PROFESSOR: Graph coloring is the abstract version of a problem that arises from a bunch of conflict scheduling situations. So let's look at an example first and then define the problem. So let's think about a bunch of aircraft that have to be scheduled on the ground at jet ports or gates.

Now, if two flights are on the ground at the same time, they need to be assigned to different gates since a gate serves one airplane. And what we'd like to do is try to figure out how many different gates do we need to be able to service all the planes that might be on the ground. How many gates are needed?

So let's look at a sample schedule. There are six slides here numbered 122, 145, through 99. And the horizontal bar is, say, times during the day.

And this blue block indicates that flight 122 is on the ground from, let's say, 3:00 AM to 7:00 AM, and flight 145 is on the ground at a completely disjoint time interval. So is 67. 257 is on the ground from midnight until about 6:00 AM. It does overlap with 122, and so on.

So this is the information we have. And what we're trying to figure out is how many gates do we need. Well, it's easy to see here that the worst case, if you just think of this vertical green line sliding across the bar, and you look at the maximum number of blue intervals that the green line ever crosses, it's three. The largest number of planes that are on the gate at any given moment is three, which means we can get by with three gates.

So we have to cope with that conflict. So abstractly, what we're going to do is assign each aircraft to be a vertex of a graph. And we're going to put an edge in to indicate not compatibility, but conflict. Compatibility was what we were looking at previously with our examples of matching. Now this line means that 306 and 145 are on the ground at the same time. They conflict. They need the same gate, and we have to serve them with different gates.

And likewise, 99 and 145 are on the ground. 306 and 99. And this was the three flights that were on the ground at the same time. And then if I fill in the graph with all the other vertices and draw an edge when two flights are on the ground at the same time, I wind up with this little graph.

OK, now we can talk abstractly about the coloring problem, which is let's assign colors to the vertices in such a way that no two adjacent vertices have the same color. Adjacent vertices should have different colors. And it should be clear from the description of how we derive this graph from the aircraft schedules that the minimum number of distinct colors needed to color the graph corresponds to the minimum number of gates needed to serve the aircraft.

So let's try coloring this graph. I'll start with coloring 257 red, and 122 yellow, and 99 green. There's no loss of generality here because these are the three that are on the ground at the same time, reflected by the fact that they're in a triangle. And I'm going to have to use three different colors since each one is adjacent to the other two.

OK, what next? Well, let's color 145 yellow. I might as well reuse it since it's not adjacent to a yellow vertex. And then here, I've got another triangle. So if I'm not going to use an extra color, the sensible thing to do would be to color that red. But oops, I didn't do that. I used a red here.

There's another triangle, I guess, that allows me to color. And then I color this black because here, I'm stuck. I'm adjacent to both a yellow, a black, and a green vertex. So I have to come up with a fourth color.

All right, we did it with four colors. It means that we could have gotten away with four gates. And the colors tell us which aircraft to assign to which gate. So 257 and 67 can both be assigned to the red gate because they are not on the ground at the same time. There's no edge between them. 122 and 145 can be assigned the yellow gate, and so on.

Now, this was not the smartest way to color. A better coloring is shown here. You can check that every two adjacent vertices have different colors. And now I've done it with only three colors-- red, yellow, and green. So now there are three gates and I get a better schedule.

Another example of this kind of conflict problem comes up with scheduling final exams. Two subjects conflict if a student is taking both. Because if a student's taking both, I can't have the final exams at the same time. And so I need to assign different time slots during exam period to subjects that overlap, that have a student in common.

And then the question is, given this data about which pairs of subjects have a student in common, we want to know how short an exam period can we get away with. Again, it becomes a simple graph model and a coloring problem.

So here, we've drawn a graph with some sample subjects. 6.042 and 18.02 have a student in common. That's what that edge means. They need to have final exam scheduled at different times. Likewise, 8.02 and 6.042 have a student in common, so they need to be scheduled at different times.

On the other hand, 6.042 and 18.02-- sorry. What are some two that are not adjacent? 3.091 and 18.02 have no edge between them, which means that they can be scheduled at the same time. There's no student who's taking both 3.091 and 18.02, at least according to the data in this graph.

So let's try coloring it. And again, there's a triangle. I'm going to have to use three different colors for a triangle. And here's another triangle. And to be economical, let's just reuse green. Now, here, I've got another vertex that's adjacent to three different color vertices. And so it's going to have to be colored with a fourth color.

This time, it turns out that the four colors are best possible. You can check that. And it corresponds to a schedule in which the 6.042 is scheduled on Monday morning at 9:00, and 6.001 is scheduled on Monday at 1:00. But 8.02 and 3.091 can both be scheduled for Tuesday 9:00 AM. And finally, 18.02 is scheduled on Tuesday at 1:00 PM.

OK, so this kind of a conflict modeling situation comes up all the time. Another place where you get these kind of compatibility graphs or incompatibility graphs that you want to color would be if you were running a zoo and you had to have separate habitats for certain kinds of species of animals that you don't want to mix together.

Big fish eat little fish. It's a truism in the aquarium world. And so you need to keep big fish separate from little fish. And you don't want the tigers living together with the chimpanzees. So we could again model this problem as how many cages do we need. We create a vertex for each species and put an edge between two species that mustn't share a habitat or share a cage.

Another one would be assigning different frequencies to radio stations. And again, if two radio stations are close to each other, they will interfere. So they have to be assigned to different colors or different frequencies. So now, we would be using radio stations as vertices. And radio stations that were near enough to interfere with each other would get connected by an edge, indicating that they needed to be assigned different color frequencies.

And one of the classic ones is literally to color a map. If you were trying to take, say, a map of the US and assign colors to it in such a way that you never had two states that shared a border with the same color-- and this is an illustration of doing it with four colors. The question is if I give you some kind of a planar map like this, what's the minimum number of colors that will work?

Now, you're allowed to have two countries share the color if they only meet at one point. But if they have a positive length boundary, they have to be different colors.

OK, the way that this turns into a vertex coloring problem is if you take a planar graph like this- - here's just an arbitrary one-- what I can do is I'm interested in coloring the regions, the countries, with different colors, but I'll just replace each region by a vertex. So I'm going to stick a vertex in the middle of each of the regions. Notice there's an outer region here too that gets a vertex. So one, two, three, four, five, six regions, or six vertices.

And then I'll simply connect two vertices when there is a positive length edge that their regions share. So that edge corresponds to the fact that there's this boundary that's shared between this region and this region. If you look at this same triangular-shaped region, it has a boundary with the outside region. So there's going to be an edge from here to the vertex that represents the outside. And there's the rest of the edges. An edge appears between two regions that share a boundary.

And now, the question is coloring the countries corresponds to coloring the vertices. And we'd like to color the graph with as few colors as possible. Well, a famous result that was proved in the '70s is that every planar graph is in fact four-colorable.

Now, this was first claimed to be proved in the 1850s, but in fact, the published proof was wrong. It sat around in the journal literature for decades before somebody found a bug. Or that is to say that the proof was wrong, not the result. There was a big hole in the proof that had not been adequately justified.

The proof did give a correct argument for five coloring, and the four color problem was opened for over 100 years. Then in the 1970s, two mathematicians came up with a proof of the four color theorem that was very controversial because a lot of their proof required a computer program to crank through several thousand sample graphs that needed to be verified for four-colorability.

They had an argument that showed that there could only be a few thousand counter examples if there was-- or rather, if there was any graph that couldn't be four colored, it would be one of these several thousand graphs. And then they went to work on coloring these several thousand graphs, which were generated with the aid of a computer and then colored with the aid of a computer, and also by hand.

This did not make the mathematical community happy because a proof like that is essentially uncheckable. A much improved version was developed in the 1990s, but it still requires, in the end, a computer program to generate about 600 maps and verify their colorability. So it remains to find a proof of the four color theorem that you could say is humanly comprehensible without the aid of a computer. But there's no longer any doubt, really, about this theorem in the mathematical community.

In general, if I take an arbitrary graph and I ask what's the minimum number of colors to color it, that's called the chromatic number of the graph. So chi of G is the minimum number of colors to color G. Let's look at some chis for different kinds of graphs.

So here we have a simple cycle of length 4. And it's obvious that that can be colored with two colors-- just alternate the colors. On the other hand, somewhat-- and in fact, generalizes, by the way, to any even length cycle. The chromatic number of an even length is simply two. You color the vertices alternately.

On the other hand, if the cycle is of odd length, you're going to need a third color. There's no way around it because even if you try to get by with two colors, then you color things alternately. And then when you wrap around, you discover that you can't alternately color. You're going to need a third color in order to avoid a clash. So in general, the chromatic number of an odd length cycle is 3.

The complete graph on five vertices is shown here. This is a five vertex graph in which every vertex is adjacent to the other four. And since every vertex is adjacent to the other four, you're going to need five colors. You can't do any better. They have to all have different colors. And so the chromatic number of the complete graph on n vertices where every vertex is adjacent to the other n minus 1 is n.

Another simple example that comes up is if I take the cycle and I put on an axle in the middle-- we call it a wheel then. A wheel with a cycle of length of 5 around the outside, a perimeter of length 5, is called W5. And we can color it with four colors.

And in general, the argument that the chromatic number for an odd length wheel is four is that I know I'm going to need three colors to color the rim. And since the axle is adjacent to everything on the rim, I'm going to need a fourth color for it. On the other hand, again, if the perimeter is even, then I can get by with three colors.

One more remark about chromatic numbers is there's an easy argument that shows that if you have a graph, every one of whose vertices is at most degree k-- there are at most k other vertices adjacent to any given vertex-- then that implies that the graph is k plus 1 colorable. And the proof is really constructive and trivial.

Basically, you just start coloring the vertices any way you like subject to the constraint that when you color a vertex, it's supposed to not be the same color as any of the vertices around it. But that's easy to do. Because when it's time to color some vertex, even if all the vertices around it are colored, there's only k of them. And so I will always be able to find a k plus first color to assign it and get us a satisfactory coloring. So I can get by with k plus 1 colors.

Now, the general setup with colorability is that to check whether a graph is two-colorable is actually very easy. And we may talk about that in some class problems. But three-colorability dramatically changes. We're back in the realm of NP-complete problems.

In fact, a result of a student of mine almost 40 years ago was that even if a graph is planar where you know it can definitely be colored with four colors, determining whether or not it can be colored with three colors is as hard as satisfiability. And it is, in fact, an NP-complete problem. In fact, a proof of how you reduce satisfiability to colorability appears in a problem in the text, which we may assign as a problem set problem.

So in general, finding chi of G is hard, even for three colors. Now, it's not any worse, again, from a theoretical point of view for checking what chi of G is even if it's a very large number, although pragmatically, three color will be easier to check than n-colorability. And that is our story about colorability.