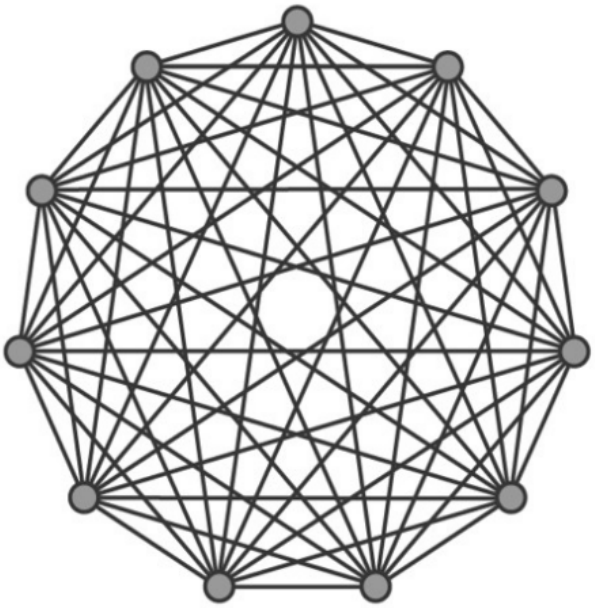
Graph project

Stable Marriage Problem



Authors : Eliot COLOMB, Pierre TEODORESCO
Teacher : Gentian JAKLLARI
Year : 2022-2023

Table of contents

I. Subject

The goal of the project is to implement a stable marriage problem solver. A stable marriage problem is a matching problem between two sets. For this project, we are working in the following context. There are "n" students applying to "m" schools. Each school has "p" seats.

If $\sum_{i}^{m} pi = n$, every student will be matched to a school.

But $\sum_{i}^{m} pi < n$, some students will not be matched to any schools.

II. Gale-Shapley algorithm

A. Algorithm

We chose to implement the Gale-Shapley algorithm. In the resources page we have included a link to the Wikipedia web page of the Gale-Shapley algorithm. We used the pseudo-code provided as a starting point for our implementation.

The algorithm has been implemented in Python. We wanted to use an "easy" language to focus ourselves more on the implementation and data structures that we needed to use.

The Algorithm takes 3 entries. Two dictionaries for students and schools preferences. These dictionaries associate names to its actual preferences. So students have a list of schools as their preferences. And schools have a list of students as their own. And another dictionary to match schools to their number of seats.

B. Data structures

For the algorithm, we choose to use a priority queue to store each “column” of the resolution table that we build in class. So for each balcony we store all its serenades inside this queue. All the queues are sorted by the priority of the associated balcony. That allows us to pop elements from the queue until we only store the amount that we want (for example: the number of seats of the school). To find out in which queue we should store the serenade, we use the balcony it likes the most and which it hasn't visited yet.

We also use a list to store all the unmatched serenades.

C. Time complexity

The upper case time complexity is $O(n \cdot m)$ where “n” is the number of students and “m” the number of schools. Or $O(n^2)$ if the number of students equals the number of schools.

D. Formal proof

The format proof is a little bit long to explain in this report, but you can find an interesting article about it in this link : article.

III. Entries format

We chose to use a JSON file as a storage for our entries. The JSON file is available on the GitHub repository linked in resources. With Python it is really simple to read a JSON file and extract from it data directly in Python dictionary shape. In the *main.py* file, we read the *test_1.json* file that contains the first test entries. We execute with it the Gale-Shapley algorithm.

The JSON format is also really simple to use. So adding new data is fast and pretty straightforward as we can store lists inside JSON.

IV. Tests and results

A. Test set number one

	MIT (2)	Harvard (1)	Stanford (1)	Caltech (2)	Oxford (1)
Alice	3,1	4,4	2,2	1,5	5,6
Bob	1,2	2,3	4,1	3,4	5,5
Charlie	1,3	5,2	3,4	4,3	2,4
Dave	2,4	3,1	1,3	5,6	4,3
Eve	2,5	4,5	5,5	3,1	1,2
Frank	4,6	1,6	3,6	2,2	5,1
Grace	3,7	5,7	2,7	1,7	4,7
Henry	1,8	2,8	4,8	5,8	3,8

Preferences table of students and schools

In rows, there are the different students, and in columns the different schools with the number of seats in brackets. The format of the values is (u,v) where u is the preference of the student and v the preference of the school.

test file name : **test_1.json**

Now the following table will introduce to you our result obtained using our implementation of the Gale-Shapley algorithm.

Schools	Student 1	Student 2
MIT (2)	Bob	Charlie
Harvard (1)	Frank	X
Stanford (1)	Dave	X
Caltech (2)	Alice	Grace
Oxford (1)	Eve	X
X	Henry	X

Results

B. Test set number two

	MIT (1)	Oxford (2)
Alice	1,3	2,2
Bob	1,1	2,1
Charlie	2,2	1,3

Preferences table of students and schools

test file name : **test_2.json**

Here are our results obtained using our implementation of the Gale-Shapley algorithm.

Schools	Student 1	Student 2
MIT (1)	Bob	X
Oxford (2)	Alice	Charlie

Results

C. Test set number three

	MIT (1)	Harvard (1)	Stanford (1)	Caltech (1)
Alice	1,1	2,1	4,2	3,3
Bob	1,2	2,3	4,1	3,1
Charlie	1,3	3,2	2,3	4,2

Preferences table of students and schools

test file name : **test_3.json**

Here are our results obtained using our implementation of the Gale-Shapley algorithm.

Schools	Student 1
MIT (1)	Alice
Harvard (1)	Bob
Stanford (1)	Charlie
Caltech (1)	X

Results

V. How to use

In the project directory:

python3 main.py tests/test_<number of test file>.json –studentBiding=<True/False>

If the –studentBiding is set to True, the algorithm will be run with the students proposing to the schools. Otherwise, the school will propose.

VI. Conclusion

We are satisfied with the final results. We think that we did a good job. If there is a thing that we wouldn't do the same: at start we have hesitated for a while before starting to write code, try a piece of the algorithm. Now, we think that we should have begun quicker to have a first result sooner.

VII. Resources

GitHub repository:
https://github.com/pierre-teodoresco/projet-graphes

Wikipedia: Gale-Shapley algorithm
https://en.wikipedia.org/wiki/Gale%E2%80%93Shapley_algorithm