

--- Day 20: Trench Map ---

With the scanners fully deployed, you turn their attention to mapping the floor of the ocean trench.

When you get back the image from the scanners, it seems to just be random noise. Perhaps you can combine an image enhancement algorithm and the input image (your puzzle input) to clean it up a little.

For example:

```
..#.#..#####.#.#.#.###.##.....##.##.##.###.#####.#####.#....#.#..#.#..#.#
#..#####.###...#####..#..#####.##.#.#.#####...##.#.#..#.#.#.....#.#.###
.#####.###.#####...#.#.#.##..#..#..#####.....#.#....###..#.#.##.....#.....#
.#..#...##..#...###.#####.#####.###.#.#...#.....#..#.#.#...#####.##.#.....
.#..#...##.#.##..#...##.#.##..###.#.....#.#.....#.#.#.#####.###.##...#..
...####.#...#.#.##.#.....##..#.#####....##...##...#...#.....#.#.....#.....
..##..#####..#...#.#.#...##..#.#..###.#####.....#..#####.....#..#

#..#
#....
##..#
..#..
..###
```

The first section is the image enhancement algorithm. It is normally given on a single line, but it has been wrapped to multiple lines in this example for legibility. The second section is the input image, a two-dimensional grid of light pixels (#) and dark pixels (.)

The image enhancement algorithm describes how to enhance an image by simultaneously converting all pixels in the input image into an output image. Each pixel of the output image is determined by looking at a 3x3 square of pixels centered on the corresponding input image pixel. So, to determine the value of the pixel at (5,10) in the output image, nine pixels from the input image need to be considered: (4,9), (4,10), (4,11), (5,9), (5,10), (5,11), (6,9), (6,10), and (6,11). These nine input pixels are combined into a single binary number that is used as an index in the image enhancement algorithm string.

For example, to determine the output pixel that corresponds to the very middle pixel of the input image, the nine pixels marked by [...] would need to be considered:

```
# . . # .
#[. . .].
#[# . .]#
.[. # .].
. . # # #
```

Starting from the top-left and reading across each row, these pixels are [...], then [#..], then [#.]; combining these forms ...#...#. By turning dark pixels (.) into 0 and light pixels (#) into 1, the binary number 000100010 can be formed, which is 34 in decimal.

The image enhancement algorithm string is exactly 512 characters long, enough to match every possible 9-bit binary number. The first few characters of the string (numbered starting from zero) are as follows:

0	10	20	30	34	40	50	60	70
..#.#..#####.#.#.#.###.##.....##.##.###.#####.#####.#....#.#..#.#..#.#								

In the middle of this first group of characters, the character at index 34 can be found: #. So, the output pixel in the center of the output image

Our [sponsors](#) help make Advent of Code possible:

[1Password](#) - The world's most-loved password manager





Truly incredible - now the small details are really starting to come through. After enhancing the original input image twice, `35` pixels are lit.

Start with the original input image and apply the image enhancement algorithm twice, being careful to account for the infinite size of the images. How many pixels are lit in the resulting image?

Your puzzle answer was `5884`.

--- Part Two ---

You still can't quite make out the details in the image. Maybe you just didn't `enhance` it enough.

If you enhance the starting input image in the above example a total of `50` times, `3351` pixels are lit in the final output image.

Start again with the original input image and apply the image enhancement algorithm `50` times. How many pixels are lit in the resulting image?

Your puzzle answer was `19043`.

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.