CSCI270 Homework 1

Jacob Ma

September 8, 2022

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 0.1: 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 on page 2.

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.
2. Prove that the modified Gale-Shapley algorithm (Algorithm 1) always terminates.
3. Show that if there are more men than women, then the algortihm may output a matching that is not stable.
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.

Algorithm 1 Gale-Shapely 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.
    end if
  else
    Nothing happens, but m will not propose to w again.
  end if
end while
Finalize the current temporary matching.

Solution: Problem 1-1

Proof. If there is both a single man called m and a single women called w, lets consider the pair (m, w). Because they are both single, thus they are not matched with each other.
According to the definition of stable, for each pair (m, w) of a man and woman who are matched with each other, at least on of the following two is true:

(1) m is not single and prefers his current partner w' over w.

(2) w is not single and prefers her current partner m' over m

As for (1), m *is* single, thus, (1) could not be true. As for (2), w *is* single, thus, (2) could not be true. Thus, none of the conditions met the requirements for any matching M to be stable. □

Solution: Problem 1-2

Proof. We conduct proof by contradiction.

We assume the program does not terminate. If the program does not terminate, the exists both a single man m and a single women w.

Because the program does not terminate, the single man m must have exhausted all his proposals, and have proposed to w but failed, thus he will not propose to w again.

However, as for w, w only rejects m's proposal because w is not single and does not prefers m to her current partner m'. Because w is single now, she must have dumped m' after m 's proposal. However, according to the algorithm, w only got better partners over time and could not become single again once they have been matched. She will not dump m' because w would not prefer nobody to m'.

Thus, there exist a contradiction between that w would not be single in this case but w is actually single. □

Solution: Problem 1-3

Proof. We prove the existence by giving an example.

Let's say there are two men m_1 and m_2, and one women w_1. Both men prefer w_1 than being single, and for w_1, she ranks m_2 than m_1.

In this situation, there is one more man than woman. Let's run the algorithm starting from m_1 and w_1. Because there exist both a single man m_1 and single woman w_1, m_1 proposes to the highest-ranked woman, which is w_1, on his list he has not proposed to.

Because w_1 is single, then (m_1, w_1) become (temporarily) partnered.

After this run, only m_2 is single, and no single woman is left, so the algorithm would terminate. In this case, m_1 and w_1 would be matched and m_2 would be single.

However, this matching is not stable. Consider the pair (m_2, w_1), m_2 is single so he does not satisfy the condition that m_2 is not single and prefers his current partner over w_1. w_1 is not single, but she does not prefers her current partner m_1 over m_2. Thus, this matching is not stable.

This conclusion follows the example. □

Solution: Problem 1-4

Proof. We prove by contradiction.
Let's assume the matching calculated by this algorithm *is not* stable while there are as least as many women as men. Then, according to the definition of stable, there exist at least a pair of man m and woman w are not paired together, but does not satisfied any of the 2 conditions in the definition. In other words,

(1) m is single or m prefers w over his partner w'

(2) w is single or w prefers m over his partner m'

Let's consider by cases:

(1) m is single. This is impossible in this case while there are more women than man by 1-2. w must not be single if m is single. Thus, w prefers m over her partner m'. However, since m is single, he must have exhausted all his proposal and has proposed to w before. It means w rejects m immediately or dumped m later, and later got a worse partner m', which contradicts with the lemma that the woman only get better off while matching with others. Thus, there is a contradiction.

(2) m is not single, meaning m prefers w over his partner w'. Because m proposed according to his ranking, thus m must have proposed to w before, but either got rejected immediately or dumped later. This have two cases:

(A) w rejected m immediately, meaning that w had a better partner than m at that time. Because w would only reject m if her temporary partner at least as good m, and the final partner m' would also be at least as good as m. Thus, there exist a contradiction with the second definition of unstable (2).

(B) w dumped m. Similarly to the previous reasoning, w must would only dumped m for some man better than m, thus, m' would only be better than m, and w would not be single. This also contradicts with (2).

Thus, for both cases, there exist contradictions.
The conclusion follows the contradiction. □

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 0.2: 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.

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.]

Solution: Problem 2-1

Here is one input where no perfect matching exists that is both men-stable and women-stable simultaneously:

$$M_1 : W_1 > W_2 \qquad W_1 : M_2 > M_1$$
$$M_2 : W_2 > W_1 \qquad W_2 : M_1 > M_2$$

In this case, $M_1 and M_2$ are men, and $W_1 and W_2$ are women. There are only two possible sets of matching, consider them one by one:

(1) $\{(M_1, W_1), (M_2, W_2)\}$. In this case, the matching is not women-stable, because W_1 prefers M_2 and W_2 prefers M_1, they would love to switch.

(2) $\{(M_2, W_1), (M_1, W_2)\}$. In this case, the matching is not men-stable, because M_1 prefers W_1 and M_2 prefers W_2, they would love to switch.

Thus, there exist inputs where no perfect matching exists that is both men-stable and women-stable simultaneously.

Solution: Problem 2-2

Consider the algorithm 2. We will first prove it correct.

> *Proof.* We prove this by contradiction. Assume the algorithm is not student-stable, in other words, there exist student s_1 who assigned to dorm d_1 would love and be agreed to change his dorm with student s_2 who is assigned with d_2.
> According to the algorithm, since s_1 prefers d_2 over d_1, he must have applied for d_2 before d_1, but got rejected since he did not get to assigned to d_2 at last. According to the algorithm, the only reason he got denied by the dorm is that d_2 has already be assigned.
> There are two possibilities,
>
> (1) The dorm is assigned to another student other than s_1 and s_2.
>
> However, this is impossible because s_2 would be rejected while applying for this dorm d_2, since d_2 is already assigned to other student. Thus, this is impossible.
>
> (2) The dorm is directly assigned to s_2.
>
> This means that s_2 got to choose his dorm room beforehand s_1. Since d_1 is still available while s_2 started to apply after s_1, both d_1 and d_2 would be available to s_2. Since s_2 ranks d_1 higher than d_2, s_2 would apply to d_1 first and got assigned. The result that s_2 would assigned to d_2 contradicts with the current situation that s_2 got assigned to d_1.
>
> Thus, there exist contradiction, and the algorithm could not be not student-stable. Conclusion follows. □

Consider runtime analysis.
The algorithm has a big while loop, which goes over every student. Assume the number of student = number of dorms = n, then the big while loop goes for n times.
Inside the big while loop, each student needs to apply for every dorm on his list until got matched, this would take at most $n - i$ applications for each student, while i is the number of dorm already occupied.
Inside each application, there are only one if statement, which includes a constant time operation, taking constant time $O(1)$ to compute.
Thus, the overall time complexity would be $O(1) \cdot O(n) \cdot O(n) = O(n^2)$. More specifically, the worst-case time complexity is

$$O((n-1) + (n-2) + \ldots + 1) = O\left(\frac{n(n+1)}{2}\right) = O(n^2)$$

Algorithm 2 Algorithm Problem 2-2

Start with the empty matching
while there exists student without matched dorm rooms **do**
  Let s be any unmatched student who has not assigned to any dorm yet
  while s is still not assigned to any dorm root yet **do**
    s applied for the highest-ranked dorm d on his list he has not applied for.
    if d is not matched by other students **then**
      (s, d) become matched, s is assigned to dorm d
    else
      Nothing happens, but s will not apply to d again.
    end if
  end while
end while
Get the all the matching calculated by the algorithm.