

Advent of Code [About] [Events] [Shop] [Settings] [Log Out] mehaase 42\*  
 int y=2018; [Calendar] [AoC++] [Sponsors] [Leaderboard] [Stats]

--- Day 12: Subterranean Sustainability ---

The year 518 is significantly more underground than your history books implied. Either that, or you've arrived in a vast cavern network under the North Pole.

After exploring a little, you discover a long tunnel that contains a row of small pots as far as you can see to your left and right. A few of them contain plants - someone is trying to grow things in these geothermally-heated caves.

The pots are numbered, with `0` in front of you. To the left, the pots are numbered `-1`, `-2`, `-3`, and so on; to the right, `1`, `2`, `3`... Your puzzle input contains a list of pots from `0` to the right and whether they do (`#`) or do not (`.`) currently contain a plant, the initial state. (No other pots currently contain plants.) For example, an initial state of `#...##....` indicates that pots `0`, `3`, and `4` currently contain plants.

Your puzzle input also contains some notes you find on a nearby table: someone has been trying to figure out how these plants spread to nearby pots. Based on the notes, for each generation of plants, a given pot has or does not have a plant based on whether that pot (and the two pots on either side of it) had a plant in the last generation. These are written as `LLCRR => N`, where `L` are pots to the left, `C` is the current pot being considered, `R` are the pots to the right, and `N` is whether the current pot will have a plant in the next generation. For example:

- A note like `..#.. => .` means that a pot that contains a plant but with no plants within two pots of it will not have a plant in it during the next generation.
- A note like `##.## => .` means that an empty pot with two plants on each side of it will remain empty in the next generation.
- A note like `.##.# => #` means that a pot has a plant in a given generation if, in the previous generation, there were plants in that pot, the one immediately to the left, and the one two pots to the right, but not in the ones immediately to the right and two to the left.

It's not clear what these plants are for, but you're sure it's important, so you'd like to make sure the current configuration of plants is sustainable by determining what will happen after `20` generations.

For example, given the following input:

```
initial state: #..#.#...##.....###...###

...## => #
..#.. => #
.#... => #
.#.#. => #
.#.## => #
.##.. => #
.#### => #
#.#.# => #
#.### => #
##.#. => #
##.## => #
###.. => #
###.# => #
####. => #
```

For brevity, in this example, only the combinations which do produce a

Our sponsors help  
 make Advent of  
 Code possible:

Rightpoint - We  
 create impactful  
 digital  
 experiences using  
 technology

	0	1	2	3
	0	0	0	0
0:	...	...	...	...
1:	...	...	...	...
2:	...	...	...	...
3:	...	...	...	...
4:	...	...	...	...
5:	...	...	...	...
6:	...	...	...	...
7:	...	...	...	...
8:	...	...	...	...
9:	...	...	...	...
10:	...	...	...	...
11:	...	...	...	...
12:	...	...	...	...
13:	...	...	...	...
14:	...	...	...	...
15:	...	...	...	...
16:	...	...	...	...
17:	...	...	...	...
18:	...	...	...	...
19:	...	...	...	...
20:	...	...	...	...

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

You can also [\[Share\]](#) this puzzle.