STUPOVI

We can use a greedy algorithm. In each step, we find which meter on the sidewalk should be blocked so that the largest number of possible parking spots is blocked. We repeat this until no cars can park or we've run out of pillars.

Notice that, for each sequence of $2L-1$ consecutive free meters, placing a pillar on the middle meter will block L spots that cars could have parked at. If there are no such sequences then it is best to place a pillar in the longest sequence of free meters. There is no way to achieve better efficiency.

There are also dynamic programming solutions, but these are less efficient than the greedy solution.

VANDAL

The problem in the task is not counting one cell multiple times. A mathematical solution "with if statements" is possible, but since N is at most 10^7, a O(N) solution will be efficient enough, and will be considerably simpler.

One such solution is:

1. Count all cells in row A.
2. Count all cells in column B, except those in row A.
3. Count all cells on diagonal C which aren't in row A or column B.
4. Count all cells on diagonal D which aren't in row A, column B or on diagonal C.

Each of these steps counts at most N cells, so the algorithm does at most 4N steps.

GULIVER

We need to flood the smallest number of squares to divide the island into two parts, Northern and Southern. In other words, after flooding, there must not be a path from the first row to the last row over white cells (not flooded), moving in the four main directions.

Observe that such a path doesn't exist only if there is a path from the first column to the last column over black (flooded) cells, when diagonal moves can also be used (for a total of eight directions).

We can construct a graph in which each cell is connected to eight others, its neighbours. The weight of each edge is 0 if the edge enters an already flooded cell and 1 if it enters a dry square. In such a graph, we need to find the shortest path from the first column to the last. With this we have also found the fewest cells that need flooding to separate the first and last rows.

The shortest path can be found using Dijkstra's algorithm, or a simple modification of breadth-first search (BFS): when using an edge of weight 0, the new vertex is placed at the front of the queue; when using and edge of weight 1, the vertex is placed at the end of the queue.