Theory of Algorithms Class Notes

Calvin Walker

Lecture 1: Gale-Shapley Algorithm for Stable Matching

Stable Matching Problem: There are two groups of people: $A = \{a_1 \dots a_n\}$ and $B = \{b_1 \dots b_n\}$, and we want to find a stable matching between A and B such that there is no pair of people a_i, b_j who would rather be with each other than their current partners. i.e We want to find a permutation π such that there are no $i, j \in [n]$ for whom

1. $\pi(i) \neq j$
2. a_i prefers b_j to $b_{\pi(i)}$
3. b_j prefers a_i to $b_{\pi^{-1}(j)}$

Gale-Shapley algorithm: The people in group A make offers in the order of their preference lists. When b_j receives an offer from a_i, if b_j is unmatched or preferes a_i to their current partner, they accept and become partners with a_i. Otherwise, b_j rejects and a_i moves onto the next person.

Properties:

- The Gale-Shapley algorithm terminates in $O(n^2)$ steps and results in a stable matching.

  Proof: There are n^2 possible offers and each offer is made at most once.

  Oberve that if some a_i makes an offer to their last choice b_j, then all people except b_j must be unavailable for a_i and thus already matched. There are only $n - 1$ other people in A, so b_j must be unmatched, and b_j accepts a_i's offer, and the algorithm terminates.

  Assume there are $i, j \in [n]$ such that a_i and b_j prefer each other to their current partners. Let b_j be a_i''s current partner and b_j' be matched with a_i. Then b_j is available to a_i. However, since a_i is matched with b_j', b_j must have been unavailable to a_i at some point, a contradiction.

- As Gale-Shapley progresses, (1) people in B only become happier, and (2) if b_j becomes unavailable to a_i they never become available again.

  Proof: (1) b_j only breaks off from its match if it prefers the new a_i. (2) Assume b_j becomes unavailable to a_i and then becomes available again. Let a_i' be b_j's partner when b_j became unavailable to a_i. Let a_i'' be b_j's partner when b_j became available to a_i again. Then, b_j prefers a_i' to a_i and a_i to a_i''. However, b_j must prefer a_i'' to a_i', a contradiction.