Owen Fahey
CMSC 27230 Problem Set 1

Problem 1.a

The matching that results from running Gale-Shapley with *A* making proposals is (a_1, b_3), (a_2, b_2), (a_3, b_4), (a_4, b_1). The steps of this process are as follows:

Step	Proposal	Accepted or Rejected	Current Matching			
			a_1	a_2	a_3	a_4
1	a_1 ask b_1	Accepted	b_1			
2	a_2 asks b_1	Rejected	b_1			
3	a_2 asks b_2	Accepted	b_1	b_2		
4	a_3 asks b_4	Accepted	b_1	b_2	b_4	
5	a_4 asks b_4	Rejected	b_1	b_2	b_4	
6	a_4 asks b_1	Accepted		b_2	b_4	b_1
7	a_1 asks b_4	Rejected		b_2	b_4	b_1
8	a_1 asks b_3	Accepted	b_3	b_2	b_4	b_1

Problem 1.b

The matching that results from running Gale-Shapley with *b* making proposals is (a_1, b_2), (a_2, b_4), (a_3, b_1), (a_4, b_3). Every person in *B* is happier when *B* makes proposals. The steps of this process are as follows:

Step	Proposal	Accepted or Rejected	Current Matching			
			b_1	b_2	b_3	b_4
1	b_1 asks a_3	Accepted	a_3			
2	b_2 asks a_4	Accepted	a_3	a_4		
3	b_3 asks a_2	Accepted	a_3	a_4	a_2	
4	b_4 asks a_2	Accepted	a_3	a_4		a_2
5	b_3 asks a_3	Rejected	a_3	a_4		a_2
6	b_3 asks a_4	Accepted	a_3		a_4	a_2
7	b_2 asks a_3	Rejected	a_3		a_4	a_2
8	b_2 asks a_1	Accepted	a_3	a_1	a_4	a_2

Problem 1.c

Answer: There are no other stable matchings

Approach:

A. Using *Optimality of Gale-Shapley for the Proposing Group*, restrict the range of possible stable matches
B. Check if there are any valid bijections from the remaining possible combinations
C. Test any valid bijections for stability

Part A

Let M_A be the stable matching given by the Gale-Shapley algorithm when *A* makes the offers. Recall *Optimality of Gale-Shapley for the Proposing Group*: for any other stable matching M_A', for all $a_i \in A$, a_i either has the same partner in both M_A and M_A' or prefers the partner given by M_A to the partner given by M_A'. Hence, we know no stable matching can occur between a_i and b_j if a_i prefers b_j to a_i's partner in M_A assuming b_j is not a_i's partner in M_A. Hence, we can conclude that no stable matches are possible between all a_i and any red b_j in a_i's preference row:

a_1: b_1, b_4, b_3, b_2
a_2: b_1, b_2, b_4, b_3
a_3: b_4, b_1, b_2, b_3
a_4: b_4, b_1, b_3, b_2

Key:
■ = a_i's partner in M_A
■ = partners that a_i prefers partner in M_A

Let M_B be the stable matching given by the Gale-Shapley algorithm when group B makes the offers. Similarly, we can conclude that no stable matches are possible between all b_j and any red a_i in b_j's preference row.

b_1: a_3, a_4, a_1, a_2
b_2: a_4, a_3, a_1, a_2
b_3: a_2, a_3, a_4, a_1
b_4: a_2, a_3, a_1, a_4

Key
■ = a_i's partner when M_B
■ = partners that b_i prefers to their partner in M_B

Combining this information into A's preference lists, we can conclude if a stable matching between any a_i and b_j exist, b_j must have either been a_i's partner in M_A or in M_B.

a_1: b_1, b_4, b_3, b_2
a_2: b_1, b_2, b_4, b_3
a_3: b_4, b_1, b_2, b_3
a_4: b_4, b_1, b_3, b_2

Key:
■ = a_i's partner in M_A
■ = a_i's partner when M_B
■ = partners that cannot form a stable match with a_i

Part B

Enumerating all possible combinations given these restrictions:

a_1 — b_3 a_2 — b_2 a_3 — b_4 a_4 — b_1 Valid, stable	a_1 — $\underline{b_3}$ a_2 — b_2 a_3 — b_4 a_4 — $\underline{b_3}$ Invalid	a_1 — b_3 a_2 — b_2 a_3 — $\underline{b_1}$ a_4 — $\underline{b_1}$ Invalid	a_1 — $\underline{b_3}$ a_2 — b_2 a_3 — b_1 a_4 — $\underline{b_3}$ Invalid	a_1 — b_3 a_2 — $\underline{b_4}$ a_3 — $\underline{b_4}$ a_4 — b_1 Invalid	a_1 — $\underline{b_3}$ a_2 — $\underline{b_4}$ a_3 — $\underline{b_4}$ a_4 — $\underline{b_3}$ Invalid	a_1 — b_3 a_2 — b_4 a_3 — $\underline{b_1}$ a_4 — $\underline{b_1}$ Invalid	a_1 — $\underline{b_3}$ a_2 — b_4 a_3 — b_1 a_4 — $\underline{b_3}$ Invalid
a_1 — b_2 a_2 — b_4 a_3 — b_1 a_4 — b_3 Valid, stable	a_1 — b_2 a_2 — b_4 a_3 — $\underline{b_1}$ a_4 — $\underline{b_1}$ Invalid	a_1 — b_2 a_2 — $\underline{b_4}$ a_3 — $\underline{b_4}$ a_4 — b_3 Invalid	a_1 — b_2 a_2 — $\underline{b_4}$ a_3 — $\underline{b_4}$ a_4 — b_1 Invalid	a_1 — $\underline{b_2}$ a_2 — $\underline{b_2}$ a_3 — b_1 a_4 — b_3 Invalid	a_1 — $\underline{b_2}$ a_2 — $\underline{b_2}$ a_3 — $\underline{b_1}$ a_4 — $\underline{b_1}$ Invalid	a_1 — $\underline{b_2}$ a_2 — $\underline{b_2}$ a_3 — b_4 a_4 — b_3 Invalid	a_1 — $\underline{b_2}$ a_2 — $\underline{b_2}$ a_3 — b_4 a_4 — b_1 Invalid

The only valid bijections between the A and B given these restrictions are the two stable matchings we previously found by running Gale-Shapley. Hence, there are no other stable matchings.

Problem 2.a

1. $T(n) \ = n^3 + 2n^2 + 20$

   T(n) is O(n^3) because *T(n)* $\leq 4.5n^3$ $\{n \geq 2\}$

   T(n) is $\Omega(n^3)$ because *T(n)* $\geq n^3$

   Hence, *T(n)* is $\Theta(n^3)$

2. $T(n) \leq \ 10n + 3n(log(n))^5$

   As $n \rightarrow \infty$, $3n(logn)^5$ is the dominant term in the upper bound

   Hence, *T(n)* is O($n(log(n))^5$)

3. $T(n) \geq 200log(n) + 10$

   As $n \rightarrow \infty$, $200log(n)$ is the dominant term in the lower bound

   Hence, *T(n)* is $\Omega(log(n))$

4. $\frac{2^n}{5} \leq T(n) \leq 7n^2 2^n$

   T(n) is $O(n^2 2^n)$

   T(n) is $\Omega(2^n)$

5. $T(n) = n + (1 + (-1)^n)n^2 + 1000$

   If n is even then $(1 + (-1)^n) = 2$, since $(-1)^{2k} = (-1^2)^k = 1$

   If n is an odd integer, then $(1 + (-1)^n) = 0$ since $(-1)^{2k+1} = (-1)(-1^2)^k = -1$.

   If n is odd and not an integer then *T(n)* will be undefined in $\mathbb{R}^+$ as $(-1)^n$ will be a complex number

   When n is even, *T(n)* $= n + 2n^2 + 1000 \leq 12.1n^2 \{n \geq 10\}$ and $n \leq$ *T(n)* $= n + 2n^2 + 1000$

   Hence, *T(n)* is $O(n^2)$

   When n is an odd integer, *T(n)* $= n + 1000 \leq 12.1n^2 \{n \geq 10\}$* and $n \leq$ *T(n)* $= n + 1000$

   Thus *T(n)* is $O(n^2)$ and $\Omega(n)$

** This value of n is slightly above where the right side of the inequality overtakes the left*

Problem 2.b

In ascending order:

A. $O(1)$
B. $O(log(log(n)))$
C. $O(log(n))^{10}$
D. $O(2^{\sqrt{log(n)}})$
E. $O(\sqrt[3]{n}\,)$
F. $O(n)$
G. $O(nlog(n))$
H. $O(n^2)$
I. $O(log(n)^{log(n)})$
J. $O(n^{\sqrt{n}})$
K. $O(2^n)$

To arrive at these answers, I first made a guess about the likely arrangement of the runtimes. Then I divided each runtime by the subsequent runtime and took the limit of this fraction (either by hand or using Wolfram Alpha). When I was wrong, I made a switch and reevaluated. I continued this until everything was in the right place. As an example:

$$\lim_{x\to\infty} \frac{x^{\sqrt{x}}}{2^x} = 0$$

Problem 3

<u>Answer</u>*:* For $n \in \mathbb{N}$, only for $n = 1, 2$ is it possible that all matchings between $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ can be stable.

<u>Approach</u>*:*

A. Show that there for $n \geq 3$, there are no preference lists such that all matchings between A and B are stable
B. Provide example cases for $n = 1, 2$

<u>Part A</u>

Claim: If there exists an $a_i \in A$ and a $b_j \in B$ such that a_i is not b_j's last choice and b_j is not a_i's last choice then at least one matching between A and B will have an unstable match.

Proof: Let a_{last} be b_j's last choice where $a_{last} \neq a_i$ and let b_{last} be a_i's last choice where $b_{last} \neq b_j$. If a_i is matched to b_{last} and b_j is matched to a_{last}, then a_i will prefer b_j to b_{last} and b_j will prefer a_i to a_{last}. This is an unstable match because both a_i and b_j prefer each other to their current partners.

Claim: For all $n \geq 3$, all possible preference lists for people in A and B contain an $a_i \in A$ and a $b_j \in B$ such that a_i is not b_j's last choice and b_j is not a_i's last choice.

Let ★ mean the following criteria is satisfied: in the preference lists of A and B, there exists an $a_i \in A$ and a $b_j \in B$ such that a_i is not b_j's last choice and b_j is not a_i's last choice. Consider the following preference list for the people in A where $b_i \neq b_j \neq b_{last}$ and $b_{i'} \neq b_{j'} \neq b_{last'}$.

a_1: $b_i, b_j, \ldots, b_{last}$
a_2: $b_{i'}, b_{j'}, \ldots, b_{last'}$
$\vdots \qquad \ddots$
a_n: $b_{i''}, b_{j''}, \ldots, b_{last''}$

If b_i's last choice is not a_1, then ★ is met since b_i is not a_1's last choice. Similarly, if b_j's last choice is not a_1, then ★ is met since b_j is not a_1's last choice. If b_i's and b_j's last choice is a_1, then their last choice cannot be a_2. While a_2's last choice could be b_j or b_i, it cannot be both, thus ★ is met. Therefore, in all cases, ★ is met.

This only works when $n \geq 3$ because there must be a unique b_i, b_j, and b_{last}.

For the sake of intuition, another way to think about this is in order for ★ not to be met, every a_i would either have to be every b_j's last choice or b_j would have to be a_i's last choice. However, because there are only $2n$ possible last slots and n^2 relationships, the demands of ★ quickly becomes infeasible.

∴ For $n \geq 3$, all possible preference lists for the people in A and B contain an $a_i \in A$ and a $b_j \in B$ such that a_i is not b_j's last choice and b_j is not a_i's last choice.

∴ For $n \geq 3$, for all possible preference lists for the people in A and B, at least one matching between A and B will have an unstable match.

∴ For $n \geq 3$, there are no possible matchings between A and B such that all matchings are stable.

<u>Part B</u>

n=1:

All possible matchings are stable for $n = 1$, since there is only one possible way to construct preference lists and only one possible matching. Neither person can prefer anyone else over their partner since there is no one else.

n=2:

It is possible to construct a preference list for $n = 2$ such that all possible matchings between A and B are stable. Consider the following preference list:

A's Preferences a_1: b_1, b_2 a_2: b_2, b_1	B's Preferences b_1: a_2, a_1 b_2: a_1, a_2

There are two possible matchings:

Case 1 a_1 — b_1 a_2 — b_2	Case 2 a_1 — b_2 a_2 — b_1

Both matchings are stable. In Case 1, neither a_1 and a_2 would not switch as they are both matched with their first choice. Likewise, in Case 2, both b_1 and b_2 would not switch as they are matched with their first choice. Hence, all matchings given these preference lists are stable.

Problem 4.a

Answer: It is not possible for a_i to do better by misrepresenting their preferences.

Proof

We are given the following lemma:

Lemma 0.1. Let a_i be a person in group A who is currently unmatched. If we look at chains of offers, if a_i makes an offer which fails, this does not affect the success or failure of the next offer which a_i makes. In other words, if a_i makes an offer to b_j which fails and then makes an offer to another person $b_{j'}$, this second offer succeeds if and only if it would have succeeded if a_i had ignored b_j and just made an offer to $b_{j'}$.

Setup: Let b_m be the partner a_i would be matched with by running Gale-Shapley with group A as the proposers. Suppose everybody in A except a_i has made their proposals. This means everybody is matched except for a_i and some b_u. Because the order in which proposals occur does not affect the final matching, we know that if a_i were to propose in accordance with their true preference list then they would end up with b_m. Let b_1 through b_{m-1} be all the people a_i prefers ahead of b_m (and in that order).

There are three ways in which a_i might try to misrepresent their preferences in order to gain an advantage:

(1) a_i can rank the people they prefer more than b_m at the end
(2) a_i can rank the people they prefer more than b_m in a different order
(3) a_i can rank someone they prefer less than b_m ahead of b_m in order to create a displacement that allows them to match with some they prefer more than b_m

Claim: If a_i were to propose in line with their true preferences, then all a_i's proposals, b_1 to b_{m-1} must fail in order for them to end up with b_m.

Claim: (1) does not affect the success or failure of other offers

Proof: We know from *Lemma 0.1* that if a_i asks b_m, this offer will be successful. Hence, how a_i ranks people after b_m is irrelevant.

Claim: The order in which a_i asks $\{b_1, \dots b_{m-1}\}$ makes no difference on the failure of the offers in this set.

Proof: If $\{b_1, \dots b_{m-1}\} = \{b_1\}$ then this is trivially true. Hence assume $\{b_1, \dots b_{m-1}\} = \{b_1, b_2, \dots b_{m-1}\}$ (i.e., that a_i prefers at least two people before b_m). If they were to propose to b_1 at any point after b_2, their offer to b_1 would fail. We can conclude this because we know that if a_i immediately proposes to b_1, the offer fails. Hence, we can use the biconditional in *Lemma 0.1* that any failing offers made before b_1 does not affect the success of b_1. Hence, no matter when b_1 is asked, the offer will fail. This argument can be applied to b_2 with b_1 inserted at each spot in $\{b_2, \dots b_{m-1}\}$ and so on to prove that the ordering of $\{b_1, \dots b_{m-1}\}$ does not change the failure of the offers in this set.

Claim: a_i cannot benefit by asking some b they prefer less than b_m ahead of b_m

Proof: If a_i makes an offer to some b they prefer less than b_m, either an unfavorable match or a failed match occurs. If the match is an unfavorable match, then a_i ends up worse off. If a failed match occurs, then we know from the previous claim, that the order in which failed matches occurs makes no difference on the failure of other offers.

Hence, no strategy improves a_i's match, a_i cannot improve their match by misrepresenting their preferences.

Problem 4.b

It is possible b_j to do better by misrepresenting their preferences. Consider the following two cases:

Case 1: True Preferences		
A's Preferences: a_1: b_2, b_1, b_3 a_2: b_1, b_2, b_3 a_3: b_1, b_3, b_2	B's Preferences b_1: a_1, a_2, a_3 b_2: a_2, a_1, a_3 b_3: a_3, a_2, a_1	Resulting Match a_1 — b_2 a_2 — b_1 a_3 — b_3

Case 2: b_1 modifies their list		
A's Preferences: a_1: b_2, b_1, b_3 a_2: b_1, b_2, b_3 a_3: b_1, b_3, b_2	B's Preferences b_1: a_1, a_3, a_2 b_2: a_2, a_1, a_3 b_3: a_3, a_2, a_1	Resulting Match a_1 — b_1 a_2 — b_2 a_3 — b_3

In Case 2, b_1 ends up with a_1 whom they prefer more than b_2, their partner in Case 1.

Problem 5.a

<u>Answer</u>**:** The game terminates in O(n)

Let $H_n = \{h_1, . . . , h_n\}$ be the set of heights of the people playing the game. We can define a potential function as the sum of h_i over $[1,n]$:

$$\phi(h_1, . . . , h_n) = \sum_{i=1}^{n} h_i$$

Claim: ϕ decreases by 1 every turn until the game terminates

Proof: Let j, k, and l be 3 unique players on the same level that get selected by the referee to move. After a move, the sum of the heights of j, k, and l will decrease by 1 since one of them will go up and two of them will go down. As the height of the other players stays the same as they are not involved in the turn, we can conclude the potential function must decrease by 1 until the game terminates.

Claim: All games terminate and an upper bound to this process is O(n)

Proof: When the game begins, there will be n people at $h = 1$, so the initial value of the potential function will be n. In any non-terminated game, at least 3 people need to all be at some $h \geq 1$ in order for a valid next turn to be possible. If the potential function were to ever be equal to 0, the game would necessarily be over since all players would have to be at $h = 0$. Hence, because the potential function starts at n and decreases by 1 with each turn, all games will terminate and an upper bound on this process must be O(n).

Problem 5.b

Answer: The maximum height (h_{max}) any player(s) reach in game of n people is $\Theta(log(n))$

Approach:

A. Argue $log_2(n + 1)$ is an upper bound on h_{max}
B. Argue $log_3(n) - 1$ is a lower bound on h_{max}
C. Conclude h_{max} given n is $\Theta(log(n))$

Part A:

Let us define a potential function:

$$\phi(h_1, \ldots, h_n) = \sum_{i=1}^{n} 2^{h_i}$$

Claim: ϕ is the same at the beginning and end of the game as ϕ is invariant in regards to each step of the game

Reasoning: Let h_i be the height of three people who are selected by the referee in an ongoing game.

Before a move occurs, they collectively add $2^{h_i} + 2^{h_i} + 2^{h_i}$ to the potential function. After a move occurs, they collectively add $2^{h_i+1} + 2^{h_i-1} + 2^{h_i-1}$ to the potential function. All other values in the potential function stay the same and these two sums can be rewritten to demonstrate they are equal:

$$2^{h_i} + 2^{h_i} + 2^{h_i} = 2 * 2^{h_i} + 2^1 * 2^{h_i-1} = 2^{h_i+1} + 2^{h_i-1} + 2^{h_i-1}$$

Hence, the potential function does not change after a move.

Claim: $\phi(h_1, \ldots, h_n) = 2n$

Reasoning: We know that $\phi(h_1) = 2$ since if $n = 1$, then $h_1 = 1$ at the beginning of the game and $2^1 = 2$. Each person that is added to a game increases the potential function by 2. If the number of people playing is increased by n then the potential function increases by $2n$.

Claim: $log_2(n + 1)$ is an upper bound on h_{max}

Reasoning: Consider a case where any $h_i \neq h_{max}$ is 0 and only one person occupies h_{max}. Then:

$$\phi(h_1, \ldots, h_n) = 2^{h_{max}} + \sum_{i=1}^{n-1} 2^{h_i} = 2n$$

$$(n - 1) + 2^{h_{max}} = 2n$$

$$2^{h_{max}} = n + 1$$

$$h_{max} = log_2(n+1)$$

This case maximizes h_{max} by assuming all $h_i \neq h_{max}$ have the minimum possible value. Hence, $log_2(n+1)$ is an upper bound for h_{max}.

Part B

Consider an alternate version of the game where after the referee chooses three people, one goes up a level and two are eliminated (i.e. directly sent to $h = 0$). The value of $h_{max-alt}$ in an n-person-game is a lower bound on h_{max} in an n-person-game (for the same n). The number of people who can ever reach a given level in an alternate n-person-game will either be the same or less than in a "real" n-person-game. This can be concluded from the fact that people who are eliminated cannot return to any level in the alternate game but can in the real game. Hence, less people or the same number of people will reach h_{max} in an alternate n-person-game than in a real n-person-game. Thus, $h_{max-alt}$ is a lower bound on h_{max}.

We can conclude that a lower bound on $h_{max-alt}$ is $log_3(n) - 1$. Observe that it takes a minimum of 3 people at $h = i$ for 1 person to reach $h = i + 1$, a minimum of 9 people at $h = i$ for 1 person to reach $h = i + 2$ and so on. To be precise, $h_{max-alt} = log_3(n) - mod(log_3(n), 1)$. Since $mod(log_3(n), 1)$ will always be less than 1, we can conclude $log_3(n) - 1 \leq h_{max-alt} \leq h_{max}$.

Part C

As $n \rightarrow \infty$, $log_3(n)$ becomes the dominant term in the lower bound. Thus, h_{max} in an n-person-game is $\Omega(log(n))$. As $n \rightarrow \infty$, $log_2(n)$, becomes the dominant term in the upper bound. Thus, h_{max} in an n-person-game is $O(log(n))$. Thus, h_{max} in an n-person-game is $\Theta(log(n))$.

Note: The change of base formula allows us to ignore the bases of the logs