Advent of Code   [About] [AoC++] [Events] [Settings] [Log Out]   bildzeitung 28*
        y(2017)   [Calendar] [Leaderboard] [Stats] [Sponsors]

--- Day 14: Disk Defragmentation ---

Suddenly, a scheduled job activates the system's disk defragmenter. Were
the situation different, you might sit and watch it for a while, but today,
you just don't have that kind of time. It's soaking up valuable system
resources that are needed elsewhere, and so the only option is to help it
finish its task as soon as possible.

The disk in question consists of a 128x128 grid; each square of the grid is
either free or used. On this disk, the state of the grid is tracked by the
bits in a sequence of knot hashes.

A total of 128 knot hashes are calculated, each corresponding to a single
row in the grid; each hash contains 128 bits which correspond to individual
grid squares. Each bit of a hash indicates whether that square is free (0)
or used (1).

The hash inputs are a key string (your puzzle input), a dash, and a number
from 0 to 127 corresponding to the row. For example, if your key string
were flqrgnkx, then the first row would be given by the bits of the knot
hash of flqrgnkx-0, the second row from the bits of the knot hash of
flqrgnkx-1, and so on until the last row, flqrgnkx-127.

The output of a knot hash is traditionally represented by 32 hexadecimal
digits; each of these digits correspond to 4 bits, for a total of
4 * 32 = 128 bits. To convert to bits, turn each hexadecimal digit to its
equivalent binary value, high-bit first: 0 becomes 0000, 1 becomes 0001, e
becomes 1110, f becomes 1111, and so on; a hash that begins with a0c2017...
in hexadecimal would begin with 10100000110000100000000101110000... in
binary.

Continuing this process, the first 8 rows and columns for key flqrgnkx
appear as follows, using # to denote used squares, and . to denote free
ones:

```
##.#.#..--->
.#.#.#.#
....#.#.
#.#.##.#
.##.#...
##..#..#
.#...#..
##.#.##.--->
|      |
V      V
```

In this example, 8108 squares are used across the entire 128x128 grid.

Given your actual key string, how many squares are used?

Your puzzle answer was 8148.

--- Part Two ---

Now, all the defragmenter needs to know is the number of regions. A region
is a group of used squares that are all adjacent, not including diagonals.
Every used square is in exactly one region: lone used squares form their
own isolated regions, while several adjacent squares all count as a single
region.

In the example above, the following nine regions are visible, each marked
with a distinct digit:

```
11.2.3..-->
.1.2.3.4
....5.6.
7.8.55.9
.88.5...
88..5..8
.8...8..
88.8.88.-->
|       |
V       V
```

Of particular interest is the region marked 8; while it does not appear
contiguous in this small view, all of the squares marked 8 are connected
when considering the whole 128x128 grid. In total, in this example, 1242
regions are present.

How many regions are present given your key string?

Your puzzle answer was 1180.

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.

Your puzzle input was vbqugkhl.

You can also [Share] this puzzle.