Nautilus

Spoiler

Abridged problem statement. There is a $R \times C$ grid of characters '`#`' and '`.`' and a ship that can only navigate the '`.`' tiles. The ship can move in the four main directions. You know some of the last M movements of the ship, the others are marked with '`?`'. Find out the number of possible positions the ship could currently be at.

Subtask 1. ($1 \le R, C, M \le 100$; there are no '`?`' queries)

The solution to this subtask is straightforward — try all possible starting positions. It is simple to verify whether the ship could have started at a given position: simply simulate the ship's movement, starting from that position. If you hit an edge or island, you know that the ship couldn't have started there.

Verifying this for each position takes $\mathcal{O}(M)$ time. There are RC possible starting positions: total time complexity will be $\mathcal{O}(RCM)$.

Subtask 2. ($1 \le R, C, M \le 100$)

We solve this using dynamic programming. We want $\texttt{dp}[i][j][t]$ to be 1 if and only if it is possible that the ship is in position i, j after the first t steps, considering only the first t characters of M.

We can make some observations. If there is an island at position (i, j), then $\texttt{dp}[i][j][t]$ will be 0, so let's look at the case where there is no island at position (i, j). It's clear that $\texttt{dp}[i][j][0] = 1$ for any i and j. If the message at position t is a letter, for example '`E`', then $\texttt{dp}[i][j][t] = \texttt{dp}[i][j-1][t-1]$, because if the ship indeed was at position (i, j) at time t and went east before that, then it must have been at $(i, j-1)$ at time $t-1$. If the message at position t is a question mark, then $\texttt{dp}[i][j][t]$ is 1 if and only if at least one of $\texttt{dp}[i][j-1][t-1], \texttt{dp}[i][j+1][t-1], \texttt{dp}[i-1][j][t-1], \texttt{dp}[i+1][j][t-1]$ is 1, because the ship must have come from one of these squares.

Using these relations, we can calculate $\texttt{dp}[i][j][M]$ for all i and j. The answer is the number of pairs (i, j) such that $\texttt{dp}[i][j][M] = 1$.

The complexity is once again $\mathcal{O}(RCM)$.

Subtask 3. ($1 \le R, C \le 500; 1 \le M \le 5000$)

The complexity shall still be $\mathcal{O}(RCM)$, but we are going to use a very powerful way to optimize: bitsets! In C++, `bitset` is a data type that can be thought of as an array of booleans that supports, among other things, the following operations: bitwise "or" (denoted $|$); bitwise "and" (denoted $\&$) and the left and right bitwise shifts (denoted $\ll$ and $\gg$ respectively). Bitwise "and" and "or" are simply the logical "and" and "or" operations, applied to each bit separately. For example:

$$\begin{array}{c|cccccccc} & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ | & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ \hline & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{array} \qquad \begin{array}{cccccccccc} & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ \& & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ \hline & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$$

The bitwise shifts shift the whole string by an integer number of positions, appending or prepending zeroes if necessary. For example: $00110101 \ll 3 = 10101000$; $11001111 \gg 2 = 00110011$. Compared to manually doing those operations on an array of booleans, they are very fast, because there exist machine-level instructions for doing them.

We represent $\texttt{dp}[i][t]$ as a bitset, where $\texttt{dp}[i][t]$ consists of what $\texttt{dp}[i][1][t], \texttt{dp}[i][2][t], \ldots, \texttt{dp}[i][C][t]$ were in the last subtask. Also let $\texttt{sea}[i]$ be a bitset of the i-th row of the grid – where '`.`' is converted to 1 and '`#`' is converted to 0. Now notice that we can do the entire computation in bitsets:

$$\texttt{dp}[i][t] = \begin{cases} \texttt{dp}[i+1][t-1] \,\&\, \texttt{sea}[i] & \text{if the } t\text{-th message is } \texttt{N} \\ (\texttt{dp}[i] \ll 1) \,\&\, \texttt{sea}[i] & \text{if the } t\text{-th message is } \texttt{E} \\ (\texttt{dp}[i+1][t-1] \mid \texttt{dp}[i-1][t-1] \mid & \\ \mid (\texttt{dp}[i][t-1] \ll 1) \mid (\texttt{dp}[i][t-1] \gg 1)) \,\&\, \texttt{sea}[i] & \text{if the } t\text{-th message is } \texttt{?} \end{cases}$$

Cases `S` and `W` are handled analoguously.

How to do this in Java or Python? In Python, the jury simply used Python's big integers and treated them as bitstrings. In Java, both the `BigInteger` and `BitSet` classes exist, but neither work for this problem: `BigInteger` is too slow and `BitSet` lacks the bit-shift operators that we require. Instead, we replaced each bitset with a short array of `long`s.

In fact, there are other ways to optimize the C++ solution. For example including the following line

```
#pragma GCC optimize("Ofast")
```

will include optimizations hardcore enough to be comparable with bitsets.

Credits

- Task: Marijonas Petrauskas (Lithuania)
- Solutions and tests: Tähvend Uustalu, Andres Unt (Estonia)