Presentation for use with the textbook, Algorithm Design and Applications, by M. T. Goodrich and R. Tamassia, Wiley, 2015

The Stable Marriage Problem

Trees with snow on branches, "Half Dome, Apple Orchard, Yosemite," 1933. Ansel Adams. U.S. government image. U.S. National Archives and Records Administration.

The Coupon Collector Problem

- Imagine that there is a set, C, of n coupons and we are interested in collecting at least one of every coupon in C.
- We can go a ticket window once a day and request a coupon, and a clerk will choose one of the n coupons at random and give it to us.
- The **coupon collector problem** is to determine the number of times we need to go to the ticket window before we have collected all n coupons.

Analysis of Coupon Collecting

- Let X be a random variable representing the number of times that we need to visit the ticket window before we get all n coupons.
- We can write X as follows:
  - $X = X_1 + X_2 + \ldots + X_n$,
- Here X_i is the number of trips we have to make to the ticket window in order to go from having $i - 1$ distinct coupons to having i distinct coupons.

More Analysis of Coupon Collecting

- $X_1 = 1$, since we are guaranteed to get a distinct coupon in our first trip to the ticket window.
- After we have gotten i −1 distinct coupons, our chance of getting a new one in any trip to the window is

$$p_i = \frac{n - (i-1)}{n}.$$

- This is because there are n coupons, but only n−(i−1) that we don't already have at this point in time.
- This implies that each X_i a **geometric random variable** with parameter, p_i.
- That is, if we imagine that we have a biased coin that comes up heads with probability p_i, then X_i is the number of times we have to flip this coin until we get it to come up heads.

Yet More Analysis of Coupon Collecting

- By linearity of expectation,

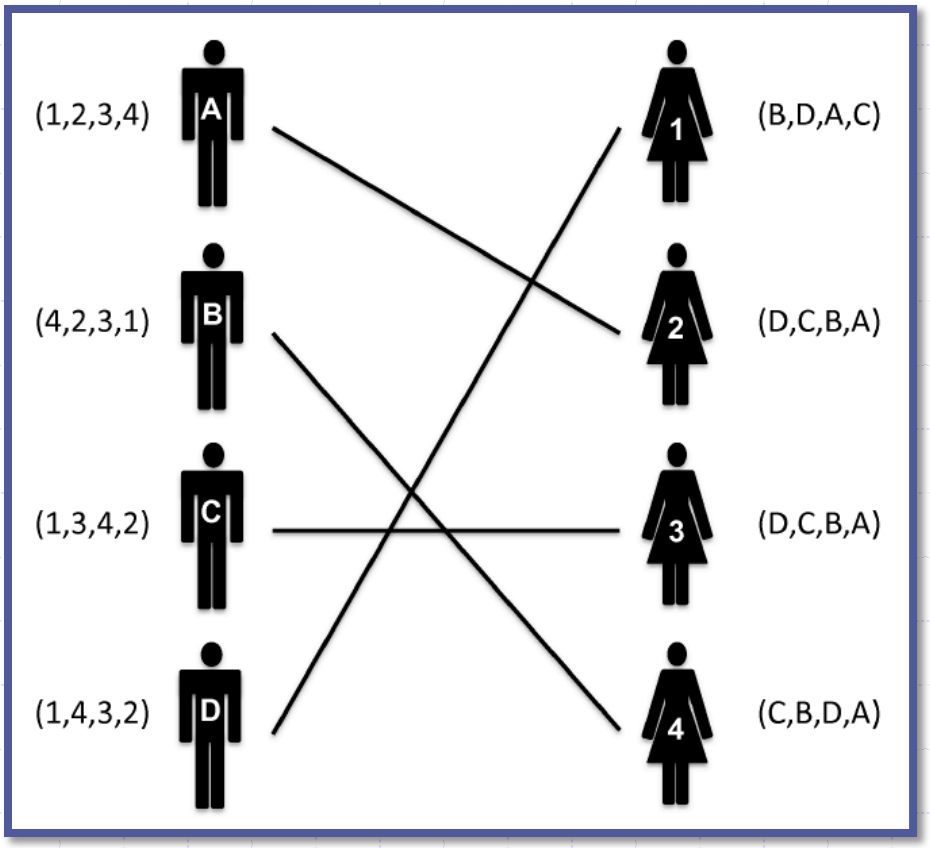
$$\begin{aligned} E[X] &= E[X_1] + E[X_2] + \cdots + E[X_n] \\ &= \frac{1}{p_1} + \frac{1}{p_2} + \cdots + \frac{1}{p_n} \\ &= \frac{n}{n} + \frac{n}{n-1} + \cdots + \frac{n}{1} \\ &= n \sum_{i=1}^{n} \frac{1}{i} \\ &= n\, H_n, \end{aligned}$$

- Here, H_n is the nth harmonic number, which is at most ln n + 1.
- Thus, the expected number of trips to the ticket window in the coupon collector problem is O(n log n).

The Stable Marriage Problem

- Imagine a village consisting of n men and n women, all of whom are single, heterosexual, and interested in getting married.
  - Every man has a list of the women ordered by his preferences, and, likewise, every woman has a list of the men ordered by her preferences.
- The **stable marriage problem** is to match up the men and women in a way that is stable.
- Such a matching is **stable** if there is no unmatched man-woman pair, (x, y), such that x and y would prefer to be married to each other than to their spouses.
  - That is, it would be unstable if x preferred y over his wife and y preferred x over her husband.

A Stable Marriage Example



- Each man and woman is listed with his or her preference list, and the matching shown is stable.
- Note that even though female 2 is married to her last choice, there is no man who prefers her over his current wife.

Application: Hospital Matching

- The stable marriage problem arises in in practice in the annual placement of residents in hospitals.
- Residents rank order the hospitals that they would like to work in, and hospitals rank the set of available residents for each of their available open slots.
- Then a stable "marriage" is computed between residents and available slots in hospitals.

The Proposal Algorithm

- There is a simple algorithm for solving the stable marriage problem, which involves men making proposals to women in a series of rounds.
- A round begins with an unmatched man making a proposal to the female highest-ranked on his list.
- If she is unmatched, then she accepts his proposal and the round ends. If, on the other hand, she is matched, then she accepts his proposal only if she ranks him higher than her current partner.
- In the case when the woman receiving the proposal is already matched, then whichever man she rejects repeats the computation for this round, making a proposal to the next woman on his list (his highest-ranked woman that he has not previously proposed to).

The Proposal Algorithm: Details

Algorithm StableMarriage(X, Y):

Input: A set, X, of n men and their prioritized lists of women in Y, and a set, Y, of n women and their prioritized lists of men in X
Output: A stable marriage for X and Y

for each man, x, in X **do**
    Let y be x's highest-ranked woman
    Have x propose to y
    while y is matched to some man, z, such that $z \neq x$ **do**
        if y prefers x over z **then**
            Match x and y
            Unmatch z
            Let $x \leftarrow z$
        Let y be x's highest-ranked woman he has not proposed to yet
        Have x propose to y
    Match x and y

Correctness

- To see that the matching resulting from this proposal algorithm is stable, suppose there is an unstable pair, (x, y), that are not matched by the algorithm, that is, both x and y prefer each other to the partners they end up with by following the proposal algorithm.
- Note that at the time x made his last proposal, to his current partner, w, he had made a proposal to every woman that he ranked higher than w.
- But this means that he would have made a proposal to y, which she would have accepted, because she ranks x higher than the man she ended up with.
- Thus, such a pair, (x, y), cannot exist.

Worst-Case Analysis

- The worst-case running time of this algorithm is $O(n^2)$.
  - To see this fact, note that at the beginning of the algorithm, the total length of the lists of all n men is $O(n^2)$, and at each step of the algorithm, some man is using up a proposal to a woman on his list to whom he will never again propose.
  - Thus, if we charge each entry in the list of preferences by the men 1 cyber-dollar for the work we do in having a man make a proposal, then we can pay for all the proposals using $O(n^2)$ cyber-dollars.

Average-Case Analysis

- For the sake of average-case analysis, suppose the preference list for every man is an independent random permutation of the list of women.
- Then, each time it is a man's turn to make a proposal, he is making that proposal to a random woman he has not proposed to before.
- We can simplify this analysis further so each time it is a man's turn to make a proposal, he makes his proposal to a random woman ignoring his previous proposals.
- Such an algorithm is **memoryless**, in the sense that each proposal a man makes is independent of any proposal he made earlier.

Average-Case Analysis, cont.

- The key observation to analyze the memoryless algorithm is to focus on the women and realize that each round in this algorithm consists of a sequence of proposals to independently chosen random women until a proposal is made to an unmatched woman.
- That is, the memoryless algorithm is an instance of the **coupon collector problem** where the names of the women are the coupons.
- Thus, by the analysis of the coupon collector problem, the expected running time of the memoryless stable marriage algorithm is **O(n log n)**.