CSCI 3100

Flows in graphs

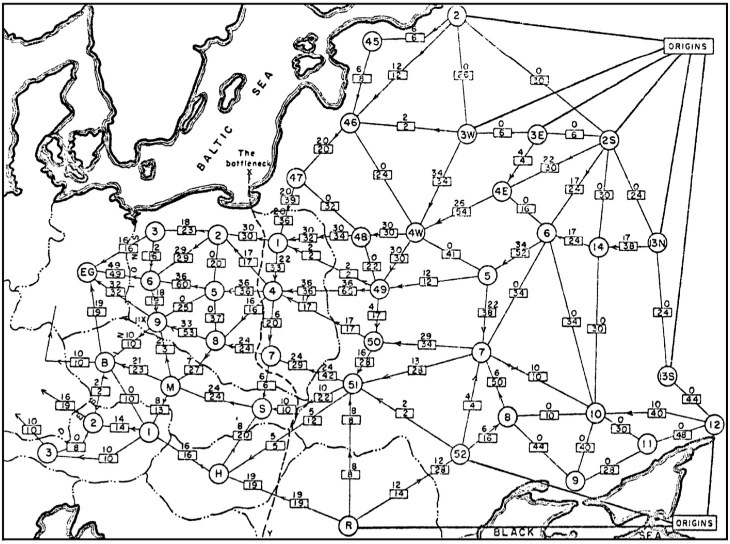
Today

- Grab a practice exam
  (two cheat sheets this time)
- Let me know if any
  issues w/ schedule
  for Thursday/Friday

→ Also fair game:
Simple examples:
ie Compute DFS/BFS,
SPSS, MST, etc.

Maximum flows + minimum cuts

Classified report from 1950's:



Harris and Ross's map of the Warsaw Pact rail network

Model of a Soviet railway.

- each edge given a capacity (how much it could ship)

Goal: Find how much could be shipped

(and cheapest way to disrupt this shipping)!

More formally:

Given a directed graph with two designated vertices, s and t.

Each edge is given a capacity $c(e)$.
↳ maximum amount that can be sent along it.

Assume:

- No edges enter s.
- No edges leave t.
- Every $c(e) \in \mathbb{Z}$.
  (in integers)

Goal:

Max flow: find the most we can ship from s to t without exceeding any capacity

Min cut: find smallest set of edges to delete in order to disconnect s + t

Flows:

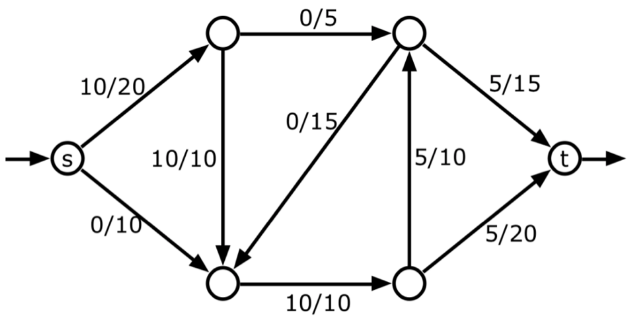
A **flow** is a function $f: E \to \mathbb{R}^+$, where $f(e)$ is the amount of flow going over edge e.

Must satisfy:

capacity: $\forall e \in E$, $0 \le f(e) \le c(e)$

conservation: $\forall v \in V$ (besides s + t),

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$



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

$$\text{Value}(f) = \sum_{e \text{ out of } s} f(e)$$

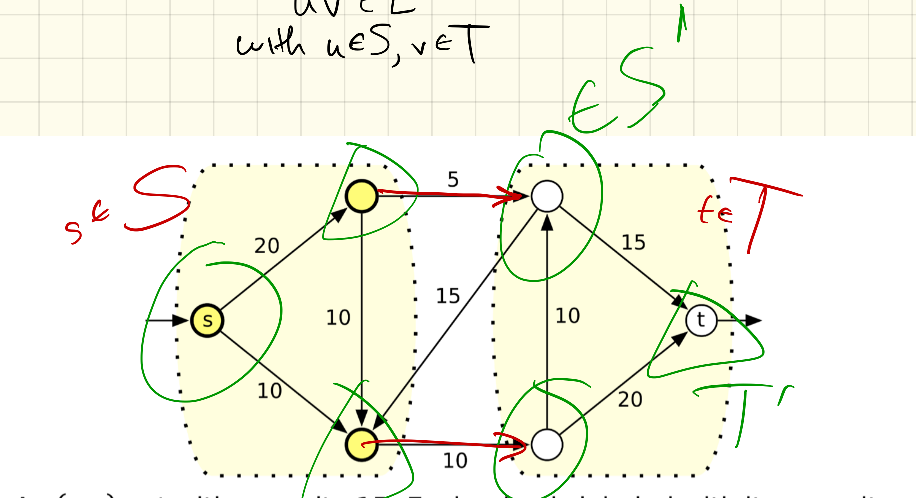
$$= \sum_{e \text{ into } t} f(e)$$

Cuts:

An <u>s-t cut</u> is a partition of the vertices into 2 sets, S and T, so that:

- $s \in S$
- $t \in T$
- $S \cap T = \emptyset$, $S \cup T = V$

The <u>capacity</u> of a cut is $\sum_{\substack{\vec{uv} \in E \\ \text{with } u \in S, v \in T}} c(\vec{uv})$

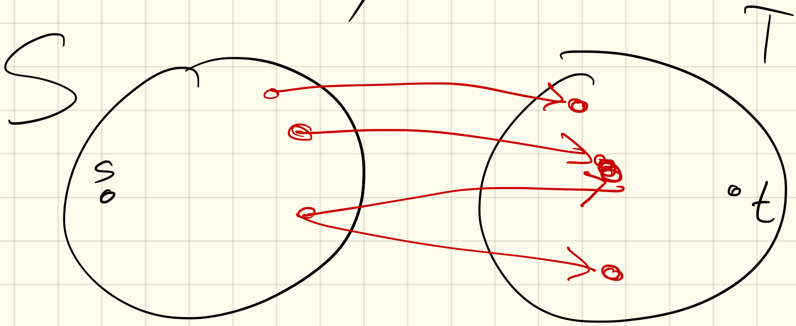


An (s,t)-cut with capacity 15. Each edge is labeled with its capacity.

min cut is the one of smallest capacity

Intuitively, these are connected:

Consider <u>any</u> cut:

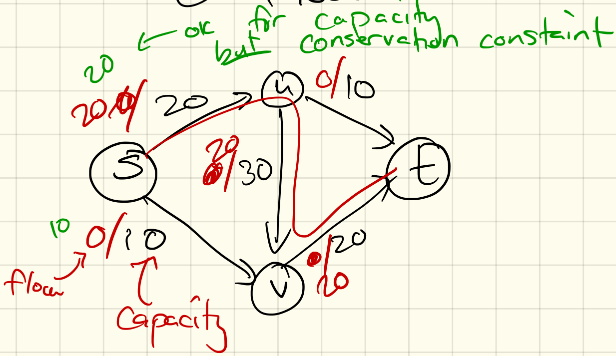


any s-t flow $\leq$ any s-t cut

<u>Why?</u> must leave S some time, + use one of the cut's edges to do so

An algorithm:

Suppose we start with a 0-flow:



How could we increase it?

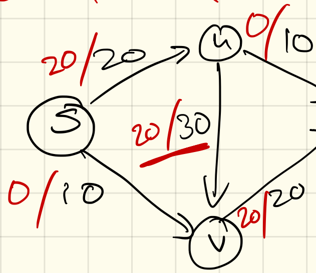
Augment:
Find an s-t path. P.
Find max c(e) for e∈P.
Send that amount

↪ Push flow along some path.

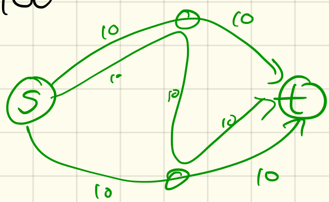
New flow won't violate conservation constraint:
send $\max_{e \in P} c(e)$ in + out of each vertex on P.

After pushing on a path:

New flow f':



Could be "stuck": need to unpush to get more flow!

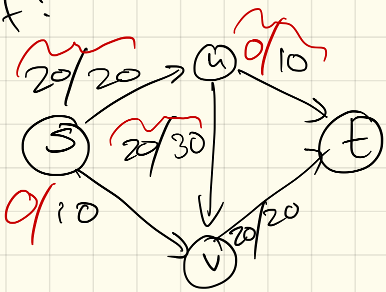


The residual network:

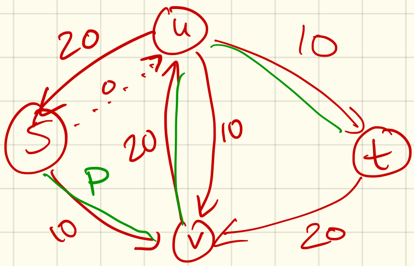
Given a flow network G + flow f, construct G_f:

- Keep edge, but w/ capacity $C(e) - f(e)$ (amount you could still send on e)
- add reverse edge w/ capacity $f(e)$ → amount you could "un-push"

G + f:



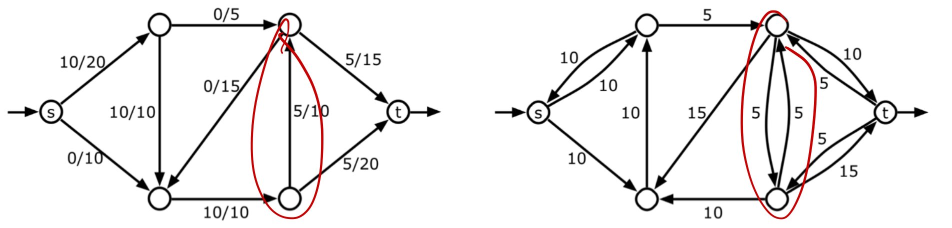
↳ G_f:



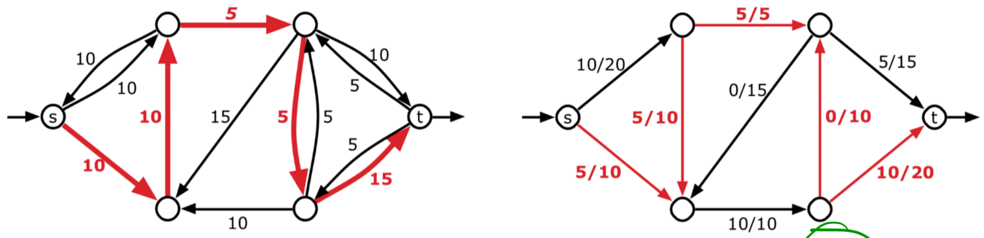
augment!

Augmenting paths

Now, given G_f, any path from s-t gives some more possible flow to send:



A flow f in a weighted graph G and the corresponding residual graph G_f.



An augmenting path in G_f with value $F = 5$ and the augmented flow f'.

Let f' be the new flow:

Prop: If f is a flow, then f' is still a feasible flow.

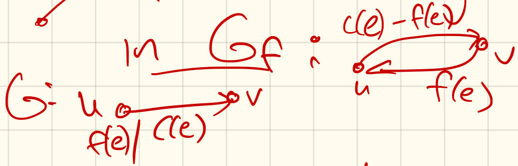
pf:

① capacity constraints still hold:

$\forall e,\ 0 \le f'(e) \le c(e)$

look at edges on P

in G_f:



if on P, either increased or decreased flow.

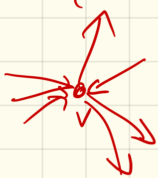
Amount up/down was $\le$ max edge in G_f on P.

adding $\le c(e) - f(e)$

or subtracting $\le f(e)$ ✓

② convervation still holds:

$\forall v \in V$ (not s or t) $\quad \sum_{e \text{ into } v} f'(e) = \sum_{e \text{ out of } v} f'(e)$



any alteration happens for same value in + out of $v \in P$. ✓

So an algorithm: Ford-Fulkerson (1956)

```
MAXFLOW(G):

    Let f(e)=0 initially ∀e
    Construct G_f

    While there is s-t path in G_f:
        Let P be a simple s-t path
        f' ← augment(f, P)
        f ← f'
        update G_f

    return f
```

Need to show:

- terminates
- returns max flow

Lemma: At each stage, flow & residual values are integers.

pf:

all capacities are integers

Find a path
↪ bottle neck edge is also integer value

In G_f, only integer edges

(repeat)

Lemma: In each iteration,

$$\text{value}(f') > \text{value}(f).$$

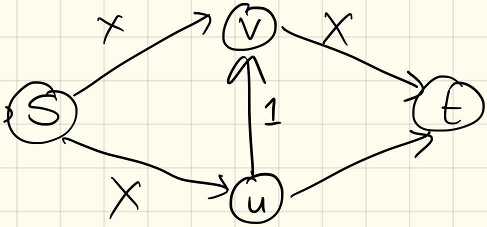
In each iteration, value improves

pf:

Lemma: The while loop halts after $O(\text{value}(f^*))$ iterations, where f^* is a maximum flow.

So: Running time is:

Note: This is the best we can do!



Worst path:

To do better, need to consider how to choose a "good" augmenting path.

Thm: The F-F algorithm terminates with a maximum flow.

To prove this, we'll use cuts!

Fact: For any S-T cut,

$$\text{value}(f) = f^{out}(S) - f^{in}(S)$$

More next time...