Homework 2

Zicong Mo

January 24, 2018

Chapter One

5. (a)
Consider the following algorithm:

```
while there exists an unpartnered man who has not proposed to every woman:
    m = next single man
    w = highest ranking woman man has not proposed to
    if w unmarried:
        match m and w
    elif w strictly prefers m to current partner m':
        unmatch m' and w
        match m and w
    elif w equally prefers m to current partner m':
        do nothing
    else:
        do nothing
```

We show that this algorithm produces a perfect matching with no strong instability.

We show through contradiction that this algorithm produces a perfect matching. Assume that at the conclusion of the algorithm, there exists a man who does not have a partner but has proposed to every woman. Since matching is one-to-one, there must also be a woman who does not have a partner. Since an unmarried woman gets matched with the first man to propose to her, the man must not have proposed to her. But this contradicts the assumption that the unmatched man has proposed toe very woman. Therefore the algorithm produces a perfect matching.

We again use contradiction to show the algorithm produces no strong instability. Assume that at the conclusion of the algorithm, there exists a strongly unstable pair m and w. Let w' and m' be the partners that m and w end up with at the conclusion of the algorithm, respectively. Note that:
(1) Men propose to women in non-increasing order of preference, meaning that women who are proposed to later cannot be more preferred than women who are proposed to earlier. They can, however, be equally preferred.

(2) A woman who has a partner can only end up with a partner who she prefers strictly more than her original partner, since she will reject partners who she prefers equally or less than her current partner.

There are two possibilities as to why m ends up with w'.
Case 1: m never proposes to w.
By (1), men propose to women in non-increasing order of preference. Since m ends up matched with w', m proposes to w' before w, meaning that he either prefers w' more than or equally to w. However, this contradicts the assumption that m strictly prefers w to w'.
Case 2: m proposed to w.
Since m does not end up with w, at some point w rejected m, either immediately or after she was proposed to by a more favorable man. By (2), she either prefers m equally or less than her current partner m'. However, this contradicts the assumption that w strictly prefers m to m'.

Since both of the possible cases lead to a contradiction, it cannot be that m does ends up with w' over w. Therefore, the strongly unstable pair (m, w) cannot exist, so the algorithm does in fact create a perfect matching without strong instabilities.

(b)
There cannot be an algorithm that guarantees no weak instabilities for every set of preferences. Let A and B be the two women we are matching with the two men X and Y, with the following preference lists.

	X	Y
A	1	1
B	1	2

	A	B
X	1	2
Y	1	2

The two possible perfect matchings for this group are $[(A, X), (B, Y)]$ and $[(A, Y), (B, X)]$.

For the first matching, (A, Y) is a weak instability, since A prefers Y equally to X but Y prefers A strictly more than B.

For the second matching, (A, X) is a weak instability, since A prefers X equally to Y but X prefers A strictly more than B.

Therefore there cannot be a matching algorithm that guarantees no weak instabilities, since certain preference lists such as the one above are guaranteed weak instabilities regardless of how they are matched.

8. We show that switching the order of a pair on a woman's preference list can improve her partner in the Gale-Shapley algorithm. Consider the following preferences of the women A, B, C and the men X, Y, Z.

	X	Y	Z
A	1	2	3
B	2	1	3
C	1	2	3

	A	B	C
X	2	1	3
Y	1	2	3
Z	1	3	2

Step 1: X proposes to B.
Current matches: (B, X)
Step 2: Y proposes to A.
Current matches: (A, Y), (B, X)
Step 3: Z proposes to A.
Because A is already matched with Y and A prefers Y to Z, A rejects Z.
Current matches: (A, Y), (B, X)
Step 4: Z proposes to C.
Final matches: (A, Y), (B, X), (C, Z)

However, if A lies about her preferences, claiming instead that she prefers (X, Z, Y) in that order, Gale-Shapely will end up matching her with X.

	X	Y	Z
A	**1**	**3**	**2**
B	2	1	3
C	1	2	3

	A	B	C
X	2	1	3
Y	1	2	3
Z	1	3	2

Step 1: X proposes to B.
Current matches: (B, X)
Step 2: Y proposes to A.
Current matches: (A, Y), (B, X)
Step 3: Z proposes to A.
Because A lied about her preferences, claiming that she prefers Z to Y, the algorithm will unmatch A and Y to match A and Z.
Current matches: (A, Z), (B, X)
Step 4: Y proposes to B
Since B prefers Y to X, the algorithm will unmatch B and X to match B and Y.
Current matches: (A, Z), (B, Y)
Step 5: X proposes to A
Since A prefers X to Z, the algorithm will unmatch A and Z to match A and X.
Current matches: (A, X), (B, Y)
Step 6: Z proposes to C
Final matches: (A, X), (B, Y), (C, Z)

Since A was able to get matched to X by lying about the ordering of Y and Z, it is possible for a woman to get a better match by lying about her preference list.

Chapter Two

4. Exponential functions grow faster than polynomial functions, which grow faster than log functions. Since the degree of $g_3(n) = n(\log n)^3$ is lower than that of $g_4(n) = n^{4/3}$, g_4 grows faster than g_3. Since $\log n$ grows faster than any constant, g_5 follows immediately after g_4. We can compare the exponents of the remaining terms, since they all have the same base. We see similarly that $n < n^2 < 2^n$, so $g_2 < g_7 < g_6$. To determine the ranking of g_1, we compare $g_1(n)$ to the function $h_1(n) = 2^{\log n}$. Since $\sqrt{\log n} < \log n$ for $n > 2$, and we are comparing asymptotic growth rate, we have:

$$g_1(n) < h_1(n) = 2^{\log n} = n$$

Therefore $g_1(n)$ grows slower than n, and also grows slower than $g_3(n) = n(\log n)^3$.

$$\begin{aligned} g_1(n) &= 2^{\sqrt{\log n}} \\ g_3(n) &= n(\log n)^3 \\ g_4(n) &= n^{4/3} \\ g_5(n) &= n^{\log n} \\ g_2(n) &= 2^n \\ g_7(n) &= 2^{n^2} \\ g_6(n) &= 2^{2^n} \end{aligned}$$

5. (a) False. Consider $f(n) = 4$ and $g(n) = 1$. Since $\log_2(4) = 2 < \log_2(1) = 0$, we do not have $\log_2(f(n)) = O(\log_2(g(n)))$.

   (b) False. Consider $f(n) = kn$ for some constant k and $g(n) = n$. While it is true that $f(n) \leq cg(n)$ for some constant $c = 1/k$, $2^{kn} = (2^k)^n \geq 2^n$. Therefore $2^{f(n)}$ is not necessarily $O(2^{g(n)})$.

   (c) True. If $f(n)$ is $O(g(n))$, then $f(n) \leq cg(n)$. Since both $f(n)$ and $g(n)$ are positive functions, this implies $f(n)^2 \leq c^2 g(n)^2$, or $(f(n))^2 = O((g(n))^2)$.