Algorithms - Spring '25

MSSPs
Intro to Flows

Recap

Multiple Source Shortest paths

For all pairs $u, v \in V$, store $dist(u, v)$.

MSSP Algorithms

Approaches:	no negatives	negatives
$\forall v$, compute SSSP(v)	$O(V \cdot E \log V)$	$O(V^2 E)$
Johnson's alg: reweight		$O(V^3 \log V)$
Fisher: Divide + conquer		$O(V^3 \log V)$

Can we get better?

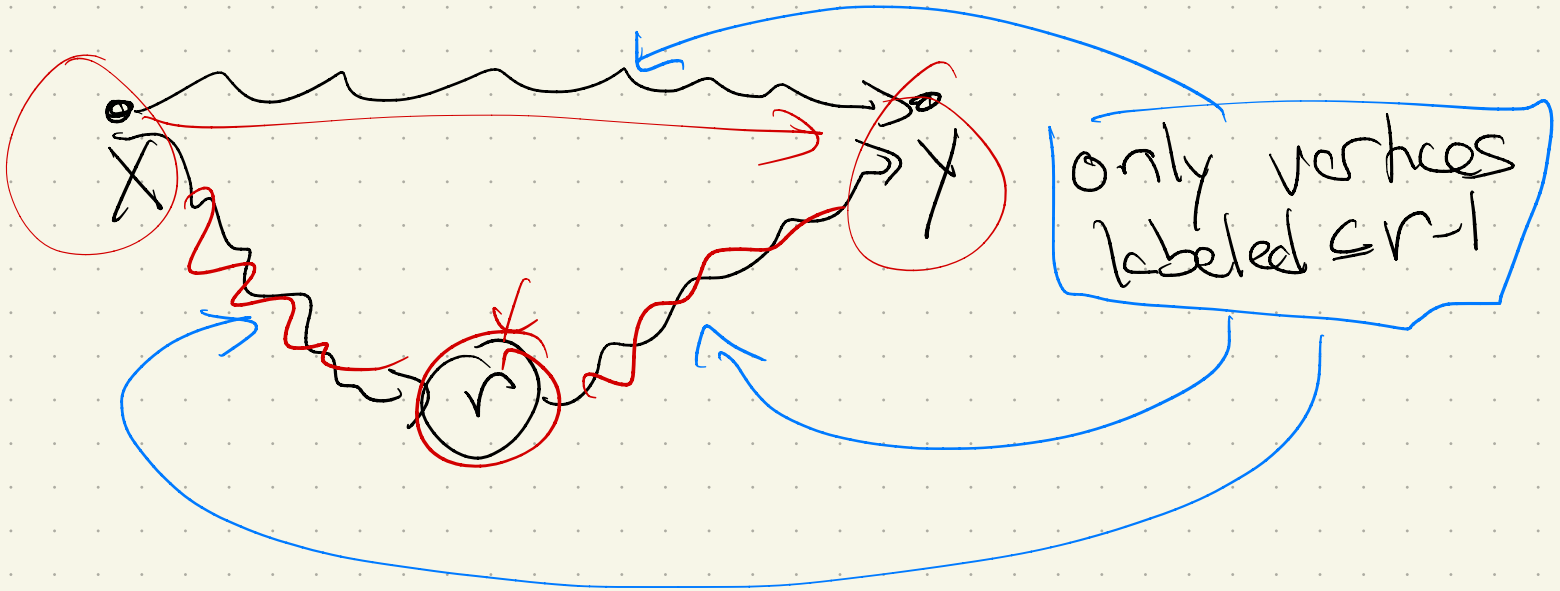
Floyd-Warshall: instead of path length

order vertices $1, \ldots, V$

Let $d(x, y, r) =$

best length path via vertices labeled $1 \ldots r$

Then:



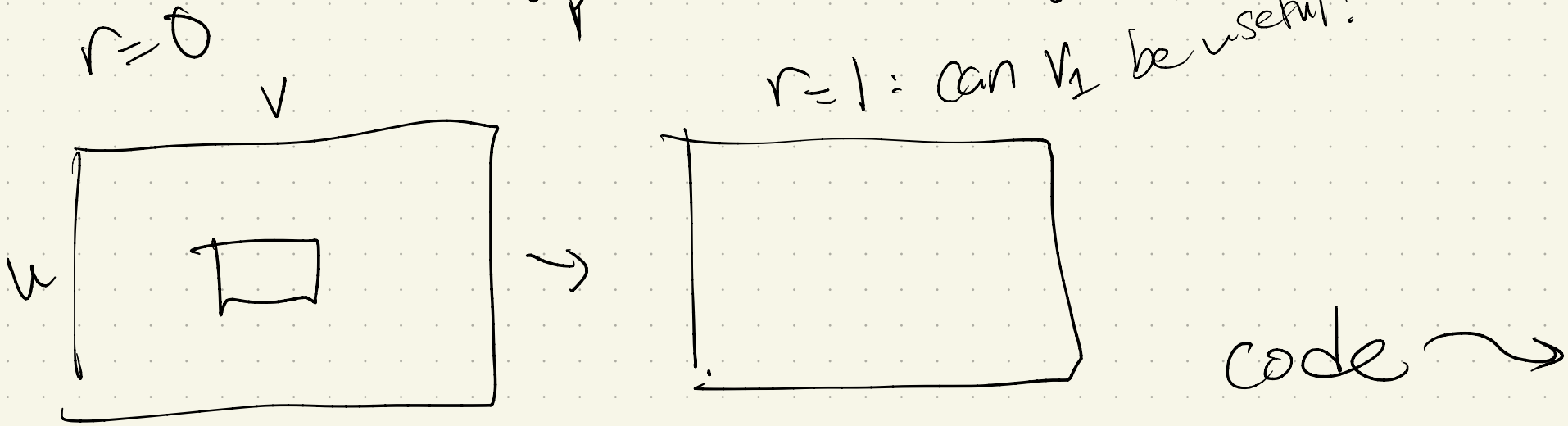
Recursion:

$$dist(u,v,r) = \begin{cases} w(u \to v) & \text{if } r = 0 \\ \min \begin{cases} dist(u,v,r-1) \\ dist(u,r,r-1) + dist(r,v,r-1) \end{cases} & \text{otherwise} \end{cases}$$

don't use vertex r

use vertex r

So: for $r \leftarrow 1$ to $V-1$
for all $u, v \in G$
update $dist(u,v,r)$



code ↝

```
KleeneAPSP(V, E, w):
  for all vertices u
      for all vertices v
          dist[u, v, 0] ← w(u→v)

  for r ← 1 to V
      for all vertices u
          for all vertices v
              if dist[u, v, r − 1] < dist[u, r, r − 1] + dist[r, v, r − 1]
                  dist[u, v, r] ← dist[u, v, r − 1]
              else
                  dist[u, v, r] ← dist[u, r, r − 1] + dist[r, v, r − 1]
```

Runtime:

save space: don't keep older r's, just overwrite

```
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]
```

Special cases

Best known published result for general graphs: $O(V^3)$

Conjecture: no $O(V^{3-\varepsilon})$ algorithm

Planar graphs: $O(V \log V)$

Meshes (graphs embedded in 3d):

$O(g^2 V \log V)$

Many others...

Ch 10: Flows

Motivation:

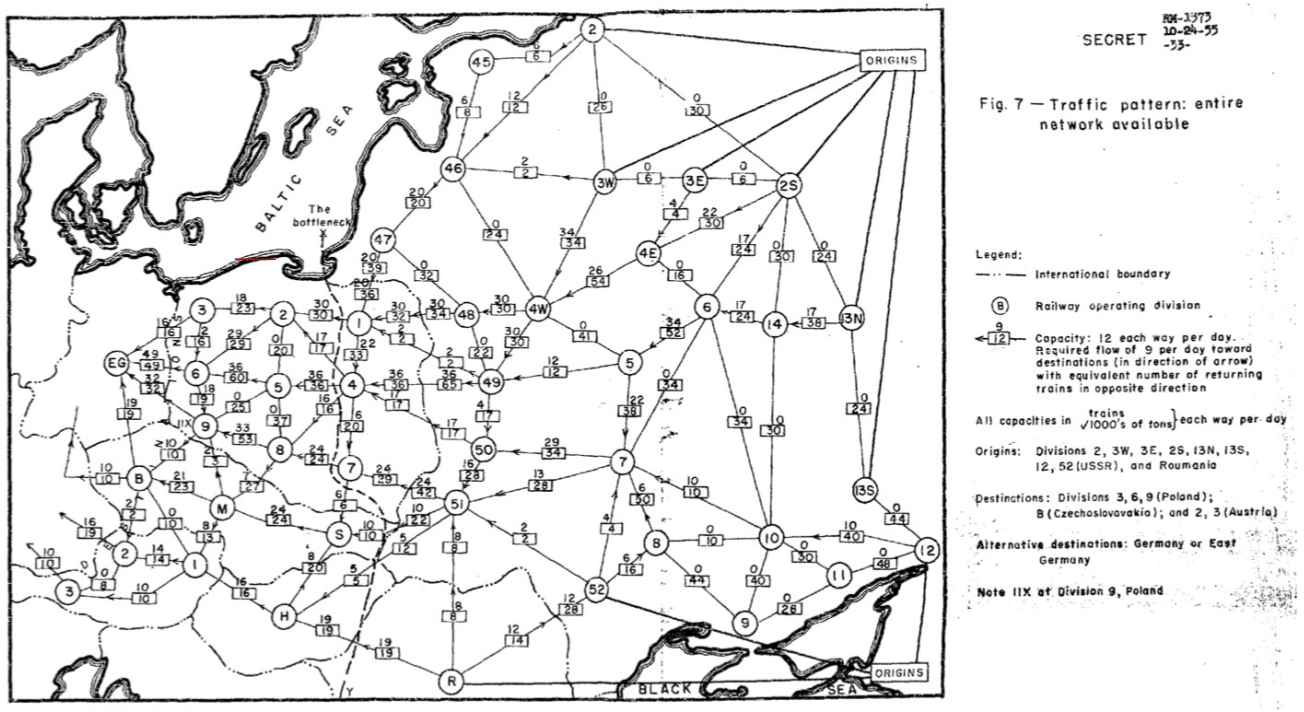


Figure 10.1. Harris and Ross's map of the Warsaw Pact rail network. (See Image Credits at the end of the book.)

Question: How much can I ship from u to v?

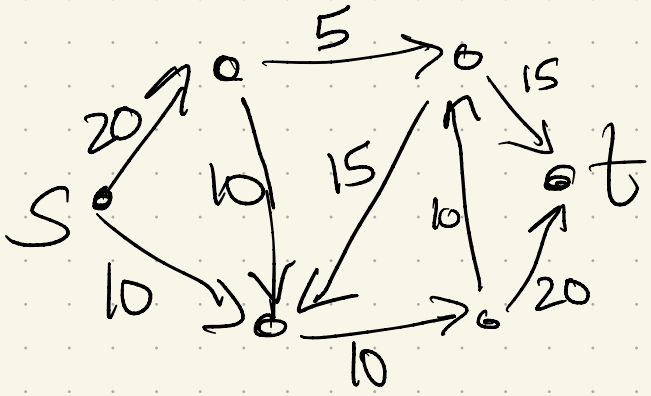
Question: How can I separate u from v with least cost?

More formally:

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

Each edge is given a capacity c(e).

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

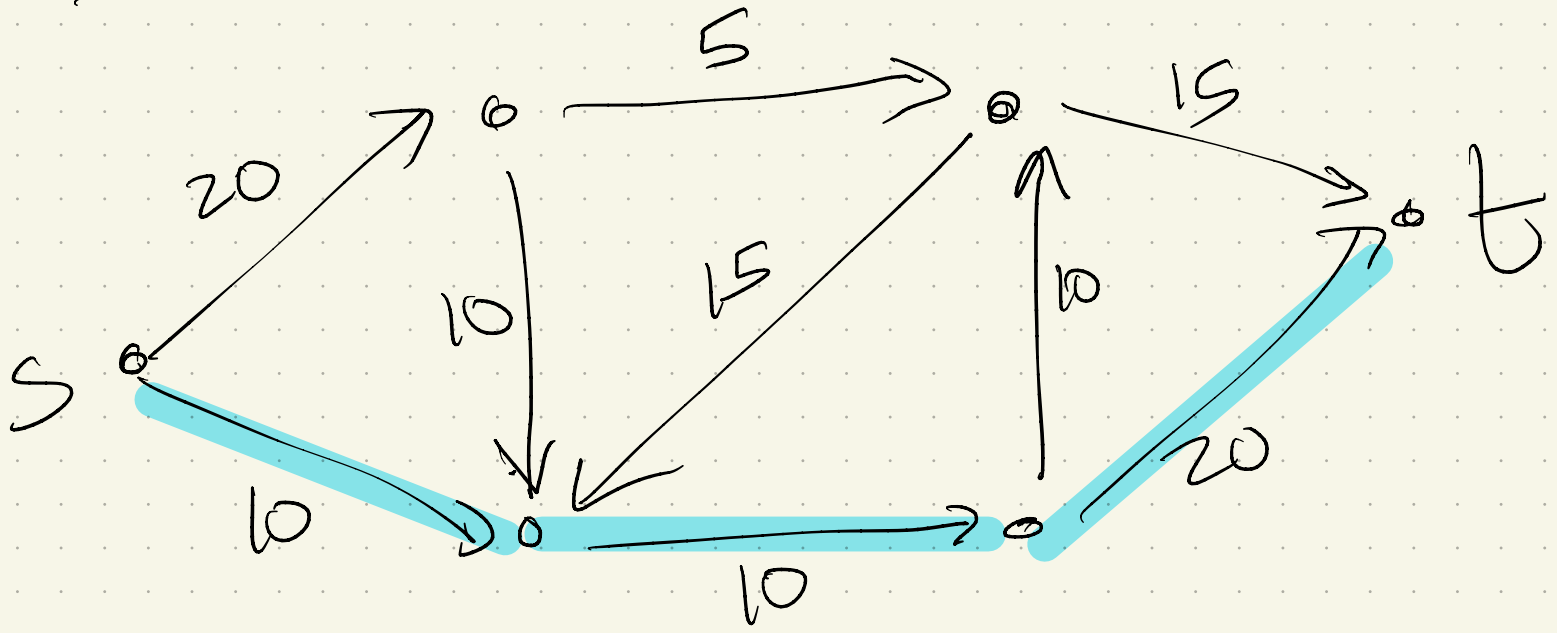


Max flow: Find most I can send from s to t without exceeding edge capacities.

Min cut: find lightest set of edges separating s from t

Aside:

Not path length:



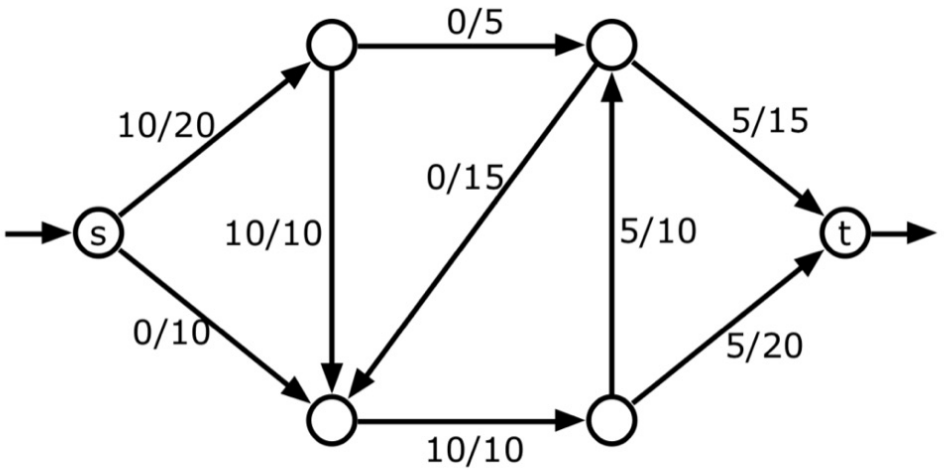
Consider a path s→t:

Formalizing flow:

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:

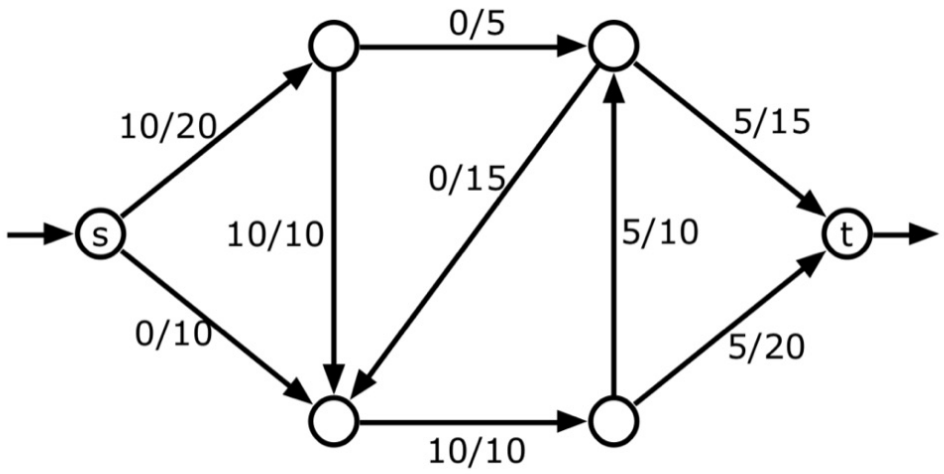
- Edge constraints:
- Vertex constraints:



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

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

Note on notation & conventions:



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

A flow is a function on edges!
(so are capacities)

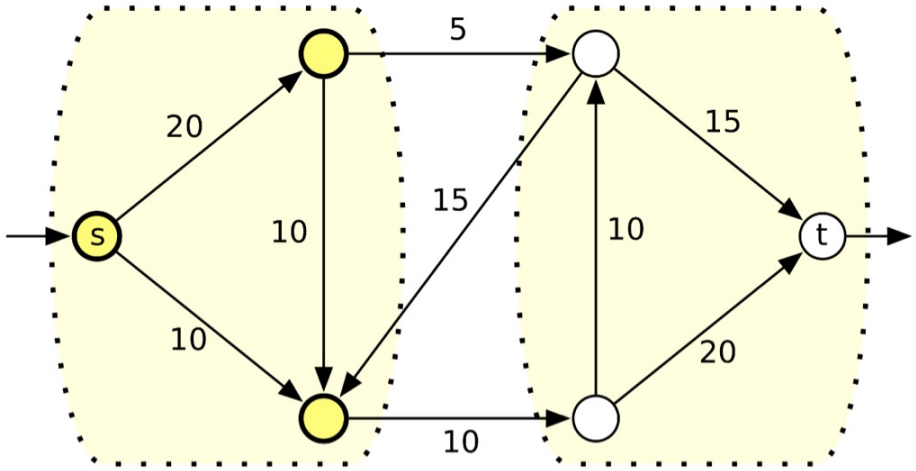
Here:

Formalizing Cuts

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

- $s \in S$
- $t \in T$
- $S \cap T = \phi$,

  $S \cup T = V$



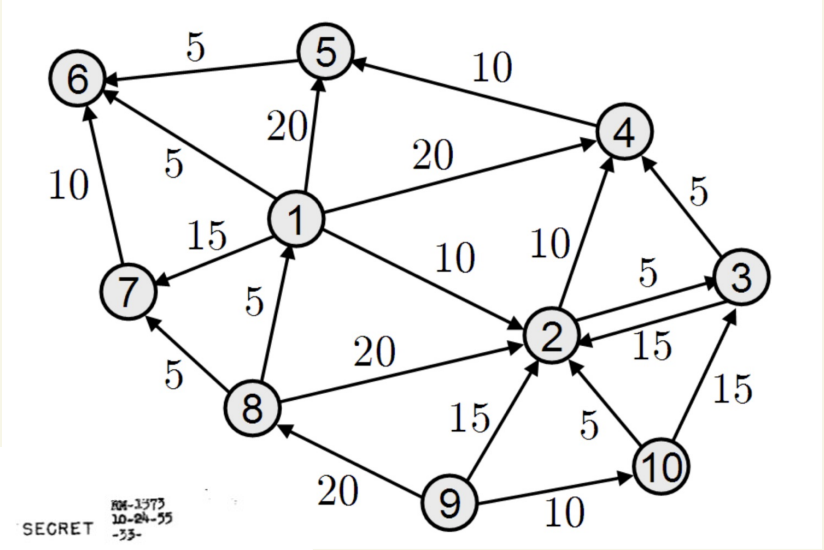
An (s, t)-cut with capacity 15. Each edge is labeled with its capacity.

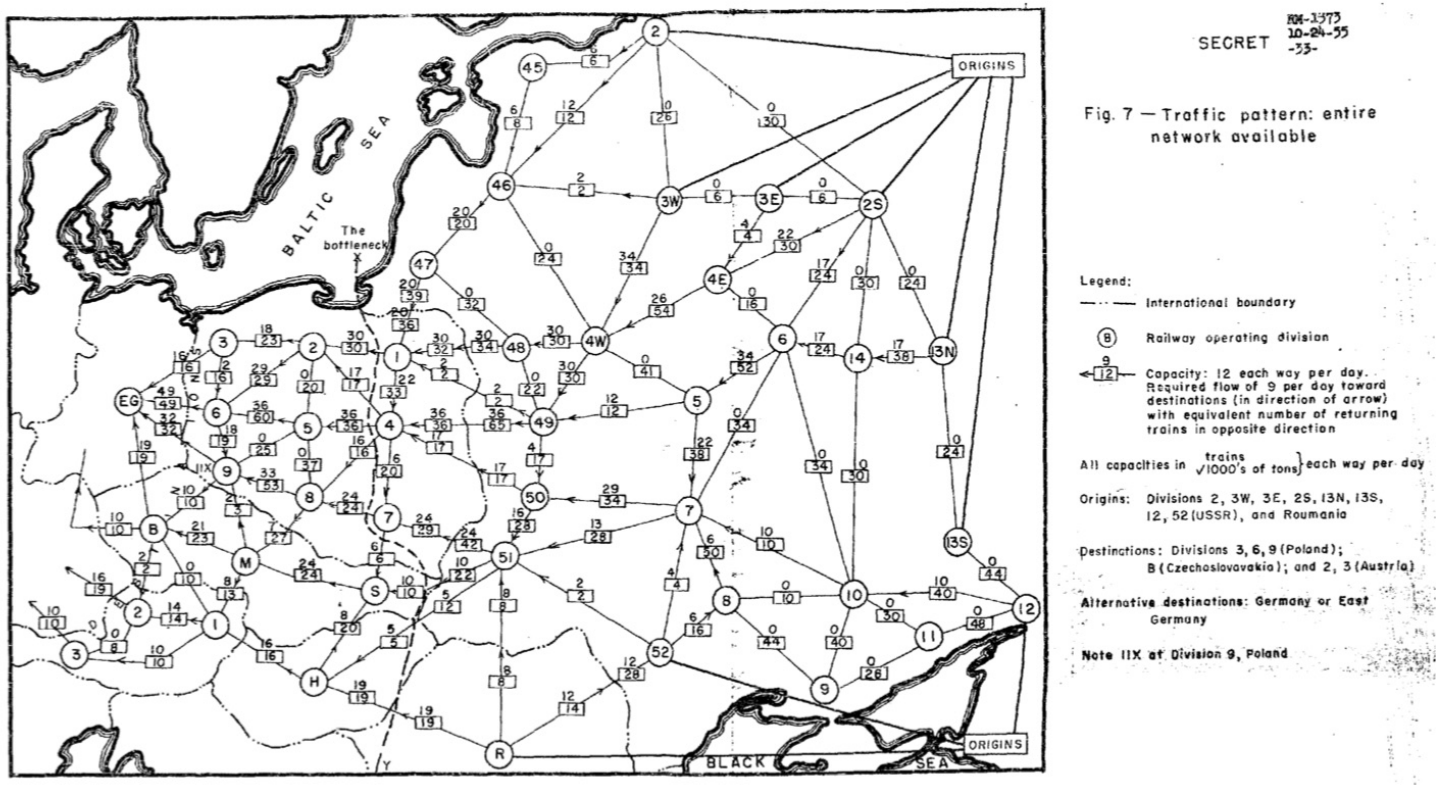
The capacity of a cut is $\sum_{\vec{uv} \in E \text{ with } u \in S, v \in T} c(\vec{uv})$

Min Cuts: not always so obvious!

There are many s-t cuts.

Finding any cut?





SECRET RM-3373 10-24-55 -33-

Fig. 7 — Traffic pattern: entire network available

Legend:

—·—·— International boundary

(8) Railway operating division

9/12 Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

All capacities in trains / 1000's of tons } each way per day

Origins: Divisions 2, 3W, 3E, 2S, 13N, 13S, 12, 52 (USSR), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); B (Czechoslovakia); and 2, 3 (Austria)

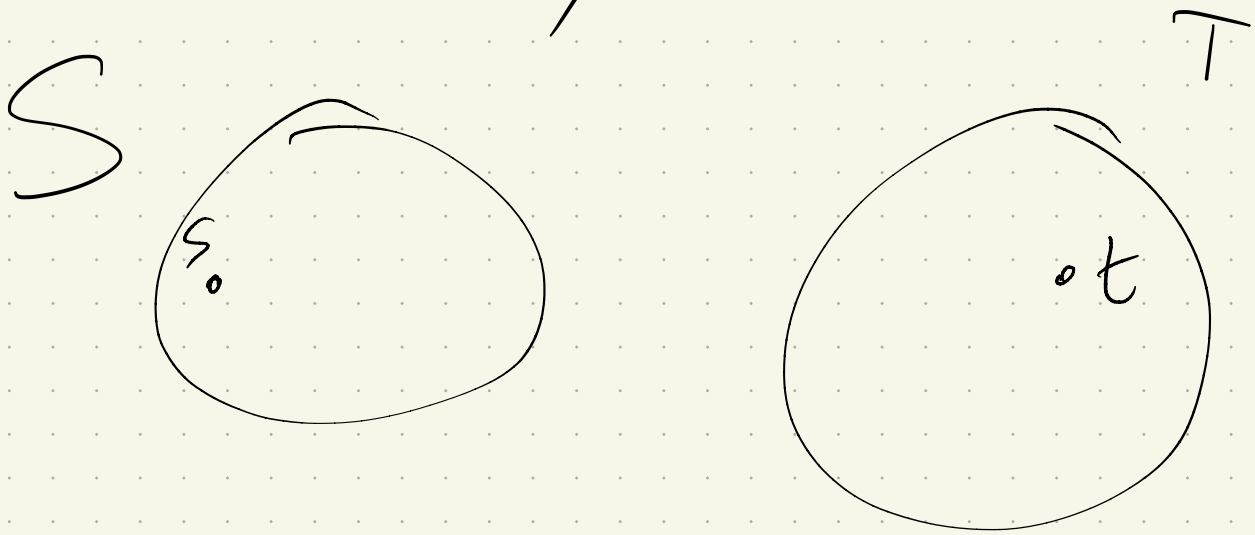
Alternative destinations: Germany or East Germany

Note IIX at Division 9, Poland

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

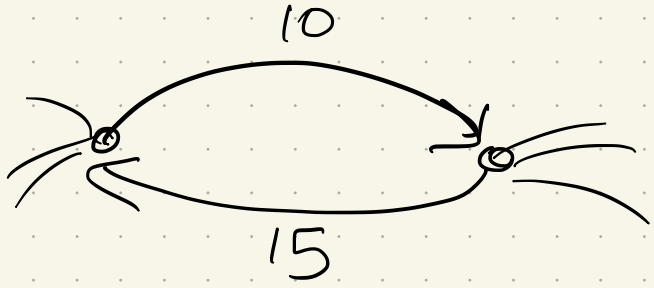
Intuitively, these are connected:

Consider any cut:



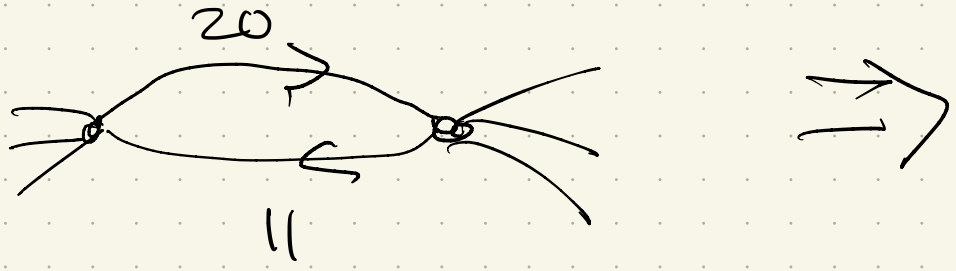
Note: We'll assume every pair of vertices has at most one edge.

So no:



Why? - Makes calculations easier!
(stay tuned for why...)

How? Simple transformation:



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

The max flow value

= min cut value

Wow!.

One way is easy:

Any flow $\leq$ any cut.

Why?

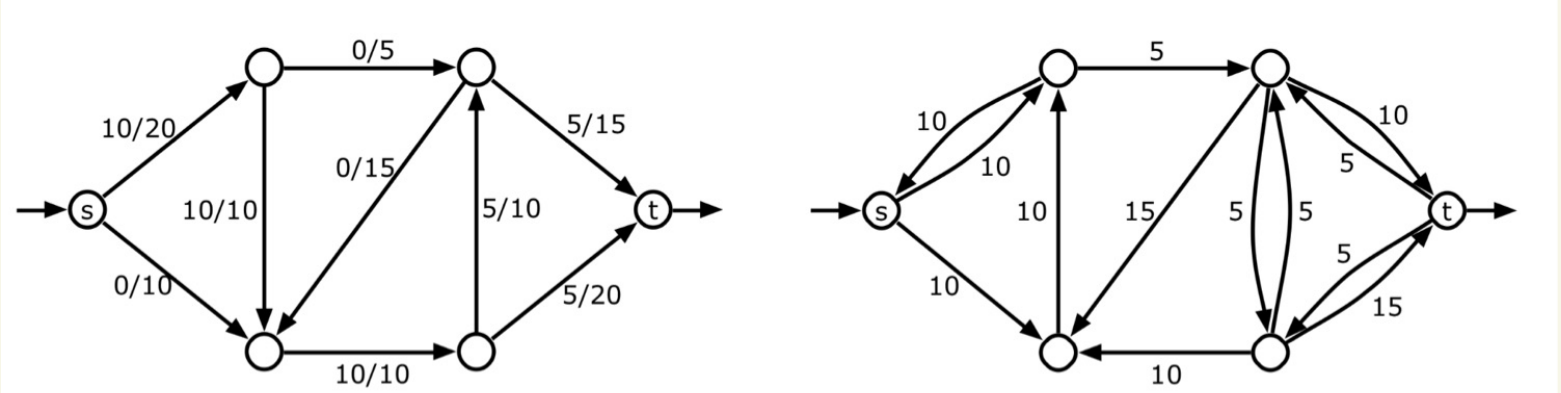
More formally's

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

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

Key tool in proof:

Residual network G_f:

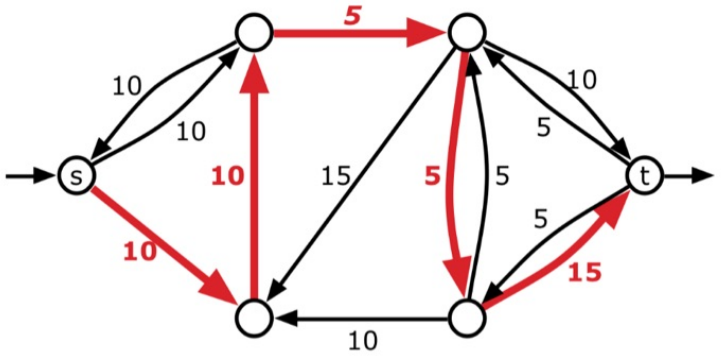


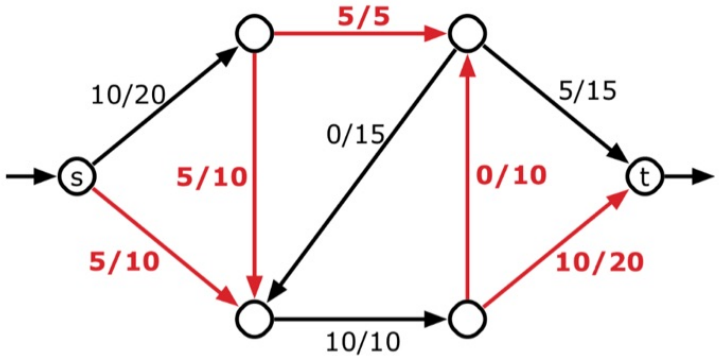
A flow f in a weighted graph G and the corresponding residual graph G_f.

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

f/c ⟹

Augmenting a path:





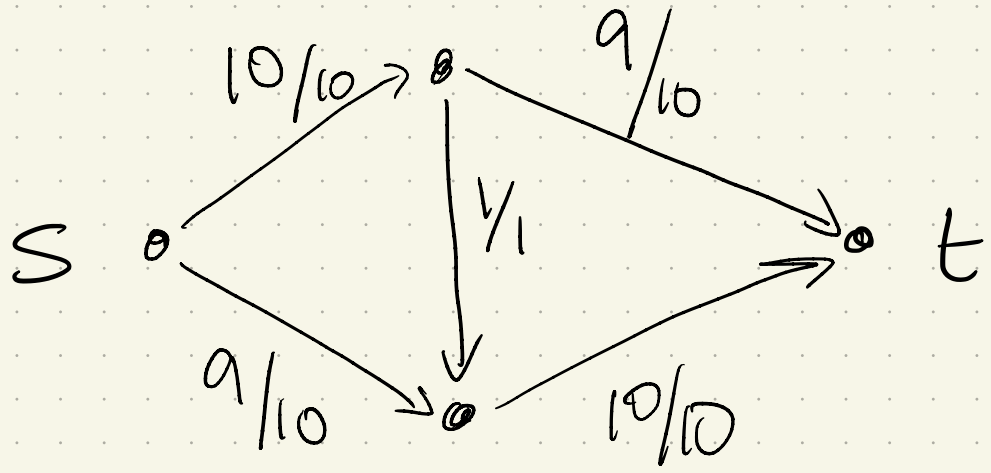
An augmenting path in G_f with value $F = 5$ and the augmented flow f'.

This is just an s-t path in G_f.

Then, find min capacity edge on that path

Claim: I can build a new flow whose value is bigger than f's

Why can't we just be greedy?



Can get "stuck" if we choose wrong initially:

Are there any more flow paths?

Next week:

an algorithm to find max flows

↳ which will prove the F-F theorem along the way.