--- Day 20: A Regular Map ---

While you were learning about instruction pointers, the Elves made considerable progress. When you look up, you discover that the North Pole base construction project has completely surrounded you.

The area you are in is made up entirely of rooms and doors. The rooms are arranged in a grid, and rooms only connect to adjacent rooms when a door is present between them.

For example, drawing rooms as `.`, walls as `#`, doors as `|` or `-`, your current position as `X`, and where north is up, the area you're in might look like this:

```
#####
#.|.#
#-###
#.|X#
#####
```

You get the attention of a passing construction Elf and ask for a map. "I don't have time to draw out a map of this place - it's huge. Instead, I can give you directions to every room in the facility!" He writes down some directions on a piece of parchment and runs off. In the example above, the instructions might have been `^WNE$`, a regular expression or "regex" (your puzzle input).

The regex matches routes (like `WNE` for "west, north, east") that will take you from your current room through various doors in the facility. In aggregate, the routes will take you through every door in the facility at least once; mapping out all of these routes will let you build a proper map and find your way around.

`^` and `$` are at the beginning and end of your regex; these just mean that the regex doesn't match anything outside the routes it describes. (Specifically, `^` matches the start of the route, and `$` matches the end of it.) These characters will not appear elsewhere in the regex.

The rest of the regex matches various sequences of the characters `N` (north), `S` (south), `E` (east), and `W` (west). In the example above, `^WNE$` matches only one route, `WNE`, which means you can move west, then north, then east from your current position. Sequences of letters like this always match that exact route in the same order.

Sometimes, the route can branch. A branch is given by a list of options separated by pipes (`|`) and wrapped in parentheses. So, `^N(E|W)N$` contains a branch: after going north, you must choose to go either east or west before finishing your route by going north again. By tracing out the possible routes after branching, you can determine where the doors are and, therefore, where the rooms are in the facility.

For example, consider this regex: `^ENWWW(NEEE|SSE(EE|N))$`

This regex begins with `ENWWW`, which means that from your current position, all routes must begin by moving east, north, and then west three times, in that order. After this, there is a branch. Before you consider the branch, this is what you know about the map so far, with doors you aren't sure about marked with a `?`:

```
#?#?#?#?#
?.|.|.|.?
#?#?#?#-#
    ?X|.?
    #?#?#
```

After this point, there is `(NEEE|SSE(EE|N))`. This gives you exactly two options: `NEEE` and `SSE(EE|N)`. By following `NEEE`, the map now looks like this:

```
#?#?#?#?#
?.|.|.|.?
#-#?#?#?#
?.|.|.|.?
#?#?#?#-#
    ?X|.?
    #?#?#
```

Now, only `SSE(EE|N)` remains. Because it is in the same parenthesized group as `NEEE`, it starts from the same room `NEEE` started in. It states that starting from that point, there exist doors which will allow you to move south twice, then east; this ends up at another branch. After that, you can either move east twice or north once. This information fills in the rest of the doors:

```
#?#?#?#?#
?.|.|.|.?
#-#?#?#?#
?.|.|.|.?
#-#?#?#-#
?.?.?X|.?
#-#-#?#?#
?.|.|.|.?
#?#?#?#?#
```

Once you've followed all possible routes, you know the remaining unknown parts are all walls, producing a finished map of the facility:

```
#########
#.|.|.|.#
#-#######
#.|.|.|.#
#-#####-#
#.#.#X|.#
#-#-#####
#.|.|.|.#
#########
```

Sometimes, a list of options can have an empty option, like `(NEWS|WNSE|)`. This means that routes at this point could effectively skip the options in parentheses and move on immediately. For example, consider this regex and the corresponding map:

```
^ENNWSWW(NEWS|)SSSEEN(WNSE|)EE(SWEN|)NNN$

###########
#.|.#.|.#.#
#-###-#-#-#
#.|.|.#.#.#
#-#####-#-#
#.#.#X|.#.#
#-#-#####-#
#.#.|.|.|.#
#-###-###-#
#.|.|.#.|.#
###########
```

This regex has one main route which, at three locations, can optionally include additional detours and be valid: `(NEWS|)`, `(WNSE|)`, and `(SWEN|)`. Regardless of which option is taken, the route continues from the position it is left at after taking those steps. So, for example, this regex matches all of the following routes (and more that aren't listed here):

- `ENNWSWWSSSEENEENNN`
- `ENNWSWWNEWSSSSEENEENNN`

- `ENNWSWWNEWSSSSEENEESWENNNN`
- `ENNWSWWSSSEENWNSEEENNN`

By following the various routes the regex matches, a full map of all of the doors and rooms in the facility can be assembled.

To get a sense for the size of this facility, you'd like to determine which room is furthest from you: specifically, you would like to find the room for which the shortest path to that room would require passing through the most doors.

- In the first example (`^WNE$`), this would be the north-east corner `3` doors away.
- In the second example (`^ENWWW(NEEE|SSE(EE|N))$`), this would be the south-east corner `10` doors away.
- In the third example (`^ENNWSWW(NEWS|)SSSEEN(WNSE|)EE(SWEN|)NNN$`), this would be the north-east corner `18` doors away.

Here are a few more examples:

```
Regex: ^ESSWWN(E|NNENN(EESS(WNSE|)SSS|WWWSSSSE(SW|NNNE)))$
Furthest room requires passing 23 doors

#############
#.|.|.|.|.|.#
#-#####-###-#
#.#.|.#.#.#.#
#-#-###-#-#-#
#.#.#.|.#.|.#
#-#-#-#####-#
#.#.#.#X|.#.#
#-#-#-###-#-#
#.|.#.|.#.#.#
###-#-###-#-#
#.|.#.|.|.#.#
#############
```

```
Regex: ^WSSEESWWWNW(S|NENNEEEENN(ESSSSW(NWSW|SSEN)|WSWWN(E|WWS(E|SS))))$
Furthest room requires passing 31 doors

###############
#.|.|.|.#.|.|.#
#-###-###-#-#-#
#.|.#.|.|.#.#.#
#-#########-#-#
#.#.|.|.|.|.#.#
#-#-#########-#
#.#.#.|X#.|.#.#
###-#-###-#-#-#
#.|.#.#.|.#.|.#
#-###-#####-###
#.|.#.|.|.#.#.#
#-#-#####-#-#-#
#.#.|.|.|.#.|.#
###############
```

What is the largest number of doors you would be required to pass through to reach a room? That is, find the room for which the shortest path from your starting location to that room would require passing through the most doors; what is the fewest doors you can pass through to reach it?

Your puzzle answer was `4121`.

--- Part Two ---

Okay, so the facility is big.

How many rooms have a shortest path from your current location that pass through at least 1000 doors?

Your puzzle answer was 8636.

Both parts of this puzzle are complete! They provide two gold stars: **

At this point, you should return to your advent calendar and try another puzzle.

If you still want to see it, you can get your puzzle input.

You can also [Share] this puzzle.