CS 70 Discrete Mathematics and Probability Theory
Summer 2023 Huang, Suzani, and Tausik

DIS 1A

1 Stable Matching

Note 4

Consider the set of jobs $J = \{1, 2, 3\}$ and the set of candidates $C = \{A, B, C\}$ with the following preferences.

Jobs	Candidates
1	A > B > C
2	B > A > C
3	A > B > C

Candidates	Jobs
A	2 > 1 > 3
B	1 > 3 > 2
C	1 > 2 > 3

Run the traditional propose-and-reject algorithm on this example. How many days does it take and what is the resulting pairing? (Show your work.)

Solution:

The algorithm takes 5 days to produce a matching. The resulting pairing is as follows. The circles indicate the job that a candidate picked on a given day (and rejected the rest).

$$\{(A,2),(B,1),(C,3)\}.$$

Candidate	Day 1	Day 2	Day 3	Day 4	Day 5
A	(1),3	(1)	1,(2)	(2)	(2)
B	(2)	2,(3)	(3)	(1),3	(1)
C					(3)

2 Propose-and-Reject Proofs

Note 4

Prove the following statements about the traditional propose-and-reject algorithm.

(a) In any execution of the algorithm, if a candidate receives a proposal on day i, then she receives some proposal on every day thereafter until termination.

(b) In any execution of the algorithm, if a candidate receives no proposal on day i, then she receives no proposal on any previous day j, $1 \leq j < i$.

(c) In any execution of the algorithm, there is at least one candidate who only receives a single proposal. (Hint: use the parts above!)

Solution:

(a) The idea is to induct on the number of days passed so far.
Base case: Candidate C receives a proposal on day i
Inductive Step: Assume C receives a proposal on day $j \geq i$ from job J. We want to show she will also get a proposal on day $j+1$. There are two cases: C prefers J to all other offers, or C prefers some job J' to J. In the first case J proposes to C on day $j+1$ and in the second J' proposes to C on day $j+1$ so C receives at least one proposal on day $j+1$.

(b) One way is to use a proof by contradiction. Assume that a candidate receives no proposal on day i but did receive a proposal on some previous day j, $1 \leq j < i$. By the previous part, since the candidate received a proposal on day j, she must receive at least one proposal on every day after j. But $i > j$, so the candidate must have received a proposal on day i, contradicting our original assumption that she did not.

(c) Let's say the algorithm takes k days. This means that every candidate must have received a proposal on day k. However, this also means that there is at least one candidate C who does not receive a proposal on day $k-1$–if this were not the case, the algorithm would have already terminated on day $k-1$. Then from part (b), since C did not receive a proposal on day $k-1$, she didn't receive a proposal on any day before k. Furthermore, we know she got exactly one proposal on day k, since the algorithm terminated on that day. Thus, we have that C receives exactly one proposal throughout the entire run of the algorithm.

3 Be a Judge

Note 4

By stable matching instance, we mean a set of jobs and candidates and their preference lists. For each of the following statements, indicate whether the statement is True or False and justify your answer with a short 2-3 line explanation:

(a) There is a stable matching instance for n jobs and n candidates for $n > 1$, such that in a stable matching algorithm with jobs proposing, every job ends up with its least preferred candidate.

(b) In a stable matching instance, if job J and candidate C each put each other at the top of their respective preference lists, then J must be paired with C in every stable pairing.

(c) In a stable matching instance with at least two jobs and two candidates, if job J and candidate C each put each other at the bottom of their respective preference lists, then J cannot be paired with C in any stable pairing.

(d) For every $n > 1$, there is a stable matching instance for n jobs and n candidates which has an **unstable** pairing where **every** unmatched job-candidate pair is a rogue couple or pairing.

Solution:

(a) **False**: If this were to occur, it would mean that at the end of the algorithm, every job would have proposed to every candidate on its list and has been rejected $n-1$ times. This would also

require every candidate to reject $n-1$ jobs. We know this is impossible though, as we learned above that at least one candidate receives a single proposal. Thus, there must be at least one candidate who is not proposed to until the very last day.

(b) **True**: We give a simple proof by contradiction. Assume that J and C put each other at the top of their respective preference lists, but J and C are not paired with each other in some stable pairing S. Thus, S includes the pairings (J, C'), (J', C), for some job J' and candidate C'. However, J prefers C over its partner in S, since C is at the top of J's preference list. Similarly C prefers J over her current job. Thus (J,C) form a rogue couple in S, so S is not stable. We have arrived at a contradiction that S exists where C and J are not paired.

Therefore if job J and candidate C put each other at the top of their respective preference lists, then J must be paired with C in every stable pairing.

(c) **False**: The key here is to realize that this is possible if job J and candidate C are at the bottom of everybody else's preference list as well. For example, a two job, two candidate instance where the first job is best for both candidates, and the first candidate is best for both jobs has its only stable pairing where the first job and first candidate are paired (by part (b)), and the second job and second candidate are paired. The second job and second candidate are each other's least preferred option.

(d) **True**: The key idea to this solution is that we want a set of preferences for which J_i and C_i like each other the least and make a pairing J_i and C_i together. In this matching M each unmatched couple is a rogue couple. In particular, an instance can be constructed by forming preferences where job i's (candidate i's) least favorite candidate is candidate i (job i and an *arbitrary* ordering on all the others. Thus, for any job i and candidate j, where $i \neq j$, then job i prefers candidate j to i (its partner in M) and candidate j prefers job i to j (its partner in M.) That, is every pair i and j is a rogue couple.

An example, for two jobs and two candidates, is the instance:

$A:1$	2
$B:2$	1

$1:A$	B
$2:B$	A

The pairing $P = \{(A,2),(B,1)\}$ is the pairing where each pair of jobs and candidates that are not in P, $(A,1)$ and $(B,2)$, are rogue couples.

4 Pairing Up

Note 4

Prove that for every even $n \geq 2$, there exists an instance of the stable matching problem with n jobs and n candidates such that the instance has at least $2^{n/2}$ distinct stable matchings.

Solution:

To prove that there exists such a stable matching instance for any even $n \geq 2$, we just need to show how to construct such an instance.

The idea here is that we can create pairs of jobs and pairs of candidates: pair up job $2k-1$ and $2k$ into a pair and candidate $2k-1$ and $2k$ into a pair, for $1 \leq k \leq n/2$ (you might come to this idea since we are asked to prove this for *even* n).

For n, we have $n/2$ pairs. Choose the preference lists such that the kth pair of jobs rank the kth pair of candidates just higher than the $(k+1)$th pair of candidates (the pairs wrap around from the last pair to the first pair), and the kth pair of candidates rank the kth pair of jobs just higher than the $(k+1)$th pair of jobs. Within each pair of pairs (j, j') and (c, c'), let j prefer c, let j' prefer c', let c prefer j', and let c' prefer j.

Our construction thus results in preference lists like follows:

J_1	$C_1 > C_2 > \ldots$
J_2	$C_2 > C_1 > \ldots$
$\vdots$	$\vdots$
J_{2k-1}	$C_{2k-1} > C_{2k} > \ldots$
J_{2k}	$C_{2k} > C_{2k-1} > \ldots$
$\vdots$	$\vdots$
J_{n-1}	$C_{n-1} > C_n > \ldots$
J_n	$C_n > C_{n-1} > \ldots$

C_1	$J_2 > J_1 > \ldots$
C_2	$J_1 > J_2 > \ldots$
$\vdots$	$\vdots$
C_{2k-1}	$J_{2k} > J_{2k-1} > \ldots$
C_{2k}	$J_{2k-1} > J_{2k} > \ldots$
$\vdots$	$\vdots$
C_{n-1}	$J_n > J_{n-1} > \ldots$
C_n	$J_{n-1} > J_n > \ldots$

Each match will have jobs in the kth pair paired to candidates in the kth pair for $1 \leq k \leq n/2$.

A job j in pair k will never form a rogue couple with any candidate c in pair $m \neq k$. If $m > k$, then c prefers her current partner in the mth pair to j. If $m < k$, then j prefers its current partner in the kth pair to c. Then a rogue couple could only exist in the same pair - but this is impossible since exactly one of either j or c must be matched to their preferred choice in the pair.

Since each job in pair k can be stably matched to either candidate in pair k, and there are $n/2$ total pairs, the number of stable matchings is $2^{n/2}$.