Suprnation

Open Book Coding Exercise

Thank you for applying for a position with the SuprNation development team. This exercise is intended to help you understand your ability to undertake research whilst delivering a solution under time pressure. This is an open-book test; you will not be penalised for using appropriate references and external libraries and this is in fact encouraged.

You will be assessed on two grounds; **fulfilling the task at hand** and **elegance of the coded solution**.

Good luck with the exercise, and we look forward to seeing your code!

1 Submission Guidelines

We are firm believers in version control system at SuprNation. We would also like to see how your solution evolves. As such, we would like you to use Git to manage your version history. Please ensure that you make regular commits of your code with descriptive message to ensure that we can understand how your solution evolved as you tackled the requirements.

The top level of your code-base should contain a file called `README.md`, which should explain how to run your code, and a file called `ASSUMPTIONS.md`, which details any assumptions you made during implementation, the reasons you made these assumptions and why they were justified.

Please send the zipped project to `mark@suprnation.com` by **31st July 2018**. Don't forget to include the `.git` directory so that we can go through the git commits.

2 Introduction

The National Radio Astronomy Observatory in New Mexico started receiving four weird periodic signals from outer space on four different antennas; `1d-1`, `1d-2`, `2d-1`, and `2d-2`. The signals seem to exhibit some patterns but Dr. Martinez assigned on the case has not been able to crack the message. What is known thus far is that two of signals carry one-dimensional information and two signals carry two-dimensional information. You have been appointed as her assistant, can you help?

The provided project contains two parts; a client skeleton (which will contain your analysis code) and a server which allows you to connect to the National Radio Astronomy Observatory antenna. Each antenna has one or two channels depending on whether a one or two dimensional signal is being transmitted.

In order to connect with the National Radio Astronomy Observatory (NodeJs server), run `npm install && node server.js`. Running the server will expose four http controllers on port 3000 and web-socket endpoint on port 3030. An HTTP endpoint is used to authenticate with an antenna (e.g. `/1d-1/request`) and obtain a `session-id`. After authentication a client needs to connect to a web-socket channel (per dimension) in order to start receiving the mysterious signal (e.g. `/<session-id>-x`, `/<session-id>-y`).

The client application contains an Angular skeleton application which allows you to select which signal you want to analyse. Each of the four signals has its own controller setup on which the analysis should be carried out. The code which requests the session and connects to the web-socket channels has been provided. Take some time to get familiar around the skeleton application. You can run the angular skeleton by executing `npm install && ng serve`

3 Task

Your task is to understand the pattern originating from each antenna (programmatically of course) and try to guess the pattern. Example: if the antenna is receiving `123451234512345...` your algorithm needs to detect the periodic pattern `12345`.

Additional, you need to display the signal being transmitted over the web socket (graphs, particles, images...) in realtime and display the status of your guess. Try to impress Dr Martinez! She needs this promotion!

In order to help you understand whether you guessed the right pattern we have provided a `guess` endpoint for each antenna (e.g. `/2d-2/guess/:session-id`). Guessing the correct pattern will disconnect the socket. ***Do not use*** *the* `guess` *endpoint to analyse the pattern but only to check your answer. No brute force senior!*

3.1 Antenna 1d-1

Signal 1 is a unique periodic one-dimensional signal with non-repeating inner patterns (there are no duplicate numbers). E.g. `123451234512345`. Your analysis should detect the pattern `12345`.

Use `/1d-1/request` to authenticate and obtain a session identifier

```
Path : /1d-1/request
Type : GET
Response (JSON): {id: session-id}
```

To receive the signal (after authenticating) connect to the web socket (socket.io) on channel `/session-id`

```
Path: /session-id
EventName: "update"
Payload (JSON): {value: some_digit}
```

Use `/1d-1/guess/:session-id` to check your answer

```
Path : '/1d-1/guess/:session-id'
Type : POST
Body (JSON): {pattern: your_pattern_as_a_string_of_digits} (E.g. "12345")
Response: true/false
```

After you have guessed correctly, try interpreting the pattern using ASCII.

3.2 Antenna 1d-2

Signal 2 is a one-dimensional signal with inner repetitions e.g. `1234534512345345`. Your analysis should detect pattern `12345345`.

Use `/1d-2/request` to authenticate and obtain a session identifier

```
Path : /1d-2/request
Type : GET
Response (JSON): {id: session-id}
```

To receive the signal (after authenticating) connect to the web socket (socket.io) on channel `/session-id`

```
Path: /session-id
EventName: "update"
Payload (JSON): {value: some_digit}
```

Use `/1d-2/guess/:session-id` to check your answer

```
Path : /1d-2/guess/:session-id
Type : POST
Body (JSON): {pattern: your_pattern_as_an_array}
(Example: [1, 2, 3, 4, 5, 3, 4, 5])
Response: true/false
```

After you have guessed correctly, try interpreting the pattern using ASCII. There is a hint which will help you interpret the 2D pattern!

3.3 Antenna 2d-1

Signal 3 is a two-dimensional signal with no repeating digits e.g. `(1,2),(2,3),(3,4),(1,2),...`. Your analysis should detect pattern `(1,2),(2,3),(3,4)`.

Use `/2d-1/request` to authenticate and obtain a session identifier

```
Path : /2d-1/request
Type : GET
Response (JSON): {id: session-id}
```

To receive the signal (after authenticating) connect to the web sockets (socket.io) `/session-id-x` and `/session-id-y`.

```
Path: /session_id-x (Example: 2F5fa8s-x, 2F5fa8s being the session-id)
EventName: "update"
Payload (JSON):  {id: some_index, value: some_digit}
```

```
Path: /session_id-y (Example: 2F5fa8s-y, 2F5fa8s being the session-id)
EventName: "update"
Payload (JSON):  {id: some_index, value: some_digit}
```

Use `/2d-1/guess/:session-id` to check your answer

```
Path : /2d-1/guess/:session-id
Type : POST
Body (JSON): {pattern: your_pattern_as_an_array_of_2d_signal_entries}
(Example: [{x:1,y:2}{x:2,y:3}{x:3,y:4}])
Response: true/false
```

After you have guessed the answer, try interpreting the x, y values. The hint is encoded in the `1d-2` signal.

3.4 Antenna 2d-2

Signal 4 is a two-dimensional signal with inner repeating digits.

E.g. `(1,2),(2,3),(2,3),(2,3),(3,4),(1,2),(2,3),(2,3),(2,3),(3,4)`. Your analysis should detect pattern `(1,2),(2,3),(2,3),(2,3),(3,4)`.

Use `/2d-2/request` to authenticate and obtain a session identifier

```
Path : /2d-2/request
Type : GET
Response (JSON): {id: session-id}
```

To receive the signal (after authenticating) connect to the web sockets (socket.io) `/session-id-x` and `/session-id-y`.

```
Path: /session_id-x (Example: 2F5fa8s-x, 2F5fa8s being the session-id)
EventName: "update"
Payload (JSON):  {id: some_index, value: some_digit}
```

```
Path: /session_id-y (Example: 2F5fa8s-y, 2F5fa8s being the session-id)
EventName: "update"
Payload (JSON):  {id: some_index, value: some_digit}
```

Use `/2d-2/guess/:session-id` to check your answer

```
Path : /2d-2/guess/:session-id
Type : POST
Body (JSON): {pattern: your_pattern_as_an_array_of_2d_signal_entries}
(Example: [{x:1,y:2},{x:2,y:3},{x:2,y:3},{x:2,y:3},{x:3,y:4}])
Response: true/false
```

After you have guessed the answer, try interpreting the x, y values. Do you see it?!