Implementing the Gale-Shapley Algorithm

INF421

2015–2016

In this DM, we ask you to implement the Gale-Shapley algorithm in Java. You will write a class `StableMatching` (in the file `StableMatching.java`.) The class must implement the interface `StableMatchingInterface`, whose definition is given to you in the file `StableMatchingInterface.java`. In the ZIP archive provided to you, you will find this interface file as well as other files for testing your code (in the `src` subdirectory), but the file `StableMatching.java` is absent. Your goal is to create and complete this file and submit it to Moodle. **Note : Do not modify or submit any other file.**

For your submission to be considered acceptable, we emphasize two requirements :

1. **Your code must be accepted by the Java compiler.** There should be no compilation error. To compile the code, you can use Eclipse, or you can use the command-line : just run `make` in the `src` directory.
2. **Your code must pass all the tests provided.** We have provided you with some files (`StableMatchingTest.java`, `Main.java`, etc.) whose job it is to run a series of tests on your code. Your code must pass all these tests. To run the test-suite, you can execute the `Main.java` program from Eclipse, or you can execute `make test` from the command-line.

Some details

The test-suite will evaluate your `StableMatching` class over a number of instances of the stable marriage problem, both small instances and large. Some of these instances were pseudo-randomly generated ; others were chosen to test for special corner cases. All of these instances should be solved by your algorithm. We will verify that your code terminates (within a reasonable number of seconds), that it does not throw an exception, and that it produces a valid stable matching solution.

If everything works as expected, executing the test-suite should print out a series of messages that looks something like the following :

```
n = 25: running...
Elapsed time: 0 milliseconds
SUCCESS!

n = 26: running...
Elapsed time: 0 milliseconds
SUCCESS!
```

(Note that the time taken by your program may differ from the messages above.)

However, if a test fails, you will see a message that lists input that was tested, the result that your code produced, and why this result was unacceptable. For example, you may see :

```
n = 2: running...
Elapsed time: 0 milliseconds
FAILURE: NOT A STABLE MATCHING!
The pair formed by the man 1 and the woman 0 is unstable.
Indeed, man 1 prefers woman 0 to his bride 1
and woman 0 prefers man 1 to her groom 0.
The parameters of this test run were:
n = 2
menPrefs =
0 prefers: [0, 1]
1 prefers: [0, 1]
womenPrefs =
0 prefers: [1, 0]
1 prefers: [1, 0]
The result of this test run was:
bride =
[0, 1]
```

The result of running the full test-suite may be quite long, and it may be useful to save this result in a file that you can study in more detail. If you use the command-line, executing `make test > log.txt` will save the output of the tests in a file called `log.txt`. To achieve the same result with Eclipse, you should first click on the menu `Project`, then click on `Properties`, then on `Run/Debug Settings`, select a « run configurations », then click on the button `Edit`, choose the tab `Common` and choose to send the (« *standard output* ») not to the « console » but instead to a file of your choice (e.g. `log.txt`).

In some cases, you may want the execution of the test-suite to stop at the first failure. You can obtain this behavior by setting the constant `STOP` in the file `StableMatchingTest.java`.

General Advice

Do not be tempted to prematurely optimize your code. Your code must, most importantly, be clear and correct. Its absolute speed is not very important,provided that it has good asymptotic complexity, that is $O(n^2)$.

If you use Eclipse, you may have to change the « run configurations » to increase the heap size (because our tests take a lot of memory.) To do this, go to « run configurations » (as described above), then choose the tab `Arguments` and insert the phrase `-Xmx2G` in the field titled `VM arguments`. This increases the memory available to your program to 2GB.

We don't require you to comment your code, but it will definitely be useful to you and to us if you insert comments that explain the functioning of the algorithm.

You can also insert assertions into your code using the `assert` statement. The cost of doing so is minimal, but you will find that it can help you detect coding errors more easily. In Eclipse, the `assert` statement is deactivated by default. To enable it, follow the steps described above to access the `Arguments` tab in « run configurations » and add the phrase `-ea` to the `VM arguments` field.

This DM is meant to be completed individually. You are, of course, free to study and discuss the principles of the Gale-Shapley algorithm in groups. However, we require that you write your code by yourself. Plagiarism will be detected !