1 Matching

A common problem is to find a *matching* on some set of items. Let items be represented by vertices in a graph and let allowed pairings of items be represented by (unweighted, undirected) edges in this graph, $G = (V, E)$. A *perfect matching* $M = (V', E')$ is a subgraph of G such that $V' = V$, i.e., every vertex in the input graph is also in the matching, the selected edges are a subset of the original edges $E' \subseteq E$, and if $(i, j) \in E'$ then no (i, k) or (j, k) can also be in the matching. That is, each vertex is paired with exactly on other vertex in the graph.

If G is weighted, i.e., we are also given some weight function $w : E \to \mathbb{R}$, then a *maximum matching* is a matching (not necessarily perfect) with maximum weight, where the weight of a matching is simply the sum of the matched edges. (If edges are unweighted, i.e., they have unit weight, then an maximum matching of cardinality $|V|$ is also a perfect matching.)

Now, consider the following problem. Exactly n individuals are engaged in a matching process, such that when the process ends, all individuals are paired with exactly one other individual. However, unlike in the perfect or maximum matching cases, in this version, each individual has its own set of *preferences* for whom they would prefer to be matched with. These preferences can be compactly represented by a vector containing a total ordering[1] of the $n - 1$ possible partners.

To illustrate some properties of such a problem, consider the following small example with Alice (A), Bob (B), Carol (C) and Daniel (D), whose preference lists are the following.

person	1st	2nd	3rd
Alice	**Bob**	Carol	Daniel
Bob	Carol	Daniel	**Alice**
Carol	Alice	Bob	**Daniel**
Daniel	**Carol**	Alice	Bob

If we choose the matching (Alice, Bob) and (Carol, Daniel), then we have a problem. Alice and Daniel are happy because they are matched with their first choices, but their partners Bob and Carol would prefer to be matched with each other rather than matched with their current partners.

These pairs are called *unstable* because two individuals who are not currently matched would prefer to be matched with each other over their current partners. If a matching contains no unstable pairs and all individuals are matched, we call it a *stable matching.* The rest of this lecture will focus on the Gale-Shapley algorithm for finding stable matchings.

[1]An ordering on n items such that there are no ties and every item is included in the list.

1.1 Stable Marriage Problem

The version of the stable matching problem considered by Gale and Shapley (1962) is called the "stable marriage problem," and assumes n heterosexual men and n heterosexual women; thus, the outcome of the algorithm should be a set of n stable "marriages" such that no pair of marriages is unstable.[2] As in our example above, each person submits a preference list of all of their possible matches, i.e., each man submits a rank-ordered list of the n women, and each woman submits a rank-ordered list of the n men. These lists are the input to the algorithm, and the output is a stable matching on the $2n$ individuals such that each man is paired with exactly one woman, and each woman is paired with exactly one man.

1.2 Gale-Shapley Algorithm

The underlying idea of the Gale-Shapley algorithm is to have one set of people (men, or women) do all the proposing of matches, starting with their most preferred partners and working their way toward less preferred ones, while the other set accepts or rejects matches, working their way from less preferred to more preferred partners.

More specifically, a man begins by proposing a match with his most preferred partner. If this woman is not yet matched, then she provisionally accepts and the pair is placed into the matching. On the other hand, if she is already matched, then she does one of two things: (i) if the man is lower ranked on her preference list than her current partner, then she rebuffs the proposal, otherwise (ii) she accepts the proposal and rejects her current partner. If the man's proposal is unsuccessful, he repeats the process, moving down his preference list. Once a man is matched, we move to the next unmatched man and start at the next woman down in his list (which, if the man was formerly matched but then became unmatched, is the woman just lower-ranked than his previous partner.) We repeat this process until all men are matched. A catchy way to remember the procedure is that "men propose, women dispose."[3]

Without yet going into the data structures for storing and managing the preference lists, the pseudocode for the Gale-Shapley algorithm is straightforward. Let $m \in M$ denote a man and $w \in W$ denote a woman.

[2]The first stable matching algorithm by Gale and Shapley was actually motivated by the college application process, in their article titled "College Admission and the Stability of Marriage." In later work, Gale and Shapley worked on a more general version of the problem, motivated by matching medical residents and hospitals. Shapley subsequently won the 2012 Nobel Memorial Prize in Economic Sciences, in part for his work on "the theory of stable allocations and the practice of market design," a direct reference to the field of algorithmic game theory, of which stable matchings is an early component.

[3]The reverse, where women propose and men dispose, is an equivalent algorithm, which we will visit shortly.

```
S = {}                              % set of pairs (m,w)
while some man m is free {
    w = getWoman(m)                 % choose woman w from m's list
    if w is free { add (m,w) to S } % m,w both free, match them
    else if (m',w) in S {           % w already matched with some m'
        if m > m' in w's list {     % but w prefers m to m'
            remove (m',w) from S    %    breakup (m',w) match
            add (m,w) to S          %    create (m,w) match
            set m' as free          %    mark m as free
}}}
return P
```

This algorithm is guaranteed to halt and upon termination, it returns a stable matching. Additionally, the order in which we consider unmatched men has no impact on the stable matching that is found. And finally, this version of the Gale-Shapley algorithm, in which men do all the proposing, is called "man optimal" because when the algorithm halts, men are matched with their highest-ranked woman, conditioned on the matching being stable. (f the women did all the proposing, then it would be the woman-optimal version of the algorithm.)

The man-optimal version, however, is not necessarily woman-optimal, and can produce a matching that is the worse of all possible outcomes for the women. Consider the following set of preferences. Notice, in particular, the first column of the man matrix, and the last column of the woman matrix.

man	1st	2nd	3rd
m_1	1	2	3
m_2	2	3	1
m_3	3	1	2

woman	1st	2nd	3rd
w_1	2	3	1
w_2	3	1	2
w_3	1	2	3

Following the above algorithm, m_1 proposes to w_1, who accepts because she is currently free. Man m_2 proposes to w_2, who accepts because she is currently free. Finally, man m_3 proposes to w_3, who accepts, because she too is currently free. The algorithm then halts because no men are free. This matching is stable because there are no unstable pairs, and each man is matched with his most preferred partner; however, each woman is matched with her least-preferred partner, the worst possible stable matching from the women's perspective.

Note that if we swapped the roles, and had women propose first, the resulting matching would be $\{(w_1, m_2), (w_2, m_3), (w_3, m_1)\}$. This matching is also stable, provides the women with their most-preferred partners, and the men with their least-preferred partners. We'll revisit this possibility below.

1.2.1 Running time

To derive the running time of the Gale-Shapley algorithm, first consider the experience of some woman w over the course of the algorithm. At the beginning of the algorithm, she is free and will become matched with the first man m who proposes to her. With each pass through the `while` loop, she may receive proposals from alternative men. However, she only ever changes partners if the new proposal is from a more preferred man in her preference list. Thus, the following property is true: a woman w remains matched from the time she receives her first proposal until the end of the algorithm, and the sequence of her partners is a monotonically increasing sequence on her preference list.

Now consider the experience of a man m over the course of the algorithm. Unlike the women, men may become free again, as a result of his current partner receiving a proposal from a more attractive man. However, because m begins by proposing to his most-preferred partner and works his way down his list, the sequence of his partners is a monotonically decreasing sequence on his preference list.

Given these insights, we can now prove that the algorithm terminates after at most n^2 iterations.

First, observe that no man can be rejected by all women. Assume that some man m has been rejected by all n women. Under the algorithm a free woman will not reject a man's proposal, i.e., only a matched woman can reject a man's proposal. Thus, if m has been rejected by all n women, then all n women must be already matched. However, a woman can only be matched to at most one man, implying that if m is free, then at most $n-1$ women are matched. thus, either at least one w must still be free and m cannot have been rejected by all n women.

Second, each iteration of the `while` loop involves exactly one proposal. Note that because men move monotonically down their preference lists, no man will propose to the same woman twice. Because no man can be rejected by every woman, in the worst case, a man will propose to every woman before becoming matched. Thus, the number of iterations of the `while` loop is at most n^2 before the algorithm halts, and when it halts, every man and woman is matched.

(At home exercise: what preference lists will generate the worst-case running time?)

1.2.2 Correctness

We now know the Gale-Shapley algorithm will halt. But it remains to be shown that it also produces a stable matching on every possible set of preferences, i.e., is correct. Let S denote the matching produced by the Gale-Shapley algorithm. We claim that S is always a stable matching.

Because the number of women and men are both n, when the algorithm halts, S is a perfect matching. Assume that within S there exists an unstable pair, denoted (m, w) and (m', w') such that for m, $w' > w$ and for w', $m > m'$. That is, m and w' would prefer each other over their current partners. (Note that the preferences of w and m' are irrelevant.)

Over the course of the algorithm, the last successful proposal by m must have been to w. This implies two possibilities for m and w'. First, if m did not propose to w' before proposing to w, then w must occur higher in m's preference list than w', which contradicts our assumption that for m, $w' > w$. Second, if m did propose to w' at some point before being matched with w, then he was rejected by w', who preferred her current partner m'' over m. Because w' was ultimately matched with m', either $m' = m''$ or $m' > m''$. The latter case further implies that $m' > m$ because w' rejected m in favor of m''. Thus, w' cannot prefer m to her partner m', and S must be a stable matching.

1.3 Implementing the algorithm

The preference lists themselves may be stored in two $n \times n$ matrices M and W. For the men, we let $M_{i,j}$ give the jth most preferred woman for man i, i.e., each row contains a man's list, in order of preference. For the women, let $W_{i.j}$ give the rank of man j in woman i's list. Note that these two matrices are storing the same information, but in a slightly different way.

If men are proposing, we must also store the index of the most recently proposed-to woman, which we may do in an array v of length n. Initially, we set all $v_i = 0$ to denote the fact that no men have proposed to any women. At some intermediate pass through the `while` loop, $v_i = j$ denotes that if i becomes free, the next woman he will propose to is $M_{i,j+1} = w'$. We must also store the matching itself; which we do using an adjacency list, one for the men and one for the women.

Now, consider the pseudo-code on page 3 and ignore for the moment that we have not yet specified how we store and maintain the set of free men. Given the above data structures, the function `getWoman(m)` takes time $O(1)$ because $v_m + 1$ is the index of the next-most-preferred woman in m's preference list, whom m has not already proposed to.

Let $w = M_{m,v_m+1}$, the next woman to whom m will propose. If w is free, then the adjacency list containing the women's partners will be empty for w, which takes $O(1)$ time determine. Adding the pair (m, w) to S takes $O(1)$ time to update these two adjacency lists.

Otherwise, if w is already matched, then m' is stored in w's adjacency list, and it remains only to determine whether $m > m'$ in w's list. This can be checked in $O(1)$ time by asking $W_{w,m} > W_{w,m'}$. If m is more preferred, removing the pair (m', w) from S and adding the pair (m, w) to S can be done in $O(1)$ time. Thus, all of the operations within the `while` loop take constant time.

1.3.1 S is independent of the order of execution

A nice property of the Gale-Shapley algorithm is that the stable matching S it produces is independent of the order in which men make their proposals. This property provides us with great flexibility in how we choose which free man to attempt to match next, and allows us to use a data structure that let's us choose a new free man in $O(1)$ time by simply storing the free men in a linked list and choosing a new free man either from the front or back of the list. Implementing the algorithm this way provides an overall running time of $O(n^2)$.[4]

Let's prove that the outcome of the algorithm is independent of the order of proposals. As before, the arguments are symmetric for men-as-proposers or women-as-proposers. Thus, without loss of generality, we will let the men do the proposing. To begin, some definitions:

Stable partner: some w for m such that there exists some stable matching S on all men and women with $(m, w) \in S$.

Best stable partner: some w for m, such that w is a stable partner and there exists no better woman $w' > w$ for m who is also a stable partner for m.

Claim: All possible orderings on proposals in the Gale-Shapley algorithm yield the same stable matching S and this matching is optimal for the proposers.

Proof: Assume there exists an ordering σ that produces a matching S in which a man m is matched with a woman w (i.e., $(m, w) \in S$), and assume that there exists some other stable matching S' in which m is paired with another woman w' (i.e., $(m, w') \in S'$), and furthermore that for m, $w' > w$. That is, both w and w' are stable partners for m, but w' is m's best stable partner.

Men propose to women in decreasing order of their preferences. Thus, over the sequence σ, to arrive at S, w' must have rejected m because w' would have been proposed to by m before proposing to w. Let this be the first such rejection for m during σ, and suppose that this rejection occurred because w' was already matched with some m'. Thus, for w', $m' > m$.

By symmetry, because this was the first time a man has been rejected by a stable partner, m' cannot have a stable partner whom he prefers to w'. Thus, for m', w' is preferred to his partner in M', and the matching M' is not stable because m' and w' would leave their current partners to be with each other (they are a "blocking pair").

[4]The best case running time is $O(n)$, in which each man is matched with a woman after a constant number of proposals. The average case, however, is not $O(n^2)$ but is instead $O(n \log n)$, implying that for most preference lists, a stable matching is found fairly quickly. A nice treatment of the average case can be found here: `http://bit.ly/15c3AXt` .

Thus, each man m is matched in M with his best stable partner, and because σ was chosen arbitrarily, all possible executions must lead to the same stable matching M. $\square$

1.4 Related problems

The stable matching (marriage) problem is closely related to a number of other problems, which can be viewed as variants on the stable matching problem.

- *Stable roommates*: this problem is almost identical to the stable marriage problem, except that all individuals are in a single pool.
- *Hospitals and residents* (or *college admissions*): in this version, hospital (college) i can be matched with k_i residents (applicants); the stability criterion is still enforced. As with the marriage version, the algorithms can be hospital-optimal or resident-optimal.[5]
- Sets with unequal size: instead of requiring that the number of women and men be equal, we allow for unequal set sizes. Results similar to the Gale-Shapley algorithm apply.

2 For Next Time

1. Crowd-source lectures begin in earnest

[5]The matching algorithm proposed by Gale and Shapley for this problem was resident optimal, while the version used by the National Resident Matching Program, the official market clearing mechanism for matching medical residents with hospitals, is hospital optimal.