Count Unguarded Cells In The Grid

solved by	Senan
Platform	LeetCode
difficulty	Medium
# Serial	2257
tags	DFS
language	C++
solved on	@21/11/2024
link	https://leetcode.com/problems/count-unguarded-cells-in-the-grid/submissions/1458771426/
Completion	☑

Intuition

Guards and walls block certain cells in the grid. To solve this, we track all cells that are guarded using directional sweeps from each guard while stopping at walls or boundaries. We count the remaining unvisited cells as unguarded.

Approach

1. Use a 2D grid `visited` to track the status of each cell:
    - `0`: unvisited (potentially unguarded).
    - `1`: guarded.
    - `2`: wall.
    - `3`: guard.
2. Initialize the grid with walls and guards.
3. For each guard, simulate its guarding directions (up, down, left, right) using vectors `dr` and `dc`. Guard cells until hitting a wall, boundary, or another guard.
4. Track the number of unblocked cells (`notBlock`) as the total cells minus walls and guards.
5. Deduct from `notBlock` whenever an unvisited cell is marked as guarded.
6. Return `notBlock` as the final count of unguarded cells.

Complexity

Time Complexity:

- Initializing the grid with walls and guards: **O(k + w)**, where **k** is the number of guards and **w** is the number of walls.
- Guarding each direction from each guard: **O(k max(m, n))**, since each guard can cover at most one full row or column.
- Total: **O(k max(m, n) + w)**.

Space Complexity:

- **O(m n)** for the `visited` grid.

Code

```
class Solution {
public:
    int countUnguarded(int m, int n, vector<vector<int>>& guards,
                       vector<vector<int>>& walls) {
        vector<vector<int>> visited(m, vector<int>(n, 0));
        vector<int> dr = {1, 0, -1, 0};
        vector<int> dc = {0, 1, 0, -1};

        int notBlock = m*n - guards.size() - walls.size();

        for (auto& wall : walls)
            visited[wall[0]][wall[1]] = 2;
        for (auto& guard : guards)
            visited[guard[0]][guard[1]] = 3;

        for (auto& guard : guards) {
            int row = guard[0];
            int col = guard[1];

            for (int i = 0; i < 4; i++) {
                int nRow = row + dr[i];
                int nCol = col + dc[i];

                while (0 <= nRow && nRow < m && 0 <= nCol && nCol < n &&
                       visited[nRow][nCol] != 2) {

                    if (visited[nRow][nCol] == 0){
                        visited[nRow][nCol] = 1;
                        notBlock--;
                    }

                    nRow += dr[i];
                    nCol += dc[i];
                }
            }
        }

        return notBlock;
    }
};
```