PROFESSOR: So we're going to do a problem that looks like the guitar problem. But it's a bit cooler. At the same time, there are fewer states, so I think it's easier to manage. How many people know how to play DDR?

AUDIENCE: Well.

PROFESSOR: Yeah. OK. How many people know the rules of playing DDR? Because otherwise I won't be able to lift my hand up. OK. OK, so let's go through an algorithmic model.

So, you have a board that basically looks like this. And you have four touch sensitive pads. Up, down, left, right. And then you have a center position.

You also have a sheet of notes that looks something like this. Say, up, up down, down, up down, up, up down, left right, left, left right, right. So on and so forth.

So when you see an arrow of up, that's a constraint that says one of your feet has to be on the up arrow. Has to tap the up arrow, actually. There's no constraint for the other foot. So your other foot can be anywhere else on the board. It doesn't matter.

You are also allowed to tap the board some extra times. So if you don't have a note showing up, you can still hit the board. And there's no penalty. So you can hop around as many times as you want, or as many times as you can, until you run out of breath.

So up basically means, it's actually one foot has to be up. The other one center. Also, of course, because of the design of the board, it doesn't matter what foot is where. So what are possible goals for this game?

AUDIENCE: Not to get an F.

PROFESSOR: OK.

AUDIENCE: Hit all the arrows when they line up with the top row.

PROFESSOR: OK. So there are two different ways of doing it. One, you said, "not to get an F." So I would say, given some limited set of skills. So given a set of skills, or your skill level maximizes your score. right?

AUDIENCE: Yeah.

PROFESSOR: This is dynamic programming, so we always want to maximize something. Now, if you're really good, then another possible goal is hit all the notes with a minimum amount of effort. If you're going to be in a competition, and you're going to be doing 10 songs, you don't want to die after the first song, right? So, assuming you can do everything reasonably well, this is another possible goal.

Any other goals? So we're brainstorming here. This isn't set in stone. We've already seen the formal solution for this problem. So I want to play with it for a little bit, right? We brought a DDR pad, so I want to play with it. What else could you hope to maximize or minimize for with dynamic programming?

AUDIENCE: Not look ridiculous.

PROFESSOR: OK.

AUDIENCE: I guess we're looking at the cost of each move, right?

PROFESSOR: So, hit all the notes. Minimum effort, or otherwise say best appearance. So every move looks somewhat good or somewhat bad. And you want to do the moves that look as good as possible.

So this maximizes entertainment, right? if you're on TV, you probably want this. Now suppose you are in a competition and you only have one track to play.

AUDIENCE: Well, you know where all the notes are.

PROFESSOR: Sure. Like, we assume that we know this ahead of time. So you've memorized this. And you're trying to compute the best strategy, so you can memorize that. Then when you go and play, you dominate.

AUDIENCE: Yeah.

PROFESSOR: So, if you have one track, then I would say that what you'd want to do is minimize your probability of failure. Right? So given the probability of failure for each possible move. You want to minimize the overall probability of failure so your teammates won't hate you. So, hit all the notes and minimize.

So this one's a little bit different from the other ones because all the other ones are already set up as a nice problem where you add up things. So you can solve them with graphs or with dynamic programming. If you want to minimize the probability of failure, then you want to maximize the probability of success.

The probability of succeeding on all your moves is the product of the probability of succeeding on each move. All right? So if you have five moves, then you have five individual probabilities.

You multiply them up. And that's the probability that you execute the whole sequence correctly, and not stumble. So we have products instead of sums. Yes?

AUDIENCE: Isn't it related to the maximizing score goal?

PROFESSOR: You can. Well, not necessarily because this one says that you only have some possible moves. I think in the end, all of them, except for this last one, can be solved using the recursion depth we have for it. The last one, you have to do a bit of massaging. And we're going over that right now.

So, you have products instead of sums. How do you turn products into sums?

AUDIENCE: Logs?

PROFESSOR: Yep. So, for this one, I would want to use logs, so that I can say that I want to maximize the log probability of success. Which is just the sum of the log probabilities for each move. And then it looks like dynamic programming, as usual.

Now, if I wouldn't want to use logs, if I'm using the DP formulation, where I'm doing recursion, I can incorporate products there. But it turns out, I didn't practice. If you have a lot of numbers that are close to 1, or that are close to 0, so all the easy moves are going to be really close to 1, for example.

If you have a lot of numbers that are close to 1 or close to 0, if you multiply them up, you get numerical instabilities. So we get the number that's really close to 1. So for example, if I'm choosing whether to do, say I'm here, and I'm thinking, do I want to move like this? Or do I want to move like this, and then like this?

These are both pretty easy moves. So, in both cases, we're looking at probabilities of success of 99.991%. And maybe 99.992%. So if you have numbers with probabilities that are really close to 1, and if you have a lot of them and you try to multiply them, you're going to get a bad result.

So all the products are going to start looking the same. So we're going to get a random solution, instead of what you want. This is a practical thing.

It's called numerical instability. And it's a practical reason why you'd want to use logs, instead of multiplying things up. So it's not just a theoretical thing. It also makes life nice in practice.

OK. So we have a goal that looks like this. Say, let's go for this one. Hit all the notes, minimize the effort. And all the other ones can be reduced to this one.

We need to define effort, right? Let's say that we have a function called delta, that takes two foot locations-- so, where both of my feet are-- from and to. And it gives me a number from 0 to 1, where 0 is really easy. And 1 is really hard.

So from would look something like, my left foot is up. My right foot is centered. To would look something like, my left foot is centered and my right foot is up. So

basically going from here to here. So from left foot up. Right foot center. To here.

So there is some difficulty there, right? We have to jump, so it's not 0. I'm burning some calories here. So any move that I make has some difficulty. And I want to solve the game so that overall I have a reasonably small total difficulty. OK, are we understanding the problem?

So we have three possible avenues here. We can start solving it using dynamic programming. We can start solving it using graphs. Or we can have someone else play the DDR again so that we can get more hands on experience, and look at what a good strategy or a bad strategy looks like.

So you guys get to vote. Who wants to solve this using the dynamic programming? Who wants to solve this using graphs? Who wants to get more hands on experience? Damn. I was hoping that DDR would win. You guys are boring.

So, graphs, right? We're going to have notes. Notes represent states, right? So a state is where you are at some point in the game. And we'll have to define what where you are means.

And then, when you jump, you make a move. So moves go between states. So in any game, you have states and moves. A state is very hard to put in time. And then you make a move.

Like in chess. Your state is your board position. And whether you're moving, or your opponent is the next to move. And the move is when you take a piece and put it from one place to another.

So your vertices are states. And your edges are moves. So what's in a state? This is our problem. Whether we're solving it with graphs or with dynamic programming, this is what everything comes down to. What's in the state?

AUDIENCE: So your current state is how much difficulty have you-- what is the problem? Are those representing the probabilities, the difficulties? Or.

PROFESSOR: So this is the difficulty for one move. Does it depend on previous moves?

AUDIENCE: No, but.

PROFESSOR: No. So it's a nice thing to note that the difficulties for each move are independent.

AUDIENCE: Would it be dependent in real life? You'll get tired at the end. And he's able to have less energy.

PROFESSOR: Yep. So, in real life, we might want to say that this actually depends on how much energy you've expended so far. And if we have time, let's solve that problem. So the nice thing about this problem is there are many directions in which we can take it. This is one of them.

I'm fine. And I think we'll learn a lot by doing that. So back to the original problem.

Suppose that it doesn't matter how tired you are. You're pretty good, so you're not going to be tired enough to go [GASP]. So, mostly independent difficulties for each move. What's in a state?

AUDIENCE: The sum of the difficulties up to that point.

PROFESSOR: OK. So.

AUDIENCE: Your appearance factor, if we're including that. Or are we only doing--

PROFESSOR: So we're doing this.

AUDIENCE: Oh, I was just. Oh, effort. OK, we're minimizing effort.

PROFESSOR: So the way I like to think of this is, what is does a solution look like? It's made up of decisions, right? What are the decisions?

AUDIENCE: To make that move or don't make that move?

PROFESSOR: We'll see. But once you know what the decisions are, then you have to think, what do I need to make a decision?

AUDIENCE: The note.

PROFESSOR: So, a note is one of these. For some reason, they're called notes.

AUDIENCE: So you know that somehow you have to hit all of them? Like, you can have steps in between.

PROFESSOR: Yep.

AUDIENCE: Moves in between. You have to hit all of them.

PROFESSOR: OK, so I like this. So, every one of these is going to have to generate a move. And there might be some moves in between. Well, each move is going to map to some edge. And I'm going to need vertices.

So, as an algorithms program where when I hear that, oh. I can have some moves in between, I get really uneasy. Like, what is this? How many moves am I going to have? Is this going to become really huge? And am I not going to be able to run any algorithm on it?

So let's think about that. How many moves would I have in between? And, what would I achieve with them?

AUDIENCE: I guess you'd have a max of, like, four, right? I mean, you can put your right leg someplace, and you can put your left leg someplace. That's like a max of two. Right? Or no?

PROFESSOR: So, let's think of an example. You want to go, say, I want to go, what? From here to here, right?

AUDIENCE: Um-hmm.

PROFESSOR: And you're thinking what? They can do this, oh sorry. So I can do this as one move. And this is one move.

AUDIENCE: Yeah. Or you can just do one jump, right? So it's-- oh, yeah. I guess that's more than one.

PROFESSOR: OK. So looking at this, if I'm considering whether to go from here to here. If I want to go either this way, or if I want to go this way. So let's write this down on the board.

So I'm going from up down to left right. And I'm considering, do I want to go directly or do I want to go up right first. And then left right. Let's not do these arrows because they're confusing.

So, do I want to transition like this? Or do I want to transition like this? Well, I claim that, since my moves are independent, going from here to here, if it's beneficial to go this way instead of this way, I would want to go this way all the time.

So I would just say that, look. Inside my delta here, I would say that, actually, if you know what you're doing, whenever you have to go from here to here, you're going to go like this. And the delta is going to report to the total delta for this. So, if the problem is that they have one note, and then I have my next note like this, between these two notes, I can always go in one move.

AUDIENCE: So you're saying 1 would like, including in our difficulty function, it'll also tell us, like, what move we should make to get there?

PROFESSOR: How to get there. The best way to get from one state to another. Well, like when you're learning DDR, these are the basic steps, right? Like, you practice until you know how to get from one place to another. There is something else where you need intermediate steps.

AUDIENCE: Or if you're holding one and then you've got to move to another. Right?

PROFESSOR: So, let's not worry about holds. Let's say that we would just represent holds as-- so if you have a hold in a game, then we're just going to represent it like this. We're going to cheat. So we don't have to deal with it. In real life, you would have to deal with it. Like if you're actually writing a program that trains people.

AUDIENCE: What happens if you have, like, right, left, up or something. So let's say you start off hitting your right foot on the top, and then you have to hit the right one again. And

then you have to hit up again and then you have to transfer. So I guess anytime you have to through the middle. You have move your foot.

PROFESSOR: So I like the idea of going through the middle. Why or how do you, why would you go through the middle?

AUDIENCE: To avoid crossing your leg over or something.

PROFESSOR: Well, what if I have something like this? If there are DDR experts, I know this isn't the right way to do it. But one way of doing it is left, right, left, right, left, right. So then, I have some states that account for the fact that I'm centering.

So, some people like to do this, right? Some people like to go to the center as much as possible, if they have time between the moves. So then, I would need at least one state in between where I'm allowed to center, right? In order to represent what I just did there, I need one move.

So, one way I could meet this is I have left center. And then left right. And then left right, left right, left right. So on and so forth.

But this would mean that instead of doing this, which is reasonably easy, I would have to do this. This. And then jump, jump, jump, jump every time. Harder. Not good. Not good for my knees. Not good for my score.

So I don't want to do this. I want to have an intermediate state so I can say, I start here and then I get back here. Then I do this. Then I get back here. Then I go here. And then I go back here. And then go back. Do guys see how this works?

AUDIENCE: You could also rock back and forth to because they don't require both feet to be on the ground the entire time.

PROFESSOR: Rock back and forth. Hold that thought. That might come in handy. Assuming we have enough time. So, if we only do one move every time we see a note, we're stuck with this. In our to allow re-centering, we have to add states between the

notes.

We need to add an intermediary state. And you can either think about it, or take my word for it, but if you represent the notes using the trick that I said above, and you want to represent centering, then all you need is one extra state. So one extra move between every two states. Between two notes.

And that's enough. Because that one extra state allows you to center. And it allows you to do pretty much everything a beginner like me would know how to do.

AUDIENCE: So you only need one extra state between each note?

PROFESSOR: Yep. That is the inside. That is where I'm thinking, right? So I have to do both this trick, to see that I don't need an infinite number of steps. And come up with the need for centering to see why I need one intermediate step.

By the way, does anyone know what's the right way of doing that? What's the right way of doing this move? You said, rock back and forth. So I wouldn't rock back and forth in this case, right? Because that wouldn't be helpful. How would I rock?

AUDIENCE: Well, you hold down one and then you go to the other one.

PROFESSOR: Yeah, so rock left and right. OK.

AUDIENCE: I said back and forth, I meant rocking--

PROFESSOR: Sorry. I thought back, forth. So, the optimal way of doing this is left, right, left, right, left, right. So what am I doing?

AUDIENCE: You have another move. You have, like, one state for your foot where it's not touching the ground.

PROFESSOR: OK. It's an interesting, and important, distinction. So if I wanted to hit both of them, if I'm here, and I need to hit both of them, I need to hit them. If I only need to hit one of them, if I'm here, I can hit it like this. If I'm here, though, and I want to hit it, I have to lift myself up.

So the difference is whether I'm like this. Or like this. Where is my weight? So if I want to allow for these moves, if I want to go past the stage of very beginner, aside from keeping track of where my feet are, I have to keep track of whether my weight is on my left foot or on my right foot. So, I would actually have l, r, and W, where W is either l or r.

AUDIENCE: So your state now has what you're currently putting more weight on.

PROFESSOR: And where each of the feet are.

AUDIENCE: Yeah. Where your two feet are. And then also, the song difficulty. So it's everything in your state, right?

PROFESSOR: Ah, yes. I didn't think about it. I was just talking about foot position. But we need to get back to the state thing. So that's a good point. So before we get to that, how many feet positions do we have? If we don't include where do I have my weight, how many possible positions for both of my feet do I have?

So I have two feet. Each of them can be in any of these five squares, right? So how many total positions?

AUDIENCE: 5 choose 2. Using 2.

PROFESSOR: So I can do this.

AUDIENCE: Sure. Why not?

PROFESSOR: Well, if I have 5 choose 2, then you--

AUDIENCE: Oh, then that doesn't quite work.

PROFESSOR: Yeah.

AUDIENCE: Yeah, no.

PROFESSOR: So flip.

AUDIENCE: Times 2.

PROFESSOR: Yeah. So you have two feet. Each foot, five possibilities. Now, what if we have weight? If we are tracking on where I keep my weight. How many possible positions?

AUDIENCE: [INAUDIBLE]? Oh no, because there's only two.

PROFESSOR: So, for each of these, there's two possibilities now. So 50.

AUDIENCE: You also have to be so your weight is like this.

PROFESSOR: In the middle?

AUDIENCE: Yeah.

PROFESSOR: You can. Based on that what I've been reading, you never want to. Because if you're weight is in the middle, then you're extending a lot of effort to move either foot. So you always want to have your weight somewhere. So that, at least for one of your feet, it's easy to move it.

AUDIENCE: Could it be the back?

PROFESSOR: If that's not true, then yeah. We'd need to have a center. Oh, so, like, whether I'm like this or like this?

AUDIENCE: Yeah. If you were, like, doing the same thing. Rocking back and forth.

PROFESSOR: So, if I have two feet, then this says my weight is on my left foot. My weight is on my right foot. So l and r is which foot, not where on the board.

AUDIENCE: So, did you not do center because it was optimal?

PROFESSOR: Because I claim that, in an optimal strategy, you wouldn't have it. But you don't have to take my word for it. If you don't believe me, then you add center. And you just have more possibilities. And if it happens to not be an optimal thing, then the dynamic programming will ignore it.

So how many positions do I have if I add center? So if I have left, center, right for each position.

AUDIENCE: 75.

PROFESSOR: Cool. Yes.

AUDIENCE: So why is it 25? You wouldn't actually have both of your feet on the right.

PROFESSOR: Why? Maybe, well especially, for maximizing this one actually. Especially for maximizing entertainment. Then it's way cooler to do this, right?

[LAUGHTER]

You get the point.

AUDIENCE: --possible weight. Like combinations for like, where is you should have your feet at one time. Like, not necessarily what the game is asking you to do.

PROFESSOR: Yeah. So this is where my feet are. The game will always ask something like this. So the game won't always care about both of my feet. Sometimes it'll only care about one foot.

So, yeah. There are two different concepts, right? There's the position of my feet. And then there's the note on the screen. And my feet have to match the note on the screen. And we'll have to capture that somehow.

So what's in a state? What are my decisions? Come on, now. We should be in a good shape to know what my decisions are. What do I decide every time?

AUDIENCE: What move to make?

PROFESSOR: Yeah. How am I going to jump, right? Where my feet are going to be in the new position. So every time, I'm deciding what's the new foot position.

AUDIENCE: Aren't you also deciding weight position, where you're going to hold to, right?

PROFESSOR: OK. So when I say foot position, it's a cheat for everything. So yeah. Good observation. OK. What do I need to know, in order to make this decision?

OK. I would need to know where I was before if I want to compute the difficulties. But aside from that-- what?

AUDIENCE: Sir, it came up with this polynomial.

PROFESSOR: All right.

AUDIENCE: Is that a hint to what the run time of this will be?

PROFESSOR: It's not on purpose. But yes. So, yeah. That makes sense. Probably not good. I should probably not have my screen up without me looking at it. OK, so. Back to decisions.

I think I want to know what's going to be on the screen when I land, right? Because if the screen says, yo, you need to go up and down. Then maybe I shouldn't do this, right? That would not be good. So I need to know what the note is going to be when I land. Note.

AUDIENCE: Is that the same as the next?

PROFESSOR: Yep. Next note. Yes. So the next note here on the screen. So, my position in this list, basically.

AUDIENCE: So it's kind of like the knapsack problem, where it's like which thing we're considering picking up. But in this case, it's coming [INAUDIBLE].

PROFESSOR: So it's close to knapsack. But the difference is, in knapsack, I have to decide, do I choose this or do I ignore it. So, if I'd be like, if I can only do a few things and I want to maximize my score, then maybe I would choose, all right.

This is easy. I'm going to do it. This is hard. Screw it. This is easy. I'm going to do it. So then my decisions are 0, 1. Pick or not pick. In this case, every time my decision is a new state of my feet. So it's not 0, 1.

AUDIENCE: OK.

PROFESSOR: That's the difference. And then, because of that, the state is going to look different. And the recursion's will look a bit different.

OK. So I want to know the note that I'm landing at. I want to know the sum of-- well, we'll see if I want to know the sum of difficulties up to this point. And I want to know one more thing, so.

If this is going to be my new state, and I know what note I'm landing at, I'm making a decision. The decision tells me where my feet are going to be. Right?

The move decides where my feet are going to be. What do I need to do? What else do I need to know for my total solution?

So what do I need to know to consider between decisions? So say I'm considering the state of-- I'm at note two. Note two says, left right. And I'm considering of coming here from note one, where my foot position was up down. Or come here from note one, where my foot position was left right.

I need to know where my foot position was, to choose between these, right? If I don't know that, then I can't choose.

AUDIENCE: That's not from your state, though?

PROFESSOR: Not yet. It's not here, so it's not part of my state. We've been talking about it a lot, but we didn't put it here. So let's say that the state is going to have the note that I'm landing at. And my foot position after landing.

So, if I know that I'm going to land like this, then I know where I was here. l r. Say, l r. I can compute the costs here, right? This is delta of going from this to this. And this is the delta for going from this to this. Can you see? Is that too small?

Delta wants to know where my feet were before. And where they're going to be now. So when I call it, I need to know where I was before. And I need to know where

I'm going to be now. So this should be part of my state.

So our decision says, what's my new position going to be? So when I make a move, I influence my new foot position. They're going to be in someplace.

And independently of that, the note-- the position here-- the position of the note, increases all the time, right? Because time can only go forwards. How are we going to account for these extra moves?

AUDIENCE: [INAUDIBLE].

PROFESSOR: So, we said that between every two notes, we might want to make one extra move.

AUDIENCE: Like, a potential new note in between your destination note?

PROFESSOR: Yep.

AUDIENCE: And then just fill in the difficulties for that and see which ones--

PROFESSOR: So, if I want to use the algorithm that I already have, that compares my landing position to my note, to see whether I can go there or not, what note would I want to put in that intermediate note? So, you're basically saying that between every two notes, I'm going to have one more note. And my feet can be wherever here.

AUDIENCE: Oh. I mean, you'll also have the edge originally connecting them. So, you'd have it going to the intermediate note, but also from the destination. Yeah, like that. You'd have to--

PROFESSOR: So, what note would I have here?

AUDIENCE: It makes you go to the center.

PROFESSOR: Do I have to be at the center?

AUDIENCE: Well, you can be anywhere.

PROFESSOR: Yeah. So I'm going to invent the blank note. That's a big O. That means you can do whatever you want. There are no constraints. So that is just an intermediary state.

So then I'm going to take this input, and I'm going to add these blank notes here. And these map to adding notes in the graph.

AUDIENCE: Well, then, when you just run through your difficulty function between each two notes and see if, for all the potential moves, which one minimizes that path.

PROFESSOR: Yep. So basically, I'm assuming I don't need this. I'm assuming that. This is going to be where there's at least one choice where this is going to have the same cost as a direct note. And then I'm going to say that instead of having more complex shapes in the graph, I'm going to insert extra moves here.

AUDIENCE: OK. I would think that, if you have an intermediate move that might be less expensive, you should, like, just hop to that new position than to have an intermediate move, right?

PROFESSOR: Yeah. And I'm going to assume that either that's not the case, or that I can always find an intermediate move that makes my life better.

AUDIENCE: Every case?

PROFESSOR: Yeah.

PROFESSOR: These results do work. You'll have to take my word for it that this works. And this makes your dynamic programming easy. And it makes your graph building easy. So the advantage of this is you're changing the input. And then, when you're building your graph or when you're doing the DP, you only worry about an input that looks like this.

You don't care about any intermediate stuff. You have notes, and you have to transition between the notes. Period. So we reduce the complexity of the problem.

So whenever you can get away with that, it's really nice. We can do that for centering. We might not be able to do it for other more complicated stuff. OK.

So, we sort of know what we want in a state. Now we want to build a graph, so that

the path from some source to some destination has a cost that's equivalent to the sum of these difficulties. Right? Because then I can run the shortest path. And my solution will be right there.

I know. You guys seem awfully sad. Do you want to take a break and play for a little bit more? Yes? How many people want to play, instead of. You don't have to play yourself.

You can just rest for these two or three minutes. And you can watch someone else play. And get more intuition about how you're supposed to play. OK. So one person wants to take a break. Everyone else wants to keep going? Or are you guys passed out already?

AUDIENCE: Can we get out early?

PROFESSOR: Sort of. Yeah. You'll get out three minutes early, if we don't do another round. So who wants to keep going?

AUDIENCE: No. Let's play.

PROFESSOR: OK. So who wants to keep going? You can only vote on one thing. Who wants to play?

AUDIENCE: OK.

AUDIENCE: Five.

AUDIENCE: Four.

AUDIENCE: Victor, go.

PROFESSOR: OK. Who wants to go? So we decided what we're going to have in notes. And we're going to build a graph, where the notes are the states. And the edges are the moves. Right?

So a note tells me, just like there, what note I'm at. Say I'm at note two. And where

my feet are on. So, left foot left. Right foot right.

So, given a state, what outgoing edges do I build? Where can I go to? So let's say I have a state that says I'm at note n and my left foot is in some position. My right foot is in some position. What are my outgoing edges from here?

AUDIENCE: Is there an outgoing edge to every move? Because there's a point.

PROFESSOR: OK. Well, actually, the blank is a note, right? So if the next note, if note three's a blank, then the note is a blank. And then I can have.

AUDIENCE: I guess, isn't the difficulty of the move dependent on what note that blank actually is?

PROFESSOR: So, it doesn't depend on what the screen says. It depends on where my feet are.

AUDIENCE: Oh, OK. I see.

PROFESSOR: So I have a number that tells me what's on my screen. This number is enough for me to know what's going to be on my screen from now on until forever. Because I have the list ahead of time.

And then I have the position of my feet. So these are both part of the state. So I have notes for all the notes. And all the positions of the feet.

AUDIENCE: Right. But the blank note is just one position of feet?

PROFESSOR: No.

AUDIENCE: Or, it's all positions.

PROFESSOR: Yeah.

AUDIENCE: Isn't the difficulty dependent on what position your feet end up?

PROFESSOR: Yeah. So it's from where I start to where I land. So, if I decide to start from here, and go here, the difficulty of this move is-- so it doesn't depend on whether I have a blank there.

And this was OK. Or, if I had an up down, then I just missed the note. But the difficulty, the effort I'm expending, is the same.

AUDIENCE: OK. But if you start from that other move and go to a different position, that's--

PROFESSOR: If I go from this to this, then that's a different difficulty.

AUDIENCE: But it still could be a blank note?

PROFESSOR: Yeah. So, a blank note will let me go wherever I want.

AUDIENCE: I see. But there are different kinds of blank notes, with different-- I guess I'm confused as to what--

PROFESSOR: OK. So let's go through them again.

AUDIENCE: Is it kind of like it's, like, spreading, so really when you have your source note, it's actually connecting to every possible move. And every possible move is connecting to your next-- so it's kind of like a diamond shaped thing?

PROFESSOR: Not quite. Not quite. So, let's go through the notes again. And then let's go through that. So the source move will be connected to some notes. And then those notes will be connected to other notes.

And it looks more like you have a big, outgoing degree. Then you have a big network. Then the big, incoming degree. It looks like a messed up sorting network.

So, notes are constraints. This accepts any possible feet combination, right? So this accepts this.

AUDIENCE: Is the first combination contained in that note? In that blank note?

PROFESSOR: No. The blank is a constraint. So, if I have a note that looks like this, then if one of my feet is here and the other one is in the center, then that's fine. The constraint is met. If one of my feet is up and the other one is down, then the constraint is met.

AUDIENCE: But the edge running into that blank note, depending on how your feet land, is

different difficulty. Is that right?

PROFESSOR: Yep. But when I compute the edge width, I need to know where I'm landing exactly. Not just where the note is because the note might admit multiple positions.

AUDIENCE: I see.

PROFESSOR: So this guy admits all positions.

AUDIENCE: OK.

PROFESSOR: This guy admits some positions, right? These are l r, l r. If I have up down, how many feet positions does it admit?

AUDIENCE: Just one.

PROFESSOR: Not great.

AUDIENCE: No.

PROFESSOR: How many?

AUDIENCE: As an intermediate?

PROFESSOR: If it's up down, then it's on the screen. My intermediates are all blank.

AUDIENCE: Oh, oh. I thought you were saying start with that.

PROFESSOR: So up down. If you see up down on the screen, how many feet positions?

AUDIENCE: Left to right what it could have.

PROFESSOR: Yep. So, if the screen says up down, the only two ways I can do that is this. Or this. Right? So, two positions.

AUDIENCE: I see.

AUDIENCE: Well, I mean, but you could have intermediate where you step in the first one, then back.

PROFESSOR: But I don't care because I've added these blank notes. And they take care of my intermediates for me.

AUDIENCE: But so, we're saying where the feet land aren't part of the note.

PROFESSOR: Nope. They're just states. So they're a decision I'm making.

AUDIENCE: So, do we have the multiple-- no, we don't because they're not part of the--

PROFESSOR: For each note position here, for every position in the screen thing, I have multiple feet positions that I could be in.

AUDIENCE: Right. Right. But then, are there multiple moves to get to those?

PROFESSOR: Yep.

AUDIENCE: OK. So there's multiple moves connecting each.

PROFESSOR: Yeah. So for example, if I want to get to l r, up down here, I can go from l r, up down in the blank. Or, I could have l r be like this. And then transition. So I can have a lot of possible foot positions that I'm transitioning from in order to get here.

AUDIENCE: But those aren't actually separate notes on the graph.

PROFESSOR: They are. They have to be. So these all have to be separate notes. Because the edges are going to have separate weights. Right? This edge is going to have one weight. This edge is going to have another weight.

AUDIENCE: OK. That's what I thought. But then you said there was only one blank. Don't there have to be multiple blank notes for every foot thing. If you're approaching a note from different--

PROFESSOR: So, I'm not sure what you're asking. So, if you want to care about centering, if you want to have intermediate moves, you need one blank note between every real note. So that you're allowed to make one intermediate move between.

AUDIENCE: Right. I get that that's talking about the on the screen.

PROFESSOR: Yeah. That's the screen.

AUDIENCE: That's the screen.

PROFESSOR: Yep.

AUDIENCE: OK. So, that's not the graph, though. Here, we're talking about the screen on the graph.

PROFESSOR: Here, we're talking about the graph. So two tells me that the screen is at note two. Whatever that happens to be. And my feet are in this position. OK. So that's very important. Thanks for asking because that's very important. So, a state has the note that I'm landing at. So, where the screen is. And where my feet are.

AUDIENCE: OK.

PROFESSOR: OK.

AUDIENCE: The state corresponds to the nodes that you're on.

PROFESSOR: A note. A note is a state. So, a note is a state. And a move goes between two states. This is how we represent all the graph problems. OK.

AUDIENCE: So, you need multiple blank.

PROFESSOR: Yeah. So here, I'm just drawing one example. Yeah. Yeah. So for example, here I would need-- sorry. Oh, OK. I get your confusion now. So, yeah. This would be one move. This would be another move.

AUDIENCE: That's all for a single thing on the screen.

PROFESSOR: Yep.

AUDIENCE: OK. OK. That makes sense.

AUDIENCE: So the next note would have a really large end degree, right? I guess they always

have--

AUDIENCE: Well, no. It'd have lots of--

PROFESSOR: So, this note has a huge out degree.

AUDIENCE: Yeah. But I'm saying the ones that all those intermediates connect to.

PROFESSOR: Yeah.

AUDIENCE: So, then, they would connect with some note that have, like, really large degree.

PROFESSOR: Yep.

AUDIENCE: But there's two of them after that, right? Because there would be two different--

PROFESSOR: So, what's next? Next, is position, hmm. How did U number them? I guess, crap. No. It's not going to match that, so. Whatever we do, it's not going to match that. Say the next note, 4, is just an up.

4 is up down. And then I have two notes for it, right? Wild One note, l r. So this is note 4. That happens to be up down. And then I have this. And every one of these connects to every one of these.

AUDIENCE: If the next move is up down, how can notes have down up? Oh, because you can switch the feet.

PROFESSOR: Yep.

AUDIENCE: Oh.

PROFESSOR: Yeah. So, I aimed to ask you guys, how am I going to draw the edges? But you guys made me do it with questions. So, suppose I met note n, l, r. What edges do I draw going out of it?

So, note n, left foot at some position. Right foot at some position. What are my outgoing edges?

AUDIENCE: This is a note on the screen. Or--

PROFESSOR: In the graph.

AUDIENCE: --the new one or intermediate one?

PROFESSOR: Doesn't matter. So, intermediate notes have a note blank.

AUDIENCE: That's just everything.

PROFESSOR: It doesn't matter. Yeah, we're looking at all of them at the same time.

AUDIENCE: Wait, so you're asking what are--

PROFESSOR: So, given this note, who am I connecting it to? Let's draw the outgoing edges. So, if you're solving this as a graph problem, you build the notes. You draw the edges. You're on some algorithm. So let's draw the edges.

AUDIENCE: I think it depends if you're an intermediate note or not.

PROFESSOR: Nope. So the note tells you what your constraints are. And that covers intermediate notes. If you're note is a blank, then, well, there's no constraints. If the note is not a blank, then there are constraints.

AUDIENCE: Oh, note. I thought that was referring to the note, not the--

PROFESSOR: No. So this is a note. Maybe n is not good. Let's use i, instead, then. If n is hard. So, i is the position here, in this guy. Right?

Say for example, i equals 1 here. I equals 2 here. I equals 3 here. So and so forth. And that doesn't match this, so.

AUDIENCE: We're talking about note--

PROFESSOR: Yep.

AUDIENCE: --not--

PROFESSOR: If one note is represented by these three numbers, right? Each note has this tuple in it.

AUDIENCE: Then i is the note.

PROFESSOR: Yep.

AUDIENCE: So i connects to i plus 1. And then all possibilities known.

PROFESSOR: OK. So this connects to i plus 1. And then all possible destinations, right? So I'm going to say left. This is after the jump. So, left after the jump, and right after the jump. And this is for all possible left after a jump, right after the jump. And lost a weight on this edge.

AUDIENCE: The way that it's from--

AUDIENCE: l r to

AUDIENCE: --l r to l r.

AUDIENCE: l j-- l sub j r.

AUDIENCE: l r.

PROFESSOR: So you want to say from l r to l j, r j. But what?

AUDIENCE: Difficulty level?

PROFESSOR: Yeah. Where is that?

AUDIENCE: Your difficulty function.

PROFESSOR: Which is?

AUDIENCE: Delta.

PROFESSOR: Delta. So, yeah. The weight is delta. I drew d. Delta of these two foot positions. Almost. If I want to minimize the difficulty, then this is right.

If I want to maximize a score, then I would have to add a minus sign. But if you're minimizing the difficulty, this is exactly the way because it maps the shortest path. Usually, in dynamic programming, things are maximized.

AUDIENCE: Wait, why would you add a minus if it's--

PROFESSOR: So, if we're maximizing something, we have to flip-- we have to add a minus to the edge costs because--

AUDIENCE: Right.

PROFESSOR: Yeah. But this time we're minimizing something.

AUDIENCE: Oh. Right, right, right.

PROFESSOR: I promised you that we'll maximize something all the time. Well, I guess I lied.

AUDIENCE: OK. Right, right, right.

AUDIENCE: You don't want to maximize d. Then you're dead.

PROFESSOR: No. You want to maximize show offs. So if we have the entertainment thing, then that would be maximized. And then you'd have minus.

AUDIENCE: Then you'd have everything you need.

PROFESSOR: Yep.

AUDIENCE: OK.

PROFESSOR: OK, so we're almost good. Source notes and destination notes.

AUDIENCE: Well, I guess source, we know that means start in the middle. Right? Like, you would start off with--

PROFESSOR: What's the source? You want to start at the first note, right?

AUDIENCE: Yeah.

PROFESSOR: But in which position?

AUDIENCE: Well, in the center, right? I mean, the [INAUDIBLE] start. Or, I guess you could start--

AUDIENCE: Wherever you want.

AUDIENCE: Could you add a source note going to a bunch of intermediate notes?

PROFESSOR: Yeah. Yeah. So what I'm going to do is I'm going to say there is a note 0. That let's me choose how I'm going to start.

AUDIENCE: Oh, it could be any position. It's like an intermediate note itself.

PROFESSOR: Yep. So I'm going to have a note 0, with all possible foot positions. And then this is going to be connected to a source, by edges of weight 0. So the intuition is that I get to choose my starting position, right?

Starting position. I see the first note. I make a move. So I want to be able to choose the starting position. That's why I have all those notes. And then make a move to touch the first note.

AUDIENCE: The starting position is also a note.

PROFESSOR: Yep. Because, if the first note is here, I don't necessarily want to start like this because maybe this is harder than this. All right. What's the destination?

AUDIENCE: The last note.

PROFESSOR: OK. So, all the notes that are at note n, and have any position, would be connected to a destination note, using an edge of weight 0. Or I can consider them as destinations. And then choose the destination that has the shortest path.

AUDIENCE: Oh, I see. The final destination is going to be, like, two notes. Right? It's going to be that last position, and then--

PROFESSOR: Or it might be more than two if the note is just a simple note. Because that wouldn't constrain my feet too much. OK. And last, last question. Dijkstra or Bellman-Ford?

AUDIENCE: Bellman-Ford, because you have negative edges, negative weight, as if you were going to maximize to death.

PROFESSOR: So, this is always going to be from 0 to 1. So I could possibly use Dijkstra. The answer is neither. The answer is, we're building a DAG, so.

AUDIENCE: Dijkstra.

AUDIENCE: We don't use that.

PROFESSOR: DAG, shortest path. Every time we have dynamic programming, DAG, shortest path. OK?

AUDIENCE: Question.

PROFESSOR: Yeah. Come on, last question. You can't get away that easily. OK, so. Who wants to play?