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

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**)
   ANSWER

```
Algorithm 1: UniqueMatching($mList, wList$)
Input: $mList$ is a dictionary with keys being men, and values for each key being an
       ordered list of women. $wList$ is the opposite.
$freeMQueue \leftarrow mList.keys().toQueue()$;
$matchedWDict \leftarrow \{\}$;
while $freeMQueue.length() > 0$ do
    $m \leftarrow freeMQueue.pop()$;
    $w \leftarrow mList[m][1]$;
    if $matchedWDict[w] == undefined$ then
        $matchedWDict[w] \leftarrow m$;
    else if $wList[w].indexOf(m) > wList[w].indexOf(matchedWDict[w])$ then
        $freeMQueue.push(matchedWDict[w])$;
        $matchedWDict[w] \leftarrow m$;
    else
        $freeMQueue.push(m)$;
$freeWQueue \leftarrow wList.keys().toQueue()$;
$matchedMDict \leftarrow \{\}$;
while $freeWQueue.length() > 0$ do
    $w \leftarrow freeWQueue.pop()$;
    $m \leftarrow wList[w][1]$;
    if $matchedMDict[m] == undefined$ then
        $matchedMDict[m] \leftarrow w$;
    else if $mList[m].indexOf(w) > mList[m].indexOf(matchedMDict[m])$ then
        $freeWQueue.push(matchedMDict[m])$;
        $matchedMDict[m] \leftarrow w$;
    else
        $freeWQueue.push(w)$;
$answer \leftarrow "Unique"$;
for $winmatchedWDict$ do
    if $matchedMDict[matchedWDict[w]]! = w$ then
        $answer \leftarrow "Morethanone"$;
Output: $answer$
```

Algorithm Description
In order to produce an algorithm that gives these results, we would create an algorithm that receives two dictionaries, one with man as their keys, and for each man an ordered list of women as the values, and vice versa.

Then, we'd proceed with running the Gale-Shapley algorithm with the men being the ones that propose, and women the ones who accept, and save the resulting stable-match.

With this matching saved, we'd then run the Gale-Shapley algorithm again, but with women being the ones that propose, and men the ones that accept, and we'd save the resulting stable-match.

With these two matchings saved, we could then compare the two and see if they are equal (ie: if man X is matched with woman Y in one, they should also be matched in the other), if so, we'd return "Unique", otherwise we'd return "More than one".

Runtime
This algorithm would always finish running, as it is simply 2 runs of the Galey-Shapley and a loop through the resulting answers, which we already know (due to the Galey-Shapley alg.) to be finite. With that said, we'd need 2 times the runtime of the GS alg., or $O(2 \times n^2)$, plus the final loop, which, considering that there are at most $n/2$ matches, would take $2 \times \frac{n}{2} = n$, so, in total this algorithm would have a runtime of $O(n + 2 \times n^2) = O(n(1 + 2n))$.

Correctness
As we proved before, the algorithm always finishes, as for the rest of the correctness of this algorithm, it is based on the statements 1.7 and 1.8 of our book, which, respectively, state that: "Every execution of the G-S algorithm results in the set S", with S being the set of pairs (m,best(m)): $m \in M$", and that "In the stable matching S, each woman is paired with her worst valid partner".

In essence, these two statements say that, since the men propose, they always end up with their best valid partner, while in the matching, all women end up with their worst valid partner. This means that, if we reverse the roles and have women proposing, we'll find that women will always end up with their best valid partner, and men with their worst valid partner. Meaning that if they are the same, both men and women have the best valid matches.

To further this point, lets image that this input only has a single, unique, solution, then both runs (with men and women proposing) must be equal. On the other hand, if we imagine that these runs return and equal matching S, then we know, from what is explained above, that all men and women have their best possible matching under S, so if there was another matching, say S^1, at the very least 2 men and 2 women would have better matchings, but since we know this cant happen to them when they are proposing, then we reach a contradiction.

As such, this algorithm will result in "Unique" if and only if they matching is Unique.

(b) (**Lower bounds**)

ANSWER

An example of this would be an input where each woman would prefer a different man, and the ONLY stable matching is the one where each woman is with their preferred man. This is the case because, for each woman to be with their preferred match, they first must have been engaged with every other man.

So, in theory, if the unique stable match is the one above, then every woman has already proposed to every man, meaning that there have been $woman \times man$ proposals, and since these inputs are the same size, it means there have been at least n^2 proposals. Meaning that the algorithm preformed $\Omega(n^2)$.

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.

ANSWER

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