SRINIVAS DEVADAS: All right. Good morning, everyone. Welcome back from spring break. Hope you had a nice time off and couldn't wait to get back to 6.046.

So exciting second half. Going to do flow networks today on Thursday. Next week, it's linear programming, on to complexity, distributed algorithms, cryptography. Topics going to come fast and furious. Hopefully, they'll all be interesting.

So there's a lot of setup associated with network flow, max flow. It's an optimization problem. And so I'm going to spend like the first hour here-- and hopefully, it won't be too boring-- setting up flow networks, describing the max flow problem. As you can see, this outline is fairly involved, talking about cuts in a network, residual networks.

And we'll, more or less, end the lecture with the statement, though not the proof-- we'll save that for next time-- of the mas-flow min-cut theorem, which is really an iconic theorem in the literature, and suddenly, the crucial theorem for flow networks. And we'll take the max-flow min-cut theorem and use that to get to the first ever max-flow algorithm, which was due to Ford and Fulkerson. And that should be, pretty much, at the end of today's lecture.

And next time, we'll talk about the proof of max-flow min-cut, talk about some of the issues with Ford-Fulkerson, and then use max flow as a hammer to solve interesting problems in bipartite matching, baseball playoff elimination, and things like that. So just like shortest paths, can be used not just to compute the shortest distance from point A to point B, you can imagine that other problems, for example, scheduling problems, time problems, could be solved using Dijkstra-- and you've probably seen examples of that-- max flow is another algorithmic hammer that's being used to solve a wide variety of problems. We won't really touch on that aspect today, but we'll spend a bunch of time on Thursday talking about that.

So let's get started. We're going to start with defining what a flow network is. And at some level, it's really simple. I mean, it's a graph. A graph is going to have vertices and edges. We're going to only look at directed graphs. And we're going to have a couple of constraints associated with these directed graphs that are going to make the algorithms, and the proofs, and the notation a lot simpler. I'll get to that in a few minutes.

And the key thing with the flow networks-- I mean, just like you have a source and a destination for shortest paths, in a flow network as well, you're going to have a source, and you're going to have a sink. So we're going to have distinguished vertices, two distinguished vertices. And we're going to call them the source, s, and a sink, t.

And so the basic idea here-- and just to sort of set things up right up front-- is that there's going to be some flow coming out of s. And it's going to have to obey some constraints associated with capacities of the edges, in order to make its way to t. And along the way-- this is flow. This is water. You can think of it as water, or cars, or what have you. It's a rate. You're not going to allow accumulation in the intermediate nodes.

So you're going to have law of conservation associated with this commodity that's flowing, be water, or cars, or people. And that essentially comes down to, for any vertex, other than s or t, everything entering the vertex has to leave the vertex. So that's like Kirchoff's current law, for example. So there's a lot of analogies here with respect to real life, or electricity in this case. And you'll see that, I think, over and over as we go along.

So let's take a look at a flow network. I'm not going to draw it out here. I'm going to keep this one example all through the lecture. But just to go off and talk about edges and capacities before we actually draw an example, we're going to have edges, directed edges, u, v. So you have an edge from u to v. And it's going to belong to E, obviously. That's the reason the edge is in the network.

And this edge is going to have a non-negative-- each edge is going to have a non-negative capacity, C(u,v). Right? And if perchance theres' no edge from, let's say, s1 to s2-- these are different vertices from the source or the sink, just as an example, you can say it's between a and b-- then, we can assume that the capacity is 0.

So if u, v does not belong to E, then assume that C(u,v) is 0. So there's no way of getting from u to v. This rate, wall water, cars-- you know, there's no road that gets you from u to v, OK?

So let's draw an example of a flow network. We'll talk a little bit about what the flow is and what the max flow is in example-driven algorithm design, if you will. So I've got a source, s. And I've got a sink, t. And then, I've got a bunch of nodes in the middle. Just draw them out.

I'm not going to bother naming those nodes. Well, that might be a mistake. We'll see. So this is not like it's an acyclic graph. You're allowed cycles in this graph. But we will have a couple of constraints that are associated with this graph that I'll get to in just a minute.

So first, that's all we've got. You've got g,v,e, directed graph. Can't have cycles in it. And I'm going to draw a couple of numbers here. The numbers I'm putting up here are the capacities associated with each of these edges. So those are just the capacities. Those are C(u,v). And I said that, if u,v does not belong to E, then you're going to assume that C(u,v) is 0.

So there's no edge, for example, between this node and that node. So there's no way that you can directly get from here to there. You can go like this, like this, and like that, and get from here to there. And you can obviously go like so, and like this, and like that, to get from the one on the top left to the one on the bottom right, OK?

So that's essentially the set up, except for the fact that we've just talked about capacities. And this is sort of the bandwidth, if you will. This is the amount of traffic that can go through this road. And now, we have to talk about the specific case where we're actually going to shove things through the network.

So we're going to have another number. So typically, we're going to have two numbers associated with each edge. One of them is going to be the capacity, and the other one is going to be the flow that goes through the edge. And as you can imagine, we're going to have this constraint that says that the flow can never exceed the capacity.

And that's the local constraint associated with each edge, OK? That doesn't mean that there aren't variations possible with respect to the overall flow of the network. Things can change. You obey the edge capacities. And there could be different flow coming out of s and going into t, et cetera. And so there is going to be an optimization here associated with the exact numbers that all obey these edge constraints, so these edge capacity constraints, OK?

So let's take a real simple example of a flow in this network. And ah, I have some colored chalk here. So I can put down the flow over here. And so this is flow. Wow. That is an ugly color.

[LAUGHTER]

But now, we're stuck for the rest of this lecture. So I've got 2, 2, 1, colon, 3. So the first number is the flow, and the second number is the capacity. Let me just write that out. This is-- and then, this is 1, colon, 3. This is 1, colon, 2. So that meets the capacity. This is 2, colon, 3. And then 1, colon, 3, 2, colon, 3, and 1, 2, right?

So that's my first example of a flow. And we want to make sure that this flow makes sense, OK? And we're going to write this down a little more precisely in just a minute, but I've sort of given you the intuition already.

I've talked about it's OK to have a flow from the source. It's the mountain spewing water, or that's the source of the river. And this is the sea, for example. And the river flows into it. But along the way, you really can't have accumulation, OK? Because when you think about it as a rate-- and so this is sort of gallons per second-- then maybe you'll have a little bit of accumulation allowed. But over a huge period of time, you can't have infinite accumulation. So that's where the conservation law kicks in, which says, anything that goes into a node that is not marked s or t has to leave the node.

So you look at this, and you go, well, there's two things coming in here. There's 1 and 1, which adds up to 2. And there's two 2 leaving. So we're good there. OK? And then pick another one. This one, for example. You've got 2 coming in, and you've got 1 and 1 leaving. We're good there, right? For t, you've got 3 coming in, 2 plus 1. And another check to do is, well, you've got 2 coming out of the source, right?

So hopefully, that all will make sense. And ask questions if you're confused.

One other thing to look at, which is interesting, is that you could have flows that are essentially cyclic. You could have commodities that are flowing in a little cycle. And can you see that? Can anyone see that over here? Yep? Go ahead.

STUDENT: The bottom-right triangle [INAUDIBLE].

SRINIVAS DEVADAS: Bottom-right triangle? This thing over here?

STUDENT: Yeah.

SRINIVAS DEVADAS: Like that? Good. Right? So you see this one going over this way, this way, and that way? And so there's nothing that's stopping us from taking this and-- I had 1 over here, so I'm going to make that 0. I had 2 over here, and I'm going to make that 1. And I had 2 over here, and I'm going to make that 1.

STUDENT: [INAUDIBLE].

STUDENT: [INAUDIBLE]

SRINIVAS DEVADAS: Oh. What is it?

STUDENT: It was 1.

SRINIVAS DEVADAS: It was 1 before? Oh, you're right. It was 1 before. So I should make that 0 then. Good.

I wanted to subtract 1 from what I had before, right? And I screwed that one up, the simple thing. All right? Wow. No wonder I didn't major in mathematics.

[LAUGHTER]

OK. Computer Science is this forgiving field.

So you can go 0-- well, this is 0. So now, let's take a look at what we have here and check that nothing went wrong, right? We want to check that nothing went wrong. And so 2 coming in, 1 going out, 1 going out. And nothing coming in here, so we're all good. And so on and so forth, OK?

So there's interesting things happening here with respect to the conservation laws and having to obey them. In general, you can imagine that what we're interested in is simply maximizing the flow from the source, and getting as much flow out of the source as possible, and pushing it into the sink. OK?

Now, the flow here into t, as you can see, is 2 plus 1, which is 3, OK? Can anyone take this particular flow network and increase the flow? I mean, it's easy to decrease the flow. You can make everything 0, and that'd be valid, right? But can you increase the flow? Do you think that this flow network allows for a larger flow than 3, given that the capacities of those edges are fixed and I'm not going to change them?

Let's see. I think all of you have Frisbees, right? Yeah. Yeah, back there.

STUDENT: I don't think you can, because you're maximizing two edges, which would have to be increased. Like, the bottom one coming out of s would have to be increased. Or the one on the very top would, too, also have to be increased.

SRINIVAS DEVADAS: People agree with that? Over there? Yep?

STUDENT: You can have a path going along the top with a flow of 2, so that s is the one on the top of it has 2. That, from that, to the right, has 2, like it has now. From that node, the t has 2. And then, along the bottom, we have another path that has 2, from s to the 1 [? plot of it. ?] And then [INAUDIBLE].

SRINIVAS DEVADAS: OK. So--

[LAUGHTER]

That made perfect sense-- to me. [CHUCKLES] It did.

So one of the crucial observations that [? Rajesh ?] made here-- and let me just focus in on that-- is-- and that's why I think the other gentleman said no-- is that you can actually decrease the flow in particular edges. And that's going to help you increase the overall flow, right? And that's why this problem is challenging. That's why we need so much of a setup, all right?

So one thing that I could do is I could essentially say, I'm going to take this and make this 0. So when I do that, essentially what I have is I get to push more flow out from here, right? So I get to push more flow. I can turn this into a 2. And I can turn this into a 2, right? Am I done?

STUDENT: No.

SRINIVAS DEVADAS: Not quite. I mean, I've got one little bug here, but I can fix that, right? I've got one little bug here which says, I've got 2 coming out here. I made this 0, so I have to push something more. But hey, I'm lucky. I've got a s here, which is giving me as much as I want, correct? So I can just make this a 2. And so now we get a flow of 4, right?

You both get Frisbees. Shall we do blue and purple? Here you go. Could you stand up? Oh, OK. Over there.

So that actually kind of summarizes, at some level, our task ahead, right? So we have to find ways of increasing the flow. And sometimes, we have to take a step backwards in the sense that we decrease the flow on a particular edge, right?

So it's not this monotonic increase that Dijkstra would do or a greedy algorithm, like MST that Eric talked about. There's going to be something a little more interesting here. We eventually are going to end up doing things in a monotonic way, in terms of the overall flow.

So the max flow that we're trying to get at is going to start with the current flow. And we're going to improve the current flow constantly. But that doesn't mean that the edges are going to look monotonic in terms of the flows on a particular edge in relation to the capacity. You'll never exceed the capacity.

But as you saw in this little example already, we increase from 3 to 4, by taking that edge that was vertical up there, which says 0, colon, 3, and that was 1, and we shrank the flow on it. And so how are we going to discover these paths, especially if we have a 5,000-node network and-- I don't know-- 10,000 edges? And so that's essentially what we have to do for the rest of this lecture. Any questions so far?

OK. So we've done flow networks. I kind of defined what the max flow problem is. And let me just write that out more precisely. Given a flow network, G, find the flow with maximum value on G. And for that, the max flow is 4, right?

So that's another thing we haven't actually done, which is obviously important for us to do, which is we have to show that the 4, in this case, is the max flow, right? So now, of course, I've given that away. And so I'm not going to ask you can you push this over to 5. But you might think that 5 is a possibility, simply because the capacities of the edges that are coming out of the source are 3 plus 2, which is 5, right?

So it's certainly possible that you could push at least, if you only look at those two edges, that you could push 5 units from the source. But in this case, in this example, if you obey the laws of conservation, you cannot obey those laws and get 5 units from the source to the sink, t. But we have to prove that. And we have an algorithm that says this is the best that you can possibly do. And the algorithm terminates when that happens. And that's our Ford-Fulkerson algorithm, right?

Good. So that's what we have so far. I want to talk about flow network assumptions. And I can do that over here. This is going to make our life easier.

One of the things that's a little bit confusing sometimes is the circular flow and the fact that we're going to potentially have flows that correspond to edges coming in and edges going out. So for example, if I had something like s and u, for example-- and s could be the source in this case, or it could be another node-- suppose I had a little subnetwork that looks like this. And I'm just giving you what the capacities are.

This is a bit strange in the sense that you could have a situation where you essentially have zero flow, really, because you have one unit coming in here and one unit leaving. All right? And you can think of this-- let's just call this s1, to make it clear that it doesn't have to be the source, right? And so you could have the circularity that you saw over there. And now you're talking about, well, I might have 1, colon, 1 here, and 1, colon, 2 here, which is fine for this subnetwork.

But if I have stuff going out, what happens with that? Well that's got to be a 0. I could have a 1 and a 2. And if I have a 2 here, then-- if I had a 1 and 2 here, then that doesn't work, because maybe I need something else coming in. So you can see that, pretty quickly, it gets kind of confusing, if we end up having these cycles that are such simple cycles, especially the ones where you have Su and Us, OK? And so we're going to disallow cycles of two kinds, right?

The first cycle we're going to disallow is this simple one where we say, if I have a, then no self-loop edges allowed. So that would involve accumulation at a particular node. And it's going to make things really confusing. And *CLRS* disallows that. And most flow network algorithms assume that you're just going to discard these cycles.

This particular transformation that I'm going to describe here is something that is going to be forced on you. And this is for your benefit in over 6 lectures and sections. But it's not actually something that *CLRS* follows. And it's going to make things simpler, though.

And what we're going to do is we're going to take any pair of vertices that has this characteristic, where you have s1, u, and u, s1, and they have non-zero capacities. So s1, u has a capacity of 1. u, s1 has a capacity of 2. Both of these edges exist. And if these edges exist, that means they have non-zero capacity, positive capacity.

And we're going to transform that, very simply, into-- this is not changing the generality of the algorithm, but all I'm going to do is transform it-- call this s1-- into something that satisfies this restriction. So I could have 2 here, 1, and 1, OK? So all I've done is introduce u prime, right?

Then I can always do this-- if this is trivial for any pair of vertices, I can introduce one other vertex. It all works out. Linear expansion, constant factors, ignore them. Life is wonderful.

So all I've done is taken away the situation where I have-- you can think of it as two-way streets, right? Two-way streets are annoying. You don't quite know what the rate of traffic is from one end of the street to another, because cars are going in both directions. And you have to do subtraction. Subtraction is painful, so we don't want that.

So we're just going to assume that this is u prime. And now, you're all set. We allow this. So we have to allow for generality reasons. As we saw in that very first example there, we are going to have cycles here, OK? But we just don't want cycles to be of length 1 or 2, OK? So that's essentially what we're going to disallow. All right?

The good news is that, if you do this, then we'll only have a single notion of flow. Whereas, if you go read *CLRS,* you'll see that there's two notions of flow in *CLRS.* There's positive flow, which is different from net flow. And the positive flow is different from net flow in graphs that have this particular structure, or have nodes with these properties.

But if you disallow them, then you can just talk about flow, and it doesn't matter. Positive flow is the same as net flow, all right? So for the purposes of [? 6 over 6, ?] for this semester, we're going to simply think about flow and equate that to positive flow, equate that to net flow. And it's all going to work out, assuming your graphs satisfy these two properties of the cycle lengths. All right? Cool. Good.

So let's keep going here. So we're up to finished up on max flow. Let me just give you some sense of notation. I've talked a lot about constraints. But we've got to write some stuff out, because we're going to be getting more precise and proving things in just a few minutes.

So what is a flow? Well, to be precise, it is going to be a function that satisfies the following properties. It satisfies the capacity constraint. This is the obvious capacity constraint, intuitive capacity constraint. And then we've got flow conservation. And the important thing here is that I don't have it for all V, but I do have it for vertices, V, that are not the source or the sink. And I'm going to require f(u,v) equals 0, right?

And the last one, which I haven't talked about, but becomes easy to talk about, given this constraint, is skew symmetry. So if you take-- this doesn't have to be an edge between u and v. Now, I'm talking about the flow, f, between u and v. And so the u could be s, which is the source. v could be t, which is the sink.

So in general, I'm not talking about a flow. And there obviously has to be a path from u to v, in order for there to be a non-zero flow, f(u,v), right? If there's no path, there's no way of getting there. But having said that, the definition of a flow here in our network is the straightforward definition, which simply says, if there's a flow from u to v, regardless of what u and v are, then the value of that flow is simply the negation of the value of the flow from v to u, which makes perfect sense, right?

And this all works out under the definition of net flow. And so this is essentially what the definition of net flow is in the textbook. But for our purposes here, we don't have to add that adjective. We just are going to be talking about flows. Positive or net, they're the same, all right? All right. Good.

So one of the things you can do with this notation-- and we're going to use what's called implicit summation notation on top of this-- is to prove some interesting things, interesting theorems, that give you some intuition as to how algorithms on flow networks are going to work. And in particular, we're going to use this notation when we talk about the value of a flow. So the value of a flow, f, is denoted-- you can think of it as a cardinality of f. And f is v, belonging to capital V, f(s,v). And that is f(s,V), so what I have written here.

Well, given a flow network, I want one particular quantity that I want to maximize. And that particular quantity is going to be defined, based on how much I can push from s, how much can I push outward from s. That's the crucial quantity that I want to maximize. That quantity is-- you think about everything that is going out of s, and you add it all up together.

So from s, you look at any other vertex, every other vertex, and you say, what is f(s,v)? And I'm not talking about just the edges that come out of s. A vertex, v, small v, can be any vertex. If I add up all of the flows that come out of s, then that is the flow that responds to my flow network. That is everything that's getting pushed out of s, OK?

Now, it may be the case that-- remember, I'm talking about flow here-- so it may be the case that you have an edge from s coming in from v4. And there may be a flow associated with that. This is maybe something like 1, colon, 2, all right?

So what this means is that f(s,vr) is-- this is f, remember-- so this is minus f(v4,s). And in this particular case, this is 1. So this is minus 1, OK? So keep that in mind.

When I talk about the flow of the network, I'm going to be looking at the source. And I'm going to be looking at all of the flows that are going outward. And I have to keep in mind the skew symmetry relationship. I obviously have to obey capacity constraints and the conservation laws, all right?

So given that, let's use this implicit summation notation and show some simple properties of flow. So let's look at-- one thing I want to emphasize is what I've done here is use this implicit summation notation, which simply says, if I see a capital letter here, that's a set. And I'm going to have to enumerate all of the members of that set. And it's implicit summation. So as I enumerate those members, I'm going to add up all of these quantities, right? So that's really what this means.

So the sigma here gets embedded into this capital V. So two things going on. The small v turned into capital V, because I'm looking at the entire set. And the sigma gets in there too. And that's why it's implicit summation, not just implicit set notation.

So some simple properties. I can say, f(x,x) is 0, where x is an arbitrary set. All that says is, let's say, x has a single member in it, which is a. Then f(a,a) is always 0, because if you don't allow self-loop edges, and that's pretty much all you need.

If you have a pair of vertices here, a and b, then what you're saying is f(a,b) plus f(b,a) is 0. And that's true, because of skew symmetry. Right? We just wrote that out. So f(x,x) is 0. And in general, you can say, even though X and Y are sets of vertices, I'm going to be able to use skew symmetry to say that f(X,Y) is minus of f(Y,X), all right? Similar argument.

And then, lastly-- there's any number of these, we just do three of them here-- X of f(XUY, to Z) is f(X,Z) plus f(Y,Z)-- we've got to use these properties to prove our first theorem here on flow networks-- if X of X intersection Y is null. So you don't want to double-count. So that's all this is. Make sure you're not double-counting. You've got f(XUY), and you want to look at that entire set, the union, and then look at the flow from any member in XUY to any member in Z. And you can do that by breaking it up, provided you're careful about double-counting. And the fact that the two sets, X and Y, do not have an intersection, or they have a null intersection, implies that you're OK with the [? write ?] inside. All right?

So you might be going, why are we doing this? Well, here's a good reason to like implicit set notation. You can prove some interesting theorems in a very elegant way, using this notation.

So let's do one example of that. You'll probably see others in section.

So one of the things that we'd like to do is prove a pretty important theorem, which I think all of you probably can assume in your heads, given all of the properties of flow networks that we have. And it's a very simple theorem that simply says, I have the law of conservation that is applied on all of these intermediate vertices, and I've got a bunch of commodities, I've got a flow going out of s, right? So where can this flow go? Where does this flow end up? It ends up at the sink, at t, right?

So the point is that, if you have all of these properties that we have up here, you're going to be able to show-- and you want to show this, you want to prove this-- that the value of a flow, which is defined as what gets pushed out from the source, is exactly what goes into the sink, right? If that's not the case, there's been a violation of some property, perhaps a capacity constraint, perhaps, more likely, a conservation constraint, OK?

So the theorem that we'd like to prove is simply that f is f(v,t), right? That's the theorem. And there's a lot going on here, so it's worth spending 30 seconds looking at what exactly this means.

What I have here is that, if you just look at this and that, I'm saying that f is what gets pushed out of the source, OK? And now what I'm saying here is that f, the value, is exactly what gets pushed into the sink, OK? So this is what I have to prove, right? And I should be able to prove that, by invoking my laws. That's it. I mean, that's my axiomatic system. I've got those laws. I've got a definition of a flow that may not necessarily be the max flow. It might be something much less than the max flow, it might be the max flow.

Regardless, what gets out of the source has to get into the sink, right? So how are we going to do that? And the implicit summation notation is going to give you, essentially, a three or four-line proof, which is very intuitive, right? So let's do that. And maybe you can help me.

So we're going to start with what we know. So that's the proof. f equals f(s,V). Right? So that's what we've got. That's the definition of cardinality of f, or a value of f, OK?

What I'm going to do is I'm going to say this is the same as-- I'll give you the first step, and then let's see if you can help me with the remaining-- is the same as f of v minus s capital V, right? So if you're having trouble differentiating between my cap V's and small v's, holler. I'm trying to write them as big as possible. Yeah?

STUDENT: Could you maybe put little hats on the top of them?

SRINIVAS DEVADAS: Put little hats on them. Yes, I will put little hats on them. I'd put little Frisbees on them, if I could, but-- I like Frisbees much better than hats. All right. That's good. That's good to do. So yeah. So I think, hopefully, I'll keep doing this, and it won't be confusing.

So what I've done here is invoke, essentially, this, except it's not exactly that, in the sense that it's written a little bit differently. But if you see what's going on here, what I've done is look at this s, and I've said, think of this s as being cap V minus s. Right? So that gives you s. And those are clearly disjoint, right? Those are clearly disjoint sets.

There is this one and this one are disjoint sets. That's what I mean to say. I mean, these two aren't disjoint, but this and that are disjoint. And that's what you need, in order to invoke the little property that you have here. And so that all make sense? You see why I did that? OK?

What can I say about either of these two quantities? Can I say something about either of these two quantities? Yeah?

STUDENT: f(V,V) is 0.

SRINIVAS DEVADAS: f(V,V) is 0. That's exactly right. f(V,V) is 0. There you go. Yep. So this is simply I'm going to invert that. I'm going to write this as f of V hat minus s, OK? I'll just flip this. I had a negative sign here, but I've flipped that. And skew symmetry tells me I can do that, right?

All right. So I'm up to this point here. Now, what I'm going to do is I'm going to do f(V,t). And the reason I want to do this is because this is where I want to get at, right? Eventually, I want to show something that corresponds to f(V,t), right?

And what I have here is V,t. But now, I could do plus f(V,V minus-- cap-- minus s minus t) right? So what I've done here is taken V minus s and pulled out t from it. Remember, t is part of cap V. Cap V contains all of the vertices. So I've pulled out t from it, but that implies that I have to do a V minus s minus t over here. And again, they're disjoint, so it's all good. What can I say about this? Yeah?

STUDENT: It's 0 because of flow conservation?

SRINIVAS DEVADAS: It's 0 because of flow conservation. That's exactly right. We didn't quite write it that way. But if you look at what the implicit summation notation would mean for that, you look at it and you say, maybe one more step would be, let me think about this as being, f(V,t). It'll become more obvious if I write it this way. f(V-- I'm putting a minus in here-- V minus s minus t and cap V again, right?

So all I've done here is flip these two. Skew symmetry allows me to do that. And now look at what I have here. I'm talking about a flow that corresponds to some-- for any vertex, I pick-- and it's not an s vertex, it's not a t vertex, so it's an intermediate vertex. And if I look at an intermediate vertex and look at the flow that goes out to all vertices, conservation says that has to be 0, right?

So that's exactly what this says. For any u that's neither s nor t but in V, the sum has to be 0. So this is zero, and we're done. All right.

Oh, you-- a Frisbee? Who is that? Ah. Here.

So that's the power of implicit summation notation. So we could invoke these different properties. It was fairly straightforward. Your first example of this. You'll probably see a few more. All right? Any questions so far? OK.

So as you can see, as I promised, or threatened at the beginning, but followed through on my threat, we have a lot of notation, a lot of baggage here before we get to algorithms. But we're slowly getting there.

The next major concept is the concept of cuts. So a cut, you think of a cut as being, well, a cut through paper, a cut through the air, whatever. It turns out that notion of a cut in a network is more general than that, right? A cut is basically a partition. A cut is a partition of nodes. And a partition means that you can't have a node in both sides, right? So a cut is going to give you two disjoint components at the end of it.

But the cut doesn't have to be something contiguous. It doesn't have to be a line through the network. And everything to the left of the line is in one half of the cut, and everything on the right of the line is in a different half of the cut. I can just break up these nodes into two disjoint parts. And the only constraint that I'm going to ask for is that s, which is the source, is on one side of the cut, and t, which is the sink, is another side of the cut, OK? That's it.

And given that, I'm going to say interesting things, really interesting things, about the flow through a cut, OK? And so let's do that. Let's define a cut.

So a cut is (S,T) of a flow network, G, is a partition of V, such that small s belongs to cap S, and small t belongs to cap T. I don't know. Do you want hats on the T too? I'll just write them large. If a flow on G-- if f is a flow on G, then the flow across the cut is f(S,T). OK?

So again, implicit summation notation here. The flow across the cut is as the sum of the flows corresponding to each pair of vertices, such that the source vertex is part of capital S. And the destination vertex is part of capital T. All right? That's it. I'm just going to add them all up. That's the flow across the cut.

So what I can do now is just talk about-- let's just go up here back to this. And I'm going to look at exactly what I have here. Is that right? Not exactly. I'm going to change this a little bit, because I want to make sure I don't have to add up numbers and do that incorrectly. So I need a 1. Yup. That's all I need to do is change it.

So I'm going to change our example here, not the topology of the example, but the actual numbers. And you'll need to verify that what I have here satisfies our flow network properties. And there's one more. OK? So I think I'm good. All right.

So this is going to be an example of a cut. I haven't defined the cut yet. Let's get rid of that. Holler if you think there's something wrong with this flow. All right? I think I got it right. It satisfies capacity constraints. It satisfies flow conservation constraints.

The flow that is going into t is 4. This happens to be a max flow. Doesn't really matter. So what we're going to talk about with respect to cuts, it doesn't require the flow to be maximum. Keep that in mind.

What do I mean by the flow across a cut via an example? I'm going to simply say that the shaded nodes, two of them, are part of capital S, OK? So as you can see, I just arbitrarily picked a couple of nodes. And that not necessarily something that can be easily partitioned using an actual cut line, a physical cut line. I just picked that one over there and the one over here with S.

And so, I can now look at this, and I can compute numerically, for this example. And it's worth doing at least once what the flow across this particular cut is, defined by the particular choice of cap S and cap T, OK? And that's what we're going to do.

So f(S,T) is-- I'm going to have to look at pairs of nodes, such that I've got a shaded node on the left-hand side and non-shaded node on the right-hand side. And I'm going to have to go through all of the combinations, right?

So if I look at this, I can first knock off this one, and that one, and that one. Let me actually put in-- let's call this a, b, and c here. And we can call that d. So we have s and d as being part of the cut, in terms of s, capital S. And the other ones are in cap T.

And so what I have is I could do Sa and Sb. So I've got 2 plus 2, all right? And this would correspond to Sa and Sb. So those are going out, right? So far, so good. And then, I'm going to write out a bunch of numbers here, minus 2 plus 1 minus 1 plus 2.

And the minus 2, where would the minus 2 come from? Well, an a,d, for example, is a minus 2, right? Because d is part of-- it would be d, a. So a, d has a flow of 2, correct? And so d, a has a flow of minus 2, right? And d, a is part of what I have here, because d is part of capital S, and A is part of capital T. You guys see that? So this is not trivial, so pay attention.

So this would be, for example, the minus 2 would correspond to d, a. That's what I need here. And I could also have-- what do I have here? I have something is going into d. So a c, d is 1. So d, c is minus 1, right? Make sense? d, c is minus 1.

What about the plus 1? Where do I get a plus 1 from?

STUDENT: d, b.

SRINIVAS DEVADAS: d, b is going out. That's exactly right, d, b. And the plus 2, it would be d, T, right?

And so you have to do the enumeration. It's worthwhile doing once. And then it gets kind of boring. We won't to do it again. But you have to realize that you have to absolutely look at every pair of vertices. And you have to use skew symmetry and ensure that, even though there's actually no edge going out, if there's an edge coming in, you've got to count that. And that's going to get a negative. Whatever is coming in, you've got to subtract, OK? So it's not that complicated. Yeah, go ahead.

STUDENT: Do we not consider S, c?

SRINIVAS DEVADAS: I'm sorry?

STUDENT: Do we not consider S, c?

SRINIVAS DEVADAS: So the beauty of this is that, when you don't have a particular edge from S to c, you can use skew symmetry to argue that S, c and c, S cancel out each other, all right? So that's the good part, right? And thanks for asking the question. That's a good question. All right. Here you go.

So you can do that by just looking at the edges. And you can add up the numbers, all right? And so I don't think this is going to be absolutely crucial to understand the rest of the lecture. Keep this in mind, that there's a process by which you define the value of a cut. And we're going to get back to this, when we prove the max-flow min-cut theorem next time. But at this point, I want to say something actually much more straightforward, which is going to be important when we look at residual networks, which is the last concept that we need to get at before we get to an algorithm. And that is simply that the capacity of a cut and the relationship between the capacity of the cut and the flow of a cut.

So the capacity of a cut is c(S,T). Oops. I didn't draw that properly. Open brackets, capital S, capital T. And we can do it exactly the same way, except this is a lot simpler, because you only look at edges and you only have positive quantities.

So in this case, you'll simply say it's 3 plus 2, corresponding to-- what did I have here? I had d, a-- S, a and S, d. And then, the capacity is you only need to look at the edges that go from a node in S to a node in capital T. And so those are 1 plus 3. And this simply would be the 1 would be d, b. And the 3 is d, t.

So you don't care about the other flows. This is not about flows, this is simply about capacity. So this adds up to 9, OK?

And so we have, at this point, we have defined the flow through a cut. And we know the capacity of a cut, OK? It's more or less obvious-- though you could certainly prove a theorem which is going to take a couple of lines-- to say that the value of any flow is bounded by the capacity of any cut. And sorry, I lied. It's not a trivial proof. And that is actually something profound going on here. And so I'll have to explain exactly what this means. And then we'll take a look at how we could prove something like this.

So what's cool about this is that you're saying that it's the value of any flow is bounded by the capacity of any cut, OK? And so that's an upper-bound on the maximum flow value, right? So I'm saying there's all these cuts that are possible in the network. And I'm making a statement about what the maximum flow can be, based on the values corresponding to the capacities of any cut, right?

So why is that the case? Well, we're not going to be able to prove that fully today. That's the max-flow min-cut theorem. But you can certainly get a sense of it, by looking at a different characterization of the flow value. So I'm going to give you one half of the proof, at least, and intuition about the other half. And we'll finish it next time.

But here's another characterization of the flow value. So our lemma here, which is going to lead us to this statement, is that, for any flow, f, and any cut, (S, T), we have a really powerful dilemma. Maybe you should call it a theorem.

But it essentially says, look, it doesn't matter what cut you choose, you've got a flow on the network. And when you look at the flow on the network, it's going to equal the flow across the cut. And the only reason for this is simply because you've got the source on one side of the cut. And you've got the sink on the other side of the cut. That's it. That's the only thing that you need, right?

You dump these vertices these into two bins. You know, dump the source on the left, and dump the sink on the right. And you compute the flow the way we've defined it. That's the flow. It doesn't matter how you partition these vertices, as long as you've got the source on the left and the sink on the right, OK?

And so we can prove this using implicit summation notation. We'll do that. And that'll give you a really good sense of why this statement is true, because we know that, for any given cut, the flow that cut is bounded by the capacity of that cut, right? You know that.

But to show this, here's how we could show that, f(S, T) is f(S, V) minus f(S, S), OK? So I'm playing around, just like I did before. I had taken the cap T-- I know that S union T is cap V, right? This is a partition. So I know that S union T is cap V. So I can put a V here and an S here, right? And that's a subtraction over there, of course, right?

So put that up here and finish this, a couple more lines. And what can I say about either of these? I could say something about one of these terms. Yep?

STUDENT: The one on the right is [INAUDIBLE].

SRINIVAS The one on the right is 0. So call this f(S,V). Right? And now, I'm going to break it up again,

DEVADAS: make it small s, big V, plus f(S minus s, cap V). So broken this up into small s, which is just joined from cap S minus s, clearly.

And what can I say about this? This is a little more subtle than, perhaps, the previous question. What can I say about that quantity? I mean, the answer is not subtle, but-- yeah, go ahead.

STUDENT: That that is equal to 0.

SRINIVAS DEVADAS: And why?

STUDENT: Because the cap S doesn't contain t.

SRINIVAS DEVADAS: Ah. Beautiful. That's right. So that's what I wanted. So this does not contain t. And so, now you can use flow conservation, right? And that's the important thing.

You can use flow conservation, because this does not contain t. And then, it clearly does not contain small s, because I just took it out of it, right? So that goes to 0. And voila. That's simply f(S, V), which we know is f. We proved that. Our first implicit summation proof was showing that-- well, this is a definition. Excuse me. So we did it for the sink.

But this is simply the definition of the flow value, right? So this is beautiful. I mean, it's like fantastic, right? Why aren't people excited?

[LAUGHTER]

Because I put people to sleep before, in the hour before. But this is absolutely fantastic, because this says that I have ways of figuring out what the maximum flow of the network would be, by making arbitrary cuts through this network and looking for capacities of these cuts, right? Because I know that the capacity of any cut-- and now you see why min cut is interesting-- but you know that the capacity of any cut is going to bound the flow of the network, because the flow through a cut is the flow through the network.

So if I go through and look at the min cut corresponding to the minimum capacity associated with the flow network, that's going to point me to my max flow, because that going to be an upper bound on the max flow, right? And so now you see why the min-- not the min flow, sorry-- the min-cut max-flow theorem is an interesting one. But it relates-- and this is the beginning of that relationship, we're not quite done to prove it-- but the beginning of the relationship is that you can look at any cut, and you can look at the flow through the cut as being the flow through the network. And then you use the second part of it, which is the capacity bounding-- of course, in a very simple way, because of edge capacities-- the flow through the cut. And you can put those two things together, all right?

We still don't quite know how to find these cuts, right? So we don't quite know how to find these cuts. And that's the last thing that we're going to do today, give you a sense of how we're going to find these min cuts, so we can find the max flow. All right? Cool.

So the one last notion that we have here that is going to allow us to go into the algorithm domain, as opposed to the analysis domain-- all we've done so far is analysis, analysis, analysis-- is the notion of a residual network. OK? And a residual network, as its name implies, is something that has residual capacities. It's the network that points you to places where you can increase the flow, because there's capacity left.

Your flow is less than the edge capacity. It's a local notion, so it's easy to compute. There's a capacity of 3 on this edge, there's a flow of 2. The residual capacity is 1, right? 3 minus 2.

And so the residual network Gf (V, Ef), right? So the actual network is G(V, E). And the vertices are going to be the same. The graph is obviously different, but the edges are going to be different. There's going to be a different set of edges in the residual network, as opposed to the flow network, OK?

And you have strictly positive. That means greater than 0, strictly greater than 0, residual capacities. So Cf) equals c(u,v) minus f(u,v). And that's strictly greater than 0. I'm going to put an edge in there, if there is a residual capacity. 0 doesn't mean there's any residual capacity. Edges in Ef admit more flow, OK?

And one last thing. If (V, u) does not belong to E-- so we are talking about the original network here, E means the original network-- then we know c(V, u) equals 0. That was our definition. If you don't have that edge, the capacity is 0.

But we are talking now about f, which is a flow, which doesn't necessarily require that there be an edge. And we are simply going to use our skew symmetry relationship. And you'll see that this may not be completely clear as to why I wrote this at this moment. But as I draw the residual network, you'll see why that is important. It's going to be the case that we're going to have extra edges in the residual network that don't exist in the original network, because of that last line there, right? So let me clear that up.

So we're going to draw a residual network for-- I'm going to change that yet again-- but we're going to draw the residual network for our example up there. Topology is going to stay the same. Numbers are going to change, because I want something slightly more interesting than what we have there.

So I'm going to take this out-- 2, 1. 1, 2-- go back to what it was before, I think. 1-- all right. Good. So I want a 1 over here. I want a 1 over here, a 2 over here. And keep checking to make sure I'm not messing up here, in terms of flow constraints. But that's pretty much all I got. OK?

So the flow here and the s and t don't particularly matter. The max flow is 4. The flow that you see up there is 3. This is what we had right at the beginning, all right?

So what I want to do now is give you what the residual network is for this particular flow. Remember, the residual network is defined, based on a flow. That's why you have Gf, G subscript f. f is a flow. So you're going to have a different residual network, if the flow is different. So that original example that I had would have a different residual network. This one is going to have the one I'm going to draw, all right?

So the residual network has the same set of vertices. So I can go ahead and draw these vertices. I'll just mark T and S over here. Those are exactly the same as before. And this is Gf, OK? That's a residual network.

And the edges are going to be different. All I have to do is look up there and say, look, I'm going to have a residual capacity of 2-- let me use a different color, since I have them-- corresponding to that edge from S to a, because I clearly have a capacity of 3, and I only have a flow of 1, right? So that's all there is to it.

Now, the interesting thing is that, because of this line over here, I'm actually going to define an edge in Ef, in Gf or Ef, that corresponds to this edge that didn't exist in E, because I can shrink the flow from 1 to 0. And that essentially says that that shrinkage is represented in the residual network by an edge that goes from this node, a, up there, to s, right? And that is going to have a residual capacity of 1, right?

So that's it. That's the only extra thing that you have to remember when you draw the residual network. You not only can increase the flow, you can also shrink it. You have to represent the shrinkage of the flow by an edge in the residual network. And you obviously represent the increase of the flow by an edge in the residual network. And now you see why this is all going to make sense.

Remember, way back, eons ago, only an hour ago, but we had this example where we had to shrink the flow in a particular edge, in order to get the overall flow to increase, right? The residual network is going to point us in the direction of, potentially, those edges whose flow has to shrink. But they're going to be represented, in effect, as these reverse edges with positive numbers associated with them.

So there's a 1 here. It's positive, because it goes from this node, call it a, back to s. And if I shrink this from 1 to 0, that is, in effect, taking what I have up here and making this 1 as 0, OK? That's the way you want to think about this. So that's pretty much it.

I could draw out the rest of this, and it should all make sense. I have an edge like that. So the edges that are at capacity up there aren't going to show up here. The edges that are not at capacity end up with two edges down below, if they have a flow that is non-zero, right? If you have a 0 flow up there, you're only going to get one edge. Obviously, you'll get one edge, because the capacity is non-zero. But the ones that are not quite at capacity end up to two edges, right? Think of that as being the simple rule.

And make sure that I'm following this rule. Good exercise is to check for bugs, lecture bugs. Best way of understanding the material.

OK? So that's my residual network. And let me just point-- I'll put in some numbers. Or maybe you can tell me what some of these numbers are. What is this number?

STUDENT: 2.

SRINIVAS DEVADAS: 2. Right? Now, this is a little more tricky. Take a look at the number that goes up, versus the number that comes down. What is this number? Goes up.

STUDENTS: 2.

SRINIVAS DEVADAS: That's 2, because I can go from 1 to 3. And this number would be?

STUDENTS: 1.

SRINIVAS DEVADAS: Beautiful. All right, you guys got it. Did my job.

All right, so that's our residual network for this particular flow. So a mechanical way of computing it. And you should be able to do that.

Now what exactly can we do with this residual network? It turns out that the algorithm now can be described in a couple of sentences, right? Essentially our Ford-Fulkerson algorithm-- I'm not going to bother writing this out, because I'm going to have to prove the max-flow min-cut theorem next time. And we're going to talk about the Ford-Fulkerson algorithm and issues with it next time.

But I'm going to show you how the Ford-Fulkerson algorithm works on this particular example. And it's only going to have one step, all right? So it's going to converge in one step. So it's going to be relatively easy. But the bottom line is the the Ford-Fulkerson algorithm is going to look for augmenting paths in Gf. Augmenting paths are defined in the residual network.

What is an augmenting path, you ask? Well, an augmenting path is simply a path from s in Gf to t in Gf, OK? That's it. That's all there is to it. If you can find a path, you could use depth-first search, you could use breadth-first search, you could use whatever you wanted. You find a path from s to t, OK?

If you find such a path, if this path exists, it means that the flow is not maximum, OK? If no path exists, the flow is maximum, and you're done. If such a path exists, you will be able to increase the flow, in this case, because we have integral quantities, by at least one, OK? And you will be able to increase the flow by one.

And not only that, the augmenting path is going to tell you exactly what to do with respect to what edges to change, sometimes subtract the flow from, sometimes increase, right? So the augmenting path is going to take care of it. But you've flipped it, so everything is positive. You know, life is great. Positive numbers.

Just look at s to t and give me any path from s to t here in the residual network. So let me just call them a, b, d, and c. And we don't care about partitions or cuts at this point. That's required for proofs. But give me a path from s to t Yeah, go ahead.

STUDENT: S to a, a to b--

All right a little bit more slowly. s a, OK. And then?

STUDENT: a to b.

SRINIVAS DEVADAS: a to b. Beautiful.

STUDENT: b to c.

SRINIVAS DEVADAS: b to c.

STUDENT: c to t.

SRINIVAS DEVADAS: And c to t. Wonderful. And I have a capacity, a residual capacity, of 2 here. I have a residual capacity of 1 here, 1 here, and 1 here. And so the value of this augmenting path is 1, right? It's not 2, because the minimum value is what I can push through the network. And I just need to take the min of all of the residual capacities corresponding to the edges that I traversed, that correspond to this particular path. And that min value of 2, 1, 1, 1, is 1, right?

So what this means is I've discovered an augmenting path of residual capacity, 1. Now I can go back to this thing over here. And these edges are going to point me, in some cases, to complimentary edges, in some cases to the direct edges that are going to have to have their flows changed, either increased or decreased, to increase the flow in the original network. So you're guaranteed now that f, which caused this residual network to have an augmenting path in it was not maximum. You're guaranteed that, because you found this path, OK?

So what happens here? Well, you go back up and you say, remember, I'm going to increment by plus 1, because that's the residual capacity. So I'm going to go up here, and I have s to a. You'll need to help me out again. So I'm going to make this 2, because I needed to add a one to it.

What happens here? This is the key step.

STUDENT: Subtract.

SRINIVAS DEVADAS: Subtract and make this a 0, right? We're not quite done. What's next? I have b to c, right? Right? You're still on the hook.

STUDENT: [LAUGHS].

SRINIVAS DEVADAS: b to c becomes?

STUDENT: 3.

SRINIVAS DEVADAS: 3. Right. And then, lastly, this becomes 2. OK? And at this point, if you create a residual network for this new flow, you will not be able to find a path in that residual network from s to t. And you know you're done, right?

So all of this works. We haven't quite shown that it works, because we haven't done enough of the proofs. But you have a sense as to why this works. You could probably code this up. It would all work. A couple of issues in applications. See you next time.