SRINI DEVADAS: Let's get started. So today, we're going to look at one of my favorite puzzles. I'll say right at the beginning, that the coding associated with the puzzle is fairly straightforward. But the analysis associated with this puzzle is quite intricate.

And in particular, we're going to have to do worst-case analysis, which is really a good thing for you to learn about, because asymptotic analysis in algorithms is all about worst case. All right. So keep that in mind. When you try to come up with the answers to my questions, always keep in mind that I'm going to be asking for what happens in the worst case.

In the context of our puzzle-- which is called the crystal ball puzzle, and it's fairly popular in literature, though I've generalized it a little bit-- you have a situation where you're supposed to figure out the hardness coefficient of a crystal ball. And in the generalized version, you have a set of identical crystal balls.

So how do you find this hardness coefficient? Well, you go up to the Shanghai Tower that has 128 floors-- 1 through 128-- and you drop this ball from say, floor 1. So just you're standing here and let's say, you just drop it from there. Or you go up to floor 2, and you drop the ball.

And the hardness coefficient is defined as the highest floor from which you can drop the ball and the ball does not break. So what this means is if at floor f, the ball doesn't break, and it breaks at f plus 1, then the hardness coefficient is f.

And you can assume a monotonicity property that says that if it did not break at floor f, then it won't break-- this implies that it won't break at any floor less than or equal to f. Which makes sense, right? And otherwise, it would be kind of complicated. It would be random. And we won't be able to do any interesting analysis.

So the easy version of the puzzle, of course, which is not particularly interesting, is if you needed to get this exactly-- so your job depends on getting that number exactly right-- so there's no shortcuts. You've got to get it right. You can't be off by one either way. And you have only one ball. So you only have one ball. So what can you do if you only have one ball and you have to get it exactly right?

AUDIENCE: Just try every floor.

SRINI DEVADAS: Try every floor. Right. And but more important, try it from lower floors to higher floors, right? So you drop it at floor 1. So one ball, you drop from floor 1. And if it breaks, then your hardness coefficient is, you return 0.

But then if you keep going and let's say, you drop off dot, dot, dot-- of from floor 65 and it breaks, and you've gone in order, then the hardness coefficient is 64. So that's easy enough. There's no optimization here. It's the only thing you can do. So that's why it's not particularly interesting.

Now if you have identical crystal balls that are guaranteed, they were manufactured in exactly the same way, et cetera, et cetera, then obviously, things get a lot more interesting, because you can try to minimize something. And I'm going to write it over here, because I want you to keep this in mind throughout the lecture. Our goal here is to minimize the worst-case number of drops for our problem.

And we want to solve the problem in the general case with d balls. And I should say, d identical balls. But more important, the reason I'm writing this up is, as I said before, we need to focus on the worst case. So it may be the case that for a given set of balls, your algorithm does very well, in the sense that the hardness coefficient happened to be 42. And you picked the floor as of 41 to begin with. And so you dropped from 41, and it doesn't break. And then you go to 42, and it doesn't break. And you go to 43, and it breaks. And you go, oh, I only needed two drops for this particular problem.

But if the hardness coefficient happened to be 65 and you started from floor 41, I mean you'd need 20-odd number of drops. So you have to be careful in that you have to analyze whatever strategy you come up with in the case where the hardness coefficient could be pretty much anything. I mean it could be 0 or it could be 128 in our Shanghai Tower example.

So the first interesting question is, what happens when I have two identical balls? So, oh, before we do that, according to my metric here, what is the worst-case number of drops for one ball? What is the worst-case number of drops for one ball? Give me a number. Someone else. Yeah, back there.

AUDIENCE: 128.

SRINI DEVADAS: 128, exactly right. So it's 128, because I could have had a ball that doesn't break from the 128 floor. I don't know that to begin with. I have to start with floor 1. And I got to go 1, 2, 3, 4, et

cetera, right? So the worst case for one ball is 128. And you can't do better than that. But of course, it gets more interesting when d is large.

So our goal now is to discover an algorithm that is going to minimize the worst-case number of drops for d equals 2. That's our current goal, right? And then we'll look at more, even more general things. So how would you do things if you had d equals 2? Just, it doesn't matter if you don't get our algorithm right. We'll analyze it. You propose something. We'll analyze it. And then we'll decide if we can do better or not. Yeah. Go ahead.

AUDIENCE: Drop the ball from the halfway point, say 64. If it breaks there, then you can use the bottom half of what we are using.

SRINI DEVADAS: Use the bottom half. Excellent. What's your name?

AUDIENCE: Dhaman.

SRINI DEVADAS: Sorry.

AUDIENCE: Dhaman.

SRINI DEVADAS: Dhaman. So Dhaman says that we should do binary search, which is very intuitive and kind of makes sense in a lot of context, including this one. And so let's drop from 64. And the reason she picked that is because 128 divided by 2 is 64. And if it breaks, then you need to look at the interval, 1, 63 because if it-- I'm sorry. Yeah, that's right.

If it breaks, you need to look at the interval 1, 63 with a second ball. I mean that first ball is gone. It's shards of crystal or whatever you want to call it. So you just have the second ball. And so now at this point, you have no choice. You only have one ball left. As I mentioned, your job depends on getting that hardness coefficient right, so you can only use that first algorithm in terms of starting from 1 and going all the way to 63.

And so what is the worst-case number of drops given that Dhaman picked the midpoint for this algorithm? Someone else. Yeah. Go ahead.

AUDIENCE: It's 63.

SRINI DEVADAS: It's 63. No actually, you're off by 1. Because I did drop the ball here, so close. So the first ball was dropped once, one drop. And this one, if in fact, the hardness coefficient was 64, you drop

the second ball 63 times. And that would be the worst case in this instance.

Now, I've skipped over one little thing. The fact that this is the midpoint implies that this is, in fact, the correct answer. But we do have to analyze. I mean remember, you know, when you do analysis, you do have to cover the entire spectrum here.

So you do have to look at the case where the drop from 64 does not break. And in that case, you're looking at 65 through 128. But you do have two balls for this. It didn't break. So you still have two balls for that.

And so you can imagine that, now that you have two balls and you have a small interval, that you're going to be able to do something like 32 for that. The bottom line is, you don't even need to. You can get a little bit lazy here. You don't really need to analyze what the number is, because you know that this number, 63 plus 1, is larger than that-- because you wanted worst case, right?

So this is the kind of thing that you'll have to do when you do algorithmic analysis. Maybe not exactly in a puzzle situation like this, but you have to look at all the cases. And you have to find kind of the max of all the cases. So is this the best you can do? Yeah. Go ahead.

AUDIENCE: You could break the 128 into smaller sections using the first ball. So instead of breaking it in half, you could do it every 16 floors.

SRINI DEVADAS: Every 16 floors. Yeah.

AUDIENCE: Starting from the bottom, and then once the first ball breaks, then you have a smaller interval to test with the second ball. So let's say it breaks at the 32nd. Then you have to only test for 16 to 32, instead of 1 to 32.

SRINI DEVADAS: That's correct. Yeah. So it shows how you can do it-- I mean, this is, obviously there's not going to be a whole lot of optimization here with d equals 2. But you'd be surprised as to how much better you can do than 64, which is obviously a big step forward from 128. But you can go a few more steps as well. And so what was your name?

AUDIENCE: Lucy.

SRINI DEVADAS: Lucy. So Lucy has it right. It's a similar idea, except you don't necessarily need to do the intuitive binary search. It's actually kind of different from that because of the way this problem

is structured. And what her idea is is that we'll look at something smaller, a smaller interval.

And she picked 16, so let's go 16, 32 is our algorithm. It's something that goes 16, 32, 48, 64, dot, dot, dot. And maybe gets up all the way to I guess, yeah 128 and 16. So 128 is a multiple of 16, right? Some math person make sure I'm right.

So what happens here? So let's analyze this to make sure that we're good in terms of an improvement. We have to be careful. So let's just arbitrarily say that it broke at 64, because that's kind of what happened over in that previous algorithm too. So that means I've done four drops without a break. I'm sorry, four drops. Three drops without a break and four drops total. So I've done four drops.

But now the cool thing is I don't have to start from 16. I know it didn't break at 48. So I can now look at four drops till break of first ball. That is what I wanted to say. And now, I can look at the interval 49. Because 48, it did not break-- 49 to 63, inclusive. And so if you do that, that's 15. It's 15. The difference is 14, but it's inclusive, so it's 15. So 15 plus 4, that is 19 drops. Yeah. Back there.

AUDIENCE: Wouldn't the worst case though be when it's at 127.

SRINI DEVADAS: Yes, that's absolutely correct. So 19 is not the worst case. I mean it's looking good, but this was exactly the point I was trying to make right from the beginning. You are not done yet. I mean, you cannot just go off and say that 19 is the worst case for this algorithm.

The worst case for this algorithm-- this particular algorithm occurs at a different point from the original algorithm that went midway. Every algorithm is different. The parameters are different here. And so I'm just calling a different algorithm.

But the case that would happen is I guess you would have 112 here. And then you'd go 1, 2, so how many? So let's say it gets to 128. And if it doesn't break, then you're clearly done. So that's not the worst case. So if it gets to 128 and it breaks, then so how many drops have we done? 128 divided 16 is what? Is 8. Are you sure? You're all sure? OK. Then I'm sure.

So that's eight drops. And now you need to do 113 to 127. And that we know is 15. So that's 23. All right. So the best you have done here is 23. But are we satisfied? We're never satisfied. We always want to do a little bit better. So how did she pick 16? Why did you come up with 16, Lucy? It looked like a nice number?

AUDIENCE: I just figured it divides 128.

SRINI DEVADAS: You just figured, oh, nice. So if you'd had 17, then we would have had real trouble with all these numbers. Yeah. Go ahead. Josh.

AUDIENCE: Think you can do a little bit better if you get closer to the square root of 128?

SRINI DEVADAS: Ah, beautiful, beautiful. I love you guys. You guys make it so easy. So that's the next thing I was going to try and pry out of you, but you got it out without much effort from me.

So you can kind of get analytical here as opposed to doing things empirically, which was try a number and figure out what happened, which is kind of what we did. And you can be a little more analytical about this and you could say, hey, suppose I want to find this number k, which is kind of the 64-- started with 64, went to 16. And now, we're thinking we can do better than 16 even. And we start dropping things at k, 2k, 3k. And I want to get this exactly right.

So let me look at my notes. And you go dot, dot, dot-- n divided by k minus 1 times k and n divided by k times k, which is obviously m. And so I'm trying to, I'm trying to write it-- I'm trying to get this k. I want to write something. I want to minimize at some quantity and discover the k that is going to give me my best case in the worst case-- minimize the worst case, right? So that's really what I want.

So if you look at what happens here, you're in a situation where you have to now, you have a count, right? I mean, well, what do you want to-- we've decided that k is this general strategy. And we don't quite know what k is. And we have to do some analysis of the worst-case situation in order to find the k that would work well for us.

And so the way you do this is by, essentially, the analysis we've done so far. And kind of we have a nice, the point that was made here, I think, it was Kevin who said we have to go to the last drop, because that's kind of the case where it's the worst case. And that's similar here as well. Because if you go up all the way here and then it breaks, if it doesn't break here, you're done. If it breaks here, then you've got, you've done a lot of drops before it broke that first ball. And so that's the situation which you can pull in as your worst-case situation. So that's exactly what we did before.

And so if you really think about it, assume first ball doesn't break until the last drop is the worst-case situation. And so what does that mean from a symbolic standpoint of with respect to the number of drops of-- so the number of drops in this case of the first ball? Can someone

give me an equation for it? Yeah. Go ahead.

AUDIENCE: n over k.

SRINI DEVADAS: n over k. You a Kevin too? Yeah. OK. So n over k. Right. So that's what happened there. Now, well, it broke, right? That's what we decided here. And so now, you need to search n k plus 1 to-- well, you got to go all the way there. I'm sorry, it broke there, so you got to go n minus 1. This is the same as what I wrote before, but it's just that I had numbers. And it was kind of easier to write numbers than write these symbols. But we have to do this because we don't know what k is.

So if you look at how many drops are-- and if you're just going to do this, you get-- if you notice, this gets simplified to n. Yeah, so this is n, and then you have minus k plus 1, all the way to n minus 1. So if you subtract, and this is a plus 1 here. Boy, this is confusing. I'm not good at math. I'm definitely not good at math. So this is k minus 1 drops in the worst case. All right.

So you now need to add these two things together. So what we'd like to do is minimize n divide it by k, which is this plus that. We want to minimize this function. And it's a constant. k is a variable. And for a given n, we want to discover the k that minimizes this quantity. And it goes back to the intuition that I think Josh had about the square root of n. So again, high school calculus here, which I've forgotten, which hopefully you haven't because you still need it. I don't.

And you end up essentially saying you can turn this into two raised to the square root of n minus 1 drops in the worst case. All right. So you can take the floor of that, and that's 11. So that's that square root of n. So 2 times 11 minus 1 is 21. So you can do 21. So you think 21 is the optimum? Yeah, Kevin.

AUDIENCE: Can we do a little better if we make it go smaller?

SRINI DEVADAS: Yes. You can do a little bit better. I don't want to do this just yet. Hopefully. We'll have time. We'll get to that because I want to go to the d greater than 2 case, but you're absolutely right. It turns out that you can do, this would give you 21, as I mentioned, 22 minus 1. It turns out, you can do better but by a few drops.

And the assumption we made here, what is the assumption that we made here? When I wrote

this, what assumption did I make, which is kind of similar to what Kevin just said, but not exactly? Would someone point out the assumption that was made here? Not you. Someone else. All right. Ganatra. Go for it.

AUDIENCE: The intervals are the same.

SRINI DEVADAS: Yeah. The intervals are equal. So this, look, it's a different problem. If it didn't break here, then I got a smaller-- I have two balls. And I had a smaller problem. So why would I use the same k? I mean, that kind of makes sense, right? So now, it's more complicated and hopefully, we'll get to that. But I want to do other things first.

So that's the reason. So at least you know why 21 isn't optimal, right? It's because of this constraint that we placed on ourselves in order to do this analysis to get to the point where we had 2 times square root of n minus 1 that corresponds to the equal interval constraint. So good. All right. Any questions about this?

So as you can imagine, I'm not going to show you code that computes 2 times square root of n minus 1. So that doesn't make sense, but we do have code to look at that corresponds to the case where d could be 3, d could be 4. And then obviously, it gets more interesting in terms of what your strategy is. And we won't go with optimizing.

In fact, there's an open problem as to what the optimum strategy would be for arbitrary d-- d meaning the number of balls-- if it's 5, 6, whatever-- partly because of what I just talked about with respect to you need to utilize unequal intervals. But the natural thing to do now is to move to d equals a arbitrary number of balls and stick with kind of the same algorithmic approach of having equal intervals.

But the question is, what are these intervals? What and how would we find the size of these intervals? Because it's kind of not immediately obvious if I told you d was 5. And you don't want to just keep the three balls away, and sell them later, and just use the two, right? That doesn't give you your minimum. It doesn't minimize your worst case. You want to use these balls. Cool. All right.

So let's talk a little bit about if I had d equals 3, I mean again, you can use your intuition. What do you think we should do? What happened for d equals 2 with respect to that interval and how is it related to n? So what do you think, just wild guess, maybe even? What do you think we need to do if we had d equals 3 for that very first ball in terms of what we need to look at in

relation to the interval size? Yeah. Back there.

AUDIENCE: We need a cube root of 128.

SRINI DEVADAS: Ah, cube root, cube root. What's your name?

AUDIENCE: Ryan.

SRINI DEVADAS: Ryan. So Ryan says, you know, cube root. And yeah, that's the correct intuition. It's a little bit unclear, you know, what happens. You know, so you do cube roots, and maybe you have what's the cube root of 128? 5, yeah, that's right. I think it's 5. OK. It's close to 5, you know. It's 6 or 5 or 6, right? And so maybe I do 6, and then I do 12. And it's a little bit strange.

So it turns out, it's actually the 6 and 12, and you start with 128 when you have three balls, doesn't seem like it's going to give you the worst case, right? Because, you know, like in the worst case, you might be dropping the ball 6, 12, 18, 24, et cetera, et cetera. So cube root is right, but it's something a little bit different, because it gave us this interval that was kind of too small.

So how would I get something that's bigger? You'd want things that are a little bit bigger. I mean, I think you're going in the right direction, but I don't want 6 and 12 and 18. I mean, clearly, that's going to give me like 25 drops and that's even worse than the two ball case, which really means you should lose your job and never get another one, right? So if you do something that bad, right? So that's-- what would you do that's kind of different from cube root, but related to cube root? Yeah, go ahead Fadi.

AUDIENCE: 128 raised to 2/3 maybe?

SRINI DEVADAS: Yeah, yeah. Two thirds. Exactly right. So what you want to do is, you want to start-- it makes sense that when you have lots of balls, in terms of being able to use them, that you start with larger intervals and you shrink them. That's exactly what happened with two balls. You started with k in the case where we had two balls. And then we are down to one when we have one ball. So that's really what you're doing.

When you have one ball, all you can do is go one floor at a time. When you have two balls, you go k, where k happened to be square root of n. And then we had three balls, you kind of want to start with like n raised to 2/3 in terms of the interval. And then if I had four balls, the argument would be that I would start with n raised to 3/4. And then I would go to square root of

n. And then I would go so on and so forth.

And what is that reminding you of? What is this, in terms of these powers and getting up to not quite n, but 5, 6th n, so on. What does that remind you of from number theory or just sort of regular arithmetic?

AUDIENCE: Bases.

SRINI DEVADAS: Yeah, different bases. It's different radix arithmetic, right? So it turns out, the way you want to solve this problem-- and this is in keeping with the intuition you have with respect to the cube roots, and so on, and so forth. The way we want to do this is, we want to think about d digit numbers. And the numbers themselves, and d is the number of balls. And so that's the d digits.

But what is the radix of these numbers? The radix of these numbers are going to be related to the k, to the intervals. So you essentially want to say, you know, I'm counting in multiples of k-- 1k, 2k, 3k, et cetera. So if I had like radix k, that means I'm just incrementing that particular digit.

You know, if I go, if I had radix 12, and I had 1 0 in radix 12, then this would be-- so when I go 2 0, then the most significant digit got incremented. And so this is plus 12. This is 12 times larger-- I'm sorry-- 12 larger than two 0's, 12 larger than 1 0 in radix 12. Make sense? Just like 20 is 10 larger than 10 in decimal.

So that's the way you want to think about this. And so you're going to do r-ary numbers. And the first thing that you need to do is, you need to figure out, based on the number of floors that you have, figure out, you've given d. You're given n, the number of floors. And you're given the number of balls. And you got to figure out what radix you're working in. And this is going to be a generalization of the square root of n, as you'll see.

We want to choose a radix such that r raised to d is greater than n. Choose r such that r raised to d is greater than n. And so that comes back to, obviously, if you needed, if d were 2, then it makes sense that r would be square root of n, which is in keeping with what we talked about. But if d is 3, then that radix-- not necessarily the interval that we choose, because the intervals are going to be different-- is going to be more like the cube root of n, which is also proposed. Ryan mentioned that.

And so in this case, let's say that n were 128 and d equals 4. Then we would choose r equals

4, because 4 raised to 4 is 256, which is greater than 128. And we can't choose 3, because 3 raised to 4 is only 81. And 81 isn't greater than 128. So you'd have to choose 4.

So what we're now saying is, we want to somehow figure out-- we're not quite there in an algorithm yet. But we have a kind of a path forward in that we want to look at that representation, that r-ary representation of these d digit numbers. And we're going to essentially increment. And so with our goal, roughly speaking, we're going to go with this representation. And we're going to start incrementing digits. So we just go 0, 1, 2 in terms of the digits.

But because the representation is r-ary that means something different in terms of the number of floors, depending on what r is. I mean, just like I said, if we're this were ternary representation then in terms of floors, this is only-- this would be, for ternary, this would be three floors. 2 0 would be three floors away from 1 0. In decimal, 2 0 is 10 floors away from 1 0, and so on, and so forth. That makes sense? Yep. Ask me questions if you're confused.

So let's take an example. So and then we'll look at code that actually builds this algorithm from scratch. So we'll take our example of d equals 4 for our canonical 128 floor Shanghai Tower, because you do want to get a sense of how things improve. We did improve from obviously d equals 1 through d equals 2 with the equal interval assumption. But with this radix idea, we'd like to see how much better we can do with d equals 4. And we want to do substantially better than 21 drops or whatever we had back there.

So r is 4. d is 4-- oh, I'm sorry-- d is 4. n equals 128. And what we're going to do is, the smallest number is 0000. And in brackets, I'm going write what the decimal representation is, because I think that does give you some intuition. You get a sense of how this translates into floors. And after a while, if you're going to tell someone to do this, you want to give them floor numbers in decimal. Because the Shanghai Tower doesn't have some weird representation for the floors when you get into an elevator or anything like that.

So this is floor 0. And now, the highest number is 3333, which happens to be 255. I'm not going to write floor each time. But when I put things in brackets here, that's the decimal representation. OK. So that's 255. So that's obviously too big. We don't need that. But we're stuck with having to choose r equals 4 for the reasons I described to you before.

So there's somewhere in here there's a 128. And I don't have it right up in front of me as to

what 128 would correspond to. But 1233, for example, would be floor 111. So the way this works, and you all know you're radix arithmetic, it's 3 times 1, plus 3 times 4, plus 2 times 4 square, plus 1 times 4 cubed. That's what you have. And that's how you get 111.

So what you have to do in order to do this right is to start with the most significant digit, which is the left-most digit. And that makes sense, because we want our intervals, when we have lots of balls, to be large. And that as you move and the balls break, you're going to start shifting rightward. And you're going to be incrementing different things.

And that's essentially going to give us give us our algorithm. And so we'll talk about-- let me just give you an example of an algorithm. And we'll look at how the algorithm is implemented. As I mentioned, it turns into 10 lines of code, because all you're doing is translating into this representation. And you're just incrementing bits of the representation.

And it's interactive. You can play with it later today or whenever. And you can sort of think of a hardness coefficient in your head then see how things work out. But before we end, we obviously will have to do a worst-case analysis of this algorithm because that's the whole point of this exercise.

So in this case, where do you think I should-- so I don't want to drop, obviously, at floor 0. I mean, that's there's no notion of that. So where do I drop at, at the beginning? The first drop is at, given what I told you? You can tell me in any representation. Where would I drop the first ball if I have four balls and 128 floors?

I said that I want to start with something that, obviously, there's some larger interval here. And I want to start on the left-hand side. And I want to increment the leftmost bit, which is the most significant bit. So I would start with 1000. So the first drop is at 1000, which is at floor 64. That's the first drop.

Now, if in fact this does not break-- all right, so let's assume that this does not break. Then what do I do next? 2000, right? And this would be 128. Let's assume that this breaks here. All right. So now, what do I do? It broke. I lost a ball. Now I have three balls left still. That's the good news.

And how many-- so what do I need, where do I need to move? If I have three balls left, I need to be in a representation that has how many digits? Three digits. So I need to move. I a broke at 2000, so I know that it's less than 2000. So I need to go back to this, because this is the

highest floor.

In general, whenever things break, you go back to the highest floor that you know that the ball didn't break. I mean, that's the canonical strategy that we've followed all through. When the ball breaks, you go back and say, oh, but I do know that there was a highest floor in which the ball did not break. And I have to start with 1 plus that, right?

So what I'm going to do here is I need to go up and because it broke, I need to look at 1001. The interval that I need to look at is 1001, because it's one more than that. And all the way to 1333, which is one less than that. So and this, you may feel a little bit better looking at this in decimal. I certainly do. It's 65, 127, which makes perfect sense. So far, we've played around with the balls, the first two balls. And so we haven't really seen anything very dramatically different from the two-ball case, but more is coming.

Let's say this does not break. If this does not break, then what would I do now? If it does not break-- I'm sorry, my bad, my bad. I haven't even told you. I've just talked about the interval. And I haven't said what drop you need to make, right? So ignore what I said just now. I think that the next question is, the ball broke.

So now I have a second ball. And I want the first drop of the second ball. And I want to look at this. I want to look at this interval. And it has to be somewhere inside this interval. But it's not going to be-- I'm not going to drop this at this floor, 65 or 1333. I want to pick something in between, right? And so now looking at this, tell me-- according to what you know so far-- where you think I should drop the second ball for the first time? Go ahead.

AUDIENCE: 96.

SRINI DEVADAS: 96. And how did you get that?

AUDIENCE: Just like I did the half point.

SRINI DEVADAS: And you put 1100?

AUDIENCE: Yeah.

SRINI DEVADAS: Perfect. What's your name?

AUDIENCE: Evan.

SRINI DEVADAS: Evan. So Evan says 96. I guess he thinks in decimal naturally, like most of us. But the way you would code this, by the way, is by incrementing this digit. And you'll get 1100. And in this case, let's say it does not break. Then where do you go, Evan?

AUDIENCE: Above floor 96.

SRINI DEVADAS: Yeah. After 96, it doesn't break. What would you do?

AUDIENCE: Then you would go to what is that?

SRINI DEVADAS: Well, it's-

AUDIENCE: In decimal?

SRINI DEVADAS: No, in this representation-- whatever is easier for you.

AUDIENCE: It would be 1200.

SRINI DEVADAS: Perfect, 1200, which happens to be I think 112. So you go to floor 112. This was 96, and then you needed to add another 16 to that-- yep, 112 and so on. So I won't bore you with this. You get a sense of what's going on here. And obviously with these things, you can wave your hands and you get a sense of what's going on.

But ultimately, you know, what's great about computer science and programming is you've got to teach a dumb computer to do exactly the right thing. And there's no two things about it in terms of you understand this algorithm. I didn't understand this algorithm until I coded it, fully. I mean, maybe I still don't understand it, considering the mistakes I'm making. But that's just because I've lost a bunch of gray cells.

So let me show you what the code looks like. It's 15 lines of code. It's not that much. And as you can imagine, the representation is-- I could have used many things. I just used a list for each of these digits. I'm incrementing these digits. And I won't spend a whole lot of time on the code, but what I want to do before we end here is talk about-- actually, let me do that first, before we go to the code.

I want to talk about the worst case here, because that's what this is all about. I want to get an equation, just like we did for 2 times square root of n minus 1 and all of these different things, I want to get an equation for the worst case in terms of the number of drops in relation to r and d, which are obviously related to n. Because r raised to d has to be greater than n.

So if you kind of look at this, and it turns out this is not a difficult analysis-- if you think about it, and you start thinking about worst case, and you think about how we're incrementing these things, there's phases. And if you can think about a phase as when a ball breaks, you know, that's when a phase is done. And you move to the second phase.

Tell me how many drops I have? You break it up in terms of the number of phases or the number of balls, whichever way you want to think about it, and use the same kind of worst-case analysis we've done with respect to what happens in the worst case. You have to keep doing a bunch of drops.

And really, for these algorithms, the floor 127, the hardness coefficient of-- actually, the hardness coefficient of 127 is the hard one, right? It's the difficult one, because you kind of go through and you go all the way there. And you do all of these things before-- you drop the ball a lot of times before it breaks. And you do that for every ball. That would be the worst case, right? So that's kind of where things are at.

So from a symbolic standpoint, what is the worst case in the first phase in terms of the number of drops of the first ball, from a symbolic standpoint? It's the first digit I'm talking about. It's the first ball I'm talking about. And I'm dropping this ball, as you can see, at these different points. So in general, how many drops would I have before I'm either, I may be done completely because I know it doesn't break a floor 128 or what have you. But what is the worst case when that ball breaks? Give me an equation. Someone.

So I drop it once. I drop it twice. Am I going to drop it like, if r is 5, am I going to drop it 10 times? What am I doing in terms of the-- what is the upper limit in terms of this digit?

AUDIENCE: r minus 1.

SRINI DEVADAS: r minus 1. And where am I starting from? 1, so there you go. What's the answer?

AUDIENCE: r minus 1.

SRINI DEVADAS: r minus 1, right? So that's it. So it turns out that I have r minus 1 drops in the worst case for the first ball. And, I mean, it's the same representation. Every digit is the same radix, obviously. And now, that ball broke and I moved to the second digit with the second ball.

And I'm going to start again. And in the worst case, I could have r minus 1 drops for that as

well, right, because I'm going to go increment from 0. The first one I'm going to do is a 1. And then I'm going to do a 2 and all the way to r minus 1. So r minus 1 drops for the second ball. And then I go all the way to d balls. So my answer-- yeah, go ahead. Over here. Ganatra.

AUDIENCE: So r is 4 in this case. And we say the worse case is 3. But how do we get 3? Because if it doesn't break on either of those, on the 1000 and 2000, aren't we just done and that's only two drops?

SRINI DEVADAS: Yeah. So it gets a little-- you're absolutely right. But if the number were 255-- so one of the things that's happening here is we've lost some precision because r raised to d needed to be this large number of 255. And this algorithm is actually going to work for 255 floors.

AUDIENCE: Oh, it's because of n.

SRINI DEVADAS: Yeah, because of n. So it's not, yeah, so that's kind of-- you know, it's a good question you asked. But unfortunately, you kind of want to choose a case where you have perfect precision in terms of r raised to d is exactly what you need it to be. And in that case, this analysis works.

AUDIENCE: So it works the max end.

SRINI DEVADAS: That's it, the max end. That's exactly right. So I mean, 255 is substantially larger than 128. So it kind of makes sense that, as you said, that you would be able to shave off certain things. And so you can kind of do that, but it's a little more involved. But I think the intuition is important. And the intuition is simply that your worst-case number of drops, assuming you're being a little bit imprecise, would simply be r minus 1 times d in the worst case. And this would be true for all n which is less than r raised to d, for all n.

So now, just in terms of numbers, I had r equals 4 and d equals 4. And so you end up getting 12 drops for the 128 case, all the way to 255 as well. You need 12 drops in the worst case, even for 255 floors, which is really what you were asking. So good. Right. That makes sense?

So I'm not going to spend a whole lot of time on this code, but that's it. It's converting. I could've used a Python library function for that, but that's simply a conversion because I wanted to print things out. I don't even need that, this particular subroutine, which is essentially something that is doing conversion from decimal to r-ary arithmetic.

And as you can see, you see I subtract and adding. The adding this is the incrementation that I just erased, but you do see it up there from 1 0 to 2 0. That's why you add. Sometimes you

need to subtract, because things break. And when things break, you need to go back to the highest floor where the ball didn't break. So that's where you do the subtraction. And so this is how, if you ran this, it's not the greatest user interface, I'll admit. But that's not the point.

The Crystal. So I'm going to do at 128 and 4, which is our example. And it says radix shows it as ball 4. Did the ball break? I say no. And then drop the ball. Did the ball Break if I say no here, then I'm done. So I'm going to go ahead and say yes. And then I'm going to say no and then no. I'm going to keep going-- no, no, no, no-- because this is the worst case.

And I go to 11 in this case. So you do get close in the sense that Ganatra was asking, you know, could you shave off? You only shaved off one because of the imprecision. So there is a case where even though you would only get 12 for 255, but you do get 11 for 128. OK. Cool. Good.

So I'll let you look at the code. If you guys can leave any time you want. I just wanted to mention, I wanted to go back to the one thing that I didn't get that I mentioned, but I haven't gotten to do, which is this is still-- you could still do better. You can still do better not because of what Ganatra mentioned-- you know, that's just that imprecision thing, but assuming even that was taken care of because the numbers worked out,

You can do better because k can be made a different. You could do unequal intervals. So let me put that in the notes and you can read it by yourself. You had a question, Evan.

AUDIENCE: Yeah. When you were going through the ball drops and you kept answering no, that the ball did not break, why did it start asking you or why did it start telling you where the next ball dropped? Because it went one, two, and then it went also to ball three and ball four. So will it just stay ball two the whole time it if never broke?

SRINI DEVADAS: That's exactly right, but that's because I'm lazy. Yeah. So it'd be a fine exercise to fix that. I mean, that's not one of the exercises. That's how you turn, you know after 30 years of teaching whenever I get a question, I turn it into an opportunity for homework. OK. Right. That's how it works. All right. Good. See you guys next time.