HW 1

Intro Algorithms EN.601.433
Fall 2023
Due: Friday 9/15/2023 at 11:59 PM

Instructions

- This homework is worth 10% of your final grade.
- This homework is due Friday 9/15/2023 at 11:59 PM on Gradescope. Solutions will be posted on Canvas at midnight. No late homework will be accepted. To access Gradescope, click the Gradescope link on Canvas.
- You are **required** to type your homework. We will provide the LaTeX source code to the homework assignments, which you may optionally use as a template. If you install LaTeX on your computer, you can generate a PDF of your homework by running the command:

  ```
  latexmk -pdf my-homework.tex
  ```

  Alternately, you can use `https://www.overleaf.com/edu/jhu` as a web based LaTeX editor.
- You can collaborate with one other person. Collaboration should happen through verbal communication, scratch paper, whiteboards, etc. You are **not** allowed to copy homework solutions from another student. All students are required to submit their own write-up.
- You are **not** allowed to use any website to find solutions to these problems.
- Do not write your name on your homework, instead write your Johns Hopkins Student ID. This will allow us to grade your homework anonymously.
- For questions that require you to write an algorithm out, you can either use pseudo-code or English description.

1 Stable Matching Many-to-One (6 points)

Five students–Alice, Bob, Cathy, Drew, and Eve–have applied for their medical residency at three different teaching hospitals–JH, NY, and UP. Each medical school has space to accept at most two students. Furthermore, all teaching hospitals have agreed to use the Gale-Shapley algorithm to determine where each student will study.

Modify the Gale-Shapley algorithm to handle the case where each hospital will accept k students. Then use your modified algorithm to compute the matching given the preferences below.

Ranking	Alice	Bob	Cathy	Drew	Eve
1	JH	JH	NY	NY	JH
2	UP	NY	UP	JH	NY
3	NY	UP	JH	UP	UP

Ranking	JH	NY	UP
1	Alice	Alice	Alice
2	Cathy	Eve	Drew
3	Drew	Cathy	Bob
4	Eve	Bob	Eve
5	Bob	Drew	Cathy

1.1 Solution: Modified Gale-Shapley Algorithm

```
INITIALIZE M to empty matching, and some number k as the number of students each hospital
can accept.
WHILE (some hospital h has not proposed to every student and h has less than k matches) {
    s ← first student on h's list to whom h has not yet proposed.
    IF (s is unmatched):
        Add h-s to matching M.
    ELSE IF (s prefers h to current partner h'):
        s rejects h' and creates h-s to matching M.
    ELSE:
        s rejects h.
}
```

1.2 Final Matching:

Alice - JH
Bob - UP
Cathy - NY
Drew - JH
Eve - NY

2 Stable Matching Malicious Intent (5 Points)

So far, we have assumed that when people give their preference list as an input to the Gale-Shapley algorithm, they are acting truthfully. In other words, their preference list truly reflects their desire. However, what will happen if someone lied about their list? Could they achieve better results by manipulating it?

More concretely, consider a student S who prefers a university U to U', yet both of them are low on the list. Could it be the case that by switching the order of U and U', that S will match with U'' who is preferred to both U and U'? Resolve this by proving that such an U and U' does not exist and there is no way for S to match with U'', or give a counterexample where S will match with U''.

2.1 Proof by Direct Counterexample:

We prove that it is possible if someone lies about their list that they can achieve better results. Consider a Gale-Shapley Algorithm matching students and universities. Assume that it is the universities who are proposing, there are an equal number of students and universities, and every student ranks every university. Consider the following preference rankings, and assume that they are truthful:

Ranking	Alice	Bob	Cathy
1	JH	JH	NY
2	NY	NY	JH
3	UP	UP	UP

Ranking	JH	NY	UP
1	Cathy	Cathy	Alice
2	Alice	Alice	Cathy
3	Bob	Bob	Bob

2.1.1 Original Matching Result

The stable matching created by the Gale-Shapley Algorithm is in this case:
Alice - NY
Bob - UP
Cathy - JH

2.1.2 What if Cathy lies?

Now assume that Cathy lies about her preferences, ranking UP above JH in an attempt to receive NY as a matching. In this case, the preferences are as follows:

Ranking	Alice	Bob	Cathy
1	JH	JH	NY
2	NY	NY	UP
3	UP	UP	JH

Ranking	JH	NY	UP
1	Cathy	Cathy	Alice
2	Alice	Alice	Cathy
3	Bob	Bob	Bob

2.1.3 Cathy's Lie - Stable Matching Result

The stable matching created by the Gale-Shapley Algorithm is in this case:
Alice - JH
Bob - UP
Cathy - NY

2.1.4 Conclusion

Therefore, it is possible for a student to be matched with a better U'' than both U and U' if they lie about the order of U and U'. By lying and switching her preferences between UP and JH, Cathy successfully matched with NY.

3 Someone is matched with their last choice (5 points)

Consider an execution of the Gale-Shapley algorithm in which n candidates are applying for n jobs, and all candidates ranking includes all possible jobs. The candidates are proposing, and the companies are accepting/rejecting the proposals.

Prove or provide a counterexample that at most one applicant gets their last choice.

3.1 Solution: Proof by Contradiction:

Let's use proof by contradiction to prove that in a Gale-Shapley it is not possible for more than one candidate to get their last choice. Assume we are matching candidates and companies, with the candidates proposing. For the sake of contradiction, assume there are 2 candidates, Candidate A and Candidate B who are both matched to their last choice companies.

Consider the Gale-Shapley Algorithm. In this algorithm, candidates propose to their preferred companies, and companies accept candidates based on their preferences. In a series of rounds, an candidate may propose to a company if they have not been accepted by a more preferred company yet. The company, on the other hand, can accept candidates based on their preferences, but they may change their minds if they receive a proposal from a more preferred candidate in a later round.

Given that both Candidate A and Candidate B are matched to their last choices, it means that there are no companies that prefer these candidates over any others they have accepted so far. If there were companies that preferred Candidate A or Candidate B over others, they would have accepted them and would not have ended up with their last choices.

Now, let's consider Candidate A. Since Candidate A is matched to their last choice, it means that there are no other colleges that prefer Candidate A over their current assignment. Otherwise, if there were such companies, Applicant A would have proposed to them and would have been accepted by a more preferred company.

The same logic applies to Candidate B. Since Company B is also matched to their last choice, there are no other companies that prefer Candidate B over their current assignment.

However, this creates a contradiction because both Candidate A and Candidate B cannot be the least preferred applicants for all companies. If they were, then all other applicants would have been matched to companies they prefer more than Candidate A and Candidate B, which is not possible in a stable matching. A company cannot have Candidate A and Candidate B as their last choices, since a company can only have one last choice. Since there are 2 companies remaining in this situation, both of them must prefer one over the other, and so only one of the 2 candidates can actually receive their last choice. If both of them receive their last choices, it means that the 2 companies ranked them both equally last, which is a contradiction.

Therefore, the assumption that more than one candidate gets their last choice in the Gale-Shapley Algorithm leads to a contradiction. Hence, it is not possible for more than one candidate to get their last choice in the final stable matching produced by the Gale-Shapley Algorithm.

4 Order the Big-$\mathcal{O}$ Complexities (3 Points)

Sort the following list of functions in ascending order of growth rate and briefly explain why you put them in such order. For example, if $f(n)$ appears before $g(n)$ then $f(n) = O(g(n))$ and you need to explain why $f(n) = O(g(n))$.

Note: A graph depicting the growth of these functions does not count as an explanation.

$$
\begin{aligned}
g_1(n) &= 2^{\sqrt{\log n}} \\
g_2(n) &= 2^n \\
g_3(n) &= n(\log n)^3 \\
g_4(n) &= n^{4/3} \\
g_5(n) &= n \log n \\
g_6(n) &= 2^{2^n} \\
g_7(n) &= 2^{n^2}
\end{aligned}
$$

4.1 Solution:

The following are the functions ranked in terms of growth rate, with the first function listed growing slowest, and the last function growing fastest.

$$
\begin{aligned}
g_1(n) &= 2^{\sqrt{\log n}} \\
g_5(n) &= n \log n \\
g_3(n) &= n(\log n)^3 \\
g_4(n) &= n^{4/3} \\
g_2(n) &= 2^n \\
g_7(n) &= 2^{n^2} \\
g_6(n) &= 2^{2^n}
\end{aligned}
$$

Definition of Big O: If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, then $f(n)$ is $\mathcal{O}(g(n))$. Calculating the limits:

$$\lim_{n\to\infty} \frac{2^{\sqrt{\log n}}}{n \log n} = 0 \text{ which means that } g_1(n) = \mathcal{O}(g_5(n))$$

$$\lim_{n\to\infty} \frac{n \log n}{n(\log n)^3} = 0 \text{ which means that } g_5(n) = \mathcal{O}(g_3(n))$$

$$\lim_{n\to\infty} \frac{n(\log n)^3}{n^{4/3}} = 0 \text{ which means that } g_3(n) = \mathcal{O}(g_4(n))$$

$$\lim_{n\to\infty} \frac{n^{4/3}}{2^n} = 0 \text{ which means that } g_4(n) = \mathcal{O}(g_2(n))$$

$$\lim_{n\to\infty} \frac{2^n}{2^{n^2}} = 0 \text{ which means that } g_2(n) = \mathcal{O}(g_7(n))$$

$$\lim_{n\to\infty} \frac{2^{n^2}}{2^{2^n}} = 0 \text{ which means that } g_7(n) = \mathcal{O}(g_6(n))$$

5 Contained in Θ (3 points)

Show that for positive real constants a and b,

$$(n+a)^b = \Theta(n^b)$$

Proof:

$$\text{To prove Big-}\Theta\text{, show that } \lim_{n\to\infty} \frac{f(n)}{g(n)} = c \text{ for some constant } 0 < c < \infty$$

$$\lim_{n\to\infty} \frac{(n+a)^b}{(n^b)} = (\lim_{n\to\infty} (\frac{n+a}{n}))^b = (\lim_{n\to\infty} (1+\frac{a}{n}))^b = (\lim_{n\to\infty} (1) + \lim_{n\to\infty} (\frac{a}{n}))^b = (1+0)^b = 1$$

Since $0 < 1 < \infty$, this satisfies our condition and proves that $(n+a)^b = \Theta(n^b)$.

6 Dropping Jars (6 points)

You're doing some stress-testing on various models of glass jars to determine the height from which they can be dropped and still not break. The setup for this experiment, on a particular type of jar, is as follows. You have a ladder with n rungs, and you want to find the highest rung from which you can drop a copy of the jar and not break it. We call this the *highest safe rung.*

It might be natural to try binary search: drop a jar from the middle rung, see if it breaks, and then recursively try from rung $\frac{n}{4}$ or $\frac{3n}{4}$ depending on the outcome. But this has the drawback that you could break a lot of jars in finding the answer.

If your primary goal was to conserve jars, on the other hand, you could try the following strategy. Start by dropping a jar from the first rung, then the second rung, and so forth, climbing one higher each time until the jar breaks. In this way, you only need a single jar — at the moment it breaks, you have the correct answer—but you may have to drop it n times (rather than $\log n$ as in the binary search solution).

So here is the trade-off: it seems you can perform fewer drops if you're willing to break more jars. To understand better how this trade-off works at a quantitative level, let's consider how to run this experiment given a fixed "budget" of $k \geq 1$ jars. In other words, you have to determine the correct answer—the highest safe rung—and can use at most k jars.

Suppose you are given a budget of $k = 2$ jars. Describe a strategy for finding the highest safe rung that requires you to drop a jar at most $f(n)$ times, for some function $f(n)$ that grows slower than linearly. In other words, it should be the case that $\lim_{n\to\infty} \frac{f(n)}{n} = 0$.

You should provide an algorithm and proof of its running time in your answer.

6.1 Solution: Strategy and Runtime

The strategy here is to use a $\sqrt{n}$ solution. Given n rungs, we can divide them into $\sqrt{n}$ sections. Then, drop the first jar on the lowest rung of every section. This gives us a maximum of $\sqrt{n}$ drops. Once the jar breaks on rung $m\sqrt{n}$, we know that the breaking point must be between $(m-1)\sqrt{n}$ and $m\sqrt{n}$. Then, we can test every rung linearly in the section between $(m-1)\sqrt{n}$ and $m\sqrt{n}$ until we find the breaking point. This also takes a maximum of $\sqrt{n}$ drops, since there are $\sqrt{n}$ rungs in a single section.

Overall, the final growth function is $f(n) = 2\sqrt{n}$, and it is the case that $\lim_{n\to\infty} \frac{2\sqrt{n}}{n} = 0$.

6.2 Solution: Algorithm

```
Given $n$ rungs on the ladder, calculate $\sqrt{n}$ (rounded), set variable $i = 1$.
    WHILE (first jar is still intact AND $i \leq n$) {
        Drop jar on the $i^{th}$ rung.
        Increase i by $\sqrt{n}$ (rounded).
    }
    Take second jar, and decrease i by $\sqrt{n}$ (rounded).
    WHILE (second jar is still intact) {
        Drop jar on the $i^{th}$ rung.
        Increase $i$ by 1.
    }
    The safest rung is the $(i-1)^{th}$ rung.
```