Code Jam 2019 - Qualification Round

Analysis: You Can Go Your Own Way

Test set 1

In the first test set, we can construct all possible paths from the northwest cell of the maze to the southeast cell using backtracking, and see which of them satisfy our requirements (we know from the Output section that at least one answer exists). When we find one that works, we output it as an answer for the test and stop looking any furthter.

Test set 2

Let's assess the number of possible paths from the northwest cell to the southeast cell. For a maze of size **N** by **N**, every possible valid path makes **N** - 1 moves to the east and **N** - 1 moves to the south. Notice that the order in which these moves are made does not matter – we will always arrive at the southeast cell after making all of them and we are guaranteed not to leave the maze in the process.

So we need to make 2**N** - 2 moves total, out of which **N** - 1 are to the east and **N** - 1 are to the south, and the order does not matter. Using combinations, we can see that there are `C(2N - 2, N - 1)` possible options. For **N** ≤ 10 in the test set 1 there are at most 48620 available paths that we need to check. For test set 2, however, in which **N** = 100, there are `22750883079422934966181954039568885395604168260154104734000` (or approximately 2.28×10^{58}) possible paths to take. This is too much to process in the time limit, so let's think of an alternative solution.

We can think of a maze as a graph, where the unit cells are nodes and there is an edge between every pair of nodes that represent neighboring cells. Now instead of moving between two neighboring cells, we will move between the corresponding nodes along the edge connecting them. Because we cannot reuse Lydia's moves, the edges that she used before are no longer available to us, and we remove them from the graph. After that, our problem of finding a valid path reduces to the problem of finding *any* path from the node representing the northwest cell to the node representing the southeast cell. This is a standard graph problem that can be solved using either Depth-first search or Breadth-first search in $O(\mathbf{N}^2)$ time, which is fast enough to pass this test set.

Test set 3

With **N** ≤ 50000 now, we must think of a different approach to the problem.

To solve this test set, let's just invert all of the moves in Lydia's path. That is, every time she moves east, let's move south, and every time she moves south, let's move east. For example, if Lydia's path is `EESSSESE`, then our path will be `SSEEESES`.

Let's understand why this inverted path is a correct answer to the problem.

First, notice that we still make **N** - 1 moves to the east and **N** - 1 moves to the south, so we will arrive at the southeast cell in the end as required, and we will not step out of the bounds of the maze.

Now let's see why we will not reuse any of Lydia's moves. Suppose this is not the case, and we reuse a move from the position that is `X` moves to the east and `Y` moves to the south in some order from the northwest cell. Recall that the order of moves does not matter, and there may be many ways to get to this position, but all of them will require exactly `X` moves to the east and `Y` moves to the south. What will be the next move? We know that Lydia's next move is the `(X + Y + 1)`-th symbol in the string representing her path (with indexing starting from one), and our next move is the `(X + Y + 1)`-th symbol in the string representing our path. But since our path string is an inverted version of Lydia's path string, we know that `(X + Y + 1)`-th symbols of the two strings will be different, which contradicts our assumption that we will have the same next move. By the same logic, we can see that the two paths will not reuse any other moves along the way.