

--- Day 16: Ticket Translation ---

As you're walking to yet another connecting flight, you realize that one of the legs of your re-routed trip coming up is on a high-speed train. However, the train ticket you were given is in a language you don't understand. You should probably figure out what it says before you get to the train station after the next flight.

Unfortunately, you can't actually **read** the words on the ticket. You can, however, read the numbers, and so you figure out the **fields** these tickets **must have** and the **valid ranges** for values in those fields.

You collect the rules for ticket fields, the **numbers** on your ticket, and the **numbers** on other nearby tickets for the same train service (via the airport security cameras) together into a single document you can reference (your puzzle input).

The **rules for ticket fields** specify a list of fields that exist **somewhere** on the ticket and the **valid ranges of values** for each field. For example, a rule like `class: 1-3 or 5-7` means that one of the fields in every ticket is named `class` and can be any value in the ranges `1-3` or `5-7` (inclusive, such that `3` and `5` are both valid in this field, but `4` is not).

Each ticket is represented by a single line of comma-separated values. The values are the numbers on the ticket in the order they appear; every ticket has the same format. For example, consider this ticket:

```
-----.
| ?????: 101    ?????: 102    ??????????: 103    ????: 104 |
| ?: 301  ?: 302          ??????: 303    ????????
| ?: 401  ?: 402          ????: 403    ?????????? |
-----|
```

Here, `?` represents text in a language you don't understand. This ticket might be represented as `101,102,103,104,301,302,303,401,402,403`; of course, the actual train tickets you're looking at are **much** more complicated. In any case, you've extracted just the numbers in such a way that the first number is always the same specific field, the second number is always a different specific field, and so on - you just don't know what each position actually means!

Start by determining which tickets are **completely invalid**; these are tickets that contain values which **aren't valid** for any field. Ignore your ticket for now.

For example, suppose you have the following notes:

```
class: 1-3 or 5-7
row: 6-11 or 33-44
seat: 13-40 or 45-50

your ticket:
7,1,14

nearby tickets:
7,3,47
40,4,50
55,2,20
38,6,12
```

It doesn't matter which position corresponds to which field; you can identify **invalid nearby tickets** by considering only whether tickets contain **values that are not valid for any field**. In this example, the values on the first **nearby ticket** are all valid for at least one field. This is not true of the other three **nearby tickets**: the values `4`, `55`, and `12` are not valid for any field. Adding together all of the invalid values produces your ticket scanning error rate: `4 + 55 + 12 = 71`.

Consider the validity of the **nearby tickets** you scanned. What is your ticket scanning error rate?

To begin, [get your puzzle input](#).

Answer: [Submit]

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

Educative.io -
From CSS to System Design, gain in-demand tech skills at the speed you want. Text-based courses with live coding environments help you learn without the fluff