The following content is provided under a Creative Commons license. Your support will help MIT OpenCourseWare continue to offer high quality educational resources for free. To make a donation or view additional materials from hundreds of MIT courses, visit MIT OpenCourseWare at ocw.mit.edu.

PROFESSOR: OK, on Tuesday we talked about sex, so today we're going to talk about marriage. Now, in terms of graph theory, marriage is expressed as a matching problem, and today we're going to talk about a matching algorithm that is used in all sorts of applications. It's used by online dating agencies to match compatible people together.

It's used for assignment problems, for example, matching interns to hospitals on match day. It's used for resource allocation problems, for example, load balancing traffic on the internet. And we'll talk about the applications at the end of class.

In its simplest form of a matching problem, you have a graph where the edges represent compatibility. Two nodes can be paired together, or married, and the goal is to create the maximum number of compatible pairs. So let's define a matching, given a graph, G, with nodes, V, and edges, E. In matching, you can think of it as a collection of edges, or a subgraph of G where every node has degree 1. So everybody can be married just to one person.

So let's draw an example, maybe not put that edge in. And let's label these nodes x1, x2, x3, x4, x5, x6, x7, and x8. Now x1, x6 and x2, x5 is a matching, so x1, x6 and x2, x5 is a matching with two edges, so we say it has size two.

All right, so I can pair these guys up and pair these guys up. Is there any bigger matching in this graph? So I found one with two marriages, here and here, two edges. Yeah?

AUDIENCE: x1, x7.

PROFESSOR: x1, x7.

AUDIENCE: x2, x6.

PROFESSOR: x2, x6.

AUDIENCE: [INAUDIBLE].

PROFESSOR: Good. All right, that's a matching of size three. So I got three couples together. Good. Can I make a bigger matching, one with size four-- four marriages here? No. Why not? Can anybody give me a reason why it can't be done? Yeah?

AUDIENCE: x8 and x7 would have to be matched with someone.

PROFESSOR: Yeah, so if-- Yeah?

AUDIENCE: They could only be paired with x1, but x1 can't be paired with both.

PROFESSOR: Good. If I were to have a matching with four edges, well, there's only eight nodes, so I'd have to have all eight nodes involved in the matching. And that means x7 and x8 would have to be in the matching, but they could only be paired with x1, and so it's not possible to do that. All right, so there is no matching of size four in this graph. Three is the best I can do.

Now, when you get every node in a matching, then it's called a perfect matching. And so in this case, it doesn't exist, but sometimes it does. So a matching is perfect if it has size half the number of nodes. In other words, if the number of edges is v over 2, then every node is in the matching.

All right, so that one doesn't have a perfect matching. What about this graph? So I got b1, b2, b3, b4, g1, g2, g3, g4, and I'll put in the compatibility edges here.

OK, Does that graph have a perfect matching? Can you pair up every boy with a girl here so that everybody is compatible with their mate, and you have just one spouse? Can you do that?

AUDIENCE: Yes.

PROFESSOR: Yeah? All right, who do I start pairing up?

AUDIENCE: 1, 1.

AUDIENCE: 1, 1.

AUDIENCE: b2, g3.

PROFESSOR: b2, g3.

AUDIENCE: b3, g2.

PROFESSOR: b3, g2

AUDIENCE: b4, g4.

PROFESSOR: There we go. All right, so there is a perfect matching in this graph. Very good.

Now, in some cases, some pairings are more desirable than others, and this can be represented with a weight. And so you might have a weighted graph where every edge has a weight on it. For example, we might weight b1, g2 with 5, and b1, g1 gets a 10.

And usually when you see weighted graphs in the matching context, a lower weight means it's more desirable, so that b1 and g2 would get along better than b1 and g1. And then the goal is to find a matching with minimum weight.

Now, the weight of a matching, call it M, is the sum of the weights on the edges of M. Now, usually when you're looking at weighted matchings, you require yourself to have a perfect matching so that everybody gets paired up. And often in that case, you'll see all the edges present, some of them with very big weights, maybe even infinity if they just can't be put together. Because otherwise you just say, don't match anybody together, and you have weight zero.

So for when you look at minimum weight matchings, you're looking for the perfect matching with minimum weight. So we say a min-weight matching for a graph, G, is a perfect matching for G with the minimum weight, overall perfect matchings. Let's try an example.

Say I've got this graph, and call this node Brad, here's Billy Bob, here's Jennifer, and then Angelina. And the weights are as follows-- I put a 10 here, a 10 down here, a 5 here, because Brad really likes Angelina and vice versa, and a 16 between Jennifer and Billy Bob. What is the weight of the min weight matching in that graph? 20. And who gets paired with who there? Who does Brad get hooked up with?

AUDIENCE: Jan.

PROFESSOR: Jan, and that leaves Billy Bob with Angelina, and the weight is 20. So even though Brad really likes Angelina, if I go that route, my weight is 21, which is not as good. So the min-weight matching would be this one.

OK. Now, it turns out that finding the maximum matching-- the maximum number of edges you can put together-- or finding the minimum weight perfect matching, those are both solvable, tractable problems. You don't get a million dollar prize for solving that. The algorithm is run in quadratic to cubic time, so not terrible, but they're not NP-complete, so people know how to do it.

Now, they are pretty complicated, so we're not going to cover them in 6042. What we're going to do is look at a slightly different version of the problem that actually turns out to be more useful in practice, because there's a very nice algorithm for it. Now, in the version of the problem that we're going to look at, everybody has preferences-- a preference list. It's not weighted, but it's a priority order of who they want to get mated to, or matched up with.

So it would look not quite like that, but it would look like this. So Brad, Billy Bob, Jennifer, and Angelina-- same players, but what we do is we know that Brad really likes Angelina, and Angelina really likes Brad, so they are first choices, at least was. Number two for Brad is Jennifer, but Jennifer really likes Brad first, and Billy Bob second. Billy Bob likes Angelina first, then Jennifer. Angelina thinks Billy Bob is number two.

So it's not necessarily symmetric. Jennifer has Brad as the first choice. Brad has Angelina as the first choice. All right? So it's an asymmetric situation.

Now, what would happen if we set up our marriages so that Brad is married to Jen, and Billy Bob is married to Angelina? What might happen if we made those as our pairings, and we put them on a desert island, all four of them? What's likely to happen there?

AUDIENCE: Brad and Angelina are going to cheat on each other.

PROFESSOR: Yeah, we're going to have a problem. Because Brad and Angelina have the hots for each other here, and they like each other better than their spouse. So before you know it, they're going to be doing their 6042 homework together late at night. All right? Now, when this happens, we have what's called a rogue couple.

Given a matching, x and y, a boy and a girl, say, form a rogue couple if they prefer each other over their mates in M. All right. So here, if we married Brad to Jennifer, and Billy Bob to Angelina, Brad and Angelina form a rogue couple, because they like each other better than who they were hooked up with. And that's sort of a bad thing. It creates instability if you were to make the matchings that way. In fact, we say that a matching is stable if there aren't any rogue couples.

And now, one thing to make clear is that your preferences can't change over time. So it's not a situation where you get bored with your spouse, and you change your mind, and then you go off and create a rogue couple. You're fixed in your preferences over all time, no playing the field, none of that stuff here, OK?

So it's fixed once and for all, and your goal, of course, is to create or find a perfect matching that's stable. That's the goal. So get everybody married up and make it stable.

All right, is it doable in that example? If I put those four people on a desert island, could I make a stable matching? Yeah, who would I match Brad to?

AUDIENCE: Angelina.

PROFESSOR: All right. Good, and then Billy Bob gets Jennifer. Now, I'm not saying that you make everybody happy, because Billy Bob and Jennifer are probably not happy there. They each got their number two choice. But it's stable, because Brad and Angelina aren't going anywhere.

They are going to stay together because they like each other best, and so there's no chance that Jennifer is going to-- that Angelina is going to sneak off, sorry, with Billy Bob, or that Brad's going to sneak off with Jennifer. All right? So it's stable. Not everybody's happy, but it's a stable set of marriages.

OK. Any questions about what we're trying to do here now? Yeah?

AUDIENCE: [INAUDIBLE] the edges?

PROFESSOR: That's a great question. You'll see it referred to both ways. Technically, it's a subgraph, so it has nodes and edges, but you'll see me, and you'll see everybody say, oh, it's a bunch of edges that don't share any nodes, and you'll see them refer to it as the edges. But really, underlying that, it's a subgraph, technically.

Any other questions about what we're trying to do here to find stable, perfect matching? All right. Well, in this example, there was a stable, perfect matching. But what about in general? If I have a lot more people, and they have arbitrary preferences, how many people think you can always find a stable perfect matching?

There's one optimistic person. How many people think there's some cases where you're just not going to be able to do it? Wow. OK, it's a pessimistic view. Well, in some sense you're both right. If you allow boys to prefer boys and girls to prefer girls, then it is not possible, and I'll give you an example.

You can find examples where there's always a rogue couple, but if you require boys to only get matched to girls and vice versa, then it is possible to always find stable marriages, stable matchings. And we're going to talk about an algorithm for that. But before I show you the algorithm, let me show you the bad case when boys can prefer boys, or what's sort of a unisex scenario.

So here's a bad example with four people, and the idea is to create a love triangle. So we have Alex, who prefers Bobby Joe, Bobby Joe prefers Robin, and Robin prefers Alex. And their second choices go in the opposite order there, all right?

So Alex wants to be with Bobby Joe. Bobby Joe wants be with Robin. Robin wants to be with Alex. And then there's Mergatoid--

[LAUGHTER]

--and nobody likes Mergatoid. So that's choice three for all of them here. And Mergatoid's choices, preferences don't really matter in this case.

[LAUGHTER]

I hope nobody is named Mergatoid in the class. I would get complaints here. All right, so I want to claim and prove a theorem that says there is no stable matching for this group of preferences. So we'll state that as the theorem. There does not exist a stable matching for this graph. The proof is by contradiction-- assume there is one.

All right, so assume there exists a stable matching. We're going to find a rogue couple in it. So call the stable matching M. Well, if there's a stable matching, Mergatoid's got to get married to somebody. Mergatoid will be matched with someone.

All right. Now, here I'm going to do something that you can do in your proofs, but you've got to be careful when you do it. I'm going to say, without loss of generality, assume Mergatoid is matched to Alex, all right? And I can do that. So this is the abbreviation, without loss of generality by symmetry.

And really, I should explain what I mean here in the proof. Well, that love triangle is symmetric. Each one has a preference for the next person around the triangle. In terms of graph isomorphism, with the weights on that, every node looks the same. So I can use symmetry.

So we're going to say, without loss of generality, we're going to assume Mergatoid is matched to Alex. And I'm implying the argument is going to be the same. What I say next is the same no matter who Mergatoid is matched to, because it's symmetric.

All right. If Mergatoid is matched to Alex, do you see a rogue couple up there? So you have Robin matched to Bobby Joe. Mergatoid is matched to Alex.

AUDIENCE: Alex to Bobby Joe?

PROFESSOR: Alex and Bobby Joe, no.

AUDIENCE: Between Alex and Robin.

PROFESSOR: Alex and Robin, yeah. Alex and Bobby Joe aren't rogue, because Bobby Joe likes Robin. He likes the person he or she is married to. All right, so they're not going to go off with Alex, but Alex and Robin are a rogue couple, because Robin likes Alex the best, and Alex for sure likes Robin better than Mergatoid. All right? So they both prefer each other to their mates, and so they form a rogue couple.

All right, so Alex and Robin form a rogue couple, and that means that M was not stable. The matching was not stable. And that's a contradiction, because we assumed it was. So we have a contradiction, and the proof is done. Any questions about that?

And I'm sort of implying here that you could match, you could have had Mergatoid matched to Bobby Joe, and then Bobby Joe and Alex would have been a rogue couple, or Mergatoid matched to Robin, and then Robin and Bobby Joe would be a rogue couple by the without loss of generality. So it's OK to use that, but you want to be careful that you're doing it OK. Questions? Yeah?

AUDIENCE: So the without loss of generality only works if it's, like, perfectly symmetric?

PROFESSOR: Yeah. Basically, the argument you're going to make is just going to be the same argument done in all three cases, and to save yourself some effort, you're saying, don't do the three cases, do one, the other two look the same. So technically, you could add case one, case two, case three, and they would've looked symmetric. OK?

All right. Now, this is not very surprising, as you all voted. Almost all of you said it's hard to find stable matchings. You might not be able to do it always. And in fact, you can't in the unisex world. The surprising thing is, you can always find a stable matching in the world where boys could only be paired with girls and vice versa.

Now, this statement, this result is pretty famous. The problem itself is known as the stable marriage problem. So let me just define it here, and then we'll talk about an algorithm to find the matching.

So we have N boys and N girls. And it's important we have the same number of each. Now, actually, tomorrow, in recitation, you're going to look at the scenario where there's more girls than boys, or vice versa. And you'll be using a similar algorithm, but it'll be a different context, and that's the context that comes up in matching interns to hospitals, and stuff like that. But for our version it's an equal number of boys and girls.

Each boy has his own ranked preference list of all the girls. So every boy sort of has his dance card of the girls that he likes in order. All right? The orders can be different for different boys. And each girl has the same thing. She has her own list, ranked 1 to n of all the boys.

The lists are complete, and there's no ties, so all the ties have to be broken here. And the goal is to find a perfect matching without rogue couples. OK, so let's try an example. Before we do the algorithm, let me just do a bigger example with five boys and five girls, and we'll get some feel for, this is not completely obvious how to find it.

OK, so let's put the boys over here. And here's boy 1, and his preference list is going to be-- the girls will be C, B, E, A, D, and boy 2 is going to have the preference list A, B, E, C, D, and boy 3 is going to have D, C, B, A, E. Boy 4 is going to be A, C, D, B, E, and boy 5, I have to write his across like this, A, B, D, E, C.

And the girls also have their lists, so let's put those up. So girl A likes the boys in order 3, 5, 2, 1, 4. Girl B likes them in order 5, 2, 1, 4, 3. Girl C has the order 4, 3, 5, 1, 2. Girl D, 1, 2, 3, 4, 5, and then the last girl, E, has 2, 3, 4, 1, 5.

All right, so say that's our matching problem, and those are the preference lists. Any ideas for how we might try to make an algorithm to do this? Any thoughts about what you'd do? What are some approaches we could take? Yeah?

AUDIENCE: Use the mating algorithm.

PROFESSOR: Use the greedy algorithm?

AUDIENCE: The mating algorithm.

PROFESSOR: Oh, the mating algorithm. Well, yeah, the mating algorithm is going to do very well, but I haven't told you what that is, and it's a little complicated. So I sort of want to explore the things you might know from what we've done so far. What approaches have we seen so far for solving problems? Yeah?

AUDIENCE: The greedy algorithm?

PROFESSOR: The greedy algorithm-- yep. And let's try that. So let's choose the first boy, and we'll just go down and give each boy the best choice available and see what happens. First thing to try usually is the greedy algorithm, and about half the time it'll work in life, and half the time it won't.

So let's try greedy. So boy 1 is going to get his first choice, girl C. Who does boy 2 get? A. Who does boy 3 get? D. Huh, this is going well.

Who does boy 4 get? Boy 4, oh yeah, boy 4 has got to go all the way down to B, because A, C, and D got taken, so boy 4 gets B. And boy 5-- who's left? E, fourth choice again. All right, well, maybe that's a stable matching.

All right, well let's see. To see if it's stable, we've got to see are there any rogue couples. Well, is there any rogue couple involving boy 1? No, boy 1 got his first choice. You're not going anywhere.

Boy 2? No-- first choice, same with boy 3. These boys are quite happy. They're not doing anything. Boy 4 may not be so happy though, because boy 4 got paired with B, so there's possible rogue couples. Let's see, is 4, A a rogue couple? No. A hates boy 4, all right? No chance she's running off with boy 4.

AUDIENCE: 4, C?

PROFESSOR: 4, C? All right, let's see that. Where's C-- whoa, yeah. C has the hots for boy 4, so she likes him better than whoever her mate was, and boy 4 likes C better than who he got matched with, boy B. That is now a rogue couple, so the greedy algorithm did not work. That's too bad. Well, it would be short lecture, I guess, if it worked.

[LAUGHTER]

Well, what would you do next, sort of, if you were-- I mean, you could try to match 4 with C, and then try to patch things up, and you could start doing that, but you might create other rogue couples. In fact, I don't know an approach-- yeah?

AUDIENCE: You could let it evolve and see if it gets it more stable.

PROFESSOR: Say it again.

AUDIENCE: You could let it evolve, let all the rogue couples evolve and see if it ends up any more stable.

PROFESSOR: Yes, you could start swapping around to get rid of rogue couples. In doing that, you might create other rogue couples. In fact, I don't know of an algorithm that works like this, that works, that's known to work, where you start patching things up, because as you're patching things up, you might make other things much worse by doing that. So I don't know of an approach that way.

What's another approach? Yeah?

AUDIENCE: [INAUDIBLE] pertaining to what order they have each other.

PROFESSOR: And then do?

AUDIENCE: And then do the highest ordered [INAUDIBLE]

PROFESSOR: Min-weight matching kind of thing?

AUDIENCE: Maybe.

PROFESSOR: Maybe? I don't of an approach like that that works. Also, min-weight matching, that algorithm is going to be more complicated than the one I'm going to show you, in the end. It takes more time to run. And I don't even know if it works. Like, I don't know if you can take these numbers and make weights on the edges get a min-weight matching.

AUDIENCE: Could you do like a merge sort [INAUDIBLE]

PROFESSOR: Oh, a merge sort, so you take the minimum weight edge, put that in, and recurse on that kind of thing. I don't know. It's possible. In fact, you know what, a recursive approach is a good idea. I don't know of a nice recursive algorithm for this.

It's true that if you found a boy and a girl who liked each other best, you could then match them safely and recurse, because you know they're not going to be in a rogue couple, because they like each other best. Then you could recurse. But that might not exist here. You might not have a boy and girl that like each other best, in which case it's hard to know.

I don't know now if you pick the minimum weight in some sense, like add the preference list or something to make a min-weight and recurse on that, if that works. I don't know of an approach like that. But those are the kinds of things you try. And as far as I know, all the simple things fail for this problem.

But there is something that's a little more involved, but does work, and that's the mating algorithm. And does everybody have the handout? There, it's back up there. I got some copies down here if you need it, but pull that out. So we're going to read this and talk about what the algorithm is, and then prove that it performs well.

So the initial condition is you have each of the N boys has an ordered list of the N girls and vice versa, and the ritual, we're going to view this is a mating ritual, and really, the program is doing it. The code is doing it. But think of it as real life. It takes place over several days.

Now, the day is broken up into three parts-- the morning, the afternoon, and the evening. In the morning, each girl comes out to her balcony and stands on the balcony. Each boy goes to the balcony of his favorite girl who is still on his list that hasn't been crossed off.

Now, initially, every girl is on his list. So he goes to his favorite girl, goes under her balcony, and serenades her. Now, if, over the course of the algorithm, the boy has nobody left on his list, he's out of luck. He just stays home and does homework. All right? There's no serenade anymore for him.

Now, in the afternoon, the girls who have at least one suitor-- a boy down there serenading her-- looks at all the suitors, picks her favorite, and to the favorite she says, maybe I'll marry you. Come back tomorrow. Girls don't want to make it too easy here for the boys. Now, to all the other boys who are lower priority she says, I will never marry you. Go away. So she writes them off for good.

Now, that night, any boy who heard a no-- like the girl said no, I'll never marry you-- crosses that girl off his list. Because, you know, it's the only practical thing to do at that point. Now, if the boy heard the maybe I'll marry you, well, he's going to go back tomorrow, because that's still his favorite girl that's not crossed off. So he goes back and serenades her again the next day in the hopes that eventually, she'll say yes.

Now, we keep doing this every day, OK? And if we ever encounter a day where every girl has, at most, one suitor, the algorithm stops, and then every girl who has a suitor says, yes, I will marry you. Now, if a girl doesn't have a suitor, there's no one to marry her. We're going to prove that doesn't happen, OK? But determination condition is, you no longer have a situation with two or more boys under one balcony. OK?

All right, so let's run that algorithm on this example, just so we make sure we understand it, because I'm going to try to prove theorems about it. So here are the serenades that are going on, and here's the girls, and the days. It's going to work over four days in this case. And then we're also going to keep track of the boy's lists-- who's gotten crossed off.

All right, so these will be the cross outs down here. Actually, maybe I'll fit it up here if I can. Girls, let's see, A, B-- no, I'm going to have to space it out. And the boys have their lists here, and we have boys 1, 2, 3, 4, 5. And here I'm going to record the cross outs.

All right, so let's look at day one. Who is under girl A's balcony on day one?

AUDIENCE: 2, 4, and 5?

PROFESSOR: 2, 4, and 5-- each like girl A the best, so she's got a lot of activity. These three boys show up. Anybody under girl B's balcony? No, nope, no, nothing there.

C, does C have anybody? Boy 1, yeah. D? 3, and E, I don't think there's any action, right? Nope.

All right, so that's the status on day one. So the action is all up here. All right, so what does girl A do? Who does she tell to hang around?

AUDIENCE: Number 5.

PROFESSOR: Number 5, yes. She's got 5, 4, and 2, and she likes 5 the best. She tells 5 to come back, and she says to these guys, we're not going to marry him. That means that boys 2 and 4 cross girl

A off their list. So they say A is no longer possible for them.

All right, now we go to day two, and 5 goes back to girl A, and, of course, boys 1 and 3 stay there. Where does boy 2 go on day two? B.

All right, so boy 2 shows up here on day two, and, let's see, boy 5 is already there, and then we've got to get boy 4. Where does he go now? C. All right, boy 4 shows up here. All right, so now the action is with girl C. And what does she do?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Yeah, she keeps 4, and she boots poor 1. You know, led him along for a day, and then he gets the boot because boy 4 showed up, right? Because girl C likes 4 better than 1, so bad luck for 1 there. So now one goes home that night and crosses off girl C.

OK. All right, and where does 1 go on day three? All right, so he goes to-- he crossed off C-- he goes to B. All right, so this is left over. We have 5, 2, 4, 3-- boy 1 now goes to B, right? OK.

And then what does girl B do? Who does she keep around? Keeps 2, boots poor 1. All right, so 1 says, all right, I'm crossing B off my list, got the message. And now where does boy 1 go on day four?

AUDIENCE: Girl E?

PROFESSOR: E-- third choice. OK. So 1 shows up down here, and these guys keep returning. Wow, so on day four there's no more fighting. Every girl has at most one. The termination condition is invoked, and these are the marriages that take place. The girls say yes. All right, now is everybody--

[LAUGHTER]

--does everybody understand the algorithm we used? Any questions on the algorithm? Yeah?

AUDIENCE: [INAUDIBLE] the first case [INAUDIBLE] or is that just in case the [INAUDIBLE]

PROFESSOR: That's in case the algorithm doesn't work. Because we haven't proved it works yet, and I gotta have a possibility for, a boy crosses every girl off his list, he gets rejected everywhere. That is a possibility, so I've got to say what would happen in that possibility. Now, we will prove in a few minutes that condition never arises, OK? But I'm giving you what might happen, in which so it never does, so no boy ends up having to stay home and do homework here.

All right, so that's the algorithm. Now let's see if we can see if there's any rogue couples. All right, so, for example, let's look at boy 1. Boy 1 got paired to his third choice, E, so 1, C might be a rogue couple. Is that possible?

No, because C got 4 here, which is her first choice. She's not going with boy 1. That's not working.

What about 1, B? Could that be rogue? No, B got 2, and B likes 2 better than 1, so 1, C and 1, B are not rogue. And 1 got E, so there's no other possibility for boy 1, so boy 1 is not in a rogue couple.

What about boy 2? 2 got B, and that's 2's second choice. What about 2, A? Let's see, A got 5, and A likes 5 better than 2, so that's not rogue. Boy 2 is not rogue.

All right, boy 3-- boy 3 got his first choice, right? 3 got D, so 3 is not going anywhere. Boy 4 got-- who did boy 4 get-- got C. 4 got his second choice.

What about 4, A? A hates 4. A is not gonna get caught dead with 4, so that's not rogue. So 4 is OK. 4 is not in a rogue couple.

And finally, boy 5-- boy 5 is paired with A, and that is boy 5's first choice, so he's not wandering off here. All right? So in fact, we've just argued that this is a stable set of marriages, a stable matching in this case. Any questions now about what we're trying to do?

OK, because we're going to try to show now it always produces a stable matching. And to do that, we need to do a few things. All right, so what are the things we need to show to prove everything is going to be good here? What are some facts we to prove? Yeah?

AUDIENCE: Don't we need to show that the algorithm, it does come to an end?

PROFESSOR: Yes.

[LAUGHTER]

The algorithm terminates, so we need to show that the marriage algorithm, TMA, terminates. Otherwise, the boys are serenading forever, and that's not too good. All right, what else do we want to show? Yeah?

AUDIENCE: If it does terminate, then one is left empty.

PROFESSOR: Then?

AUDIENCE: Everyone gets someone.

PROFESSOR: Everyone gets married. Yep. Stability is easy if nobody gets married. All right, what else do we want to show here? Yeah?

AUDIENCE: There are no rogue couples.

PROFESSOR: No rogue couples. All right, what else might we like to show? These are the three main ones. There's a couple of other things you might like. Yeah?

AUDIENCE: It runs quickly.

PROFESSOR: It runs quickly. Well, you can only serenade for so long. In fact, it does run quickly, and that's why it's useful in practice. Anything else you might want to show about this algorithm?

AUDIENCE: How many people you crossed out?

PROFESSOR: How many people you crossed out-- yeah, you could, and that'll tie into how long it takes. Yeah. Anything else? Yeah?

AUDIENCE: The average likeness between the couples.

PROFESSOR: The average likeness, oh. So yeah, we haven't had a notion here of how happy people are at the end, because we don't have a weighting on the edges, but you might want to think about that. That's actually a good point. Yeah?

AUDIENCE: If they [INAUDIBLE] if it matters who serenades [INAUDIBLE] girls or boys to get her.

PROFESSOR: That's a great point. Is this algorithm good for girls, or good for boys? Yeah, that's a good point-- fairness. All right, so we'll take a look at fairness also. Is it better to be a serenader, or be on the balcony making your choices?

OK, so this is what we've got to do. So let's start by showing TMA terminates, and that it terminates pretty quickly. Now, in fact, I'm going to prove a fairly crude bound on the time, but it actually does fairly well.

So our first theorem is going to be that TMA terminates in, at most, N squared plus 1 days. N is the number of boys and girls. The proof is going to be by contradiction. This is probably the only day where we'll do a bunch of proofs and none of them use induction. They're pretty much all by contradiction.

Suppose TMA does not terminate in N squared plus 1 days, because we're going to show that leads to a contradiction. We need to show that some kind of progress is made each day to show that it terminates. Yeah?

AUDIENCE: [INAUDIBLE] the number of causes that have [INAUDIBLE], so if it has not terminated, then at least one girl is seeing a group of at least one guy--

PROFESSOR: Yes.

AUDIENCE: --so the number of crosses cannot be defined [INAUDIBLE]

PROFESSOR: Very good. OK, let's state that as a claim, and you've given the proof of the claim, which is great. If we don't terminate, on a day, that must be because a girl had two boys there, or more, therefore she rejected some, at least one. And that night, the rejected boy crosses a girl off his list.

So if we don't terminate, then at least one boy crosses at least one girl off his list. And so we're going to measure progress by the cross outs. So every day we don't terminate, a boy crossed a girl off his list, so if we didn't terminate at N squared plus 1 days, we must have crossed off N squared plus 1 girls across all the lists.

Well, is that possible? To have done N squared plus 1 cross outs? What do you think? How many names are on each list?

AUDIENCE: N.

PROFESSOR: N, and how many lists are there?

AUDIENCE: N.

PROFESSOR: N, so there's N lists with N names implies there's, at most, N squared cross outs ever. But we just said we had N squared plus 1 cross outs. But we have also N squared plus 1 cross outs, and that's a contradiction. All right? So we're done. It has to terminate within N squared plus 1 days.

Any questions? This is a very common proof technique in computer science. You're analyzing some system, and every step or every day or every time period, you want to argue progress got made, and then after you've made enough progress you have to be done, and therefore, the algorithm is completed. All right, so we know that TMA terminates. Now, we've still got to get everyone married and have them all be happy, or at least stable. Now, to do this, we're going to use an invariant. Yeah?

AUDIENCE: If a girl has some guy in front of [INAUDIBLE] on something, then she will always have [INAUDIBLE]

PROFESSOR: That is true. Something even stronger is true.

AUDIENCE: All the [INAUDIBLE]

PROFESSOR: Have a--

PROFESSOR: If she has a preference for someone who [INAUDIBLE] only better guys.

PROFESSOR: Yeah, that's a great invariant. As the girl is sitting there on her balcony, things only get better, because she always keeps the best one around that's there. And when new ones come in, they've got to be better than the last one for her to keep them. So as she rejects boys, she only does that because she's got better ones there, and whoever she says maybe to always comes back.

So an invariant of this algorithm is that when a girl has a suitor, going forward, she only has suitors she likes at least as well. And if she ever rejects a boy, then she's got somebody better there forever. All right, so let's state that as an invariant and prove that.

OK, so we're going to let P be our invariant here. P is the statement that if a girl, G, ever rejected a boy, B, then the girl, G, has a suitor, or if the algorithm is terminated, a husband who she prefers to B. All right, that's going to be our invariant, and now we've got to prove it's an invariant.

So we'll do that with a lemma called lemma one. P is an invariant for TMA. All right, let's prove that. Now, what's the first thing you've got to prove when you're proving something as an invariant? There's two things you've got to do to establish a variant. What's the first one?

AUDIENCE: Base case?

PROFESSOR: Base case. Show it holds true at the beginning. Proof is going to be by induction, and so we've got to show that P holds true at the beginning. But what are we going to induct on here? What's the parameter we're inducting on?

AUDIENCE: Time?

PROFESSOR: Time, the number of days. Good. The base case is day zero, the beginning. All right. Well, is that statement true on day zero?

Yeah, it's true for sort of a weird reason. Nobody's been rejected yet, so it's what's called vacuously true. Because no girl, G, has rejected any boy, so it's true. So no one is rejected yet. So it's vacuously true.

All right, next we have the induction step. So we'll assume P holds at the end of day d, and we need to argue that it holds now at the end of day d plus 1. So say it's true up to now, up to day d. Why Is it true at the end of the next day?

Well, there's two cases to look at here, depending on when G rejected B. So if she rejects B on this day, day d plus 1, well, why would she reject B on day d plus 1? There's only one scenario. Yeah?

AUDIENCE: There's a better boy.

PROFESSOR: There's a better boy. And so she says maybe to him, and he becomes her suitor. So, in fact, P is true. So then there was someone better, and that implies P is true on day d plus 1.

Case two is very similar. It's a very simple proof. G rejected B before day d, before d plus 1. Well, now we use the fact that P was true at the end of day d, which means P now implies that G had at least as good a suitor. Actually, it's better, because it was rejected. A better suitor on day d, that's what the hypothesis says, the invariant says. And now we just have to look at what happens on d plus 1.

Well, either she has the same suitor on d plus 1, or somebody better came along. And so we're going to be done. She has the same or better suitor on day d plus 1, and that implies P is true on d plus 1, and we're done.

All right, so I went through this proof. It was sort of obvious, but this is the careful way you'd write it down to show that the invariant holds. Any questions about the invariant dilemma?

All right, so things only get better for the girls. If she ever rejected somebody, she's got somebody better. So now we can prove the main result, that everyone is married. And again the proof will be by contradiction. So we assume not everyone was married.

Assume, for the purpose of contradiction, that some boy, B, we'll call him, is not married. Because if everyone is not married then some boy is not. If not everyone is married then some boy is not married. So when it terminates, B is not married. Well, what do you know about B if he was not married at the end?

PROFESSOR: He was rejected by everyone.

PROFESSOR: He was rejected by everyone, because if at the end, he's still under a balcony, if he still had somebody on his list, he'd be there. And he'd be getting married, because it's the end. So this means that if B is not married, he's rejected by everybody. Yeah?

AUDIENCE: [INAUDIBLE] list?

PROFESSOR: Oh, he's on everybody's list.

AUDIENCE: Yeah.

PROFESSOR: Everybody has everybody of the opposite sex on their list, but he had to cross everybody off. That's true. Yeah?

AUDIENCE: That would mean that everyone had somebody better on the same day that [INAUDIBLE] telling him she didn't want to be with him.

PROFESSOR: That's true. That means that B crossed every girl off. B is rejected by every girl, which means every girl has somebody better than B, which is not possible, because that would mean every girl was married. And therefore, the equal number of boys and girls, that means B would have been married. Good. All right, let's write that down.

This means that B was rejected by every girl. OK, that means that every girl, by lemma 1, has a better suitor, and that's where we use lemma 1. And that means that every girl is married, and that means that every boy is married, including B, and that's a contradiction, because we said B wasn't married. OK? Everybody buy that proof? Any questions about that?

Yeah, proof by contradiction is a pretty powerful technique. Once you assume something is not going to be true, it gives you a lot of power to find a contradiction. All right, so now we know that the algorithm ends, and everybody gets married. All's we got to do is show that there's no rogue couples, so let's do that.

TMA produces a stable matching. Now, how do you suppose we're going to prove this? What's going to be the approach to prove this? Yeah?

AUDIENCE: Assume that there's a rogue couple.

PROFESSOR: Assume that there's a rogue couple, that namely this is not true, so there must be rogue couple. So let Bob and Gail be any pair that are not married. I need to prove the Bob and Gail are not rogue, and then we'll be fine. Because if it says everybody who is not married is not rogue, then we know we have a stable matching.

Now, there's a couple of cases here. Bob and Gail weren't married, so there's two ways that could have happened. What's one of them? What's one reason they might not be married, something that happened that made that impossible? Yeah?

AUDIENCE: Gail rejected Bob.

PROFESSOR: Good. Case one-- Gail rejected Bob. Well, what do we know in that case, if Gail rejected Bob?

AUDIENCE: Gail had better suitors.

PROFESSOR: Gail has another suitor that she likes better. And, in fact, what do we know about who Gail married?

AUDIENCE: Better than Bob.

PROFESSOR: Better than Bob-- things only get better for the girls. That's the lemma one. All right, so this means that Gail marries someone that she thinks is better than Bob. And that's by lemma one.

Well, can Gail and Bob be a rogue couple here? No, because Gail likes her spouse better than Bob, so she's not going to be in a rogue affair with Bob. So that means that Gail and Bob are not rogue.

All right, case two is Gail did not reject Bob. She never did. That's the other case, she didn't reject Bob. Could Bob have ever serenaded Gail in this case?

No, because if he did and he was never rejected, they would have ended up married. All right?

So that means the Bob never serenaded Gail. What does that mean about how Bob feels about Gail? Yeah?

AUDIENCE: Bob must have never serenaded for Gail.

PROFESSOR: Yeah, Bob never got far enough down on his list to serenade Gail. He got married before he got down there. All right, so that means that Gail is lower on Bob's list than Bob's wife. And that means that they're not rogue, because Bob likes his wife better than Gail.

All right, so in each case, the cases clearly cover everything. Gail rejected Bob, or she didn't. Either way, Gail and Bob are not rogue. So that means there's no rogue couples, and that implies M is stable, TMA is stable. OK?

All right, so TMA terminates, everyone gets married, no rogue couples, nice outlook. So we're done. We actually proved it works. One issue left to think about here. Any questions on that before we launch off into the last issue?

AUDIENCE: Is it unique?

PROFESSOR: Is it unique? TMA gives you a unique answer because it's an algorithm. It's deterministic. But there may be other stable matchings. OK, so there's not just one stable matching, necessarily. You could make examples with multiple stable matchings.

That's a great question. Any other questions? All right. Oh, yeah?

AUDIENCE: Is there generally any other way to assess optimality besides the fact that it's stable, or--

PROFESSOR: Yeah, you can make up lots of them. You could put weights on the edges and get a min-weight matching. You could try to get the perfect matching with the least unfavorable marriage kind of thing. There's a lot of criteria you can make.

This one turns out to be useful in practice in a variety ways that we'll talk about, and also have a nice, fast, simple algorithm that can actually run in a distributed environment, which is really nice. So it is probably the most practical approach to matching out there. Any other questions?

OK, the last issue-- I don't know if we still have the issues up there-- is fairness. So who thinks the TMA is favorable to the boys? Just a couple.

Who thinks it's favorable to the girls? More, that's the common response. Who thinks you can't even define it one way or the other? It's unclear and hopeless to decide.

OK, well, it seems like maybe the girls, because they get the best of their suitors. They sit back, and they just take the best as they come along. On the other hand, the boys do try to go out and get their first choice. The girls have to wait. And Mr. Right may never come along.

The boys are out there. I'm going to my first choice, and they get denied, OK, they just move right on to the next choice. So this is actually one of these questions of study in sociology. In the animal species, which is better, proposers or acceptors? What's the more powerful result? Who has the better power in courtship?

It turns out that we can answer this question in a very clear way, and prove it here. And the answer is the boys have all the power here. This is very favorable to the boys, and we'll see why.

Now, to prove that, we need some definitions. OK, so I've got to define a couple things here to be able to prove this. The first is, for any collection of preference lists, we're going to let S be the set of all stable matchings. Now, we know that S is not empty, right? How do we know that S is not empty here?

AUDIENCE: [INAUDIBLE] stable.

PROFESSOR: Yeah, because TMA produces a stable matching, so we know that S is not empty. There's at least one. And, in fact, there could be many.

Now, for each person, P, we define the realm of possibility for P to be the set of mates that you might have in a stable matching. So it's a set queue of people for which there exists a matching that's stable such that you're mated to that person. So I've written this in some-- that's sort of complicated mathematics, but somebody is in your realm of possibility if there is a stable matching where you married them. If it exists, there's a stable matching where you could marry them.

And this is vaguely like you see sometimes, you're going out with somebody and your parents say, oh, they're not in your league, or something. This is a mathematical formulation for that, that there is some way in the world to marry everybody up, so it's all stable, and you could be married to that person. OK? Let's do an example with four people.

All right, so for example, say we have Brad, Jen, and Angelina again. And of course Brad likes

Angelina and vice versa. Gen likes Brad, and then there's Billy Bob here. All right.

Now, Brad is not realistic for Jen. Brad Is not in Jen's realm of possibility, because in any perfect matching where Brad marries Jen, you're going to have a rogue couple. Brad's going to go for Angelina. So Brad is not within Jen's realm of possibility.

And similarly, vice versa-- Jen is not in Brad's realm of possibility, because there's no stable matching where they're married, in this scenario, for this problem. All right? Now that we have that notion, we can define who your optimal mate is, and who your pessimal mate is, the worst case.

All right, so we define a person's optimal mate is his or her favorite in the realm of possibility. So your optimal mate is not your favorite overall, it's just your favorite among those who are possible, that there is a stable matching that you could be matched to that person.

And similarly, you have a person's pessimal mate is your least favorite from the realm of possibility. OK? Because you have all the people you might be married to in stable matchings. Your favorite is the optimal, your least favorite is the pessimal.

Any questions about that? Does that make sense? And you don't count the ones you can't be married to, because if you were, it would be unstable.

OK, now we can state the blockbuster theorems here. Theorem four says TMA marries every boy with his optimal mate. Theorem five, probably can guess, TMA marries every girl with her pessimal mate.

All right, so it is optimal for every boy. They get the best possible they could have in stable marriages, and every girl gets the worst possible mate. Now, you wouldn't necessarily think that when you first look at that algorithm, where the boys are working down their list, and the girls are just getting better and better. But that's the case.

Now, let's see, I'm not going to have time to prove them both, but I will prove theorem five. And I'm going to assume theorem four is true to prove theorem five. Maybe we'll do theorem four tomorrow in recitation. I don't know. So let's do theorem five so you can see how this works.

The proof is by contradiction, so we'll assume theorem four is true, and we'll prove the proof of theorem five by contradiction. So assume there was a stable matching where a girl got worse off than in TMA. All right? Assume three and five is not true, then there's some girl in some stable matching.

So suppose, for the purpose of contradiction, that there exists a stable matching, we'll call it M, where some girl, and she fares worse than in TMA. I want to show that results in a contradiction. And the proof is pretty simple, just a simple picture. So G did worse in M than in TMA.

All right, so let's let B prime be the mate of G in M. So girl G did worse in M than in TMA, so here's her mate in M, and let B be her mate in TMA. Who does she like better, B or B prime? B, because we're saying that she did worse in M than she did in TMA, so she likes B better.

And let G prime be the mate of B in M. Now, who does B like better, G or G prime?

[INTERPOSING VOICES]

PROFESSOR: Not G prime. B got married to G prime in M, but what does theorem four say? Yeah, G. Theorem four says that across all the stable matchings, B gets his favorite mate in TMA. So G prime is in the realm of possibility, and so is G, and we know from theorem four B likes G best, better than G prime, because of theorem four.

So what happened here? There's a rogue couple in M. B and G are a rogue couple in M.

And that means that M is not stable, and that is a contradiction. All right? And so theorem five is true, because we assumed we had a stable matching M where there's a girl who did worse off.

So it really pays off to be aggressive party in courtship. Everybody lives happily ever after, especially the boys in TMA. Now, TMA arises in lots of applications. The most famous is the matching program that's used to match MDs to residency programs.

So how many pre-meds are here? There's one, at least. So you're going to go to medical school some day, and at the end of medical school, there's a big day called match day, where you get assigned to a hospital for your internship, and the way that works is using this algorithm. And who do you suppose is the boy in this algorithm?

AUDIENCE: The hospital.

PROFESSOR: The hospital. And they want it so that when they make the matchings, that there's not some pair of an intern or doctor in a hospital where they'd like each other better than what they got. This is used in online dating for the obvious reasons, and we actually use this algorithm lot at Akamai to load balance traffic on the web.

And here you have the boys are web servers, and the girls are requests for service. And the goals are to balance performance, getting a server that's nearby you that's fast. And on the other side, our cost, and by who we make a boy and a girl, we can trade off cost for performance in a very nice, distributed way. All right, so that's it for today.