Algorithms - 2020

Last of MSSP

Flows

Recap

HW1 is graded

HW3 - half done

Midterm grades due tomorrow

HW5 - due today

over WFS

- HW due Monday: a note

Many graph problems fall into 2 types:

1 - Given random odd problem, convert to G ← core
& call an alg. ← find alg.

Caution: input is not a graph! ← put V & E explicitly in terms of input

2 - Modify an existing alg (carefully). So - choose pseudocode, then change the data/loop/whatever.

Get in touch if you are struggling!

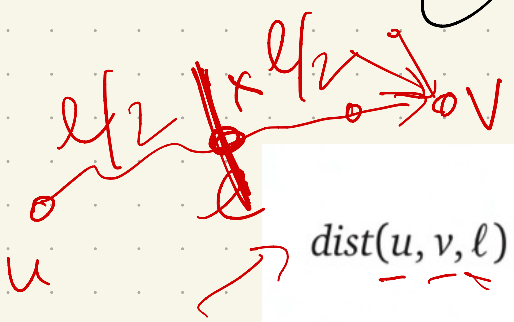
Last time: MSSP

- Johnson's alg: $O(VE \log V)$

weighting $\mid$ one SSSP (B-F) $O(VE)$

gets positive graph

- Dynamic programming $O(V^3 \log V)$



$$dist(u,v,\ell) = \begin{cases} w(u\to v) & \text{if } i = 1 \\ \min_x \left(dist(u,x,\ell/2) + dist(x,v,\ell/2)\right) & \text{otherwise} \end{cases}$$

$\Downarrow$

due from B-F

```
FischerMeyerAPSP(V, E, w):
    for all vertices u
        for all vertices v
            dist[u, v, 0] ← w(u→v)
    for i ← 1 to ⌈lg V⌉        ⟨⟨ℓ = 2^i⟩⟩
        for all vertices u
            for all vertices v
                dist[u, v, i] ← ∞
                for all vertices x
                    if dist[u, v, i] > dist[u, x, i−1] + dist[x, v, i−1]
                        dist[u, v, i] ← dist[u, x, i−1] + dist[x, v, i−1]
```

relax!

Can we do better?

Floyd-Warshall:

Use a different 3rd parameter:

→ Index the vertices $1..V$.

(Essentially "random")

Let $path(u, v, r)$ =

shortest $u \rightsquigarrow v$ path

using only vertices $1..r$

(if there is a path)

...

note: $dist(u, v) =$

$dist(u, v, V)$

all vertices

How does this help??

Make a recursion:

path ~~X~~(u, v, r) is the shortest path (if any) from u to v that passes through only vertices numbered at most r.

Consider path$(u, v, \overset{m}{r})$:

- Use vertex r:

  find best $r-1$ paths that now use vertex r

- or don't: (use $r-1$)

  $= \text{path}(u, v, r-1)$

- or base case: $r=0$

  can't use any other vertices

  $= w(u \to v)$ if edge is present

  or 0 if no edge $u \to v$

In other words:

no other vertices

not use r

$$dist(u,v,r) = \begin{cases} w(u \rightarrow v) & \text{if } r = 0 \\ \min \left\{ \begin{array}{c} \underline{dist(u,v,r-1)} \\ dist(u,r,r-1) + dist(r,v,r-1) \end{array} \right\} & \text{otherwise} \end{cases}$$

use r

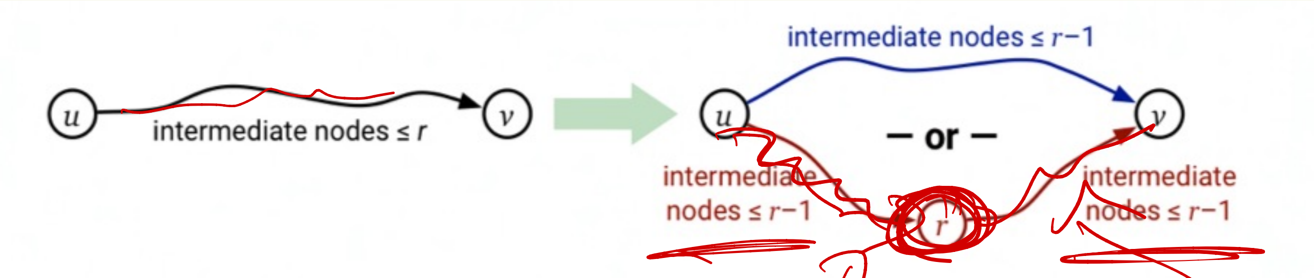


Figure 9.3. Recursive structure of the restricted shortest path $\pi(u, v, r)$.

path(u, r, r−1)

path(r, v, r−1)

Memoize!

Table:

Code:

$O(V^3)$

??

$O(VE)$ of one SSSP

fill in n×n array

```
FloydWarshall(V, E, w):
    for all vertices u
        for all vertices v
            dist[u, v] ← w(u→v)

    for all vertices r
        for all vertices u
            for all vertices v
                if dist[u, v] > dist[u, r] + dist[r, v]
                    dist[u, v] ← dist[u, r] + dist[r, v]
```

Runtime:

3 nested loops, each $O(V)$

$\Rightarrow O(V^3)$

Amazing!
Same as B-F, SSSP.

Ch 10: Flows

Motivation:

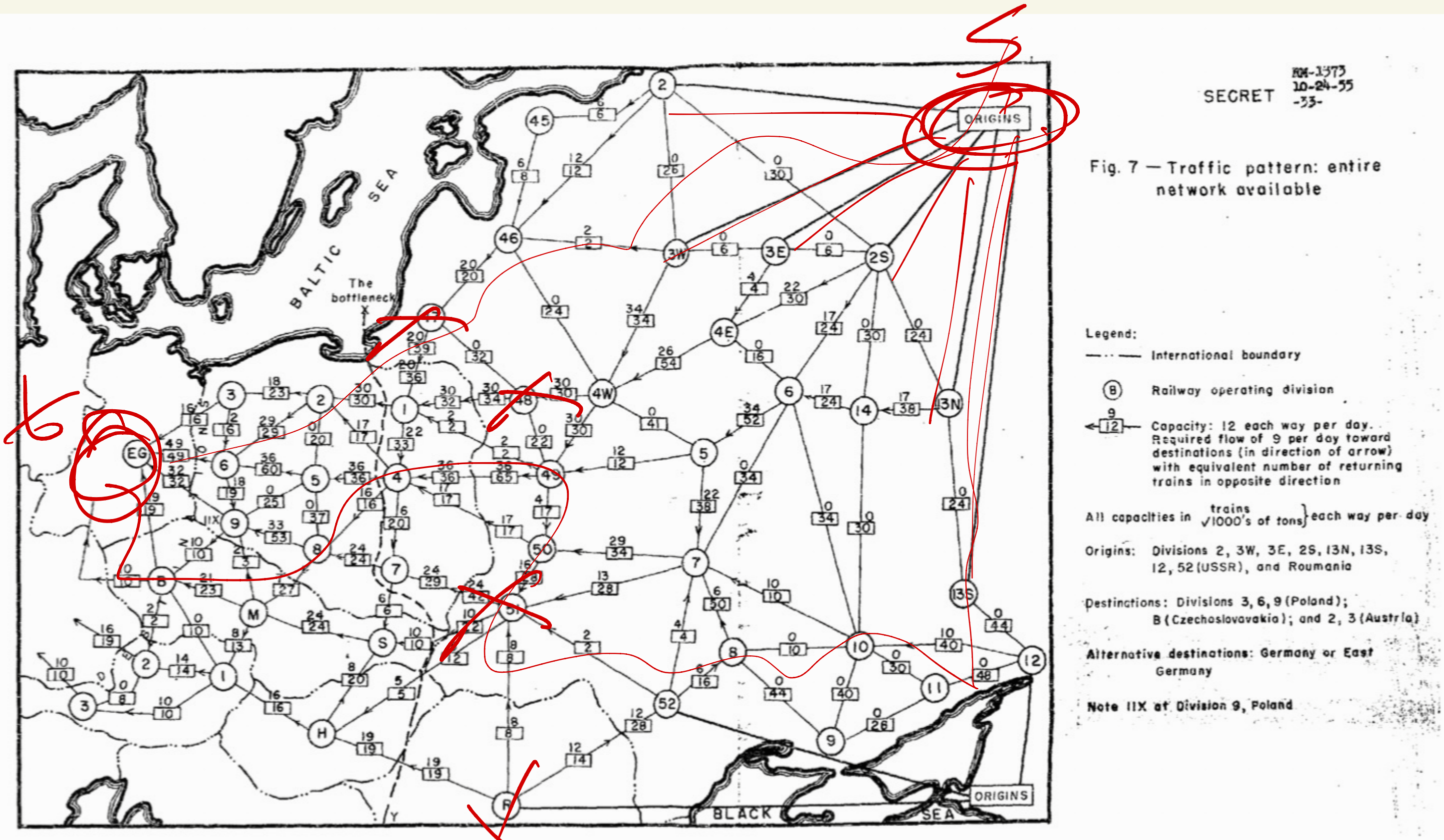


Figure 10.1. Harris and Ross's map of the Warsaw Pact rail network. (See Image Credits at the end of the book.)

How to send from one vertex to another?

How to divide one vertex from another?

More formally:

?

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

Each edge is given a capacity c(e).
→ maximum amount it can carry

Assume:

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

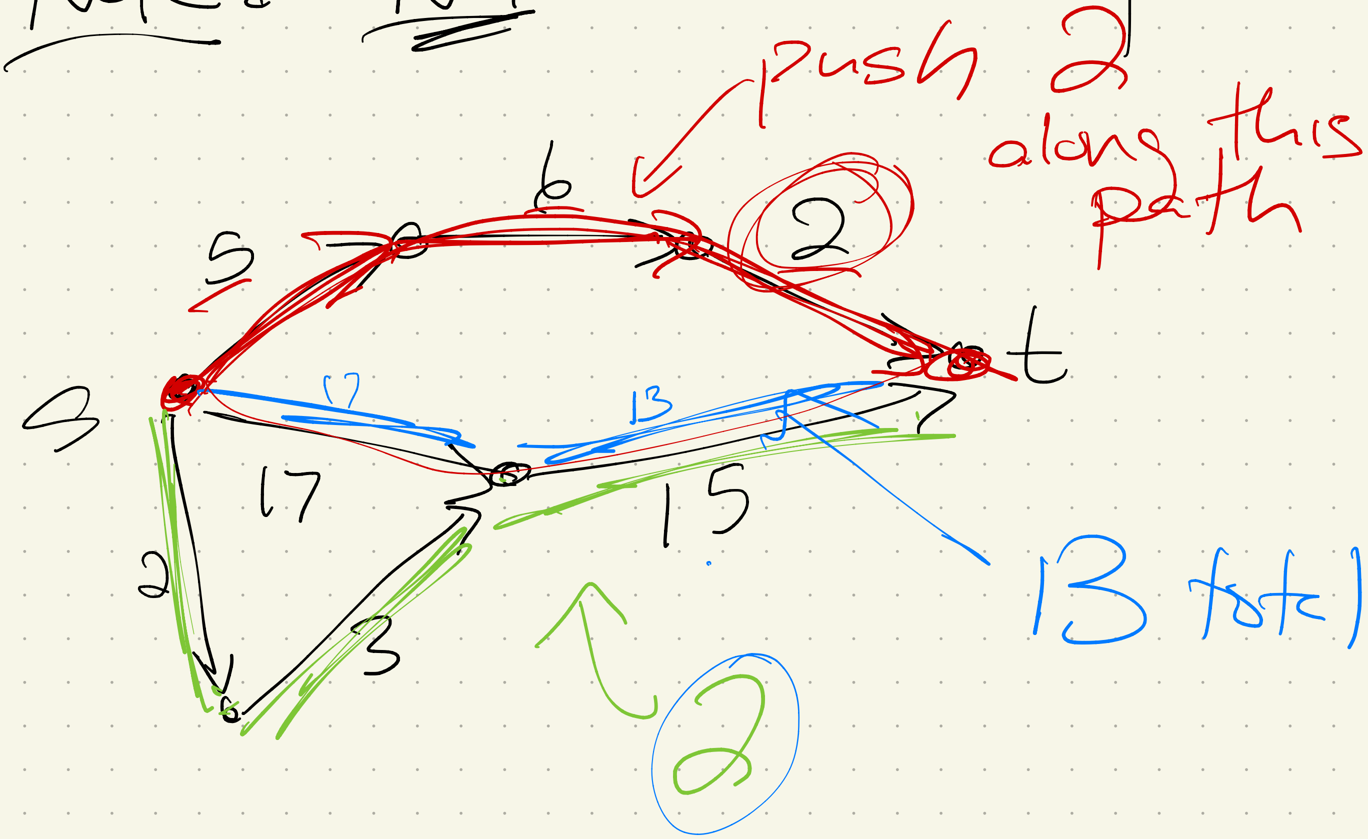
never are irrational → can't store those in computer

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
~~vertices?~~

Note: Not shorstest paths!



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 2 things:

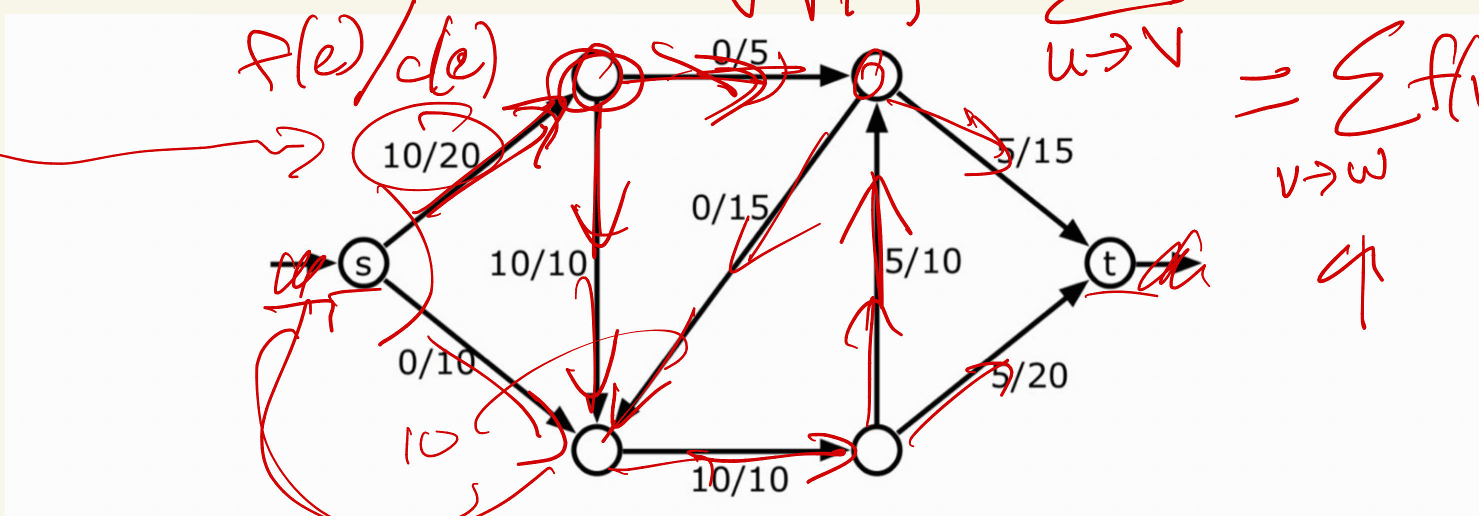
- Edge constraints: don't overload edge

  $0 \le f(e) \le c(e)$

- Vertex constraints: by design

don't want product shipped unless it can get to t

$\forall v \neq s,t: \sum_{u \to v} f(u \to v) = \sum_{v \to w} f(v \to w)$



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

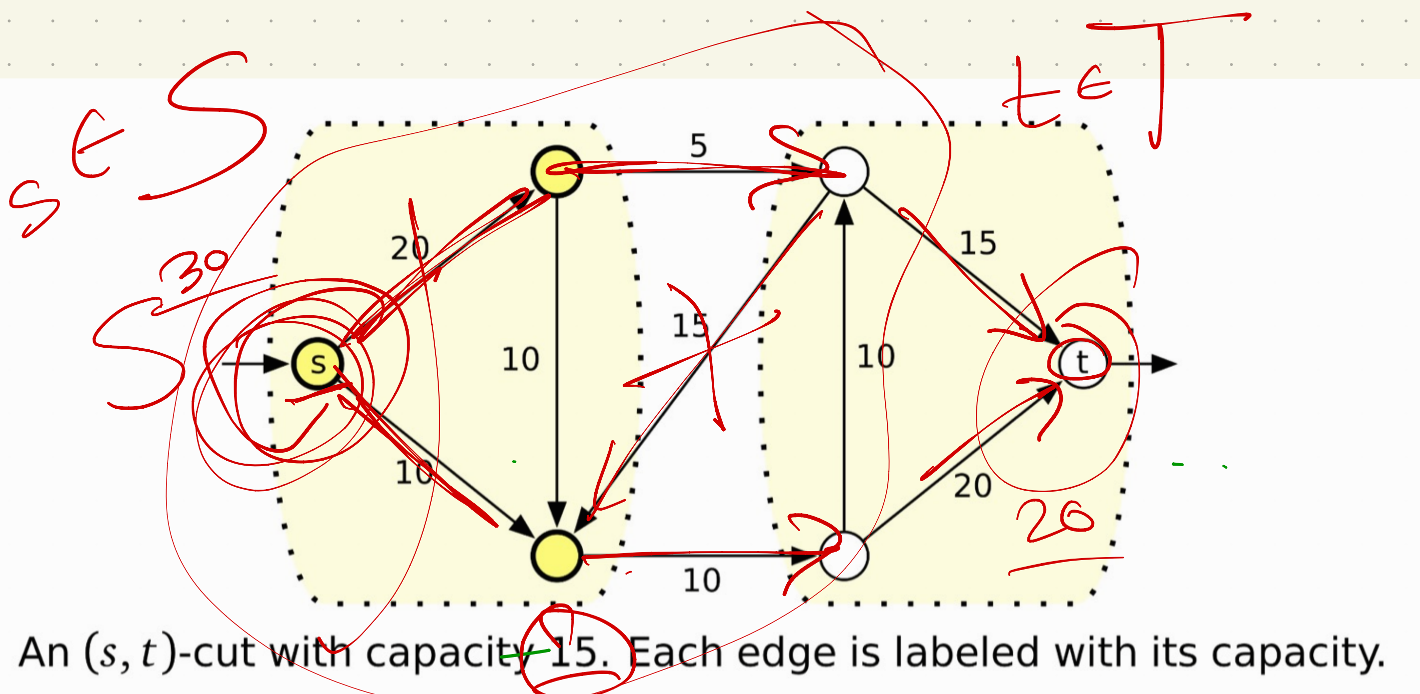
$$\text{Value}(f) = \sum_{e \text{ out of } s} f(e) = \sum_{e \text{ into } t} f(e)$$

Cuts:

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

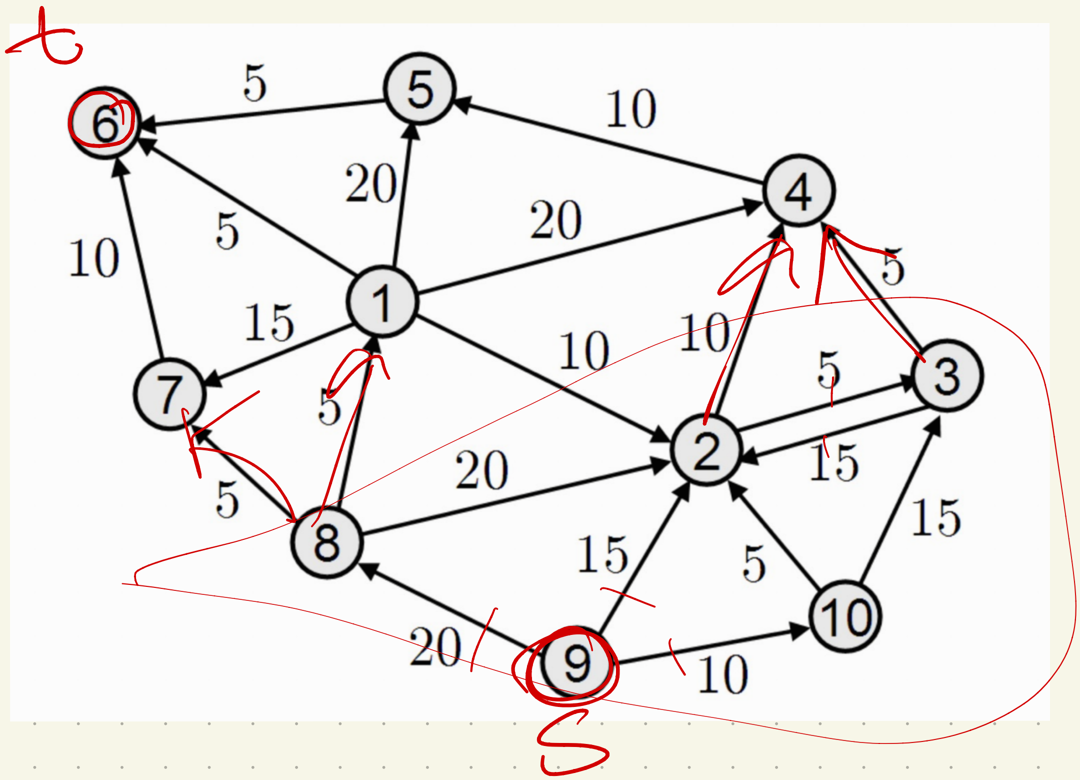
- $s \in S$
- $t \in T$
- $S \cap T = \emptyset$, $S \cup T = V$

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



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

Cuts: not always so obvious!



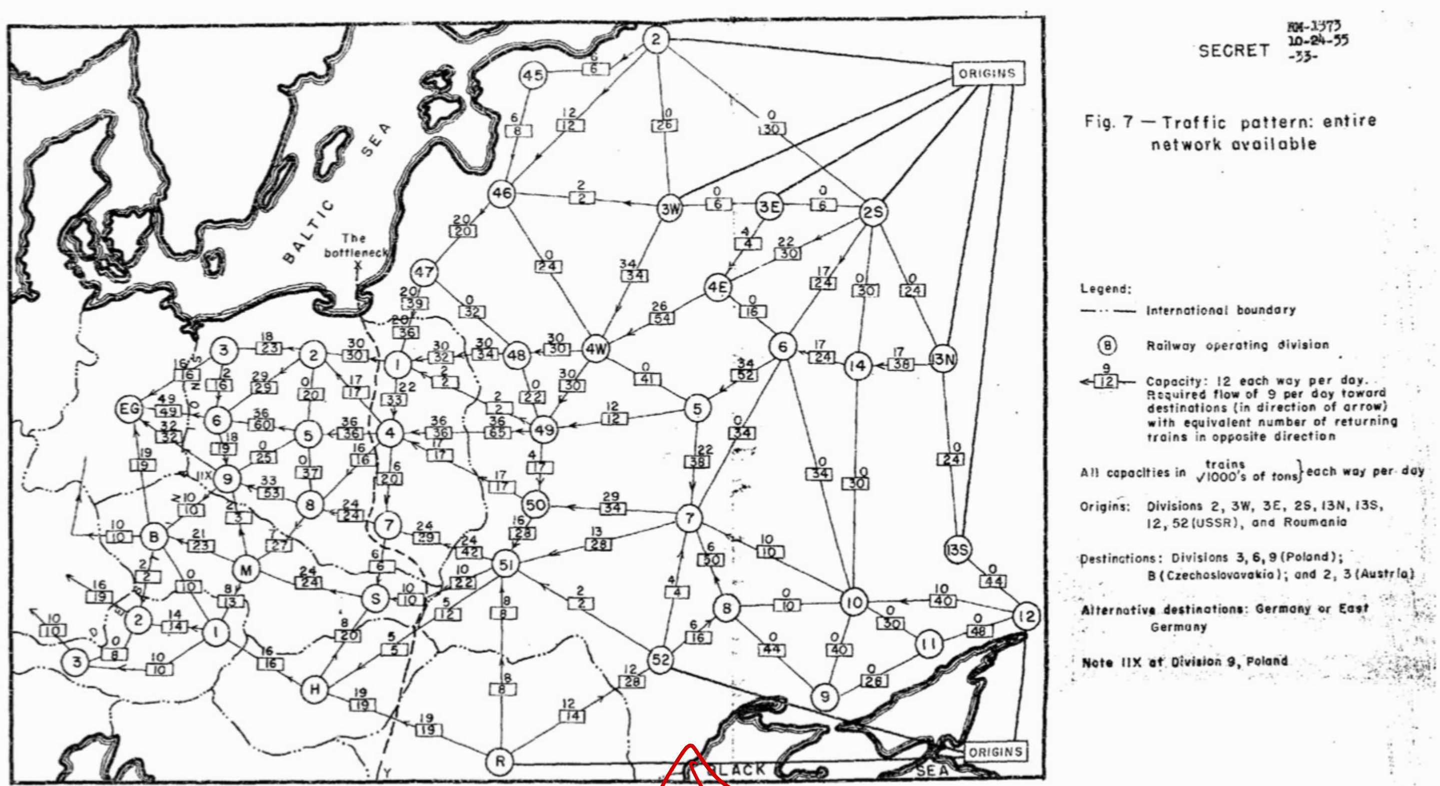
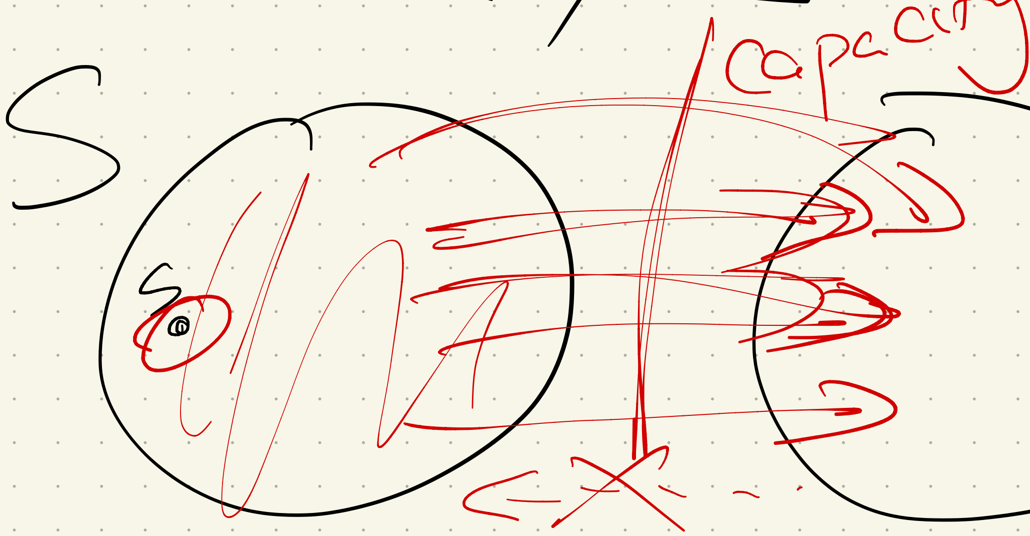


Figure 10.1. Harris and Ross's map of the Warsaw Pact rail network. (See Image Credits at the end of the book.)

Intuitively, these are connected:

Consider <u>any cut</u>:



Any flow from s to t needs to use those edges!

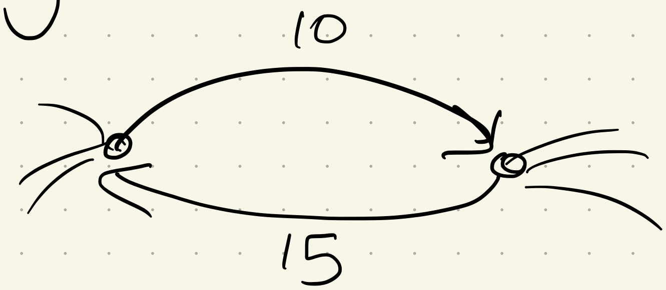
↳ this means ~~best~~ biggest flow ≤ this cut

(smaller is goal)

Note:

We'll assume every pair of vertices has at most one edge.

So no:

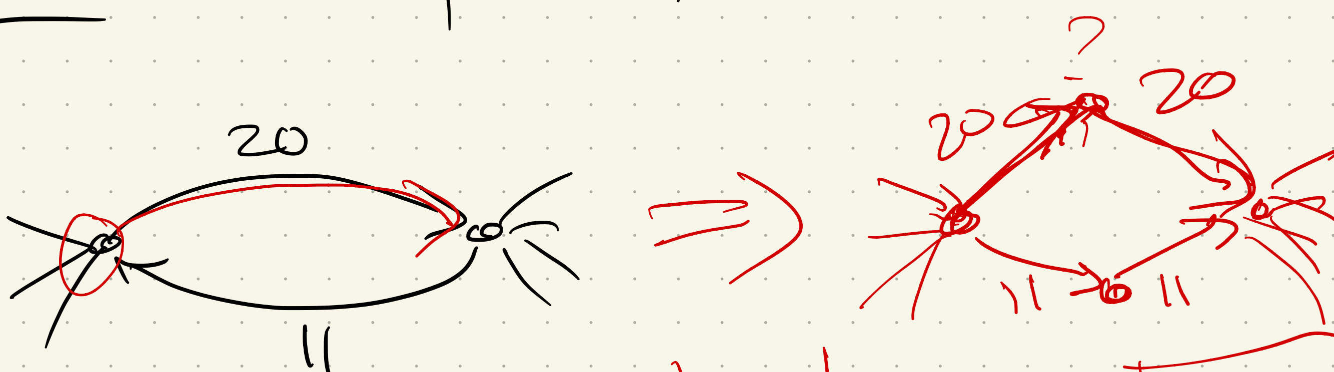


Why?

- Makes calculations easier!

How?

Simple transformation:



makes graph have $\Theta(E)$ more vertices

$$\boxed{\begin{aligned} V &\to V+E \\ E &\to 2E \end{aligned}}$$

Thm: (Ford-Fulkerson '54, Elias-Feinstein-Shannon '56)

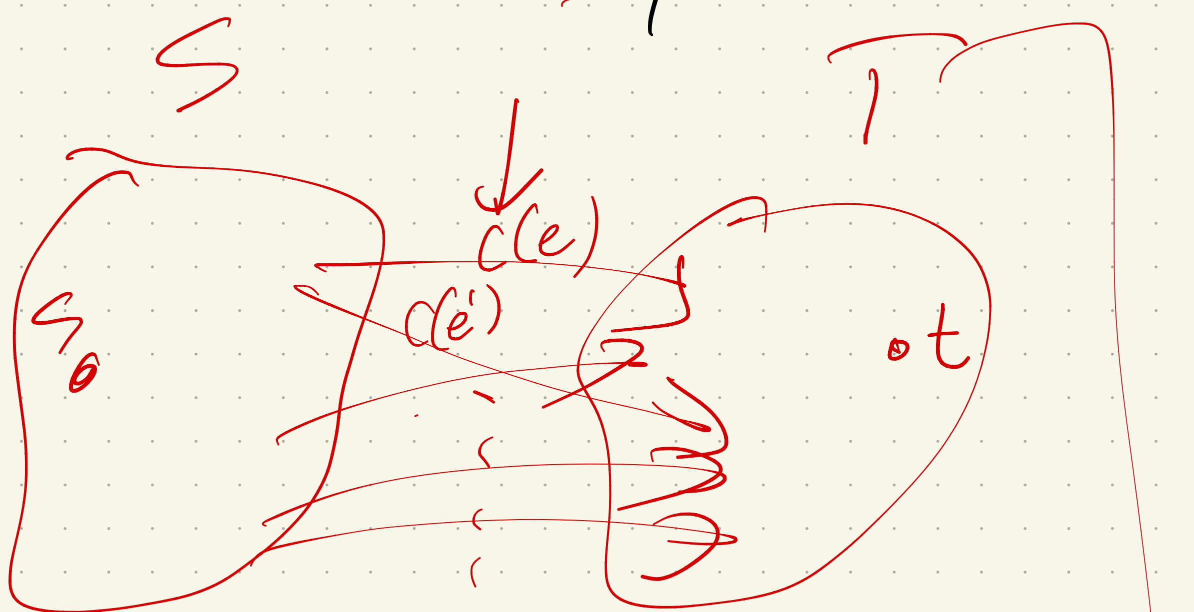
The max flow value = min cut value

Wow! these seem so different...

One way is easy:

Any flow $\leq$ any cut.

Why?



any flow has to get out of S in order to reach $t \in T$

More formally:

$\delta(v) = \sum_{u \to v} f(e) - \sum_{v \to w} f(u \to w)$ should $= 0$ unless s or t

Proof: Choose your favorite flow f and your favorite cut (S, T), and then follow the bouncing inequalities:

flow out of S

$$
\begin{aligned}
|f| &= \partial f(s) && \text{[by definition]}\\
&= \sum_{v \in S} \partial f(v) && \text{[conservation constraint]}\\
&= \sum_{v \in S} \sum_{w} f(v \to w) - \sum_{v \in S} \sum_{u} f(u \to v) && \text{[math, definition of } \partial\text{]}\\
&= \sum_{v \in S} \sum_{w \notin S} f(v \to w) - \sum_{v \in S} \sum_{u \notin S} f(u \to v) && \text{[removing edges from } S \text{ to } S\text{]}\\
&= \sum_{v \in S} \sum_{w \in T} f(v \to w) - \sum_{v \in S} \sum_{u \in T} f(u \to v) && \text{[definition of cut]}\\
&\le \sum_{v \in S} \sum_{w \in T} f(v \to w) && \text{[because } f(u \to v) \ge 0\text{]}\\
&\le \sum_{v \in S} \sum_{w \in T} c(v \to w) && \text{[because } f(v \to w) \le c(v \to w)\text{]}\\
&= \|S, T\| && \text{[by definition]}
\end{aligned}
$$

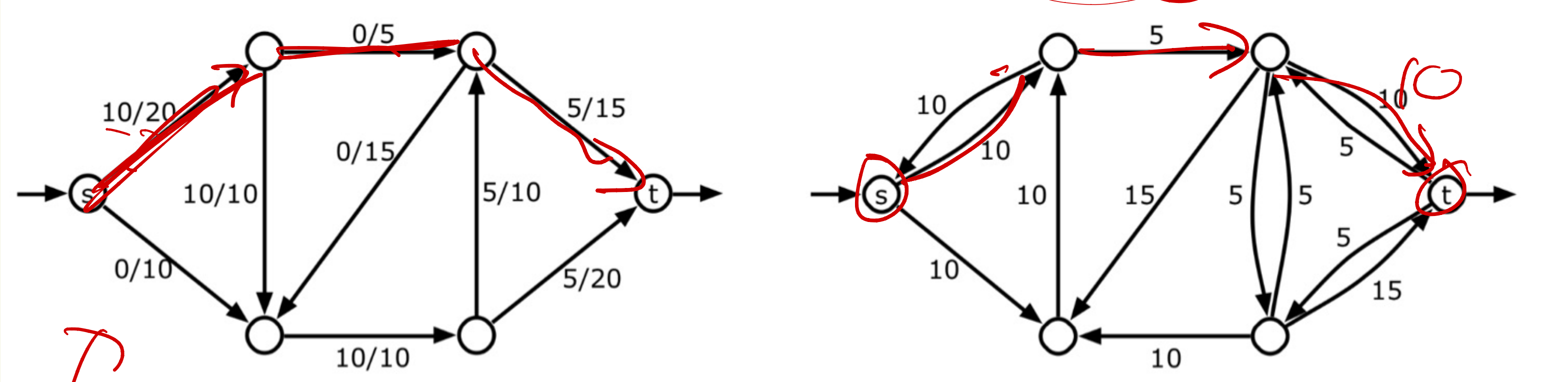
Amazingly - no questions??

Comment & tag me if you have questions

Key tool in proof:

Residual network G_f:

depend on flow f



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

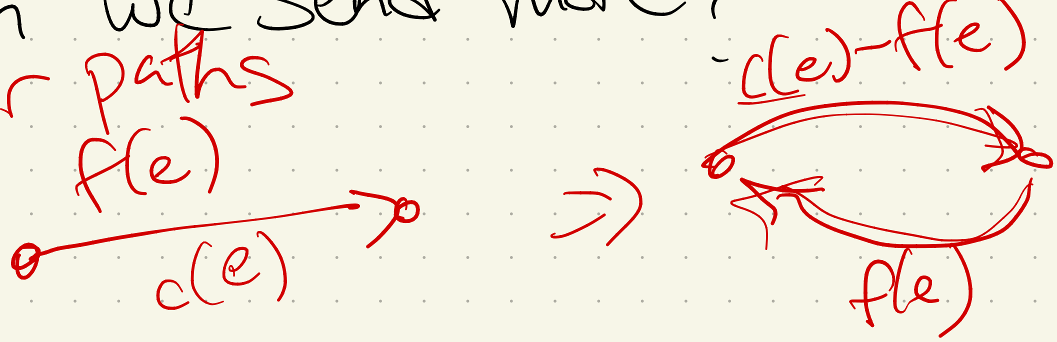
f on G

Intuitively:

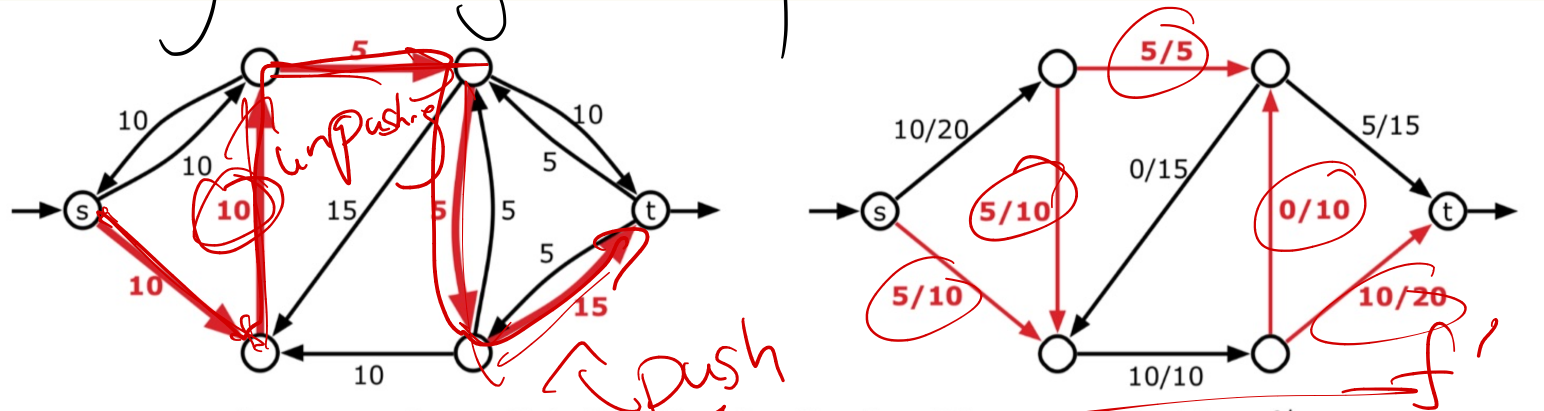
Shows how much more (or less) flow can be pushed through an edge.

How can we send more?

look for paths



Augmenting a path:



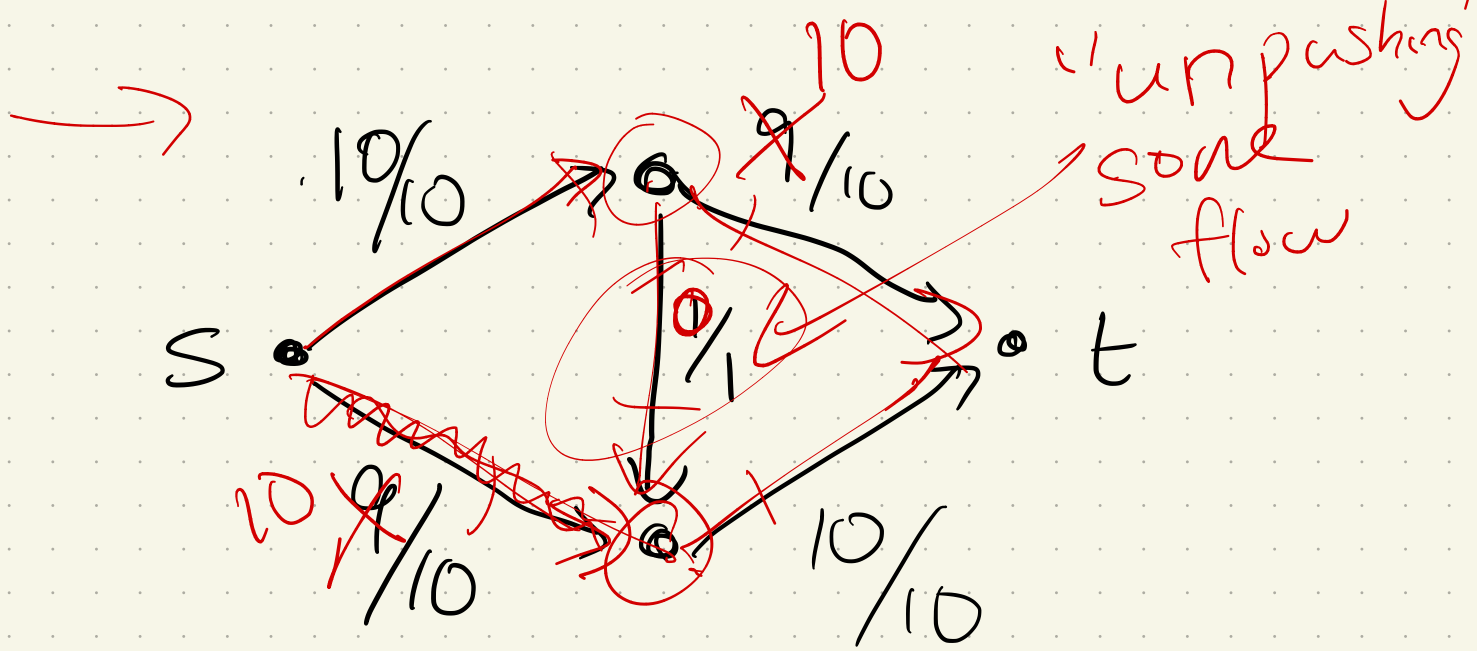
An augmenting path in G_f with value $F = 5$ and the augmented flow f'.

This is just an s-t path in G_f.

Then, find min capacity edge on that path.

<u>Claim</u>: I can build a new flow whose value is bigger than f's!

Why can't we just be greedy?



Are there any more flow paths?

Yes! but need to do is not use an edge