CS 4820, Spring 2018 **Homework 1, Problem 2**

Name: Rongguang Wang
NetID: rw564
Collaborators: Siyao Liu (sl2928); Yihao Chen (yc2288)

(2) *(10 points)*

Consider the following scenario: n students get flown out to the Bay Area for a day of interviews at a large technology company. The interviews are organized as follows. There are m time slots during the day, and n interviewers, where $m \geq n$. Each student s has a fixed *schedule* which gives, for each of the n interviewers, the time slot in which s meets with that interviewer. This, in turn, defines a schedule for each interviewer i, giving the time slots in which i meets each student. The schedules have the property that

- each student sees each interviewer exactly once,
- no two students see the same interviewer in the same time slot, and
- no two interviewers see the same student in the same time slot.

Now, the interviewers decide that a full day of interviews like this seems pretty tedious, so they come up with the following scheme. Each interviewer i will pick a *distinct* student s. At the end of i's scheduled meeting with s, i will take s to one of the company's many in-house eateries, and they'll both blow off the entire rest of the day sipping espresso and rubbing elbows with celebrity chefs.

Specifically, the plan is for each interviewer i, and his or her chosen student s, to *truncate* their schedules at the time of their meeting; in other words, they will follow their original schedules up to the time slot of this meeting, and then they will cancel all their meetings for the entire rest of the day.

The crucial thing is, the interviewers want to plan this cooperatively so as to avoid the following *bad situation*: some student s whose schedule has not yet been truncated (and so is still following his/her original schedule) shows up for an interview with an interviewer who's already left for the day.

Give an efficient algorithm to arrange the coordinated departures of the interviewers and students so that this scheme works out and the *bad situation* described above does not happen.

Example:

Suppose $n = 2$ and $m = 4$; there are students s_1 and s_2, and interviewers i_1 and i_2. Suppose s_1 is scheduled to meet i_1 in slot 1 and meet i_2 in slot 3; s_2 is scheduled to meet i_1 in slot 2 and i_2 in slot 4. Then the only solution would be to have i_1 leave with s_2 and i_2 leave with s_1. If we scheduled i_1 to leave with s_1, then we'd have a bad situation in which i_1 has already left the building at the end of the first slot, but s_2 still shows up for a meeting with i_1 at the beginning of the second slot.

	Time slot			
	1	2	3	4
s_1	i_1		i_2	
s_2		i_1		i_2

	Time slot			
	1	2	3	4
s_1	i_1		(i_2)	
s_2		(i_1)		i_2

	Time slot			
	1	2	3	4
s_1	(i_1)		i_2	
s_2		i_1		i_2

Solution:

The interview scheduling problem is a variation of stable matching problem. An efficient algorithm is needed to avoid the bad situation as below:

For matched student - interviewer pairs (s_1, i_1) and (s_2, i_2) such that,

- s_1 meets i_2 earlier than i_1
- i_1 meats s_1 later than s_2

The interview scheduling problem can be transformed into stable matching problem by,

1) Students treat as Men; Interviewers treat as Women
2) Each student (man) lists interviewers (women)in the order (from the earliest to the latest) in which he is scheduled (Preference list ordering of students)
3) Each interviewer (woman) lists students (men) in the reverse order (from the latest to the earliest) in which she is scheduled (Preference list ordering of interviewers)

The interview scheduling solution consists of a set of student-interviewer pairs. For each pair of student-interviewer, the schedule of student after meeting with the interviewer will be truncate, and they will spend the rest of the day in cafe.

Algorithms Correctness:
With a set of ordered pairs $\{(s_b, i_b)|\text{b} = 1,2,\ldots \text{n}\}$,
Trying to prove that the bad situation will not happen, which means that there no such case that student s_a arrives for an interview with interviewer i_b already left for coffee with student s_b for all pairs of a,b.

According to the stable property, the output of the stable matching algorithm has no two matched pairs that,
For (s_a, i_a) and (s_b, i_b)

- s_a prefers i_b to i_a
- i_b prefers s_a to s_b

According to the preference list ordering, student s_a will not meet with interviewer i_b before i_a and interviewer i_b will not meet with student s_a after s_b. Therefore, there are only 2 cases: student s_a meets with interviewer i_a before i_b or interviewer i_b meets with student s_a before s_b. In the first case, the student s_a meets i_a and goes to cafe before meeting with i_b, which avoid the bad situation. In the second case, the interviewer i_b is able to meet s_a since she has not yet left the office, which avoid the bad situation.

Running Time Analysis:
The running time depends on how to transform the interview scheduling problem into stable matching problem.
Student s is scheduled to meet with interviewer i at time slot m can be interpreted as a set of n^2 ordered arrays (s, i, m). According to the time slot numbers, the arrays set can be sorted in the order of time slot numbers in time $\text{O}(n^2 \log n)$. The preference lists for each student can be built by append women for each array (s, i, m) when traversing the array list. It is similar for the preference lists construction of the interviewers. Therefore, the transformation from interview scheduling problem into stable matching can be done in $\text{O}(n^2 \log n)$. The stable matching problem takes $\text{O}(n^2)$ time. Overall, the algorithm runs in $\text{O}(n^2 \log n)$.