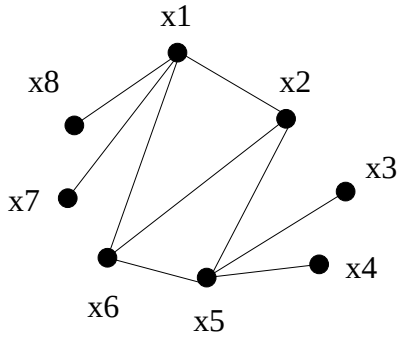
Graph Theory II

1 Matchings

Today, we are going to talk about matching problems. Matching problems arise in numerous applications. For example, dating services want to pair up compatible couples. Interns need to be matched to hospital residency programs. Other assignment problems involving resource allocation arise frequently, including balancing the traffic load among servers on the Internet. We'll talk more about each of these applications later today.

In the simplest form of a matching problem, you are given a graph where the edges represent compatibility and the goal is to create the maximum number of compatible pairs.

Definition. *Given a graph $G = (V, E)$, a* matching *is a subgraph of G where every node has degree 1. In particular, the matching consists of edges that do not share nodes.*

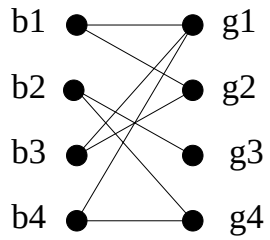


In this graph, x_1—x_6, x_2—x_5 is a matching of size two. But there is a larger matching – namely, x_1—x_8, x_2—x_6, x_4—x_5 is a matching of size three. Can there be a larger matching? Well, that would mean that every node is paired. But each of x_7 and x_8 can only be paired with x_1, and x_1 can only be paired with one other node in a matching. So, the answer is no!

Let's now define a matching that includes every node:

Definition. *A matching of a graph $G = (V, E)$ is* perfect *if it has $\frac{|V|}{2}$ edges.*

There is no perfect matching for the previous graph. Matching problems often arise in the context of the bipartite graphs — for example, the scenario where you want to pair boys with girls.



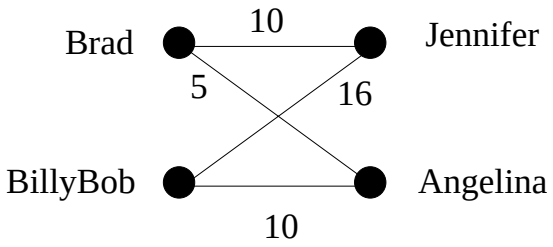
For example, the above graph has a perfect matching, namely b_1—g_2, b_2—g_3, b_3—g_1, b_4—g_4.

In many applications, not all matchings are equally desirable. For example, maybe b_1 and g_2 like each other a lot more than b_1 and g_1. Often, we can represent the desirability of a matching with a weight on the edge. For example b_1 and g_2 get weight 5 while b_1 and g_1 get weight 10.

The goal then is to find the perfect matching with the minimum weight.

Definition. *The* weight of matching M *is the sum of the weights on the edges in* M*. The* min-weight matching *for a graph* G *is the perfect matching for* G *with minimum weight. (If it exists)*

For example, the min-weight matching for the following graph is 20 (Brad gets matched with Jennifer, and Billy Bob with Angelina[1]).



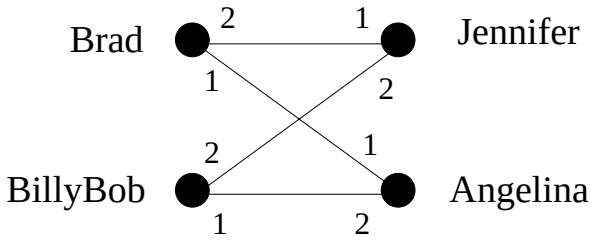
It turns out that there are fast algorithms for finding maximum matchings in unweighted graphs and min-weight matchings in weighted graphs, but they are complicated and we don't have time to cover them in 6.042 (in particular, the greedy algorithm doesn't work in general).

2 Will you marry me?

Instead, we are going to talk about a different variant of the matching problem that does have an elegant solution and that is frequently used in practice. In this version of the

[1] Any similarity between our choice of names and the TA names is purely coincidental. However, on occasion, Professor Leighton has been known to suggest that he has a slight resemblance to Brad.

problem, every node has a preference order of the possible mates. The preferences don't have to be symmetric. For example, maybe Jennifer really likes Brad but Brad has the hots for Angelina. Suppose Angelina also likes Brad more than Billy Bob but that Billy Bob really likes Angelina.



In the above figure, suppose we were to pair Brad with Jen and Billy Bob with Angelina. Well, that would lead to a very dicey situation! Suffice it to say that pretty soon, Brad and Angelina are likely to start spending late nights doing 6.042 homework together. The main problem is that Brad and Angelina each prefer each other to their mates in the matching. In such a circumstance we say that Brad and Angelina form a *rogue couple*. More precisely, we'll say that given a matching M, x and y are a *rogue couple* for M if x and y prefer each other to their mates in M.

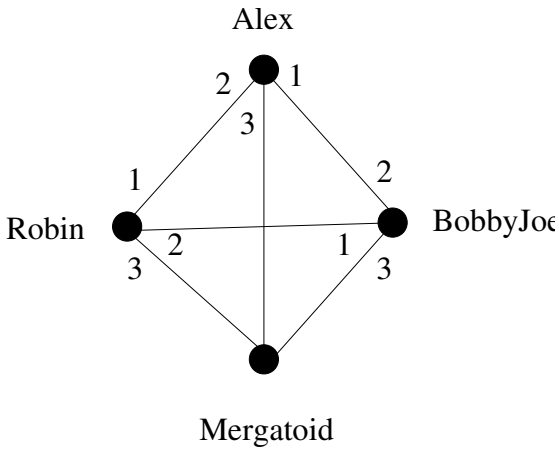
Obviously, the existence of rogue couples is not a good thing if you are making matchings, since they lead to instability. So, we'll say that a matching is *stable* if there are no rogue couples.

We are going to assume that preferences do not change with time. So we are not modeling the situation where you get tired of your mate or want to "play the field". Preferences are known at the start and never change.

Our main goal is to find a perfect matching that is stable. In this example, a possible stable perfect matching is to pair Brad with Angelina, and Billy Bob with Jen. Billy Bob and Jen may not be so happy, but no rogue couple is possible and so it is a stable matching. That's because neither Brad nor Angelina like anyone better than each other, so even though Jen and Billy Bob are not happy with each other, no one else will form a rogue couple with either of them. This assumes they are only four people on Earth. Or, more reasonably, suppose you put these four on a desert island— there would be a stable matching.

In general, it isn't so clear that there is always a stable matching for any number of people and set of preference orders. In fact, the answer is kind of complicated. If you allow boys to prefer boys and girls to prefer girls, then there are examples where there is no stable matching. But the strange thing is that in the special case where boys only get pairs with girls, then you can *always* find a stable matching.

We're going to show how to find such a stable matching shortly. But first, let's look at a unisex example where a stable matching is not possible. The idea is to create a love triangle with a fourth person who is everyone's last choice:



Turns out Mergatoid's preferences don't even matter. Let's see why there is no stable matching...

Theorem. *There is no stable matching.*

Proof. We'll prove this by contradiction. Assume, for the purposes of contradiction, that there is a stable matching. Then there are two members of the love triangle that are matched. Without loss of generality, (by symmetry) assume that Robin is matched to Alex. Then the other pair must be Bobby-Joe matched with Mergatoid. But then there is a rogue couple since Alex likes Bobby-Joe best and Bobby-Joe prefers Alex to Mergatoid. So, Alex and Bobby-Joe are a rogue couple. Thus, there cannot be a stable matching. □

This theorem is not very surprising. Getting a stable matching is a hard thing to do. What is surprising is that you can always do it in bipartite graphs – that is, where boys are only allowed to pair with girls and vice versa.

Note that we're not making a political or social statement here— this is just a fact of mathematics. So let's formalize the statement of the problem that we are discussing here.

3 The Stable Marriage Problem

The setting:

- There are N boys and N girls. We assume the number of boys and girls is the same.[2]
- Each boy has his own ranked preference list of girls.
- Each girl has her own ranked preference list of boys.

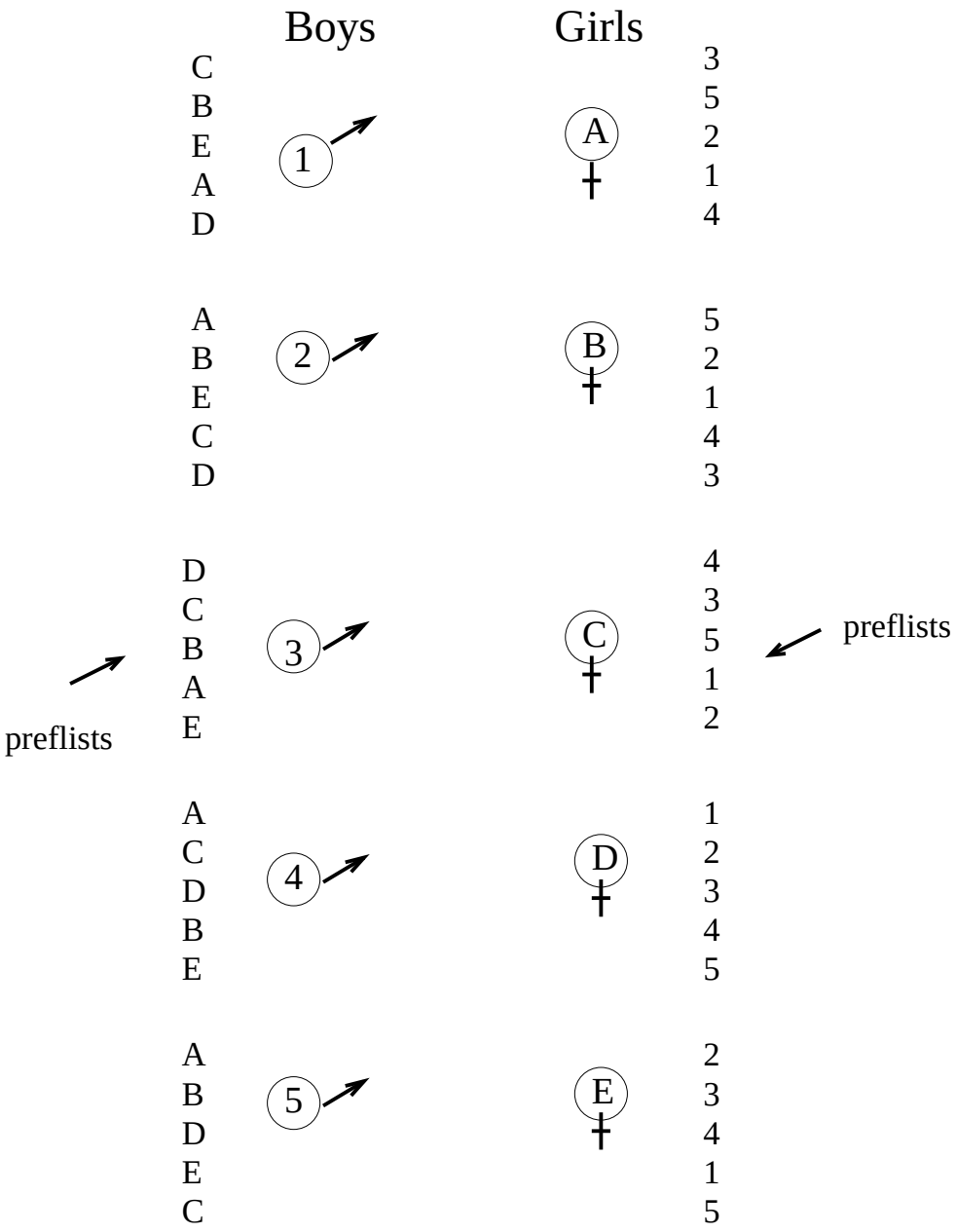
[2] A scenario where there are two or more boys for every girl or vice versa is also interesting. We will consider it in recitation, since it turns out to be useful in practice (not with boys and girls, of course, but in real world assignment problems).

- The lists are complete and have no ties. Each boy ranks every girl and vice versa.

The goal: Pair each boy with a unique girl so that there are no rogue couples. That is, find a perfect matching so that every boy and girl are paired up one to one, with no potential funny business.

4 Finding the stable matching

Let's see if we can figure out a method for finding a stable matching by looking at an example:



Let's try to use a *greedy* algorithm to find the matching. In this case, the greedy algorithm will have each boy pick his favorite girl that remains by the time his turn comes up.

Running the greedy algorithm on our example, boy 1 picks his favourite, which is C, boy 2 picks his favorite, which is A, boy 3 picks his favorite, which is D, boy 4 picks his favorite remaining girl, which is B (since his top 3 choices are already taken), and finally, boy 5 picks his favorite remaining girl (which at this point, is the only remaining girl), which is E.

Let's see – is there a rogue couple? Well, 1,2,3 are matched up with the loves of their lives, so they are too happy to be thinking of running off. However, 4 is not so happy with B, who is his third choice. He approaches A, but she isn't interested in him, since she prefers boy 2 – in fact, she ranked 4 dead last so she wouldn't be caught dead in an affair with him. However, he runs into his love of his life, that is C, and she definitely prefers him to 1, who is way down on her list. So we have a situation here. Both 4 and C prefer each other to their own mates. We could try to patch things up and pair 4 with C and then B with 1, but it's not clear that we would reduce the number of rogue couples, since for all we know, 4 and C could still be in rogue couple. It happens that in this case pairing up 4 and C is an ok thing to do. But, this is getting more and more complicated.

How about using an algorithm that is based on induction (or recursion)? Pair 1 with C and solve the rest by induction. By the induction hypothesis, the only rogue couples would involve 1 or C. But, they can't involve 1, since he got his first choice. On the other hand, they might well involve C since 1 might be her last choice! Induction would work if there were some boy and some girl who each ranked the other first. If there were such a boy and girl, then they have to get paired to each other, or they would be a rogue couple. But, there might not be such a boy and girl. Too often people do not like those that like them!

Turns out that finding a good way of pairing up the boys and girls is a tricky problem. The best approach is to use the dating protocol that was popular in the 50's.

5 The Mating Algorithm (TMA)

Here is a method for getting everyone paired up. The mating ritual takes place over several days. The idea is that each of the boys go after the girls one by one, in order of preference, crossing off girls from their list as they get rejected. Here is a more detailed specification:

Initial Condition: Each of the N boys has an ordered list of the N girls according to his preferences. Each of the girls has an ordered list of the boys according to her preferences.

Each Day:

- Morning:

  - Each girl stands on her balcony
  - Each boy stands under the balcony of his favorite girl whom he has not yet crossed off his list and serenades. If there are no girls left on his list, he stays home and does 6.042 homework.

- Afternoon:

  - Girls who have at least one suitor say to their favorite from among the suitors that day: "Maybe, come back tomorrow."
  - To the others, they say "No, I will never marry you!"

- Evening:

  - Any boy who hears "No" crosses that girl off his list.

Termination Condition: If there is a day when every girl has at most one suitor, we stop and each girl marries her current suitor (if any).

6 An example:

Let's run TMA on the example from before. On the first morning, boy 1 serenades girl C, boy 2 serenades girl A, boy 3 serenades girl D, boy 4 serenades girl A and boy 5 serenades girl A. In the afternoon, girls A,C, and D say "Maybe, come back tomorrow" to boys 5, 1 and 3 respectively. Girl A says "No!" to boys 2 and 4, who cross A off their lists that evening.

On the second morning, boy 1 serenades girl C, boy 2 serenades girl B, boy 3 serenades girl D, boy 4 serenades girl C and boy 5 serenades girl A. In the afternoon, girls A,B, C, and D say "Maybe, come back tomorrow" to boys 5, 2, 4 and 3 respectively. Girl C says "No!" to boy 1, who crosses C off his list in the evening.

On the third morning, boy 1 serenades girl B, boy 2 serenades girl B, boy 3 serenades girl D, boy 4 serenades girl C and boy 5 serenades girl A. In the afternoon, girls A,B, C, and D say "Maybe, come back tomorrow" to boys 5, 2, 4 and 3 respectively. Girl B says "No!" to boy 1, who crosses B off his list in the evening.

On the fourth morning, boy 1 serenades girl E, boy 2 serenades girl B, boy 3 serenades girl D, boy 4 serenades girl C and boy 5 serenades girl A. In the afternoon, the girls realize that each girl has at most one suitor, so all five couples start planning their weddings.

7 Mating at work

Now let's show that the algorithm works. We need to show that

- *TMA terminates* – We don't want these poor guys singing forever.
- *TMA terminates quickly* – In fact, their singing is pretty bad, so we'd like them to finish as quickly as possible.
- *At termination, there are no rogue couples* – This is in fact the main goal. It would be a shame if after all this work, a rogue couple spoiled everything.
- *Everyone is married* – Stability is very easy to show if there are no marriages!

We will also analyze the fairness of the protocol. It is better for boys or for girls?

Let's start by showing that this algorithm terminates:

Theorem 1. *TMA terminates within $N^2 + 1$ days.*

Proof. We'll prove this theorem by contradiction. Suppose (for the purposes of contradiction) that TMA does not terminate in $N^2 + 1$ days.

Well, let's notice something that *must* happen on a day in which TMA doesn't terminate – it must be that some boy crosses a girl off his list that evening! Why is this? If TMA doesn't terminate, then some girl must have had at least 2 suitors. If a girl has at least 2 suitors then at least one gets rejected, and that boy crosses that girl off his list.

So if TMA doesn't terminate in $N^2 + 1$ days, there are at least $N^2 + 1$ names crossed off in total. But at the start, each list is of size at most N, so the total size of all the lists put together is at most N^2. So we couldn't have crossed off $N^2 + 1$ names, and thus we have our contradiction. □

This is a typical proof technique in Computer Science used to bound the running time of an algorithm. We show that the algorithm is always making progress by some measure. Then, since there is only a finite amount of progress to make, it must eventually terminate. Here the measure is the number of names on the union of the lists.

Next, we'll prove that everyone gets married by TMA, but first we'll need a couple of lemmas.

Lemma 1. *If a boy marries, then he courted every girl he liked better.*

Proof. In TMA, boys cross girls off their lists one at a time in preference order, starting with the girl he likes most. A boy marries the girl (if any) that he is courting at termination. □

So if a boy marries, he marries his *least* favorite girl among those he courted. Tough luck for the boy, but at least he likes her better than all the girls he never courted.

Lemma 2. *If a boy never marries, then he courted every girl.*

Proof. By Theorem 1, TMA terminates. At the time of termination, the boy is not courting. How can that be? If he is home doing his 6.042 homework, then he must have crossed off every girl on his list. So, he has courted every girl. □

Lemma 3. *A girl marries her favorite among her suitors.*

Proof. A girl only rejects a boy when a better one comes along and always keeps stringing along her favorite among those seen so far. □

This also means that:

Lemma 4. *If a girl is ever courted, she gets married.*

Proof. Once a girl has a suitor, she keeps him until she trades up. □

Now we can prove that everyone gets married:

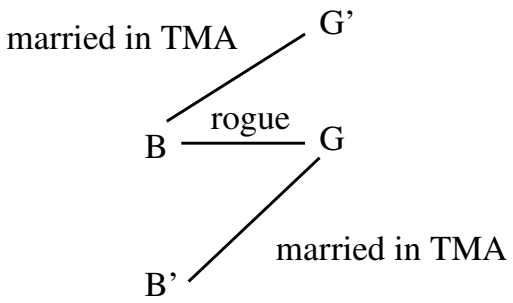
Theorem 2. *Everyone is married in TMA.*

Proof. We'll show this one by contradiction. Assume (for the purposes of contradiction) that some boy B is not married. But then, by lemma 2, B has courted every girl. So, every girl has been courted. But then, every girl is married by lemma 4. But since there are an equal number of boys and girls, it must be the case that every boy (including B) is married. So the theorem is true by contradiction. □

Next we'll prove the main result, namely that TMA always produces stable marriages.

Theorem 3. *TMA produces stable marriages.*

Proof. Assume, for the purposes of contradiction, that there is a rogue couple $B - G$. Suppose B married G' and G married B' in TMA.



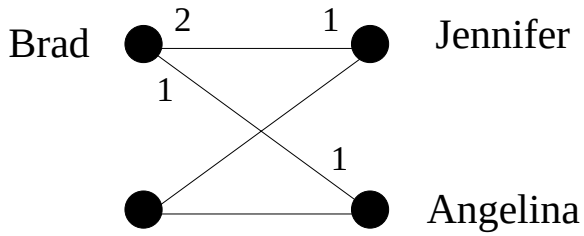
If B married G', but likes G better, then B visited G first by lemma 1, and G said "no" to B. But then, G must have married someone that she likes better than B by lemma 3. So, G likes B' better than B, which means that $B - G$ is not a rogue couple.

□

Well, who do you think is better off in TMA? In other words, who has the power, the proposers or the acceptors? Since the girls marry their favorite from among their suitors, and the boys get the worst girls that they court, it seems reasonable to assume that the girls do best. It seems hard to answer this question formally, especially since it isn't even clear what we mean by "doing better". But, in fact, we can show in a very precise and formal way that the algorithm is heavily biased toward the boys. To formalize this, we need to define the set of realistic potential mates.

Let S be the set of all stable matchings. Since TMA gives a matching, we know that $S \neq \emptyset$. For each person P, we define the *realm of possibility* for P to be $\{Q | \exists M \in S, \{P, Q\} \in M\}$. That is, Q is within the realm of possibility for P iff there is a stable matching where P marries Q.

Some mates just might be out of the question, since no stable pairings are possible if you married them. For example, Brad is just not realistic for Jennifer since if you ever pair them, Brad and Angelina will form a rogue couple – so there is no stable matching with Brad paired to Jennifer.



Definition. *A person's* optimal mate *is his/her favorite from the realm of possibility.*

An optimal mate must exist, since we know there is at least one stable matching, namely the one produced by TMA.

Definition. *A person's* pessimal mate *is his/her least favorite from the realm of possibility.*

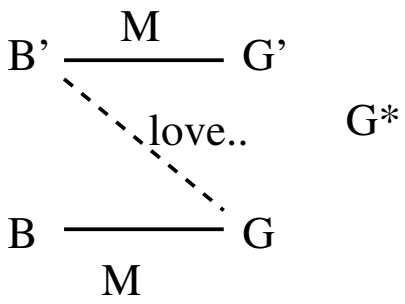
Ok, now here is a pair of shocking results:

Theorem 4. *TMA pairs every boy with his optimal mate!*

Theorem 5. *TMA pairs every girl with her pessimal mate!*

Wow! Talk about a sexist algorithm! This is too hard to believe, so let's do the proof.

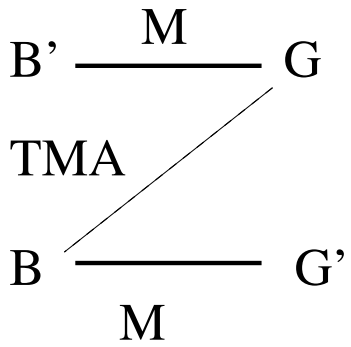
Proof of Theorem 4. Assume for the purposes of contradiction that some boy does not get his optimal girl (that is, his favorite girl within the realm of possibility). Let B be the first (in time) boy that gets rejected by his optimal girl G (resolving ties arbitrarily). (Did you catch the use off the well ordering principle here?) Define B' to be the boy that caused G to reject B in TMA. Then G prefers B' to B.



Since B is the first to be rejected by the optimal mate in TMA, B' has not (yet) been rejected by the optimal mate when he is courting G. So, B' likes G at least as much as he likes his optimal mate G^* (G and G^* might be the same person). Let M be a stable matching where B marries G. M exists since G is in the realm of possibility of B. M is not produced by TMA by assumption. Let G' be the spouse of B' in M. By definition, B' likes G^* at least as much as G' (again, they might be the same person), so B' prefers G to G' since B likes G at least as much as G^*, whom he likes at least as much as G'. (Note that G can't be the same person as G'.) So B' and G are a rogue couple, which contradicts the fact that M is a stable matching! □

Now let's show that the girls get their pessimal mate.

Proof of Theorem 5. Suppose, for the purposes of contradiction, that there is a stable matching M where there is a girl G who fares worse than in TMA. Let B be the mate of G in TMA. Let B' be the mate of G in M. Then G likes B better than B' since she fared worse in M than in TMA. Let G' be the mate of B in M.



We know that B likes G better than G' since (by Theorem 4) TMA gives an optimal mate for B. Then B and G form a rogue couple in M, which is a contradiction. □

So, it really pays off to be the aggressive party in a courtship! All that pain of rejection and effort put into singing really pays off! The moral of the story is this:

And they all lived happily ever after, but the boys lived especially happily!

TMA arises in all sorts of applications. Perhaps the most famous application is in matching fresh MDs to residency programs. Fourth year medical students have to fill out a form with their top 20 choices for residency programs. Teaching hospitals do the same thing with their top choices for doctors. Then the data is fed to the algorithm which matches doctors to hospitals. The doctors find out their assigments on *match day*, which is a huge event.

The algorithm used is a variant of TMA, where the hospitals are boys and the doctors are girls, but in this case there are multiple girls for every boy. We still want a solution that is stable so that no swapping will occur – rogue couples can cause chaos and instability in the medical community as well! Actually, TMA is very civilized. There is no waiting list or delay. Of course, the hospitals get their optimal choices and the doctors get their pessimal choices.

Not surprisingly, TMA is also used by at least one large dating agency.

Akamai uses a variation of TMA to assign web traffic to servers. In the early days, Akamai used other combinatorial optimization algorithms that got to be too slow as the number of servers and traffic increased. Akamai switched to TMA since it is fast and can be run in a distributed manner. In this case, the web traffic corresponds to the boys and the web servers to the girls. The servers have preferenes based on latency and packet loss, the traffic has preferences based on the cost of bandwidth.