Ghost Gym

Let's consider a simpler version of the problem where instead of traps, we have walls—i.e. attempting to walk into a wall would just result in you staying in your previous position, instead of being teleported back to the start.

Then, this is just a straightforward application of depth first search. Create an $r \times c$ grid that serves as your "map", where each tile can be in one of three states: visited, a wall, or unknown. Initially, all tiles are in an unknown state, except for your starting position. To explore from some tile, do the following:

- Inspect the four tiles that are adjacent to it and attempt to visit each unknown one:
  - If the tile is a wall, then mark it as a wall and immediately return.
  - If the tile is safe, then mark it off as visited and then recursively try exploring from there.
- If none of these tiles led to the destination, then backtrack one step, so you can try a different branch from the previous tile.

This explores all reachable tiles in $\leq 2rc$ steps, since each tile contributes at most 2 steps to the total: once when you first attempt to enter, and (for non-walls) once when you backtrack to its parent tile.

How about the version of the problem with traps instead of walls? If we try to explore a tile that contains a trap, we are teleported back to the entrance instead of staying where we are. Well... then just retrace your steps so that you can return to your previous position!

Let's maintain a stack-like list of steps, such that at all times, it contains the path to our current position.

- Whenever you are to enter a new tile, append the direction of your step (`N`/`S`/`E`/`W`) to the end of the list.
- Whenever you are to exit a tile (to backtrack), pop off the most recent step in the list.
- Whenever you get caught in a trap and are teleported back to the start, read out the steps in your list, in order, to return to your previous position.

To count the number of extra steps introduced by retracing your steps, note that the order in which you visit the nodes during a DFS induces a spanning tree called a *DFS tree.* When you get caught in a trap and need to return to your previous position, the number of steps you need to take is one less than the depth of the trap node in the spanning tree.

But, you can show by induction that the sum of the depths of all nodes in a tree with rc vertices is $\leq 0 + 1 + 2 + 3 + \cdots + (rc - 1) = rc(rc - 1)/2$.

With $rc \leq 400$, our solution takes at most this many steps:

$$2rc + \frac{rc(rc-1)}{2} = 80600,$$

which is fewer than the upper bound of 10^5.