PROFESSOR: Today, we are talking about the local behavior of a crease pattern. So you take some crease pattern for some flat folding-- we're thinking about flat foldability. This is a foldability question. I give you a crease pattern like this. I want to know, does it fold flat, like this one does.

And we're studying what happens locally right around a single vertex. So I didn't mention graph terminology. It's probably useful to mention that. These corners-- where all the edges come together-- those are vertices. These triangles-- or in general, some regions divided by the creases-- we call faces.

If we just sort of imagine cutting a little disk around that vertex and seeing how it behaves, we get a nice circular piece of paper with some crease pattern. And we want to understand when those things fold flat and when they don't. Sometimes they do, sometimes they don't. This is kind of like a one dimensional folding problem. So let me write some stuff about that.

So we get what we call single vertex crease patterns. So something like-- this is the example I have made-- just something like that. That should be flat foldable.

So we think of having a disk of paper. It doesn't really matter, but it's going to be easier to reason about and think about a disk of paper. Really, we just mean some small region around that vertex. There's obviously one vertex in the crease pattern. Let's say we have n creases emanating from one vertex.

This pattern is going to be defined by some sequence of angles. In general, theta 1 up to theta n. If there are n creases, there will be n angles between them, say in clockwise order. And let's see.

When we fold this thing flat-- like in this picture-- we take this disk. We fold it along all of those creases. What I'd like to focus on is what happens to the boundary of the paper.

So there's this outer circle. The boundary is a circle. And locally, if you look at one of these creases, it's like folding the circle in half onto itself. And when you make all of these creases, if you're successful, you end up folding the circle onto a portion of the circle.

So if you look at this boundary-- which is a little hard to see. Why don't I trace it? It's going to do something like this. So this is what the 3D thing looks like blown up a little bit-- and I changed the angles a little bit.

But this is what a flat folding would look like. And I'm really just focusing on how the circle maps around this circle. So a flat folding of a disk also folds a circle onto a circle. So we have a flat folding on one of these single vertex crease patterns, we get a folding of a circle onto a circle.

This is nice, because circles are one dimensional. And last class, we talked about folding one dimensional line segments. A circle is just a little bit more complicated. It's different topology. But we can use a lot of the same mindset, at least, as we did for folding line segments onto the line.

In particular, you can see, yeah, this is kind of circular. But you could also imagine unrolling it a little bit and making it straight. Something like this. So that it lies on a straight line, if you just unroll that circle. So you also get a folding of a circle in some sense onto a line by unrolling.

Now, at this point, I want to mention a slight issue here. I mean for this unrolling to work-- if you're taking a circle like this-- in particular, you must make at least one fold for this to be possible. Because right now, if I haven't folded anything, I just have a big circle, I can't unroll circle onto a line. I haven't collapsed it.

Yet here, I've made it small enough that it only occupies a portion of the entire circle. And so I can unroll it. So this is true, as long as I make at least one fold.

OK. Technically, I need to assume another thing, which is that this thing came from a flat piece of paper. I said there were these angles. And what I intended to mean is that the sum of those angles is 360 degrees, as it is in a flat piece of paper.

But I'm not always going to require this assumption. It's what we care about for thinking about a crease pattern of a flat piece of paper and we look at each vertex. This will be true. But when we reason about these single vertex crease patterns, it's really useful to think about, sort of in the middle, this could get smaller.

So for example, you have a pattern like this. Let's say I just take these two creases on the bottom and I fold a crimp. Because I know a cramp is kind of a good thing to do. So now I have what's called a cone of paper.

The sum of the angles-- there's two angles here, this one and this one. Each one looks like 270-- no, not 270, 135? In total, it's 270. So the total angle there is now less than 360. This is what we call a convex cone.

So if you relax this constraint to allow less than or equal to 360, then we get a convex cone. And today, I only want to talk about flat paper and convex cones. We need this for the induction, essentially-- for proving things about the flat case.

In the textbook-- if you look at this chapter on single vertex crease patterns-- we also prove or study the case where the sum of the angles is greater than 360, which you could never make from flat paper. But hey, it's fun to think about. It's a natural generalization. And all the things that we do here, you can generalize to that situation. It's just a little bit harder.

So what does it take to fold a circle onto a line? In this situation of a convex cone, where we make at least one fold, our folding is going to lie along a portion of the circle. We can unroll it a little bit, get something lying on a straight line.

It's kind of like folding a line segment onto a straight line, if it was flat folding of one dimensional pieces of paper. But there's a lot of differences. I mean, they are similar. But a lot of the things we proved last time do not hold in the case of circular paper. So I have a bunch of them here.

So one problem-- you may recall before, if we had a line segment and we put an arbitrary crease pattern, then we could just assign a mountain valley assignment, alternating mountain valley. And that would always fold flat, no collisions. OK.

In this world of having a circular piece of paper, that's no longer true. For example, if you have this kind of crease pattern on a circle or in the disk, it would look like this, those two creases. You can't fold that thing flat. It's not going to work. We'll see exactly what you need to forbid for that to make sense.

Another example is something like this. I mean, if I tried to draw this in the line, what's happening is I have a short segment and then I have a long segment. So the problem is those ends don't meet.

But even when the ends meet-- like in this picture, imagine these are vertically aligned-- if I do this alternating mountain valley pattern-- mountain valley assignment-- I can't join these ends and make a circle, because that would collide with everything. So this thing actually does have a flat folding starting from a circle. And I'm not going to try to un-map it to a circle. It does fold, but not like that. Not with that alternating mountain valley pattern.

There are other annoying things. Let me tell you one more on the proof side. Last time, we had a lemma that said, if your crease pattern is mingling-- which still is a meaningful notion here-- then you have a crimp, or an end fold in that case. Here, of course, we don't have any end folds.

Here, we also do not get this implication. Mingling does not imply the existence of a crimp. And where the proof breaks down is if you have-- so if you remember, our parentheses from last time-- we had this kind of pattern. And we said, well, you keep going and either on the left side, you have a beginning open paren or on the right side you have a closing round parenthesis. In either case, you've got an end fold.

Now, this can actually happen in a circle. And so you have nothing you can do. So you could have this pattern. There would be no crimp. And probably, in that case, you would not be flat foldable, but in particular, this implication fails.

So we have to do more work. But we're going to get the same kind of result that, in linear time, we can tell whether one of these things folds flat. So that's good news.

So last time, telling whether a crease pattern folded flat was trivial. The answer was always yes. And now, it's not so trivial. So let's start with characterizing crease patterns that fold flat. Then we will go to mountain valley patterns that fold flat.

It'd be great if I could just say that, but I mean here, of course, single vertex. So I give you one of these sequence of angles. Let's say it sums to 360 or less. When is it going to be flat foldable? And the answer is very simple. Let me write it down.

This thing is going to be flat foldable if and only if the sum of the odd angles is equal to the sum of the even angles. And, yeah, I'm implicitly assuming and requiring here that n is even. That's something we'll prove. So n is actually even and minus 1 is going be odd.

And if you happen to have started with a flat piece of paper-- for the sum of all these things is 360-- then of course, if you have them evenly divided, this sum will be 180 degrees. But if you started with the convex cone, it'll be smaller. For this reason, a lot of people-- or some people call this the 180 degree property, or the pie property-- if you like to think in radians instead of degrees.

And we can do a little example here. This guy-- this crease pattern-- has a 90 degree angle here. And it has a 90 degree angle down here. So the sum of those two is 180. Those are alternating angles, different parity classes.

This is 45. This is the complement, 135. And those also sum to 180, of course. One does only if the other does. So it's flat foldable. We're golden.

Something like this is bad, because I mean it's an even number. But the odd angles do not equal the sum of the even angles. One's clearly bigger.

And this is a general property. This is also called Kawasaki's theorem or Kawasaki-Justin's theorem. They proved it in 1989. This is pretty much the beginning of origami mathematics. So let's prove it.

There's two things to prove. One is that any flat foldable crease pattern has this property, which is pretty easy. And the other, which is a little bit harder, is that if you have this property, you really are flat foldable. That's the full characterization.

So let's start with the easy direction. If you're flat foldable, you must have that sum. OK. So it's all about thinking what happens at a crease.

So we're thinking about the boundary of this piece of paper. We're thinking about the circle. And what's happening on the circle as you travel, say counterclockwise for some amount of time, that would be first angle. Then you make a fold. It's either a mountain or valley.

At this point, we don't care. Either way, you turn around-- maybe you turn around this way, or you turn around this way. But in terms as your travel along a circle-- so we went theta 1 counterclockwise. Now we're going to go theta 2 clockwise; then we're going to go theta 3 counterclockwise, and so on. We keep alternating back and forth.

It's much easier to think drawn a line. So we go to the right, theta 1. Then we go to the left, theta 2. Then we go to the right, theta 3.

And I don't know about the layering here. I'm just drawing it in terms of horizontal travel. The y-coordinate means nothing.

In order for this thing to be a flat folding-- to be a valid folding of a circle of paper-- these guys have to have the same x-coordinate. So I don't know-- I mean, the y-coordinates, they also have to work out. But we're not thinking about the y-coordinates. We're just thinking about x travel. We've got to get back to where we started if we're folding a circle. That's it.

So how much total travel do we do, in a sign sense? How much do we go right? Well, we go theta 1 to the right. Then we go theta 2 to the left, which is like going negative theta 2 to the right.

And then we go through theta 3 to the right. Then we go negative theta 4 to the right. And you add up this alternating summation that must equal 0.

The other thing that I should say is that n has to be even. This is maybe obvious at this point. But we're supposed to alternate. Every time we hit a crease, we have to change direction.

And so in the end, it's important that after theta 4, we change direction to visit theta 1. If they were aligned-- like if theta 4 went over here and then theta 5 went like that, and there was an odd number of creases, this would be bad.

Even though they're lined up, there's no crease here. You just went straight from theta 5 theta 1. You're supposed to change direction every crease. So also, n is even to alternate directions. Question?

AUDIENCE: Oh, no. You already--

PROFESSOR: All right, good. Clear? This is pretty easy, once you realize the angles correspond to x-coordinates in this linear space.

So why is this enough? In order to show that this property is enough for flat foldability, we need to actually worry about the stacking order of these edges. We need to worry about the y-coordinates. We need to make sure that they can avoid collision with some mountain valley assignment.

The good news is we're free to use whatever mountains and valleys we want. We can stack things however we want. We just need to find some valid state.

And our intuition, from the one dimensional case, was we should try alternating mountains and valleys. That seemed like a good thing. It avoided collisions for an open piece of paper-- for a line segment.

Of course, we know that's not enough. Here-- somewhere-- I drew this example. So it's alternating in the middle, and then that last crease is a problem. But that's all we need to fix. It turns out it's not so hard.

So we have a nice circle. What I'd like to do is cut that circle. So here, I sort of cut it and thought about it as a line segment and then I wanted to rejoin. OK. I'm going to be careful about where I cut it.

OK, I want to cut at an extreme left or extreme right. It doesn't matter. We can see here the cut ended up being in the middle. That's bad. So how do I fix it?

What does extreme mean? How do I tell-- given a circle, it's a little hard to say what extreme means, because it's circular. But if I take a picture like this, I've already drawn it with nice x-coordinates-- I don't know about the y-coordinates yet-- I say, oh that's bad. My cut point ended up being non-extreme.

Well, this is extreme. So cut here instead. So what that means is redraw this picture with this being the cut point.

So maybe, this is the beginning. So we go like that again and now we wrap around. So now we have this segment. And now we have this segment. And lo and behold, it remained extreme.

And this is always true. You can do this sort of cut and rejoin the ends that were messed up before. And you'll preserve the x-coordinates. The x-coordinates aren't changing when you do this kind of transformation.

So there's a clear leftmost and there's a clear rightmost. You pick either one, you cut there instead. And now we have this nice property that it's easy to join the ends, because it's as far left as you can go. There can't be anything that penetrates, because that would be farther left.

Let me write down some words corresponding to that. So once we cut at this extreme crease, you can fold this 1D segment just like we used to be able to. So we can fold it flat using the accordion fold, alternating mountain and valley. We know that works.

And then all we need to show is that the two ends can join. Those ends are the two copies of the cut crease that we did in this first step.

We need two things. One is that they are aligned and that's where we need this condition-- that the sum of the odd angles equals the sum of even angles, which is the same as saying the alternating sum is equal to zero. That tells us that whatever we do, the x-coordinates are lined up at the end. So they are aligned by the assumption that we made here. And you can join without crossing.

So this is a statement about y-coordinates. And we know it's OK, because it's at the left extreme. So by this choice of the proper cut, there could be other things that come right to the boundary here, but that's considered OK. That's not crossing, that just touching. Nothing can go farther left because it was the left extreme.

That's the end of the proof. Any questions about that? That's Kawasaki's theorem-- Kawasaki-Justin.

I would mention just for fun-- this is sort of classic-- if you allow one of these-- a non-convex cone, where you have more than 360 degrees of material-- this statement is not true. And there's one other situation which can happen, which is that the alternating sum of angles is not zero, but it's plus or minus 360 degrees, which is fun. And you can see examples of that in the book.

All right. So obviously, if you want to tell whether a crease pattern-- or a single vertex is flat foldable, you just compute this thing in linear time and you know yes or no. So crease patterns are pretty easy-- just have one extra condition.

What about mountain valley assignments? Last time, we looked at mountain valley assignments and we showed that crimps and end folds were always enough to fold any mountain valley assignment that's out there. That will continue to be true here, except now we don't have end folds. Now, it's crimps all the way. What does it mean?

So that is the flat foldable mountain valley patterns. We're going to prove that crimps are enough, but before we get there, I need a bunch of sort of warm up facts about valid mountain valley patterns.

The first thing is just counting mountains and valleys. So even if you play with a simple example like this one, you see it as three mountains and one valley or three valleys and one mountain. In fact, if you have a degree 4 vertex, with four creases emanating from that point, those are your only choices.

You could try, for example, making it all valleys, and it-- I mean, you can't do it. It's hard to demonstrate. You could try making it two mountains and two valleys. Maybe I try to do mountain mountain here and then somehow this is going to be valley valley there. It's no good.

But what's happening here is that the number of mountains and the number valleys must differ by exactly two. It doesn't matter who's bigger. So the difference could be plus 2 or minus 2, because you can always flip over and negate that difference, swapping mountains for valleys. But they always have to differ by exactly two. Why is that?

Well, this is-- yeah, we're again going to think of a picture like this. We have horizontal travel according to the theta i's. We're also going to think about the vertical picture. If this thing is flat foldable-- so I'm assuming, it's flat foldable-- then this must happen.

This is called Maekawa's theorem, by the way. It was proved by Jun Maekawa in 1986-- a little earlier. Also proved by Jacques Justin around the same time-- back before we were good at internet and communication and whatnot. And there were little pockets of mathematical origamists and they found each other basically in '89, when there was the-- '89 was the first origami science math and education conference-- started bringing these people together. The last one was this summer.

Right. So why is this true? I don't know. A flat folding looks like something.

It has no crossings. It has-- the horizontal travel, we understand. It ends up back where it started. There's also some notion of layers and vertical travel, which is a little tricky to think about.

But the one thing that's easy to think about-- I mean, this forms essentially a polygon. It's like a really squashed polygon. And these vertical segments are actually really just points. OK, imagine. That's just where you turn around.

So it's like a polygon, just stretched or squashed down onto a line. And we know some things about polygons. OK, as an end vertex polygon-- you know that the sum of the interior angles is-- whatever it is. I always get confused with that formula. So I don't like to think about it.

But one thing we-- an easier way to think about that same statement is just think about how much turning the polygon does. So you start here, you turn right 180 degrees, you turn left 180 degrees, you turn left 180, right 180, right 180, right 180. How much is it in total? Oh gosh, do I have to add? No.

It's so simple. It's 360. Right? Just, if you think about if you're going around in a circle, any polygon-- not just flat-- but anything in two dimensions, the total amount of turning you do is 360. Or if you count backwards, it's negative 360.

So this is the key-- something you should all know about polygons. If you look at sum of the turn angles-- how much turning you do at each vertex-- that will always be plus or minus 360 in any polygon. This is equivalent to the sum of the interior angles being, whatever, pi times n minus 2, which I don't remember. But this is much easier to remember.

And it's useful, because we can map mountains and valleys to left and right turns. As I said, every time this is a valley, it was a left turn by 180. Every time it was a mountain, it was a right turn by 180. So valleys, mountains, right turns-- so I'm going to think of that as negative 180.

Now, of course technically, it could be the other way around. It could be this is negative, this is positive. But I already have a plus and minus here. So it's symmetric.

So if we sum up the turn angles-- which we know is supposed to be plus or minus 360-- the sum of the turn angles is going to be-- we're going to have 180 for each valley, so 180 times the number of valleys. And we're going to have negative 180 for each mountain. And so the 180's factor out. And this thing is supposed to equal 360-- plus or minus 360.

How could that be? Well, the number of valleys minus number of mountains should be either 2-- for plus 360-- or negative 2-- for minus 360. So that proves the theorem.

And that's why, in a degree 4 vertex, one of them has to be 3, the other one has to be 1. There's no other option, because the total number is 4. It's the only way to get the difference to be plus or minus 2. So that's kind of nice.

But it's not enough, unfortunately. If I gave you a mountain valley pattern that satisfies this condition, it still might not be flat foldable. That's maybe no surprise. It's not like we had such a simple test for the one dimensional case, which should be easier.

We had to give an algorithm. We said repeatedly crimp and end fold. If you get stuck, the answer is no. If you don't get stuck, you folded it great.

I love this eraser-- increases entropy. Let me tell you one case that is easy to think about. Because it's nice. It reduces directly to the one dimensional case. And that's what I call the generic case.

Generic is this very convenient term we use in mathematics. It's actually really tricky to define. So I'd like to not define it, if I can. But maybe I should.

The simple version, which is not quite enough, is to say that all of the angles in the crease pattern are different. OK, here, for example, two of them are the same. There's two 90 degree angles. This is not generic. It's just easier to draw.

But in general, I mean, you imagine, you just draw some random thing. None of those lengths are going to be the same. Yeah, the alternating sum is zero. But that's it.

And that's what I-- the generic case is that is the only thing that holds true about these angles. You can take the alternating sum. That equals zero. If you take some alternating sum of some subset of the angles, that will not equal zero. If you take a random example, this is going to be true.

OK, just bear with me. Suppose never any two angles are the same. Then look at the globally smallest crease, so globally smallest theta. Say it's theta i. So the picture is theta i, theta i minus 1, theta i plus 1.

And what do we know if it's globally smallest and none of the values are equal? Then we know that these two are bigger. No big surprise.

Think about what happens if this is small, these are bigger, and you try to make these both valleys or both mountains, that's, of course, bad. This is one of the situations we had in the one dimensional case. I mean, really, this is a circle, but I can think just locally about what's happening here.

So in this situation, one of these must be a mountain and the other must be a valley. I don't know which order they are, but I know they're different. And then I know I can apply one of these crimps.

Incidentally, the word crimp, Jason Koo was asking about, like, why do you call it crimp and not pleat? Probably, I should call it pleat. But it's crimp in my mind. It's been crimp in my mind since 1998 when we wrote the 1D paper. It was before I knew the word pleat, so that's my excuse.

So pleat or crimp, take your pick. Crimp is usually many pleats together. That's crimping. Cool.

So crimps, we kind of like. Why do we like crimps? We proved, in the one dimensional case last time, that if you make a crimp and your original thing was flat foldable, the new thing will be flat foldable. Good news is, that is still true.

I told you, all these things were no longer true in the circular case. It's still true in the circular case. Should I tell you--

Let me just remind you-- the same proof works-- remind you what the proof looked like. We had a crimp, and we're hoping to fold that first. We know at some point, these creases get folded. But there may be other junk in here, or maybe other junk in here. We'd like to get rid of that junk.

And so what we did is move this junk up to here, which was always safe. We moved this junk down to here. And that was still a folded state-- a flat folded state. OK, remember all this stuff.

Therefore, we could have folded the crimp first. And then we have a flat folding in this situation, where this paper is effectively glued together, which means, it was safe to fold this crimp and glue these together and never touch them again. So that is still true in the circular case. That proof didn't really assume anything about the paper. Would even work for like tree shaped paper or something weird.

So if we can find a crimp, which we can in the generic case, then you just make a crimp, make a crimp, repeat. If you ever get stuck, you know that the original thing was not flat foldable, because the thing you have is not flat foldable. Why? If there's no cramp in the generic case, that means you have two valleys around the smallest angle, and that's clearly bad.

So-- I'm not writing a lot of details here. Is that clear? If I have an angle that's surrounded by strictly larger angles, I know there must be one mountain and one valley. Then I can safely make a crimp and repeat.

The trouble comes in when these angles are equal. We never had to think about angles being equal in the one dimensional case. It didn't matter much. We just said, oh, you know, it's safe to make a crimp, as long as this was greater than or equal to this and this was greater than or equal to this. That is still true; it's safe to make a crimp.

But how do we know that there is a crimp? How do we know that there's a mountain followed by valley? We don't. I mean, if they're equal, this would be all right. That's two valleys is valid if these three angles are equal.

But I claim still somewhere there's got to be a crimp. There's no longer a notion of the globally smallest angle, because some of them are equal. We've got to find that crimp, but it's out there. So we just need to prove that it exists.

So to do that, we're going to generalize this theorem-- Meakawa-Justin-- that number of mountains minus number of valleys is plus or minus 2. That's true, and it's a statement about the sum of all the angles-- all of the creases. What I'd like is something a little bit more localized. And in particular, we're going to think about when I have a whole bunch of angles that are equal, how many mountains and valleys can there be in there. So it's what I call local counts.

So while everything I said so far is pretty classic, this result was proved in 2001 by Tom Hall, so much more recent, over a decade later. So let's think about k equal angles. And I want that to be a maximal sequence of equal angles, so the ones right after and right before are different. And furthermore, I'm going to assume that they're bigger-- strictly larger angles.

OK, so something like that. A bunch of equal angles-- k of them-- bigger angles on either side. This almost always exists. I should mention-- so I'm going to say something about this situation.

But this situation should exist, because I take the smallest angle out there and then I see how many friends it has that are all equal. If it's the smallest angle-- or it's one of the smallest angles-- then the ones surrounding it are going to be bigger-- unless all the angles are equal. So there's one case, which we'll worry about later. It's very easy. All the angles are equal.

In fact, I could tell you how to worry about that. If all the angles are equal, then everything is crimpable as long as we can find a switch between mountain and valley-- somewhere. And because the number of mountains and valleys is plus or minus 2, there's got to be-- so they're almost in equal balance-- there's got to be a valley. It can't be all valleys, can't be all mountains.

So somewhere, there's a transition between mountain and valley. So you get a crimp out of that. But otherwise, if they're not all equal-- let's think about k equal angles surrounded by two larger guys. Forgot how technical this is.

All right. Then I want to look at the number of mountains and the number valleys among these k plus one creases, so within that range. It's going to be 0, if k is odd. And it's going to be plus or minus 1, if k is even.

It's not plus or minus 2. Intuitively, plus or minus 2 is when you go all the way around. Here, we're just looking at a segment-- a portion of the entire circle. Remember, this is actually a circle somewhere.

And this corresponds to, in the odd case, you're going to be going in the same direction afterwards. In the even case, you're going to have turned around but not a full circle. So it's going to be plus or minus 1.

There's a lot of ways to prove this. One way is to think about convex cones. So if k is even, and you fold this thing-- let's see, one, two, three, four, five, six-- equal angles. And then there's the two longer guys. They may not be the same length, but that's what you get in the even case.

So what I'd like to do is just, in order to just reduce to things that we've already thought about-- I mean, you could talk about turn angles again, or you could just say, hey, I know Maekawa's theorem. So as long as I could make this into a circle, which I can do by adding that stuff, now I have a nice circular-- it's a flat folded state.

There's no crossings here, presumably because this did something. I don't actually know that it's zigzagging. It might do other stuff. But it's not going to cross what I just added.

So the number of mountains minus number of valleys, in total here, must be plus or minus 2. I only added one crease-- this one. So that's going to make it either plus or minus 1 or plus or minus 3. If you think about it a little bit, it has to be plus or minus 1. And hopefully, that's what I wrote here, yes.

Because I mean, really, we're thinking about turn angles again, I'm afraid. Can't totally reduce it. We just turned around here. And we were about to finish the circle, which would make it plus or minus 2. Before we finish the circle, it's plus or minus 1.

k is odd. Again, this might go like that, whatever. But, because it's an odd number, these guys are going in the same direction. And so the total turn angle here must be 0.

And one way to see that is to extend it into a full polygon. You know that it's plus or minus 2 for this whole thing. And I added two guys in the same direction. If I remove them, it's going to go down to plus or minus 0.

That was a bit hand wavy. But do you buy that?

AUDIENCE: [INAUDIBLE] plus or minus 0?

PROFESSOR: Plus or minus 0 is 0, yeah. You went from 2 down to zero, because you remove two identical guys. So let me tie this all together, because there's so many little cases.

So I want to know what mountain valley patterns are flat foldable, single vertex, m/v patterns. I claim there is a crimp. If it's flat foldable, there has to be a crimp that you can find to do.

Remember, a crimp was where this length was less than or equal to this length and less than or equal to this length. And one of these was m. One of them was v. That was the definition of a crimp.

I claim such a thing always exists. The proof is two parts. If all the angles are equal, we know that these conditions are always satisfied, because everything is equal.

So we just need to find an m followed by v or a v followed by an m. And by Maekawa's theorem, we know it's not all m's or all v's. So somewhere, there has to be a transition from m to v's. So just use that lemma theorem or the count property.

Otherwise, if they're not all equal, then I can apply this local counts result, take the smallest angle-- take one instance of the smallest angle. There could be many of them. Find all of its neighboring friends that are equal.

Then I know that the next angles before and after have to be strictly bigger. Then I know that the number of mountains and valleys in here-- if I look at the difference and it's 0, that means the number of mountains equals the number of valleys. Therefore, there's at least one of each. And therefore, there's a transition from mountain to valley in there.

Because the transition from mountain to valley-- like right here, that's a valid crimp. This guy's bigger; this guy's equal. This would also be a fine transition, because everybody's equal. This would also be a fine transition. Any transition from mountain to valley in here, I've got a crimp.

The even case is a little-- I got to think a little bit more. k is even. k plus 1, which is the number of creases here, is odd. And the difference between mountains and valleys among those k plus one creases is plus or minus 1.

Think about it for second. That means there's, again, at least one mountain and at least one valley. Because k, to be even, has to be at least 2. We've got to have at least three things. So there's going to be at least one of one and two of the other, in fact.

And I just need one mountain and one valley somewhere in there. And then I know there has to be a transition from mountain to valley. Otherwise, use local counts.

And in either case, we get either a mountain to valley transition or a valley to mountain transition. And I get it in such a way that these inequalities on the lengths hold. So in other words, that it's a crimp.

Therefore, any flat foldable, single vertex, mountain valley pattern has a crimp. Make it. And as I was arguing here, just like last time, that crimp will still-- after you do the crimp, you'll still be flat foldable, so repeat. Cook until done is how I like to put it. Question.

AUDIENCE: What about the trivial case of just folding a circle in half?

PROFESSOR: The trivial case of folding a circle in half. All right. That's a good question. It's kind of a technicality.

One version would be to say, that's zero vertices. That doesn't count. And we're thinking about one vertex.

And as soon as you think about it as one vertex, which is fine, then you actually have two creases there. So it's two creases. And hopefully, this works.

Oh, I see what you're saying. It's not a crimp. That's a good point.

AUDIENCE: But then it's flat foldable.

PROFESSOR: It is flat foldable, and yet, there's no crimp. Damn it. All right. Sorry?

AUDIENCE: But it follows Maekawa's, right?

PROFESSOR: Yeah. So I made a slight mistake here. I said if all the angles are equal, then Maekawa says there's at least one of each. That's not true when n equals 2. Then there's two of one and zero of the other.

As long as you have more than two creases, then this is true. That's enough. So you have to handle this as a special case.

AUDIENCE: At the end of your algorithm, you're always left with two of the same--

PROFESSOR: That's true. This is an important special case to remember, because this will be the final picture. After you do a sequence of crimps, this is the good case. If your thing is flat foldable, you will always end up with that.

AUDIENCE: That's a case where everything is the same, smallest.

PROFESSOR: It's another example where all the angles are equal. So that's why this is the situation. It's either Maekawa and you get a cramp here, because n is greater than 2.

And here, I should say, if n is greater than 2, then this is true. Of course the n equals 2 case, it looks like that. And it had better be both valleys or both mountains. Otherwise, it's not going to be flat foldable.

I forgot about that. I should add it in the notes. But it's in the textbook, right.

AUDIENCE: At the end of the algorithm, it's not going to be a circle. It's going to be a cone.

PROFESSOR: That's true. It won't actually look like this. At the end of the algorithm, it's going to be a cone. The two angles will be equal, because of Kawasaki's theorem. And they should still be both mountains or both valleys.

But it's going to be-- we can see it right here. So I apply my crimp, and now I have a cone, and there's two angles. There's one here, which has been fused together. And there's one here, which is the original.

They're equal. And it's two mountains or two valleys, if I turned it upside down. And then it finishes. So that's the algorithm in action-- two steps.

First you find this crimp here. This is actually the globally smallest angle. It's surrounded by bigger angles. So it's really easy.

Then I have a cone with two equal angles, which is what has to happen at the end. And then I'm done. Good?

AUDIENCE: Is that the only way to do it?

PROFESSOR: This is what always happens. Oh, is this the only way to flat fold these things? No. There are other-- yes, one of the two.

I think it depends what you're counting. If you're counting mountain valley assignments-- I mean, if you just want to know, can every mountain valley assignment be folded by crimping. Then the answer is yes. That's what we proved.

But if you want to get every possible folded state, crimps are not going to be enough. The reason they're not enough is because we're using this thing. So, hey, here's some hypothetical folded state. We can rip it out and make the crimp has been done. But then there's a folded state you're not able to reach by crimping.

We need to work that into an actual example, where crimping forbids you from reaching some folded state. But I'm pretty sure one exists because of this.

AUDIENCE: Could that be more like spirals?

PROFESSOR: Yeah, maybe some kind of spiraling thing.

AUDIENCE: Well, layering takes into account--

PROFESSOR: Right. What we're missing here is getting all the possible layer stacking orders. So we're just trying to match a mountain valley assignment. We're not going to match a target layer ordering, because we simplified the layer orders in order to make crimps possible. So crimps won't get you some of the folded states.

But one thing you can play with over here-- that has been played with by Tom Hall-- is if I give you a crease pattern, how many flat foldable mountain valley assignments does it have? OK, we have this nice linear time algorithm to tell you whether a particular one is doable. How many are there?

And there is-- if you work through all of the things I've shown you in a little bit more detail, you can recover-- this was finding a crimp. But you can actually look at what are all the crimps that are possible and actually count how many different ways there are to fold. How many crimps you can do. And then given those crimps, how many crimps you can go after that.

And in the same linear time kind of algorithm, you can figure out how many different mountain valley patterns are flat foldable. That takes a little bit more care. I will tell you the extremes.

How small could the number of flat foldable mountain valley assignments be for a given crease pattern? Well, do you have any guesses when? When do I have the least choice of where to make crimps? Yeah.

AUDIENCE: We have three angles.

PROFESSOR: For general n. Sorry. But, yeah, three angles or, I guess, two angles. We really can't do any-- I don't have any choice.

But for general n, how should I disperse the theta i's in order to make there be very few possible crimps?

AUDIENCE: When there are only two transitions.

PROFESSOR: When there are only two transitions between mountains and valleys. All the mountains are together. All the valleys are together. That's right, if I was asking a question about mountains and valleys. But I was asking a question about theta i's.

So I want to count how many mountain valley patterns there are-- mountain valley assignments that are flat foldable for a given crease pattern.

AUDIENCE: So if you have a bunch of angles, consecutive theta i's are increasing?

PROFESSOR: All the angles are consecutive increasing. Yeah, that pretty much works. In fact, what I need is the generic case, where all angles are different and they never become equal by folding, which is what generic means.

So then there was only really one crimp I could do, which is the globally smallest angle. So actually, I still have two choices. I could do mountain then valley or valley then mountain. But I only have two choices for that crimp. Then I will have two choices for the next crimp, two choices for the next crimp. In general, I get 2 to the n possible mountain valley assignments for the generic case.

OK, what about, what's the opposite, where I get as much choice as possible?

AUDIENCE: Isn't 2 to the n every mountain valley assignment?

PROFESSOR: Yes, 2 to the n-- no-- 2 to the n over 2. Thank you. Right.

There are 2 to the n conceivable mountain valley assignments. It can't be that big. Every time I do a crimp, I eat two creases, not one. That's what I forgot. Good.

It's still pretty big though. It's like square root of all the possible things you could imagine are indeed feasible. What, from 2 to the n to 2 to the n over 2. When would I get the most choice in crimping?

AUDIENCE: All angles are equal.

PROFESSOR: All angles are equal. That's the opposite extreme. And in this case-- a little messier.

When all the angles are equal, the only property you need-- and you can see that from the proof-- the only property you need is that the number mountains minus the number valleys is plus or minus 2, and n is even. As long as you have that, we showed here, you're going to have-- on alternation from mountain valley-- that's a valid crimp, because all the angles are equal. And you keep going.

So it's just, how do you disperse that many-- how do I disperse n over 2 minus 1 mountains among n different positions. That's what this represents. And positions, how do I pick n over 2 minus 1 of them to be mountains. Or it could be n over 2 minus 1 of them are valleys. And that's this factor of 2.

So it could have more mountains or more valleys. And then you somehow place those among those. And if you don't know this notation, that's just what it means.

If you want to know using other notation, it's like this, n over 2 minus 1 factorial, over 2 plus 1 factorial. If you don't know factorials, I'll tell you about them later. Cool.

So that's kind of the end of the single vertex situation. Yeah, I think I'll mention one interesting open question here, which I would love to explore maybe in our problem session-- which looks like it'll be Mondays at 5:00, I think. This is one vertex, and we went through all this work to solve a single vertex situation. And it's interesting, as we'll see, to think about locally how each vertex behaves.

But you would think, if you have a crease pattern with two vertices, it shouldn't be that much harder. So here we have linear time. How quickly can you tell whether a two vertex crease pattern is flat foldable?

As far as I know, no one has looked at that problem. Surely, we can do it in quadratic time but maybe even linear time, I think. It can't be that hard, I think.

In general, if I have a small number of vertices-- say k vertices-- much smaller than n creases, is there-- for the algorithms people-- is there fixed parameter tractable algorithm in k? Or can I get something-- even getting something like n to some function of k, would be progress. I think this should be doable. But ideally, we get a running time that's exponential in k and linear in n. That would be my hope.

Why do I say it has to be exponential? Because in general, if I give you a crease pattern with n vertices-- lots of vertices-- this problem is NP-complete, which is-- there's not going to be a polynomial time algorithm for it. Nothing good.

We will prove that next Wednesday. So I'll hold off on that a little bit. But for two vertices, how hard could it be? All right.

I want to talk about one related topic. And then we will go to origami design and do a little bit on the tree method. But before we get there, I want to talk about local foldability, which is a cool topic. People tend to forget about it.

I really like it. I think it would make a cool project also. It's a nice algorithm. It goes back to Bern and Hayes, 1996. So it's also right at the beginning of origami mathematics.

And it's this idea, all right, I give you a crease pattern now arbitrarily many vertices. And if I ask you, does it fold flat? That's NP-complete-- intractable-- same paper. But what if I ask you just to give me a mountain valley assignment that might fold flat? Well, that's also NP-complete.

But if you actually want it to fold flat, this is really the same problem. But if I ask you, give me a mountain valley assignment, so that if I checked every vertex according to this algorithm, at least every vertex folds flat. That would seem nice.

Definitely, I have to find a mountain valley assignment that satisfies these conditions, that as I do successive crimps, every vertex-- if I cut out the vertex separately, it would fold flat. This is the notion of local foldability. And there's a linear time algorithm to give you a mountain valley assignment that ought to work, in that at each vertex it works. It still may not work globally for some other reason. But it's pretty good.

I think this would be, actually, pretty practical. I think in a lot of real world origami settings-- when you're doing flat folding anyway-- locally foldable is going to be enough to be globally foldable. That's a guess. I don't know if it's true. Consistent mountain valley assignment, if there is one, so that each vertex locally folds flat.

Let me try to concoct a small example that's relevant here-- got to think. Yes, I think that works. All right, here we go. I think this is in the textbook. I remember drawing it in the past three years ago or whatever.

So here's a crease pattern on a square or whatever.

AUDIENCE: The top one should have--

PROFESSOR: Yes, it should have this. Thank you. So this satisfies Kawasaki's theorem-- that was the hard part-- because these angles sum to 180. And it's symmetric all around.

OK, these angles are the smallest. They are 60 degrees. This is an equilateral triangle. So we have this-- not quite the generic case, but we have the smallest angle is surrounded by two larger angles. Therefore, one of these is a mountain; the other is a valley.

That means that these two creases have to have different assignments-- I'm going to write a not equal sign. One of these is a mountain, and the other is a valley. They can't be the same. Also, these two cannot be the same. Also, these two cannot be the same.

That's not possible. Because I've got three, it has to alternate. You can alternate three times. You can only alternative at even number of times.

Two of these are mountains. One is valley. And then you've got a problem. OK, so this thing is not flat foldable.

And this algorithm will tell you that. Because it will say, hey, I can't even find a mountain valley assignment that could possibly fold each vertex flat. So this is a nice algorithm. At least, it will detect annoying situations like that.

So in order to solve this, we're going to use-- and I'm just going to sketch how this algorithm works-- we're going to use this characterization and this idea that we can really find all possible mountain valley assignments just by trying all the possible crimp sequences. Now, there are exponentially many. So it's not like I'm actually going to try them all. But I need to explore that space of candidate crimps and see what happens.

So the idea is kind of crazy. The beginning of the algorithm is fold each vertex flat-- somehow. I don't care how. Just pick a crimp, do it. Pick a crimp, do it-- separately for each vertex. They're not going to be compatible.

And don't look at the mountain valley assignment you get. But look at the crimping sequence you get. So here, let's do a little example-- a non-trivial example.

So here, we have the peace sign, and these two angles are equal. They're smallest. I could crimp this angle first, or I could crimp this angle first. I have a choice. So I'll draw both of them.

If I crimp this angle first, I know these guys are paired up in the sense that they must be not equal-- one's mountain and one's valley. After I do that crimp-- just like in the real example I had-- this angle will equal that angle. And these two guys must have equal assignments. They must be both mountain or both valley.

In this situation, these two guys are not equal, if I crimp this pair first. And these two guys must be equal-- equal now in the sense of being mountain or valley. OK, so there are still exponentially many possibilities in how to do this.

But just pick one-- pick one of these ways of pairing up. You're going to pair up each of the n creases into n over 2 pairs. And they're going to have some not equal and equal signs.

So now, if you imagine the general picture, like here, for example, I might get-- in this pattern, I would be forced to get not equals and these guys paired up. And in general, I want to look at these kinds of cycles. If I come into a vertex-- here's a vertex-- it's paired up with somebody.

So if I come in here, I can go out somewhere. And I come to this vertex, it's paired with somebody. So I'm going to just-- I can keep wandering around. And in general, there will be a bunch of these paths that you can follow.

What could the paths do? They either close up on themselves-- maybe things are paired up in such a way that you make a return trip. Or it could be some other path.

Let me draw a dotted path. It could come in here, and maybe it gets paired up with this guy. Maybe it goes off to infinity. It reaches the boundary of the paper. Those are the two possibilities.

You get paths, which go off to the edge-- off to infinity on either end. Or you could get cycles. The cycles are the problem. Because whenever I have a cycle, I have a parity constraint.

For example, when they're all not equals, the length of the cycle must be even. If there were-- what's the general statement? It's like the parity of the cycle, which is whether it's even or odd, should be equal to the parity of the number of not equal signs. Something like that. I've got a sheet here.

Great, I just said "parity problems." It's something like that. It's either the parity of the cycle should equal the parity of the number of equals or the parity of the number of not equals. I think, number of equals. Anyway, one of the two.

And you can just check. I mean, you're forced. If I say, OK, let's make this mountain, then this is either equals or not equals. It'll tell me whether this is mountain or valley.

Just walk around the cycle. Either you get a contradiction or you don't. If you get a contradiction, we have a problem. How could we possibly fix the problem?

Well, when we made-- you look at each of these cramps. I mean, in fact, you could look at each of these vertices separately. But you think about one of these crimps and say, well, could I have done it another way?

Sometimes, there are crimps that have other equal choices. Maybe, there are a bunch of equal angles. And I could have done a different pairing.

What happens when I try a different pairing? Well, instead of this being in one cycle and, let's say, this being in another cycle, if I pair these guys up instead, I'll end up merging those two cycles into one bigger thing. It could be a path or a cycle.

And the algorithm says, just keep doing those cycle merges. And if you get stuck, your thing is not locally foldable. That's the hard part to prove. Otherwise, you will find one of these patterns that you can actually resolve mountains and valleys all the way through.

So let's say, so start with some folding-- say local folding, whatever. I'm going to say with some pairing of creases at vertices. And merge two paths or cycles whenever possible.

And when I say merge, I mean whenever you have-- what are the possible things you could possibly merge-- when you have at some point during the algorithm a bunch of equal angles, you have a choice which of these you crimp. Obviously, the mountain valley assignment is not fixed. You can crimp any of them.

You picked one of them, and it's sort of merging whatever this thing is attached to, to whatever this thing is attached to. If there's something else that's disconnected from that thing, I want you to instead merge two of them that combines two different connected components-- two paths or cycles. Merging means I decrease the total number of paths or cycles. I combine two into one.

Whenever that's possible, do it. You can prove that if you have parity problem in the merge thing, you had to have had a parity problem originally. And merging can only fix parity problems. That's the claim, and that's what I will not prove.

Once you know that-- and it doesn't matter in what order or how you choose to merge-- you just merge as much as possible. And either the resulting thing is OK or not. And accordingly, you will tell whether this thing is locally foldable.

Sorry, I want to move on to other things. But I think this would be a fun thing to actually implement. It's an easy algorithm. And it's, I think, a pretty good test it for problems like this that prevent flat foldability.

So let's move on from foldability to origami design. So a bit of a big transition. And we're going to talk about origami design a lot more next class also but just start it off today.

And the particular algorithm for origami design I want to talk about is called the tree method. This is probably the oldest algorithm for origami design, in that people have been thinking about it and developing it for many years through this period called the Bug Wars, when people were trying to design more and more complicated insects.

It's like, well, I can make an insect with six legs. Oh yeah, well I can make a spider with eight legs. Oh yeah, well I can make an insect-- a beetle that has wings and horns and there's thorns on the horns-- you know, all these crazy things.

During that time, there were a lot of people thinking about how do I make more and more complicated-- especially, more limbs in my creatures and very precise arrangements of those limbs, let's say. And that is what the tree method deals with and was really formalized by Robert Lang, who published a paper in '96, describing it as a sort of complete algorithm.

And it's still not known for sure that that algorithm always works. But that's what we've been working on-- me and Marty and Rob Lang-- for the last four years or so. And soon, we will publish that thing and prove that this thing always works.

But I'm going to describe the algorithm without the proof that it works. And what it does-- I'll tell you its goal from a mathematical perspective. So it's interested in practical origami design. So we're going to start from a square piece of paper.

It would also work for triangular pieces of paper or anything convex. But squares are what people usually care about. Sometimes it's used for rectangles also.

The idea is I give to you a stick figure. So that's formally, it's a tree-- a graph without any cycles. It's a metric tree, meaning that I put lengths on the edges. I know this length-- this edge length-- is maybe twice as long as this one. So I really draw it with edge lengths in mind.

Then what I want you to do is find some folding of a piece of paper-- I should really be looking at what I'm trying to match. So here, here, this goes down, such that I want to find some folding of a square paper-- in fact, the smallest square possible, so that when I project-- like this-- vertically, the projection of that folding is exactly that metric tree. And this is called a uniaxial-- this thing is called a uniaxial base.

Let me tell you a little bit why it's called a uniaxial base. We're thinking about what are called origami bases. These there like the beginning of origami models. And most classic origami models-- like more than 60 years ago-- start from one of these bases.

You've got waterbomb base on the top left, preliminary base, fish base, bird base windmill base, and frog base. I can't remember them all. I have a little example here.

This is the waterbomb base. So it's just very simple crease pattern. And why this is useful is, it gives you sort of four flaps of paper to work with. Maybe you make one of them the head and the other two wings and the back one a tail, if you're making a crane. If doesn't actually start not from this space but from the other one.

But the same idea. If you're folding a crane, one of these would be the head, the other tail, and two wings. This is great if you're making a four flap animal.

And if you think about its projection-- and it's easier to think about in this other picture-- the projection of this thing is a four limbed star. You can see it's a plus sign. All of them are the same length.

And so this is actually something you can get out of the tree method. You just give that as your input. You will get this 3D thing and this crease pattern as your origami- - as the output.

Let me show you the program in action. So Robert Lang implemented this thing. It's called tree maker. It's freely available, open source, all that good stuff.

And I'm not an expert at using it, so bear with me. But if we wanted to say, I would like that star. OK, now, I drew it obviously not with all the lengths equal, but it's ignoring the lengths that I drew. And they're actually specified here. So all the lengths here are supposed to be 1.

And then I say, OK, optimize. And then make a crease pattern. And then show me the crease pattern. And there it is, exactly the crease pattern I made, although actually I can see from the mountain valley assignment-- because this is not really-- this is not flat origami.

It made it this way. So it's flat. Of course, the projection is the same-- still four limbs. And so it's dashes for valleys and solid lines for mountains.

But you can make anything you want. So let's say we want to make-- I don't know-- a lizard or something. So the blue lines are the tree. So here's-- can I do this? Yes, there we go.

So here, I have a forearm, a head, a foreleg, another forearm, a body segment, tail, and two hind legs. Maybe I want to make that. So I say optimize and then make a crease pattern, and then boom. You fold that, and it will have exactly that projection.

And assuming it did a reasonable job at computation, this will be the best way to fold this thing. And it's the smallest square that matches exactly that shape. You get the best scale factor between the size of your piece of paper and the target shape.

But actually, doing that optimization-- the first step I did-- is NP-complete. So it's not going to do it perfectly, and we'll prove that Wednesday. But the heuristics are pretty good.

AUDIENCE: It finds a local.

PROFESSOR: It finds a local minimum, and often it finds a pretty good one. And sometimes you can coax it to find better ones, but, yeah. It's not perfect, but hey, it's NP-complete. So you can't do it.

This actually shows you what it would look like in x-ray view. And then you can say, oh, that was nice, but let's-- where is my-- yeah, that was good. But maybe I really wanted the head segment to be shorter length of 0.5. And then I wanted the tail to be really long.

And then you can optimize that and find a crease pattern. And it'll complain. Because it's having trouble. Oh, dear.

Demo effect. I should have tried this example before. It's not-- oh, gosh. It's one of these annoying ones. I should say-- I should add some feature like maybe a strain split something, add a little bit, maybe do that and-- there we go.

I cheated, and I'll explain how I cheated last time. But if sometimes the particular method fails but you can fix it by adding in another tiny limb off the edge somewhere-- and of course, you can then get rid of that at the end when you're folding, it only makes the problem slightly harder. And it'll still find a folding. But we'll talk about that next time. And I'm way out of time, so we will stop there.