

[Advent of Code](#) [\[About\]](#) [\[Events\]](#) [\[Shop\]](#) [\[Settings\]](#) [\[Log Out\]](#) (anonymous user #1056941) [8★](#)
[ay.2022](#) [\[Calendar\]](#) [\[AoC++\]](#) [\[Sponsors\]](#) [\[Leaderboard\]](#) [\[Stats\]](#)

--- Day 4: Camp Cleanup ---

Space needs to be cleared before the last supplies can be unloaded from the ships, and so several Elves have been assigned the job of cleaning up sections of the camp. Every section has a unique ID number, and each Elf is assigned a range of section IDs.

However, as some of the Elves compare their section assignments with each other, they've noticed that many of the assignments **overlap**. To try to quickly find overlaps and reduce duplicated effort, the Elves pair up and make a **big list of the section assignments for each pair** (your puzzle input).

For example, consider the following list of section assignment pairs:

```
2-4,6-8
2-3,4-5
5-7,7-9
2-8,3-7
6-6,4-6
2-6,4-8
```

For the first few pairs, this list means:

- Within the first pair of Elves, the first Elf was assigned sections **2-4** (sections **2**, **3**, and **4**), while the second Elf was assigned sections **6-8** (sections **6**, **7**, **8**).
- The Elves in the second pair were each assigned two sections.
- The Elves in the third pair were each assigned three sections: one got sections **5**, **6**, and **7**, while the other also got **7**, plus **8** and **9**.

This example list uses single-digit section IDs to make it easier to draw; your actual list might contain larger numbers. Visually, these pairs of section assignments look like this:

```
.234..... 2-4
.....678. 6-8

.23..... 2-3
...45..... 4-5

....567.. 5-7
.....789 7-9

.2345678. 2-8
..34567.. 3-7

.....6... 6-6
...456... 4-6

.23456... 2-6
...45678. 4-8
```

Some of the pairs have noticed that one of their assignments **fully contains** the other. For example, **2-8** fully contains **3-7**, and **6-6** is fully contained by **4-6**. In pairs where one assignment fully contains the other, one Elf in the pair would be exclusively cleaning sections their partner will already be cleaning, so these seem like the most in need of reconsideration. In this example, there are **2** such pairs.

In how many assignment pairs does one range fully contain the other?

Your puzzle answer was **526**.

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

[Ahrefs](#) - Work on the next generation purpose search engine, a world class crawler, and real big data. Leveraging bleeding-edge hardware and advanced programming technologies. From anywhere in the world. OCar ReasonML, Dlang C++

--- Part Two ---

It seems like there is still quite a bit of duplicate work planned. Instead, the Elves would like to know the number of pairs that **overlap at all**.

In the above example, the first two pairs (2-4,6-8) and (2-3,4-5) don't overlap, while the remaining four pairs (5-7,7-9, 2-8,3-7, 6-6,4-6, and 2-6,4-8) do overlap:

- 5-7,7-9 overlaps in a single section, 7.
- 2-8,3-7 overlaps all of the sections 3 through 7.
- 6-6,4-6 overlaps in a single section, 6.
- 2-6,4-8 overlaps in sections 4, 5, and 6.

So, in this example, the number of overlapping assignment pairs is 4.

In how many assignment pairs do the ranges overlap?

Your puzzle answer was 886.

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.