Lecture 17: April 19, 2001

Administrative:

- Quiz II handed out 1pm on Monday.
- No class Tuesday (due to Take-Home)
- No recitation Friday

Today: Networks and Network Flow.

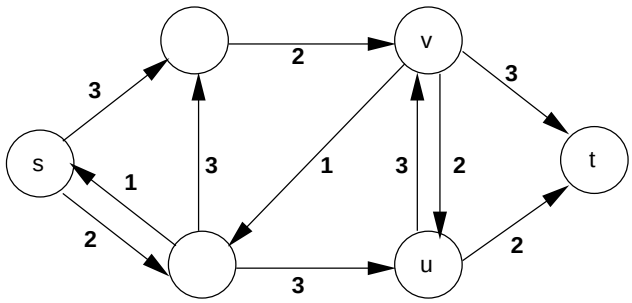
- Flow networks
- Positive flow
- Net flow
- Upper bound on (max) flow
- Residual networks
- Augmenting paths
- Max-flow Min-cut theorem
- Ford-Fulkerson algorithm
- Edmonds-Karp algorithm

Flow networks

Model fundamental notion of material being transported across finite-capacity channels

Examples: water, current, data, etc.

Single source/sink, multi-source/sink, "multi-commodity'



Model as **network** (digraph $G = (V, E)$) with:

"source" node s and "sink" node t

Non-negative **capacities** $c(u, v)$

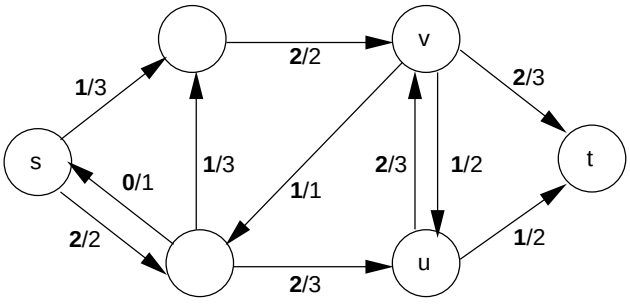
If $(u, v) \notin E$, then $c(u, v) = 0$.

Positive Flow on G

A *positive flow* is a function $p : V \times V \to \Re$ satisfying:

1. (Capacity constraint) $0 \le p(u, v) \le c(u, v)$ $\forall u, v \in V$
2. (Flow conservation) $\forall u \in V - \{s, t\}$,
$$\sum_{v \in V} p(u, v) - \sum_{v \in V} p(v, u) = \boxed{0} \qquad \text{(Kirchhoff)}$$

Example (notation is p/c or positive flow/capacity):



Note:

- All flows are $\le$ capacities
- Conservation of flow at vertices
- Setting all flows $= 0$ legitimate

Flow cancellation

WLoG, say positive flow $u \to v$ or $v \to u$, but not both!

If both, then transform by "cancellation":

$$\begin{array}{c} v \\ 2/3 \uparrow\downarrow 1/2 \\ u \end{array} \quad \Rightarrow \quad \begin{array}{c} v \\ 1/3 \uparrow\downarrow 0/2 \\ u \end{array}$$

"Net flow" from u to v is 1 **in both cases**

Note:

- capacity constraint still satisfied (because flows only decrease)
- flow conservation still satisfied (flow in, out reduced by same amount)

Net Flow on Digraph G

A function $f : V \times V \to \Re$ satisfying:

1. (Capacity constraint) $f(u,v) \le c(u,v)\ \forall u,v \in V$
2. (Flow conserv.) $\forall u \in V - \{s,t\}$, $\Sigma_{v \in V} f(u,v) = 0$
3. (Skew symmetry) $f(u,v) = -f(v,u)\ \forall u,v \in V$

Net flow, positive flow definitions **equivalent**:
Can define net in terms of positive:

- Define $f(u,v) = p(u,v) - p(v,u)$
- (f satisfies capacity constraint and flow conserv.)
- Skew symmetry trivially satisfied

Can define positive in terms of net:

- Define

$$p(u,v) = \begin{cases} f(u,v) & \text{if } f(u,v) > 0 \\ 0 & \text{if } f(u,v) \le 0 \end{cases}$$

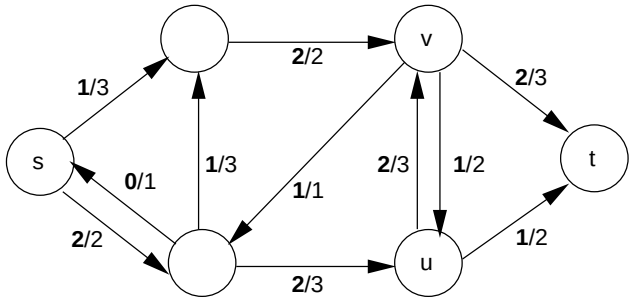
- (p satisfies capacity constraint and flow conserv.)

Maximum-Flow problem

Define **value** $|f|$ of flow f as net flow out of s.

$$|f| = \sum_{v \in V} f(s,v)$$

For G below, what is flow out of s? Into t?
Can you do better? Yes!



Flow networks correspond to instances of fundamental, real-world problems
Natural question: what is **max flow** of a network?
(Will see max-flow algorithm later.)

Implicit Summation Notation

- Functions over sets imply summation
  (because we will write lots of sums)
- Example: $|f| = f(s,V) \equiv \Sigma_{v \in V} f(s,v)$
- Conservation: $\forall u \in V - \{s,t\}$, $f(u,V) = 0$
- We will also do double sums:

$$f(X,Y) = \sum_{x \in X} \sum_{y \in Y} f(x,y)$$

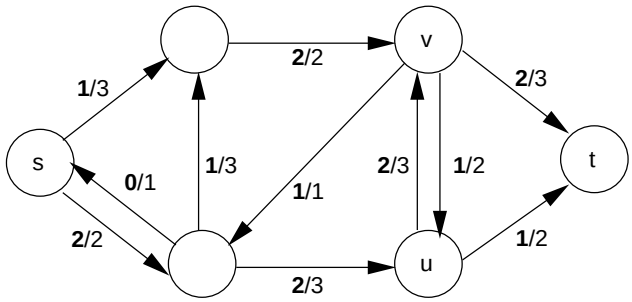
- Lemma 1:
  - $f(X,X) = 0$
    because each $f(u,v)$ canceled by $f(v,u)$
    (since $f(v,u) = -f(u,v)$ by skew symmetry)
  - $f(X,Y) = -f(Y,X)$
    (generalization of $f(X,X) = 0$)
  - $f(X \cup Y, Z) = f(X,Z) + f(Y,Z)$ if $X \cap Y = \emptyset$
    Sum of two summations: one over X, one over Y.

Sink lemma

Lemma 2: $|f| = f(V,t)$

$$\begin{aligned} |f| &= f(s,V) && \text{(Definition)} \\ &= f(V,V) - f(V-s,V) && \text{(Lemma 1)} \\ &= f(V,V-s) && \text{(Lemma 1)} \\ &= f(V,t) + f(V,V-s-t) && \text{(Lemma 1)} \\ &= f(V,t) && \text{(Conservation)} \end{aligned}$$

Interpretation:



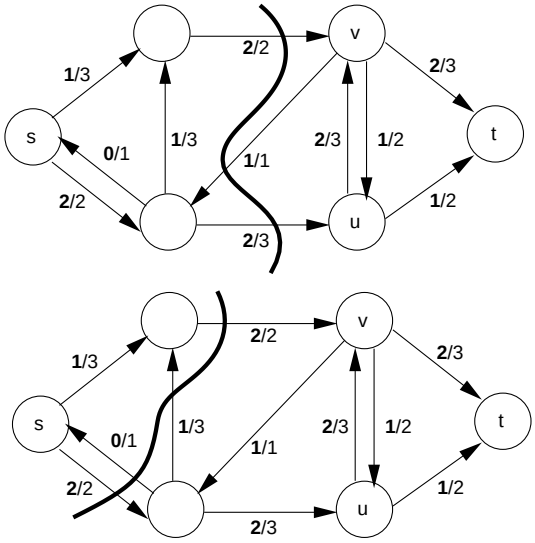
In any valid flow network, the amount of stuff leaving s must equal the amount of stuff entering t.

To show $f(V, V-s-t) = 0$, simply break it up into separate sums, one for each vertex not including s and t; each sum is 0 by conservation.

Flow across a cut

A **cut** (S, T) is a partition of V (i.e., $T = V - S$) such that $s \in S$ and $t \in T$

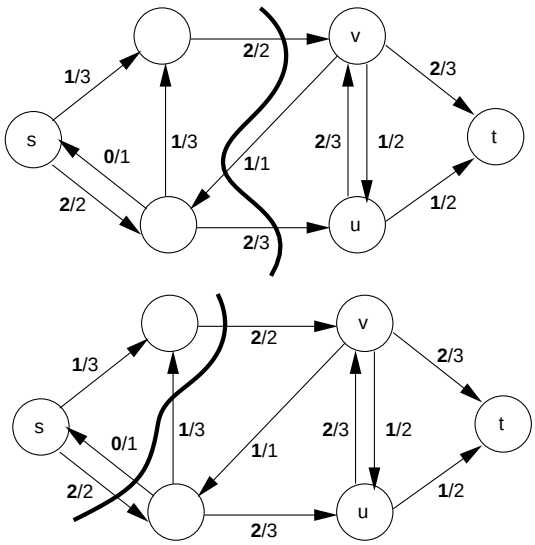
For flow f, the **flow across cut** (S, T) is $f(S, T)$.



What is $f(S, T)$ in each case above? $\boxed{4 - 1 = 3}$

Capacity across a cut

The **capacity across cut** is $c(S, T)$



Note:

Unlike flow (no skew symmetry)

Simply **add** capacities of all edges $S \to T$.

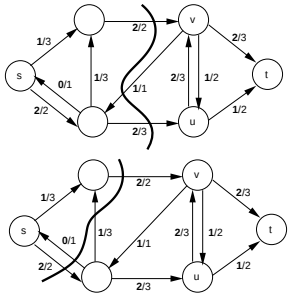
What is $c(S, T)$ in each case above? $\boxed{5;\ 4}$

Maximum Flow

- Lemma 3:

  For any flow f, the flow across *any* cut (S, T) is the flow value $|f|$.

  (Remember, by definition $s \in S$ and $t \in T$)



Proof:

$$\begin{aligned} f(S,T) &= f(S,V) - f(S,S) \\ &= f(S,V) \\ &= f(s,V) + f(S-s,V) \\ &= f(s,V) \\ &= |f| \end{aligned}$$

Flows and Cuts

Corollary: The value of any flow is bounded above by the capacity of **any** cut.

Proof: Let (S, T) be any cut, f any flow.

By Lemma 3 and capacity constraints,

$$\begin{aligned} |f| &= f(S,T) \\ &= \sum_{u \in S} \sum_{v \in T} f(u,v) \\ &\le \sum_{u \in S} \sum_{v \in T} c(u,v) \\ &= c(S,T) \ . \end{aligned}$$

Thus $|f| = f(S, T) \le c(S, T)$

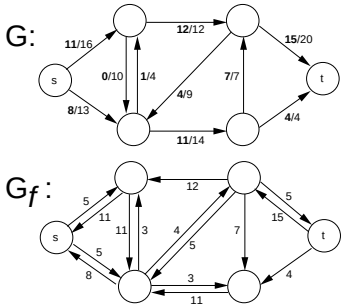
Max-flows and min-cuts are related, as are algorithms for computing them.

Residual Networks

Remember definition of **flow** on $G = (V, E)$:
amount of stuff moving across each channel

$$f(u, v) \leq c(u, v)$$

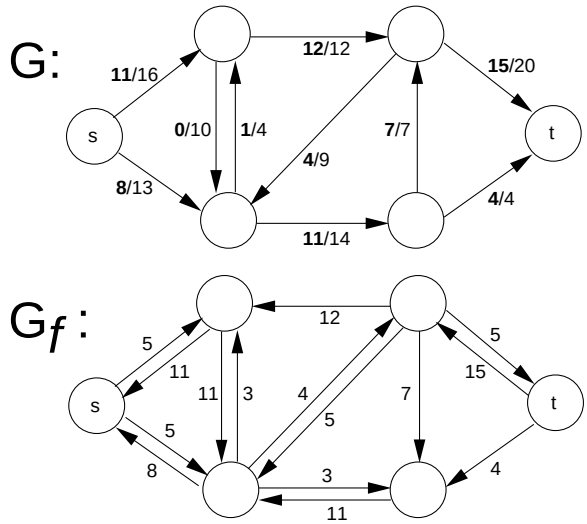
Define **residual network** $G_f = (V, E_f)$ of strictly positive **residual capacities**:



$$c_f(u, v) = c(u, v) - f(u, v) > 0$$
$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

Example: $f = 11$, $c = 16$. Then
$c_f(u, v) = \boxed{16 - 11 = 5}$, $c_f(v, u) = \boxed{0 - (-11) = 11}$.

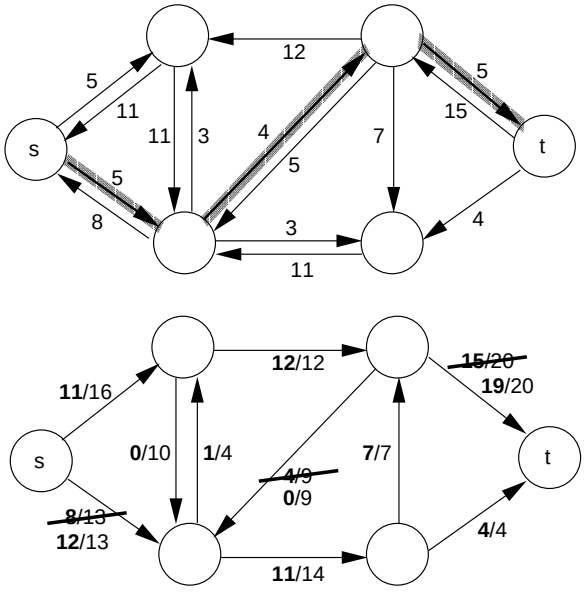
Residual Networks



Residual capacity can be $>$ capacity!
Example: $c = 10$ above, but $c_f = 11$ (Why?)
G_f contains edges that can admit more flow
Its edges come from $G \cup G^T$ (G, edge-reversed)
Thus $\boxed{|E_f| \leq 2\,|E|}$

Augmenting Path

An *augmenting path* is a path p from s to t in G_f
(or "in G with respect to f")
Increase flow in G by $c_f(p) = \min_{(u,v)\in p} c_f(u, v)$
Example:



Max-flow, Min-cut theorem

Theorem: the following are equivalent:

1. f is a maximum flow
2. f admits no augmenting path in G_f
3. $|f| = c(S, T)$ for some cut (S, T)

Proof:

- (1) $\Rightarrow$ (2): if augmenting path, could add flow from s, so couldn't have had max flow.
- (2) $\Rightarrow$ (3): Suppose f admits no augmenting path.
  Def. $S = s \cup \{v : \exists \text{ path } p \text{ from } s \text{ to } v \text{ with } c_f(p) > 0\}$
  Def. $T = V - S$. Note $t \in T$ (else aug. path)
  For each $u \in S$ and $v \in T$, $f(u, v) = c(u, v)$.

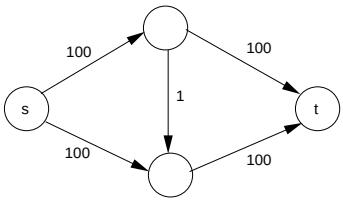
  $$\textcircled{s} \rightsquigarrow \textcircled{u} \longrightarrow \textcircled{v} \quad \textcircled{t}$$

  (O/wise $c_f(u, v) = c(u, v) - f(u, v) > 0 \Rightarrow v \in S$.)
  Then $|f| = f(S, T) = \Sigma_{u\in S}\, \Sigma_{v\in T}\, c(u, v) = c(S, T)$.
- (3) $\Rightarrow$ (1): $[|f| \leq c(S, T)$ for all cuts $(S, T)]$
  So $|f| = c(S, T)$ implies f is a maximum flow.

Ford-Fulkerson Algorithm

$f(u,v) \leftarrow 0 \ \forall u, v$
while $\exists$ augmenting path p in G_f
do augment f by $c_f(p)$

See previous graph, path example.
Notice: F-F doesn't specify *which path*!



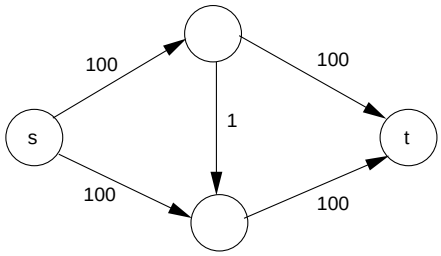
Analysis (with Integer Capacities):

- Cost per path: $O(E)$ using DFS or BFS Note: no V term, Since we can stay in connected component containing s, thus $E \geq V - 1$.
- Thus run time is $O(E\ |f^*|)$, where f^* is max flow
- This is *not* polynomial in $|V|$, $|E|$!

A More Efficient Max Flow Algorithm

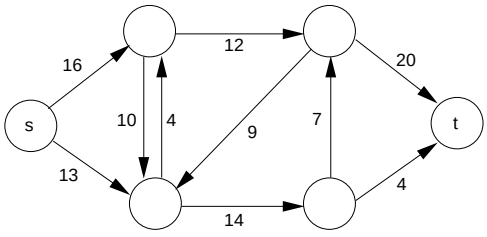
Augment along breadth-first (shortest) aug. path
This is the "Edmonds-Karp algorithm";
it runs in $O(VE^2)$ time

Same example:



Edmonds-Karp Analysis

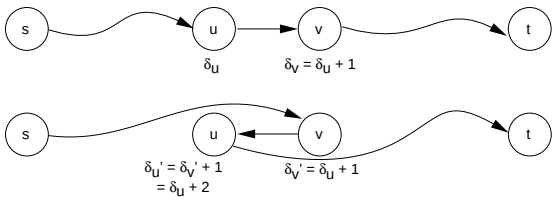
Lemma:
Let $\delta(v) = \delta_f(s,v) =$
B-F distance from s to v in G_f
Then $\delta(v)$ increases monotonically during E-K.



Intuition: Each time flow pushed, some
edge (route) of G_f is saturated (blocked).
Thus, it's at least as hard to get to v
in $G_{f'}$ as in G_f!
Much more complex proof in CLR.

Edmonds-Karp Analysis

E-K performs at most $O(V \cdot E)$ augmentations.
Intuition: think about **critical** edges for path p (I.e., those for which $c_f(p) = c_f(u,v)$)



When flow is augmented, edge (u,v) disappears;
Can't return until (v,u) appears on aug. path
Next time (u,v) appears, $\delta(s,u)$ larger by 2
But no path can have more than V vertices –
Thus (u,v) can become critical at most $V/2$ times!
Finally – at most $O(E)$ edges in residual graph
Thus total number (throughout algorithm) of critical edges is $O(VE)$; E-K does at most this many augs.
E-K Running time:
Use BFS – $O(E)$ time for each iteration
Then E-K algorithm runs in $O(VE^2)$ time.