Graph Theory and Its Applications

Dr. G.H.J. Lanel

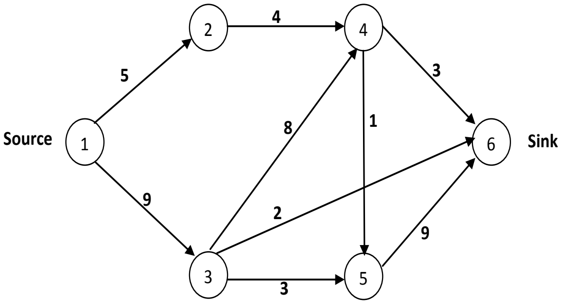
Lecture 10

Outline

Outline

Suppose that edges as pipes of varying sizes and vertices as places where the pipes run together and the material we are piping can change direction.

Suppose that edges as pipes of varying sizes and vertices as places where the pipes run together and the material we are piping can change direction.



Networks

- Since pipes either flow one way or the other, we will be working with digraphs.
- To model the fact that the pipes can be different sizes. We will assign a *capacity* to each arc, which we will write next to the arc.
- We also have a *source*, which is where the things are being pumped from, and a *sink*, which is where the things are being pumped to. What we get when we do this is called a *network*.
- $c_{i,j}$ the capacity of the arc from vertex i to vertex j.

Networks

- Since pipes either flow one way or the other, we will be working with digraphs.
- To model the fact that the pipes can be different sizes. We will assign a *capacity* to each arc, which we will write next to the arc.
- We also have a *source*, which is where the things are being pumped from, and a *sink*, which is where the things are being pumped to. What we get when we do this is called a *network*.
- $c_{i,j}$ the capacity of the arc from vertex i to vertex j.

Networks

- Since pipes either flow one way or the other, we will be working with digraphs.
- To model the fact that the pipes can be different sizes. We will assign a *capacity* to each arc, which we will write next to the arc.
- We also have a *source*, which is where the things are being pumped from, and a *sink*, which is where the things are being pumped to. What we get when we do this is called a *network*.
- $c_{i,j}$ the capacity of the arc from vertex i to vertex j.

Networks

- Since pipes either flow one way or the other, we will be working with digraphs.
- To model the fact that the pipes can be different sizes. We will assign a *capacity* to each arc, which we will write next to the arc.
- We also have a *source*, which is where the things are being pumped from, and a *sink*, which is where the things are being pumped to. What we get when we do this is called a *network*.
- $c_{i,j}$ the capacity of the arc from vertex i to vertex j.

Flows

To figure out how to pump the maximum amount possible from the source to the sink. In doing so we will construct a *flow*.
A flow will tell each pipe how much stuff will pipe.

So a flow will be specified by a set of numbers $x_{i,j}$ which tell how much stuff the pipe from i to j will pipe.

Flows have to satisfy following three constrains.

(1). First, a flow can't have a pipe piping more than it can pipe, so we will insist that $x_{i,j} \leq c_{i,j}$ for all valid i and j

(2). Pipes can't pipe negative amounts, i.e., $x_{i,j} \geq 0$ for all valid i and j.

(3). The final constraint is known as Kirchoff's Law. It says that the amount flowing into a vertex must equal the amount flowing out of the vertex.

- The amount flowing out of vertex i is given by $\Sigma_{k=1}^{n} x_{i,k}$, where n is the number of outward vertices of vertex i, and the amount flowing into vertex i is given by $\Sigma_{k=1}^{m} x_{k,i}$, where m is the number of inward vertices of vertex i.
- So in mathematical terms Kirchoff's Law says, $\Sigma_{k=1}^{n} x_{i,k} = \Sigma_{k=1}^{m} x_{k,i}$ for all valid i except for the source and the sink.

(3). The final constraint is known as Kirchoff's Law. It says that the amount flowing into a vertex must equal the amount flowing out of the vertex.

- The amount flowing out of vertex i is given by $\Sigma_{k=1}^{n} x_{i,k}$, where n is the number of outward vertices of vertex i, and the amount flowing into vertex i is given by $\Sigma_{k=1}^{m} x_{k,i}$, where m is the number of inward vertices of vertex i.
- So in mathematical terms Kirchoff's Law says, $\Sigma_{k=1}^{n} x_{i,k} = \Sigma_{k=1}^{m} x_{k,i}$ for all valid i except for the source and the sink.

Outline

- We start with no flow at all, that is, with every $x_{i,j}$ set equal to 0.
- Then we find what is called an *augmenting path* from the source to sink.
- This is, as it says, a path from source to the sink, that has excess capacity.
- We then figure out how much more we could pipe down that path and add this to the flow we are building.

- We start with no flow at all, that is, with every $x_{i,j}$ set equal to 0.
- Then we find what is called an *augmenting path* from the source to sink.
- This is, as it says, a path from source to the sink, that has excess capacity.
- We then figure out how much more we could pipe down that path and add this to the flow we are building.

- We start with no flow at all, that is, with every $x_{i,j}$ set equal to 0.
- Then we find what is called an *augmenting path* from the source to sink.
- This is, as it says, a path from source to the sink, that has excess capacity.
- We then figure out how much more we could pipe down that path and add this to the flow we are building.

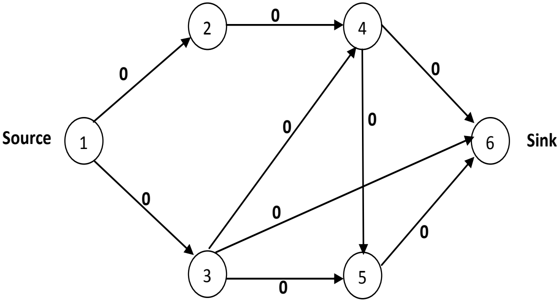
- We start with no flow at all, that is, with every $x_{i,j}$ set equal to 0.
- Then we find what is called an *augmenting path* from the source to sink.
- This is, as it says, a path from source to the sink, that has excess capacity.
- We then figure out how much more we could pipe down that path and add this to the flow we are building.

Example

We will apply the algorithm to the network (that we have given previously) and start with the flow set equal to 0 everywhere.

Example

We will apply the algorithm to the network (that we have given previously) and start with the flow set equal to 0 everywhere.



Example

- We need to find an augmenting path.
- Let take the path $1 \rightarrow 3 \rightarrow 6$ to start with.
- The pipe from $1 \rightarrow 3$ has capacity 9 ($c_{1,3} = 9$), and is currently piping 0 ($x_{1,3} = 0$ right now), so it could pipe 9 more.
- The pipe from $3 \rightarrow 6$ has capacity 2 and is currently piping 0, so it could pipe 2 more.
- We can only pipe the minimum of these numbers, so we will only be able to send 2 units of stuff down this path.

Example

- We need to find an augmenting path.
- Let take the path $1 \rightarrow 3 \rightarrow 6$ to start with.
- The pipe from $1 \rightarrow 3$ has capacity 9 ($c_{1,3} = 9$), and is currently piping 0 ($x_{1,3} = 0$ right now), so it could pipe 9 more.
- The pipe from $3 \rightarrow 6$ has capacity 2 and is currently piping 0, so it could pipe 2 more.
- We can only pipe the minimum of these numbers, so we will only be able to send 2 units of stuff down this path.

Example

- We need to find an augmenting path.
- Let take the path $1 \to 3 \to 6$ to start with.
- The pipe from $1 \to 3$ has capacity 9 ($c_{1,3} = 9$), and is currently piping 0 ($x_{1,3} = 0$ right now), so it could pipe 9 more.
- The pipe from $3 \to 6$ has capacity 2 and is currently piping 0, so it could pipe 2 more.
- We can only pipe the minimum of these numbers, so we will only be able to send 2 units of stuff down this path.

Example

- We need to find an augmenting path.
- Let take the path $1 \to 3 \to 6$ to start with.
- The pipe from $1 \to 3$ has capacity 9 ($c_{1,3} = 9$), and is currently piping 0 ($x_{1,3} = 0$ right now), so it could pipe 9 more.
- The pipe from $3 \to 6$ has capacity 2 and is currently piping 0, so it could pipe 2 more.
- We can only pipe the minimum of these numbers, so we will only be able to send 2 units of stuff down this path.

Example

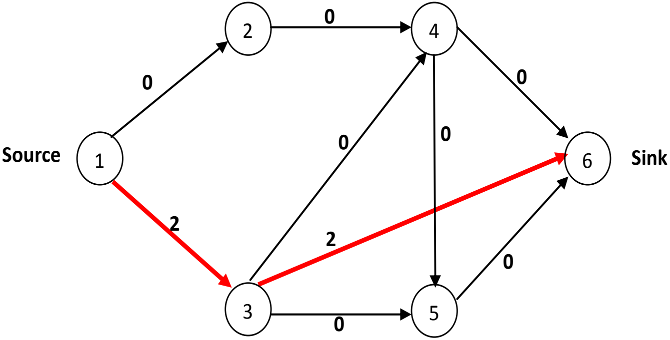
- We need to find an augmenting path.
- Let take the path $1 \to 3 \to 6$ to start with.
- The pipe from $1 \to 3$ has capacity 9 ($c_{1,3} = 9$), and is currently piping 0 ($x_{1,3} = 0$ right now), so it could pipe 9 more.
- The pipe from $3 \to 6$ has capacity 2 and is currently piping 0, so it could pipe 2 more.
- We can only pipe the minimum of these numbers, so we will only be able to send 2 units of stuff down this path.

Example

Then we update the flow:

Example

Then we update the flow:



Example

- Now repeat above steps. Let's take the augmenting path $1 \rightarrow 3 \rightarrow 5 \rightarrow 6$. For each are we check how much spare capacity it has:

Example

- Now repeat above steps. Let's take the augmenting path $1 \to 3 \to 5 \to 6$. For each are we check how much spare capacity it has:

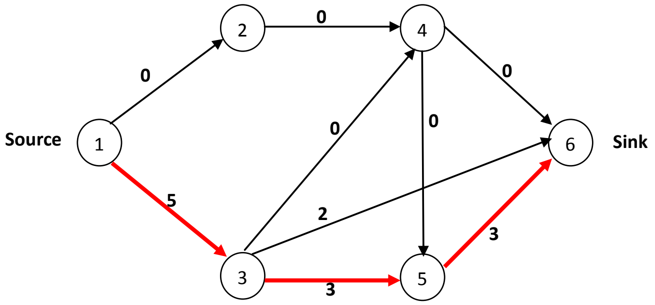
arc	Total capacity		current load		excess capacity
1→3	9	-	2	=	7
3 → 5	3	-	0	=	3
5 → 6	9	-	0	=	9

Example

The smallest excess capacity is 3, so that's what we'll add to the flow, which is shown below.

Example

The smallest excess capacity is 3, so that's what we'll add to the flow, which is shown below.

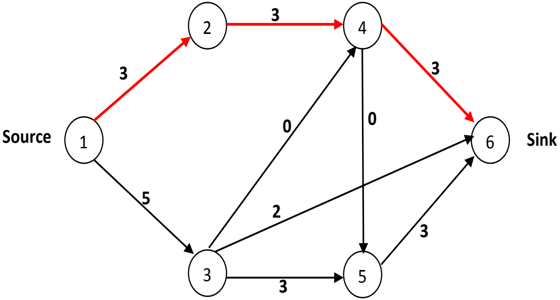


Example

- Now, we can find another augmenting path. This time let's take 1 $\rightarrow$ 2 $\rightarrow$ 4 $\rightarrow$ 6. The excess capacities of these arcs are 5, 4, and 3, respectively, so the most we can send down this way is 3.

Example

- Now, we can find another augmenting path. This time let's take 1 $\rightarrow$ 2 $\rightarrow$ 4 $\rightarrow$ 6. The excess capacities of these arcs are 5, 4, and 3, respectively, so the most we can send down this way is 3.



Example

- it's getting little harder to spot augmenting paths now, but there's at least one more: $1 \to 2 \to 4 \to 5 \to 6$. We have,

Example

- it's getting little harder to spot augmenting paths now, but there's at least one more: $1 \to 2 \to 4 \to 5 \to 6$. We have,

arc	Total capacity		current load		excess capacity
1→2	5	-	3	=	2
2 → 4	4	-	3	=	1
4 → 5	1	-	0	=	1
5 → 6	9	-	3	=	6

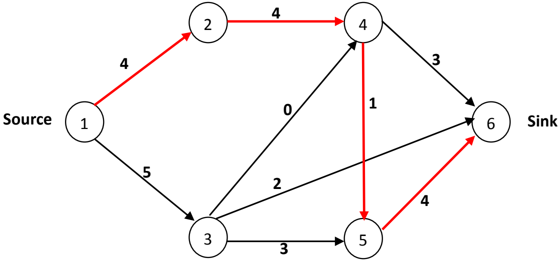
Example

- This shows that we can send one unit of stuff down this path. The updated flow is,

- Now we are managing to get 9 units from the source to the sink. But, is the best we can do?

Example

- This shows that we can send one unit of stuff down this path. The updated flow is,



- Now we are managing to get 9 units from the source to the sink. But, is the best we can do?

Outline

1. Networks and Flows
2. Ford-Fulkerson Algorithm
3. Minimum Cut
4. Max-Flow Min-Cut

How you know when you are done?

Trying to move stuff from the source to the sink depends very much on the paths from the source to the sink. It depends on the ways in which those paths can be cut off.

Definition

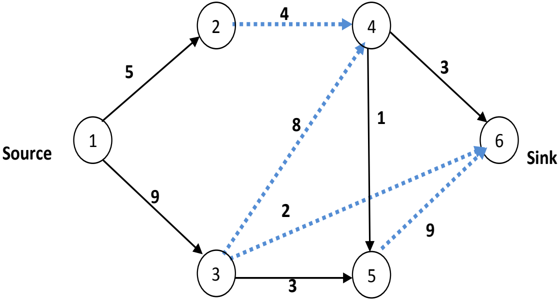
A cut in a network (or just in a digraph) is a set of arcs such that if they are removed, there is not path from the source to the sink.

Definition

The capacity of a cut is defined to be the sum of the capacities of every arc in the cut.

So to figure out the capacity of a cut, we want to look at the original capacity graph, **not** the flow graph we may or may not have just made. For example below capacity graph has capacity in a cut $4 + 8 + 2 + 9 = 23$.

So to figure out the capacity of a cut, we want to look at the original capacity graph, **not** the flow graph we may or may not have just made. For example below capacity graph has capacity in a cut $4 + 8 + 2 + 9 = 23$.



Outline

Max-Flow Min-Cut Theorem

Theorem

In every network, the maximum flow equals the minimum capacity of a cut.

- This theorem was proved in 1956 independently by Ford and Fulkerson and by Feinstein and Shannon. The proof by Ford and Fulkerson is a very straight-forward way.The theorem can also be proved by applying the Duality Theorem from Linear Programming.
- The only problem with the Max-Flow Min-Cut Theorem is that the two quantities is says are equal are both hard to get handle on. But, if you can find a flow and cut with the same value, you are done.

Max-Flow Min-Cut Theorem

Theorem

In every network, the maximum flow equals the minimum capacity of a cut.

- This theorem was proved in 1956 independently by Ford and Fulkerson and by Feinstein and Shannon. The proof by Ford and Fulkerson is a very straight-forward way.The theorem can also be proved by applying the Duality Theorem from Linear Programming.
- The only problem with the Max-Flow Min-Cut Theorem is that the two quantities is says are equal are both hard to get handle on. But, if you can find a flow and cut with the same value, you are done.

Let's move to our example.

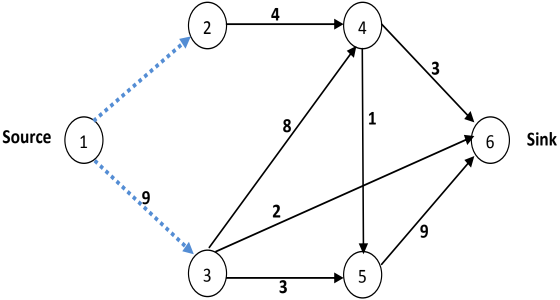
- The cut we found had capacity 23, so there must be a smaller cut.
- Let's keep playing around with the cuts for a while, and see if we can make a better one.
- Here's cut with capacity $5 + 9 = 14$

Let's move to our example.

- The cut we found had capacity 23, so there must be a smaller cut.
- Let's keep playing around with the cuts for a while, and see if we can make a better one.
- Here's cut with capacity $5 + 9 = 14$

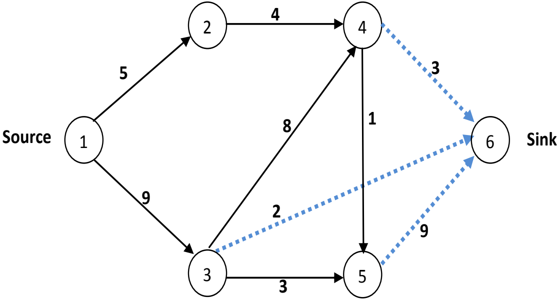
Let's move to our example.

- The cut we found had capacity 23, so there must be a smaller cut.
- Let's keep playing around with the cuts for a while, and see if we can make a better one.
- Here's cut with capacity $5 + 9 = 14$



And here's another cut with capacity $3 + 2 + 9 = 14$

And here's another cut with capacity $3 + 2 + 9 = 14$



Finally, here's a tricky cut with capacity $2 + 1 + 3 + 3 = 9$

Finally, here's a tricky cut with capacity $2+1+3+3=9$