Max-Flows

Lecture 2, MS&E 339, Spr 2022

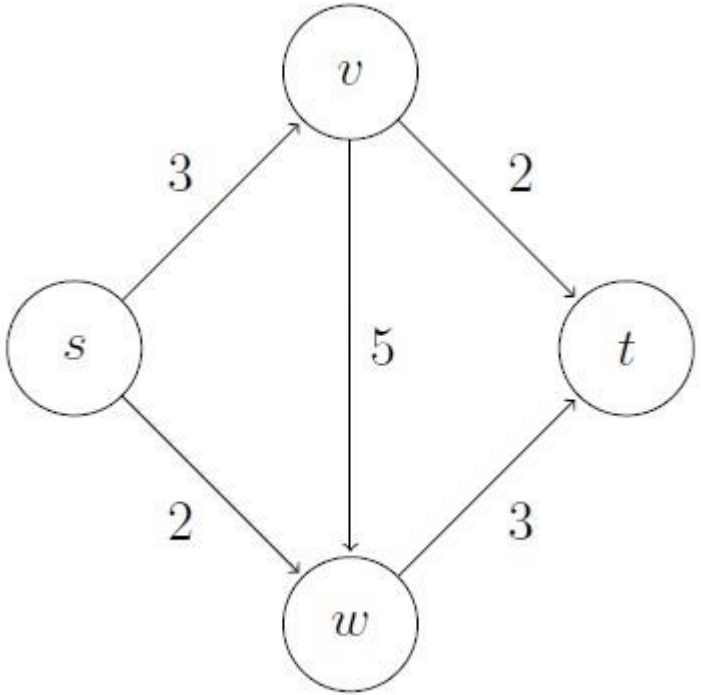
Ashish Goel, Stanford University

Informal description

Network of unidirectional pipes, with capacities.

Max-Flow: How much “flow” can be sent from one special node (the source s) to another (the terminus t).

The terminus is also often called the sink.



The labels on the edges are capacity.

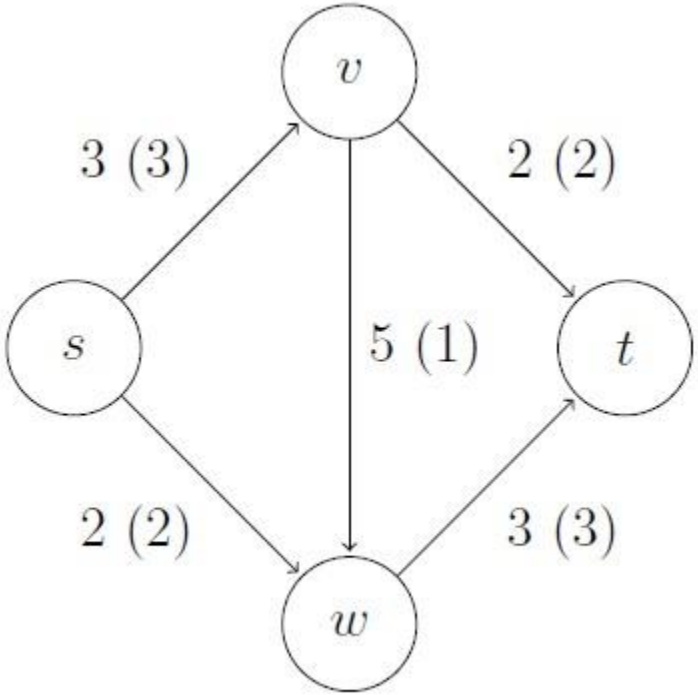
(Uses images and notation from notes by Tim Roughgarden)

Informal description

Network of unidirectional pipes, with capacities.

Max-Flow: How much “flow” can be sent from one special node (the source s) to another (the terminus t).

The terminus is also often called the sink.



The labels on the edges are capacity, and in brackets, flow.

Formal description

GIVEN:

A directed Graph G = (V,E)

(Source, Sink) nodes (s, t)

A non-negative (and often integral) capacity u(e) for every edge e.

Goal: to find the feasible flow f(e) for every edge e such that the amount of flow from s to t is maximized.

FEASIBLE FLOW:

(1) CAPACITY CONSTRAINT:

$f(e) \leq u(e)$

(2) CONSERVATION CONSTRAINT:

For all $v \neq s,t$: total flow on edges coming into v = total flow on edges going out of v

OBJECTIVE:

(Total flow on edges going out of s) - (Total flow on edges coming into s)

Formal description

GIVEN:

A directed Graph G = (V,E)

(Source, Sink) nodes (s, t)

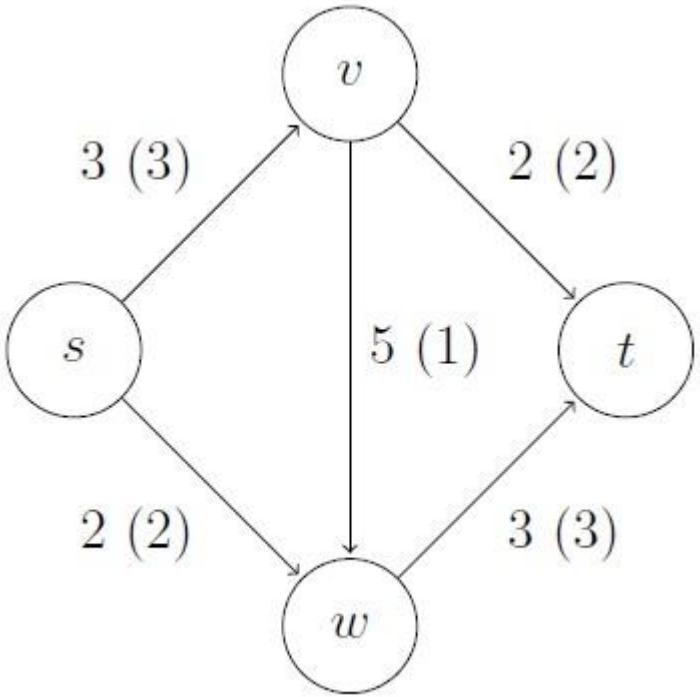
A non-negative (and often integral) capacity u(e) for every edge e.

Goal: to find the feasible flow f(e) for every edge e such that the amount of flow from s to t is maximized.

MNEMONIC:

$$IN_f(v) = \sum_{w:\ (w,v) \in E} f(w,v)$$

$$OUT_f(v) = \sum_{w:\ (v,w) \in E} f(v,w)$$



$IN_f(w)$:

$OUT_f(v)$:

Notation: Don’t confuse this v and w with the v and w in the definition.

Make sure you know notation such as for all, belongs to, set containment, there exists, etc.

Formal description

GIVEN:

A directed Graph G = (V,E)

(Source, Sink) nodes (s, t)

A non-negative (and often integral) capacity u(e) for every edge e.

Goal: to find the feasible flow f(e) for every edge e such that the amount of flow from s to t is maximized.

FEASIBLE FLOW:

(1) CAPACITY CONSTRAINT:

$f(e) \leq v(e)$

(2) CONSERVATION CONSTRAINT:

$\forall\ v \in V,\ v \neq s,t$:

$IN_f(v) = OUT_f(v)$

OBJECTIVE:

Maximize $OUT_f(s) - IN_f(s)$

Naive Algorithm

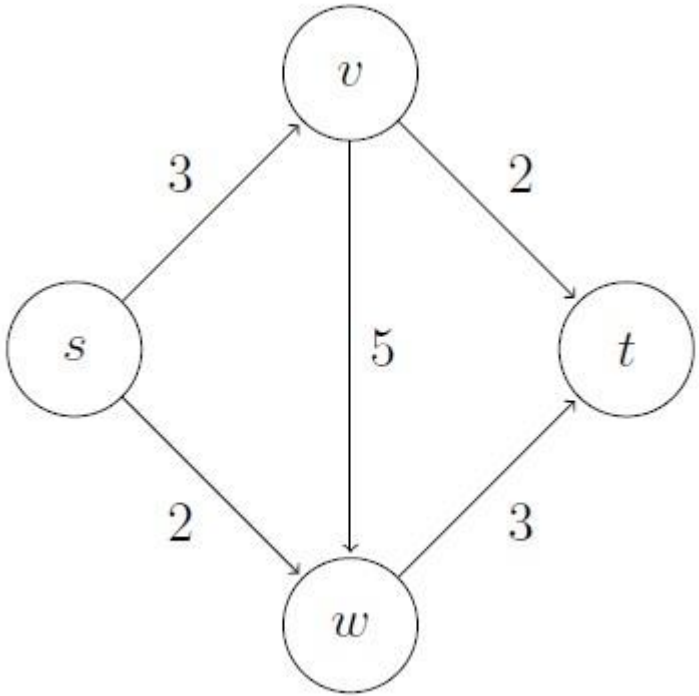
Initialize:

f(e) = 0 for all e

Repeat:

Find a path with spare capacity from s to t (using bfs, dfs, etc.)

Send as much flow as possible along this path.



Naive Algorithm — counter example

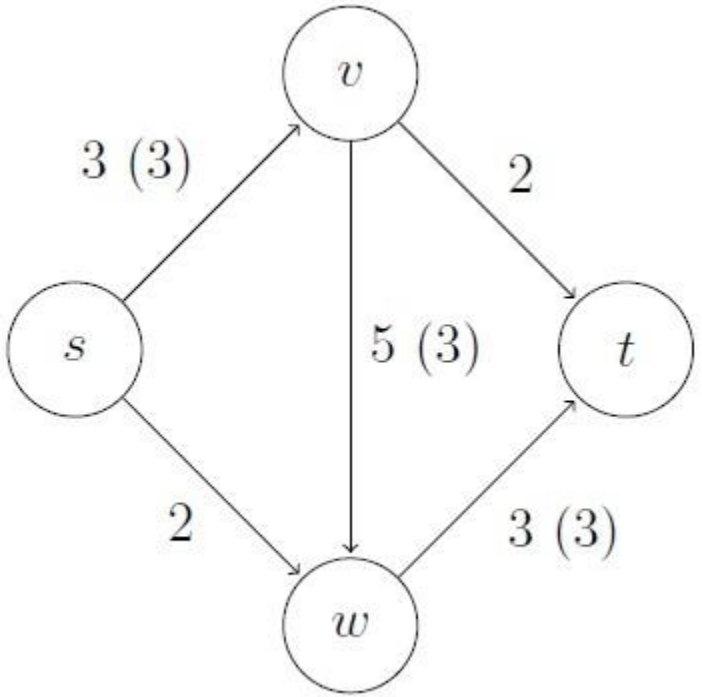
Initialize:

f(e) = 0 for all e

Repeat:

Find a path with spare capacity from s to t (using bfs, dfs, etc.)

Send as much flow as possible along this path.

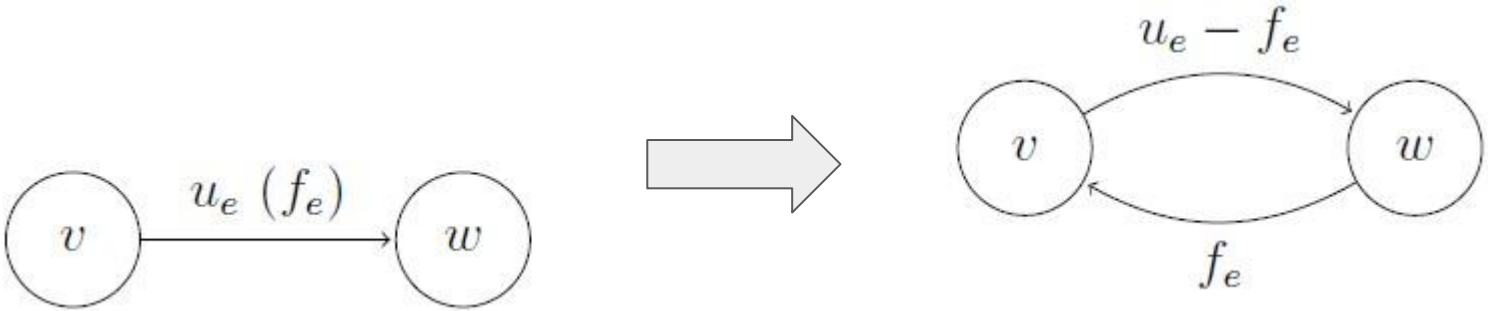


Residual Graph

Residual capacity of an edge e: **u(e) - f(e)** in the forward direction, **f(e)** in the backward direction.

Captures how much more flow can be sent in the forward direction, and how much of previously sent flow can be undone.

Residual graph G_f: Replace the capacity by the residual capacity (and introduce back edges as needed)



Fixing the Naive Algorithm

FORD-FULKERSON

Initialize:

f(e) = 0 for all e

Repeat:

Find a path with spare capacity from s to t (using bfs, dfs, or anything) in the **residual graph**

Send as much flow as possible along this path (*Augmentation)*.

Fixing the Naive Algorithm

FORD-FULKERSON

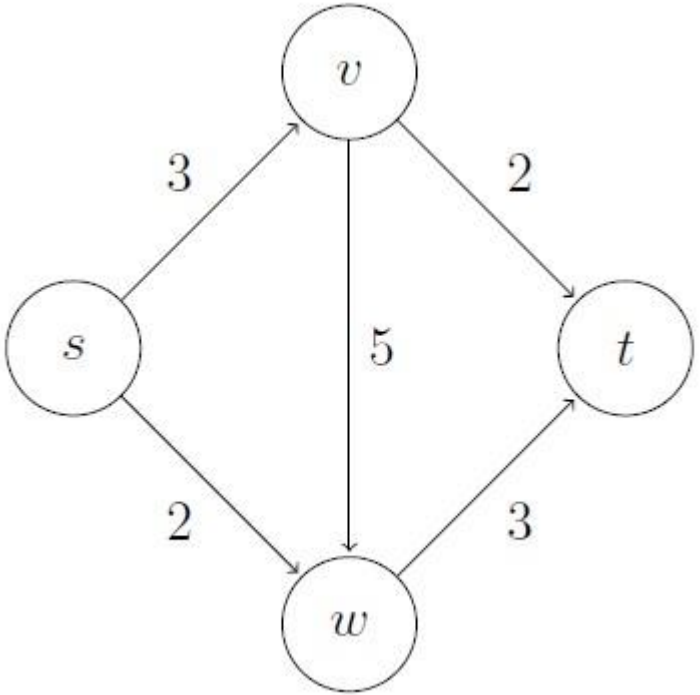
Initialize:

f(e) = 0 for all e

Repeat:

Find a path with spare capacity from s to t (using bfs, dfs, or anything) in the **residual graph**

Send as much flow as possible along this path (*Augmentation)*.



Fixing the Naive Algorithm

FORD-FULKERSON

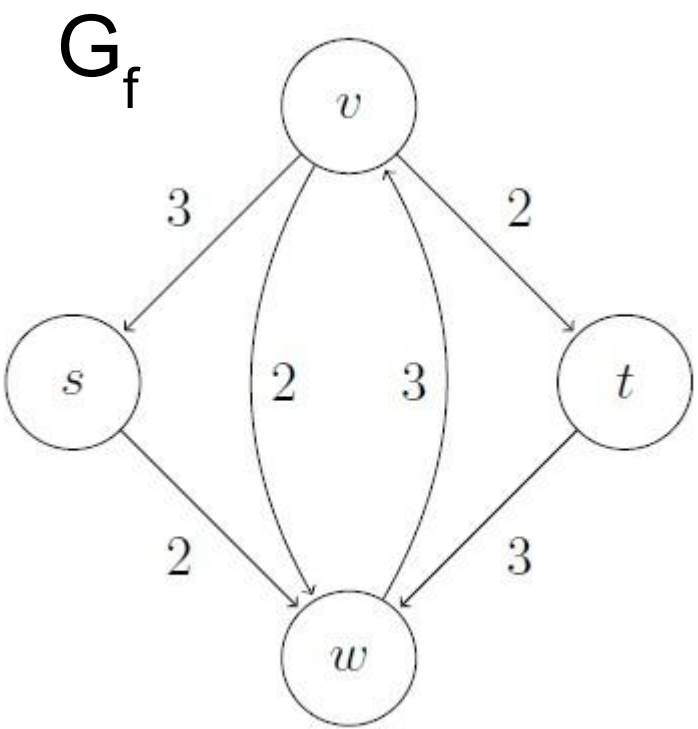
Initialize:

f(e) = 0 for all e

Repeat:

Find a path with spare capacity from s to t (using bfs, dfs, or anything) in the **residual graph**

Send as much flow as possible along this path (*Augmentation)*.



Ford -Fulkerson Running Time

Must terminate if capacities are integer, since the amount of flow from s to t increases by at least one in every step.

Not polynomial time. Why not?

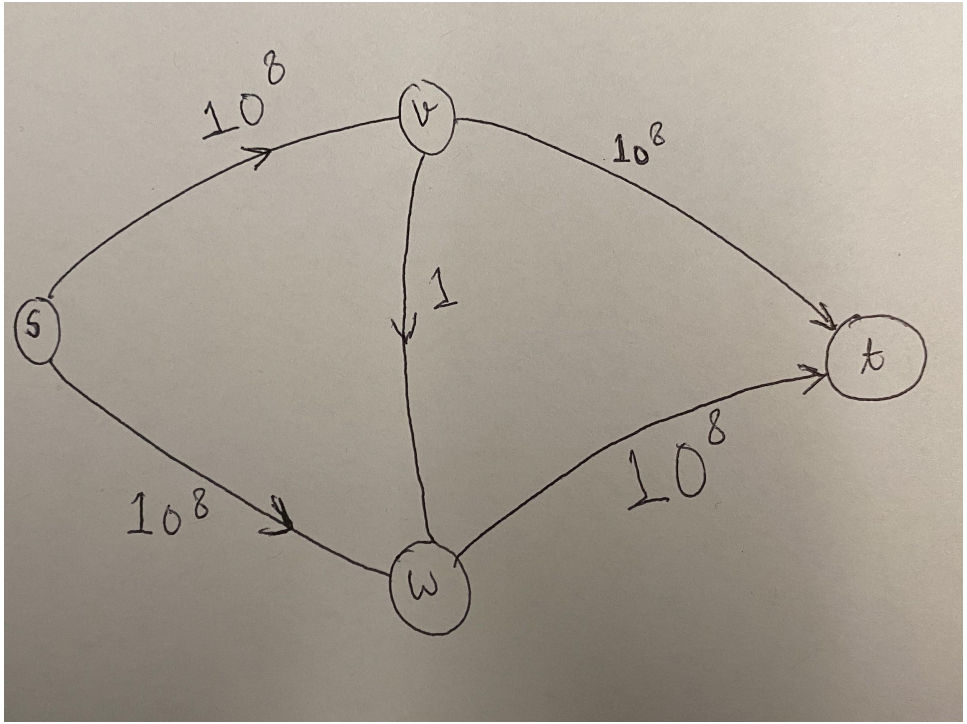
The algorithm is highly underspecified. Benefits and Problems?

Ford -Fulkerson Running Time

Must terminate if capacities are integer, since the amount of flow from s to t increases by at least one in every step.

Not polynomial time. Why not?

The algorithm is highly underspecified. Benefits and Problems?



Ford-Fulkerson Correctness

(1) Always finds a feasible flow. Trivial.
(2) Finds the maximum flow possible: quite deep.

Claim: If there is no augmentation possible in the residual graph, then we have found a maximum flow.

FF is correct if we prove the above claim.

Ford-Fulkerson Correctness

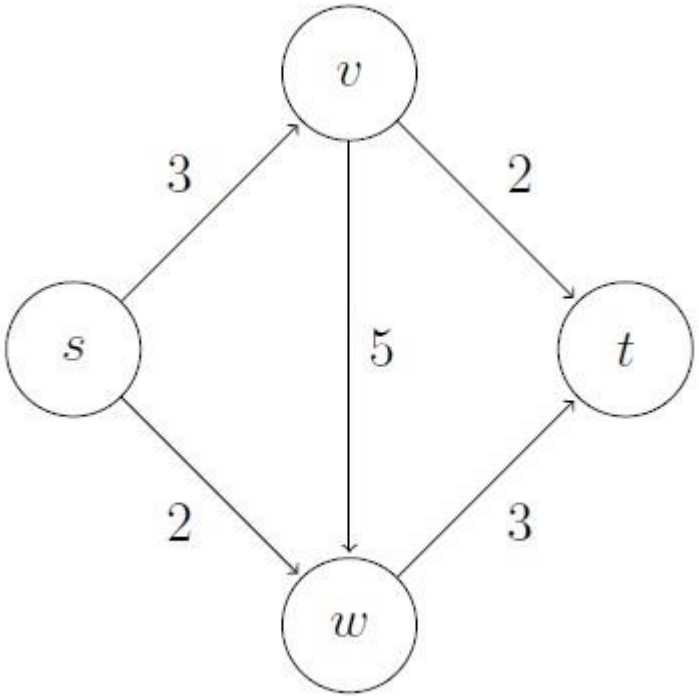
Claim: If there is no augmentation possible in the residual graph, then we have found a maximum flow.

Proof: Via the celebrated max-flow min-cut theorem.

An s-t cut of G: a partition of the nodes V into two parts, A and B = V - A such that s is in A and t is in B

Size of an s-t cut (A,B): total capacity of all the edges that go from a node in A to a node in B.

Minimum s-t-cut of G: The minimum possible capacity of an s-t cut of G.



A	B	Cut size
{s}	{v, w, t}	
{s, v}	{w, t}	
{s, w}	{v, t}	
{s, v, w}	{t}	

Max-Flow Min-Cut Theorem

Any path from s to t must leave a node in A to go to a node in B at least once.

Hence, Maximum s-t flow in G ≤ Minimum size of s-t cut in G. (Size of any flow from s to t is less than the size of any s-t cut).

Modified claim: If there is no augmentation possible in the residual graph, then we can find an s-t cut (A,B) such that the amount of flow is at least the size of the cut.

If we prove this claim, we would have proved both the **correctness of FF** and the proof of the max-flow min-cut theorem, which says:

Maximum s-t Flow in G = Minimum size of s-t cut in G

Summary of Chain of Reasoning

Assume that we prove: *If there is no augmentation possible in the residual graph, then we can find an s-t cut (A,B) such that the amount of flow is at least the size of the cut.*

Then we have found a valid s-t flow which is no smaller than the size of an s-t cut.

But any valid s-t flow must be no larger than any s-t cut. Hence this valid s-t flow must be exactly the size of the minimum s-t cut and must be the maximum flow.

Hence

(a) FF finds the maximum flow

(b) We have proved the max-flow Min-cut theorem

Continuous time Markov Chains: Informal

Finite state space S = {1, 2, …, N} with N states, Transition Rate matrix R of size NxN

If state of the system is j at time t, then in the limit as $\Delta t \rightarrow 0$, the probability that the state of the system is k at time $t + \Delta t$ is given by $R(j,k)\Delta t$.

We will assume that the Markov Chain is ergodic (in our case, there is a path of non-zero rate from any state j to any other state k).

Stationary distribution: P(k) = probability that system is in state k at time t as t tends to infinity; does not depend on the state of the system at time 0.

Important property of ergodic Markov Chains: If R(j,k) = R(k,j) for all j, k, then P(k) = 1/N for all states k.

The next four slides are not part of the class — they are provided for completeness and in case someone wants to dig into it more

Finishing off the Proof of Max-Flow Min-Cut

Still need to show: *If there is no augmentation possible in the residual graph, then we can find an s-t cut (A,B) such that the amount of flow is at least the size of the cut.*

PROOF:

(1) Let A be the set of nodes reachable from s in the residual graph G_f when FF terminates and B = V-A. Then (A, V-A) is a valid cut.

Proof: Sink t is not reachable from s in the final residual graph

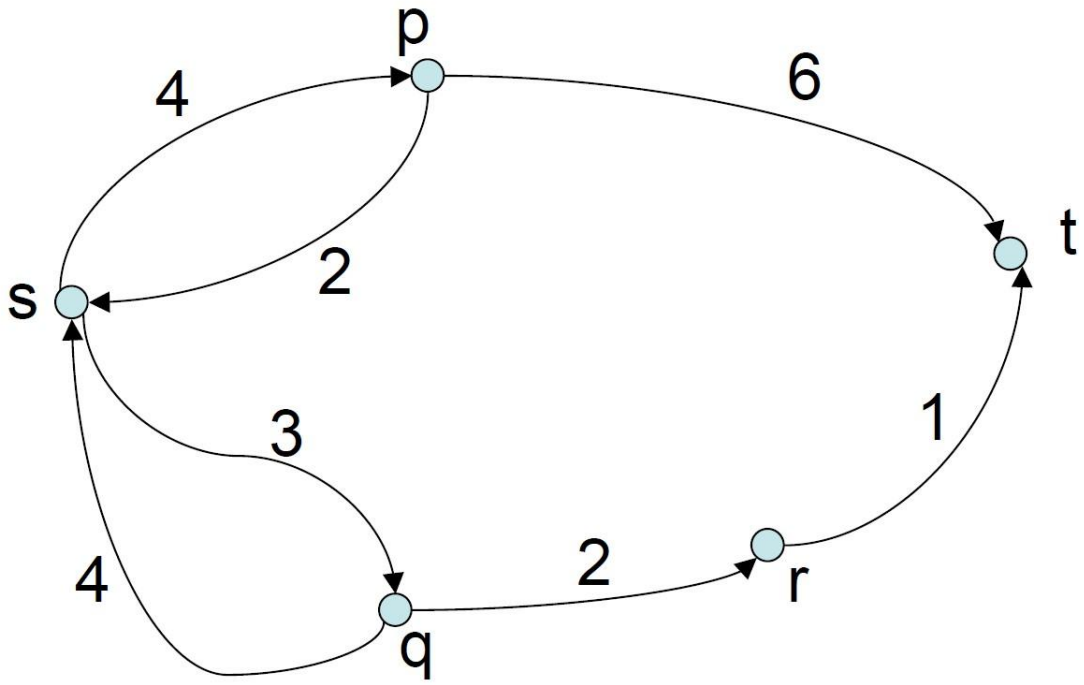
(2) All the edges going from a node v in A to a node w in B must be saturated, i.e. f(v,w) = u(v,w) and all the edges from B to A must have no flow on them (*in the original graph).*

Proof: An unsaturated edge from A to B or a non-zero flow on edge from B to A *in the original graph* will result in an edge from A to B *in the residual graph.*

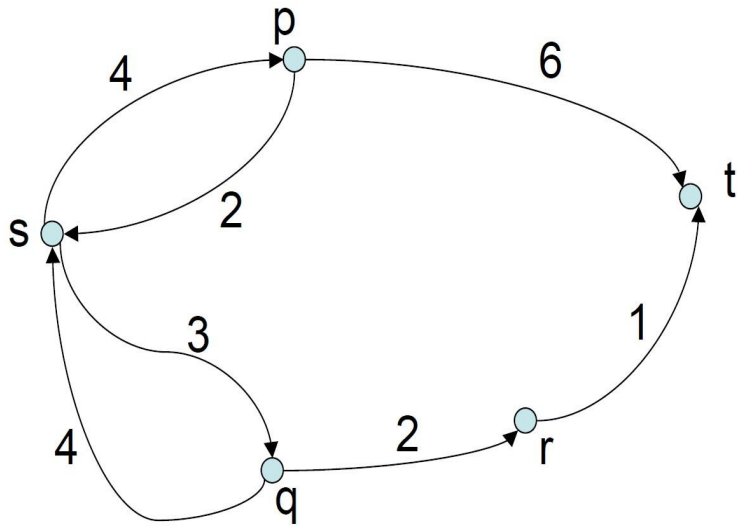
Finishing off the Proof

(1) Let A be the set of nodes reachable from s in the residual graph G_f when FF terminates. Then (A, V-A) is a valid cut.

(2) All the edges going from a node v in A to a node w in B must be saturated, i.e. f(v,w) = u(v,w) and all the edges from B to A must have no flow on them (*in the original graph).*



Finishing off the Proof



Extend the definition of OUT and IN to sets of nodes, e.g.

$OUT_f(A) = \sum_{v \in A} OUT_f(v)$

Now:

(3) amount of current flow from s to t = $OUT_f(s) - IN_f(s)$

= $OUT_f(A) - IN_f(A)$ (since OUT - IN is zero for all other nodes in A)

= (Total flow on edges from A to B) - (Total flow on edges from B to A)

(since all edges internal to A get cancelled out)

= (Total capacity of edges from A to B) - (0) (from step 2))

= The size of the (A,B) cut

Standing Back

Leonhard Euler (via Yinyu Ye): “... nothing at all takes place in the Universe in which some rule of maximum or minimum does not appear.”

Max-Flow Min-Cut: An example of LP Duality from Optimization. This is easily one of the five most beautiful algorithmic theorems I have personally seen.

The chain of reasoning we have followed will not lead to the fastest algorithms, but the notion of augmentation has diverse applications, and also is useful for algorithmic modeling.