Homework 1

Winston (Hanting) Zhang

September 9, by 11:00pm

General Instructions. The following assignment is meant to be challenging, and we anticipate that it will take most of you at least 10–15 hours to complete, so please allow yourself plenty of time to work on it. We highly recommend reading the entire assignment right away — you never know when inspiration will strike. Please provide a formal mathematical proof for all your claims, and present runtime guarantees for your algorithms using asymptotic (big-$O/\Omega/\Theta$) notation, unless stated otherwise. You may assume that all basic arithmetic operations (multiplication, subtraction, division, comparison, etc.) take constant time.

Collaboration. Please carefully check the collaboration policy on the course website. When in doubt, ask an instructor.

Consulting outside sources. Please carefully check the policy regarding outside sources on the course website. Again, when in doubt, ask an instructor.

Submission. Homework submission will be through the Gradescope system. Instructions and links have been provided through the course website and Piazza. For this homework and Homework 2, we will still allow photographs/scans of handwritten solutions, but if they are illegible, you may lose points. Starting with Homework 3, only typed solutions will be accepted, and we highly recommend producing your solutions using LaTeX (the text markup language we are also using for this assignment).

Recommended practice problems (do not hand in): KT Problems 1.2, 1.4, 1.5, 1.6

Problem 1. [3+3+2+7=15]

In class, for the Stable Matching problem, we assumed that the number of men and women were the same. This was to keep our algorithm simple, and we saw that it was not too hard to reduce more general instances to this special case, by adding dummy men/women as needed. However, one might also want to solve the problem directly, without all those dummies. To do so, we would first have to define what it means for a matching to be stable when some individuals may end up unmatched. Since we assume that everyone prefers any partner over being single, the “obvious” way to define stability now is the following:

Definition 1 (Stability)**.** *The matching M is* stable *if for each pair (m, w) of a man and woman who are not matched with each other, at least one of the following two is true: (1) m is not single and prefers his current partner w' over w, or (2) w is not single and prefers her current partner m' over m.*

Next, we can revisit the Gale-Shapley algorithm. We can definitely not run it the way we defined it in class: it only terminates when all men are matched, and that is impossible when there are more men than women. We might be tempted to just change the termination condition to the following: “while there is still both an unmatched man and an unmatched woman”, and picking a man who has not yet exhausted all proposals. We obtain the following algorithm:

Algorithm 1 Gale-Shapley with non-equal numbers of men and women

```
Start with the empty matching
while there exists both a single man and a single woman do
  Let $m$ be any single man who has not proposed to all women yet.
  $m$ proposes to the highest-ranked woman $w$ on his list he has not yet proposed to.
  if $w$ is single or prefers $m$ to her current partner $m'$ then
    $(m, w)$ become (temporarily) partnered.
    if $w$ had a previous partner $m'$ then
      $m'$ becomes single again.
  else
    Nothing happens, but $m$ will not propose to $w$ again.
Finalize the current temporary matching.
```

1. As a simple warmup, and to make sure you understood the definition, prove the following: there cannot be any stable matching M under which there is both a single man and a single woman.

   Proof. Suppose we have a single man m and single woman w. Then m prefers w over nothing, and w prefers m over nothing. Hence there is a instability. Thus any stable matching M cannot have both a single man and single woman. □

2. Prove that the modified Gale-Shapley algorithm (Algorithm 1) always terminates.

   Proof. Each man must propose to at most n women before being accepted or rejected by the last woman. Hence at most n^2 proposals can occur, after which the algorithm terminates. □

3. Show that if there are more men than women, then the algortihm may output a matching that is not stable.

   Proof. Consider the following case with men m_1, m_2 and woman w. Both m_1 and m_2 have only one preference, namely w. Suppose w prefers m_1 over m_2. We start Gale-Shapley and let m_2 propose first. Then (m_2, w) are paired and there are no more single women, so the algorithm terminates. However, this is unstable since m_1 prefers w over nothing and w prefers m_1 over m_2. □

4. Prove that if there are at least as many women as men, then the matching that the algorithm outputs is guaranteed to be stable.

   Proof. Assume for the sake of contradiction that we have an unstable matching M. Note that each man must be matched to a woman; otherwise, we would have at least one single man and woman and the algorithm would not terminate. Suppose there is an instability between (m, w) and (m', w'), where m prefers w' over w and w' prefers m over m'.

   Since $w' > w$, the algorithm dicates that man m proposes to w' before w. But since w has m' as her final parnter instead of m, it must be that $m' > m$. This is contradiction since we assumed that $m > m'$. Hence M must be stable. □

Problem 2. [2+8=10]

The definition of stability was motivated by being worried that the solution by the algorithm would be overridden by pairs comprising a woman and a man — this could lead to a chain of further overrides, until the solution has nothing to do with the one output by the algorithm any more. When we addressed this, we were worried about overrides (or "backroom deals") involving a man and woman. But we might also be concerned about backroom deals between two men (who want to swap their assigned women) or two women (who want to swap their assigned men). Going back to the example of universities, this could be two students who trade their admissions, or two universities who trade students. We will focus on instances with the same number of men and women (so we can talk about perfect matchings again), and define these new notions of stability as follows:

Definition 2 (Men-Stable, Women-Stable)**.** *Given the rankings of all men and all women. We say that a perfect matching M between n men and n women is* men-stable *if there are no two men m, m' who prefer each other's partner over their own. We say that M is* women-stable *if there are no two women w, w' who prefer each other's partner over their own.*

1. Show that there are inputs (i.e., preferences) where *no* perfect matching exists that is both men-stable and women-stable simultaneously.

   Proof. Consider the following preferences for 2 men and women: $m_1 : w_1 > w_2$, $m_2 : w_2 > w_1$, $w_1 : m_2 > m_1$, and $w_2 : m_1 > m_2$. These preferences are only men stable if we pair (m_1, w_1) and (m_2, w_2) and only women stable if we pair (m_1, w_2) and m_2, w_1. Hence no matching exists that is both men-stable and women-stable simultaneously. □

Algorithm 2 Gale-Shapley for student-stable matching

```
Start with the empty matching
while there exists a nonmatched student do
  Let $s$ be any student who has not requested for all rooms.
  $s$ proposes to the highest-ranked room $r$ on his list he has not yet requested.
  if $r$ is empty then
    $(s, r)$ become matched.
  else
    Nothing happens, but $s$ will not request $r$ again.
Finalize the current temporary matching.
```

2. Now, let's make the problem easier, and assume that only one side has preferences. Think of students and single-occupancy dorm rooms: students have preferences over dorm rooms, but dorm rooms don't actually have preferences. Give and analyze (prove correct, and analyze the running time) an algorithm

which finds a student-stable matching between students and dorm rooms. Your algorithm must run in polynomial time.

[Hint: This is much easier than Stable Matching.]

Proof. First, we claim that Algorithm 2 terminates in polynomial time. Each student must request to at most m rooms before finding a vacant room. Hence at most nm proposals can occur, after which the algorithm terminates. This is clearly polynomial in n and m.

Note that once a student is paired with a room, that matching is permanment. We look at the current matches at each iteration of the algorithm. At first, nothing is matched so the current matches are vacuously stable. Now assume that current matchings are stable up to the k^{th} iteration. It is now the $k+1^{\text{th}}$ student's turn to choose a empty room. Say the pair is (s_{k+1}, r). Then r was empty for each student before s_{k+1}, but no one chose r. Hence no student before s_{k+1} prefers r over their current room. Hence there are no instabilities involving s_{k+1}. By assumption, there are no instabilities within the first k matches, so we conclude that our matching on the $k+1^{\text{th}}$ iteration is stable. By induction, this shows that the final matching of n students is stable. □

Problem 3. [0]

Chocolate Problem: 2 chocolate bars

Chocolate problems (which we will have on some of the assignment) are extra challenging problems. Solving them has zero impact on your course grade — they are meant as challenges for students who want something a little more interesting. The reward is literally chocolate — the number of bars is given with the problem, and roughly relates with the difficulty of the problem (as perceived by David). You can work on them in groups of up to 3 and submit your solution as a group, in which case you share the chocolate. If you solve a chocolate problem, please e-mail David with the solution — do not submit it on Gradescope. Also, feel free to list preferences or dietary restrictions for/against particular types of chocolate.

In our description of Stable Matching, we had assumed that all the preference lists were given to the algorithm ahead of time. Here is a different model: the algorithm starts out knowing nothing about any preferences. It just keeps trying out perfect matchings, and learns from its mistakes. Specifically, in each round, the algorithm outputs a perfect matching of the n men and n women. If the matching is stable, the algorithm is done. Otherwise, there must be one or more instability, and of those possible instabilities, one pair of a man m and woman w who would rather be with each other complain to the algorithm. There may be other instabilities, but the algorithm only hears about one for now. From this complaint, if the algorithm had matched m with w' and w with m', it learns that m prefers w over w', and that w prefers m over m'. By doing this repeatedly, the algorithm may learn more facts, which can then be combined: for instance, if it has learned that m prefers w over w', and that m prefers w' over w'', it also knows that m prefers w over w''.

Based on all the information it has learned so far, the algorithm may now propose another perfect matching, and gets more feedback, and so on, until it gets a stable perfect matching. It then becomes an interesting question how many iterations this takes in the worst case. By "worst case", we mean that some adversary who knows the algorithm carefully chooses which instability to reveal in each iteration of the algorithm.

1. Give and analyze an algorithm that finds a stable matching in at most $O(n^3)$ iterations.[1]
2. Prove that every deterministic algorithm can be forced to take at least $\Omega(n^2)$ iterations.[2]

[1]It is possible to do this in $O(n^2 \log n)$ iterations, but (a) that uses techniques you are not familiar with, and (b) the algorithm is randomized.

[2]This also holds for randomized algorithms, but you have not learned about randomized algorithms yet.

3. I suspect that every deterministic algorithm can be forced to take at least $\Omega(n^2 \log n)$ iterations, but as of now, this is open. If you happen to have any great ideas on this, I'd love to hear them. (And of course, the chocolate does not depend on this part.)