CS 314 - Network Flow part 2

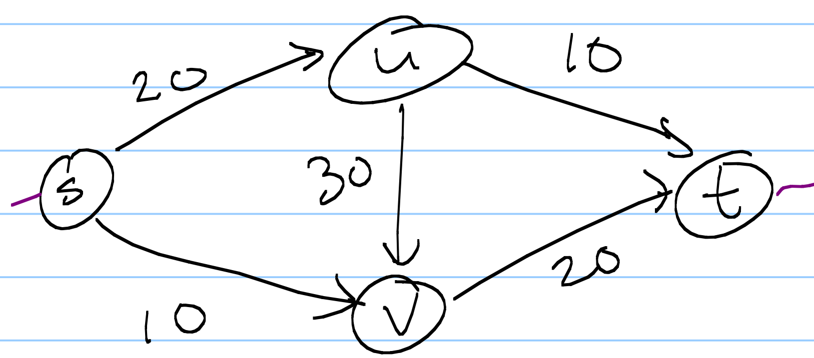
Note Title 4/16/2010

Announcements

- HW due (written) next Wednesday
  in class

Network Flow

- A directed graph $G=(V,E)$
- Each edge has a maximum capacity c_e
- Two special vertices $s, t \in V$
  - s is the source
  - t is the sink



Note: s has no incoming edges, & t has no outgoing

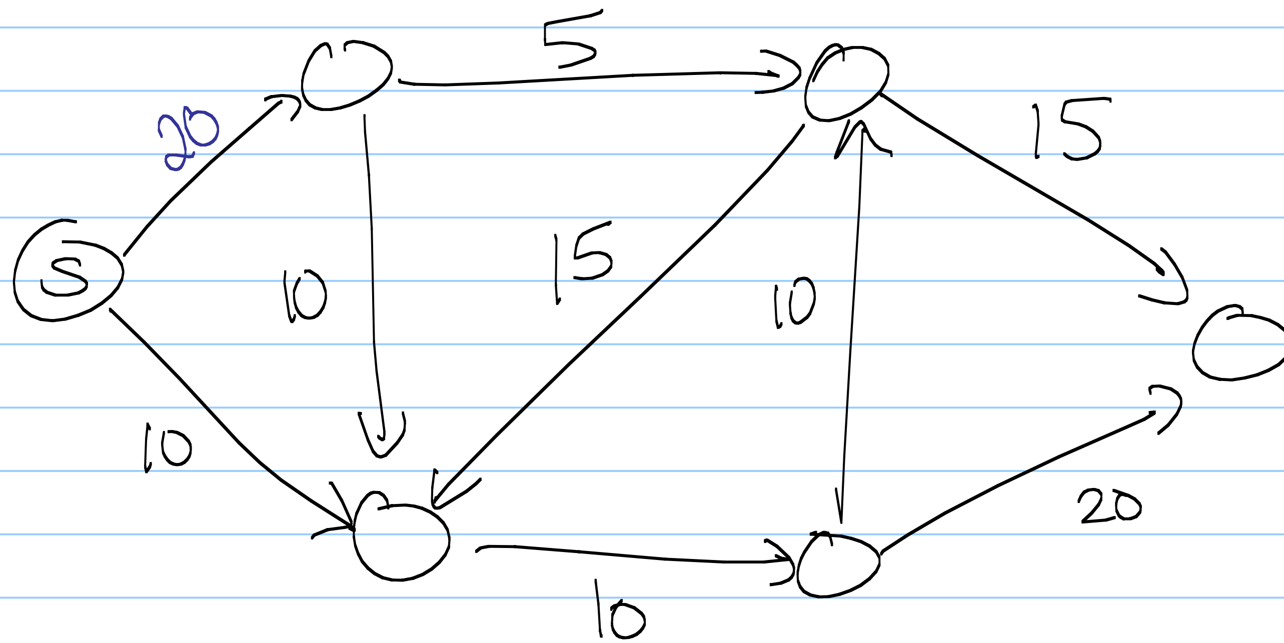
Formally:

A **flow** is a function $f: E \to \mathbb{R}^+$ (some amount sent along each edge) such that:

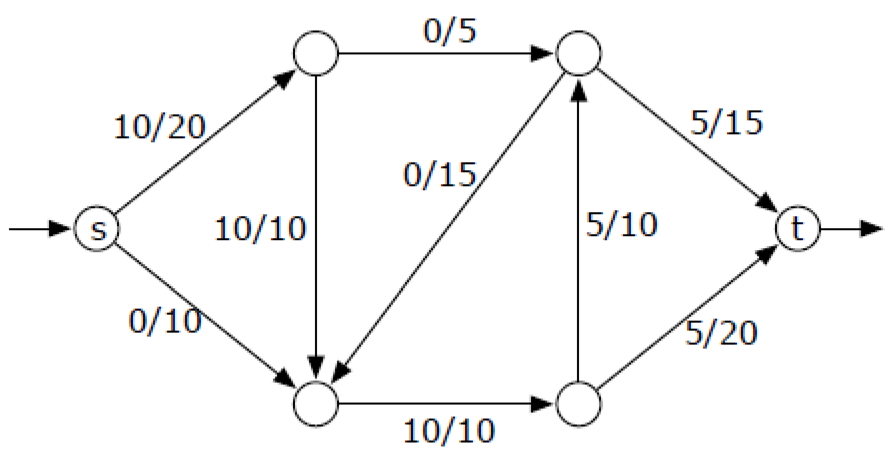
- capacity constraint: $\forall e \in E,\ 0 \le f(e) \le c_e$
- conservation constraint: $\forall v \in V$, if $v \ne s$ or t,

$$\underbrace{\sum_{e \text{ into } v} f(e)}_{f^{in}(v)} = \underbrace{\sum_{e \text{ out of } v} f(e)}_{f^{out}(v)}$$

So: graph



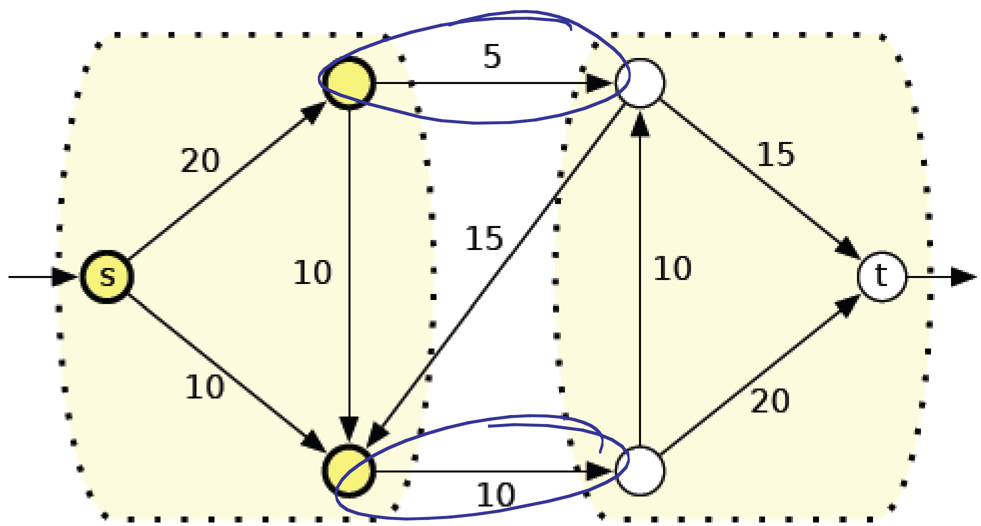
A flow in this graph:



An (s, t)-flow with value 10. Each edge is labeled with its flow/capacity.

Dfn: An s-t cut is a partition of V into 2 sets (S, T) with $s \in S$, $t \in T$.

The capacity of a cut $c(S,T) = \sum_{e \text{ out of } S} c_e$

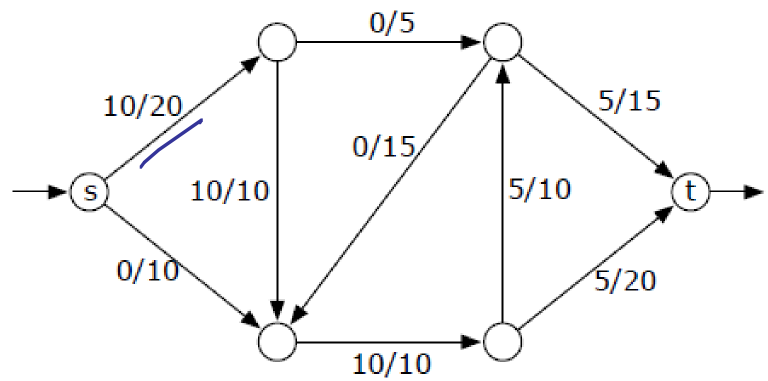


An (s,t)-cut with capacity 15. Each edge is labeled with its capacity.

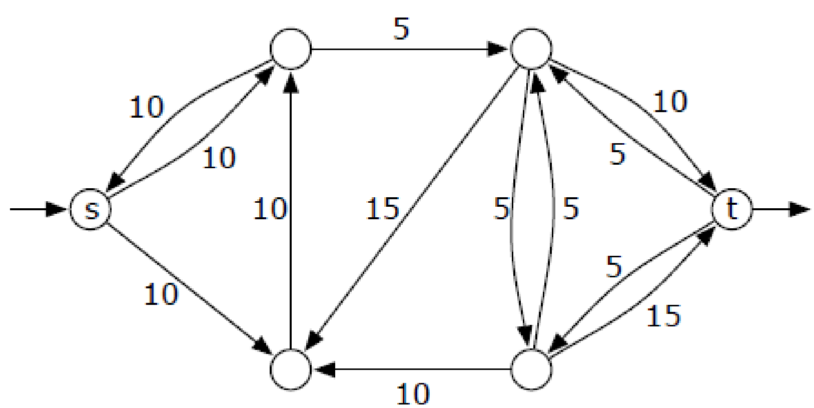
15

Our algorithm (in pictures)

Consider some flow:

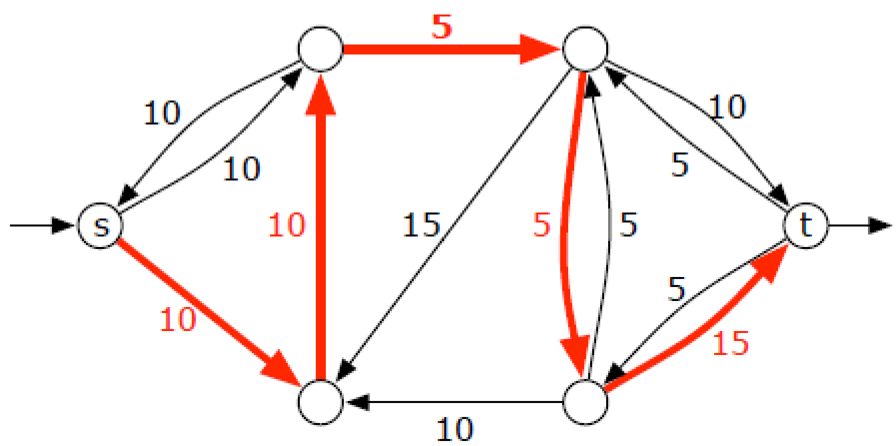


Form the residual graph G_f (of G with respect to flow f):

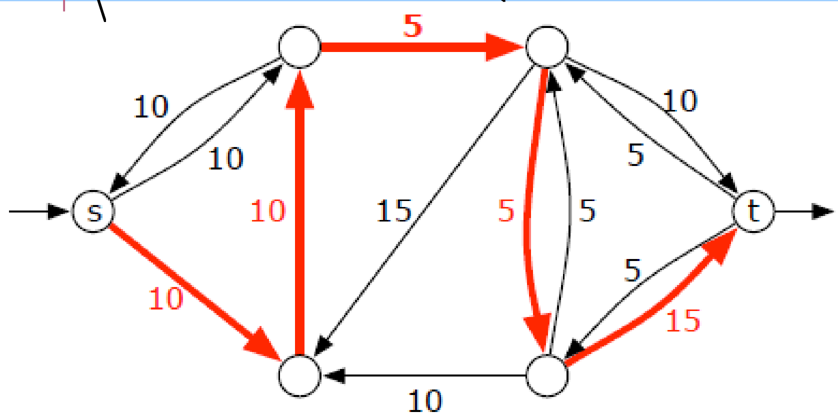


Find a path in this residual graph, & let F = bottleneck edge.

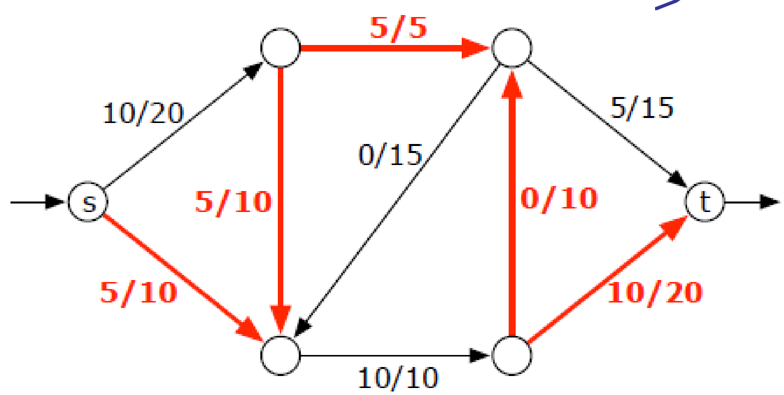
Consider the edges in this path, & update the original flow:



path in G_f:



new flow: (in G)



$$f'(u\to v) = \begin{cases} f(u\to v) + F & \text{if } u\to v \text{ is in the augmenting path} \\ f(u\to v) - F & \text{if } v\to u \text{ is in the augmenting path} \\ f(u\to v) & \text{otherwise} \end{cases}$$

Pseudocode

Maxflow(G):

$f(e) \leftarrow 0 \quad \forall e \in E$

$G_f \leftarrow G$

while there is an s-t path in G_f

$O(m)$ Let $P \leftarrow$ s-t path in G_f

$O(n)$ $f' =$ Augment(f, P) ← update flow using path in G_f

$f \leftarrow f'$

$O(m)$ Update G_f

return f

☆ Need to show that this algorithm gives maximum flow

Strategy: 2 things

① Thm: Let f be any s-t flow, and (S,T) any s-t cut.

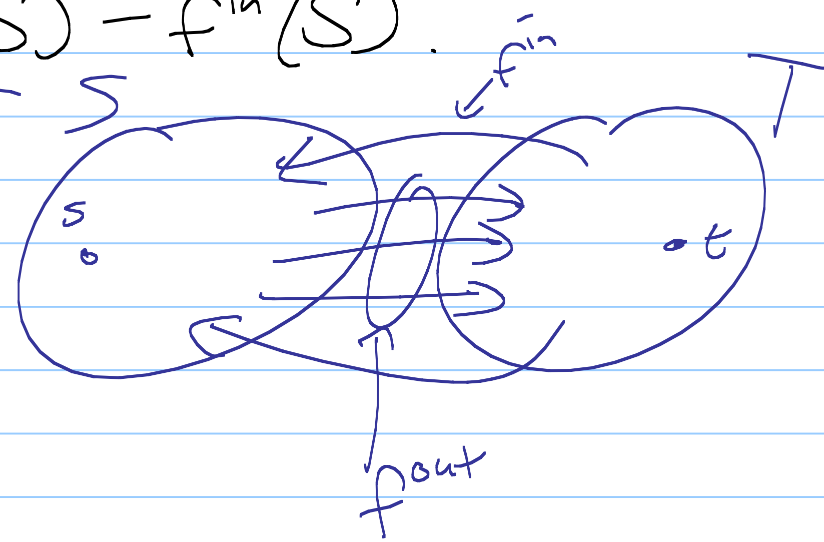
Then $v(f) \le c(S,T)$

② Given a flow f where there is no s-to-t path in G_f, we can find a cut (S^*,T^*) with:

$$\underbrace{v(f)}_{\text{max flow}} = \underbrace{c(S^*,T^*)}_{\text{min cut}}.$$

First, a lemma:

<u>Lemma:</u> Let f be any s-t flow, and (S, T) any S-T cut. Then $v(f) = f^{out}(S) - f^{in}(S)$.



<u>pf:</u> By definition:

$v(f) = f^{out}(s)$

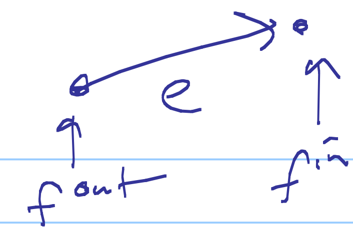
also $f^{in}(s) = 0$

So: $v(f) = f^{out}(s) - f^{in}(s)$

For any other $v \in S$

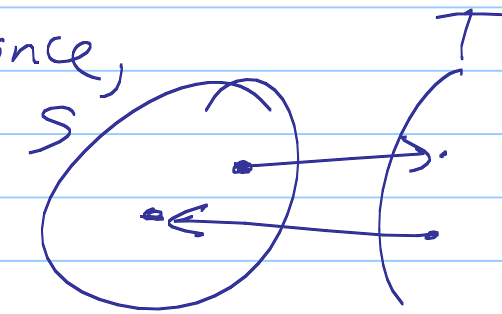
$f^{out}(v) = f^{in}(v)$

so $v(f) = \sum_{v \in S} \left(f^{out}(v) - f^{in}(v) \right)$



$$v(f) = \sum_{v \in S} \left(f^{out}(v) - f^{in}(v) \right)$$

Think about edges in S.
Any edge with 2 endpoints in $\underline{S}$ appears twice in sum.
Any edge out of S appears once, in f^{out} of a vertex.
Any edge into S appears once, in some f^{in} term.



So: $$\sum_{v \in S} \left(f^{out}(v) - f^{in}(v) \right) = \sum_{e \text{ out of } S} f(e) - \sum_{e \text{ into } S} f(e)$$

$$= f^{out}(S) - f^{in}(S) \qquad \blacksquare$$

Thm: Let f be any s-t flow, and (S,T) any s-t cut.

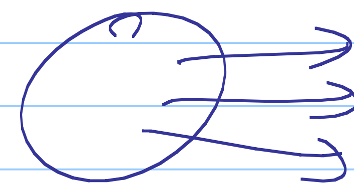
Then $v(f) \le c(S,T)$ ←

pf: $v(f) = f^{out}(S) - f^{in}(S)$ (by lemma)

$\le f^{out}(S)$

$\le \sum_{e \text{ out of } S} c(e)$

$= c(S,T)$



∎

Thm: Given a flow f where there is no s-to-t path in G_f, we can find a cut (S^*, T^*) with:

$$v(f) = c(S^*, T^*).$$

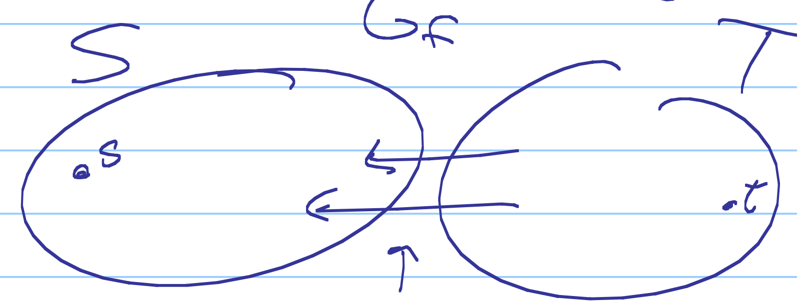
pf: Consider G_f.

No s-to-t path, so let

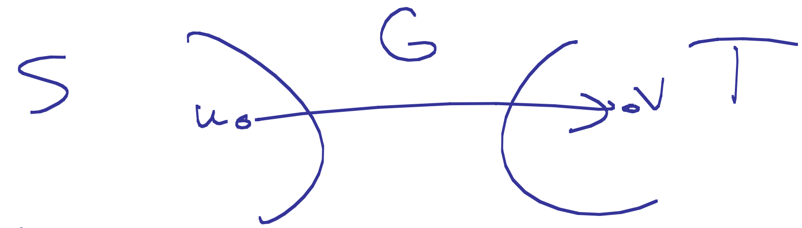
$$S = \{v \in V \mid G_f \text{ has an } s \text{ to } v \text{ path}\}$$

(Note: $t \notin S$)

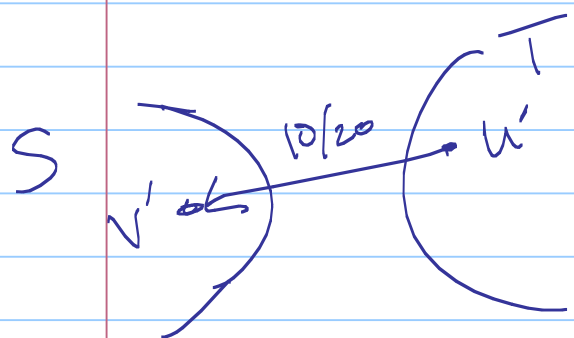
Let $T = V - S$



Consider
e in G.



pf (cont) Consider $e \in G$ going from S to T, $e = (u, v)$.
$v \in T$, so $f(e) = c_e$ (or else v would be reachable in G_f).



Consider $e' \in G$ from T to S, $e' = (u', v')$
Know $\underbrace{u' \notin S}$, so reversed edge is not in G_f
$\Rightarrow f(e') = 0$.

$$\Rightarrow v(f) = f^{out}(S) - f^{in}(S)$$

$$= \underbrace{\sum_{e \text{ out of } S}} f(e) - \sum_{e \text{ into } S} f(e)$$

$$= \sum_{e \text{ out of } S} c_e - \sum_{e \text{ onto } S} 0 = c(S, T)$$

Runtime: (A first try)

- In each loop, flow increases by at least 1.

- Each time in loop takes $O(m+n)$

$\Rightarrow O(m|f|)$

value of max flow

Ideas for improving:

- Choose path with largest bottleneck edge
- Choose path with min. # of edges

both lead to "good" poly. time algs