EE360C: Algorithms
University of Texas at Austin — Programming Assignment #1
Dr. Evdokia Nikolova & Dr. David Soloveichik — Due: Tuesday, October 03, 2017 11:59pm

---

Programming Assignment #1

Programming assignments are to be done individually. You may discuss the problem and general concepts with other students, but there should be no sharing of code. You may not submit code other than that which you write yourself or is provided with the assignment. This restriction specifically prohibits downloading code from the Internet. If any code you submit is in violation of this policy, you will receive no credit for the entire assignment.

The goals of this lab are:

- Familiarize you with programming in Java
- Show an application of the stable matching problem

Problem Description

In this problem we will consider a version of the problem for professors and students and their fully ordered list of preferences. Please read through this document and the documentation in the starter code thoroughly before beginning.

The Stable Matching Problem, as discussed in the text, assumes that all men and women have a fully ordered list of preferences. In this problem we will consider a version of the problem for professors and students and their fully ordered list of preferences. Note that ties in preference lists are not allowed. As before we have a set P of n professors and a set S of n students. Assume each professor and each student ranks the members of the opposite group.

Part 1: Implement a Brute Force Solution [30 points]

A brute force solution to this problem involves generating all possible permutations of men and women, and checking whether each one is a stable matching, until a stable matching is found. For this assignment, you are provided with `Preferences.java` class which includes the necessarry input structures for the problem. **Please see the comments in Preferences.java file for details.** You are also given `Assignment1.java` file where you will put your implementation for this part under `stableMatchBruteForce()` function which returns `ArrayList<Integer>`. This ArrayList should contain information for matching information representing the index of student matched with a professor, -1 is not matched. For example, if ith element of the returned ArrayList is j, than professor i is matched with student j. There might be a stable matching which is neither professor nor student optimal. This is because in a brute force, you are trying to find all possible stable matches.

Part 2: Implement Gale-Shapley Algorithm [40 points]

In order to solve this matching problem more efficiently, you need to implement Gale-Shapley Algorithm and give a solution for Professors Optimal Matching. For implementation, we provide you with again with `Preferences.java` and you will put your implementation to `Assignment1.java` file under `stableMatchGaleShapley()`. This algorithm is discussed in the class, so you can use

lecture slides and textbook for reference. This function will again return ArrayList, described in previous part. Keep note that your algorithm should run in $O(n^2)$ time. Make sure that a student can compare between his current professor and another professor in constant time.

Part 3: Matching with Costs [30 points]
In this part, in addition to the previous part, we will be having costs for the stable matching algorithm. What this means is that there are going to be two stable matching combinations (one will be Professors Optimal and one will be Students Optimal). The way we calculate the costs is as follows.

Lets take the example of a professor. Suppose his preference list is $\{s_1, s_2, s_3, s_4, s_5\}$. If the final pair comes to be (p_1,s_4), then cost of this for professor is going to be 3. (4-1=3). Similarly suppose if the preference list of student 4 was $\{p_3, p_2, p_1, p_4, p_5\}$, then his cost will be 2. Thus the cost of a pair to someone is the difference in rank of his most preferred choice and rank of the person currently assigned to him.

Now, you need to apply the stable matching algorithm for both professors optimal and students optimal. The output will be the stable pairs and their costs. Your output should look like this (p_1,s_4,3,1) where this result shows index or professor, index of student, cost to professor and cost to student respectively. You should put your implementation to same file`(Assignment1.java)` under `stableMatchCosts()` function. For output, you are provided with `Cost.java` class and the function should return an ArrayList of Costs. Your input to the function will be same as previous part, an instance of Preferences class.

Instructions

- Download and import the code into your favorite development environment. We will be grading in Java 1.8. **It is YOUR responsibility to ensure that your solution compiles with the standard Java 1.8 configuration (JDK 8).** For most (if not all) students this should not be a problem. If you have doubts, email a TA or post your question on Piazza.
- Please make sure that the header of your `Assignment1.java` file includes your necessary information. See the sample header.
- If you do not know how to install Java or are having trouble choosing and running an IDE, post on Piazza for additional assistance or come speak to a TA.
- There are several `.java` files, but you only need to make modifications to `Assignment1.java`. **Do not modify the other files we have given you, we will be using our own when grading.**
- You must implement the methods `stableMatchBruteForce()`,`stableMatchGaleShapley()` and `stableMatchCosts()` in the file `Assignment1.java`. You may add helper methods to this file if you feel it necessary or useful.
- Make sure your program compiles on the LRC machines before you submit it.

- We will be checking programming style. A penalty of up to 10 points will be given for poor programming practices (e.g. do not name your variables foo1, foo2, int1, and int2).

What To Submit

You should submit a single `Assignment1.java` file to Canvas BEFORE 11:59pm on the day it is due.