# CSE 417 HW1

Zachary McNulty (zmcnulty : 1636402)

January 2019

## 1 Problem 1

**Q:** In every Stable Matching $S$, there is a pair of people $(m, w)$ such that $m$ and $w$ are each other's top choice.

False: Consider the group of men and women with the following preferences (left is top preference, right is lowest preference):

$$m_1 = [w_1, w_2]$$
$$m_2 = [w_2, w_1]$$
$$w_1 = [m_2, m_1]$$
$$w_2 = [m_1, m_2]$$

Clearly, any stable matching cannot contain a pair $(m_i, w_j)$ where each partner is the top preference of the other, as there exists no such pair in the first place: everyone's top choice prefers someone besides them. But, by the Stable Matching Algorithm we can find a stable matching such as the one below:

$$(m_1, w_1), (m_2, w_2)$$

The two males have their top choices, so no one would be willing to swap. Sadly, the women are left with their last choice.

## 2 Problem 2

**Q:** Consider an instance of the Stable Matching Problem where there exists a male $m$ and woman $w$ who are each other's top preferences. Then in every stable matching $S$, $(m, w) \in S$.

True. Suppose not. Then $m$ and $w$ must both be paired with some partner that is not their first choice. As $m$ is the first choice of $w$, $w$ would prefer $m$ over their current partner, and similarly $m$ would prefer $w$. Thus, the matching is not stable.

## 3 Problem 3

**Q:** There are $n$ medical students and $m$ hospitals. Each hospital has one or more open positions, and has a preference for some students over others. Each student has a preference for some hospitals over others.

To show there exists a stable matching, we will simply convert this problem into a stable matching problem. As GS showed all such problems have a stable solution, then the proof would be complete. Assume we have preference lists for both students and hospitals (i.e. where left is top preference and right lowest preference):

$$s_i = [h_1, h_2, ..., h_m]$$
$$h_j = [s_1, s_2, ..., s_n]$$

**Hospitals offer multiple positions**
To account for the fact that some hospitals offer multiple positions, in each student preference list expand each hospital preference into multiple terms, one for each opening. Since these openings are all equally valuable to the student, the order does not really matter, so just arbitrarily order them. For example, if $h_1$ has 3 spots open and $s_1$ has the following preference list:

$$s_1 = [h_1, h_2, ..., h_m] \rightarrow s_1 = [h_{11}, h_{12}, h_{13}, h_2, ..., h_m]$$

where $h_{1i}$ would have the same student preference list as $h_1$.

**Less positions than students**
To account for the surplus of students, simply add in fake positions (representing unemployment) which are at the bottom of each student's preference list. Specifically, add exactly enough fake positions such that the number of students equals the number of positions.

Now, we have a group of students and positions (including fake ones) of the same size. By the theorem of the Stable Matching Problem, there exists a stable matching. Run the Stable Matching Algorithm to find this matching: any students paired to the fake positions are simply unemployed/not assigned.

**Proof of correctness**
By the stable matching problem, any matching we get by this algorithm will be stable. Thus, there exists no $(s, h)$ and $(s', h')$ such that $s$ prefers $h'$ and $h'$ prefers $s$. Since $h$ could possibly be a fake position (i.e. $s$ is not assigned). This prevents both instability 1) and 2).

## 4 Problem 4

**Q:** Suppose your computer can compute $10^{10}$ operations a second. At the following runtimes, what is the largest $n$ that could complete under an hour?

Since we can complete $10^{10}$ operations a second and have $60^2 = 3600$ seconds to work with, simply find the largest $n$ such that $f(n) \leq 36 * 10^{12}$

- $n^2 \leq 36 * 10^{12} \rightarrow n_{max} = 6 * 10^6$

- $n^3 \leq 36 * 10^{12} \rightarrow n_{max} = 33,019$

- $100n^2 \leq 36 * 10^{12} \rightarrow n_{max} = 6 * 10^4$

- $nlog_2(n) \leq 36 * 10^{12} \rightarrow n_{max} \approx 9.06316 * 10^{11}$
- $2^n \leq 36 * 10^{12} \rightarrow n_{max} = 45$

- $2^{2^n} \leq 36 * 10^{12} \rightarrow n_{max} = 5$

## 5 Problem 5

**Q:** Arrange these functions in ascending order of run-time (fast) $\rightarrow$ (slow).

$$\sqrt{2n} \rightarrow n+10 \rightarrow n^2 log(n) \rightarrow n^{2.5} \rightarrow 10^n \rightarrow 100^n$$

None of the relations are $\Theta$ rather than $O$. The first four differ by a polynomial factor (the $log(n)$ is outgrown by the differences in any of the polynomials so it does not play a role) so they cannot be in the same complexity class and $n^{2.5}$ is certainly not in the same complexity class as the exponentials. Lastly, we show $10^n$ and $100^n$ are in different complexity classes.

To show $10^n \in O(100^n)$, simply choose $c = 1$ and $n_0 = 1$. Then clearly:

$$10^n \leq c * 100^n$$

For all $n \geq n_0$

To show $10^n \in \Omega(100^n)$:

$$c * 10^n \geq 100^n$$

$$log(c * 10^n) \geq log(100^n)$$

$$log(c) + n \geq 2n$$

$$log(c) \geq n$$

However, $log(c)$ is simply a constant. For any $c$ we choose, we can always find a natural number $n$ larger than it by the Archimedean Theorem. Thus, $10^n \notin O(100^n)$ and thus it cannot be $\Theta(100^n)$. Furthermore, this also implies the reverse cannot be true: $100^n \notin \Theta(10^n)$.

## 6 Problem 6

**Q:** Assume $f(n) \in O(g(n))$, i.e. $\exists c, n_0$ such that $f(n) \leq c * g(n)$ for $n \geq n_0$. Assume that $f(n), g(n)$ are strictly non-negative.

1) Is $log_2(f(n)) \in O(log_2(g(n)))$?

False: Consider $f(n) = 2$ and $g(n) = 1$. Clearly $f(n) \in O(g(n))$ as simply choosing $c = 2$ gives $f(n) \leq cg(n)$ for all $n$. However:

$$log_2(f(n)) \leq clog_2(g(n))$$

$$1 \leq c * 0$$

Clearly there exists no such $c$, so $log_2(f(n)) \in O(log_2(g(n)))$ is not necessarily true.

2) Is $2^{f(n)} \in O(2^{g(n)})$ ?

No. Consider $f(n) = 2n$ and $g(n) = n$. Clearly, $f(n) = O(g(n))$ as we can simply choose $c = 2$ and $n_0 = 1$ and $f(n) \leq c * g(n)$ for all $n \geq n_0$. However, below we see that $2^{f(n)} \notin O(2^{g(n)})$:

$$2^{2n} \leq c * 2^n$$

$$log_2(2^{2n}) \leq log_2(c * 2^n)$$

$$2n \leq log_2(c) + n$$

$$n \leq log_2(c)$$

By the Archimedean theorem, we can always find a natural number $n$ greater than any real number $log_2(c)$ so there exists no sufficently high $c$ that works for all sufficiently large $n$. Thus, $2^{f(n)} \notin O(2^{g(n)})$.

3) Is $f(n)^2 \in O(g(n)^2)$?

True. As $f(n) \in O(g(n))$ there exists a $c$ and $n_0$ such that for all $n \geq n_0$ :

$$f(n) \leq c * g(n)$$

Since $f(n), g(n), c \geq 0$ it holds that for all $n \geq n_0$:

$$f(n)^2 \leq c^2 g(n)^2$$

Thus, $f(n)^2 \in g(n)^2$.