CS386 Assignment 8

Due: Friday 11/20 10 PM ET

Conventions. Rankings are strict unless Problem 7 says otherwise.

Stable marriage. There are disjoint sets M (men) and W (women). Every person p has a list $L(p)$ of acceptable people from the other side, in decreasing order of preference. We write $q \succ_p r$ if q comes before r on $L(p)$, and being unmatched is worse than every person on $L(p)$. A *matching* μ is a set of disjoint pairs (m, w) such that $w \in L(m)$ and $m \in L(w)$; write $\mu(p)$ for the partner of p and $\mu(p) = p$ if p is unmatched. A pair $(m, w) \notin \mu$ with $w \in L(m)$ and $m \in L(w)$ is *blocking* if $w \succ_m \mu(m)$ and $m \succ_w \mu(w)$. A matching is *stable* if it has no blocking pair. From class: the deferred acceptance algorithm (DA) with men proposing outputs a stable matching μ_M; every man gets his best stable partner in μ_M and every woman her worst. By symmetry, the women-proposing DA outputs μ_W, which gives every woman her best stable partner. The total input size is $\mathcal{L} = \sum_p |L(p)|$.

Stable roommates. There is a single set N of n agents, and each x has a list $L(x) \subseteq N \setminus \{x\}$. A *matching* is a set of disjoint pairs $\{x, y\}$ with $y \in L(x)$ and $x \in L(y)$; again $\mu(x) = x$ means that x is unmatched. A pair $\{x, y\} \notin \mu$ with $y \in L(x)$ and $x \in L(y)$ is *blocking* if $y \succ_x \mu(x)$ and $x \succ_y \mu(y)$, and a matching is *stable* if it has no blocking pair.

Of the following, do $\{1, 2, 9\}$, and one each from $\{3, 4, 5\}$ and $\{6, 7, 8\}$.

1. **Warm-ups.** Do 4 of the following.

   (a) *Tracing DA.* Consider the instance

   $$\begin{array}{ll} m_1 : w_4\ w_1\ w_3 & w_1 : m_2\ m_4\ m_1\ m_3 \\ m_2 : w_4\ w_3\ w_2\ w_1 & w_2 : m_1\ m_4 \\ m_3 : w_2\ w_1\ w_4\ w_3 & w_3 : m_1\ m_4\ m_2\ m_3 \\ m_4 : w_1\ w_4\ w_3 & w_4 : m_4\ m_2\ m_3 \end{array}$$

   (note that, for example, m_1 finds w_4 acceptable but w_4 does not find m_1 acceptable). Run DA with men proposing. List every proposal and whether it is held or rejected. Give μ_M, say who is unmatched, and count the proposals (a proposal to a woman who finds the man unacceptable counts). Then find the other stable matching. Which one is μ_W? (Hint: Compare the number of proposals with the position of each man's final partner in his list. This will be Problem 3(b).)

   (b) *Three stable matchings.* Let $M = \{m_1, m_2, m_3\}$, $W = \{w_1, w_2, w_3\}$ with

   $$\begin{array}{ll} m_1 : w_1\, w_2\, w_3 & w_1 : m_2\, m_3\, m_1 \\ m_2 : w_2\, w_3\, w_1 & w_2 : m_3\, m_1\, m_2 \\ m_3 : w_3\, w_1\, w_2 & w_3 : m_1\, m_2\, m_3 \end{array}$$

Find all stable matchings and identify μ_M and μ_W.

(c) *Women can lie.* Let $M = \{m_1, m_2, m_3\}$, $W = \{w_1, w_2, w_3\}$ with

$$\begin{array}{ll} m_1 : w_1\, w_2\, w_3 & w_1 : m_3\, m_1\, m_2 \\ m_2 : w_1\, w_2\, w_3 & w_2 : m_1\, m_3\, m_2 \\ m_3 : w_2\, w_1\, w_3 & w_3 : m_1\, m_2\, m_3 \end{array}$$

Compute μ_M. Now suppose w_2 declares that only m_1 is acceptable to her (and everybody else reports truthfully). Compute the result of DA and show that w_2 is strictly better off than under truthful reporting.

(d) *Indifference: breaking ties is safe (marriage).* Suppose now that each list is a ranking with ties: $L(p)$ is an ordered list of groups, people in the same group are tied. A matching is *weakly stable* if no pair (m, w) has both $w \succ_m \mu(m)$ and $m \succ_w \mu(w)$ *strictly.* Let $\succ'$ be obtained by breaking every tie arbitrarily (so every list becomes a strict ranking that refines the original). Show that every matching that is stable for $\succ'$ is weakly stable for $\succ$. Conclude that a weakly stable matching always exists and can be found by DA.

(e) *Roommates Phase 0.* (Phase 0 is defined in Problem 9.) Suppose $y \in L(x)$ but $x \notin L(y)$. Show that deleting y from $L(x)$ does not change the set of stable matchings.

(f) *When Phase 1 ends with short lists.* Let T be the table produced by Phase 1 (defined in Problem 9). Recall from class that $y \in T(x)$ if and only if $x \in T(y)$.

i. Show that for every pair $\{x, y\}$ that is mutually acceptable but not in T, one of the two agents, say u, has $T(u) \neq \emptyset$ and ranks every member of $T(u)$ above the other agent v. (Hint: Look at why a pair gets deleted in Phase 1. If u holds x's proposal, how does u rank everybody in $T(u)$ compared to what was deleted from u's list?)

ii. Suppose that $|T(x)| \leq 1$ for every x. Let μ consist of the pairs $\{x, y\}$ with $T(x) = \{y\}$. Show that μ is a matching and that it is stable.

(g) *Tracing the roommates algorithm.* Consider the six agents

$$1 : 4\,6\,5\,2\,3, \quad 2 : 3\,6\,5\,4, \quad 3 : 5\,6\,4\,2\,1, \quad 4 : 2\,3\,5\,1\,6, \quad 5 : 1\,4\,6\,3\,2, \quad 6 : 4\,3\,2\,5\,1.$$

Run the algorithm of Problem 9. Write down the table after Phase 0 and after Phase 1. Then find an all-or-nothing cycle, say which pairs are eliminated, and write down the final table and the matching.

(h) *Many-to-one.* Colleges c_1, c_2, c_3 have capacities $2, 1, 2$ and the students $s_1, \ldots, s_5$ have

$$\begin{array}{ll} s_1 : c_1\, c_2\, c_3 & c_1 : s_3\, s_1\, s_5\, s_2\, s_4 \\ s_2 : c_1\, c_3 & c_2 : s_1\, s_4\, s_3\, s_5 \\ s_3 : c_1\, c_2 & c_3 : s_5\, s_2\, s_4\, s_1 \\ s_4 : c_2\, c_1\, c_3 & \\ s_5 : c_1\, c_3\, c_2 & \end{array}$$

Run student-proposing DA, where each college tentatively keeps its best (up to capacity) proposers. Give the final assignment and check that it is stable in the sense of Problem 8.

2. For this problem, imagine you own a fleet of freight trains and an equal number of stations. Each one of these trains T_i has a schedule S_i for this month (you can assume months are arbitrarily long in this universe), during which every train visits every station exactly once

and no two trains visit the same station in the same day. You've been informed by your engineers that every one of these trains has a problematic flaw in design and will need to be held in one of your stations for the rest of the month. These problems aren't exactly immediate - they can be held until the end of the month. Unfortunately, your stations aren't very big - each station can only hold one train, and another train cannot pass through the station while another is held there. You've been asked to truncate (end early) each train's schedule at a unique station: whatever this station is, it will remain there for the remainder of the month. Your goal is to do this so that:

(a) No other train passes through a station that holds another train

(b) The number of stops made by all trains is maximized - equivalently, the number of stops lost due to truncation is minimized.

As an example, imagine that we had two trains and two stations, where the schedule was:

Train	Day 1	Day 2	Day 3	Day 4
T_1	S_1	$\cdot$	S_2	$\cdot$
T_2	$\cdot$	S_1	$\cdot$	S_2

If T_1 were truncated at S_1, then T_2 wouldn't be able to pass through the station, and thus no valid truncation would follow. Hence T_1 must be truncated at S_2, meaning T_1 visits both S_1 and S_2 while T_2 must truncate at S_1, losing its visit to S_2.

Determine an algorithm that successfully achieves your goals. (Hint: How is this just a Stable-Marriage problem?)

3. **Running time of DA.** Throughout, the input is the list $L(p)$ of every person as an array in decreasing order of preference, and $n = |M| = |W|$. The algorithm is: keep for every man a pointer to the next woman on his list, and a stack of free men; repeatedly take a free man m, let him propose to the next woman w on his list (and advance the pointer); if w does not find m acceptable, or she holds a better proposal, she rejects him and he stays free, otherwise she holds m and releases the man she held before (who becomes free). The algorithm stops when no free man has a woman left on his list.

   (a) *Counting proposals.* Show that DA makes at most $\sum_m |L(m)|$ proposals. Now assume complete lists, $L(p)$ is the whole other side. Show that DA makes at most $n^2 - n + 1$ proposals. (Hint: Consider the moment just before the last woman receives her first proposal. At that moment, how many women are engaged? How many proposals can each man have made so far?)

   (b) *An exact formula.* For a man m, let r_m be the position of $\mu_M(m)$ in $L(m)$ (starting from 1) if m is matched in μ_M, and $r_m = |L(m)|$ otherwise. Show that DA makes exactly $\sum_m r_m$ proposals, regardless of the order in which free men are processed. (Hint: A man proposes to the women of his list in order and never proposes to a woman below his final partner. Why does he propose to every woman above?)

   (c) *Best and worst case.* Assume complete lists.

      i. Show that DA makes at least n proposals, with equality if and only if all men have different first choices.

      ii. Give an instance with exactly $n(n+1)/2$ proposals (so $\Theta(n^2)$). (Hint: Give all men the same list and all women the same list.)

   (d) *Data structures.* We count one step for reading or writing one array cell.

i. Suppose that women do *not* get any extra data structure. To decide whether w prefers m to her current man m', scan $L(w)$. Show that the total time is $O(n^3)$ for complete lists.

ii. Now build, before DA starts, a table rank$[w][m]$ for every woman and every man on her list. Show that building the table takes $O(\mathcal{L})$ time and that every proposal then takes $O(1)$ steps. Conclude that the worst-case running time is $O(\mathcal{L})$, which is $O(n^2)$ for complete lists, and that this matches the number of proposals in the worst case.

iii. *Best case.* Modify the algorithm so that the rank table for w is only built when w receives her second proposal. Show that for complete lists, if all first choices of the men are distinct, the running time is $\Theta(n)$. Explain why no algorithm that outputs a perfect matching can be faster.

(e) *Summary.* State the best-case and worst-case running time of DA, in terms of n for complete lists and in terms of $\mathcal{L}$ for incomplete lists. (For incomplete lists, in the best case DA takes time proportional to the number of men with a nonempty list.)

Remark (not to be proved). Any algorithm that finds a stable matching must, in the worst case, look at $\Omega(n^2)$ entries of the lists (Ng and Hirschberg 1990), so DA is optimal in the worst case.

4. **Coding up DA.** A starter file `stable_starter.py` is provided; it contains instance generators, a brute-force enumerator of all stable matchings, and function stubs. Instances are dictionaries mapping every person to a list (most preferred first). Submit your code and a short report (one page) with the answers to the questions below.

   (a) Implement `deferred_acceptance(men_prefs, women_prefs)` in time $O(\mathcal{L})$, handling incomplete lists, and returning the matching and the number of proposals.

   (b) Implement `blocking_pair(men_prefs, women_prefs, mu)` (running in time $O(\mathcal{L})$) which returns a blocking pair, or `None` if the matching is stable.

   (c) *Testing.* For at least 1000 random instances with $n \leq 6$ (both complete and incomplete lists), compare with the brute-force enumerator and check that your output is stable, that every man gets his best stable partner (or is unmatched in all stable matchings), and that every woman gets her worst stable partner.

   (d) *Counting proposals.* For $n = 100, 200, 400, 800, 1600$, run DA on random complete instances (average over at least 5 instances) and report the average number of proposals. Compare with $n \ln n$ and with the worst case from Problem 3. Also run it on the identical-lists instance from Problem 3 and check that you get $n(n+1)/2$.

   (e) *Many-to-one.* Implement student-proposing DA with capacities in two ways: directly, and by cloning each college c into q_c copies (as in Problem 8) and calling your function from part 1. Check that both produce the same assignment on 1000 random instances.

5. **Coding up Irving's Algorithm** Implement the algorithm of Problem 9 and compare it with the brute-force enumerator on random instances with $n \leq 8$ (with and without incomplete lists). For random complete instances with $n = 4, 8, 12, 20, 50$, estimate how often a stable matching exists. (As in 4, the file `stable_starter.py` is provided) Try to get the same runtime you find in Problem 9(g).

6. **Honesty.** A *preference profile* $\succ$ lists $\succ_p$ (including the list $L(p)$) for every person p. We compare an outcome to $\succ_p$, the *true* preferences of p: a person that is matched to somebody

not on their true list is considered worse off than being unmatched. Let $\succ'$ be a profile that differs from $\succ$ only for the members of a set $C \neq \emptyset$ (the *coalition*), and let μ' be a stable matching for $\succ'$. For a person p write $\mu_*(p) = \mu_M(p)$ if $p \in M$ and $\mu_*(p) = \mu_W(p)$ if $p \in W$, so $\mu_*(p)$ is the best partner of p in any matching that is stable for $\succ$. We will prove the following.

Theorem (Dubins and Freedman 1981; Demange, Gale, and Sotomayor 1987). There is a $p \in C$ that does *not* strictly prefer $\mu'(p)$ to $\mu_*(p)$ (with respect to $\succ_p$).

We'll need the following definition and lemma (Gale and Sotomayor 1985; Hwang 1986): a matching μ is *individually rational* (for a profile) if every matched person is on the lists of their partner and on their own list.

Blocking Lemma. Let μ be individually rational for $\succ$, and let $M' = \{m \in M : \mu(m) \succ_m \mu_M(m)\}$. If $M' \neq \emptyset$, then there are a man $m \in M \setminus M'$ and a woman $w \in \mu(M')$ such that (m, w) is a blocking pair of μ (for $\succ$).

(a) Prove the Blocking Lemma. Let $W' = \mu(M')$. First show that every $w \in W'$ is matched in μ_M to a man that she prefers to $\mu(w)$. *Case 1:* some $w \in W'$ has $\mu_M(w) \notin M'$. Then $(\mu_M(w), w)$ is blocking. *Case 2:* $\mu_M(W') \subseteq M'$. Consider the last moment in the men-proposing DA (which produces μ_M) at which a man of M' makes a proposal, say m_0 to w_0. Why is $w_0 = \mu_M(m_0) \in W'$? Show that w_0 rejects at that moment a man $m_2 \notin M'$ that she prefers to $\mu(w_0)$, and that (m_2, w_0) blocks μ.

Now we'll actually prove the above theorem. Suppose for contradiction that every $p \in C$ strictly prefers $\mu'(p)$ to $\mu_*(p)$.

(b) Show that μ' is individually rational for $\succ$. (Hint: If $p \in C$ then $\mu'(p) \succ_p \mu_*(p) \succeq_p p$. What about $p \notin C$ and its partner? Note that μ' is stable for $\succ'$.)

(c) *The men's side.* Assume first $C \cap M \neq \emptyset$. Apply the Blocking Lemma to μ' and get m, w with (m, w) blocking μ' for $\succ$. Show that $m \notin C$. (Hint: $C \cap M \subseteq M'$.)

(d) Show that $w \notin C$. (Hint: Let $w = \mu'(m'')$ with $m'' \in M'$. Since $w \succ_{m''} \mu_M(m'')$ and μ_M is stable, what does w think of $\mu_M(w)$ and m''? If $w \in C$, compare with $\mu_W(w)$.)

(e) Conclude that (m, w) also blocks μ' for $\succ'$, a contradiction. Then treat the case $C \cap M = \emptyset$, so $C \subseteq W$, by symmetry. This proves the theorem.

(f) *Consequences.*

i. Show that for any $\succ'$ obtained by one man m changing his report, the man-proposing DA gives him an outcome that is not strictly better than $\mu_M(m)$. So honesty is a dominant strategy for men under μ_M. More generally, show that no coalition of men can all strictly benefit.

ii. Why does the theorem say nothing useful for women when we use the men-proposing algorithm? (Compare with warm-up 1c.)

iii. *No stable mechanism is honest for both sides (Roth 1982).* A *stable mechanism* is any rule that maps every profile to a stable matching for it. Take $M = \{m_1, m_2\}$, $W = \{w_1, w_2\}$ with $m_1 : w_1\, w_2$, $m_2 : w_2\, w_1$, $w_1 : m_2\, m_1$, $w_2 : m_1\, m_2$. Show that this instance has exactly two stable matchings, $\mu_M = \{m_1 w_1, m_2 w_2\}$ and $\mu_W = \{m_1 w_2, m_2 w_1\}$. If a mechanism outputs μ_M on this instance, show that w_1 can profit by declaring m_1 unacceptable. If it outputs μ_W, show that m_1 can profit by declaring w_2 unacceptable. Conclude.

7. **Indifference.** Now allow ties in the lists, as in warm-up 1d. A matching is *weakly stable* if there is no pair of people who both *strictly* prefer each other to their partners, *strongly stable* if there is no pair where one strictly prefers the other and the other likes the first at least as much as their partner, and *super stable* if there is no pair where both like each other at least as much as their partners.

   (a) *Marriage: ties are harmless for existence.* By warm-up 1d every instance has a weakly stable matching. Prove the converse: a matching μ is weakly stable if and only if there is a way of breaking ties such that μ is stable for the resulting strict profile. (Hint: For one direction, break the ties so that $\mu(p)$ is ranked first among the people tied with it.)

   (b) *Sizes can differ.* Let $m_1 : (w_1\ w_2)$ (a tie), $m_2 : w_1$, $w_1 : m_1\ m_2$, $w_2 : m_1$. Find two weakly stable matchings of different sizes, and show how the two tie-breakings of m_1 produce them. For strict lists (even incomplete ones) all stable matchings match the same people; this is the Rural Hospitals theorem, which we do not prove. (It is NP-hard to find a largest weakly stable matching; Iwama, Manlove, Miyazaki, and Morita 1999.)

   (c) *Strong and super stability may not exist.* Let $m_1 : (w_1\ w_2)$, $w_1 : m_1$, $w_2 : m_1$. Show that there is no strongly stable and no super stable matching.

   (d) *Roommates: ties do matter.* Let $N = \{a, b, c, d\}$ with

   $$a : d\ b\ c, \qquad b : d\ (c\ a), \qquad c : b\ d\ a, \qquad d : c\ b\ a,$$

   where b is indifferent between c and a.

      i. Show that $\{ab, cd\}$ is weakly stable.

      ii. Break the tie as $b : d\ a\ c$ and as $b : d\ c\ a$. Show that for the first the matching $\{ab, cd\}$ is stable, and that for the second there is no stable matching at all. (Hint: For the second, check the three perfect matchings; why can the matchings with unmatched agents be dismissed?)

      iii. Prove the analogue of (a) for roommates: a matching is weakly stable if and only if it is stable for some tie-breaking. Why does this not give an algorithm, unlike marriage? (Deciding whether a roommates instance with ties has a weakly stable matching is NP-complete; Ronn 1990.)

8. **Many-to-one and many-to-many.** *College admissions.* Students S each have a list $L(s)$ of colleges. Colleges $c \in C$ have a capacity $q_c \geq 1$ and a list $L(c)$ of students. An assignment μ (each student gets one college on their list that has them on its list, or none) respects the capacities. A pair (s, c) with $s \in L(c)$ and $c \in L(s)$ and $\mu(s) \neq c$ is *blocking* if s prefers c to $\mu(s)$ (or s is unassigned) and either c has fewer than q_c students or c prefers s to at least one student assigned to it. An assignment is *stable* if it has no blocking pair.

   (a) *Cloning.* Replace every college c by q_c copies $c^1, \ldots, c^{q_c}$ with capacity 1, all having the list $L(c)$, and replace c in every student's list by $c^1\ c^2 \cdots c^{q_c}$ (in this order and consecutively). This is a one-to-one instance.

      i. Show that if μ is stable for the cloned instance and we map every student to the college of their copy, we get a stable assignment. (Hint: Given a blocking pair in the original instance, find a blocking pair in the cloned one.)

      ii. Show that every stable assignment arises this way. (Hint: For each college c, give the best student to c^1, the second best to c^2, and so on.)

iii. Conclude that DA, where every college keeps its q_c best proposers, outputs a stable assignment that every student likes at least as much as any other stable assignment, and that it is not profitable for a student to misreport (use Problem 6).

(b) *Many-to-many.* Now both sides have quotas: each man m has a quota q_m and each woman w a quota q_w, and lists as before. A matching is a set of mutually acceptable pairs such that nobody exceeds their quota (a pair can occur at most once). A pair $(m, w) \notin \mu$ is *blocking* if both find each other acceptable and each of them either has fewer partners than the quota or prefers the other to at least one of their current partners. The matching is *pairwise stable* if there is no blocking pair. The algorithm: while some man m has fewer than q_m partners and has not yet proposed to everyone on his list, he proposes to the next woman on his list; she accepts if he is acceptable to her, and if she then has more than q_w proposals, she rejects the one she likes least.

i. Show that the algorithm terminates after at most $\sum_m |L(m)|$ proposals.

ii. Show that its output is pairwise stable. (Hint: Suppose (m, w) blocks. Why did m propose to w? Why did w reject him? What can you say about the set of partners of w after that?)

(c) *Cloning fails.* Let $M = \{m\}$ with $q_m = 2$ and $L(m) = w_1\, w_2$, and let $W = \{w_1, w_2\}$ with $q_{w_1} = 2$, $q_{w_2} = 1$, and $L(w_1) = L(w_2) = m$. Show that the unique pairwise stable matching is $\{(m, w_1), (m, w_2)\}$. Clone as in (a) (cloning both sides, and replacing every person on a list by their copies in order). Show that the cloned instance has a stable matching, that in every stable matching of it, m^1 and m^2 are matched to the two copies of w_1, and conclude that mapping back gives a matching that is not pairwise stable. Which feature of many-to-one made (a) work?

9. **Correctness of the roommates algorithm.** In this problem we prove that the algorithm from class (Irving 1985) is correct, where lists may be incomplete. We first give the full algorithm. For a list $T(x)$ we write $\text{first}(x), \text{second}(x), \text{last}(x)$ for its first, second, and last entry. *Deleting the pair* $\{x, y\}$ from a collection of lists means removing y from $T(x)$ and x from $T(y)$.

Algorithm. The input is an instance L.

- *Phase 0.* Delete every pair $\{x, y\}$ with $y \in L(x)$ but $x \notin L(y)$. Call the new instance I; acceptability in I is mutual.
- *Phase 1.* Every agent proposes to the agents on its list, in order, simultaneously being a proposer and an accepter. Precisely: every agent y holds at most one proposal. While there is an agent x whose proposal is not currently held and who has not yet proposed to everyone on their list, x proposes to the next agent y on their list. If y holds no proposal, or y prefers x to the proposer y currently holds, y holds x's proposal (and releases the previous one, whose proposer must continue proposing); otherwise y rejects x. At the end, *reduce*: for every y who holds a proposal of x, delete the pairs $\{y, z\}$ for all z ranked below x by y, and delete the pairs $\{x, z\}$ for all z ranked above y by x. Call the resulting table T (the lists $T(x)$ are sublists of the lists of I, in the same order). An agent with $T(x) = \emptyset$ is *out*.
- *Phase 2.* While some list of T has at least 2 entries: choose an *all-or-nothing (AoN) cycle*, that is, distinct agents $a_1, \ldots, a_r$ ($r \geq 2$) with $\text{second}(a_i) = \text{first}(a_{i+1})$ for all i (indices modulo r). Put $b_i = \text{first}(a_i)$, so $b_{i+1} = \text{second}(a_i)$. Let D be the set of pairs

$$\{a_i, b_i\} \quad \text{and} \quad \{b_{i+1}, z\} \text{ for every } z \text{ ranked below } a_i \text{ in } T(b_{i+1}), \qquad i = 1, \ldots, r$$

(all computed before deleting anything), and delete all pairs in D simultaneously from T. If some list that was nonempty before this step is now empty, stop and report that *there is no stable matching.*

- *Output.* Output $\mu = \{\{x, y\} : T(x) = \{y\}\}$.

Intuitively, each a_i gives up b_i and moves to b_{i+1}, who now holds a_i and rejects everybody worse. We are going to prove that the algorithm outputs a stable matching when one exists and otherwise reports so. Here are the facts from class that we will use freely. Phase 0 does not change the stable matchings (warm-up 1e). After Phase 1, every stable matching of I is *inside* T, which means that every matched pair $\{x, \mu(x)\}$ is in T (Claim 1). The table T is symmetric. If $T(x) \neq \emptyset$ then $T(x) \ni y$ is x's first choice if and only if x is y's last choice. If some $|T(x)| \geq 2$, then there is an AoN cycle (Claim 4).

A table T (a family of sublists of the lists of I) is *good* if

(S) $y \in T(x)$ if and only if $x \in T(y)$;

(F) for all x, y with $T(x) \neq \emptyset$: $\text{first}(x) = y$ if and only if $\text{last}(y) = x$ (in particular, $T(y) \neq \emptyset$);

(J) for every pair $\{x, y\}$ that is mutually acceptable in I but not in T, one of the two, say u, has $T(u) \neq \emptyset$ and ranks every member of $T(u)$ above the other agent v.

By class and warm-up 1f, the table after Phase 1 is good, and every stable matching of I is inside it.

Let T be a good table and let μ be a stable matching of I that is inside T.

(a) *Everyone with a list is matched.* Show that every x with $T(x) \neq \emptyset$ is matched in μ. (Hint: Suppose x is unmatched and let $z = \text{last}(x)$. What is $\text{first}(z)$, and who is blocking?)

(b) *All or nothing (Claim 5).* Let $a_1, \ldots, a_r$ be an AoN cycle in T with $b_i = \text{first}(a_i)$.

  i. Show that either $\mu(a_i) = b_i$ for all i or for no i. (Hint: Suppose $\mu(a_i) = b_i$. By (F), $a_i = \text{last}(b_i)$. Since $b_i = \text{second}(a_{i-1})$, we have $a_{i-1} \in T(b_i)$, so b_i ranks a_{i-1} above a_i. Where is $\mu(a_{i-1})$ in the list of a_{i-1}? Use (a) and the blocking pair $\{a_{i-1}, b_i\}$.)

  ii. Show that if μ contains a pair of D, then $\mu(a_i) = b_i$ for all i. (Hint: A pair $\{b_{i+1}, z\} \in D$ in μ is $\mu(b_{i+1}) = z$. Where can $\mu(a_i)$ be if $\{a_i, b_{i+1}\}$ is not blocking?)

(c) *The shift.* Suppose $\mu(a_i) = b_i$ for all i.

  i. Show that $a_1, \ldots, a_r$ and $b_1, \ldots, b_r$ are disjoint sets. (Hint: Suppose $a_j = b_k$. What does (F) say about $\text{last}(a_j)$, and what is $\mu(a_j)$?)

  ii. Define μ' by matching a_i to b_{i+1} for all i and leaving everybody else as in μ. Show that μ' is inside $T \setminus D$ (the table where all pairs in D are deleted).

  iii. Show that μ' is a stable matching of I. (Hint: First, if $\{x, y\}$ is mutually acceptable but not in T, use (J) and (9a) to show it does not block. Then, if $\{x, y\}$ is in T and $x = a_i$: a_i got $\text{second}(a_i)$, so the only agent that a_i prefers is b_i; what does b_i think of a_{i-1} versus a_i? Finally, the b's are all better off than in μ.)

(d) *Elimination keeps a stable matching.* Let $T' = T \setminus D$. Show that if I has a stable matching inside T, then it has one inside T', and no nonempty list of T becomes empty in T'. (Hint: Combine (9b) and (9c); then use (9a).)

(e) *Elimination keeps the table good.* Now assume that no nonempty list of T becomes empty in T'. We show that T' is good.

i. Show that the a_i are distinct and the b_i are distinct. Show that $\{a_i, b_{i+1}\} \notin D$ for all i. (Hint: If $\{a_i, b_{i+1}\} = \{a_j, b_j\}$ then $a_i = b_j$ and $b_{i+1} = a_j$ (why?); use (F) to show that $T(a_i) = \{b_i, b_{i+1}\}$ and that $T'(a_i) = \emptyset$. If $\{a_i, b_{i+1}\} = \{b_{k+1}, z\}$ with z ranked below a_k in $T(b_{k+1})$ then $b_{k+1} = a_i$ and $z = b_{i+1}$ (why?); show $a_k = b_i$ and again $T'(a_i) = \emptyset$.)

ii. Show (S) and (J) for T'. (Hint: For a pair deleted now, use b_i (for $\{a_i, b_i\}$) or b_{i+1} (for $\{b_{i+1}, z\}$) as the agent in (J), and use that a_i is the last entry of $T'(b_{i+1})$. For earlier deleted pairs, use (J) for T and the fact that the justifying agent still has a nonempty list.)

iii. Show (F) for T'. (Hint: Show that the first entry of $T(x)$ changes only for $x \in \{a_1, \ldots, a_r\}$, to b_{i+1} for $x = a_i$, and that the last entry of $T(y)$ changes only for $y \in \{b_1, \ldots, b_r\}$, to a_i for $y = b_{i+1}$. For the first claim: if $\{x, \text{first}(x)\} \in D$ with $x \notin \{a_j\}$, then either $\text{first}(a_j) = b_j = x$ and $\text{first}(x) = a_j$, which makes $T(a_j)$ a single entry by (F); or $\text{first}(x) = z$ is ranked below some a_k, which is impossible; or $\text{last}(b_{k+1}) = x$, which means $x = a_{k+1}$.)

(f) *Correctness.* Prove the following.

i. The algorithm terminates, after at most $\mathcal{L}/2$ eliminations.

ii. If the algorithm outputs μ, then μ is a stable matching of L. (Hint: Show by induction that all tables are good, using (9e); then use warm-up 1f(b). Note that in this case no list became empty during Phase 2.)

iii. If L has a stable matching, then the algorithm outputs one. (Hint: Show by induction that there is a stable matching inside the current table, and use (9d).)

iv. If the algorithm reports that there is no stable matching, then there is none. In particular, the algorithm also decides existence.

(g) *Running time.* Show that a direct implementation takes time $O(n\mathcal{L}) = O(n^3)$. (Hint: By (9fa) there are at most $\mathcal{L}/2$ eliminations. How long does it take to find an AoN cycle (follow the sequence $p \mapsto \text{last}(\text{second}(p))$ from some p with $|T(p)| \geq 2$) and to delete the pairs of D?) (Although I'm not asking for it, with doubly linked lists and pointers, one can implement it in time $O(\mathcal{L})$.)

References

- Gale and Sotomayor. 1985. Some remarks on the stable matching problem. *Discrete Applied Mathematics* 11(3), 223–232.
- Demange, Gale, and Sotomayor. 1987. A further note on the stable matching problem. *Discrete Applied Mathematics* 16(3), 217–222.
- Dubins and Freedman. 1981. Machiavelli and the Gale–Shapley algorithm. *American Mathematical Monthly* 88(7), 485–494.
- Gale and Shapley. 1962. College admissions and the stability of marriage. *American Mathematical Monthly* 69(1), 9–15.
- Gusfield and Irving. 1989. *The Stable Marriage Problem: Structure and Algorithms.* MIT Press.
- Hwang. 1986. Comment on "Some remarks on the stable matching problem". (See also Roth and Sotomayor, below.)

- Irving. 1985. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms* 6(4), 577–595.
- Iwama, Manlove, Miyazaki, and Morita. 1999. Stable marriage with incomplete lists and ties. In *Proceedings of ICALP*, 443–452.
- Ng and Hirschberg. 1990. Lower bounds for the stable marriage problem and its variants. *SIAM Journal on Computing* 19(1), 71–77.
- Ronn. 1990. NP-complete stable matching problems. *Journal of Algorithms* 11(2), 285–304.
- Roth. 1982. The economics of matching: stability and incentives. *Mathematics of Operations Research* 7(4), 617–628.
- Roth and Sotomayor. 1990. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*. Cambridge University Press.