PROFESSOR: All right, welcome back to 6.890. Today we continue our theme of reductions-- NP-hardness reductions from three partition. Yesterday we saw a couple of different reductions from three partition and then a bunch of reductions from those problems, various puzzles. The last one we covered was packing squares into a square. This is just a reminder visually of what it looks like. But we're going to just take this result as given.

And our first NP-hardness reduction will be from packing squares into a square. And then we will see a bunch of-- possibly as many as eight-- reductions from three partition. So without further ado, let's get started.

The first problem we're going to talk about is edge-unfolding polyhedra. This is a geometric folding problem. So if you're interested in more, you can take 6.849. But here's a simple-- if you've ever made a cube out of paper, probably you cut out a cross shape and then folded it into the cube. But if you want to know what shape should I cut out to make a cube, then you want to know where should I cut, which edges of the cube should I cut? They're drawn in red here. So that, when I unfold it, I get one connected piece without overlap.

So for surfaces that are topologically a sphere, you want to cut along a spanning tree of the edges. And for a cube, that will always work. But for some polyhedra like these two, when you unfold, you get overlap. And in fact, no matter which spanning tree of the edges you take for either of these two polyhedra, you're guaranteed to fail. You cannot make these out of one piece of paper by folding and get an exact cover.

So that's bad news. And these are old results. These are back in 1998. But it

remained an open problem, how many other examples are there? If I give you a polyhedron, I want to know is there an edge unfolding? We only cut along the edges. What's the complexity of that? And the answer is, it's NP-complete, strongly NP-complete, slightly stronger.

This is not, obviously, a number problem. But in geometry, you have this issue of what are the coordinates of the vertices. So the coordinates could be exponentially large or they could just be polynomially large. Here they only be polynomially large, so strongly NP-complete. This is a complicated proof so I'm just going to sketch the idea of how it works. At the beginning, like most proofs, there's some kind of like overall infrastructure the holds all the pieces together. This is one part of that. Well, this is the main infrastructure in this reduction.

There's some stuff around the side. This is mainly to make the surface topologically a sphere. The focus is on a square face here that has a little slit in it. And there's a tower coming out. It's drawn edge-on here. This is the part we actually care about. I think, actually, it will be redrawn here. So we'll really just be thinking about this. And then, there's some stuff that wraps around it makes it a sphere without changing unfoldings So don't worry too much about that.

Mainly we have this giant-- think of it as an obstacle because we are not allowed to overlap when we unfold this thing. So this shape can't be cut up because it has no edges except for here.

So this part can move around it. It turns out that it can just stay in there but it might move up. So we'll make a little bit of space here. There's this hole of this face. Then there is this tower that is sticking out into the third dimension and we need to somehow unfold that thing and fit it in here. Doesn't look possible because this is not quite drawn to scale. This is going to be super long. So a lot of the stuff here is just going to be stuffed

up the chimney, so to speak, but also on the surface there is more stuff going on. So let's zoom into this thing. If we unfold that tower into a plus sign. Most of it is lots of tiny stuff called atoms, which we will get to. And then, the interesting part from a

three partition standpoint-- or sorry, from a square packing standpoint-- is there are some square faces.

And because we're edge cutting our surface those squares will remain squares. There's no edges on them. So those guys somehow have to move around when you unfold. And it's not drawn this way, but the squares are large enough that they can't fit up the chimney. OK, this chimney's actually pretty narrow, constant width, and those guys are all really big.

So this other stuff can move up the chimney, but the squares will have to stay in this square. And that's how we're going to get packing squares into squares. OK? Now, the challenge is, what we want to be able to do is move the squares around freely. But in reality, we need to make a connected 2D shape. So somehow, we need to connect together these squares by some stuff-- that's the mystery atoms-- so that, basically, for any square packing, we can arrange to have those faces in that rough arrangement. And then the other stuff somehow connects everything together. And whatever doesn't fit in here goes up the chimney.

So that's the plan. If we zoom in-- all right, so this is sort of the diagram of that plan. Imagine you have some placement of the squares. And we're going to try to connect all the pieces together maybe to some common root in the lower left corner. First we're just going to draw some L's conceptually to connect them all to here. Those cross, which is not allowed. But then we're going to reroute the L's to go around the-- so, to get up to this box, we're going to go around all the other boxes below it. And in this way, we get a non-crossing arrangement.

If we zoom in even closer, these, I've drawn as lines, but really they're little strips of paper. And they may do weird things like zigzag back and forth. How that happens we'll see in a moment. But this is to convince you that, at least if we could place squares and they're not like right touching each other. So we're going to shrink each square a tiny bit, which I won't get into the details. But if they're integer square, let's say, then we'll shrink them, maybe, by minus 1 over n or something polynomially small.

And then, that gives us a little bit of room to route all of these paths if make them like 1 over n squared or 1 over n cubed in thickness. Still polynomial, so still strongly NP-hard. So if we have a square packing, we can connect them all up in this way if we have these magical atoms that can do whatever we want freely. And that's the-- so, in a little more detail, in this picture, we imagine the atoms as somehow being connected together in a chain, potentially. And then when they unfold, this is the fun part.

So we maybe-- for connectivity purposes, maybe first we want to visit B1, then we want to visit B2, and so on. And that corresponds to some traversal on the surface of this plus sign or the tower that's sticking out. And then, in 2D when we unfold it, we have a different thing in mind of whether we want it to go left or right. So there's moving left or right intrinsically on the surface. And then there's moving left and right in the unfolding. So two separate things.

And conveniently, this one structure is one of these boxes-- this is what we call the atom- can do all of those things. Certainly not obvious from the picture, it's just a little bumpy square. But you just trial the cases, these are three of the cases. Not obvious, but there turn out to be 10 after you remove reflections and rotations.

There's a few things going on, but the main thing going on is whether you're going left or right in the-- let me get this straight-- whether you're going left or right intrinsically on the surface and whether you're going left or right or straight in the unfolding. So here you can see you're turning left in the unfolding, you're going straight in the unfolding, and you're turning left in the unfolding. So the second coordinate is L, S, and L.

What's less obvious is if you fold this up and you get the atom structure, but the thing that you attach to here is on the folded structure either to the left or to the left or to the right. So you just check all these things. And then, this is like a universal glue that can connect together the squares however you want, both on the surface where they have to be connected originally and in the unfolding where you want to match some square packing. So we'll have a little bit of hand waving there, lots of

hand waving, but in the end, you get NP-strong, NP-completeness of edge-unfolding polyhedra.

Any questions? All right, perfectly clear, then?

Let's move to another puzzle. This is called the snake cube puzzle or the cubra puzzle. Here is one instantiation of it. So we have-- it's a chain of blocks-- I'm going to try to unfold it in a way that I can put it back together, because it's actually quite tricky to solve. It's a chain of blocks and there's an elastic through the centers, going through the center of the faces. And so, the result is you can't really pull the squares apart, it's a pretty tights elastic. But you can spin. Because from the elastics point-of-view, these are all equally happy.

So you're basically told that-- I could also spin here, but it doesn't do anything if I stop at a 90 degree angle. So I'm forced to be straight here. And then I'm forced to have a 90 degree turn. And then I'm forced to go straight and then have a 90 degree turn and then go straight and then have a 90 degree turn. Here I have a few 90 degree turns in a row. And that's the general structure. And can I still solve it? Maybe. The goal is to get into a cube.

And there are a bunch of different versions, plastic, wood, whatever. This one is not solved and so I will not do so well, but you get an idea of what it looks like in its sort of flat state. I can make it into a kind of staircase. So in general, this puzzle is specified by a sequence of lengths of straits. And then, in between those lengths are the 90 degree turns. OK?

So that's the puzzle, pretty simple specification. And your goal is to fold it into a cube. You know how big the cube is. You just take the number of little cubes and take the cube root. So this puzzle maybe goes, is at least 1990, but could be older. We don't know exactly. And this puzzle is also NP-complete.

And this proof is also complicated so we won't see all the details. But this one, we'll probably see a little bit clearer. This is a proof by, in particular, the three of us and Marty in the back, so all four of us. And let's see, so this doesn't look like a cube. So

lets-- before we try to solve the problem of folding into a cube shape, let's suppose I give you one of these chains and I want to know, can it fold into this shape? Let's say this is the target.

So it's a big box down here. I think we call it the hub. And then there are these spokes, or just big boxes, long boxes, sticking out of the hub. There's a hold here. They're connected to the main structure. And you want to fold that. And let's see, so the big idea-- we're reducing from three partition here-- we're going to represent a number, ai, as a zigzag-- well, sorry, shouldn't call it a zigzag-- but we have a really long bar, then a length 1 bar. And then a really long bar and a length 1 bar. I'm drawing sort of the thinned version. This is really a little cube. This is a little cube. These are lots of cubes, huge number of cubes. OK? So that's really, really big. So big that the only place it could fit is down one of these spokes.

So of course, the idea is these represent our triples whose sum is equal to t. And so, we have to take these things and because these are so short, you can't like go out of one hub and jump to the next one because there's a gap there that's bigger than 1. So once you commit to having one of these long bars into one of these pockets, then all of them have to go into the pockets.

So this is, in general, a big issue with three partition proofs is you want to represent the number ai in some way that's not divisible. If you could take ai and split it into two parts, than the packing problem becomes trivial. You'd split everything into units or something. So we have to make sure ai remains a single chunk.

We are, of course, encoding ai in your unary here, because we have one long bar per each ai. But for three partition, unary is OK. There's some things going on here a little bit weird. So you see this thing is occupying the full length. And the length here is going to be 8 times the target sum instead of just the target sum. And we're multiplying this by 8. So those are equal, we're fine. There's a reason for the 8 which we will get to.

OK, now, like the previous proof, the edge-unfolding polyhedra, we have these great units. And we just like them to be free floating. And if you generalize this

puzzle little bit and say, well, actually, I don't just have one chain. I have two chains. Or I have n chains. And together, collectively, they have to fold into this shape, then we're done except that you would remove the box, the box over here. Because then, these guys just somehow have to be assigned to these little pockets and we're happy. OK.

All of these numbers-- although again, not drawn to scale-- all these numbers are much smaller than huge. So you can't like take this guy and put it here, even though in this picture it looks like you might be able to. All right. So the issue is, now we want to connect this together into one big chain. We have these units. For each i, we have one of these units. That's cool. And we want to attach them together in a way that is super flexible, just like the atoms we had before.

Now we're going to use something which we called the zigzag, that's why I didn't want to call this a zigzag. This is all unit segments. So you have 90 degree turn, 90 degree turn, 90 degree turn, 90 degree turn. No straits. This turns out to be universal in a strong sense. It can fold into anything. And so, we can use it to connect these parts together. So what we're going to do is have the A1 gadget. Then we're going to have a whole bunch of zigzags so you can go wherever you need to go. And then we'll have the A2 gadget. And then we'll have the universal zigzag gadget for a long length of it. And then the A3. And then alternating between zigzags and the ai.

And at the end, we'll have enough zigzag that we can fill all this sort of extra garbage space. But we will use this garbage space in order to route around. So maybe A1's going to go here. And so we do this thing, this, yeah, looks like some kind of cooker. We go back and forth like that. Then we'll have the zigzag and say, OK, I want to go over here next. And then we'll do A2 there. And then we'll route over here.

And because this is in 3D, crossings are not an issue. I can just got out into the third dimension. We don't need much of a third dimension, 8 is enough. You can just, enough to go over each other. OK. And then at the end, we're going to fill in all this

space with the zigzag.

So let's talk about the zigzag. What we claim-- there's a couple different senses in which it's universal. But the key idea are these sets of gadgets. So we're drawing here a two by two by two cube made up of 8 little cubes. And if you look at the-- what we've drawn here is the elastic that goes through the centers of the cubes. And notice it turns 90 degrees at every voxel, at every little cube. So this is part of-- these are zigzag constructions.

So what we want to say is that, if you sort of-- so one sense of universality is that the zigzag structure can fold into any polycube structure. Polycube is the 3D generalization of polyominoes that we saw last time from polyominoe packing. That's not true. But if you take any polycube structure and you just scale it by a factor of 2-- so you replace every cube with a 2 by 2 by 2 grid like this picture-- then it's still not true. But it's better.

So what's true at that point is if you take any path of two by two by two cubes, this is what we would call a Hamiltonian shape, because there's one path that visits everything with no collisions. So it has to be a path that does not hit itself in 3D. Then you can follow these gadgets in order to make any turns you need to do. Or you can go straight, everything is possible. I will talk a little bit more about that in a moment.

So 2 by 2 by 2 refinement, meaning scaling everything by a factor of 2, allows the zigzag to make any Hamiltonian shape scaled by 2. But that's Hamiltonian shapes. Now conveniently, there's this theorem which is used a lot in computational geometry and we use it in this harness proof, that if you take any shape and you refine it by two by two by two, every scale by a factor of 2, then it becomes Hamiltonian by a similar kind of gadget construction. Although it's easier-- if you just want Hamiltonian paths, you don't care about where you're turning and where you're not turning, so that's a pretty easy proof left as an exercise to you and sort of an aside from this issue.

So we take any shape we 2 by 2 by 2, refine it, we make it Hamiltonian. Then we

two by two refine it again. In total 4 by 4 by 4 refinement. And then, a zigzag path can make that polycube shape. And this is for any polycube. So that's cool.

Now, I'll mention here in this proof, one of the issues which we'll see a bunch of times, I'm just going to mention it here, is we have a parity issue because the cubicle grid is two-colorable, right, you can checkerboard the 3D grid of squares. And so, what that means is every time you go from one cube on the grid to another cube on the grid, you're switching parity. If you checkerboard the 3D space black and white, you're always going from white to black, then black to white, then black to white, sorry, alternating black and white, easier to do than to say.

And so, that's a constraint. So for example, it is impossible in a 2 by 2 by 2 thing, to enter from this cube, the front cube, and exit from this one just by a counting argument. So you start, let's say this is a black cube. This one also a black cube. And so, you start black and then you alternate black, white, black, white, and there's 8 things to visit and you want to end black, that's not possible. You have to end white. So if you enter on a black cube, you will leave on a white cube. And if you enter on a while cube, you will leave on a black cube. So that's a constraint. Doesn't really get in the way here, but it's something you have to keep track of.

And the theorem wouldn't be-- what you'd like to say is, well, I can enter from any cube and leave from any cube. And you could actually prove that by induction, except the base case would be wrong. So this is something to be careful about. This is the base case. This is 1 cube, can I enter from anywhere? One 2 by 2 by 2 super cube, can I enter anywhere, leave anywhere?

The answer is no, but I can enter and exit anywhere that have the right parity. So if the entrance cubes and the exit cubes have opposite parity, then this is actually all of the cases. These four are where the entrance and exit cube are adjacent to each other. And these two cases are where they're opposite from each other, not touching. And that's all the cases with all the rotations and reflections.

So once you prove that, base case is actually really easy to do by induction, you just are essentially pasting all these things together to make any path. And then we

could make any Hamiltonian shape. And then further refinement, any shape becomes Hamiltonian. So that's how we prove zigzags are universal. And this connects together all of these gadgets. There's really two gadgets in this proof, the ai gadgets-- there's n of n copies of that-- and the zigzag gadget-- there's n copies of that as well.

Now there are some details to check here, of course, that the zigzag gadgets can actually navigate around and not cross each other with height, 8, and that at the end, you can fill all the blank space. If you're just organized about how you traverse each of the ai's and where you put the blank space, I think we just leave some headroom. So you can just go up, walk over to where all the blank space is, just serve do a scan line traversal. Wherever there's blank space, you eat it up with your zigzag at the end.

So what this does is it let's the ai's move around completely freely. There shouldn't be any constraints on how the ai's are assigned to these pockets. Because we want every solution to the three partition instance to work here. And the zigzags let us do that because they're so universal. Cool?

So I think that's what I wanted to say about this proof. Double check my notes. Ah, there was one other thing which I found a little funny re-reading this paper is why 8? Because we know everything has to scale up by a factor of 4. And that's all we're doing in some sense. But we do make these-- we scale all the ai's by a factor of 8. Not a big deal, but there's a reason for 8.

If we just did 4 and say-- so we want to go, so this thing gets mapped onto here and we go out and back and out and back. If we just did it four times, we would be arriving at basically the same position in the 4 by 4 by 4 block. Or sorry, we would be arriving at the same 4 by 4 by 4 block that we started from. And that's not good because we can only visit a block once. So this is sort of an interface issue between the zigzag-- which is nice and universal, you can always paste two of these chains together, no problem-- with this thing, which is not universal. It folds into basically this shape. That's it.

So we need to make sure that, I mean, we can universally do whatever we want in here. But we need to make sure that when we connect between this zigzag structure and then we paste in one of these and then we come back, that we do that in a valid way from the perspective of this big trunk So the factor of 8 makes sure that if we leave from one 4 by 4 by 4 chunk, we come back on a different one. And then it's easy to paste all the paths together. So that is [INAUDIBLE]. Yeah.

AUDIENCE: So, I don't quite see that this proves that snake cube is NP.

PROFESSOR: Right. Ah, yeah there's one more issue, which is this was folding this shape. But we want to fold a cube. Yeah. This is the kind of annoying part of the proof. It's very hard to draw. This isn't my attempt at drawing it. And yeah, I will sketch the idea.

I tell you, the first part is conceptually easy, again, hard to draw. The issue is forcing this chain of blocks to do anything. And one key idea is if you want to fold, let's say, a k by k by k box, if you have a straight path of length k, there aren't very many places you can put that. I mean, I guess there are still k or there's k squared of them. But you have to start and end at the beginning and end of the overall cube, if you have a full length of straight.

So the first step we do is take your-- sorry, this is the first step. It's upside down because we're doing things in reverse order. First thing we do is we take a target cube and we're basically going to have a lot of really long guys will force you to in a row. And you see then these kinds of puzzles. You tend to have straits of length 3 like this. These are three 3's in a row. And we're making a 3 by 3 by 3 cube. So that sort of has to live in, well, it can do weird things. But it's relatively restricted. Maybe you can do something like this. But then, these must be the edges of the cube in that case.

So we have a bunch of those and lots of care, and we end up reducing the cube to a box, meaning we cover all of the cube. We're forced to cover all of the cubic and leave exactly a box behind. Sort of like what we did when we were doing squares into square. We put a bunch of big boxes that were relatively constrained and it just left empty space, which is a box. Here it's actually forced exactly.

Then, once we have a box, so the box will be this wide. And it will go over this way. But it won't have this sort of fine feature stuff. This will just all be filled in as a box. Then this step that's drawn is to carve out this hub and spoke shape. And that's based on a structure like this. We have a huge thing. And then a relatively short thing of length about 8t, And then we go down. And then immediately back up. And then over and down and up and so on. And that roughly carves out this structure.

Now, you have to be careful that this is actually unique. This is why we only want the height of this box to be 8, because then you can't go out of plane very much. And while this is short, it's longer than 8 if t is bigger than 1. So you can't have this spoke go up. So it does have to live in a plane. Sounds right. And this is much smaller than huge, so you can't fold it some other way. So that's pretty much forced.

And then we have to get rid of the other seven planes, which we do with really long guys. Unfortunately, we don't have all of that drawn, so it's especially hard to see. But this is a rough idea of how you start with a box and you do these sort of infrastructure gadgets to just carve out the shape you want. And then you go into the interesting part of the chain. This is all at the beginning of the chain. And as you show, it's forced to do all this. And then we start say here with some zigzag and then an ai gadget and so on. Whew. Easy, right?

So, you do see here, I would say, a very algorithmic approach. I mean, this idea of showing that zigzags are universal, they can make any shape, this is something we do in computational geometry all the time. We prove that these folding structures fold into whatever we want. So we're-- and we hadn't prove this one before, but we were armed with a lot of algorithmic tools to do this. And that let us do this reduction. So lower bounds in our case is really all about designing the right algorithms to give you powerful gadgets like this. That's why this is the class called algorithmic lower bound.

Other questions about snake cubes? Adam.

AUDIENCE: Why 8 instead of 6?

PROFESSOR: I think we want it to be a multiple of 4 so that it's compatible with the grid. 6, yeah, that seems cleaner. That way we can predict if we leave on this 4 by 4 by 4 thing, we know which one we'll come back on. Whereas, if it could be in a half-grid position, that would be messy. Other questions?

You might be able to make that to work, but the advantage of hardness proofs, because it's not a real algorithm. I mean, it's an algorithm that you'd want to run. The point is to show this problem is hard. You can be inefficient in lots of places. As long as it's still polynomial, it doesn't really matter. Except for drawing the pictures, that's one place where it matters.

All right. That's snake cubes. I'm gonna go on to another packing problem. So we've seen rectangles into squares, squares into squares, how about circles into squares? So this is a real life puzzle where you have these-- I think they're water jet cut or laser cut-- metal disks. And you have to fit them into this box. So these are real life puzzles. That's one motivation.

There's a stronger motivation for studying this problem, because it's related to computational origami. There's this method called the tree method of origami design. If you're interested in it, you can take 6.849 or read Robert Lang's book, *Origami Design Secrets*. And basically, it lets you fold things like this. That's at a high level.

And a key component of it is to pack disks into a square. The square is your piece of paper. The disks correspond to sort of the limbs of your creature. And you can't use the same piece of paper to represent this limb and this limb, so the disks have to be disjoint. So it's all about packing those disks into given square. And yeah, so that's at a high level why we care about disk packing.

And people have thought about disk packing a lot. But surprisingly, it hadn't been proofed hard until four years ago. So this proof is also complicated. And it involves a lot of geometry and trigonometry and some calculus and fun stuff. So I'm going to give you a sketch of it. You should be believable that this works, but the details, again, are messy.

For starters, let's say that we're packing disks into an equilateral triangle. This is much easier to work with. I'll go to a square in a second. So if I have this equilateral triangle, what I'm going to do is, first, I'm going to give you a bunch of disks that are forced to pack in this nice, triangular, grid arrangement, leaving these dark holes. And I'm going to give you more disks, and so the more disks have to go into those holes. And these are actually kissing disks, they're touching. So you can only put-- every disk you put from now on will be in one of these holes.

And I'm just going to do this big enough so that I have at least n over three holes. Each hole is going to represent a bin that I'm trying to put a triple of numbers into. And then, here is the construction for, I guess, this would be a three partition gadget. We want to get that-- oh, slight typo there-- ai plus aj plus ak is less than or equal to t.

And so, these are the three disks representing ai, aj, and ak. And there's an extra disk in the center. And what's drawn here is everything is maybe tight, almost as touching. But we're going to shrink these disks a little bit-- or rather, I'm going to shrink-- yeah, I guess I'll shrink-- I'll get this right-- I'm going to shrink this disk a little bit. And I'm going to make these one's a little bit bigger according to the ai values. So this one will be proportional to ak larger than the disk that would perfectly fit here if this one was maximally-sized.

And so, that's going to force this guy to move away. Well, he's a little bit shrunk, so maybe he doesn't have to move away. But this center disk, when it shrunk down, it has a little bit of flexibility. It could move more this way. And if this disk is smaller, it will move more this way. It could move more this way, more this way. And in general, you show that-- so these guys are sort of placing demands by being a little bit bigger. They need to push out. And this one is a little bit smaller. And its size is just a function of t. And so, it's going to move around. And the claim is, it has a valid position if and only if ai plus aj plus ak is less than or equal to t.

I have written down a little bit of the details. So we're going to choose some big number capital N, maybe it's like n to the 100, little n to the 100, something like that.

And we'll shrink the big disk by that amount. I'm going to scale everything so that t equals 1. That will simplify things a little bit. So just shrink everything, all the values, down, divide everything by t, ai's in the t.

And then, I think we're going to shrink the ai disk by minus 1 over n squared plus ai over n. This is a funny notion of shrink. I'm going to first make it smaller by 1 over n squared. And then I'm going to grow it by ai over n. This term, I believe, is to get rid of smaller things in the Taylor expansion. So disks are annoying because they're circular. Because they're quadratic curves. And so, you get-- you know, it's an awkward shape to deal with. And you're worried about how I move this center relative to how I move the other centers and whether they're hitting each other. And whether hitting each other is a quadratic constraint. You have to measure the distance between these two points. It should be greater than or equal to the sum of the radii.

So that's tough to work with. So to simplify things, we take first derivatives and say, well, to the first order, what's happening when I move these disks around? And when you say first derivatives, you get an approximation to the truth. And the second derivative, it tells you a little bit more of the truth. And if you've ever done Taylor series, you know you get something like 1 over n, something times 1 over n plus something times 1 over n squared and so on. And so, if we just subtract off this term, make these disks a little bit smaller than they need to be, that lets you deal with all those terms all at once.

So it lets us focus on the lead term. And to the first order, the claim is, what happens is, well, we made this disk smaller by 1. Remember 1 is t. We made the other guys larger by ai. So that's going to work out exactly when the sum of the three ai's is less than or equal to 1. Then this will have room. And that's true to the first order. And you have to do this ugly business to make it true exactly.

So I think, without going into vector algebra, this is the intuition of what's going on. And that's that proof. Any questions?

The point here is to expose you to lots of different ways to represent numbers and

three partition. And a lot of times it comes up in geometric settings. But so far, we've seen rather different ways to represent it. We'll see graph theory in a moment. Probably, next up is games and puzzles. So let's play some games.

Right, there was one more part of this proof I forgot. What if you're packing into a square? This is particularly important for origami. This paper appeared in an origami conference, so we needed to solve the case of square packing. This is more awkward. The idea, because you can't make a nice, regular grid on a-- you know, square grid packing is not very efficient for disks. But if you have these four really big disks, I should mention here the goal is for the disks to fit-- the centers of the disks to fit inside the square. That turns out to be the right constraint from an origami perspective.

So, we would make these four big disks, they have to be at the corners. Then we make this big disk, it has to be in the center. These are all perfectly touching. Then we add these four disks and it forces a particular arrangement here, which is a little awkward to compute but you can. And yes, we do that four times.

Then, for each of these constructions, this is the shape that we want. We want these kind of equilateral triangle-ish gaps. I say ish because they're not straight sides, they're arc sides. But it is-- the centers form an equilateral triangle perfectly. Then we recurse. So wherever we have a gap like that, which is this thing, then we add the center guy. And then we add these four disks again, just like over here, three times now. Now, of course, we can't force the disks to go here. But they will.

This is a very symmetric construction. So we're going to do this symmetrically in all the things. The other challenge is we left these rather large gaps here. I mean, relative to these tiny disks, which are all fit inside that gap, this is pretty big gap. So we also have to make these gaps unusable by adding a maximal disk and then adding maximal disk. And just recursively adding maximal disks until the gaps that are left are tinier than all the ones that we're going to produce in this recursion. This recursion has depth like log n, so in the end, we get roughly n of these things or n

over 3. If we have extras, we'll just throw in maximum disks and make them unusable. So that's it.

So this is a lot harder because you have to fill in all of these gaps and make sure everything is going to be happy. And no disk can go in the wrong place. But if you sort of do it in the right order, you can be convinced at every stage you're disk has to go where you're putting it. Except at the very end where you have freedom. The freedom you have is where the ai's go, which pocket they belong to. That's the only freedom you have. It's the only freedom we want you to have. OK. So that was disk packing.

Next is Clickomania . How many people have played Clickomania? A few. Good. It's not completely obscure yet. So here I have the real thing, I think this is called Clickomania, Next Generation. And I want to start a new game so I don't have a really bad score. So the idea is you're going to click on a group, a connected group, of one color. That group is destroyed. And then things fall vertically. So column-by-column. Rules clear?

Your goal is to remove all of the blocks, which I probably won't succeed in doing. I haven't played in a long time. But let's-- I would like to show you one other rule, but I may fail to do so. This kind of checkerboarding, not so good. Don't do that when you play. Also very bad. OK.

The one rule, which I won't demonstrate here, is if you clear a column, then these guys just sort of eat the column. They just move closer to each other. I think usually the right moves to the left one step. OK, so that's-- here's a successful execution from the Clickomania website. But I have done it a few times. I just haven't in a long time.

The only rule is, the group that you click on, the connected color chunk has to be of size greater than 1. Otherwise you could just keep clicking on everything. So that's why the checkerboarding is bad, because there you have singleton guys. So you may have some singleton guys. But if you're careful, you sometimes can get rid of them. The question is, when can you? And the answer is, it's NP-complete. So this

is actually my first NP-hardness proof, I think ever. Definitely of a game. I think ever.

So there's a few results here. One is that, if you have a single column, that case we can solve. A single row actually behaves the same as a single column, because even though they're asymmetric in general, for when it's 1, they're the same. And if you happen to know context-free grammars, then here's the answer. It's not obvious that this is the answer.

But this is saying something like, well, it could be you already solved the game. That's the empty string. Or it could be you have a solvable game followed by a solvable game. You could just sort of do them separately. And if you're careful, they won't mess each other up. And then you solve the overall game. Or it could be you have a solvable game and you have the same color, one block of the same color, on either side. That would also be solvable because you could just solve the center thing. Then these two guys come together. And then you click that group of size 2.

Or this could also happen where you solve two things and then three blocks come together. And you could also imagine other rules. But it turns out these are enough rules to capture all solvable puzzles for one row or column. But that's not a hardness proof so we won't talk about it.

We have two NP hardness results. One is for two columns and five colors. And the other one is a hardness proof for five columns and three colors. And these are the proofs. This one is three partition. This one is three set. So we might talk about that in the three set section of the class. But I'm going to focus here on the three partition-- two columns, five colors.

Open problem is whether two colors, you can get any hardness. Question?

AUDIENCE: Is it obvious that S to SS is--

PROFESSOR: No. It's not obvious that S to SS is valid. Because you worry that when you solve one of them, you might eat a group from the right-hand side. But you can prove it's always possible. I don't remember how that proof works, but you can check the paper. Yeah. If you look at the-- if you ever go to the course website and watch this

lecture or click around the slides of this lecture, it has links to all the papers that I'm talking about. So if you ever want to know more details, just go there. Link is right there. You don't have to type in all these names and Google for it.

Cool. So this is my attempt at drawing the hardness proof. This was challenging. And I should say, it's not drawn to scale. So it's not exactly right. But it gives you a flavor of what happens. And I believe all the braces are correct. But it just wouldn't fit on the screen if we drew everything properly to scale. So reduction from three partition.

We have n numbers, ai through an integers. We're going to scale everything up. We're going to scale the ai's and t by a factor of b. b is 4/3n. I don't totally know why, but you know. It's n plus the number of groups. I'm guessing a larger number would also work, but that's definitely enough. OK. So basically, there's two columns, remember. It's always going to stay in two columns. It never increases. There are some clickable blocks, clickable groups, which correspond to the ai's. So for every day ai, we're going to have exactly one block, which is b times ai by 1.

Don't draw the lines when they're in the same connected component. But technically, these are separate squares. There's also some clickable things up here. Those are not too essential. They're basically to spread out these red blocks. There are n over three red blocks here. And there are n over 3 red blocks here. You will never win the game unless you destroy these red blocks.

So for example, you could click all these blue things, make the red together, and then destroy the red. But that's not a good idea. Because these are only, this is a very small space. This is a relatively large gap. I know it doesn't look like it, but this is only n over 3 total. The gap between these two things is b times t, which is way bigger than n over 3. OK, so there's big gaps between these red guys.

So what you're basically going to do is make this column fall down until the first red guy aligns with this one. Then you delete those two. Then you're going to make it fall down farther until this red guy hits this one. And then you can delete those two. Just two at a time. And we're going to bring it down until the third guy hits third guy,

and then the fourth guy hits the fourth guy. Then the reds are gone.

Before I get to how that happens, let me tell you what happens when the reds are gone. When the reds are gone, I mean, if there are any purples left-- turns out there won't be-- but you could click them all. Then you could click all the blue. Then you're done on the right column. So right column was finishable, no problem, once you've gotten rid of the red.

Now the left column is mostly checkerboard. And if you remove the red pixels, you're left with exactly a checkerboard except here, the topmost red guy. It's white on either side. And in general, everything below this point is the same as everything above this point upside down. And so then you can always just click on the center, center, center, center. I guess you have to move your mouse down as that happens, but you will then destroy the black and white part. Total is five colors. We've got red, purple, blue, black and white.

OK, now this is my attempt at showing you how we get the red things to align. But the idea is simple. The initial gap from this guy to this guy. And then, the gap henceforth from this one to this one to this one is always b times t plus some constant. The constants are just annoying. So they're not essential. The idea is, well, I've got to bring this down b times t. So in order to do that, I'm going to click on some purple things whose total size is b times t. In other words, the sum of the ai's that I'm clicking on should be equal to t.

And we happen to know that will be three things, but that's not essential here. We can just use the simple version of three partition, where you're asking for sets that happened to sum to t. Because everything is scaled up so big by these factors of b, I mean, these gaps-- you have to, even to get in the right proximity, you have to choose ai's that sum to exactly t. Yes.

So the constants are annoying because you want to make sure-- so here's an example. Here are, in this particular picture, the height of one of these things is 6. And so, I click on one purple thing of size 4 and one purple thing of size 2. In reality, these numbers would be much bigger, but for the picture, that works. Then things

fall. And we get this picture. And these red things are almost aligned, but they're off by 2, so I click on this little 1 by 2 block. And then they're aligned. Then I can click on the red thing. I also click on this blue thing to sort of advance to the next red thing. And then these guys fall and I get this picture. I could click on the white but it won't matter. That's just, I could do that all at the end later.

Now my goal is to get this red thing to here. Again, the gap is exactly 5. So I click on, let's say, 6. Let's say I click on 2 things of size 3. And so then everything falls. And the reds align. Then I delete it. I also delete the blue thing to advance. And I think that works. Yeah, question.

AUDIENCE: How do you ensure that it's always the leading red block that has to go with the trailing red block on the left?

PROFESSOR: Right. So it could be you take some other block, other red block, and align it with this one. That is mostly a worry from a construction standpoint. If you did that, you might get stuck. But I claim if I did that, it's never a problem to do that. Because this length is, this total extent of these red guys, is relatively small. The whole length here is n. Sorry, it's 3 times n over 3. Whole length is n whereas all these ai's are scaled by a factor of much bigger than n. Well, not a huge amount bigger, but bigger than n.

So even to get them vaguely to the right-- this is what I was saying-- to get this red block or any of these red blocks near this one, because these things are scaled so huge, to get one close is the same as getting all of them close. If you happen to succeed in a way that gets the reds to align in an out of order, that would still give a valid solution to the three partition. That's the point. Because we scaled everything so huge.

To do the other direction, say, if you have a solution of three partition, then there's a solution to this instance, there we do it in order so that we're in control and make sure all the constants add up right. And that's sort of the annoying part of this proof. But this is a relatively easy proof, except for these little additive constants to make sure everything lines up. It would be great if I could just put these reds on top of

each other. But then they're one group and you delete them all at once. So I have to put these things in between. They have to be length 2 so that I can destroy them whenever I want to. And so, all these additive constants kind of get in the way. But it's not too bad.

Other questions about this proof? That's Clickomania. Cool.

Next is Tetris, as advertised in the original, first lecture. So if you haven't played Tetris, here is the-- I hope you've played Tetris. I played a lot as a kid. You have these tetragonal blocks. You can rotate them. And then you can force drop them or you can slowly drop them and do weird things like that, you know, sort of last minute moves. Let's see. I don't think that's-- yeah, the rotation centers are always a bit funny, so you can do cool things. So let's have some fun. I'll just play Tetris for the rest of the class.

So important rule is when you complete a line, then that line disappears. Although in the proof, we will prevent that from happening, it is a rule of the game. That's annoying. Also annoying. OK, well, you get the idea. You're super impressed by my Tetris skills. All right, so cool. I used to play it on GameBoy and then NES. You can play it on the green building. This happened in 2012 finally, after it was dreamed for many years.

I am a Tetris master according to the Harvard Tetris Society. This was a for proving. I didn't have to play a game to win this as you can tell, for proofing NP-completeness and maximization of aligns Tetris's pieces played, or minimization of square height, we masters of the Harvard Tetris Society hereby confer the title of Tetris Master upon Erik D. Demaine on the 16th day of the 12 month in the year 17 Ano Tetri, which is since the invention of Tetris. OK.

This is a proof done early in my career at MIT with two MIT students, David Liben-Nowell and Susan Hohenberger, now professors at other schools. This was a first attempt at a proof. We had many attempts at a proof, all of which were wrong except the last one. Yeah, anyway, you could try to see what's wrong with that proof. But here is a working proof.

This is actually not the first working proof we had. So there is me, David, and Susan. And then we published a much harder version of the proof, with like 500 cases, and then these three guys read our paper and said, hey, I think we can simplify it like this. And then we wrote a joint, journal version. So reduction is from three partition, surprise.

So I should say, what does Tetris mean from a computational complexity standpoint? There are multiple interpretations. Our interpretation is, I give you an initial board. If you've already been playing for awhile, here's what you have. And then I give you the entire sequence of pieces that's going to come. And I want to know, can you survive maximized number of lines.

This is not yet the approximability part of the class, but we get some really easy inapproximability results, so I will mention them. For now, it's just can you survive. The ceiling is like here, a couple rows above this initial picture. So you're almost dead. So it's like every second counts. All right? Your goal is to fill up these chambers, these bins, exactly, with no holes. So if you could do it, at the end I'm going to give you a T and then a whole bunch of straights. And then you can win the game. All right?

So we have n over 3 of these buckets. Each one is roughly t tall. There's some additive constant to deal with some stuff, but there's t notches. Each of these is basically counting one unit of an integer. OK? Yeah. The point is, if you don't open this thing, where are the i's going to go? And you will only be able to open this thing if this is full and this is full and this is full and this is full. So you've got to get these perfect so that at the top, you can open this and then clear everything. OK.

So here are the gadgets from the piece perspective. That was the initial configuration. Of course, there are no ai's yet. So the piece sequence has to encode the ai's in unary. Because if there are only a constant number of different tetromino pieces, so you have to encode-- well, I guess you could try to encode in binary. It's very hard to add when you're working with tetrominoes unless you do it in unary.

So here's the idea. We have an initial set-up like this. It will always have this little thing in the corner. And so to say, hey, a new ai is starting, we're going to give you, you might call this a right-ward pointing L. If you look from the barrel of the gun, then it goes to the right. So that makes a nice flat face here. Because the next piece that's coming is a 2 by 2 square. So what you're going to do is let it drop all the way. And then at the very last second, you slide it to the right. OK?

And in general, we're going to represent ai by this pattern-- square, left L, square, to the power ai. Like we're going to repeat that ai times. So here I repeated it twice and then dot, dot, dot. OK? That is encoding ai in unary-- or I don't know if there's something smaller than unary, where you use three objects to represent one unit, but there you go.

And then at the end of the ai, we're going to sort of finish things off and reset to this original kind of configuration by saying square and straight. OK? So the square nestles in there. We get a straight. And if all goes according to plan, what you've done is filled up basically ai units of this thing. And maybe plus 1. But we know there's only three ai's going into each bucket, so we can deal with a plus additive 3, that's no big deal. And that's good.

So now you have a choice. You have this long sequence of pieces representing ai. In general, the piece sequences is the piece sequence for a1, the piece sequence for a2, piece sequence for a3. The intent is, you put all of these pieces into one column, one of these buckets. The claim is you can't against the divisibility issue. The big issue is, can we divide ai into two parts? Put part of it in one bucket, part of it in another bucket. That would not give us a solution to three partition. But the claim is you can't.

And the proof of this claim is not too hard for this proof. It was much more tedious for the original proof. But basically, you try everything else and show that you're doomed. So every other possible move. So one type of bad move, so when you do this thing of putting in the right-ward facing L, remember, there's only right-ward facing L's, there's only n of them, one per ai. So that's the only sort of nice piece.

We call this priming a bucket. Because now it has a nice flat bottom and so, these pieces fit nicely, especially the squares.

So one issue is, what if you have an unprimed bucket? It still has this stuff over on the left, and you try to just insert into it. Because the first ai comes, you prime one of the buckets. What if you switch midstream and try to move the rest into an unprimed bucket? The claim is, you die. You look at the squares, you say, well, if could go here or here or higher. And they're all bad. This one means you'll never fill this stuff, so you're basically going to run out of area. Something's going to have to go into the sky. If you put this here, you'll never get anything here. Put that there, you'll never get anything here. The squares won't fit anymore and so on. L's can't go pass through that thing.

OK. Similarly, if you try to put a straight away in, something bad happens. I don't quite remember what the bad thing is here, but that looks-- right. Once you cover one, you can't cover the other. Yeah. It's not even obvious you can cover this one, but I think with some clever rotation you can. Tetris is a little weird in that respect. And then, with the left-ward facing L's, you can try all the things. And again, you're toast.

So that's one type of bad thing, where you switch midstream from one of-- you start putting an ai into a bucket, and then you try to put it into a bucket that's currently unprimed. So what's left is, if you stick to one bucket, do you have to play like we want to play? And so, in the sequence of pieces, when you haven't placed anything, well, maybe you try to prime it in a different way and then that opens up something else. Then you'd have to go through that entire analysis in the last side again for however the unprimed bucket might look.

But it turns out, however you do it, you're toast. You've hidden some square which you won't be able to open up. Unless you do it the right way. And once you've done that piece, you check that the next piece has only one place to go. Obviously if it's to the left, then those are not good. And then, the next piece and so on. So you check all these things. And you conclude there's really only one way to play this sequence.

Except for the flexibility of when a new ai stream comes, which bucket do I put it in?

We're assuming here infinite dexterity. So you can slide to the left and right, pick the bucket you want, and then let it fall. Questions? Yeah.

AUDIENCE: So you set up the infrastructure so that if you leave even one space unfilled, then you're going to die before you get the T?

PROFESSOR: Yes. So the idea is that if you leave any square out here, you will be in trouble. And there are probably many ways to do this. And I forget exactly what we do in the paper. I think the easy one would be that, if you put any square up here, you die. So you're right at the limit of the game. If anything doesn't go into a bucket, you're in trouble. And then it's just a volume argument.

So you've got all these pieces. The total volume of them is exactly the sum of the bucket sizes. So if you leave any blank space, that means that the material that was there is going to somehow end up on the top row. And then you're dead. I don't remember if that's exactly what we did, but that certainly seems to work. Adam?

AUDIENCE: If you place something to high, but at the same time finish a line, do you die? And assuming not, could we force our way through the block early?

PROFESSOR: I believe in real Tetris, if you clear a line, that happens before you potentially die. I don't think that's an issue here, because you won't be able to clear anything until the T piece. So I think you use the argument I said up to here. There are no other T's in the construction. So any other piece you try to put in here, I think, will get you above this row. And then you die. So that's essentially unusable space until the T And then, up until that point, you have to exactly fill all the stuff. Yeah.

AUDIENCE: So if we ignore filling the lower stuff in the main gadgets and just stick something up high and just try to fill just the tops of each of them, so that with and L or something we can clear one space of the--

PROFESSOR: Oh, I see. You could try to put an L like that or like this. Yeah, but it if the row clears first-- all right. Now I have to look up the rules, I guess. Yeah.

AUDIENCE: I think it might be OK because then there's no way ever to fill the next part.

PROFESSOR: Let's talk about this offline. I think there's something to check here, which hopefully we did in the paper. But I've by now forgotten. Just for fun, if you succeed, at the end, you get all of these right-ward facing L's. Then you get the T. Those lines clear and then you get all the I's and it's so satisfying, all those Tetrises.

And then, if you're-- so a Tetris is when you get four rows in one move, which can only be done with the straights. So this is, if you survive, then you clear a whole bunch of columns. But if you want to make it really embarrassing, then you put this entire construction over a nice big wall, big empty space, and you put a whole bunch of pieces at the end, like let's say L's all of the same orientation or something. Those are really easy to pack.

So if you want to maximize your score and you don't solve this thing, then you're really going to lose bad. Because if you do solve this thing, you're going to get tons of points by just playing in here forever. So n to the 1 minus epsilon of the game is down here. And if you can't solve this thing at the top, then you're getting a score of basically zero. I mean, you do place a few pieces, but very small compared to the number of things down here, you place like n to the epsilon pieces. Whereas if you do succeed, then you get to place all n pieces.

This is just the initial configuration, or which floating square? This thing? Oh, this one. That is possible to construct, I believe. This is a challenge. It's a whole industry of making cool initial patterns by playing a lot of things. So in fact, there's a paper by Hoogeboom and Kosters that shows that any reasonable Tetris configuration can be constructed from an initially empty board.

And so, what we get is to approximate Tetris, let's say the number of pieces that you play, less than n to the 1 minus epsilon factor is NP-hard. That's what we just showed or sketched. So this is an example of an inapproximability result. This is a relatively easy one. We just need to assume P does not equal NP. And we get that you really can't approximate within a very good factor. It's sort of all or nothing in this game. Later in the class, we'll see much more subtle inapproximability results.

Other questions? Hopefully no more issues.

There are a lot of other open problems about Tetris. You mentioned construction. What if the board is initially empty? Seems like quite a challenging problem, constant number of rows or columns here. We assume that both dimensions are arbitrarily large, it's part of the input. But these seem very annoying. Interesting. If you just have straights, I imagine Tetris is pretty easy.

So how many of the Tetris pieces do you need to get a harness proof or are there certain small combinations that are easy? We need this ability to slide at the last minute. Be nice if you could just always be hitting space bar and always drop your pieces from positive infinity. We don't know whether that problem is NP-complete. That might be the most tractable, but it's proof would have to be a little different. Two-player Tetris, online Tetris where you don't know the pieces in advance, that's something we might get to later in the class. But these are all open. OK.

So next topic is in graph theory. So suppose you have a graph, vertices and edges, you want to draw it in the plane of no two vertices touching, no vertex on an edge. That's a regular notion I'm drawing, but edges are allowed to cross. If you're lucky, graph is planar. Don't need any crossings. There's this notion of 1-planarity, which goes back to the '80s, where every edge can cross at most one other edge. So this is an example of a one-planar drawing. Some edges don't cross anything, but any edge that has a crossing only has one. OK?

So this is NP-complete. On a notion of strong here, there's no numbers. This is from 2007 and a reduction from three partition. So here are two gadgets. This is a double-- well, OK, we'll get to this one in a moment. Here's a gadget which is the complete graph on six vertices, K6. And the idea is that we're going to plug this in instead of a single, bold edge. So wherever there's a bold edge here, what it really means is this gadget. OK? This is meant to be an uncrossable edge, because usually an edge can take up to one crossing.

What we're going to replace this edge with this thing, which has a bunch of crossings in it, no matter how you draw it. K6 is very symmetric, so it's not too hard

to argue about the different drawings. We see a crossing there, a crossing there, and a crossing there. And in general, if you look at any path from this vertex to this vertex, you visit a crossing. There's no way to get-- wait, what about this one? What do I mean?

If I attempt to draw an edge through here, I will hit an edge that has a crossing on it. That's what I want. So in other words, it's not possible to draw an edge across here and still be 1-planer So this effectively becomes an uncrossable edge. This is notation. This is super useful. When you have a complicated construction, if you use the right notation, it's not necessarily so complicated. If we drew this everywhere in these pictures, it would be really hard to see what's going on. So we use this visual idea that this is a symbol for that. And then we can draw really simple pictures, relatively simple pictures.

the thing to the end of the thing over there. I mean, this looks a little bit weird like, OK, how do the ai's choose which side they're on here? Well, they just choose whichever side they're on. If they're in the bottom chunk over here, then they'll just connect to the bottom chunk over here. If they're in the top chunk over here, they'll connect to the corresponding chunk over there.

This is basically to force the ai's to start on the outside of this construction. So there's a lot of, again, there's a lot of cases here to worry about, about other ways you might draw this. Maybe you'd take this entire picture and stuff it into this little triangle. They all end up with lots of crossings on a single edge. If you do that, then you have to go, you have to cross this edge many times, for example.

I think we convinced ourselves that with a single wheel, without this extra thing, the construction, I think, we are not certain about whether the construction works. There's definitely more cases to consider. So at the very least, it simplify things by adding a second wheel. It would also mean you'd have to have multiple edges here between the same two vertices. So maybe that's why they did the double wheel, so that you have these paths of length 2 connecting the same thing instead of the exact same edge.

Anyway, so again, you have to proof. This is essentially only one planar bedding for each of these wheels. And for these things to connect them together. And so, the ai's have the freedom of which bucket they go into. But because each of these edges can only be crossed once, each unit of each ai-- only one unit can occupy each of these cells. So you get three partition. Cool? Yet another way to represent numbers in unary. Question?

AUDIENCE: I don't understand how-- what those numbers are.

PROFESSOR: So OK, so these are the ai's-- A1 is 2, A2 is 3, A3 is 3, A4 is 3, A5 is 4, and A6 is 5. And they are represented by, this thing is a 2, this thing is a 3, this thing is 3, this thing is a 4, and this is a 5. And that should be in the same sequence. I think I missed this 3. Good. OK, that was 1-planarity. Four minutes-- which proofs to cover?

I will wave my hands at this one. This is a chain of blocks with hinges on them. And it has colors on it. Your goal is to form a particular pattern of colors. I find it interesting, this is our hardness proof. It's interesting in that it's-- a lot of the proofs you've seen, it's all about making the ai's being a straight line. Here, that's not a big deal. We've sort of drawn out-- this is, again, a kind of universal construction. You could build any polyomino with it. It's all about the colors.

And so, the idea is that the ai's are represented by how long of a blue segment you have here in the overall chain of pieces. And these are the buckets. And so, you just come in at some point and fill in however big your ai is and then leave. And there's just enough blue to cover all the blue. So you'd better not waste it. But all the other stuff, you have a lot of flexibility. So it's all about just kind of filling these amorphous blobs that have exactly the right area to fill these blue regions.

And there's three times you're allowed to visit. And that gives you three partition. So I want-- it's hard, especially without a physical model, to understand what the rules are at this game. But it gives you some idea of another way to represent three partition.

Next, you may have seen this kind of object. It is called a carpenter's ruler or rule. It's got hinges. It's got rigid bars. And usually, when you're not using it-- I mean, you can measure lengths, that's the ruler part. And when you're not using it, it folds into this nice, compact form. It's great. And everyone used these until the invention of the retractable one. Though they still use these a lot in Europe.

So has unit length in this case. But what if you have a really annoying carpenter's rule and the lengths are not all equal? And you have this nice box which is unit length or, you know, some length. You'd like to fit your ruler into that one-dimensional box. Let's ignore the thickness here. OK? Then this problem is hard. It is weakly NP-hard. You can solve it in pseudo-polynomial time. So this is what-- this is like one of the very few problems we'll see of this type.

And this is the proof. It's one of the simplest proofs from reduction from partition, which is two partition, which is divide your numbers into two groups. And I apologize here. In this book of ours, we use x instead of a. So those are ai's. So you're given some ai's. And the idea is, well, if I would like to partition the ai's into two groups of equal sums, like so, that's the same as saying, well, some of the ai's, or the xi's, are going to go left. Some of them are going to go right. And whether I fold something-- I could either leave it at 180 degrees or I could fold it to 0 degrees, that's sort of my binary choice if I'm going to go into one-dimensional box-- if I fold it, I change direction. If I don't fold it, I stay in the same direction.

But what that really means is, by something running summation thing, I can choose for each ai whether it goes left or right. OK? And then, this is gonna-- if this happens, what it means is my starting point is equal to my ending point. Now, could be in between. I go left and right and who knows what happens. But the point is, if they sum up to the lefts equal the rights, then my starting point will be horizontally aligned with my ending point.

So that's not the problem. The problem is to fit into a box of a given size. So this says nothing about the box. But, if you do this very simple construction, which is add a super-long length and then add half of that at the beginning. And then, at the end,

you add half of the long length and then the super-long length, and your goal is to fit it in a box of equal to the super-long length, that will be possible if and only if these are aligned. If they're not aligned, then if you go halfway left or right and then add the super-long length, if it's not aligned with this one, then you'll be a little bit too big. And you won't fit in the box. OK? So that represents two partition. Cool.

20 seconds, one more proof. So here's another problem. Map folding. You've probably done this before. Easiest way to refold a roadmap is differently. Generally, what we're interested in here is just very simple folds. Fold along one line, then fold along another line and so on. OK? So, this is NP-hard. If I give you a crease pattern, this doesn't look much like a map, but we can talk about that. If I give you a map, a polygon, and some creases on it. And I want to fold all of these creases, this is possible if and only if two partition is solvable.

Notice these lengths-- here we used a's. That's smart of us. So these are the ai's down the distances between these creases, consecutive creases. And again, the idea is, well, I can fold each one or not. And just like the carpenters rule, I can fold or not. And if I am clever and I do it with the right choice, then this endpoint. Will be vertically aligned, have the same y-coordinate as this endpoint.

And if I do that-- and again, we've added half of the super-long length and then the super-long length here. There's also the half of the super-long length and the super-long length there. And now I can fold these two folds. So when I fold one, it's going to go here. And it won't hit anything if and only if I solve two partition. Right?

Again, this is the box that I'm trying to fit everything into and this is the thing that I'm trying to shift up and down to be perfectly aligned with that. If I can perfectly aligned it, then I fold this, miss, then fold this and it's out again. And then I can finish these other creases that I haven't folded yet. OK? So it's all about when do you execute these two folds? And so I'm going to do some of the horizontal folds so that things fit nicely. Then I'll do the two vertical folds. Then I'll do the rest of the horizontal folds. And that will work if and only if two partition is solvable.

Now, this may look like a weird map, but it turns out if you have a regular

rectangular map-- this is not maybe a regular one, but-- and you have some diagonal folds, 45 degree, then that will basically force you to fold something like this. Oops, I did it again. One, two, and then here. So you can put in diagonal faults to force you to fold into this initial structure. And then add folds on top of that. So this is like composing two gadgets. Then you add horizontal and vertical folds just like that picture. And then you'll have a rectangular map with horizontal, vertical, and diagonal creases that you can fold all the creases by simple folds if and only if two partition is solvable.

Open problem, is this strongly NP-hard? We don't know. Is there a pseudo-polynomial algorithm? Obviously we don't know. It's quite tantalizing. But this is basically why maps don't have diagonal folds.

And that's it for today.