[SQUEAKING]

[RUSTLING]

[CLICKING]

PETER SHOR: OK. So, today, we're doing linear programming, and we are going to basically do a very, I think, four- or five-lecture unit on linear programming. And linear programming is one of the big, I guess, engines behind optimization.

And to show how important it is-- well, when I started working in 1986, there were a bunch of linear programming tools on the market. And Robert Bixby, in 1987, founded a company and wrote CPLEX, which was, by far, the best linear programming tool on the market, and he charged an immense amount for it.

And a lot of people bought it, and he-- and he sold his company 10 years later for, I don't know, $40 or $50 million. And IBM recently bought the company that he sold it to for $180 million. So it's big business and very important. OK. Just-- yeah.

So what is linear programming? So I'm going to give two examples of problems you might want to solve with linear programming, and then we will continue with other things. So what is linear programming? It's essentially optimize-- either minimize or maximize-- a linear function subject to linear constraints.

And we'll start with a diet model. And so this is actually taken straight from the lecture notes. And I think, given the costs, I think it only works if you're feeding mice. So let's say you have some mice. And there are nutritional requirements. They need a certain amount of starch. And there are two grains you can give them, grain 1 and grain 2.

And they need 8 units of starch, 15 units of protein, and 7 units of vitamins. And grain 1 contains 5, 4, and 2 and costs $0.60. And grain 2 contains 7, 2, and 1, and costs $0.35. And the question is, how cheaply can you feed the mice and satisfy their nutritional requirements?

So what we do with linear programming is we need to write down a model. And how do you write down a model? You choose variables. Write objective function and constraints. And then you feed your linear program to a linear programming package, and it will give you the answer. So, any suggestions as to what the variables should be? Yeah? Yeah. How much of each grain?

So let's let x1, amount of grain 1, and x2 be the amount of grain 2. And now, we want to write down an objective function, so we either need to minimize or maximize something. So what do we want to minimize? The cost. So let's write it down here. Minimize-- I guess it's 60x1 plus 35x2.

And what are the constraints? Well, the constraints are that they get enough starch, get enough proteins, and get enough vitamin. So that would be-- and what we do when we write down linear programs is we say subject to-- I believe we say subject to. Let's see if-- yeah, subject to, before we write down all the constraints.

So subject to-- and now we need some things. So we need 5 times the grain 1 plus 7 times grain 2 has to be at least 8. So 5x1 plus 7x2 is greater than or equal to 8. And 4x1 plus 2x2 is greater than or equal to 15. And finally, 2x1 plus x2 is greater than or equal to 7.

And I'm not going to solve it in part because it has an utterly trivial answer. And also, we're not going to actually tell you how to solve linear programming in this class. There's something called the simplex method, and I will give you a very rough overview of how the simplex method works at some point.

But actually-- explaining exactly how the details work is something, first, you'll probably never need because there are tons of linear programming packages, and you can just download one from the internet and use that. And second, explaining how it works is somewhat interesting, but it's also rather tedious, and there are lots of little technicalities you have to go through, so I don't really want to do that. So that is linear programming.

Oh, wait. I just realized there are two constraints I did not write down. Does anybody-- can anybody tell me what they are? Yes?

STUDENT: The non-negative?

PETER SHOR: Non-negativity. Yeah. So x1 greater than or 2, x1 greater than or equal to 0, and x2 greater than or equal to 0. And these are non-negativity constraints. Non-negativity constraints. And they're actually handled a little bit differently than most constraints in-- are often handled a little differently from most constraints in linear programming. OK.

OK. And now I'm going to come up with another example. OK. So let's say you're having a bake sale. And there are two things you know how to bake. Let's say you know how to bake cookies. And we'll simplify the ingredient list. Don't go home and use this ingredient list for real cookies.

So eggs, we need 1; sugar, we need 2 cups; and flour, we need 1 cup-- no, 1 and 1/2 cups. And then you also know how to bake meringues. And for those, you need 3 eggs, you need 1 cup of sugar, and you don't need any flour. And you can sell a batch of meringues for $20, and you can sell a batch of cookies for $20.

So the first thing we need to do is write down our variables. So what variables would you use? Yeah? No? Yes?

STUDENT: Like, how many cookies and how many--

PETER SHOR: Right. How many batches of cookies and how many batches of meringue? So let's say x1 is the number of batches of cookies and x2 is the number of batches of meringues. And what do we have?

Well, you want to maximize the amount of money, so you want to maximize-- so our notes call the things you're maximizing or minimizing z, and much of the time you don't actually need to-- much of the time you don't actually need to write down z in your linear program, but there are some times when it's very useful to have a variable name for the thing you're maximizing, so we'll use it.

Well, you don't want to run out of eggs, so that means that the number of batches of cookies, x1 times 1 plus 3 times x2, had better be less than or equal to the total number of eggs you have, which I didn't tell you how many eggs you had. OK.

So let's say we have 9 eggs, 8 cups of sugar, and 15 cups flour. So x1 plus 3x2 is less than or equal to 9. And 2x1 plus x2 is less than or equal to 8. And there's another one. 1.5 times x1 is less than or equal to 15. And of course, we have x1 greater than or equal to 0 and x2 greater than or equal to 0.

And one nice thing you can do with linear programs, if there's not very many variables and not very many constraints, is you can actually draw the feasible region on the plane. So I think I need to give you some notation before I go ahead with drawing this.

Notation. A feasible solution-- it's not necessarily optimal, is a point, let's say, x1 through xn are our variables, x1, x2, through-- I'm looking to see if the notes-- OK. Let's say xk. That satisfies all the constraints.

An optimal solution is a feasible solution that optimizes our objective function. So, OK. So what does the feasible region look like?

Well, let's write it down. Let's say this is x1 and this is x2. And we have x1-- well, we have 2x1 plus x2 is less than or equal to 8, which means we have the 0.8 up here, and the 0.4 down here, and we have the linear region there. So that's our first constraint.

Second constraint is x1 plus 3x2 is less than or equal to 9. So now, let's put 9 here, 3 here. And our second constraint will be everything under this. And the third constraint is 1.5x1 is less than or equal to 15, which means x1 is less than or equal to 10, which means the third constraint is everything to the left of this line.

So you can see that this third constraint is completely meaningless because anything that satisfies the first two constraints must satisfy the third one. And the actual feasible region is this region and the intersection of the first two constraints. OK.

I think, before I need to say anything more, I should say something about the standard form and the canonical form for linear programming problems because-- so the notes-- form.

So maximum z is equal to z transpose. These are our constraints. No, this is our-- not constraints, this is our objective function x. So here, I guess, z transpose is the vector 20, 20. Ax equals b, and x greater than or equal to 0.

Canonical form. Maximum-- and it's nearly the same. c transpose x. A sub x less than or equal to b. x greater or equal to 0.

Theorem. Any linear program can be put into, well, either standard or canonical form. So, how do we do this? OK, so there are some easier ones.

So suppose our objective function-- suppose objective function-- is minimize 3x plus-- 3x1 plus 5x2. How would you turn that into a maximization problem? Yes?

STUDENT: You just the negative.

PETER SHOR: Yeah, exactly. Stick a negative on it. Maximize minus 3x1 minus 5x2. OK. Suppose you have unconstrained variable x sub i. Can be either positive or negative. Maybe I should just tell you the answer to this because it's not quite as obvious as the last one.

OK. So replace xi with x sub i plus minus x sub i minus. x sub i plus is greater than or equal to 0, x sub i minus is greater than or equal to 0. So whenever you have-- so, I mean, it doesn't matter whether xi is positive or negative, there's also ways some xi plus and some xi minus which will give you any value of x sub i. And now these are non-negative variables.

OK, so there's two more cases I should talk about. And they are a little bit more complicated than this. Suppose-- so let's look up here. So canonical form, all the constraints are inequalities. Staggered form, all the constraints are equalities.

So to go back and forth between canonical form or standard form, or just to put an arbitrary linear program in canonical form, we need to turn inequalities into equalities and vice versa. So suppose we have x1 plus 3x2 less than or equal to 9, x1, x2 greater than or equal to 0, how do we make this an-- how do we make this an equality? Any suggestions? Yes?

STUDENT: You can add another variable?

PETER SHOR: You add another variable. Very good. What we do is we add something that's called a slack variable, and we get x1 plus 3x2 plus s is equal to 9 with s greater than or equal to 0. And if we have x1 plus 3x2 greater or equal to 9, we would add something called a surplus variable, x1 plus 3x2 minus s equals 9 s greater than 0.

And the last thing we need to do is we need to, say, how to turn any equalities into inequalities. And that is actually, again, pretty easy.

So if we have x1 plus 3x2 equals 9, we can turn that into x1 plus 3x2 greater than or equal to 9 and x1 plus 3x2 less than or equal to 9. So anything that satisfies this satisfies that.

And now, if you don't like-- if you want to put your thing in canonical form, you need to get rid of all the greater than or equal to signs, but that's easy. Then you just say minus x1 minus 3x2 is less than or equal to minus 9. And this is canonical form.

So why do we need to worry about canonical form and standard form? Well, standard form use in simplex method. So the simplex method is one of the standard ways of solving linear programs, and we're not going to go into it, but as I said, I will sketch you the basic ideas for it at some point.

And we use this in duality. And duality is one of the most important, I guess, aspects of linear programming. Well, another very important aspect of linear programming is that there's an efficient method of solving them, but as I said, we're not going to explain that in very much detail.

Because if you're an applied mathematician employed at some company, you never actually solve linear programming by hand, you use whatever tool you have available, which is probably CPLEX if it's a company that solves a ton of different linear programs, and you don't need to know what's underneath the hood in CPLEX. But if you are an applied mathematician, you really do need to know how to use duality, and I we will explain duality in this course.

So let's write down a theorem. Every linear program that has a solution, an optimal solution, has an optimal basic feasible solution. OK. So what is a basic feasible solution? Well, the easiest way to explain is by looking at this diagram. Of course, real linear programs, this diagram isn't planar, it's n-dimensional and impossible to draw on a blackboard, but the basic idea is the same.

So the basic feasible solutions are basically the intersection are the extreme points of the feasible region. So these are basic feasible solutions. And the theorem is that one of these is optimal. So why are they called basic feasible solutions?

Well, to explain that, I think I need to go into standard form and put this linear program in standard form. The above LP in standard form is maximize z equals 20x1 plus 20x2 subject to x1 plus 3x2 less than or equal to 9. x2. Or rather, plus s1. This is our slack variable.

And now we need 2x1 plus x2. And now we need a different Slack variable for the second equation. You need a slack variable for each equation when you're doing this. And finally, 1.5x1 is less than or equal to 15. And so that would be plus s3 equals 15.

So a basis here is a linearly independent set of columns. So, let's just take this one. OK. So that is x1 plus s1 equals 9. 2x1 plus s2 equals 8. And 1.5x1 plus x3-- actually, we're just picking these three columns, so we leave this one. So that is equal to 15.

And for every basis, there's a solution because this is just a matrix times a vector equals a vector. I mean, it's just a set of linear equations with k equations and k variables, so it has a solution. And I guess this solution would be x1 equals 10. And now you could use this to get x2 equals-- that's 20, plus s2, equals 8, negative 12, and s1 equals minus 1.

So you can see that this basis is not a feasible solution because it violates the non-negativity constraints. But every linear program has a feasible solution, which the feasible points here correspond to the extreme points of the feasible region, and the optimal one is the 20x1 plus 20x2 that is furthest in the direction of the vector, which is the objective function.

And what does the linear programming-- what does the simplex algorithm do? Well, what the simplex algorithm does is it starts at one basic feasible solution, which, in this case, you would use 0, 0, and it walks along the edges until-- and it chooses an edge which improves the objective function every time. And there's always an edge that improves the objective function unless you're actually at the optimum. So it just walks along the edges until it reaches the optimum.

So, is this an efficient algorithm? Well this is a little bit of history, which we're not going to-- we're not going to test you on, but this was Dantzig's simplex algorithm, just to walk along the edges.

And he proposed it-- it worked really well in practice, but pretty much for any-- I mean, you need some way of choosing which edge you're going to walk along, and pretty much for any way of choosing which edge you want to walk along, people could come up with a counterexample which it took exponentially many steps along the edges.

So it works really well in practice. In the worst case, nobody could prove that it worked really well. And this lasted until Dan Spielman and Tang-- Dan Spielman was a professor at MIT at the time-- invented a new technique called smoothed analysis that says that it works really well almost all the time. So, that is probably all we're going to say on the simplex algorithm.

OK. So now I want to talk about duality. And we're going to use this linear program as an example. And actually, as you can see from that diagram, we could just get rid of this constraint, it doesn't make any difference, and just look at these first two constraints.

So what we're going to do is we're going to ask, how can we find a lower bound on the optimum and how can we find an upper bound on the optimum? And then we will show that these two bounds actually will end up being equal under some conditions.

So, lower bound. Any feasible solution gives a lower bound. So let's take a feasible solution for that one. How about x1 equals x2 equals 2? That satisfies all the constraints. x1 equals x2 equals 2. And is that 2 plus 6 is 8 and 4 plus 2 is 6, and those are satisfied.

And the objective function is 20 times 2 plus 20 times 2 is equal to 80. So, that is one way you could do this. And I should say that for this linear program, even though I've called them batches, you have to assume that you can use fractional numbers for x1 and x2, otherwise you get an integer program and not a linear program, and those are much more complicated. OK. So any feasible solution gives you a lower bound.

How about an upper bound? So, I'm going to write down something. 20x1 plus x2. OK. Is certainly less than 20x1 plus 3x2 because we have-- well, because x2 is positive. And now we have x1 plus 3x2 is less than or equal to 9. So this is less than or equal to 20 times 9 equals 180.

And we can do the same thing for the second constraint. 20x1 plus x2 is less than or equal to 20 times 2x1 plus x2 is less than or equal to 160. So these are both good upper bounds, and this is a good lower bound. Can we do better? OK, let's do better.

Let's add this and this. We have two-- let's see. We have 3x1 plus 4x2. So we're adding these two constraints. It's less than or equal to 17. And we have 20/3. Oh. 20x1 plus x2 is less than or equal to 20/3 times 3x1 plus 4x2 is less than or equal to 20/3 times 17, which is equal to 113 and 1/3. OK.

So, we've got a much better upper bound by adding these two constraints. So what happens if we find the best some of these constraints? How does it look?

OK, so I want to go about this more systematically so we can find the dual-linear program. Every linear program has a dual-linear program, and we want to do this. So, we have-- and the way we will do this is we will assume that the linear program is canonical-- in canonical form for the time being. And luckily, this linear program just happens to be in canonical form.

So how do we do this? Well, what we want to do is we want a combination of-- a combination of these constraints which will let us give a best upper bound possible. So let's say we have y1 times our first constraint. x1 plus 3x2 is less than or equal to 9. And y2 times our second constraint, that's 2x1 plus x2 is less than or equal to 8.

And now, we want this combination to be less than or equal to-- all right. We want 20x1 plus x2 to be bigger than-- to be less than or equal to our combination. So we want the combination to be bigger than or equal to 20x1 plus 20x2. So that tells us that y1 times 1 plus y2 times 2 had better be bigger than 20. And 3y1 plus y2 is also bigger than or equal to 20.

So suppose we found y's like this, what would we have? We would have y1x1-- OK, y1 plus 2y2 x1 plus 3y1 plus-- 3y1 plus y2 x2 is greater than or equal to 20x1 plus 20x2. So this would be our things. And this is, in turn, less than or equal to, I want to say, 9y1 plus 8y2.

So to get the best bound on 20x1 plus 20x2, we need to minimize 9y1 plus 8y2. So we want to minimize 9y1 plus 8y2 subject to y1 plus 2y2 greater than or equal to 20, and 3y1 plus y2 greater than or equal to 20. And y1, y2 greater than or equal to 0.

So why y1, y2 greater than or equal to 0? Well, I mean, if y2 was less than 0, then when we add these equalities up, we have-- the less than or equal to-- if y2 was less than or equal to 0, the less than or equal to sign would turn into a greater than or equal to sign, and adding these inequalities up would not make any sense. So we need the yi is greater than or equal to 0.

So what did what did we get here? What is this thing? It's another linear program. So this is the dual-linear program. And now I want to write down-- I want to write down the dual-linear program for an arbitrary program in canonical form, and then we will show the theorem of weak duality. But maybe I want to-- maybe I want to solve these linear problems first.

So, x1 equals 3, x2 equals 2. This is a feasible point. x1 equals 3, x2 equals 2. That's 3 plus 2 times 3 is less than or equal to 9, and 6 plus 2 is less than or equal to 8. So this is a feasible point. And objective function is 20 times 5 equals 100. Now, let's solve this one.

OK, I did not write down the solution here, but there is a solution to this linear program, so the objective function is also 100. And the dual program gave you an upper bound-- and the feasible point in the dual program gives you an upper bound, a feasible point in the primal program gives you a lower bound. So because this 100 is equal to that 100, then we know that the answer is 100.

So let's try y1 equals 4, y2 equals 8 4 plus 16 is 20. 12 plus 8 is 20. And 9 times 4 plus 8 times 8 is equal to 100. So here is a feasible point for the dual program, which also gives the optimal objective value.

OK. So, canonical form dual. Well, the primal was maximize z equals c transpose x subject to Ax less than or equal to b and x greater than or equal to 0. Well, the dual turns out to be minimized y-- I'm sorry, w equals b transpose y subject to A transpose y is greater than or equal to c, y greater than or equal to 0.

And you can see why this is maybe, by looking at this example, when we did this example, we had the original linear program had the matrix A, 1, 3, 2, 1. This now has the matrix A equals 1, 2, 3, 1, and this 2 came from the Aij A 1, 2 thing-- A 2, 1 entry of A, and now it's the A 1, 2 entry of the A. So this is just-- this matrix is just the transpose of the original.

The right-hand sides are just the original objective function. And the new objective function is just the right-hand sides of these inequalities. So we have A transpose instead of A. The objective function gets turned into the right-hand side of the constraints, and the right-hand side of the constraints gets turned into the new objective function.

OK. OK. So now, I will prove something called weak duality. Theorem, weak duality.

If x is a feasible solution of the primal with objective function z equals c transpose x, and y is a feasible solution of dual with objective function w equals. b transpose y, then z is less than or equal to w. So any feasible solution of the primal is less than any feasible solution of the dual.

OK, there's also something called strong duality, which we're not going to have a chance to prove today, we will do that next time, but I'm going to tell you what it is. And then we'll give the proof of weak duality.

So strong duality is really-- I want to say this is one of the great-- one of the most important theorems in combinatorial optimization. Strong duality theorem. If the optimum z star is finite, so is w star, and z star is equal to w star.

So weak duality says that for any feasible solution, the value for the primal is less than or equal to the value for the dual. Strong duality says that if the optimal values are finite, then they are equal. OK. So, how do we prove weak duality?

So, we have z equals proof of weak duality. We have z equals c transpose x. Recall A transpose y was greater than or equal to c. So that's one of our-- that's our constraints of the dual.

So what we're going to do is we're going to plug this into here, and we get. c transpose x is less than or equal to A transpose y transpose x, which is equal to y transpose Ax.

And now, well, do we have something about Ax there? Up there, we have Ax less than or equal to b, so let's plug that into here, and that's less than or equal to y transpose b. And what is y transpose b? y transpose b, well, that's just actually the transpose of b transpose y transpose.

And b transpose y was w, so this is w because w is a scalar, so transposing it doesn't make any difference. So that says that z is less than or equal to this thing, which is less than or equal to this thing, which is equal to w, so z is less than w.

OK. There's two more things I want to do today, although I'm not entirely sure we'll get by more than one of them. Theorem, the dual of the dual is the primal. And really, I think probably all of you were expecting this to be true.

So the primal is the maximum. z equals c transpose x such that-- subject to, sorry, Ax is less than or equal to b and x greater than or equal to 0. Now the dual is minimum of w equals b transpose y subject to A transpose y greater than or equal to c, y greater than or equal to 0.

So, before we can go on, we want to put the dual in the canonical form, because I haven't told you how to take duals of anything that's not in the canonical form. So one way you can take duals of an arbitrary linear program is, first, put into canonical form and then take the dual.

Another way is to go back and look at the proof and figure out the formula for taking the dual of an arbitrary linear program, and we'll do that next time. But let's put this in canonical form.

So how do we turn a minimization into a maximization? We saw this already. We just take the negative. So we need to-- so what we need to do is we need to maximize minus b transpose y subject to our-- yeah. Well, this inequality is the wrong way around for a canonical form, so let's take the negative of it subject to minus A transpose y is less than or equal to minus c. And y greater than or equal to 0.

So this is the canonical form. So all we do is we want to take the transpose of A and put this up here. So the dual is min minus c transpose x subject to. OK, we need to take the transpose of this. That gives us a minus Ax is greater than or equal to minus b.

But now, if we just take the negative of this and the negative of this, we get back to the original thing maximize z equals c transpose x such that Ax is less than or equal to b rather than minus Ax is greater than or equal to minus b and x greater than or equal to 0.

So, the transpose of the transpose is the dual, which, we know, makes-- which justifies the name. Sorry, the dual of the dual is the primal, which justifies the name dual. OK. I have four minutes left, and I don't know if I can say anything reasonable in four minutes, so are there any questions about this? OK.

So one thing you do when you first see a linear program, if you're doing research, is take the dual and stare at it and see if it makes any sense intuitively. There are lots of linear problems where if you take the dual and you stare at it, it makes sense intuitively, and you understand why the dual is an upper bound on the optimum of the primal.

There are also lots of other linear problems when you get it. You take the dual, you stare at it, and you scratch your head and you say, why is this a bound on the primal? And, I mean, it is because it's the dual, but it's not intuitively obvious.

And some of those, if you stare at it for longer, you understand what's going on. And some of them, you can stare at it for as long as you want, and you probably still won't understand. So we're going to see illustrations of, I think, after the quiz-- or maybe before the quiz-- yeah. We will see illustrations of this kind of primal program making intuitive sense.

And then the dual makes also intuitive sense, is clearly an upper bound on the primal. So you're going to see that in, I guess, zero-sum games and in probably max flow.

So, I will see everybody on Thursday, and we will talk more about linear programs and duals. And I will put out a homework tonight, which I think will be due next Tuesday.