {
      g[e[0]][e[1]] = 2;
    }

    // Directions: up, right, down, left
    int[] dirs = {-1, 0, 1, 0, -1};

    // Process each guard's line of sight
    for (int[] e : guards) {
      for (int k = 0; k < 4; ++k) {
        int x = e[0], y = e[1];
        int dx = dirs[k], dy = dirs[k + 1];

```

```

        // Check cells in current direction until hitting boundary or obstacle
        while (x + dx >= 0 && x + dx < m && y + dy >= 0 && y + dy < n && g[x + dx][y + dy] <
2) {
            x += dx;
            y += dy;
            g[x][y] = 1;
        }
    }
}

// Count unguarded cells (cells with value 0)
int unguardedCount = 0;
for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
        if (g[i][j] == 0) {
            unguardedCount++;
        }
    }
}

return unguardedCount;
}
}

```