INFO 6205 – Program Structure and Algorithms
Assignment 1

Student Name: Yuetong Guo
Professor: Nik Bear Brown

Q1 (5 Points) Arrange the following functions in increasing order of growth:

- log(55n)
- $55^n + 11^n$
- $0.99^n + 1$
- n / log(n)
- $\log(n) + n^{0.5}$
- log(log(n))
- $\log(n^3)$
- $n!\ e^n$
- √n
- 5^n

Ans:

The functions are arranged in increasing order of growth as follows:

- $0.99^n + 1$
- log(log(n))
- log(55n)
- $\log(n^3)$
- √n
- $\log(n) + n^{0.5}$
- n / log(n)
- 5^n
- $55^n + 11^n$
- $n!\ e^n$

Q2 (5 Points) Give a brief definition for the following:

i. Algorithm

ii. Computational tractability

iii. Stable Matching

iv. Big-O notation

v. Asymptotic order of growth

Ans:

i. An algorithm is a set of instructions for solving a problem or achieving a specific task. It is like a recipe for making something, but instead of making food, it helps a computer solve a problem. Just like a recipe tells you what ingredients to use and what steps to follow to make a cake, an algorithm gives the computer a set of steps to follow to solve a problem.

ii. Computational tractability refers to how easy or difficult it is for a computer to solve a problem. It's like a puzzle for computers - some puzzles are easy, and computers can solve them quickly, but other puzzles are complicated, and it takes computers more time to solve them. In short, computational tractability is about the difficulty level of solving problems with computers.

iii. Stable matching is a pairing between two sets of elements (such as people and jobs, or schools and students) in such a way that there are no two elements that would rather be paired with each other than the elements they have been matched with. In other words, a stable matching is a pairing where there are no "unhappy" elements - no one who would rather have been paired with someone else.

iv. Big-O notation is a way of expressing the upper bound on the growth rate of a function. It provides a rough estimate of how the running time of an algorithm will scale as the size of the input increases. It's used to compare the relative efficiency of algorithms, not to provide an exact measurement of their running time.

v. Asymptotic order of growth is a mathematical concept used to describe the behavior of a function as the input to the function grows larger and larger. In simpler terms, it's a way to compare how quickly different functions grow as their input becomes very large.

Q3 (10 Points)

This exercise shows that stable matchings need not exist if there are not “two sides.” Consider the following “roommate” problem. There are four people, Pat, Chris, Dana, and Leslie. They must pair off (each pair will share a two-bed suite). Each has preferences over which of the others they would prefer to have as a roommate. The preferences are:
Leslie: Pat > Chris >_ Dana
Chris: Leslie _> Pat _> Dana
Pat: Chris > Leslie> _ Dana
Dana: Chris> _ Leslie> _ Pat
Show that no stable matching exists. (That is, no matter who you put together, they will always be two potential roommates who are not matched but prefer each other to their current roommate.) give examples to justify your solution.

Ans:

In the given roommate problem, no stable matching can be found as every pairing lead to two individuals who would rather switch roommates. All three possible pairings - Pat-Chris/Leslie-Dana, Pat-Leslie/Chris-Dana, and Pat-Dana/Leslie-Chris - result in instability. For instance, when Pat is paired with Chris and Leslie with Dana, both Chris and Leslie would prefer to switch roommates, with Chris preferring Leslie over Pat, and Leslie preferring Pat over Dana. Similarly, when Pat is paired with Dana and Leslie with Chris, both

Chris and Dana would prefer to switch roommates, with Chris preferring Dana over Leslie and Dana preferring Leslie over Pat. This demonstrates that no stable matching exists in this roommate problem.

Q4 (10 Points)

Prove the following:
In the Gale-Shapley algorithm, run with n men and n women, what is the maximum number of times any woman can be proposed to?
Solution: n^2 − n + 2

Ans:

Since the (n^2 – n +2) means the count of most rounds (or you can say days of proposal) in the Stable Matching problem, which is not the above question's solution. After communicating with TA, the question's solution has been changed to n^2-n+1, and I'll prove "n^2-n+1" in below.

There are n women in the list, and in the worst situation, n woman would need to receive (n-1) proposals before finding her ideal partner. Thus, the total number of proposals would be n(n-1). Because if a man was eliminated n times, he would have no more women to propose to, but this can't happen as the algorithm terminates in a stable pairing. Therefore, there must be one woman who would just accept the last proposal, which means the last man who is not paired will still have one proposal, thus we subtract one from the total sum, resulting in n^2 - n + 1.

Q5 (10 Points)

Consider there are x number of gaming companies, each with a certain number of available positions for hiring students. There were n Computer Science and Game Design students graduating in a given year at Northeastern, each interested in joining one of the companies. Each company has a ranking of students based on their portfolio and their GPA, and each student also has a ranking of company based on the work and the pay, benefits and location. We will assume that there are more students graduating than there are available positions in m companies.
Our goal is to find a way of assigning each student to at most one company, in a way that all the available positions in a particular company are filled. (There will be some students who did not get a company as we have assumed the number of students is greater than the number of companies).
The assignment of student to a particular company is stable if neither of the situations arises:

1. There are students s1 and s2 and company c, so that
   s1 is assigned to c, and
   s2 is assigned to no company, and
   c prefers s2 to s1 based on his academics and projects.
2. There are students s1 and s2 and companies c1 and c2, so that
   s1 is assigned to c1, and
   s2 is assigned to c2, and
   c1 prefers s2 to s1, and
   s2 prefers c1 to c2

So, we basically have a stable matching problem, except that
(a) Companies generally want more than one hiring student, and

(b) There is a surplus of Computer Science and Game Design graduates.

Either:

a. Show that there is always a stable assignment of students to companies and devise an algorithm to find one. or

b. Show that an algorithm that always generates a stable assignment cannot exist.

Ans:

We can directly use the Gale-Shapley Algorithm to solve the stable matching problem between students and companies by treating companies as positions. In other words, each company is viewed as offering several positions, which can be filled by a single student. This converts the problem into a classical woman-man matching problem, where each student is viewed as a woman and each position as a man. As for the problem about graduates is more than companies, one possible improvement to the algorithm is to modify the while loop condition to only run while there are still unmatched students and available positions. This way, once all the positions have been filled, the algorithm stops, and the remaining students are unmatched.

Additionally, the algorithm can be extended to allow companies to rank their positions in terms of priority. This way, if a company has more than one position, it can offer its lower-priority positions to students who are less preferred. This can reduce the number of unmatched students and result in a more efficient matching.

The algorithm in pseudocode proceeds as follows:

```
while (some students are unmatched and some positions are available)
        take a company with available positions
        company offers a position to its top favorite student
        if (student free)
                student accepts the offer
        else
                if (student prefers this offer)
                        student accepts this offer and refuses the offer held now
                        number of available poisons of the refused company + 1
                        number of available positions of the this accepted company -1
        else
                        student refuses this offer
```

Q6 (10 Points)

A (5 points) Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.
True or false? In every instance of the Stable Matching Problem, there is a stable matching containing a pair (m, w) such that m is ranked first on the preference list of w and w is ranked first on the preference list of m.

Ans:

False. Let's see a counterexample as below:

Pre	1st	2nd	3rd
m1	w2	w3	w1
m2	w3	w1	w2
m3	w1	w2	w3
Pre	**1st**	**2nd**	**3rd**
w1	m1	m2	m3
w2	m2	m3	m1
w3	m1	m2	m3

Because in the preference ranking of the individuals in the pairing do not lead to any instability in the final paring (m1-w2, m2-w3, m3-w1), we consider the matching is stable. However, for m1 who ranked first on w1's preference list, w1 is not ranked first on m1's preference list. Likewise, m2 is ranked first on w2's preference list, but w2 is not ranked first on m2's preference list. This means that while two individuals in a pairing may not be each other's top choice, they still form a stable pairing with no blocking pairs.

B (5 points) Suppose we are given an instance of the stable matching problem for which there is a man m who is the first choice of all women. Prove or give a counterexample: In any stable matching, m must be paired with his first choice.

Ans:

True.

In a scenario where men actively pursue women in a matching, the pursuit of a woman's first choice m would result in a pairing if that woman has not already accepted another man. If the woman has already accepted another man, she would abandon that match and accept m who is her first choice. Once a pairing between m and the woman who is his first choice has formed, the man would no longer pursue any other women, and any other men who attempt to pursue the woman would be unsuccessful since the woman has already accepted her first choice.

Q7 (10 Points)
The basic Gale-Shapley algorithm assumes all men and women have fully ranked and complete preference lists. Consider a version of the Gale-Shapley algorithm both the men and the women could be *indifferent* to some choices. That is, there could be ties and multiple members of a preference list could have the

same rank. The size of the preference lists is the same, but we allow for ties in the ranking. We say that w *prefers* m to m' if m is ranked higher on the preference list (i.e. m is not tied m'). The converse holds for men.

With indifference in the rankings, we could consider at least two forms of instability:

A. A *strong instability* in a perfect matching S consists of a man *m* and a woman *w*, such that each of m and w *prefers* the other to their partner in S. Does there always exist a perfect matching with no *strong instability?* Give an algorithm guaranteed to find a perfect matching with no *strong instability* or a counterexample.

B. A *weak instability* in a perfect matching S consists of a man *m* and a woman *w*, such that their partners in S are w' and m' respectively and one of the following holds:

i. -m prefers w to w', and w either prefers m to m' or is indifferent

ii. -w prefers m to m', and m either prefers w to w' or is indifferent

Give an algorithm guaranteed to find a perfect matching with no *weak instability* or a counter-example.

Ans:

A:

When there is a tie between two men in the stable matching problem, such as m1 and m2, we can resolve it by assigning an arbitrary order to them, for example by placing m1 in front of m2 (or m2 in front of m1). This transforms the problem into a basic stable matching problem, and we can then use the Gale-Shapley algorithm to solve it. The choice of order does not affect the outcome, as both m1 and m2 have equal standing in the tie.

B:

One counter-example is a scenario where two men, m1 and m2, both prefer woman w1 over woman w2, while w1 and w2 are indifferent to both men. In this case, no matter which man is paired with which woman, there will always be a weak instability. This demonstrates that a perfect matching with no weak instability may not always be achievable, even with the use of an algorithm like Gale-Shapley.

Q8 (10 Points)

For this problem, we will explore the issue of truthfulness in the Stable Matching Problem and specifically in the Gale-Shapley algorithm. The basic question is: Can a man or a woman end up better off by lying about his or her preferences? More concretely, we suppose each participant has a true preference order. Now consider a woman *w*. Suppose *w* prefers man *m* to *m'*, but both *m* and *m'* are low on her list of preferences. Can it be the case that by switching the order of *m* and *m'* on her list of preferences (i.e., by falsely claiming that she prefers *m'* to *m*) and running the algorithm with this false preference list, *w* will end up with a man *m''* that she truly prefers to both *m* and *m'*? (We can ask the same question for men, but will focus on the case of women for purposes of this question.)

Resolve this question by doing one of the following two things:

(a) Give a proof that, for any set of preference lists, switching the order of a pair on the list cannot improve a woman's partner in the Gale- Shapley algorithm; or

(b) Give an example of a set of preference lists for which there is a switch that would improve the partner of a woman who switched preferences.

Ans:

In this question, the main idea is that if there's a scenario exist, that the switch of a women (w) may benefit by misrepresenting her preferences. If her "not first rank" man switches, causing her "first rank" man to be ousted from his pairing and added back to the pool of eligible bachelors, and w is his second choice, then she would likely end up in a better situation by lying about her preferences.

Let me use an example to explain it:

Pre	1st	2nd	3rd
m1	w2	w1	w3
m2	w1	w2	w3
m3	w1	w2	w3
Pre	**1st**	**2nd**	**3rd**
w1	m1	m2	m3
w2	m2	m1	m3
w3	m1	m2	m3
However, w1 lied about her preference, her fake form shows			
Pre	**1st**	**2nd**	**3rd**
w1	m1	m3	m2

In the original preferences, there is a stable matching with m1 being paired with w2, m2 being paired with w1, and m3 being paired with w3. However, after w1 lies about her preferences and changes it to m1, m2, m3, m1 would prefer w1 over w2 and therefore m1 would try to pair with w1 instead. This would cause w2 to become unmatched, and w1 and m1 would be paired. Thus, the stable matching would change from (m1, w2), (m2, w1), (m3, w3) to (m1, w1), (m2, w2), (m3, w3).

In this case, w1's action of lying about her preferences has disrupted the stability of the original matching and caused her paring end up better.

Q9 (5 Points)

One algorithm requires $n\log_2(n)$ seconds and another algorithm requires $\sqrt{n}$ seconds. Which one is asymptotically faster? What is the cross-over value of n? (The value at which the curves intersect?)

Ans:

The order of growth of n is O(n), while the order of growth of $n \log_2(n)$ is O (n log n). Therefore, for large values of n, n log2(n) grows faster than n.

For the cross-over value, we could just construct an equation to make $n = n \log_2(n)$, and we could get the solution of the equation is $e^{2W(\log(2)/2)}$, which is approximately equals to 1.701.

Q10 (25 Points): Coding Problem

The World Cup is changing its playoff format using the Gale-Shapley matching algorithm. The eight best teams from groups A, B & C, called Super Group 1, will be matched against the eight best teams from groups D, E & F, called Super Group 2, using the Gale-Shapley matching algorithm. Further social media

will be used to ask fans, media, players and coaches to create a ranking of which teams they would most like to see play their favorite team.

A. Find a Gale-Shapley implementation in python on Github and modify it so that the eight Super Group 1 teams will be matched against the eight Super Group 2 teams. You can make up the preference lists for each team. Make sure you cite any code you use or state that you wrote it from scratch if you did.

   Ans:

   We could directly use the Gale-Shapley algorithm can solve the problem of pairing each team from Group 1 with a team from Group 2 in an 8-team match. The algorithm works by having each team from Group 1 propose to the highest-ranked team from Group 2 that it has not yet proposed to, and each team from Group 2 accepting the proposal from the highest-ranked team that has proposed to it. This process continues until all teams have been paired with one another. The resulting pairings will be stable, meaning that there are no two teams that would both rather be paired with each other than with their current partners.

   The code is *in the Jupyter notebook* in the folder. The original code cite from: https://github.com/Vishal-Kancharla/Gale-Shapley-Algorithm/blob/master/StableMariages.py

   After applying the Gale-Shapley algorithm, we get:

   [3, 0, 1, 4, 5, 6, 7, 2]

   The first number in the list (3) represents the team from group2 that the first team from group1 is paired with, and so on. So, in this case, group1's first team is paired with group2's third team, group1's second team is paired with group2's first team, and so on.

B. Use a loop to shuffle the preference lists for each team 1000 times. Calculate the percentage of stable playoff matches. See the function random. Shuffle (x[, random]) https://docs.python.org/2/library/random.html

   Ans:

   The code is *in the Jupyter notebook* in the folder.

   After executing we derive that the percentage of stable playoff matches is 100%.

C. Randomly assume certain teams win and lose each round and eliminate the losers from the preference lists for each team. Can the Gale-Shapley matching algorithm be applied over and over in each round (16 teams, 8 teams, 4 teams, 2 teams) to create stable matches? You can answer this with code or rhetoric.

   Ans:

   Yes, the Gale-Shapley algorithm can still be applicable in this scenario. In the case of an equal number of teams being eliminated in each round, the algorithm can be utilized without

modification. In the event that the number of teams in Groups 1 and 2 is unequal, additional rules could be added to address this imbalance. These rules could involve removing the unpaired team and assigning a bye for the current round or pairing the unpaired team with a randomly selected opponent from the opposite group, and continuing to apply the Gale-Shapley algorithm to solve the problem.

D. Now combine the lists so that any team can be matched against any other irrespective of conference. Can the Gale-Shapley matching algorithm still create stable matches? (With just one list matching against itself?) You can answer this with code or rhetoric.

Ans:

The situation of having only one list raises a challenge for obtaining a stable match using the Gale-Shapley algorithm, as the algorithm assumes that both sides have equal preferences. However, this doesn't mean that the algorithm cannot be applied to create matches in this scenario. Similar to the roommate problem above, the Gale-Shapley algorithm can still be used to make matches, but a stable match cannot be guaranteed. For example, consider a scenario where 4 teams, A, B, C, and D, are trying to be matched with each other. With only one list of preferences, team A might prefer team B, team B might prefer team C, team C might prefer team D, and team D might prefer team A. In this scenario, no stable match can be obtained as each team prefers someone else.

E. Double the size of the lists in *problem A* several times (you can make up team names like team1, team2, etc.) and measure the amount of time it takes to create stable matches. How fast does the execution time grow in relation to the size of the lists?

Ans:

The execution time of the Gale-Shapley algorithm grows in proportion to the square of the size of the input lists. In mathematical terms, the time complexity of the algorithm is O(n^2), where n is the number of elements in the lists. This means that if the size of the data is doubled, the execution time will increase by a factor of four.

It's important to note that the execution time may not exactly follow the predicted multiple of four in the case of small datasets and slow network speeds, as seen in the Jupyter notebook in the folder. However, on a larger scale, the growth rate of the execution time is expected to align with the prediction of O(n^2).