[SQUEAKING] [RUSTLING] [CLICKING]

PETER SHOR: And this network could be all sorts of things. I mean, it could be pipes, roads, electronic communications. And each edge has a capacity. And the question is, how much whatever can you get from a source to a sink?

So here, we have an example. This is a network. This 6 says that only six units of flow can pass over this edge at any given time. This says 4. This is 3. And the question is, how many units of flow can you get from s to t? And we will call the source s And the sink t. And these are the standard letters that are used for them. I don't know whether they stand for anything or not. I mean, clearly S presumably stands for source. But I don't know why t is the sink.

So the question is, how much flow can you get from here to here? Well, let's try putting flow from here to here. Does anybody see a path you could run flow on? OK, this is an easy question. What is some path from s to t? Let's say s to-- oh my, I have left out the numbers on the nodes. So this is 4. And this is 5.

So S to 2 to 4 to t, you could run three units of flow along this path. So let's put them there. Now, are there other paths you could take? Yeah, of course, there are. You could run two units of flow from here to here. So that's 2, 2, 2. And is this all? Well, no, because you could run two units of flow from here to here to here.

And we have two units of flow already on here. So this gives you 4. You have 0 units of flow already. So this is 2. You have three units of flow here. So that gives you 5. Now, is that the maximum? Does anybody see any more paths? Yes?

STUDENT: You could do one from s 2, 3, 5?

PETER SHOR: Yeah, you could do one from here. Wait-- oh, you're right. This was 4. No, wait. Yeah, this was 4. But you could run one up here, 4. You could run one down here, 1 and 3, and one up here, and one up here. And there's still one more path you could do, which I guess is one here, one here, one here, one here, and one here. So that's pretty complicated. But this would be 5. And this would be 2. And this would be 3 and 2.

No wait. And you could run one here from 4 to t. So that's there. And now, we have a flow. The flow through every edge is less than the capacity on that edge. So that's good. And we have gotten 9 units of flow from here to here. OK, I guess I copied down the-- no, I didn't copy anything wrong. I'm just looking at the green numbers rather than the blue numbers. So here is a flow of 9 units.

So what we're going to do later is write down a linear program. And the solution to this linear program will be the maximum flow. But first, I want to talk about cuts. So how could you prove-- how can you prove that this flow is maximum?

Well, let's look at these three edges-- this one, this one, and this one. There's four units of capacity on this edge, two units of capacity on this edge, and three units of capacity on that edge. And so this defines a cut. And at most, 9 units of flow can cross this cut. So that shows that 9 units of flow is maximum.

Cuts-- a cut is defined as a subset of the vertices or subset containing s but not t. And so to make this a subset of the vertices, gosh, what we do is we make the subset this guy and this guy. And maybe we should call it C. Edges from C to C-bar are edges in the cut.

And let's say the capacity of an edge is u of the edge. So summation over E equals i j inc j in C-bar. u of i j equals, I guess, the value of cut. And it should be fairly easy that the maximum flow is less than or equal to the minimum cut because if you have a flow from s to t, it must cross every edge of the cut.

So if the flow is 9 units, then-- or rather, if the cut is 9 units, then you can't have a flow larger than 9 units. And the theorem we will prove later. The value of max flow equals the value of minimum cut. So you probably recognize this as being somewhat something like LP duality. And this is the primal solution of LP. And this will turn out to be the solution of the dual LP.

And by the strong duality theorem of linear program, this guy equals this guy. So the maximum flow equals the minimum cut. So there's a bunch of things we have to do before we prove this theorem. So what do we do? First thing we do is write down the linear program. Does anybody have suggestions for what the variables should be? Yeah?

STUDENT: How many units you load to each.

PETER SHOR: Through each edge, exactly. So the variables are going to be x sub i j-- that's the flow over edge i j. And any suggestions for constraints? Yeah?

STUDENT: For any you don't know, you put the srt, like maybe the flow inverse is the [INAUDIBLE].

PETER SHOR: Right. So flow conservation, I guess, you could call it. And let me write this down. Summation, hope I'm getting this-- j such that i j is in the set of edges x sub i j. So this is a set of things going into node i and a set of things going out of node i equals-- no, wait. This is the set of things going out of node i j and the set of things going into node i. And let's put a minus sign here so that k such that k i is in a.

So this is a set of things going into node i, x sub k i equals 0 because-- linear programs, it's nice to write down, so you have the variables on the left side of the equations and our inequalities and the constants on the right side of the inequalities. And there's another set of constraints. Can anybody figure out what it is? Yep?

STUDENT: [INAUDIBLE]

PETER SHOR: Yes?

STUDENT: The capacities?

PETER SHOR: Yeah, capacities. We need to use capacities somewhere. And x i j is less than or equal to-- we call them u of i j. And also, xij had better be greater than or equal to 0. And what do we want to maximize? So what we want to do is we want to maximize some of the things into the sink, which is node t. But actually, we're going to do this a slightly different way. We're going to write out a new constraint summation.

Let's write down for all i. And I want to write down k such that kt x sub kt is in a. So this is all the things going into the sink. x sub kt is equal to z. And we want to maximize z. I did not leave enough room, so maximize z.

So that is the linear program. And I have a sneaking suspicion that I messed up a sign somewhere. But it doesn't really-- I mean, it doesn't matter. I mean, this is a linear program, which does give the solution. So it doesn't really matter if I messed up a sign, except then when we take the dual, it's going to be-- it's not going to be quite as nice.

OK, let's just keep on going and write down the dual. And if I got the sign wrong, we'll fix it after that. I want to say the sign is not wrong. This is a perfectly good linear program for max flow. It's just I am a little worried that, when we take the dual, we're going to have to flip some signs.

So why don't I take the dual now? To take dual, each equation gives a constraint, or each inequality-- no, each variable gives a constraint. Each constraint gives a variable. So we have two classes. Well, actually, I guess we have three classes of equations and inequalities. But I just want to use two of them. I'm going to call this one alpha sub ij, and this one beta sub i, and this one beta sub z, because this really is a lot like this, except slightly different. And I really should make this minus z equals 0.

So how do we get the variables of the-- how do we get the constraints of the dual and the objective function of the dual? Well, the objective function of the dual is just the right-hand sides of the-- right-hand sides of the constraints of the primal.

So we will have minimum-- because the original primal LP was a maximization, we want to have a minimization. And the minimization summation alpha ij u of ij subject to-- OK, we're going to have a equation for each variable. So let's say the equation corresponding to variable x sub ij is just-- you look at all the places where xij appears in equation. So x sub ij appears in beta sub i and in alpha sub ij.

And we should have alpha ij plus beta sub i minus beta sub j is greater than or equal to 0. So it's greater than or equal to because these were positive variables. And when we go from a maximization to a minimization, you're minimizing this. You want a greater than or equal to sign in your constraints.

Now, there's one more-- well, I guess z was also a variable in this. And so there's an equation, a constraint corresponding to z. And z appears in the objective function. So this should be greater than or equal to 1. And z also appears in-- well, I just want to say it only appears in one equation. That's beta sub z. So beta sub z is greater or equal to 1. No, actually it appears in-- yeah. Yeah, so this is the dual.

STUDENT: [INAUDIBLE]

PETER SHOR: What?

STUDENT: Why do [INAUDIBLE]?

PETER SHOR: Not negative 1? Because I should have written this equation minus and plus. And now, this left-hand side is things leading into-- oh, this is beta sub t not beta sub z. I'm sorry. That was a very good question. Beta sub t. And this thing is k-- this should be edges leading into t.

And if we look down here, edges leading into i have a minus sign. So to make it consistent, this has to have a minus sign. And then this is a plus sign. And we have beta sub t greater than or equal to 1. That was a very good question. And that explains why I was worried about the signs. Good.

So what do solutions of this equation look like? Well, how do you get cuts out of solutions to this equation? And this is a very-- this is a rather tricky thing. So for this, we have-- we put numbers on the nodes of this network. And we want to maximize alpha sub ij times the capacity and subject to the capacity beta i minus beta j is greater than or equal to 0.

And first, I want to say, for optimal alpha sub ij is equal to, well, either alpha ij-- oh, did I write forget to write down alpha ij bigger than or equal to 0? Because alpha sub ij was an inequality in-- the constraint corresponding to alpha sub ij was an inequality. So alpha ij is greater than or equal to 0. And the beta sub i's are unconstrained.

For the optimal, well, you want to make alpha sub ij as small as possible, subject to the constraints that alpha sub ij is bigger or equal to 0 and alpha sub ij is bigger than beta j minus beta i. So you want to make alpha ij equal to maximum of 0 and beta j minus beta sub i and the optimal.

So we're going to put values on the nodes. And we're going to make alpha sub ij for maximum 0 and beta j minus beta i. Now, suppose each alpha ij equals 0 or 1. And now, what we have is we have-- and I'm just going to say here, because we want to make everything as small as possible, we'll take beta sub t equal 1.

And we'll take beta sub s equals 0 because if there is no-- because if s-- if s is one of these-- if we're looking at ij or i equals s for this, then really we don't have a beta sub i. And the equation is alpha i sub j minus beta j is greater than or equal to 0. But you can just make everything uniform by setting beta s equal to 0. So we have beta t equals 1. Beta s equals 0. So we have 0, 0. OK, I should put some-- This is 1.

Let's make this 1, this 0, et cetera. So alpha sub ij is the maximum of 0 and beta j minus beta sub i. So that means that this is going to be 1. This is going to be 0. This is going to be 1. This would be minus 1, except that's less than 0. So this had better be 0. This is 0. This is 0 for the same reason. This is 1. This is 1. And this is 0.

The 1 edges are a cut because the 1 edges are just the edges leading out of the things-- nodes with alpha-- nodes with beta i equals 0 and to the nodes beta i equals 1, which was our definition of cut. So that means that this is a cut between-- which nodes are zeros? Between the green nodes and the other nodes.

So there's one here, one here, and one here. So when you add up the-- you look at the objective function, it's just the sum over all of the edges in the cut times their capacity-- or 1 for all the edges in the cut times their capacity. So it's just the value of the cut. So this is dual objective function-- value of cut f. Beta sub i is in 0, 1.

Now, here comes the trickiest part of this whole thing. Why can we assume it's the minimum cut rather than this? In other words, why can we assume that the optimal solution of the dual has integer values rather than fractional values for the variables? Look at-- OK, we're going to look at this graph in another way-- graph above, wrong in another way.

So let's put s beta sub s equals 0 on the left side, beta sub t equals 1 on the right side, and beta sub 2, beta sub 3, beta sub 4, beta sub 5. I want to say that there's no reason these beta i's have to be in order. Put beta i's online at value of beta i in the optimal solution of the dual.

So if beta was equal to 0.2, then the optimum solution of the pool at this point would be 0.2. And now, let's look at the edges in this graph. And this is 6. This is 4. This is 2. There is an edge 3, value 3 going from beta 2 to beta 4. So this is 3. There's an edge from beta 3 to beta 4. So that is 3. There's an edge from beta 4 to beta 5. Wait, there's an edge from beta 5 to beta 4, which is this way, 2. There's an edge from beta 4 to t So this is 8. And this is 2.

So the claim-- edges crossing a vertical line form a cut. So let's look at this edge. This vertical line, this has edge 2, 3, and 4 crossing it. And we have beta s and beta 2 are on the left side. So that is just this cut.

You have s and 2 on the left side of the cut and all the other edges on the right side of the cut. And we have 4, and 2, and 3 crossing them. Now, let's try another cut. And I probably should use a different color. So let's try this cut. Now, we have 5 and t on the right side and the other four edges on the left.

So what cut does that correspond to here? That's not a different color I'm sorry. Let's try making it purple. And I see I have drawn things wrong. I left out an edge because there's an edge from b3 to b5. So that should look like this because we have three going to 5. And that edge had capacity 4.

So the value of this cut is 2. No wait, it's 4 plus 8. That's 12. And we don't count this edge 2 because it crosses the cut backwards. And we only count edges that cross the cut forwards. And here, we have that cut is-- OK, I need a purple line here, purple line here.

No, those are the wrong-- I'm sorry about that. So 5 and t are on one side of this cut. So this, this, and this edge is not across the cut because it crosses the cut backwards. So the value of that cut is-- no. These vertices are in the cut. So the edges are the ones that go from something not in the cut to something in the cut. So that's 4 to t, 3 to 5, and 5 to 4 is backwards edge. So the value of that cut is 8 plus 3, which is 11, which is bigger than 9. So it's not the minimum cut.

STUDENT: I think the beta 3 to beta 5 might have the wrong value.

PETER SHOR: I think you're right. So beta 3 to beta 5 is 4. And I put down 3 on here. So let's see beta 3 to beta 5, 3. So this is supposed to be a three. Thank you. So we have a cut of value 11 for there. So these are cuts.

Now, I want to claim the value of the dual optimum. Well, I want to label these cuts. So let's label this one as C1, C2, C3, C4, C5. And let's assume that the betas are put down in order because we can always renumber the variables so that the third variable is the one right after the second cut.

So the value of the optimum and the dual is equal to summation. Well, summation alpha sub ij u of ij, which is equal to summation beta j minus beta j minus 1 times c sub j minus 1. Why is that? Well, let's look at this picture again.

So we have alpha i sub j equals maximum of 0 and beta j minus beta sub i. So this is equal to summation over forward edges beta j minus beta i u of ij. Now, let's look at this edge. This edge appears in two cuts. And its value in the first cut is beta t minus beta 5 times the value times 8. The value of contributing the second cut is beta 5 minus beta 4 times 8.

So adding them up, when you sum up all the cuts, you get beta t minus beta 4 times 8 for the value this edge contributes to the dual, which is exactly what it's supposed to be. OK, so this is the trickiest part. So should I explain it some more?

So the cuts only come from the edges where beta 5 is bigger than beta 4. So this edge does not contribute anything to the cut. If there's an edge that goes forward, it contributes beta t minus beta 5 times 8 to this cut and beta 5 minus beta 4 times 8 to this cut. And adding those two things up, you get beta t minus beta 4 times 8 to the total objective function of the dual, which is just what it's supposed to be, because alpha ij is equal to beta j minus beta i.

Oh, wait, I think I wanted to keep that.

Objective function of dual is summation beta j minus beta j minus 1 c sub j minus 1. But summation beta j minus beta j minus 1 is equal to beta t minus beta s is equal to 1 because beta t was 1, beta s was 0. Let beta j minus beta j minus 1 equals lambda j.

So this equals summation lambda j c j minus 1. OK, maybe lambda j minus 1, lambda J minus 1, c j minus 1, where summation lambda j minus 1 equals 1. This is an average of values of cuts just by the definition of average. But the average of a bunch of things is bigger than the minimum of the bunch of things. So this is greater than or equal to the minimum cut.

And in the optimal solution of the dual, you take the minimum cut, you can put zeros on the beta sub j's that are to the left of the cut and 1 on the beta j's to the right of the cut. So you get the minimum value of the cut for the solution of the dual. So the value of the dual substitution is equal to the minimum cut. And we know that the value of the primal solution is equal to the maximum flow because that's the way we set up the LP. So the maximum flow is equal to the minimum cut.

So the next thing I want to do, which I don't think I'm going to quite get through all the lecture notes today-- but what I've done today so far is the thing that I really want you to take-- the lesson I want you to take home, that looking at the duals and analyzing them can give you very interesting theorems about linear programs. I want to say this is a fundamental principle in combinatorial optimization.

And if you actually get a linear program from somewhere, one of the first things you do is you look at the dual. You stare at it. And you say, what does this mean in terms of the actual problem you're studying? And often, it will give you some insight into the problem. Sometimes, it won't. Sometimes, it just looks like a bunch of variables that you can't understand at all.

I think we can erase this now.

Suppose you have a maximum flow problem with integer capacities. Then the solution is an integer. Proof-- the dual solution is a cut. So it's a sum of integers because the capacity of a cut is the sum of the capacities of the edges. And the capacities of the edges are integers.

So we can ask the question, does the primal solution also have, I guess, integer flows? Because maybe it doesn't. I mean, there are certainly solutions where it doesn't. I mean, you can imagine coming up with a solution where you have, I guess, 6.5 units of flow flowing over this edge and 1.5 units flowing over that edge and adding up to 9. So I guess I can't do arithmetic.

But you also know that all the edges, all the flows going across a cut have to be equal to the value of that cut-- the edges and the capacities in that cut. So you know some edges have to be integers. So the question is, do all the edges-- can you find a solution with all the edges integers?

And the answer is yes. And I don't know how to do it easily, working from the linear program. So there's another algorithm for integer flows-- algorithm for max flow-- which I'm going to call augmenting paths algorithm. And the idea-- find a path, push as much flow over this path as possible.

So as much flow over this path as possible, you will take the minimum residual capacity in every edge and the path. And that means that the residual capacity is always an integer because the flows are integers and the capacities are integers. So that means that the whole thing will be an integer. So let's try this.

And let's see. We have a simple network-- 2, 2, 3, 3. These are the capacities-- 2, 2. And let's put another edge from here to here with a capacity 3. And we're going to direct all the edges to the right. Good.

So the first step-- let's take the best path to push flow from here to here. So let's take 3 along here, 3 along here, and 3 along here. Now, how much capacity remains in the graph? Well, there's 0 left in this edge, 0 left along this edge, and 0 left along this edge. So it looks like we have a cut between the left side and the right side with-- and you can't put any more flow across.

But of course, you should know-- you should realize that you could easily put a flow of 4 on the original graph. So you have to be clever to use this algorithm. So how can we push more flow across this graph? I want to claim that we could use this path.

So this path, we push a 2 across here, push 2 across here. And now, we're going the wrong way in the edge. But we can do that because there's already a flow of 3 across this edge. And so to get a flow of 2 this way, we just reduce this 3 to 1. And then we can push another flow of 2 across here and another flow of 2 across here.

So you can push flow across edges backwards, as long as there is a flow left-- as long as there's a flow already along an edge. So I want to claim define residual network. If an edge has flow x sub ij less than u of ij, the new capacity is u of ij minus x sub ij. If an edge has flow x sub ij greater than 0, there's an edge in residual network with-- so let's call this edge eij e sub ji with capacity x sub ij.

So what is the residual network for this graph when we put the orange flow across it? Well, this had capacity 3. So now, that edge has capacity 3 in the opposite direction. This edge has 0. So that has capacity 0, 2, 2. This edge also has capacity 3 and flow 3 in the forward. So now, we get capacity 3 going in the backwards direction.

And this one, we have capacity 3 in the backwards direction. And this has capacity 3 in the forwards and the capacity 3 in the forwards. So this is a residual network. And now, you can see that there's a flow of 2 that goes bum, bum, bum, bum, bum. So we push this flow of 2 here.

And we get-- here, I'm going to redraw this. We get 2, 2, and 1 going this way, and 3 going this way, and 2, 2, and 3. So now, we have a flow from the source to the sink with value 5. And that's the best you can do because you can see that there's a cut of value 5. With just taking t and everything else, you can get at most value 5 across this cut.

So claim-- if there is no flow in residual network, then there is a cut in original graph. So if you have a value-- so I guess I should have started this with suppose you have a flow of value z. If there is no augmenting path in the residual network, then there's a cut in the original graph of value v.

If there's no flow in the residual network, let s be-- maybe c be set of nodes you can reach from s. So let's say we have this thing. So what nodes can you reach from s? Actually, you can't reach any nodes from s because this is at capacity. This is at capacity. So we could have made this capacity 4 and this capacity 4. And this is still a good flow in the residual network. And now, what nodes can we reach?

Well, we can reach this node. We cannot reach this node because this is at capacity. We can reach this node. Oops. And we actually can reach this node because it's at capacity. But we cannot reach this node. And we cannot reach this node. So we have a cut here. And that corresponds to this cut here.

So every edge from c to c-bar, the flow is equal to u of ij because, otherwise, you could push flow from c to c-bar and every edge from c-bar to c x sub ij equals 0. Because otherwise, you could push extra flow backwards along this edge. And that says the cut equals some c c-bar u of ij equals flow.

So if you cannot find any augmenting paths, if you already have a partial flow, then you're done. You cannot get any further. Yeah, I mean, then you have-- that is the value of the maximum flow because you have found a minimum cut that is equal to your flow.

So this says that you can use the augmenting paths algorithm to find max flows. And it will give you an integer flow if all the capacities are integer, which is what we wanted to show. And actually, there's an application of this, which maybe I'm going to get through. And this is König's theorem. Oops.

All right. In a bipartite graph, the cardinality of the maximum matching equals cardinality. of minimum vertex cover.

Let's look at a graph. So this is a bipartite graph. And what would be a maximum matching? Well, I mean, we clearly should put this vertex in the matching, this vertex in the matching. I guess we need another vertex here. And we can put this vertex in the matching and this vertex in the matching.

So the maximum matching equals 4. So the maximum matching is just the maximum set of edges such that no two share an point. Minimum vertex cover-- set of vertices that touch every edge. And clearly, vertex cover is greater than or equal to the matching because there must be a vertex that touches one endpoint of each of the edges in the matching. I mean, you're not going to get less than that. But Konig's theorem is that these two quantities are actually equal. And proof-- maximum flow, mincut.

We want a flow whose value is equal to the matching. So what are we going to do? Well, we're going to take s on the left and put an put an edge of weight 1 to every vertex in the matching. And now, we're going to-- and let's do the same thing on the right.

We'll put an edge to the sink from every point on the right side of the bipartite graph. And these capacities are going to be 1. And now, let's see. There is a max flow with integer values. And what that means is that-- oh, and we can let the capacities of these middle edges be infinity if we want.

Well, what that means is that there are going to be some number of edges in the middle which have weight 1 here. So I want to claim that those are clearly a maximum matching because, on the right-hand side, if there's 1 unit of flow from here to here, then no other edge could use this vertex. So if you have a flow from here to here, then this edge is in the matching. And neither of these edges is in the matching and similarly on the left. So the maximum flow gives you a maximum matching.

Dual is a cut. And that means that there's a cut whose value is equal to the maximum flow. And there are going to be some edges on the right, which are cut, and some edges on the left, which are in the cut. And I don't think I did that right. But if the edge is in the cut, then this vertex is in the vertex cover. And now, we know that there aren't any other paths from s to t. And that means that we know that every vertex, every edge is covered by at least one vertex because, otherwise, there would be a path from s to t going through that edge.

Let me see. OK, I did not prepare this quite properly enough. So this is an edge in the cut. This, I believe, has to be an edge in the cut. And you can just say this is an edge in the cut. There should be one more thing.

And that means so there are three edges in this cut. And I think that cut actually cuts all the paths from left to right. So we should have a maximum matching of only 3. And well, it's easy-- 1, 2, 3. And you can't get any more things in the maximum matching because-- because there's a cut. So this is Konig's theorem. And I seem to be completely out of time. So I will see you next time on Thursday when we start a new topic on information theory.