CMPS 102 — Fall 2018 — Homework 1

*"I have read and agree to the collaboration policy." - **Kevin Wang***

Solution to Problem 2: TA-Course Matching

Given m courses and n students, we want to assign each student to a TA position in at most one course, in such a way that all TA positions are filled and the matching is stable.

We assume that $n > \sum_{i=1}^{m} p_i$, where p_i is the number of positions in course$_i$.

Algorithm 1 Perform a stable matching to assign graduate students to TA positions.

```
Initialize each student to be free
Initialize each course[position] to be free
while some student is free and has not applied to every course do
  Choose such a student s
  c = 1st course on s's list to which s has not applied
  if c has a position that is free then
    Assign s to a position in c
  else if c prefers s to the least preferred tentative assignment s' then
    Assign s to a position in c
    Assign s' to be free
  else {No positions in c are free and s is not preferred more than an existing assignment}
    c rejects s as a TA
  end if
end while
```

Observation 1. Students apply to courses in decreasing order of preference.

Observation 2. Once a TA position is filled, it never becomes unfilled; it only "trades up" for a more preferred student.

Observation 3. There will always be unassigned students.

Claim 1. *The algorithm terminates after at most $n \times m$ iterations of the while loop.*

Proof. Each time the algorithm iterates through the while loop, a student applies to a new course. There are only $n \times m$ possible applications. □

Claim 2. *All TA positions in all courses are filled.*

Proof. Suppose, for the sake of contradiction, that a position in course c is not filled upon termination of the algorithm. There is also, by Observation 3, some student s, who is not assigned upon termination. By Observation 2, s did not apply to a position in c. However, s has applied to every course, since he ends up unassigned – resulting in a contradiction. □

Claim 3. *There are no unstable assignments.*

Proof. Suppose, for the sake of contradiction, that $c - s'$ is an unstable assignment.

Case 1: The instructor of c favors an unassigned student s' over a current assignment s.

Proof of Case. Assume s' never applied to c. By Observation 1, s' prefers their current assignment to c. However, s' is unassigned. Thus $c - s'$ is stable.

Proof of Case. Assume s' applied to c. This implies s' was rejected by the instructor of c immediately or later on. By Observation 2, the instructor of c prefers their current assignment(s) to student s'. Thus $c - s'$ is stable.

Case 2: The instructor of c favors student s' over a current assignment s and the student s' favors course c over the current position in course c'.

Proof of Case. Assume s' never applied to c. By Observation 1, s' prefers their current assignment c' to c. Thus $c - s'$ is stable.

Proof of Case. Assume s' applied to c. This implies s' was rejected by the instructor of c immediately or later on. By Observation 2, the instructor of c prefers their current assignment(s) to student s'. Thus $c - s'$ is stable.

In in all possible cases of instability, $c - s'$ is a stable assignment, a contradiction. □

The total run time of initializing all the students and all the course positions is at most $O(n + (n - 1))$, respectively. Running the modified Gale-Shapley takes $O(n * m)$. The time complexity of this algorithm is: $O(n * m)$.

The total space needed for the array of student assignments and the 2-dimensional array of course position assignments is at most $O(n + (n - 1))$. The spaces needed for both the student preferences of courses and the course instructors rank of students are $O(m * n)$. A queue of unassigned students is at most $O(n)$. The space complexity of this algorithm is: $O(m * n)$.