Kaan Pete Roi
Suicide Prevention and Emotional Support Helpline

Take Home Challenge

Software Development Intern

Kaan Pete Roi is Bangladesh's first ever suicide prevention and emotional support helpline where trained volunteers provide emotional support to suicidal and emotionally distressed individuals. At Kaan Pete Roi, it is crucial to keep track of our data to make data driven decisions and provide better services to our callers. In this challenge, we will look into our volunteer shift attendance data and try to find patterns using graph theory.

Kaan Pete Roi Volunteer Network

Problem Description

At Kaan Pete Roi, volunteers work in shifts. In each of their shifts, they fill out a form to keep track of their attendance. Here is an example of a simplified version of our shift attendance data (also attached as a CSV file):

Table-1

date	shift	volunteerId	volunteerName	shiftReason
05/01/2021	3pm - 6pm	146	Bobita	Regular shift
05/01/2021	9pm - 12am	13	Shabana	Make up shift
05/01/2021	3pm - 6pm	210	Rajjak	Dropping by
05/01/2021	3pm - 6pm	22	Kabori	Dropping by
05/01/2021	9pm - 12am	345	Ilias	Make up shift
06/01/2021	9pm - 12am	22	Kabori	Regular shift

06/01/2021	9pm - 12am	146	Bobita	Dropping by
07/01/2021	3pm - 6pm	345	Ilias	Regular shift
07/01/2021	3pm - 6pm	13	Shabana	Regular shift
07/01/2021	3pm - 6pm	22	Kabori	Regular shift
07/01/2021	6pm - 9pm	345	Ilias	Regular shift
07/01/2021	6pm - 9pm	210	Rajjak	Make up shift
08/01/2021	12pm - 3am	13	Shabana	Regular shift
08/01/2021	12pm - 3am	22	Kabori	Make up shift

From this data, we want to find out which of the volunteers attended the same shift on the same day. In other words, we want to find out which of the volunteers have met in the office during their shift.

We can create a network graph from this data that would tell us which of the volunteers had overlapping shifts. A network graph is a representation technique to display relation among entities (nodes) with connecting edges. You can find more about graph theory here: https://en.wikipedia.org/wiki/Graph_theory

From the above table, we can see that,

- Babita and Rajjak came on the same shift: 3 - 6pm shift on 05/01/2021
- Kabori and Shabana came on the same shift twice: 07/01/2021, 3 - 6pm shift and 12pm -3am shift, and so on.

In this scenario, the volunteers, identified by their name or ID (it is safe to assume that both names and IDs are unique) are the nodes. The connection (edge) between two volunteers is their overlapping shift, identified jointly by the date and the shift column. If we form a graph from the relations in the table-1, it would look like this:

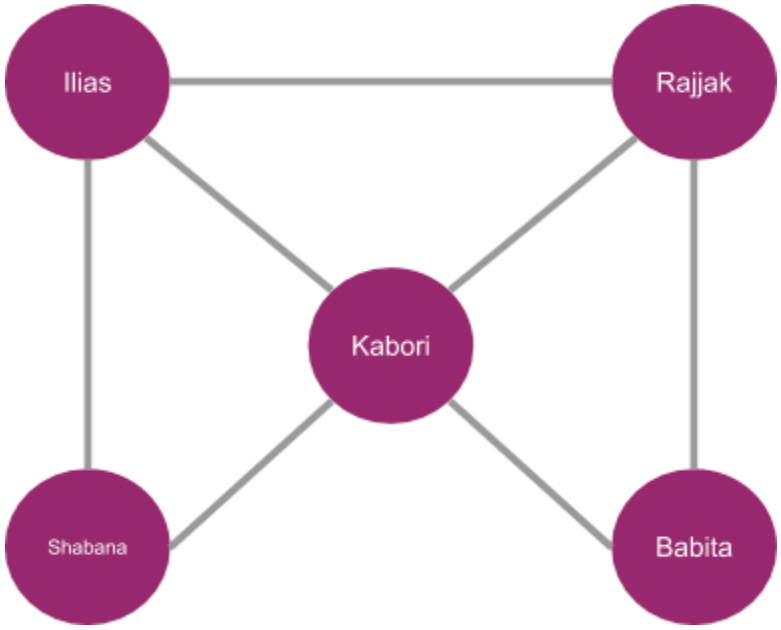

Figure-1

Your task is to find all the connections from table-1. A CSV file has been given to you that contains the data of table-1. For this challenge, you have to write a program that:

1. Reads the given CSV file
2. Finds all the nodes and their connections
3. Stores the graph representation (nodes and connections) in a suitable data structure
4. Writes the outputs in a CSV file in the following format:

Table-2

node1	**node2**
Bobita	Rajjak
Shabana	Kabori
...	

Bonus Problem

Once you find out all the connections, you will see that some volunteers had multiple overlapping shifts. We can say that those volunteers who had multiple overlapping shifts have stronger connections. In graph theory, we can represent connection strength with weights (Find more about weighted graphs here: https://mathworld.wolfram.com/WeightedGraph.html).

You will receive bonus points if you can find the weights of the connections from the given CSV file. You can add another column called "weight" in your output CSV file, which will then look like the following:

Table-3

node1	**node2**	**weight**
Bobita	Rajjak	1
Shabana	Kabori	2
...		

Solution

For this challenge, you need to write your solution using Javascript or Typescript. You need to send us:

- Your full solution in a zipped folder (even better if you can give us a public Github link)
- A separate CSV file that would contain your outputs
- A brief document on a) how you approached to solve this problem, b) what challenges you've faced and c) what are the limitations of your solution.

Please keep in mind that we need to be able to run your code in our local machine. If your code needs additional set up to run, you need to document it. We won't be able to mark your code if it does not work in Windows/MacOS machines.

You will be judged on the following criterias:

1. Usability - can we set up your code easily and run it to get the outputs?
2. Correctness - does your code solve our challenge and produce the correct outputs?
3. Code quality- how readable and clean is your code?
4. Usage of proper data structure
5. Efficiency - how long will it take your program to find the solution, especially when we have a really big dataset?
6. Testability- Is your code easily testable? **Bonus points if you write automated tests for your code.**