PROFESSOR: All right. Today I think is the last lecture at least for the while about origami, and I'm going to talk about where I got started thinking about organic mathematics. The folding cup problem, and this is sort of motivated by a magic trick.

The idea is you take a piece of paper. You fold it flat. You make one complete straight cut. You cut along a line. And you unfold the pieces, and the question is what shapes can you get by that process. So this is like a magic trick.

I showed you making a swan which I have here just for-- to jog your memory. You have a rectangle paper, and you can see the swan on there and you can see a bunch of creases. You fold along all the creases, not the swan lines, and you end up with all the edges of the swan lying right along that line. You cut along the line, and you get your swan, as we did before.

And you also get the anti-swan. The other piece-- I didn't show that last time. But it's really-- it's not like we're making we're not allowed to make any extra creases. We really want the swan. So we cut along exactly the edges of the swan by lining them up onto a line.

So really, you could think of this as a magic trick in cutting, but you can also think of it as an origami problem, which is I want to line up all these edges by folding. How do I do it? And that way it connects to a lot origami design problems.

This problem has an old history. It goes back to 1721. This is the oldest reference we know. This is a Japanese puzzle book, Wakoku Chiyekurabe by Kan Chu Sen. And I think this is kind of like-- it's called "Mathematical Contests" is the translation. And it's sort of like the old version of like these big problem solving sessions that kids do these days to get better at math, and one of the pages poses this problem. I have this Japanese emblem shape.

Can I make it from a piece of paper by folding in one straight cut, and the answer is yes. And this is a solution if you cheat and look in the back of the book. So I'll let you

read that for minute.

[LAUGHS]

You're making folds along lines that end up lining up other parts of the shape so that in the end everything lies along the line, then you cut along the line. We learned about this a bit later than the-- we learned about this problem from Martin Gardner originally, and he knew about the magic world. So Houdini, before he was an escape artist, he was a general magician, and in 1922, he wrote or probably had ghost written this book, "Houdini's Paper Magic". And one of the pages is about folding, and it says here, you can take a square paper, fold it flat, make one straight cut, and get a regular five-pointed star.

And that was pretty cool, and then other magicians picked it up, in particular this guy Gerald Lowe, who wrote this book "Paper Capers". It's more like a very small book. Magic book. And he could make all sorts of different things and he would incorporate them into magic tricks. And he was primarily using simple folds. He would just fold along one line at a time, and make one straight cut, and go to make all these cool patterns.

Like here, I have one of his examples redone. I start from a rectangle, I fold, I fold, I fold, I fold-- these are all simple folds. I take my scissors, and I make one complete straight cut. And usually when I perform this trick, I say look! I made an isosceles triangle. Wow! I made five isosceles triangles. Amazing!

And then I made everything, except the five isosceles triangles. So you saw that coming. And you could make an arrangement of five of these if you want. All that sort of very symmetric stuff is easy to do by simple folds. I'll talk more about simple folds later, but we were really curious about the general challenge. So this is the-- you can download this from my web page if you want to make one. It's pretty-- it's a fun trick.

Good. So I have some more interesting examples. See here I have a rectangle folded, and I make one complete straight cut. And this one has actually a line of

symmetry, so I fold in half at the beginning. So I get an angel fish. Ooh!

All right, you're not impressed. Keep going. Here's another one. One straight cut. We could go all day here. I mean, the point of today's lecture is to see how this is done. In general. Here we have a butterfly. All right. You guys are tough to impress here.

Here we go. This one is deemed thematic. It's almost October, so it's sort of appropriate. We'll one straight cut. Haven't done this one for quite awhile. Get tons of pieces, and if I'm lucky-- open it up the right way around. Jack-o-lantern. Wow, it's amazing! How is it possible?

Obviously you can make many shapes all at once. That's the general idea here. I'll admit, I cheated here because I wanted kids to be able to fold this. The outer octagon was cut initially. Yeah, so it's not from a rectangle. Just to make it easier to fold, but you could do it all at once.

And now, the big demo. In fact, it's so big I think I might want to use the exacto knife. Uh, we'll try with scissors. Got a lot of layers. Oh, yeah. This is why I usually bring my own scissors. Just checking the other side.

They're better scissors. That's what's special about my scissors. All right, time for the exacto knife. Make sure I'm only cutting on the boundary between red and white. Yeah, my lecture notes. Yeah, who needs those? It's flat. A little more cutting.

I believe all the pieces in this case are rectangles. Exciting. All right, straight cut, right? I don't remember which way this one opens. This should be the MIT logo. This one I encourage you to try at home. It's pretty crazy. It's definitely a hard folding. I'm getting used to it now.

All right, so you could make anything is the point. Some are a little more difficult because they have more layers and so on. If I made it out of thinner paper like some of the math magicians do, it is super easy. Onion skin paper or something.

I think I have some pictures of the piece patterns if you want to see what these look

like. And these are all available for download. You want to impress your friends, go for it. They all use slightly different color codings, but it's mountain or valley. And so we want to see how to make these.

So let me state the theorem first. We have our good friend, the universality result, which is any set of line segments on your piece of paper can be-- I'll phrase it as can be aligned by flat folding. A line means when you make the flat folding, all the segments come onto a common line. Nothing else comes to that line, and therefore you cut a long line, you get exactly those line segments.

And there are two methods for solving this problem. The first one is what I call straight skeleton method, and second one I'll call the disk packing method. By slightly different authors. This one was me, my dad, and my advisor, Anna [INAUDIBLE]. This one was Marshall Burn, me, David Epstein, Barry Hayes.

This one's slightly after this one. This is sort of my first computational origami paper. And they're quite different. I mean at a high level, this one's practical. This one is theoretically good, but impractical. This one's actually theoretically bad in a few situations which we will get to, but it works very well almost always. In a formal sense of almost always. This one always works, but it's a challenge to fold. All of the examples I showed you are made with the straight skeleton method.

So that's the idea. That's where we're going to talk about both of them. And first, I have a warm up-- three warm ups. Suppose you had a square paper and you wanted to make a single cut. What folding do I do? Nothing! Yeah, that was easy.

Let's say I have two lines I want to make. What should I do? Fold between them. Fold the lines onto each other, which is angular bisector. If I extend the lines, and I bisect the angle there, then I will fold one line to the other. I brought one just to be totally obvious. You have two lines fold along the angular bisector of their extensions. It lines up the lines, and nothing else.

OK, a little more exciting. What if I had a triangle? How would you line up the edges the triangle and nothing else? Rabbit ear. Yes. Rabbit ear is you fold along the three

angular bisectors of the triangle. This is something we talked about in the tree method.

This is one of the sort of gadgets we use in one of-- where these were active paths. And we fold along those angular bisectors. Angular bisectors intuitively are very good because locally they line up edges. So if I fold along all three of them, and you may know that they always meet at a point, then I kind of line up all those edges.

If I just fold along those three edges, it won't be flat foldable. It's like this floppy thing. And so I've added in these perpendicular-- the purple lines-- to give me my hinges so I can manipulate these arms. You don't have to use all of them. Then you flatten it, and now along this line are all the black guys. All the black lines. We've got a bunch of flat foldings.

All right, so a general idea, and in the straight skeleton method, what we are going to use are angular bisectors and perpendicular folds. Perpendicular folds are also good because locally folds are a reflection if you fold flat. So this line will fold on top of this line.

So perpendiculars are good. Angular bisectors are good. If you fold along all of them, you should line everything up. That's the intuition, and the question is how do I fold a long enough angular bisectors and perpendiculars so that it actually folds flat. And that is the straight skeleton, or one way to do that is the straight skeleton.

And this actually was invented a few years earlier. 95, 96 by a couple of Austrians. That was in [INAUDIBLE] and actually four people. So let me write down a definition, and then we will-- I'll show you what it means.

We've seen a thing like this very briefly in the universal molecule, but this is going to be more general in some sense. And actually, why don't I put up an image of one while we're defining it.

All right, so it is trajectory of the vertices of the desired cut pattern-- that's our input. The graph of edges we want to cut out. As we simultaneously shrink each region. That's every face outlined by those cuts. Keeping edges parallel and a uniform

perpendicular distance from the original edges. So, a bit of a mouthful, but let me draw some pictures.

Well, let's maybe start with a triangle. So the idea with a triangle is you shrink-- this is really the wrong order to do things. I want to shrink-- there's two regions for the triangle. There's the inside and the outside. If I shrink the inside, I get these parallel lines. All-- I want all of these distances to be equal. Those are the perpendicular distances. I keep shrinking, and at some point I can't shrink anymore because I get a single point. The in-center of the triangle.

If I watch where did the vertices go during that time, it's along angular bisectors. Hey, our good friends, angular bisectors. Now I do the same thing on the outside. Shrinking the outside region is like expanding the triangle. As I expand the triangle, the outside gets smaller area.

So actually, it's just the same thing. I get concentric triangles, and I just keep going along the angular bisectors. So that's it. It's not going to give us the perpendicular fold. It's just the angular bisector parts. Still see a triangle in there somewhere.

OK, that's a really simple example, and the only event that happened is that the polygon disappeared. When that happens, you stop with that particular polygon.

In general, there are three things that can happen. We call these events. Three interesting. Things. One is that an edge disappears. So for example, locally, if I have a picture like this and I shrink, and I shrink, and I shrink, at some point-- what's happening is these angular bisectors are meeting, and now I lost-- this edge shrinks to zero length.

So just forget about the edge. Pretend it was never there. Just keeps shrinking now what is these two edges. And I shrink, and I shrink, and I shrink. What happens is, in some sense, these vertices merge-- these edges merge, and now I have one guy going straight up there. And what is that edge doing? It's not an angular bisector of this or this, but it's an angular bisector of the extension of these two lines.

Because if you look at this-- one of these two edges-- they are parallel to the two

original. So if you bisect those parallel offsets, it's the same thing as bisecting the original edges and extension. So this looks good because these two folds will line up those two edges, or this fold will line up these two edge. This fold will line up those two edges. This fold will line up these two edges. And we're doing kind of extra alignment, but everything looks kosher.

Good. So when an edge disappears, you just forget about it. All right, forget-- it's probably "forgedaboutit"-- something like that. All right. Then we have-- a region can disappear. That's what happened with the triangle. And then again, you forget about it. There may be many regions. You have to keep shrinking the other regions, but when one disappears, you're done.

Third thing that can happen is that a region splits. So let's look at an interesting polygon. This one-- the straight edges. And when I shrink this guy-- see, what's happening is this edge is approaching that vertex, and at some point they will meet.

And what we're left with are two triangles in this case. And generally split into two parts, you just keep shrinking the parts. So it's not really like you're stopping at any point. Just the same thing over and over. But if you're implementing this on a computer, you really have to realize that happens. Otherwise you'd shrink them beyond each other and it would be self intersecting and ugly.

But you do the obvious thing, which is you cut where they split. And what will happen in this case with straight skeleton is you keep going along an angular bisector until that little triangle stops. One more edge here.

So that's what the straight skeleton will look like. This edge is an angular bisector of this one and this one. This edge is an angular bisector of this one and this one. In general, if you look at an edge, and you see what original edge can I reach without crossing another skeleton edge, those are the two that you bisect.

So it's really easy to see. Look at this guy. The only two I can reach are this one and this one. I look at this one, the only two I can reach are that one and that one. So it's an angular bisector of those two. In fact, in general, if you finish the outside here

too, it's the same deal. And then-- yeah, all right, enough.

Here's a bigger example. Little turtle drawn on a triangular grid, and you can see there's angular bisectors. This is a straight skeleton. This guy, for example, bisects this horizontal edge and this horizontal edge has a little bit of a boundary case we have to think about, but this is the right interpretation. It's like an angle of 180, so you bisect it to 90. Other fun features.

Here we get a little bit of action on the outside of the polygon. So far, we haven't seen that. So, like these guys meet, and there is some bigger-- there's a bigger turtle here somewhere. It's hard to draw. Anyway, what's happening is this edge is shrinking to zero while this one is offsetting down this way and this is offsetting down this way. So the new turtle ends up being like that. And so on.

You're shrinking every face. So in general, you have a whole bunch of polygons. Or in general we're allowing crazy things like this as this pattern of cuts I want to make. Maybe you want to cut your square into five pieces. I'm going to shrink each of them separately, or in parallel. It doesn't actually matter.

So in this case there's just going to be five angular bisectors. In general, there are several regions you shrink all of them. A lot of the time we think about polygons, and then there's two regions to shrink. And it looks like you're expanding the turtle to go out, but really you're shrinking the outside region. It just happens to be-- there's one infinite region that one looks weird.

A couple other special cases, because I want to do any graph and not just polygons. You could have something that just terminates. So a degree one vertex only has one [? edge ?] into it. In this case, it's not quite well defined what to do because you offset this and you offset this, but what happens here? And there it's sort of arbitrary.

You can do whatever you want, but the simplest thing you can do is to make a-- imagine that there's a little vertical segment here that happens to be length zero, and it expands into the edge of a rectangle. So you end up with these 245 degree

angular bisectors between this vertical edge and the horizontal one. But you have some flexibility there. You can design it how you want.

The other case you can have is a degree 0 vertex. There are no edges here. This is a little funny in the way I defined it. I just said I wanted to align line segments. You could also align points if you really feel like it, and that would be represented by a dot that has no cuts next to it.

If you want to cut out just this point-- I need to make it something. You could think of it as a tiny triangle for consistency with this picture. We think of it as a little square. And so, when you expand it, or when you shrink the outside region, you get four 45 degrees folds.

This is actually how [? Eichholtz ?] et al defined it back in '96, and it's a fine definition. But you have flexibility here in your design process. They'll all work. And this would let you take a whole bunch of points, align them onto a common line, and nothing else is on that line. Because these folds are going to push everything that's surrounding the point away from the line.

All right. Some fun facts. Straight skeleton is nice and small. If you have n original points and line segments in your desired cut pattern, the straight skeleton has a linear n number of line segments-- linear number of creases. So order n.

Other fun facts-- there is a one-to-one correspondence between the edges you want to cut along, like let me pick one over here maybe. Like this-- this is an edge I want to cut along, and regions of the straight skeleton.

So here's a region. A face of the straight skeleton. This guy. There's exactly one cut edge inside of that. That's always the case. You look everywhere here. Every region of the straight skeleton-- it's more obvious if I color them different colors. There's one cut edge inside.

And all of those guys that surround that cut edge bisect that edge and another one. And the other one is the one on the other side. In general, you take one of the straight skeleton edges. There are two sides. There's two faces of the straight

skeleton. This one's crazy and non convex. This one's just a little infinite triangle down here, and that edge bisects those two cut edges.

So it's very easy to walk around the structure. See what it bisects. lots of things get bisected. But, it's not flat foldable, so we're not done. And that's where we need the perpendiculars. So--

I'll write down the definition, and maybe show the picture we're going for.

There's a lot of structures here. There is what I call the cut graph-- the things we're trying to make. Then there's the straight skeleton. You should think of it as a graph drawn on a paper. It has vertices of straight skeleton, which is called skeleton vertices. Regions of straight skeleton, which is called skeleton regions. Edges of the cut graph, which are called cut edges, and so on.

We're going to add a new graph, which is the perpendicular graph. Which you can think of as hinges from tree method of origami design. So what is this-- what does it mean? We started a straight skeleton vertex. Usually there are three skeleton edges coming together. Vertex-- sometimes they're more like this guy. Four.

And if there's three edges coming together, there are three skeleton regions. For each one-- each of those regions-- has one cut edge in it, so we try to walk perpendicular and toward that cut edge. So here I walk perpendicular. I meet at right angles here. I just go off to infinity.

Here I walk perpendicular to this cut edge, and that's cool, but then I leave the skeleton region. At that point I enter a new skeleton region, which is this one, this non-convex thing. It contains one et edge, and when I-- where was I here? I entered. Now I want to move perpendicular to that cut edge, so that when I cross it, I cross it perpendicularly.

Now I enter a new region. It contains this cut edge. I move perpendicular to it, and I don't actually cross it, but I enter a new region. Now I'm in the region of this cut edge, so I move perpendicular to that. And wow, I hit another skeleton vertex. I

stop. OK this example is because it's on a triangular grid, there's lots of degeneracies like that. Usually you'd eventually go off to infinity, or come around to meet yourself. Here I happen to hit a different vertex.

You do that all the vertices. All the skeleton vertices. Now there's some weird ones like this one. Notice there are no purple lines coming out from here, and that's because every region you try to enter you immediately leave. So if I tried-- there's four regions here. Try each one of them. Like this region has this cut edge. If I try to go perpendicular to it, I'd enter a different region, so I can actually go at all. Like I move and then I instantly stop.

So you could think of there being like a zero radius thing there. That's sort of the degenerate case of a river being a disc Is it being a circle. Same thing going on. All of-- in general when you have reflex vertices and their regular bisectors meeting, you're going to lose some perpendiculars because you can't enter them. Here's another one where I just have one perpendicular edge coming out. This one I can reach, but if I tried to be perpendicular to either of these, I enter the wrong region.

That's the perpendicular folds, and that's pretty much the crease pattern. There are technically a few other folds you have to deal with, but that is-- if you want to make something right now, just apply those two algorithms and you get your shape. Just fold along that crease pattern. It will be flat foldable almost always.

Why is it flat foldable? So one thing we can check is local flat foldability at least satisfies Kawasaki's condition because at a typical vertex you're going to have three skeleton edges coming together. And so there are three faces here. Each of them has a cut edge somewhere-- probably draw this reasonably well-- and should have the property that when I extend these-- didn't draw that so well. And I extend these-- these are angular bisectors.

We know the skeleton edges will angularly bisect two cut edges. The two cut edges that are defined by these guys. So I should get an angular bisector here, and those meet. An angular bisector here. And then I also have perpendicular folds. So they may not actually meet this guy, but if they did, they certainly meet the extension.

Hey, that's our good friend, the rabbit ear. Just regular triangle fold. And in particular, you can see that these angles are equal. I call this three prime. It's like 180 minus that, I think. If I'm not mistaken. And this is one prime. Not the best notation. And these are two prime and two prime.

And so I've got these nice angle pairings. That means if I add the odd angles, I get the even angles. Same thing. So I definitely have Kawasaki's theorem everywhere. You could check-- it works even when you have these degenerate situations where more than three skeleton edges come together. For the same reason. You still get pairing. Just more than three of them. All right.

But some exciting things can happen. So I'm going to look at proving foldability, but one exciting thing that can happen is you get a lot of perpendicular folds at a very few original cut lines. So here I'm trying to make this weird pinwheel shape. I want to cut out the bold lines of the cut line. So I want to cut out this square, and then these four squares arranged in a pinwheel pattern around that one. Why you'd want to do that? I don't know, but we're mathematicians so we're going to consider all the cases.

So the straight skeleton is the thin black lines, and that's linear size. That's nice, but the perpendiculars-- if this piece of paper's infinite, the number of perpendiculars is infinite. If you have a finite piece of paper, which is what you usually buy in the store, then it's a finite number of creases. So in any finite region, this is a finite number of creases, but it's a lot of them.

So that's one sad thing. You can't bound the number of creases as a function of the number of cut lines. But I think that's actually necessary. I don't think it's possible to solve this problem while bounding the number of creases in terms of the number of cut lines. That's one of the open problems getting down on one of these pictures. One of these slide-- lecture notes.

There's something else even more annoying, though. It happens even in a finite piece of paper, and it's even more obscure why you'd want to make this. But the

bold blue lines are the cuts, and then the thin black lines are the straight skeleton. You could tell this spans many years because I keep changing notational style, and this is from the textbook.

And then you get these perpendicular folds. So I haven't drawn all of them but these dash lines, the light blue. And this example is set up so that-- let me get this right. This width is an irrational multiple of this width or this width. One of those. Things are irrational, so they're not very nice numbers. And what I need to do to finish this picture is these guys go-- they enter a new skeleton region.

They're actually going to keep going straight because there's two cut lines that are parallel to each other. It's going to go up there, and it's going to cycle around. Let me do one round. Who did I move? This guy.

So this guy I just extended down. He's going to turn around, make 180 degree turn, and you can check each of these. Your setups do 180 degree turn around this axis, around this axis on the bottom, and on the top around this axis and around this axis. Depending wherever you enter, it's like a racetrack. Keep going around and around.

And if you follow that one guy a little bit farther, it looks like that. And a little bit farther, and it looks like that. It never finishes. So in fact you completely filled this region with creases. It's like a dense region of creases. Now this would be a bitch to fold. I don't recommend you try it.

And this is really a situation where this algorithm fails. The good news is if I move any of these vertices-- the cut vertices the tiniest amount-- this will disappear. I really had to be very careful and get lots of degeneracy for this to happen.

We don't actually know how to prove that. It's conjecture that if you take any cuts-- any graph of cuts you want to make, and you perturb the vertices in a tiny epsilon disc then the resulting thing will not have this density behavior. I'm totally sure it's true, but we don't have the proof.

So that's life. This is why I said skeleton method works almost always. There are these annoying situations where it doesn't really give your crease pattern. So if you

feel like unless you somehow think this is legitimate to make infinitely many creases. I don't think so.

All right, let me tell you a few more things to make this practical for you. You want to-- what you really want is a mountain valley assignment. Before I showed you lots of perpendicular and skeleton edges, and basically the way it works is if you look at any skeleton edge like this one, it's bisecting in this case a convex angle. So I make it a mountain. Here, red is mountain, blue is valley. Dot dashes mountain, dash is valley. That is the standard.

Whenever I'm bisecting a reflex angle, then I make the skeleton edge a valley. And that's basically true. Convex angles, mountains, reflex angles, valleys. That's for the straight skeleton edges. Yeah.

Like this fold, or this one? Oh, this guy. All right, this is a bit of a special case. Here I'm really bisecting an angle of zero. To extend these guys out, they need an infinite, and they form a zero angle because they're parallel. And I call zero a convex angle, but I just defined it that way, and so this is a mountain.

Whereas this guy bisecting is barely convex. Is that really a mountain? No, that's a typo. Good. This one should be a valley. Pretty sure. Yeah, should be a valley-- wow, this is a weird crease pattern.

That's not a straight skeleton there. Never mind this picture-- I knew there was always a bug. I think there's a typo in the book as we say.

How did I make the initial pattern of the turtle? I just drew something that looked like a turtle. Anything. I happened to draw this on a grid, but there's no reason I had to.

I designed them. So in this example, I designed the ratios to be really nasty. Like root 2 over 10 ratio or something. The whole thing will-- if I perturb these vertices at all, the whole thing will fall apart. I won't get these 180 degree turns. Things will end up going off to infinity.

The hard part here is actually-- rational ratios are quite common. What's uncommon

in this picture is that this thing is closed up, and you never escape. Almost always, there'll be a little gap and you'll eventually reach the gap and then go off to infinity. So that's what happens in a typical case.

So if you drew a picture on the grid this would never happen. That you can prove. Yeah. Square grid. Probably also the triangular grid. You need to be a little careful because you want all these constructions to stay on the grid, but I think something like that is true.

OK, let's move on to how we construct a folded state. When this algorithm works, when it gives a valid crease pattern, you know it's locally flat foldable because it satisfies Kawasaki, but how do we actually know that it globally folds flat? To do that, you have to describe what it looks like after it's folded.

And the idea here is to look at what we call corridors but are essentially discrete versions of rivers from tree theory. So you have these constant width strips that turn at skeleton edges, and they could go off to infinity on both sides. In general they could loop around and meet themselves again, but in this case they actually all go to infinity.

And if you look at one of those strips-- you could actually just cut this out of your textbook. Just slice it up and look at how that's folding. Well, it meets a skeleton edge, and then maybe it meets a cut edge. Usually you don't fold those. Then it meets another skeleton edge. It just meets edges one at a time. It's never complicated because we divided along all these perpendicular folds. You really only meet one edge at a time.

Which is good. In fact, if you look at one of these skeleton edges it's creased along, normally you think of that as an angular bisector of these two cut edges, but you can also think of it as an angular bisector of these two perpendicular folds. Because if you bisect these two things, you also bisect two things that are perpendicular to them. It's like two wrongs make a right somehow.

So this guy bisects those two creases. So if you fold along here-- actually you align

these creases. It's a duality-- you are aligning the perpendicular folds. That fold along here, you line up this fold with that one, this fold with that one. I fold here, I lined up this fold with that one. There's like a zero length fold that's there. You fold along all these things. You line up this with itself, and so on.

So you follow along this thing. This corridor folds down to basically a rectangle. It's got some rough edges in the top and the bottom, but it lies in the strip in 3D, and I have a picture of that. So I took the one over here-- this blue corridor-- and if you fold it up, it looks like this.

OK, now in particular, you can check at this point. It's pretty easy to check because of all this bisectorness-- Bisector goodness-- that you bring into alignment all the cut edges. So for example, this guy, because it bisects that cut edge and that cut edge, it brings them into alignment. And you can see that somewhere in this picture. I think it's these two guys.

It can be a little more complicated. Like over here I have a cut edge, then I have a bisector, and then a bisector, and then a bisector, and then another cut edge. But if you think about it right, it's-- I don't happen to meet these cut edges, but I'm effectively bringing this into alignment with this, and then this into alignment with this, and then this into alignment with that. So in the end, I line this with that, and that's what's happening up here on the left. Where I don't quite come all the way down, but I still end up lining everything up.

So this is the solution to the fold-and-cut problem as long as it actually faults. And to show that actually folds, you just need to show that these corridors-- I forget, I think we call these accordions. It's been a long time. Since '98 I wrote this paper

You take these accordions and you just want to see how they fit together. And low and behold, they fit together in a tree in this picture. It gets more complicated in another picture, which I will show you in a moment. But in this situation where every quarter goes off to infinity on both sides, you get a nice tree structure. And as long as you could fold this tree flat, then you can fold this thing flat. Because each of these edges of the tree is this very simple accordion structure which trivially falls flat.

The other thing you have to check here is you actually alternate mountain valley. That's a little more subtle, especially when I don't draw the mountain valley assignment. But it turns out you always alternate between mountain and valley in this picture, which is great. That's the thing we know always folds flats. It's like a 1D folding.

So these are super easy to fold. You can fold each of them if you cut along all the dash lines You can fold each separately. Then you need to join them together where the edges here-- just like in tree theory-- the edges here correspond to these rivers.

And now you need to somehow attach them here. Check that where you attach them, there's no crossings. I'm not going to describe, but it's pretty easy. Plain area essentially of that diagram, and then you need to fold this tree flat. Folding a tree flat is actually kind of a segue into next lecture, which will be about folding linkages.

In this case, it's really easy. You just pick up some root, like the letter "A" over there, and you hang the tree. Pull up. Technically this is like a [? depth ?] [? first ?] search of the tree. So you just walk down-- aways walk down until you're finished, then you go back up. Walk down some more branches.

You end up drawing everything downward away from A. And it will be a flat folding. There won't be any crossings here. And then this is a 1D representation of what's really going on, which is that above each of these edges is really an accordion, and you need to adjoin them together there. So we'll basically do this modular folding where you fold each accordion separately, put them together according to this. Boom, you get your flat folding.

From this picture, you could read out the mountain valley assignments before the perpendicular folds. This looks like a valley, this looks like a valley-- this looks like a perpendicular fold I didn't use because there's no crease there. That's flat. Whereas a mountain-- mountain's probably at the top at A.

What really happens-- if you want to know-- really what we're deciding is whether

this starts mountain or valley, and then it will actually alternate back and forth as you move along the perpendicular. So that's basically how you construct a folded state. In this situation of so-called linear corridors. Now there are a bunch of things I haven't mentioned, but I think-- I don't want to talk about them too much so I get out to the other topics.

So, what I just talked about is something called a linear corridor case, which is really where it's most beautiful-- this construction of a folded state-- and linear corridor intuitively is something like this. It goes off to infinite on both sides. Has constant width all the way. Of course, it's really a discrete thing. Not a smooth thing.

Let me say conjecture-- if you're cut graph has that maximum degree 2-- it means at most two edges at every vertex. This is a very common scenario. This is if you want to make a polygon, every vertex has degree 2. If you want to make several polygons, every vertex has degree 2. I'll even let you have vertices of degree 1 or 0 for free, but mainly we're thinking about degree 2 everywhere.

Then, we almost always have a linear corridor. So this is why the situation's interesting, although unfortunately this is still a conjecture. I'm sure this is true, but proving it-- I don't quite know the right techniques.

So in this typical situation, you take any picture you want. You slightly perturb very vertex randomly, say, then with probability 1-- 100% probability-- you will get only linear corridors. And that's the situation where it turns into a tree. It's easy to fold life is good.

The annoying case is circular corridor case. This takes a lot more work to prove, and I'm not going to talk about it much. A circular corridor looks like this. Here we have these in with rivers also. So you just loop around. Still constant width everywhere, but you meet yourself. You don't go out to infinity.

It's harder. Why is it harder? Well in particular, if you look at how one corridor folds, it's no longer-- it's like the same situation we had in like lectures two and three where we had on the one hand a 1D folding was really easy. But then when you

made it circular-- you're just folding a circle instead of folding a line segment-- now you had this wrap around issue.

So, like, these guys would have to line up. It turns out they will line up because everything is bisecting and whatnot. These edges will line up, but now you have to join them together. And if this part went all the way down here and came back up, then you'd get an intersection.

And it turns out, in general, I get a choice of who's first and who's last. I have a circular order of things, and I get to choose where I break that circular order and make it a linear order. Where I do the wrap around. There's some circular corridors-- you can't even break them and make it work. Kind of depressing.

So this is definitely harder and that sometimes it's not possible. But we can save a little bit, which is-- I don't have an example handy. I wish I did. I'll have to reconstruct it. This is so long ago.

Here's a way to make it definitely work. Fold all the cut edges. So far in the pictures I've been drawing, I didn't fold along the cut edges because I really wanted to separate the green region here from the yellow region. If I folded this up-- this is quite a complicated one to fold-- you get the cut lines somewhere like over there, and then the green stuff will be always above the cut line and the yellow stuff will be below the cut line.

In general, this is called a side assignment. You have a bunch of regions you decide which ones you want to be above and which ones you want to be below. And usually, you have polygons, and you say the interiors are above and the exteriors are below. But in general, you could ask for anything you want. You could say maybe I want both of these to be above. If you make both of them above, you have to valley fold along all of the cut lines.

So if you do that-- you say I want all regions to be above the cut line-- you can still line them up. You end up folding along all the cut edges with valleys. And then wrap around is super easy. We take this thing, and in fact, everyone's folding along the

black lines. So really everybody comes down to the floor, and then the wrap around is just underneath the floor and life is good.

So that's one way to deal with this case, and you have to prove that works. It's complicated. I would rather go onto other topics.

I do think this would make a fun project. It's not easy to make these crease patterns. Currently we draw them always by hand, meaning with the fancy drawing program that knows how to do angular bisectors. And it's an art to move around the points so that you get lots of alignments.

Like when you get four straight skeleton edges coming together, that means you get fewer creases. That's a good thing whenever you can make that happen. So when you could give me software to help do that, I would love you.

So let's move on. Any questions about the straight skeleton method? Now I'm going to switch over to disk packing. All right. Same problem, but now I'll give you a method that always works in theory. Just difficult in practice.

I think this is good chalk because it's yellow. Generally, if it's yellow on the outside it is railroad chalk, which is the good stuff. But, what the problem is-- we have bad erasers. It's persistence of vision. You just get to remember what I used to write.

I don't want to write down the algorithms. It's complicated. I want to give you a visual overview of the main steps.

I guess there's nice figures do that for us. So we start with a very complicated shape we want to make, like this quadrilateral. And you can see disk packing is involved. The very first thing we do-- and I'll tell you why in a moment-- is thicken the graph you want to build. So maybe it's a polygon, maybe it's a graph-- I'll think about the polygon case for now because it's easier, then I'll come back to the general case.

I thicken it by tiny epsilon. Just offset in both directions. Just like as if you're starting the straight skeleton method, but then you stop after epsilon. No events happen in epsilon time.

OK, then I have this picture-- the purple stuff, pink stuff, whatever. Magenta. 50% magenta. I happen to know I drew this figure.

Now I'm going to take some 50% cyan disks and pack them to fill this region. Now what I want-- there's three properties I want. One is that every vertex, but you have to think about each region separately which is a little bit confusing. Let's think about the outside. It's a little easier. Bigger.

Every vertex of this graph on the outside should be the center of a disk. There's a disk center here. There's a disk center here. there's a disk center here, and a disk center there. On the inside it's also true, they're just different disks. Then also I want the edges of the graph to be covered by a radii of disks.

And so here's a radius of one disk. Here's a diameter of a disc. Here's a radius of a disk that covers the edge. So in other words, I want to fill-- along the edge I want to have a bunch of centers so that I completely cover that edge with blue. You'll see why later. Question?

The disks have to be non-overlapping. These properties are actually quite challenging to achieve. Your question is why do we use small disks. Disk, because if I had a big disk here, it would intersect this disk. Now I didn't have to make that disk so big, but if I made that one smaller, I'd have to have more disks here. Or here there's also two small disks. That one I probably could have gotten away with a bigger disk.

Oh, no, but then on the inside you have a problem. So these guys actually have to match up. That's another constraint. And the inside and the outside have to match up. Here there's a slight change in radii to compensate for the epsilon. Along the edges they're exactly the same. So if I made this one a big disk, it would overlap this one. So I could make that one smaller, but then-- other problems.

So you have to simultaneously balance all these constraints, which is a bit exciting. What else? The discs don't overlap. And the last property is that the gaps between

the disks have always three or four sides. Why? Because I want it to. It will make life easier. You could try to deal with more sides, but three and four is nice. Yeah, I'll get to that.

Why do we care about number of sides? Because I'm going to draw a graph. I'm going to subdivide my original graph here with these red lines to say whenever disks kiss, I will draw-- connect the centers of those disks.

And because these gaps always have three or four sides-- it's not the best example. Here's like three sides. The red lines I draw will outline a triangle. Whenever I have three sides, whenever I have four sides, I'll have a quadrilateral. So I've subdivided my regions into triangles and quadrilaterals. OK, you have to believe that this is possible. I can sketch an algorithm for you, which is you draw a tiny disk at each of the corners, and then you draw lots of tiny this along the edges to fill the edges.

And that will satisfy everything. I mean the disks will be non-overlapping because they're super tiny. They won't get near any other disks from some other side. And what other good things? Oh, but the regions will be ginormous. They won't have three or four sides. They'll have 100 sides, a million sides-- who knows.

Well, whenever you have some crazy region outlined by disks-- might not be convex. Whatever-- just draw the biggest disk you can in there. I'll get it to turn into a disk eventually. That does not intersect anybody, but if it's the biggest possible, it will actually touch at least three sides. If you degenerate, it might touch four.

But in general, it will touch three sides, which will subdivide that region into three pieces, and you can show that those pieces are all little bit smaller than what you had before in terms of number of sides. Except when you start with a quadrilateral. When there's four sides, you'll get quadrilaterals and triangle.

So you can't go below three and four. It'd be great if we could always get down to three sides in every region, but we can get down to three or four. Anything bigger than three or four you can show. This will make it strictly smaller.

So that is an algorithm. It's not super efficient, but it will find a disk packing with all

these properties.

Then we do the subdivision. Now what do you think we're going to do? What do we do with the triangles? Rabbit ear. That's the key phrase for today. So remember our good friend, the rabbit ear, and then there was the universal molecule-- Lang's universal molecule for the quadrilateral. We're going to use that for the quadrilaterals.

And it turns out there's some nice properties here, which is the perpendicular folds of the rabbit ear will always hit right at the kissing point between the two disks. And same thing in here. We've got these four disks. We've got this quadrilateral region in between. and the perpendicular folds that come out of these two points. You may not remember exactly what's happening here as we shrink.

And then in the tree method, this became an active path. There's no notion of active paths here, but we just make that so. That these perpendicular folds will end up hitting kissing disks, and we'll end up actually with the four arm starfish. In terms of the tree you get and the articulatable flaps here, these guys will all meet at a point. That's just the way this works with disk packing.

And you can think of there being disks here and you're actually applying the tree method to that flap pattern, and that's probably the easiest way to think about it. But what's good for us-- do I have a picture? Not yet. But the point is, I have perpendiculars coming out of here. I have perpendiculars coming out of here. They will meet because these disks kiss at the same point up from both sides.

Perpendiculars meet. That's good. That means I don't get perpendiculars bouncing all over the place. So all this work is to make sure that perpendiculars are well behaved. It's a lot of work to do it, but it does it.

Now when you fold this thing, what we end up doing is lining up-- remember there was two copies of my polygon. There's the inner copy and the outer copy. I end up lining up all of these guys-- I'm got to go back to the picture. I'm sorry.

So we have this inner copy, and what these molecules end up doing is lining up all

the edges of this quad, all the edges of this quad, all the edges of this triangle. All those edges on the inside will become lined up on one edge. All the lines on the outside become lined up on another line. Turns out it will be parallel to that line.

But what we really wanted to line up were these edges, and you can see why we had to do the outside of the beginning, because otherwise we'd get tons of extra junk on our line. We only want these edges on our line.

So we did the offset so that all this stuff will come to one line. All this stuff on the outside will come to another line. And then we get this picture. So this is one line at the bottom. Another line at the top. We really wanted to line up stuff-- the blue stuff there.

And there's still some junk on our line. These cyan triangles represent things that come from down here, but we really don't want them to cross our line. They just happened to. So we have to sync them repeatedly. Do lots of folds to make them underneath that epsilon line. Then we can cut along our line, and we're done. Easy.

To prove that this works, we'll view a little sketch. This is kind of fun, and it's one piece of what we're in the process of doing for tree maker. This is sort of like a special case of tree maker. You just have very simple molecules and a relatively simple way in which they're combined.

Here I've done a simpler example. I want to make a square, and I end up decomposing in this case into nine molecules-- nine quadrilateral molecules. A very simple disk packing which I have not shown the disk packing here. The idea is I'm going to make some cuts in my paper to make my problem easier.

I'm going to have to pay for that, because later I really want those edges joined. I'll have to glue them back together. But to make it easier think about, I cut those four edges. So that the way in which my molecules are connected to each other is a tree, because I like trees. They're easier to think about. Easier to do induction over.

So that's the blue line connecting the centers in a tree. The other remaining edges

in the grid have been cut away.

Now the idea is-- it's kind of like what I was drawing for the linear corridor case. You have a tree, you pick up the tree from some node, and just hang it down. And in this case, we hang it from this molecule. The red edges are mountain, so three of these are going to be valley. One mountain.

The idea is this thing reaches around the next guy, which reaches is around the next guy, which reaches around the next guy, and there's actually-- there's two valleys here-- two little pockets. Each of-- this guy goes in that pocket, this guy goes in that pocket, and recursively, it just works. I think I have a picture of what's actually happening here.

Yeah, is it's hard to really draw, but each of these forearm starfish has one mountain and three valleys, and you just nestle inside your parent in the tree. And this is really easy to show that there's no crossings here because just joined to your parent, and it's a nice nesting structure. It's just in the same way that trees can be folded flat. You can fold all these molecules flat and join them together in a tree.

But we didn't really have a tree. We had all those extra cuts that we have to re-suture. So we have this picture, and now we really have to join up these edges and think about what the mountain valley assignment is there. And it works.

This green thing is the boundary, and then I have connected the dots. Each of these dots corresponds to one green edge here. I forget whether it's this edge or that one, I think. It's just a single edge.

And so, for example, these two are-- these two joins, and then the joins above that nestle around it. And then the other branch at the top are joins, and in the leftmost cut are these two joins. You have to make these joins, and really all you need to check is that these joins form a non-crossing picture like they do here. And that's almost obvious because this is a planer diagram and we're cutting along a planer tree, and so this is again a depth first search kind of thing.

So there's one tree we call the dual tree here that works because it's a tree, and

then there's the cuts you make which are different trees-- primal tree if you want-- and that also works because of planarity. And it all works. That's the hand wavy version, and you can read the textbook if you want more details.

Oh, gosh. If you want to solve more general graphs, you can do it. In general, you have to offset all of those cut lines, and you get all these things-- along the pink lines here you line things up, but you really want to line up these blue lines-- purple lines. And so you have to do more syncing to get it to work. Now I have all the things I want to line up on this line and this line. I fold in the middle, and now they're lined up.

That fold in the middle-- yeah, that will work. Good. Might have to do more syncing. Whew. Questions about disk-packing method? This is a bit of a whirlwind tour, but I wanted to get through it quickly to tell you about a new result. Just got accepted to a conference to appear in October. Pretty soon.

And it's a project that started in this class in the problem session three years ago, and we just solved it. Took awhile. Took another co-author to chime in. And it goes back to the early history of fold and cut which is simple folds. All the magicians were using simple folds. What can you make with simple folds?

Now you've been wowed and dowed that you could make anything with arbitrary folds, but simple folds you cannot make anything. Because the first told you make, say, this one, has to be a line of symmetry of your diagram. Got to stop making my life hard.

If you can fold something, you can never unfold it. That's the usual simple fold model. This has to line up-- the cuts you want to line up over here, that are exactly be the cuts you want to line up here. So you can only make symmetric diagrams. The first fold has to be a line of reflectional symmetry.

But is that the only property you need? No. Kind of have to have symmetry for while until you're done. How do you formalize that? Well, we came up with an algorithm that in polynomial time, an efficient algorithm will tell you whether a given polygon

can be made.

Like this polygon looks good. I think-- yeah, I think this can be made. So I think I would fold along and do a bisector here, and then this basically disappears. Folding over. Then I would fold along and get a bisector here, and then this disappears into that, and then maybe I can fold here. Does that work? Barely. I mean I've got to make sure that this blank paper does not come onto that.

But if that's a problem, I can probably make-- I could make a fold here, for example. Shrinks that up. There's lots of things you can do. This is a borderline case whether it's yes or no. I will give you an algorithm that does it.

For polygons with margin. Bit of a technical condition. Something that is pretty typical. What I mean is the thing you're trying to cut out does not meet the boundary of the paper. There's no margin here. It'd be hard to print out. So I really want something that has margin. That's a typical case we care about.

We actually need this for the proof to work. We also need that it's a single polygon. It does not work with general graphs. This algorithm. Because more complicated things can happen. It might be [INAUDIBLE]. for all we know, the general case.

So here's the algorithm that I'll give you in number form. First thing you do is guess the first fold. This is a powerful idea that even most algorithmisists don't necessarily know. The idea is what could the first fold be?

Has to be a line of reflection symmetry. Turns out there's a linear number [INAUDIBLE] most. You can find them in linear time. All these good algorithms for finding them.

But which fold do I make first? The answer is I don't care. Let's just try them all one at a time. This is what I call guessing. Just imagine from now on that we made the right guess, but if you end up failing later on this algorithm, just go back here try the next one. There's only N of them to try.

So you're going to multiply the running time of the rest of the algorithm by N, and if

this is N to N squared-- which I think the rest of the algorithm is N squared-- the whole algorithm will be N cubed because you just run this over and over for each possible first fold. We don't have a great theory to find the first fold. Just try them all.

That's step one. Step two-- that's the only guess we're going to make. Fold down to convex hull. This is a central idea.

So we have this polygon we want to make. There's all this extra blank paper. I don't like extra blank paper. Just get rid of it. Make lots of folds to fold the blank paper onto itself until it gets so tiny it just goes slightly around the convex hull. Convex hull is the smallest convex polygon that contains your shape. So it'll be like that. It'll reduce the paper down to that.

And I do this a lot. I might as well. It makes the problem easier because I have less blank space to worry about. Blank space is a problem because if I [? fold ?] blank space onto a cut, it's bad. It's not allowed.

So the next thing we do is make the best possible safe fold. I need to define that, but a safe fold is just the folds we're trying to make, which is locally they are lines of symmetry. So like this one was a good fold after I made this fold. So this is-- all the right stuff here is gone.

It's a good fold because it folds this on to this, and it folds blank space onto this blank space. So anything that comes into-- on top of each other is identical. That's a safe fold. And repeat.

That's the algorithm. How does that work? Why does this work? So it's the obvious algorithm, basically. It says make safe folds until you're done. If you finish, then you're done. And then the answer's yes.

If you don't finish-- which is a little hard to check because you can always make microscopic folds-- but you take the limit and-- if it's possible to do this in polynomial time-- you find that out. Turns out I can't finish by making safe folds. Then you can show that actually there was no way to make that pattern by simple folds.

So let me give you an idea of why that's true. After I make a single fold-- the first fold-- my picture looks like this. It's no longer a polygon. It's like half a polygon. It ends at the boundary of the paper. It begins and ends with the boundary of paper, and you have some chain in the middle. We call this a passage. It's like a path you wander along.

And I want to somehow bring all those edges into alignment. Here I'm already using it as a polygon. If it weren't a polygon, there might be more than one of these.

Now I want to make a fold. For example, this fold is safe because it folds this on to this, and it folds blank space onto blank space if I do it right. And keep doing that. Now when I make a fold like this, what happens is I can think of this region that I folded-- the smaller side-- as disappearing. It just got absorbed into the paper here.

So the graph that I was trying to line up got smaller. That's clearly a good thing. Makes my problem easier. The piece of paper I was folding also got smaller. That's a good thing. But that's not always true.

Suppose you had a piece of paper like this, which could happen after a bunch folding, and then you fold along a line like that because, for example, your passage looks like that or something.

When I make the fold-- this has to go off. When I make the fold I get this crazy thing. Not drawn to scale. And this polygon does not fit inside this polygon. So my paper got bigger in some places.

And that's a worry, because now I have the stuff-- maybe it happens to be blank space. Maybe there's other junk that got out here. And now have to worry about collisions with this bigger piece of paper. And this is always our sticking point, but there's some magic you can do. In fact, the picture cannot look like this.

Because, look, you've got some portion of your passage to the left of the crease. You've got some portion of the passage to the right. One of them has to be shorter. Plus, this is a line of symmetry. So wherever I have a portion of my passage over here, I will have a portion of my passage over here until I run out of length. One of

them is shorter.

So the shorter one like this one gets totally absorbed by the larger one. So the shorter side always disappears. So in this picture, I have the long side of my passage, and it's really a subset of the original. If I reduce this to the convex hull, like this, this stuff disappears. And in general, if you do this fold down to convex hull, this repeat goes back to step two.

If you fold down to the convex hull, you can show that not only does your passage-- the thing you want to cut out-- get smaller, but you piece of paper also gets smaller. Guaranteed. And once you know the paper get smaller and your things you're trying to align get smaller, you know that every move is safe. So you never get stuck by following this algorithm. This works for polygons with margin, but not in any other situation as far as we can tell.

Cool. The last thing I wanted to leave you on is going out a little way from regular 2D flat sheets of paper. You can generalize and go up a dimension to folding polyhedra surface. Here, a surface of a polyhedron.

You've probably done this. You take a cereal box and you can collapse it flat. Is that always possible? It's called the flattening problem, and the answer is yes. And you can think of it as a fold and cut problem, because of the fold and cut problem, you have some polygon like this diamond. You make some collection of folds that brings the boundary of the diamond to align.

So if you forget about what's happening on the inside of the paper-- you just look at the boundary of the paper-- you're folding that one-dimensional boundary so that it collapses down to a single line. What I want to do is this up a dimension. I take a 3D cube of paper-- solid cube. I have embedded within it some polygons that I want to bring to a common plane. And I want to fold the solid cube through four dimensions, but flat so it ends up back in three dimensions. I get a different 3D solid, but with multiple layers right on top of each other-- little bit a fourth dimension hanging out there.

But if I just look at what's happening to the boundary of my polyhedron-- say I start with a dodecahedron or something embedded in there-- and I want to fold this thing so all the sides of the dodecahedron come to a common plane. That is the 3D folding cup problem. It remains unsolved.

I suspect it's possible to solve even with straight skeletons and perpendiculars, but it's really hard to draw the pictures. So we have not resolved that one way or the other. But the boundary problem-- forget about what's happening to the interior and the exterior of the dodecahedron. If you just look at the surface of the dodecahedron, that you can fold in 3D-- we think-- and you can show and burn in haze from the complexity proof a couple lectures ago, and also on this disk-packing method with our co-authors.

They prove just two years ago that if you have any orientable manifold, which is things like polyhedron but no Mobius strips, no Klein bundles, and other ugly things. They have to be manifold, so you're not allowed to join three triangles together along a single edge. That would be forbidden. So it's locally flat.

In such a case, you can flatten the thing. And the proof is very similar to the disk-packing method of fold and cut, and in the textbook we talk about how do you apply that to do something that's just like a sphere. A regular polyhedron. That's pretty easy to do with the disk-packing method. They generalize it to the case where you have polyhedral doughnuts and all sorts of fun things.

But there's tons of open problems here. So we know how to flatten surfaces, and that's useful for things like folding airbags flat. But can you fold the 3D solid flat? You can think of-- we have here 1D edges which we are collapsing to a 1D line. There is also zero dimensional points here, which we don't bring to a single point. It'd be nice if you could-- the generalized fold and cup problem is you take a D dimensional thing, and you have all of these-- there's 0, 1, 2, 3-- up to D dimensional parts to it-- or D minus one-dimensional parts.

You want to bring each of them down into alignment so that all the vertices come to common point, all the edges come to a common line, all the two-dimensional faces

come to a common plane, and so on up the dimension hierarchy. That is the ultimate open problem. I think we end the book with it, and it's totally unsolved.

Any questions? That's folding and cutting paper, and next time we'll start linkages.