CS 70 Discrete Mathematics and Probability Theory
Spring 2020 DIS 2A

1 Stable Matching III

1. True or False?

   (a) If a candidate accidentally rejects a job she prefers on a given day, then the algorithm ends with a rogue couple.

   (b) The Propose-and-Reject Algorithm never produces a candidate-optimal matching.

   (c) If the same job is last on the preference list of every candidate, the job must end up with its least preferred candidate.

2. As you've seen from lecture, the jobs-proposing Propose-and-Reject Algorithm produces an employer-optimal stable matching. Let's see if the candidate have any way of improving their standing. Suppose exactly one of the candidates has the power to arbitrarily reject one proposal, regardless of which job she has on her string (if any). Construct an example that illustrates the following: for any $n \geq 2$, there exists a stable matching instance for which using this power helps **every** candidate, i.e. every candidate gets a better job than she would have gotten under the jobs-proposing Propose-and-Reject Algorithm.

Solution:

1. (a) False, consider the case:

Jobs			Candidates		
A	1	2	1	A	B
B	2	1	2	B	A

   Using TMA, the matching will be: $(A,1),(B,2)$. If candidate 1 rejects job A despite having no other jobs on her string, job A will propose to candidate 2 and also get rejected. This leaves both candidate 1 and job A partnerless. In this case, the accidental rejection doesn't lead to a rogue couple, it prevents a matching from being produced at all.

   (b) False. Suppose that all jobs have a different first choice. Also supposed that the job proposing is each candidate's first choice. In this case, the algorithm would end after the first day with both jobs and candidates ending with their top pick. In this case, the result is candidate-optimal.
   An alternative solution is to use the Universal Preferences from Dis 1B, in which all jobs have the same preference list, and all candidates have the same preference list.

   (c) False, consider the following case where jobs are letters and candidates are numbers:

Jobs		
A	1	2
B	2	1

Candidates		
1	B	A
2	B	A

A is last on every candidates's list, however, $\{(A,1),(B,2)\}$ is a stable pairing where A got its top choice candidate.

2. Without loss of generality, assume that candidate 1 is the candidate with this special power. Now, assume the preference lists are ordered as follows:

Job	Preferences
1	$1 > 2 > \cdots > n-1 > n$
2	$2 > 3 > \cdots > n-1 > n > 1$
3	$3 > 4 > \cdots > n > 1 > 2$
	$\cdots$
n	$n > 1 > 2 > \cdots > n-1$

Candidate	Preferences
1	$n > n-1 > \cdots > 2 > 1$
2	$1 > n > n-1 > \cdots > 2$
3	$2 > 1 > n > n-1 > \cdots > 3$
	$\cdots$
n	$n-1 > n-2 > \cdots > 1 > n$

If the Propose-and-Reject Algorithm was run with these preference lists, then each candidate would be stuck with her least-preferred job. However, let's say candidate 1 rejects job 1 on the first day, even though she has nobody on her string. Then, job 1 will be forced to propose to its second option, candidate 2, and she will accept because the job is her first choice. Now, job 2 has no partner and will propose to candidate 3, who will accept, leaving job 3 without a partner. This process continues until job n proposes to candidate 1. One rejection has led all candidates to get their best choice instead of their worst choice!

2 Examples or It's Impossible

Determine if each of the situations below is possible with the traditional propose-and-reject algorithm. If so, give an example with at least 3 jobs and 3 candidates. Otherwise, give a brief proof as to why it's impossible.

(a) Every job gets its first choice.

(b) Every candidate gets her first choice, even though her first choice does not prefer her the most.

(c) Every candidate gets her last choice.

(d) Two or more jobs can have the same optimal candidate.

(e) A job which is second on every candidate's list gets its last choice.

Solution:

(a) One way to construct an example is to have each job have a distinct first choice. For example,

Jobs	Preferences
1	$A > C > B$
2	$B > C > A$
3	$C > A > B$

Candidates	Preferences
A	$3 > 2 > 1$
B	$1 > 3 > 2$
C	$2 > 1 > 3$

(b) An example where every candidate gets her first choice, and every job its second choice:

Job	Preferences
1	$C > A > B$
2	$A > B > C$
3	$A > C > B$

Candidate	Preferences
A	$1 > 2 > 3$
B	$2 > 3 > 1$
C	$3 > 1 > 2$

(c) One method for constructing an example of this is to have each candidate have a unique last choice, and have each job most prefer the candidate who likes it the least. As an example,

Job	Preferences
1	$A > C > B$
2	$B > C > A$
3	$C > B > A$

Candidate	Preferences
A	$2 > 3 > 1$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

(d) For the sake of contradiction, assume that we have some instance of the Stable Matching problem where both job J and job J' have candidate C as their optimal candidate. We further assume without loss of generality that C prefers J to J' (if this is not the case, we can just switch the names to make it so). Because C is J''s optimal candidate, we know by definition that there must exist some stable pairing P in which J' is paired with C. Call J's candidate in P C^*. Since C is J's optimal candidate, we know by definition that J must prefer C to any candidate it is ever paired with in any stable pairing–including C^*. As well, we previously said that C prefers J to J'. Thus, J and C would form a rogue couple in P, which is a contradiction because P is stable. So our initial assumption must be false: there must never exist two jobs who have the same optimal candidate.

(e) One method for constructing an example is to have j_i and c_i both prefer each other the most for $1 \leq i < n$. We can then put j_n in the second position on every candidate's list and c_n at the last position on j_n's list. An explicit example of this with $n = 3$ is shown below.

Jobs	Preferences
1	$A > C > B$
2	$B > C > A$
3	$A > B > C$

Candidates	Preferences
A	$1 > 3 > 2$
B	$2 > 3 > 1$
C	$1 > 3 > 2$