Lecture 4 Note Wednesday, 02/09/2022

Announcements:
- The Lecture Recordings will be available on the following YouTube Playlists Link: https://youtube.com/playlist?list=PLZaTmV9UMKlgYpo2cAiMaEWxqyvbiXDFd

Stable Matching

References:
Algorithm Design - Chapter 1 section 1

Review Exercise/In class quiz from previous class

- Prove that $1 + 1 \cdot 2 + 1 \cdot 2 \cdot 3 + \cdots + 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$ is $\Theta(n!)$.
  $1 + 1 \cdot 2 + 1 \cdot 2 \cdot 3 + \cdots + 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n = 1! + 2! + 3! + \cdots + n!$
  To show $1! + 2! + 3! + \cdots + n! = O(n!)$:
  If you have $1! + 2! + 3! + \cdots + n! \le n! + n! + n! + \cdots + n! = n \cdot n!$, which $c = n$, then c is not constant. It doesn't work.
  Instead, you can do $1! + 2! + 3! + \cdots + n! \le (n-1)! + (n-1)! + \cdots + (n-1)! + n!$ $= (n-1)(n-1)! + n! \le n(n-1)! + n! = n! + n! = 2 \cdot n!$, then you have $c = 2, n_0 = 1$.

  To show $1! + 2! + 3! + \cdots + n! = \Omega(n!)$, you want to have $1! + 2! + 3! + \cdots + n! \ge c \cdot n!$.
  $1! + 2! + 3! + \cdots + n! \ge n!$, for $c = 1$ and $n_0 = 1$.

  We have shown that $1! + 2! + 3! + \cdots + n!$ is both $O(n!)$ and $\Omega(n!)$, thus, $1! + 2! + 3! + \cdots + n!$ is $\Theta(n!)$.

- Solve for x and y in the following system of equations:
  $$\begin{cases} 2^{\left(\frac{\log_2 y^3}{3}\right)} + 4^{\left(\log_2 \sqrt{x}\right)} = 8 \\ 8^{\left(\log_4 (x^{2/3})\right)} \cdot 9^{\left(\log_3 \sqrt{y}\right)} = 15 \end{cases}$$
  $$2^{\left(\frac{\log_2 y^3}{3}\right)} + 4^{\left(\log_2 \sqrt{x}\right)} = 2^{\left(\frac{3 \cdot \log_2 y}{3}\right)} + 4^{\left(\frac{1}{2} \cdot \log_2 x\right)} = 2^{(\log_2 y)} + \left(4^{\frac{1}{2}}\right)^{\log_2 x} = y + 2^{\log_2 x} = y + x = 8$$
  $$8^{\left(\log_4 (x^{2/3})\right)} \cdot 9^{\left(\log_3 \sqrt{y}\right)} = 8^{\left(\frac{2}{3} \cdot \log_4 x\right)} \cdot 9^{\left(\frac{1}{2} \cdot \log_3 y\right)} = \left(8^{\frac{2}{3}}\right)^{\log_4 x} \cdot \left(9^{\frac{1}{2}}\right)^{\log_3 y} = 4^{\log_4 x} \cdot 3^{\log_3 y} = x \cdot y = 15$$
  To solve for x and y: $\begin{cases} x = 3 \\ y = 5 \end{cases}$ or $\begin{cases} x = 5 \\ y = 3 \end{cases}$.

Stable Matching Problem

- Consider the job market:
  - Employers have job openings to fill.
  - Applicants apply to the job openings.
- After a round of interviews:
  - Every employer has ranked applicants based on qualifications.
  - Every applicant has ranked employers based on preferences.
- After a round of job offers and acceptances, we ask the question.
- Is there a ***stable*** matching?
  - Every employer prefers every one of its accepted applicants, over every other applicant who was not received an offer.
  - Every applicant prefers the current employer over every other employer who has made an offer.

- In the stable matching problem,
  - while every applicant can accept only one job,
  - an employer may have many jobs, and
  - there may not be a job for every applicant.

- To eliminate complications caused by asymmetries:
  - n applicants apply to n employers, and
  - each employer accepts only one single applicant.

- In the marriage problem we want to arrange n marriages for n men and n women who want to get married.

Stable Marriage Problem

- Input: - A list of n men, $\{m_1, m_2, \ldots, m_n\}$, and n women, $\{w_1, w_2, \ldots, w_n\}$.
  - Preference ordering on the other set
    1) For every man m_i, list of n women in decreasing order of preference. $m_i = \{w_7, w_3, w_{10}, \ldots, w_n, \ldots, w_6\}$.
    2) Similarly, for every woman w_i, list of n women in decreasing order of preference. $w_i = \{m_9, m_3, m_n, \ldots, m_5\}$.
  - Input size: $2n^2 = O(n^2)$
- Output: A perfect matching – pairing of n men and n women such that every man is paired with exactly one woman.
  - Example: Given m_1, m_2, m_3 and w_1, w_2, w_3, a matching could be $\{(m_1, w_2),\ (m_2, w_1),\ (m_3, w_3)\}$.
  - Number of possible matching: $n!$
    - $n = 3$: 6 matchings
- Given $M = \{m_1, m_2, \ldots, m_n\}$ and $W = \{w_1, w_2, \ldots, w_n\}$.
  The set of all possible pairs $M \times W = \{(m, w) \mid m \in M, w \in W\}$.
  A *matching* is a subset of $M \times W$ such that each $m \in M$ and each $w \in W$ appear at most once.
  A *perfect matching* is a matching such that each $m \in M$ and each $w \in W$ appear exactly once.

- When is a matching stable in the marriage problem?
  - A matching is stable if it has no unstable pairing.
  - An unstable pairing is a man-woman pair (m, w) such that
    - m and w are not matched together.
    - m prefers w to his currently matched woman.
    - w prefers m to his currently matched man.

Example of Stable Matching

Men's Preference			
	1st	2nd	3rd
m_1	w_3	w_2	w_1
m_2	w_1	w_3	w_2
m_3	w_1	w_2	w_3

Women's Preference			
	1st	2nd	3rd
w_1	m_2	m_3	m_1
w_2	m_1	m_2	m_3
w_3	m_1	m_2	m_3

- Is the matching $\{(m_1, w_2), (m_2, w_3), (m_3, w_1)\}$ stable?
  No! (m_1, w_3) and (m_2, w_1) are unstable pairs.
- Is the matching $\{(m_1, w_3), (m_2, w_1), (m_3, w_2)\}$ stable?
  Yes, it's stable.

- How do we find the stable matching?
  - Brute force: 1) List all matchings
    2) Check if any of them are stable
    - Runtime will be at least $n!$ time, which approximately n^n, exponential growth, not polynomial, which consider not efficient.

- Gale-Shapley Algorithm
  - A more efficient algorithm with $O(n^2)$ runtime.

- Gale-Shapley Algorithm Pseudocode [Algorithm Design, page 6]
  - Initially all $m \in M$ and $w \in W$ are free

```
While there is a man m who is free and hasn't proposed to every
woman
    Choose such a man m
    Let w be the highest-ranked woman in m's preference list to
      whom m has not yet proposed
    If w is free then
         (m,w) become engaged
    Else w is currently engaged to m
          If w prefers m' to m then
               m remains free
          Else w prefers m to m'
               (m,w) become engaged
               m' becomes free
          Endif
    Endif
Endwhile
Return the set S of engaged pairs
```

- Description of the G-S algorithm (Men propose, women reject):
  - Runs in rounds
    - Round 1 – every man proposes to his top choice.
      What might happen?
      - A woman might not have any offer (no man proposes to her).
      - A woman has a man propose to her, then she will temporarily engage with the man.
      - A woman can get multiple offers, in this case, she will choose her favor among the offers.
    - Round 1 – every woman who received a propose temporarily engages herself to her most preferred among those that proposed.
      Reject the other proposals.
    - Round 2 – Rejected men propose to second best choice.
      - If a previously free woman gets an offer(s), she gets proposed.
      - If an engaged woman gets a better offer, she breaks her previous engagement, and gets engaged to the new best.
    - And so on (Repeat Round 2) … until no proposals made in a round

Example of Running G-S algorithm

Men's Preference			
	1st	2nd	3rd
m_1	w_3	w_2	w_1
m_2	w_1	w_3	w_2
m_3	w_1	w_2	w_3

Women's Preference			
	1st	2nd	3rd
w_1	m_2	m_3	m_1
w_2	m_1	m_2	m_3
w_3	m_1	m_2	m_3

- Round 1: m_1 proposes to w_3, both m_2 and m_3 propose to w_1.
  w_1 gets two offers, she will engage with m_2, her top choice, and reject m_3.
  w_2 will be free, since no man proposes to her.
  w_3 gets only one offer from m_1 so she will engage with m_1.

- After Round 1: We have $(m_1, w_3), (m_2, w_1)$, m_3 being rejected, and w_2 is free.
- Round 2: m_3 will propose to w_2, his second choice, and w_2 is free, so they will engage.
- After Round 2, everyone is engaged, no more proposal needs to make.
- So, we have the matching $(m_1, w_3), (m_2, w_1), (m_3, w_2)$.

Analysis of the G-S algorithm

- Does the program always terminate?
- Does it return a perfect matching?
- Is the matching stable?
- The Gale-Shapley algorithm looks simple, but it is not obvious that the returned set of pairs is a stable matching.

- Consider the viewpoint of a woman during the algorithm:
    - o If free, a woman accepts the first proposal.
    - o If engaged, then an additional proposal is accepted only if the man is ranked higher in the list of preferences.
    - o After the first proposal, a woman stays engaged.
- A woman ends up engaged to the highest ranked man, highest ranked of all men who proposed to her.

- Consider the viewpoint of a man during the algorithm:
    - o A man remains free until a proposal made to his highest ranked woman (who he has not yet proposed to) is accepted.
    - o Once engaged, a man may become free again.
- During the running of the algorithm, the ranking of women in the preference list of the man who proposes gets worse.

- [Theorem] For n men and n women, the G-S algorithm terminates after at most n^2 iterations in the loop.
  Proof:
    - o At each round, at least one new proposal is made.
      If there is no new proposal, the algorithm will stop.
    - o The same proposal cannot be made twice, since the men propose to the women by going down the preference list, and each woman is listed only once in their lists.
    - o There are n^2 possible pairs.
      Therefore, the algorithm will terminate in n^2 rounds.

- [Lemma] If a man is free at some point in the execution of the algorithm, then there is a woman to whom he has not yet proposed.
  Proof by contradiction:
    - o Assume a man is free and has already proposed to every woman.
    - o Recall the analysis from the viewpoint of a woman:
        - ▪ once a woman has been proposed to, she is no longer free;
        - ▪ either she stays in her current engagement or becomes engaged to the man who proposed if that man is higher in her ranking.
    - o The assumption implies then that all n women are engaged, and since each woman is paired only with one man, all n men must be engaged.
    - o This leads to a contradiction, so the assumption is false.

What to expect or prepare for the next class:

- We will finish up the analysis of G-S algorithm. Shown that the G-S algorithm will always return a stable matching.
- Think about whom will the G-S algorithm be in favor of?

Reading Assignment
Algorithm Design: 1.1

Suggested Problems
- Algorithm Design – Chapter 1 – 1,2

Assignment/Project [5 %]
- Implement the the G-S algorithm.

Men's Preference				
	1st	2nd	3rd	4th
m_1	w_2	w_3	w_4	w_1
m_2	w_1	w_4	w_2	w_3
m_3	w_1	w_3	w_4	w_2
m_4	w_2	w_1	w_3	w_4

Women's Preference				
	1st	2nd	3rd	4th
w_1	m_1	m_3	m_2	m_4
w_2	m_2	m_1	m_4	m_3
w_3	m_1	m_3	m_2	m_4
w_4	m_4	m_1	m_3	m_2

1) Run this example on your program, show evolution of each round.
2) Run this example again with the roles of men and women swapped (women propose, mem reject), show evolution of each round.
3) Compare your result in part (1) and (2).

- **Submission instruction:**
1) You can implement the algorithm in any language you prefer.
2) I will suggest you put your code, comments, and answer to above questions on a Jupyter Notebook. However, it's nor mandatory. You can have also sent the coding file with your implementation and answer the questions in email.
3) You will submit your work via email with the subject "**CS323 GS algorithm**". Please sent the email from your school email to xinying.chyn@qc.cuny.edu.
4) In the email, please include the following:
   a. explanation of how do your program import input.
   b. outputs of the about example for part (1) and (2)
   c. Your answer to part (3).
   d. the coding file with your implementation.
5) It's due Friday, February 18, 2022. No late submission.