PRUGE

There are many ways to solve the problem, one of which is even of constant time complexity. The simplest solution which runs in time, however, is to proceed row by row (or column by column) and calculate the number of grey squares in that row. One way to do that is to split the row into 3 parts: above the diagonal ($y > x$), below the diagonal ($y < x$) and the diagonal ($x = y$).

Since there are at most one million rows and columns, every such solution that doesn't count the squares one by one will suffice.

PETICE

We proceed from right to left, increasing the current digit by 1 until there are enough digits 5 in the number. When the current digit becomes 5, we continue on with the digit to its left. Note that, for example, increasing the digit 9 in 49 gives the number 50. If one digit 5 is sought, we are done at that point. If two digits are required, we continue increasing the rightmost digit until we are at 55.

MARATON

Imagine a solution that simulates the marathon game by game. We only need two bits of information at any give time: which team just got onto the field and which team stayed on the field as the winner of the previous game.

These two bits (call them the *state*) uniquely determine the winner of the game and the team to get onto the field next i.e. we know the next state.

Since the maximum number of states is N^2, then by the pigeonhole principle, the marathon will run into a cycle after at most that many games. We can use this fact to find the cycle, then speed up our algorithm by simulating the full cycle as many times as possible instead of game by game. We will most likely also have to simulate a couple more games that get left over, because the cycle doesn't reach an end in the last iteration. However, there will again be less than N^2 such games.