CS 4820, Spring 2019 **Homework 1, Problem 2**

Name: Yanling Wu
NetID: yw996
Collaborators: Yexiang Chang(yc2523), Zhizhong Peng(zp83)

(2) *(10 points)* Suppose we are given an instance of the stable matching problem, consisting of a set of n applicants $\{x_1, \ldots, x_n\}$ and a set of n employers $\{y_1, \ldots, y_n\}$, together with a list for each entity (applicant or employer) that ranks the entities of the opposite type from best to worst. This exercise concerns algorithms to solve the following problem: *decide whether there exists a stable perfect matching in which x_n is matched to y_n.*

(2a) A simple algorithm for this problem is the following: remove x_n from every employer's preference list, and remove y_n from every applicant's preference list. Run the Gale-Shapley algorithm (say, with employers proposing) to find a stable perfect matching, M, of the applicant set $\{x_1, \ldots, x_{n-1}\}$ and employer set $\{y_1, \ldots, y_{n-1}\}$. If $M \cup \{(x_n, y_n)\}$ is a stable perfect matching of the original $2n$ entities (with their original unmodified preference lists) then answer "yes"; otherwise, answer "no". Give an explicit input instance on which this algorithm outputs the wrong answer.

(2b) Design a polynomial-time algorithm to decide whether there exists a stable perfect matching in which x_n is matched to y_n. Prove that your algorithm always outputs the correct answer and analyze its running time.

Hint: The solved exercises at the end of Chapter 1 in the textbook may provide a useful subroutine for your algorithm.

Solution:

(2a) Since the Gale-Shapley algorithm can only give one kind of matching. But it is cannot guaranteed the right answer only according this one matching way.

For example, there are one applicants' set $\{x_1, x_2, x_3\}$ and one employers' set $\{y_1, y_2, y_3\}$, we have the preference lists are below:

$y_1 : x_1 > x_2 > x_3$ $\quad$ $x_1 : y_2 > y_3 > y_1$

$y_2 : x_2 > x_1 > x_3$ $\quad$ $x_2 : y_1 > y_2 > y_3$

$y_3 : x_1 > x_3 > x_2$ $\quad$ $x_3 : y_3 > y_2 > y_1$

We assume $n = 3$. Run Gale-Shapley algorithm and let y proposes to x. We can get the pairs $M = \{(x_1, y_1), (x_2, y_2)\}$ before we union the (x_3, y_3) with M. Then, we union (x_3, y_3) with M, donating $S = M \cup (x_3, y_3)$, we can see S is not perfect stable matching. Because when we add the x_3, y_3 into the set, x_3 prefers y_1 to y_3 and y_1 prefers x_3 to x_1, which is unstable. So the algorithm in the question 2a will decide that there is no a stable perfect matching in which $x_n and y_n$ are matched.

However, it does exist a stable matching where x_n is matched to y_n. For instance, $S = \{(x_1, y_2), (x_2, y_1), (x_3, y_3)\}$, which is a perfect stable matching and (x_3, y_3) are matched. So this algorithm outputs sometimes will be wrong.

(2b) Inspired by the solved exercise2, the basic idea to solve this question is that, firstly, we need to find the forbidden pairs which is the pairs that will make the (x_n, y_n) broken or make the matching unstable. Then, for the sets of $\{x_1, x_2, x_{n-1}\}$ and $\{y_1, y_2, , y_{n-1}\}$, use the algorithm mentioned in the solved exercise 2 in our book to get their stable matching, S. If the outputted matching set is the perfect, then the set $S \cup (x_n, y_n)$ should also be the perfect stable matching. The pseudo code and the proof show below.

Algorithm:
//Find the forbidden pairs For x_n and y_n separately do the finding forbidden pairs function and add all pairs to the set F.

finding forbidden pairs(x_n, y_n);
finding forbidden pairs(y_n, x_n);

Function: finding forbidden pairs(x_n, y_n):

For every y which ranked higher than y_n in the x_n's preference list, do:
For every x which ranked lower than x_n in the y's preference list, do:
add the pair (y, x) to the forbidden set F;
Endfor
Endfor
End Function

//Do gale-shapley alogorithm with a forbidden set Initially set all $x \in \{x_1, x_2, ..., x_{n-1}\}$ and $y \in \{y_1, y_2, ..., y_{n-1}\}$ to free
While there is a x who is free and hasn't proposed to every y for which $(x, y) \notin F$:
Choose such a x
Let y be the highest-ranked one in x 's preference list to whom x has not yet proposed
If y is free then
(x,y) become engaged
Else y is currently engaged to x'
If y prefers x' to x then
x remains free
Else y prefer x to x' then
(x, y) become engaged
x' becomes free
EndIf
EndIf
Endwhile
Add all pairs to the set M

//Test whether the set M is the perfect matching
count = 0
For every x in set M:
count++

If count == n-1:
add (x_n, y_n) to the set M
return M // which is the perfect stable matching

Else

return false // which means there does not exist this stable perfect matching Endif

Proof:

The proof will use the conclusion in the solved exercise, which is the matching generated by the forbidden gale-shapley algorithm is definitely be a stable matching.

If the result is true, assume that there exists a pair (x', y') could make the matching unstable. The only possibility that make matching unstable is this pair(x', y') and the pair (x_n, y_n) is unstable. But this is contradict with forbidden set. Because of the forbidden set, this (x', y') cannot exist during running the forbidden gale-shapley algorithm.

If the result is false, which means there will be a lonely x' and a lonely y', they cannot pair with others. Assume the result is true, which means x' will pair with at least one y'' and y' will pair with at least one x''. But when x' pairs with y", then there will be exist one x_a whose currently paired one ranked lower than x'. But it is contradict with the algorithms. Because in the algorithm, this lonely x' need to pair to every y except this pair combination is in forbidden set.