Problem Set 7

Deliverable: Submit your responses as a single, readable PDF file on the collab site before **6:29pm** on **Friday, 28 October**.

Collaboration Policy - Read Carefully

The collaboration policy is identical to that for PS6: you should work in groups of *one* to *four* students of your choice with no restrictions, and follow the rest of the collaboration policy from PS3.

Preparation

This problem set focuses on Stable Matching — Section 6.4 of the MCS book, and Class 15.

Directions

Solve as many of the 7 problems as you can. Questions 1 and 2 check your understanding of stable matching, and the remaining questions build up to completing the proof that the Gale-Shapley Algorithm always finds a stable matching. For maximum credit, your answers should be correct, clear, well-written, and convincing.

Stable Matching

Traumatized by his failure to reach Eve in Problem Set 6, Wall-E has decided to join RoboMatch.com, the Stable Macthing Service for Loney Robots. Robots do not have limitations on their potential matches, but RoboMatch is concerned that it may not be possible to find a stable matching if the robots are allowed to prefer any other robot, RoboMatch divides robots into two sets based on whether or not they are in Disney/Pixar movies, and requires every member to provide a full preference ranking of all the robots in the other set.

The rankings are (from most preferred to least preferred):

Set A: Disney/Pixar Robots	**Set B:** Real Robots
Luxo: AIBO, Big Dog, Junior	AIBO: Wall-E, Luxo, R2-D2
R2-D2: AIBO, Junior, Big Dog	Big Dog: R2-D2, Luxo, Wall-E
Wall-E: Big Dog, Junior, AIBO	Junior: Wall-E, R2-D2, Luxo

1. Show that there are stable matchings where Wall-E is paired with Junior and where Wall-E is paired with Big Dog. (Note: feel free to use the provided Gale-Shapley program code.
2. Show that there cannot be any stable matching where Wall-E is paired with AIBO.

Gale-Shapley Algorithm

In Class 15, we developed a state machine model for the Gale-Shapley stable matching algorithm:

$$
\begin{aligned}
S = \{&(\mathit{pairings}, \mathit{proposals}) \mid \\
&\mathit{pairings} = \{(a,b) \mid a \in A, b \in B, \text{no duplicate occurences of } a \text{ or } b \text{ in } \mathit{pairings}\}, \\
&\mathit{proposals} = (r_1, r_2, \cdots, r_n), 0 \leq r_i \leq n\} \\
G = \{&(\mathit{pairings}, \mathit{proposals} = (r_1, r_2, \cdots, r_n)) \rightarrow (\mathit{pairings}', \mathit{proposals}') \mid \\
&i \in \{1, \cdots, n\}, r_i < n \\
&b_x ::= \text{the } r_i\text{-ranked choice for } a_i \\
&\mathit{proposals}' = (r_1, \cdots, r_i + 1, \cdots, r_n) \\
&\mathit{pairings}' = \begin{cases} \mathit{pairings} \cup \{(a_i, b_x)\} & \forall a_z \in A.\, (a_z, b_x) \notin \mathit{pairings} \text{ (Case 1)} \\ \mathit{pairings} \cup \{(a_i, b_x)\} - \{(a_z, b_x)\} & \exists a_z \in A.\, (a_z, b_x) \in \mathit{pairings} \wedge a_z \prec_{b_x} a_i \text{ (Case 2)} \\ \mathit{pairings} & \text{otherwise (Case 3)} \end{cases} \\
q_0 = (&\mathit{pairings} = \{\}, \mathit{proposals} = (0, 0, \ldots, 0))
\end{aligned}
$$

3. Explain (in simple English) when (Case 3) in the definition of G occurs.
4. Prove that the state machine always terminates (we did this in Class 15, but didn't write out a full proof). (Hint: Explain why we need the $r_i < n$ constraint for G.)
5. Prove that the state machine always terminates in a state where *pairings* includes a match for each element of B. (Hint: prove by contradiction by showing that if the machine is in a reachable state where some element of B is unmatched, it must have a transition.)

Define, $\mathit{Prefs}_x(n)$ as the set of the top ranked n preferences of x. So, $\mathit{Prefs}_{a_i}(0) = \{\}$, and $\mathit{Prefs}_{a_i}(1) = \{b_x\}$ where b_x is a_i's top choice, $\mathit{Prefs}_{a_i}(2) = \{b_x, b_y\}$ where b_y is a_i's second choice, and $\mathit{Prefs}_{a_i}(n) = B$.

6. ($\star$) Prove that the property P defined below is a *preserved invariant* for the state machine. The value of *proposals*[i] is the count in the sequences *proposals* that corresponds to the number of *proposals* made by a_i.

$$
\begin{aligned}
&P(q = (\mathit{pairings}, \mathit{proposals})) ::= \\
&\qquad (a_i, b_j) \in \mathit{pairings} \implies \\
&\qquad\qquad \forall b_x \in \mathit{Prefs}_{a_i}(\mathit{proposals}[i]) \wedge b_j \prec_{a_i} b_x\,.\ \exists a_y \in A - \{a_i\}.\, (a_i \prec_{b_x} a_y)
\end{aligned}
$$

   That is, if (a_i, b_j) is in *pairings*, then every match a_i prefers to b_j prefers someone else over a_i.

7. ($\star\star$) Prove the *pairings* for the terminating state are a stable matching (as defined in Class 15). (You may assume all of the properties in questions 4–6 in your proof, even if you were not able to prove them.) (Note: you may first want to state the definition of a stable matching in a different way, and may also find it helpful to strengthen the preserved invariant property from question 6.)

