CS 70 Discrete Mathematics and Probability Theory
Fall 2019 Alistair Sinclair and Yun Song Note 4

1 The Stable Marriage Problem

In the previous two notes, we discussed several proof techniques. In this note, we apply some of these techniques to analyze the solution to an important problem known as the *Stable Marriage Problem*, which we now introduce. The Stable Marriage Problem is one of the highlights of the field of algorithms.

Suppose you run a dating agency, and your task is to match up n men and n women. Each man has an ordered *preference list* of the n women, and each woman has a similar list of the n men. For example, consider the case of $n = 3$, i.e., three men Alex, Bob, and Charles, and three women Anita, Bridget, and Christine, with the following preference lists (lists are ordered from most favorable to least favorable):

Men	Women		
Alex	Anita	Bridget	Christine
Bob	Bridget	Anita	Christine
Charles	Anita	Bridget	Christine

Women	Men		
Anita	Bob	Alex	Charles
Bridget	Alex	Bob	Charles
Christine	Alex	Bob	Charles

What you would like to do as head of the dating agency is to pair up each man with a woman. For example, two possible pairings are {(Alex, Anita), (Bob, Bridget), (Charles, Christine)} and {(Alex, Bridget), (Bob, Christine), (Charles, Anita)}. However, you don't want just any old pairing! In order for your dating firm to be successful, you wish the pairing to make "everyone happy", in that nobody can realistically hope to benefit by switching partners. Can you do this efficiently?

It turns out that there is indeed an algorithm to achieve this; moreover, it's remarkably simple, fast, and widely-used. It's called the *Stable Marriage algorithm* (a.k.a. the Gale-Shapley algorithm), and we present it now.

2 The Stable Marriage Algorithm

Every Morning: Each man proposes to the most preferred woman on his list who has not yet rejected him.

Every Afternoon: Each woman collects all the proposals she received in the morning; to the man she likes best among these, she responds "maybe, come back tomorrow" (she now has him *on a string*), and to the others she says "never" (i.e., she rejects them).

Every Evening: Each rejected man crosses off the woman who rejected him from his list.

The above loop is repeated each successive day until each woman has a man on a string; on this day, each woman is paired with the man she has on a string and the algorithm terminates.

Let's digest the algorithm by running it on our example above. The following table shows which men propose to which women on the given day (the men in bold are "on a string"):

Days	Women	Proposals
1	Anita	**Alex**, Charles
	Bridget	**Bob**
	Christine	—
2	Anita	**Alex**
	Bridget	**Bob**, Charles
	Christine	—
3	Anita	**Alex**
	Bridget	**Bob**
	Christine	**Charles**

Thus, the algorithm outputs the stable pairing: {(Alex, Anita), (Bob, Bridget), (Charles, Christine)}.

Before analyzing the algorithm's properties further, let's take a moment to ask one of the first questions you should always ask when confronted with a new concept: Why study stable marriage in the first place? What makes it and its solution so interesting? For this, we discuss the very real impact of the Gale-Shapley algorithm on the *Residency Match* program.

3 The Residency Match

Perhaps the most well-known application of the Stable Marriage Algorithm is the residency match program, which pairs medical school graduates and residency slots (internships) at teaching hospitals. Graduates and hospitals submit their ordered preference lists, and the stable pairing produced by a computer matches students with residency programs.

The road to the residency match program was long and twisted. Medical residency programs were first introduced about a century ago. Since interns offered a source of cheap labor for hospitals, soon the number of residency slots exceeded the number of medical graduates, resulting in fierce competition. Hospitals tried to outdo each other by making their residency offers earlier and earlier. By the mid-40s, offers for residency were being made by the beginning of junior year of medical school, and some hospitals were contemplating even earlier offers — to sophomores! The American Medical Association finally stepped in and prohibited medical schools from releasing student transcripts and reference letters until their senior year. This sparked a new problem, with hospitals now making "short fuse" offers to make sure that if their offer was rejected they could still find alternate interns to fill the slot. Once again the competition between hospitals led to an unacceptable situation, with students being given only a few hours to decide whether they would accept an offer.

Finally, in the early 1950's, this unsustainable situation led to the centralized system called the National Residency Matching Program (N.R.M.P.) in which the hospitals ranked the residents and the residents ranked the hospitals. The N.R.M.P. then produced a pairing between the applicants and the hospitals, though at first this pairing was not stable. It was not until 1952 that the N.R.M.P. switched to the Stable Marriage Algorithm, resulting in a stable pairing.

Finally, if the above still hasn't convinced you of the worth of this algorithm, consider this: In 2012, Lloyd Shapley and Alvin Roth won the Nobel Prize in Economic Sciences by extending the Stable Marriage Algorithm. (The moral of the story? Computer science pays off.)

4 Properties of the Stable Marriage Algorithm

There are two properties we wish to show about the stable marriage algorithm: First, that it doesn't run forever, i.e., it halts; and second, that it outputs a "good" (i.e., stable) pairing. The former is easy to show; let us do it now.

Lemma 4.1. *The stable marriage algorithm always halts.*

Proof. The argument is simple: On each day that the algorithm does not halt, at least one man must eliminate some woman from his list (otherwise the halting condition for the algorithm would be invoked). Since each list has n elements, and there are n lists, this means that the algorithm must terminate in at most n^2 iterations (days). □

Next, we'd like to show that the algorithm finds a "good" pairing. Before we do so, let's clarify what we mean by "good".

4.1 Stability

What properties should a good pairing have? Perhaps we'd like to maximize the number of first ranked choices? Alternatively, we could minimize the number of last ranked choices. Or maybe it would be ideal to minimize the sum of the ranks of the choices, which may be thought of as maximizing the average happiness.

In this lecture we will focus on a very basic criterion: *stability*. A pairing is **unstable** if there is a man and a woman who prefer each other to their current partners. We will call such a pair a *rogue couple*. So a pairing of n men and n women is **stable** if it has no rogue couples. Let us now recall our example from earlier:

Men	Women		
Alex	Anita	Bridget	Christine
Bob	Bridget	Anita	Christine
Charles	Anita	Bridget	Christine

Women	Men		
Anita	Bob	Alex	Charles
Bridget	Alex	Bob	Charles
Christine	Alex	Bob	Charles

Sanity check! Consider the following pairing for the example above: {(Alex, Christine), (Bob, Bridget), (Charles, Anita)}. Why is this pairing unstable? (Hint: Alex and Bridget are a rogue couple — why?)

Here is a stable pairing for our example: {(Bob, Anita), (Alex, Bridget), (Charles, Christine)}. Why is (Alex, Anita) *not* a rogue couple here? It's certainly true that Alex would rather be with Anita than his current partner. Unfortunately for him, however, Anita would rather be with her current partner (Bob) than with Alex. Note also that both Charles and Christine are paired with their *least* favorite choice in this

matching. Nevertheless, this does not violate stability since none of their preferred choices would rather be with them.

Before we discuss how to find a stable pairing, let us ask a more basic question: Do stable pairings always exist? Surely the answer is yes, since we could start with any pairing and make it more and more stable as follows: If there is a rogue couple, modify the current pairing so that they are together. Repeat. Surely this procedure must result in a stable pairing? Unfortunately this reasoning is not sound! Why? Although pairing up the rogue couple reduces the number of rogue couples by one, it may also *create new* rogue couples. So it is not at all clear that this procedure will terminate!

Let's further illustrate the fallacy of the reasoning above by applying it to a closely related scenario known as the *Roommates Problem*. In this problem, we have $2n$ people who must be paired up to be roommates (the difference being that unlike in stable marriage, a person can be paired with any of the other $2n-1$ people, i.e., there is no concept of *gender*). Now, suppose our approach of iteratively matching up rogue couples indeed *did* work. Since this approach does not exploit the existence of different genders, we can apply it to the roommates problem. Thus, we could conclude that there must always exist a stable pairing for roommates. Wouldn't you be surprised then to read the following counterexample, which gives an instance of the roommates problem which does *not* have a stable pairing!

Roommates			
A	B	C	D
B	C	A	D
C	A	B	D
D	–	–	–

We claim that in this instance, there always exists a rogue couple for any pairing. For example, the pairing {(A,B), (C,D)} contains the rogue couple B and C. How about {(B,C), (A,D)}? This pairing is unstable because now A and C are a rogue couple.

Sanity check! Verify that in this example there is *no* stable pairing!

We conclude that any proof that there always exists a stable pairing in the stable marriage problem *must* use the fact that there are two genders in an essential way. In the next section we give such a proof, and a nifty one at that: We prove that there must exist a stable pairing because the stable marriage algorithm always outputs one!

4.2 Analysis

We now prove that the stable marriage algorithm always outputs a stable pairing. Why should this be the case? Consider the following intuitive observation:

Observation 4.1. *Each man begins the algorithm with his first choice being a possibility; as the algorithm proceeds, however, his best available option can only get* worse *over time. In contrast, each woman's options can only get* better *with time.*

Thus, as the algorithm progresses, each man's options get worse, while each woman's improves; at some point, the men and the women must "meet" in the middle, and intuitively such a pairing should be stable.

Let us formalize this intuition beginning with a formal statement of Observation 4.1 via the following lemma.

Lemma 4.2 (Improvement Lemma)**.** *If man M proposes to woman W on the kth day, then on every subsequent day W has someone on a string whom she likes at least as much as M.*

Proof. We proceed by induction on the day j, with $j \geq k$.

Base case ($j = k$): On day k, W receives at least one proposal (from M). At the end of day k, she will therefore have on a string either M or someone she likes more than M, since she chooses the best among her proposals.

Inductive Hypothesis: Suppose the claim is true for some arbitrary $j \geq k$.

Inductive Step: We prove the claim for day $j+1$. By the Induction Hypothesis, on day j, W had a man M' on a string whom she likes at least as much as M. (Note that M' may be M.) According to the algorithm, M' proposes to W again on day $j+1$ (since he hasn't yet been rejected by her). Therefore, at the end of day $j+1$, W will have on a string either M' or someone she likes more than M'; in both cases, she likes this person at least as much as M. Hence the claim holds for day $j+1$, completing the inductive step. □

Detour: The Well-Ordering Principle. Let's take a moment to consider an alternate proof of Lemma 4.2; in the process, we'll uncover a fundamental principle which is equivalent to induction.

Alternate proof of Lemma 4.2. As in our original proof, the claim certainly holds on day $j = k$. Suppose now, for the sake of contradiction, that the jth day for $j > k$ is the first counterexample where W has either nobody or some M^* inferior to M on a string. On day $j-1$, she has some man M' on a string and likes M' at least as much as M. According to the algorithm, M' still proposes to W on the jth day. So W has the choice of at least one man (M') on the jth day; moreover, her best choice is at least as good as M', and therefore certainly better than M^*. This contradicts our initial assumption. □

What proof technique did we use to prove this lemma? Is it contradiction? Or some other beast entirely? It turns out that this is *also* a proof by induction! Why? Well, we began by establishing the base case of $j = k$. Next, instead of proving that for all j, $P(j) \implies P(j+1)$, we showed that the *negation* of this statement is false, i.e., that "there exists j such that $\neg(P(j) \implies P(j+1))$" does not hold; thus, by the law of the excluded middle, it must indeed hold that for all j, $P(j) \implies P(j+1)$.

Sanity check! Ensure you understand why these two proofs are equivalent.

Let us now be a bit more careful — what exactly allowed us to take this alternate approach? The answer lies in a very special property of the natural numbers known as the *Well-Ordering Principle*. This principle says that any non-empty set of natural numbers contains a "smallest" element. Formally:

Definition 4.1 (Well-ordering principle)**.** *If $S \subseteq \mathbb{N}$ and $S \neq \emptyset$, then S has a smallest element.*

Sanity check! Consider the following subsets of the natural numbers: $S_1 = \{5, 2, 11, 7, 8\}$, $S_2 = \{n \in \mathbb{N} : n \text{ is odd}\}$, $S_3 = \{n \in \mathbb{N} : n \text{ is prime}\}$. Does each of these sets have a smallest element?

Where did we use the well-ordering principle in our proof? In the line "Suppose ... that the jth day for $j > k$ is the first counterexample..." — specifically, if the natural numbers did not obey this principle, then the notion of a *first* counterexample would not be valid. Note also that induction relies on this principle: The statement "for all j, $P(j) \implies P(j+1)$" only makes sense if there is a well-defined order imposed on the natural numbers (i.e., that 3 comes before 4, and 4 comes before 5, etc...). That the natural numbers obey this principle may be a fact you have long taken for granted; but there do exist sets of numbers which do *not* satisfy this property! In this sense, the natural numbers are indeed quite special.

Sanity check! Do the integers satisfy the well-ordering principle? How about the reals? How about the non-negative reals?

Back to our analysis of the Stable Marriage Algorithm. Returning to our analysis, we now prove that when the algorithm terminates, all $2n$ people are paired up. Before reading the proof, see if you can convince yourself that this is true. The proof is remarkably short and elegant, and critically uses the Improvement Lemma.

Lemma 4.3. *The stable marriage algorithm always terminates with a pairing.*

Proof. We proceed by contradiction. Suppose that there is a man M who is left unpaired when the algorithm terminates. He must have proposed to all n of the women on his list. By the Improvement Lemma, each of these n women has had someone on a string since M proposed to her. Thus, when the algorithm terminates, n women have n men on a string not including M. So there must be at least $n+1$ men. But this is a contradiction, since there are only n men. □

To complete our analysis of the stable marriage algorithm, we need to verify the key property that the pairing produced by the algorithm is stable.

Theorem 4.1. *The pairing produced by the algorithm is always stable.*

Proof. We give a direct proof that, in the pairing output by the algorithm, no man can be involved in a rogue couple. Consider any couple (M,W) in the pairing and suppose that M prefers some woman W^* to W. We will argue that W^* prefers her partner to M, so that (M,W^*) cannot be a rogue couple. Since W^* occurs before W in M's list, M must have proposed to W^* before he proposed to W. Therefore, by the Improvement Lemma, W^* likes her final partner at least as much as M, and therefore prefers him to M. Thus no man M can be involved in a rogue couple, and the pairing is stable. □

5 Optimality

In Section 4.2, we showed that the stable marriage algorithm always outputs a stable pairing. But is this so impressive? Will your dating service really be successful outputting matches which are just "good enough"? Of course not! To offer the best dating service around, you require some notion of *optimality* in the solutions you obtain.

Consider, for example, the case of 4 men and 4 women with the following preference lists (for simplicity, we use numbers and letters below to label the men and women):

Men	Women			
1	A	B	C	D
2	A	D	C	B
3	A	C	B	D
4	A	B	C	D

Women	Men			
A	1	3	2	4
B	4	3	2	1
C	2	3	1	4
D	3	4	2	1

For these lists, there are exactly two stable pairings: $S = \{(1,A), (2,D), (3,C), (4,B)\}$ and $T = \{(1,A), (2,C), (3,D), (4,B)\}$. The fact that there are *two* stable pairings raises the question: What is the best possible partner for each person? For example, let us consider man 2. The trivial best partner for 2 is his first choice, woman A; unfortunately, A is just not a realistic possibility for him, as pairing him with A would not be stable, since he is so low on her preference list. Indeed, there is no stable pairing in which 2 is paired with A. Examining the two stable pairings, we find that the best possible realistic outcome for man 2 is to be paired with woman D. This demonstrates that the notion of "best partner" can be a bit fuzzy if we are not careful.

So, let us be careful; inspired by the discussion above, here is a rigorous definition of optimality.

Definition 4.2 (Optimal woman for a man)**.** *For a given man M, the* optimal *woman for M is the highest woman on M's preference list that M is paired with in any stable pairing.*

In other words, the optimal woman is the best that a man can do in a pairing, *given that the pairing is stable*.

Sanity check! Who are the optimal women for each man A, B, C, and D in our example above? (Hint: The optimal woman for man 2, as discussed above, is D.)

By definition, the best that each man can hope for is to be paired with his optimal woman. But can all men achieve this optimality *simultaneously*? In other words, is there a stable pairing such that each man is paired with his optimal woman? If such a pairing exists, we will call it a *male optimal* pairing. Returning to the example above, S is a male optimal pairing since A is 1's optimal woman, D is 2's optimal woman, C is 3's optimal woman, and B is 4's optimal woman. Similarly, we can define a *female optimal* pairing, which is the pairing in which each woman is paired with her optimal man.

Sanity check! Check that T is a female optimal pairing.

We can also go in the opposite direction and define the *pessimal* woman for a man to be the lowest ranked woman whom he is ever paired with in some stable pairing. This leads naturally to the notion of a *male pessimal* pairing — can you define it, and also a female pessimal pairing?

Now, we get to the heart of the matter: Who is better off in the Stable Marriage Algorithm — men or women?

Theorem 4.2. *The pairing output by the Stable Marriage algorithm is male optimal.*

Proof. Suppose for sake of contradiction that the pairing is *not* male optimal. Then, there exists a day on which some man was rejected by his optimal woman; let day k be the first such day. On this day, suppose M was rejected by W^* (his optimal mate) in favor of M^*. By definition of optimal woman, there must exist a stable pairing T in which M and W^* are paired together. Suppose T looks like this: $\{\ldots,(M,W^*),\ldots,(M^*,W'),\ldots\}$. We will argue that (M^*,W^*) is a rogue couple in T, thus contradicting stability.

First, it is clear that W^* prefers M^* to M, since she rejected M in favor of M^* during the execution of the stable marriage algorithm. Moreover, since day k was the first day when some man got rejected by his optimal woman, before day k M^* had not yet been rejected by his optimal woman. Since he proposed to W^* on day k, this implies that M^* likes W^* at least as much as his optimal woman, and therefore at least as much as W' (his partner in the stable pairing T). Therefore, (M^*,W^*) form a rogue couple in T, and so T is not stable. Thus, we have a contradiction, implying the pairing output by the algorithm is male optimal. □

What proof techniques did we use to prove this Theorem 4.2? Again, it is a proof by induction, structured as an application of the well-ordering principle.

Sanity check! Where did we use the well-ordering principle in the proof of Theorem 4.2?

Exercise. Can you rewrite the proof of Theorem 4.2 as a regular induction proof? (Hint: The proof is really showing by induction on k the following statement: For every k, no man gets rejected by his optimal woman on the kth day.)

We conclude that men fare very well by following this algorithm. How about the women? The following theorem confirms the sad truth:

Theorem 4.3. *If a pairing is male optimal, then it is also female pessimal.*

Proof. We proceed by contradiction. Let $T=\{\ldots,(M,W),\ldots\}$ be the male optimal pairing (which we know from Theorem 4.2 is output by the algorithm). Suppose for the sake of contradiction that there exists a stable pairing $S=\{\ldots,(M^*,W),\ldots,(M,W'),\ldots\}$ such that M^* is lower on W's list than M (i.e., M is not her pessimal man). We will argue that S cannot be stable by showing that (M,W) is a rogue couple in S. By assumption, W prefers M to M^* since M^* is lower on her list. And M prefers W to his partner W' in S because W is his partner in the male optimal pairing T. Contradiction. Therefore, the male optimal pairing is female pessimal. □

In light of these findings, what does the stable marriage algorithm teach us? When it comes to relationships, it pays to make the first move!

Exercise. What simple modification would you make to the stable marriage algorithm so that it always outputs a *female optimal* pairing?

Let us close with some final comments about the National Residency Matching Program. Until recently the stable marriage algorithm was run with the hospitals doing the proposing, and so the pairings produced were hospital optimal. In the nineties, the roles were reversed so that the medical students were proposing to the hospitals. More recently, there were other improvements made to the algorithm which the N.R.M.P. used. For example, the pairing takes into account preferences for married students for positions at the same or nearby hospitals.

6 Further Reading (optional)

Though it was in use 10 years earlier, the propose-and-reject algorithm was first properly analyzed by Gale and Shapley, in a famous paper dating back to 1962 that still stands as one of the great achievements in the analysis of algorithms. The full reference is:

D. Gale and L.S. Shapley, "College Admissions and the Stability of Marriage," *American Mathematical Monthly* **69** (1962), pp. 9–14.

Stable marriage and its numerous variants remains an active topic of research in computer science. Although it is by now 30 years old, the following very readable book covers many of the interesting developments since Gale and Shapley's algorithm:

D. Gusfield and R.W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, 1989.