CS 180 Problem Set 1

Max Smiley

Spring 2021

1.) We begin by proposing a preference list, and showing the algorithm runs in $\Omega(n^2)$. Let all men have the preference list

$$w_1 > w_2 > \cdots > w_{n-1} > w_n$$

and all women have the preference list

$$m_n > m_{n-1} > \cdots > m_2 > m_1$$

Applying this to the Gale-Shapley algorithm, we iterate n times (for each man). We claim that when m_i proposes, this will result in i proposals before every man who had previously proposed has a match. We prove this by induction.

Base Case: $i = 1$
On the first iteration, m_1 proposes to w_1, who accepts. This is one proposal.

Inductive Step: does $P(i-1) \Rightarrow P(i)$
On the *ith* iteration, m_i proposes to w_1, whose current partner will be m_{i-1}. w_i will reject her current partner in favor of m_i, and now m_{i-1} must make a proposal before m_{i+1} can make a proposal. However, using the induction hypothesis, this will take $i-1$ proposals, which implies the whole stage will take i proposals, when considering the initial proposal by m_i.

Thus, for n men and women, the whole matching process will take

$$\sum_{i=1}^{n} i = \frac{n(n-1)}{2}$$

This is $\Omega(n^2)$, as we can find constants c and n_o such that $\frac{n(n-1)}{2} > cn^2 \; \forall n > n_0$. For example, choose $c = \frac{1}{5}$, and $n_0 = 5$. $\frac{5 \cdot 4}{2} = 10 > \frac{25}{5} = 5$, and the former term will simply grow faster than the latter, as the constant of proportionality is greater.

Therefore, the Gale-Shapley algorithm is $\Theta(n^2)$, as it is $\Omega(n^2)$ and $O(n^2)$.

2.) We begin by proposing an algorithm, and then showing that it always results in a stable assignment. Let there be m hospitals, each with k availabilities and n students, such that $mk < n$. Suppose each hospital has a preference list of students, and each student has a preference list of hospitals.

```
while ∃ unmatched student $s$ with hospitals not yet proposed to
    propose to highest ranking hospital $h$ not yet proposed to
    if $h$ is full
        if $h$ prefers $s$ to any one of their current $k$ students
            swap $h$'s least preferred student with $s$
        else
            $s$ is rejected by that hospital
    if $h$ is not full
        $h$ accepts $s$
```

First, we make some observations about this algorithm, which will be similar to those of the regular Gale-Shapley algorithm.

- Once a hospital is full, it will never have a vacancy. This is because hospitals only lose students in the context of "switching" - i.e. they will be guaranteed a replacement anyways.
- Once a student is matched with a hospital, they will only be matched with worse and worse hospitals. This is because students propose from higher to lower priority.
- Once a hospital is filled, their worst student of the k will only get better and better. This is because hospitals only swap out students when a better one comes along.
- This algorithm will terminate with k students assigned to each of the n hospitals. If it didn't then there would be some hospital with a free spot. But, since all students must propose to all hospitals for the algorithm to end, they would have proposed to that hospital at some point, and been accepted as there was a vacancy. Using one of the previous lemmas, this hospital would never have created a vacancy after being filled, and thus this is a contradiction.

This algorithm is $O(mnk)$. This is because the algorithm iterates through n students, and each of their m hospitals. Since $mk < n$, the hospitals will be filled at some point in the algorithm, and so we will have to iterate k times through the hospital's students to figure out which student is the least preferred (so we can swap them out).

Finally, we show this algorithm will lead to no instabilities, using a proof by contradiction for each case.

Assume the first type of instability occurs. That is, for some student s assigned to hospital h, and some unassigned s', h prefers s' to s. s' applied to hospital h at some point, either before or after s applied. If it were before, then s' must have been accepted, as we know s applied later on, and was accepted. However, since s' ends up unassigned, they must have been dropped before s joined, as s would have been dropped before s' had they both been in h at the same time. However, this means there was a point when s' was the least favorable student at h, and since the hospital's worst student only increases in preferability, then s never would have been accepted thereafter. So, it must be the case that s' applied after s was accepted. In this case, s' would be guaranteed to get into h, as they have a higher preferability to s. But, if any students were to be dropped, s would have to be dropped before s', since s has a lower favorability. Therefore, we have reached a contradiction in all scenarios.

Now assume the second type of instability occurs. That is, for students s and s' assigned to h and h', respectively, h prefers s' to s, and s' prefers h to h'. If this is the case, then s' will apply to h before h', either before or after s applied. If it were before, then s' must have been accepted, as s applied later on and was accepted. However, since s' ends up unassigned, they must have been dropped before s joined, as s would have been dropped before s' had they both been in h at the same time. However, this means there was a point when s' was the least favorable student at h, and since the hospital's worst student only increases in preferability, then s never would have been accepted thereafter. So, it must be the case that s' applied after s was accepted. In this case, s' would be guaranteed to get into h, as they have a higher preferability to s. But, if any students were to be dropped, s would have to be dropped before s', since s has a lower favorability. Therefore, we have reached a contradiction in all scenarios.

3a.) The notion of a *strong instability* is the same of that in the regular Gale-Shapley algorithm. Because of this, we can simply use the regular Gale-Shapley algorithm, since we've already proved correctness, lack of strong instability, and $\Theta(n^2)$ time complexity in class/homework. The caveat is that we will artificially assign a ranking between tied preferences.

3b.) There is no algorithm that can guarantee a matching with no *weak instability*, as there exists some pairings with no possible perfect matchings (in this sense). Consider the set of two men and two women with the following preferences

$$m_1 : w_1 = w_2$$

$$m_2 : w_1 = w_2$$

$$w_1 : m_1 > m_2$$

$$w_2 : m_1 > m_2$$

There are only two matchings here, namely $S_1 = \{(m_1, w_1), (m_2, w_2)\}$ and $S_2 = \{(m_1, w_2), (m_2, w_1)\}$. S_1 produces a weak instability, as w_2 prefers m_1 to m_2, while m_1 is indifferent between the two. Similarly, S_2 produces a weak instability, as w_1 prefers m_1 to m_2, while m_1 is indifferent between the two.