CS 330, Fall 2020, Homework 2
Due Wednesday, September 16, 2020, 11:59 pm Eastern Time

Student: Justin DiEmmanuele
Collaborators: Shilpen Patel, George Padavick, Matthew Gilgo

Homework Guidelines

Collaboration policy Collaboration on homework problems, with the exception of programming assignments and reading quizzes, is permitted, but not encouraged. If you choose to collaborate on some problems, you are allowed to discuss each problem with at most 5 other students currently enrolled in the class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or TA.

Solution guidelines For problems that require you to provide an algorithm, you must give the following:

1. a precise description of the algorithm in English and, if helpful, pseudocode,
2. a proof of correctness,
3. an analysis of running time and space.

You may use algorithms from class as subroutines. You may also use any facts that we proved in class.

You should be as clear and concise as possible in your write-up of solutions.

A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand. Points might be subtracted for illegible handwriting and for solutions that are too long. Incorrect solutions will get from 0 to 30% of the grade, depending on how far they are from a working solution. Correct solutions with possibly minor flaws will get 70 to 100%, depending on the flaws and clarity of the write up.

1. **Stable Matching** (2-page limit, 15 points)

   (a) (**Unique Matching**) Give an algorithm that takes an instance of the stable matching problem as input and decides if there is *exactly one* stable matching for this instance. That is, the algorithm takes as input the preference lists for two groups of the same size, n (say, hospitals and residents), and outputs either "unique", or "more than one". Your algorithm should run in polynomial time (for example, $O(n^2)$ or $O(n^{4.2})$; it shouldn't require exponential time).

   Notice that proving your algorithm correct requires proving three statements:

   i. The algorithm always terminates;

   ii. If there is a unique stable matching, then the algorithm will output "Unique";

   iii. If the algorithm outputs "Unique", then there really is only one stable matching. (This is the most challenging statement of the three)

   Hint: It is helpful to read the section on "Extensions" at the end of KT Section 1.1; specifically, read and understand Claims 1.7 and 1.8.

   Solution The solution below relies on the Gale-Shapley algorithm as described in Section 1.1 of KT.

   Algorithm 1: UniqueMatching($male\_pref, female\_pref$)

   Input: $male\_pref$ is an array of real numbers shaped (n, m) where $male\_pref[n, m]$ is the nth male's mth preference.

   Input: $female\_pref$ is the same as $male\_pref$ but for the female participants.

   ```
   1 male_best_match, female_worst_match = GaleShapley(male_pref, female_pref)
   2 female_best_match, male_worst_match = GaleShapley(female_pref, male_pref)
   3 for i = 1 to n do
   4     if male_best_match[i] ≠ male_worst_match[i] then
             Output: Not Unique
   5     else
   6         # Do Nothing
   ```

   Output: Unique

   The requirements that need to be satisfied for this problem are discussed below:

   i. The algorithm always terminates;
   - Gale-Shapley Portion (lines 1-2)
     - This algorithm is proved to terminate in claim 1.3 in Section 1 of KT
   - Array comparison (starting line 3)
     - This section simply is composed of a for loop iterating from 1 to n. This will execute at most n times before terminating.

   ii. If there is a unique stable matching, then the algorithm will output "Unique";
   - I assert that a unique solution exists if the most preferable G-S matching is the same as the worst matching.
   - If this is not true the algorithm automatically outputs "Not Unique". Otherwise, "Unique" is output. This can be seen by studying line 3 on in algorithm 1.

- "Unique" can ONLY be output if the two compared arrays have every index matching.

iii. If the algorithm outputs "Unique", then there really is only one stable matching. (This is the most challenging statement of the three)
   - We know from claim 1.6 in KT the G-S algorithm returns a set of matchings that are stable and from 1.8 this stable matching has the best possible matching for the proposing party.
   - Claim 1.7 tells us that every execution of the G-S algorithm with the same input results in the same output. (the algorithm is deterministic)
   - Let s represent the satisfaction level of the proposing side. This would be a measure of how high on the preference list each matching is for the proposing group as a whole.
   - By definition any matching m must satisfy the following inequality:

$$min\,(s) \leq m \leq max\,(s)\,.$$

   - If we run G-S with the male group both proposing and being proposed to we can find both $m_1 = max\,(s)$ and $m_2 = min\,(s)$.
   - Then, all that is needed is to compare m_1 and m_2. If they are equal, the solution must be unique because there is no better and no worse solution from a satisfaction perspective from the inequality stated above.

(b) (**Lower bounds**) Prove that it is possible for the Gale-Shapley algorithm to perform $\Omega(n^2)$ offers before termination. [*Hint:* One way to do this is to describe, for each n, both a suitable input and a sequence of $\Omega(n^2)$ valid offers.]

Solution

- let m and s represent sets of men and women respectively. Both of length n.
- The men all have the same preference list:

$$w_1, w_2, w_3, \ldots, w_n.$$

- The women all have the same preference list:

$$m_n, m_{n-1}, m_{n-2}, \ldots, m_1.$$

- We can walk through proposals for n=3 to understand how many iterations are needed.
  - m_1 proposes to w_1, she accepts
  - m_2 proposes to w_1, she accepts. m_1 is free.
  - m_3 proposes to w_1, she accepts. m_2 is free.
    - **This is the first stable match as m_3 and w_1 are each others first preference**
  - m_1 proposes to w_2, she accepts.
  - m_2 proposes to w_2, she accepts. m_1 is free.
    - **This is the second stable match as m_2 only prefers w_1 to w_2 and w_2 only prefers m_3. m_3 and w_1 are currently in a stable matching.**
  - Finally, m_1 proposes to w_3, she accepts.
    - **This is the final match**
- This shows the characteristics of this preference list case. The resulting number of iterations is:

$$n + (n-1) + (n-2) + \ldots + 2 + 1 = n\frac{n-1}{2}.$$

  - Each stable matching must iterate through every available man because the last man is the one who will find a stable match.
  - Each stable match takes 1 less iteration to find since the previous iteration results in a stable match.
- In the expanded version of the equation representing the number of iterations it is clear G-S is $\Theta\left(n^2\right) = \Omega\left(n^2\right)$

$$n\frac{n-1}{2} = \frac{n^2}{2} - \frac{n}{2} - \frac{1}{2} = f\left(n\right).$$

$$g\left(n\right) = n^2.$$

$$\lim_{n\to\infty} \frac{f\left(n\right)}{g\left(n\right)} = \frac{1}{2}.$$

2. **Asymptotics Review** (1-page limit, 10 points)

Rank the following functions by their asymptotic growth rate (big-O relationships).

List functions on separate lines, where higher lines correspond to asymptotically larger functions (so $f(n)$ goes above $g(n)$ if $g(n) = O(f(n))$ and $f(n) \neq O(g(n))$). If two functions are asymptotically equivalent (that is, $f(n) = O(g(n))$ and $g(n) = O(f(n))$), then they should go on the same line.

n^{100}	$n^3 - 5n^2 - 10n + 12$	n^2	$\sqrt{n}$	2^n
n^3	$\log_2 n$	$\log_3 n$	$\sum_{i=1}^{n} i$	2.1^n
$10 + \frac{1}{n}$	$n \log_2 n$	$n \cdot 2^n$	$\binom{n}{2} - n + 2$	$\log_2(n!)$

Hint: You may want to review Big-Oh rules and logarithm properties.

Solution The functions are shown below sorted by asymptotic growth rate.

Table 1: Asymptotic Sorting

$n \cdot 2^n$		
2.1^n		
2^n		
n^{100}		
$n^3 - 5n^2 - 10n + 12$	n^3	
$\sum_{n=i}^{n} i$	n^2	$\binom{n}{2} - n + 2$
$n \log_2 n$	$\log_2 n!$	
$\sqrt{n}$		
$\log_2 n$	$\log_3 n$	
$10 + \frac{1}{n}$		