CSE 331: Introduction to Algorithm Analysis and Design

Gale-Shapley Algorithm

1 Stable Marriage Problem

Input

- A set of n men: $M = \{m_0, m_1, \ldots, m_{n-1}\}$
- A set of n women: $W = \{w_0, w_1, \ldots, w_{n-1}\}$
- A preference list for each man and woman: ex. $P_{m_i} = (w_{12}, w_5, \ldots, w_6)$
- define $P_{w_i}(m_j)$ to be the index of m_j in w_i's preference list
- define $P_{m_i}(w_j)$ to be the index of w_j in m_i's preference list

  *Note that a smaller index is a stronger preference. Specifically, $P_w(m) < P_w(m')$ means that w prefers m over m'.

Output

- A stable matching of men and women: $S \subseteq M \times W$
- S is a stable matching if all of the following are true:

  $|S| = n$

  $\forall_{m \in M} \exists_{w \in W}((m, w) \in S)$

  $\forall_{w \in W} \exists_{m \in M}((m, w) \in S)$

  $\neg \exists_{(m,w) \notin S}((m, w') \in S \wedge (m', w) \in S \wedge (P_m(w) < P_m(w')) \wedge (P_w(m) < P_w(m')))$

2 Gale-Shapley Algorithm

1. Begin with every man and woman being free
2. While there exists a free woman w, she proposes to her most preferred man m whom she hasn't proposed to yet as follows:
   (a) w proposes to m
   (b) If m is free: m and w get engaged and (m, w) is added to S
   (c) If m is currently engaged to another woman, w' and he prefers w' over w, $P_m(w) > P_m(w')$: w remains free
   (d) If m is currently engaged to another woman, w' and he prefers w over w', $P_m(w) < P_m(w')$: m breaks off the engagement with w' and gets engaged to w. (m, w') is removed from S and (m, w) is added to S. w' becomes free.
3. return S as a stable matching.

3 Proof of Correctness

Observation 1. *Once a man gets engaged, he becomes engaged to better women and never becomes free again.*

Observation 2. *Women keep getting engaged to worse men or stay engaged to the same man.*

Lemma 1. *The output of the Gale-Shapley algorithm is always a perfect matching*[1].

Proof. This will be a proof by contradiction using the Pigeon-Hole Principle.

- **Assume** S is not a perfect matching.

Since S is not a perfect matching, there must exist a woman, w, who is free at the end of the algorithm. Since the algorithms always chooses a free woman to propose to her most preferred man whom she hasn't yet proposed to, she must have proposed to all n men and still remained free. This implies that she was either rejected or left by every man. Both of these can only happen if a man is either already engaged, or left to get engaged to another woman. This means every man was engaged and by Observation 1, they remain engaged. Therefor, there are n men who are engaged to the other $n-1$ women (since w is free) which is a contradiction since at least one woman would have to be engaged to multiple men which can't happen in the algorithm.

- **Contradiction:** There are n men engaged to $n-1$ woman.

□

Lemma 2. *The output of the Gale-Shapley algorithm has no instabilities.*

Proof. This will also be a proof by contradiction. We will assume there is an instability in S and show that this leads to a contradiction.

- **Assume** S has an instability. Specifically, there exists men and women m, w, m', and w' such that all the following are true:

  $(m, w) \notin S$

  $(m, w') \in S$

  $(m', w) \in S$

  $P_m(w) < P_m(w')$

  $P_w(m) < P_w(m')$.

With this assumption, m and w would get engaged and break off their engagements with w' and m'. For this proof, we will reason through what would have to happen to cause this instability and show that all cases lead to a contradiction.

We will first observe that w must have proposed to m. This follows because w ended up engaged to m' whom she prefers less than m. The only way she could propose to m', she would have to propose to every man she prefers more than him which includes m.

[1] Recall that a perfect matching is a matching of size n

When w proposed to m, there are only two possible outcomes. Either m said yes or no. We will show that both answers lead to a contradiction of Observation 1.

Yes: The pair (m, w) was in S after the proposal. Since $(m, w) \notin S$, m must have broken off this engagement for a woman he prefers more than w. We know that m ended up with w' and $P_m(w) < P_m(w')$ which implies that at some point in the algorithm, m got engaged to woman he prefers less than his engagement at the time which is a contradiction of Observation 1.

No: For m to say no to w, he must have already been engaged to a more preferable woman, w''. Since m ends up being engaged to w' whom he prefers less than w and, by transitivity, less that w'', he must have gotten engaged to a less preferred woman which is a contradiction of Observation 1 by the same reasoning as the previous example.

- **Contradiction:** Either w didn't propose to m or m changed engagements to a less preferred woman, both of which contradiction the algorithm.

□

Theorem 1. *The Gale-Shapley algorithm always outputs a stable matching.*

Proof. By Lemma 1 and Lemma 2, the output of the Gale-Shapley algorithm is perfect matching with no instabilities. By definition, it is a stable matching. □

4 Runtime Analysis

The algorithm begins with $\Theta(n^2)$ time to setup the needed data structures, then runs the main proposal loop until there are no more free women. The remaining runtime analysis will be for this loop.

Lemma 3. *There are no more that n^2 proposals during a run of the Gale-Shaply algorithm.*

Proof. No woman proposes to a man more than once. If the all n women propose to all n men, there would be n^2 proposals. Since no woman proposes to a man more than once, there can't be any additional proposals. □

Lemma 4. *Each proposal can be performed in $O(1)$ time.*

Proof. This can be achieved by using certain data structures to maintain the needed information.

Comparing preferences could take $O(n)$ time if we only use the preference lists, but by storing an array of preferences indexed by people this can be performed in constant time. Setting up this array took $\Theta(n^2)$ time during setup, but results in a faster runtime within this loop. It is common in algorithm design to take more time on setup which is executed once to save time on steps that are executed many times such as the body of a loop. □

Since the setup takes $\Theta(n^2)$ and there are at most n^2 proposals each taking $O(1)$ time, the Gale-Shapley algorithm has a runtime of $O(n^2)$.