Finding Junctions

Confidentiality Notice:
The following document is the property of AutoCanvas. Do not share this with anyone and do not post online!

Problem statement:

We create maps in our day-to-day life, and quite often, we have to solve road network problems. Today's problem involves identifying the number of junctions, and the roads that are part of that junction.

Road types:

We have two types of Roads, **Primary Roads** and **Connecting Roads.**

- Two primary roads can never connect directly to each other
- Two connecting roads can never connect directly to each other
- A connecting road always has connection to two primary roads (not zero, not one, exactly 2)
- A connecting road can however join one primary road to itself (i.e can have two connections with the same primary road)
- A primary road, can have connections to multiple or no connecting roads at all (zero or more)

These were just some things to keep in mind, don't worry if you don't quite get the problem yet, you can always come back to it while solving the problem, which will be discussed in detail below.

Detailed description:

Imagine a road network as shown in figure 1. All the red colored roads are **primary roads**. Their ID's are also written on them (from 0 to 5).

The roads are only placed in a certain configuration, and no road is connected to any other road at the moment. Remember our rule in the "**Road types**" section, that no two primary roads can connect directly to each other?

We will have to use **connecting roads** to join these primary roads together. As we will do in our next step.

Please also note that the roads have two ends (**green and blue**). This is a key detail which will matter a lot later.

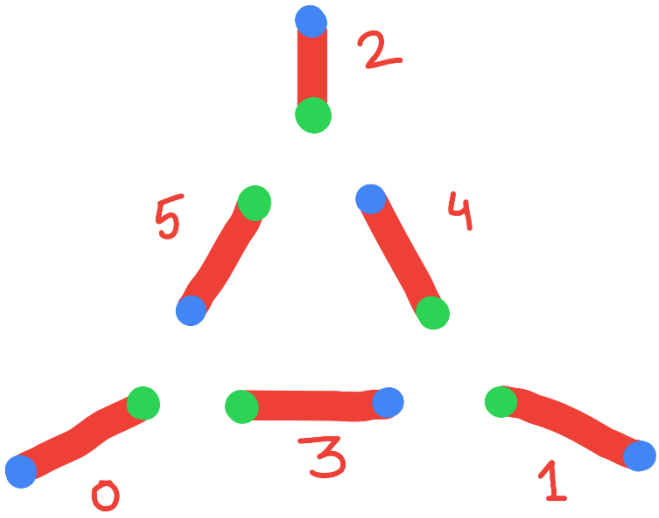

Figure 1

As you can see in the figure 2 below, the **primary roads** are now connected using **connecting roads** (colored in blue). Each of the connecting road is joining two primary roads together. **One connecting road will always join exactly two primary roads together**. The connecting roads are however allowed to join the same primary road with itself (i.e it can have two connections with the same primary road).

One more thing is that the place of connection matters. As you can see that the connecting road 6 is connected with the green end of primary road 0 and the green end of primary road 3. Similarly, the connecting road 8 is connected with the green end of primary road 0 and blue end of primary road 5.

While identifying junctions, it really matters which connecting road is connected to which side (green or blue) of a primary road.

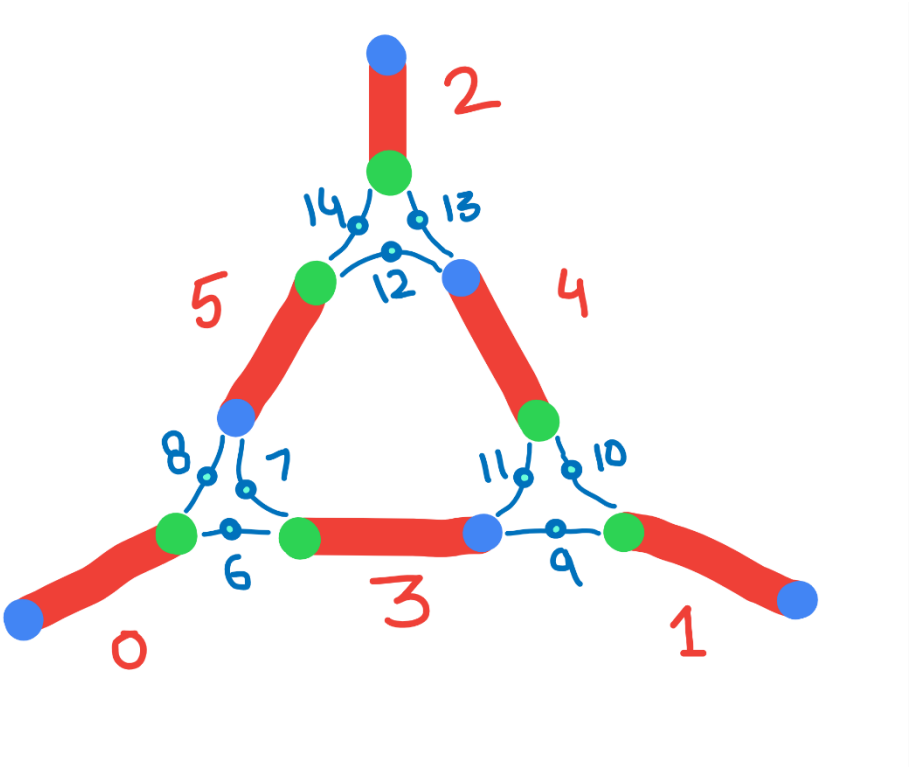

Figure 2

Let's try to imagine this problem as a graph theory problem. In fact, all the roads (both primary and connecting) are being represented by nodes in a graph. The only difference is that the primary roads can have connections of two types, green and blue. A connecting road may connect to either of the two.

Now that we have represented our problem with a graph, we can finally get to discuss what exactly we want to find out from this graph.

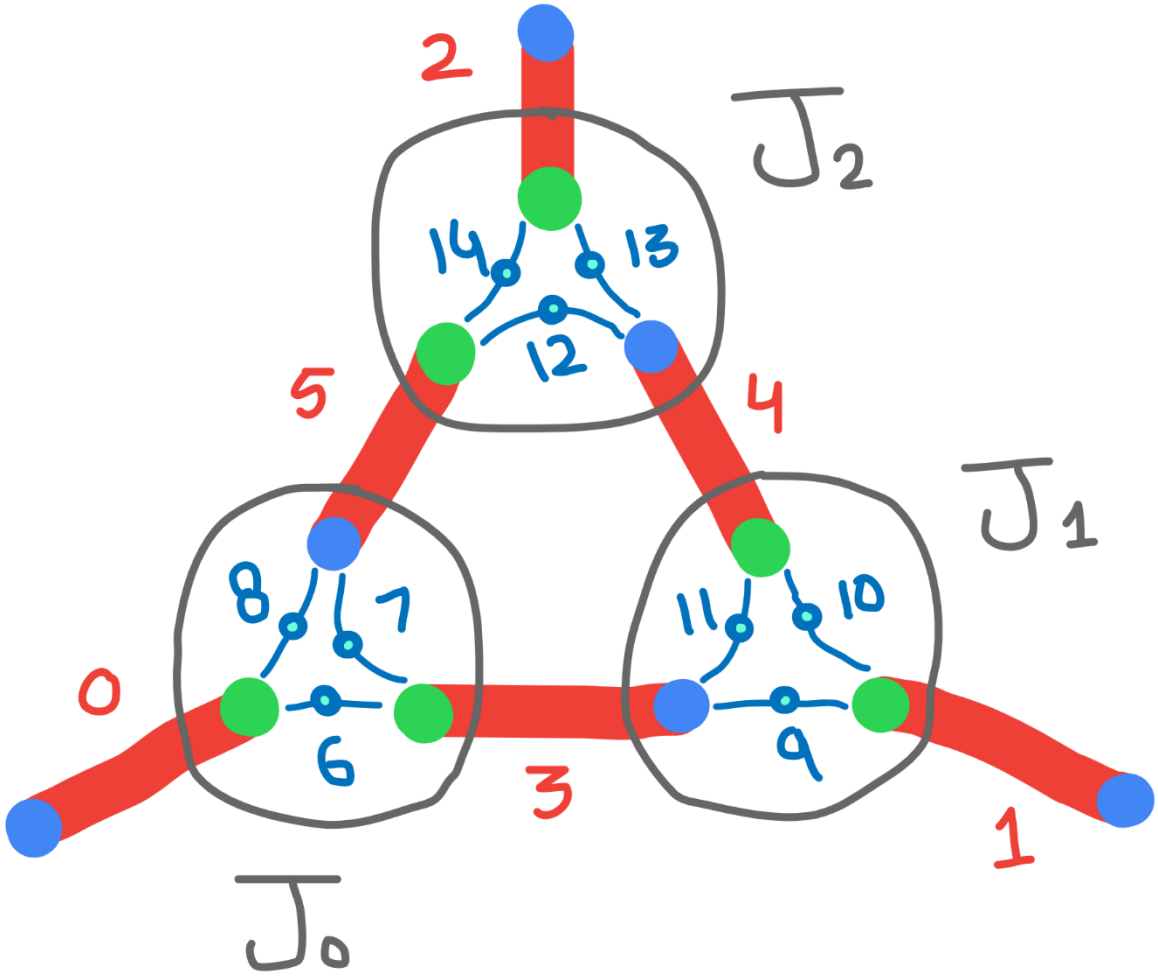

Figure 3

As you can see in the figure 3 above, some connecting roads are grouped together as a **junction**. The junctions **"J0", "J1" and "J2"** are identified in the above figure. These junctions are just lists which contain the ID's of connecting roads which are a part of it.
For example, consider the following as the final output:

- J0 = [6, 7, 8]
- J1 = [9, 10, 11]
- J2 = [12, 13, 14]

Connecting roads 6, 7 and 8 are a part of junction J0, similarly connecting roads 9, 10 and 11 are part of junction J1 and so on…

A **junction** only consists of **connecting roads**, please do not put a **primary road** in any of the junction lists.

Our end-goal is to identify **how many junctions** are there in a given network, and **which roads are part of which junction**.

Data format:

You will be provided two json files like shown below:

primary-roads-connections.json:

```
{
  "0": { "greens": [6, 8], "blues": [] },
  "1": { "greens": [9, 10], "blues": [] },
  "2": { "greens": [13, 14], "blues": [] },
  "3": { "greens": [6, 7], "blues": [9, 11] },
  "4": { "greens": [10, 11], "blues": [12, 13] },
  "5": { "greens": [12, 14], "blues": [7, 8] }
}
```

connecting-roads-connections.json:

```
{
  "6": [0, 3],
  "7": [3, 5],
  "8": [0, 5],
  "9": [3, 1],
  "10": [1, 4],
  "11": [3, 4],
  "12": [4, 5],
  "13": [2, 4],
  "14": [2, 5]
}
```

The **primary-roads-connections.json** will contain the connection information for primary roads. The dictionary key represents the **road_id**. Under each key there will be a dictionary with two **greens** and **blues** keys, each of which will hold a list of ID's. The ID's of **connecting roads**, which are connected to some particular end will be written in these lists.

The **connecting-roads-connections.json** will contain connection information for connecting roads. The dictionary key represents the **road_id**. Under each key there will be a list of exactly two elements. These two elements will the ID's of **primary roads** that this connecting road is connected to. Connecting roads always connect **exactly two primary roads**.

You can tally these json files with the network drawings in this document. These two json files pretty much sum up the description for the whole network. You may now start coding, or you can read some helpful **hints** (available on next page).

Submission Guideline:

The code you write should also work on the test cases which we have provided at the very end of this document. You are expected to submit a pdf explaining how you approached the problem, as well as, the code to solve the problem, which should be runnable at our end. The code you submit should also be generic, as we will test it on multiple edge cases on our end.

Hints:

There is always a **primary road** between two junctions i.e you cannot jump from one junction to another junction without crossing a **primary road**.

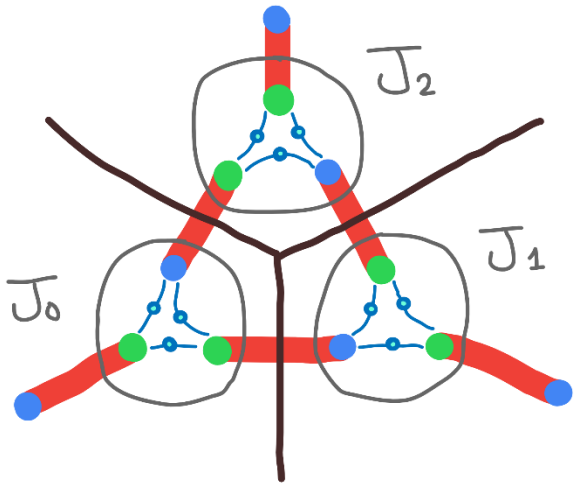

Figure 4

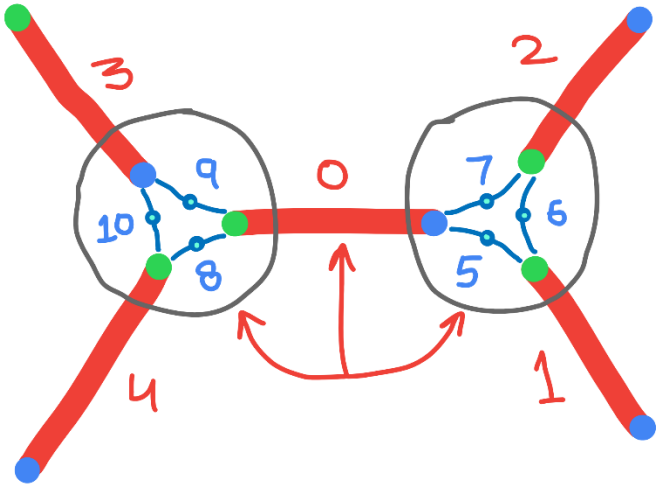

Figure 5

Figure 4 and 5 correspond to two totally different maps and they both emphasis the same fact, that if you are to jump from one junction to another, there will always be one primary road that you will have to cross.

Good Luck,

AutoCanvas.ai

Test cases are on next page!

More test cases:

Here are some additional test cases, which you can use to find out whether the algorithm you implemented is actually right or wrong.

Test Case 1:

Figure 6 shows a test case, which only has one junction. The data for this test case is in the appropriate folder.

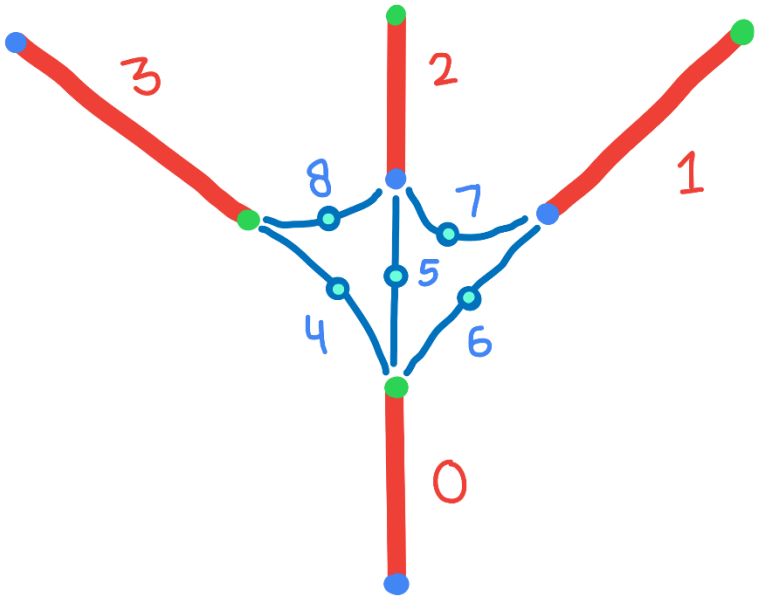

Figure 6

Test Case 2:

Figure 7 shows a test case, which has two junctions. The data will be in the respective folder.

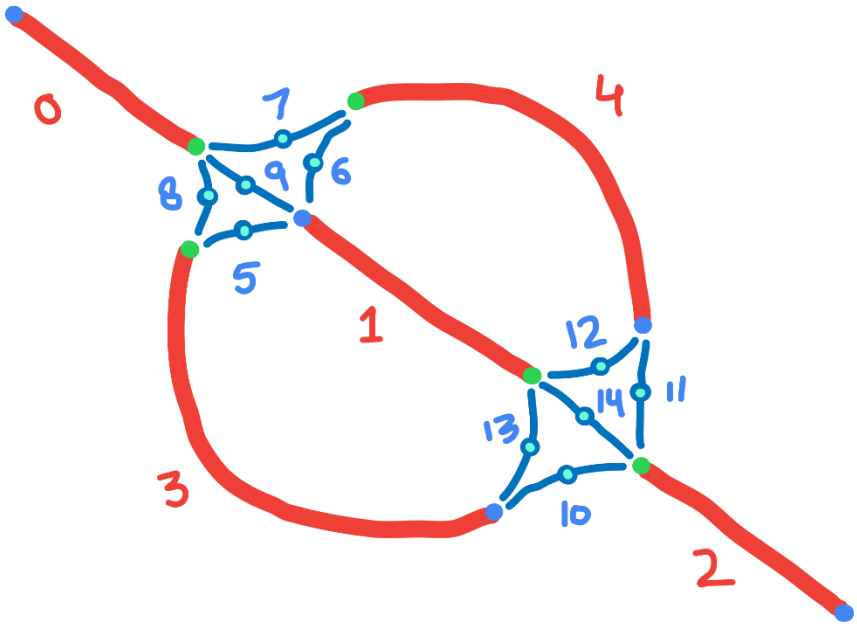

Figure 7