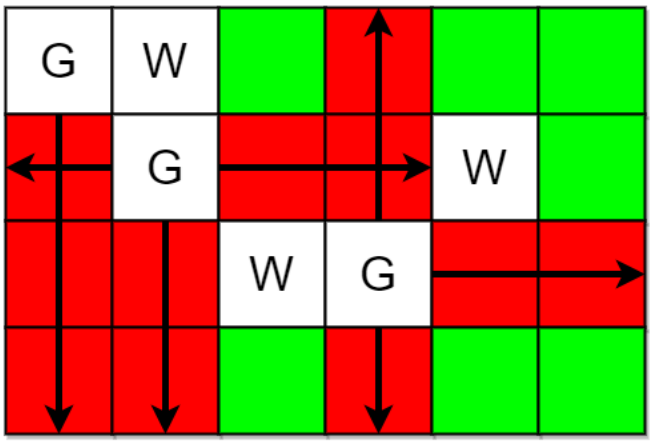
2257. Count Unguarded Cells in the Grid

You are given two integers `m` and `n` representing a **0-indexed** `m x 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`th guard and `j`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.

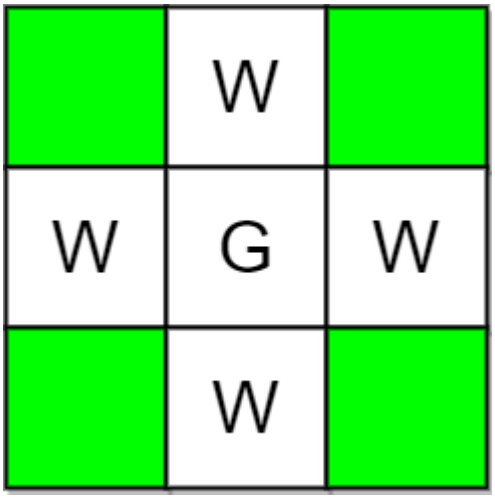
Return *the number of unoccupied cells that are* ***not guarded***.

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:




```
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 <= m, n <= 10^5`
- `2 <= m * n <= 10^5`
- `1 <= guards.length, walls.length <= 5 * 10^4`
- `2 <= guards.length + walls.length <= m * n`
- `guards[i].length == walls[j].length == 2`
- `0 <= rowi, rowj < m`
- `0 <= coli, colj < n`
- All the positions in `guards` and `walls` are **unique**.

2257. Count Unguarded Cells in the Grid

```
/*
   Straight forward Simualation
   Time compelxity: O(g+w+min(g(m+n),4.m.n)+m.n)
   Space complexity: O(m.n)
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;
class Solution {
   public:
      int countUnguarded(int m, int n, vvi& guards, vvi& walls){
         vvi grid(m,vi(n,0));

         // Fill the grid as describe in guards and walls
         for(auto& g: guards){
            grid[g[0]][g[1]]=1;
         }
         for(auto& w: walls){
            grid[w[0]][w[1]]=-1;
         }
```

```
        // For each guard
        for(auto& g: guards){
            int i=g[0];
            int j=g[1];
            // see west
            int k=j-1;
            while(k>=0 && grid[i][k]!=1 && grid[i][k]!=-1) grid[i][k]=2,k--;

            // see east
            k=j+1;
            while(k<n && grid[i][k]!=1 && grid[i][k]!=-1) grid[i][k]=3,k++;

            // see north
            k=i-1;
            while(k>=0 && grid[k][j]!=1 && grid[k][j]!=-1) grid[k][j]=4,k--;

            // see south
            k=i+1;
            while(k<m && grid[k][j]!=1 && grid[k][j]!=-1) grid[k][j]=5,k++;

        }

        // Count not guarded cells (with value equal to 0)
        int ans=0;
        for(int i=0;i<m;++i){
            for(int j=0;j<n;++j){
                ans+=int(grid[i][j]==0);
            }
        }

        return ans;
    }
};
```