Algorithms - Spring '25

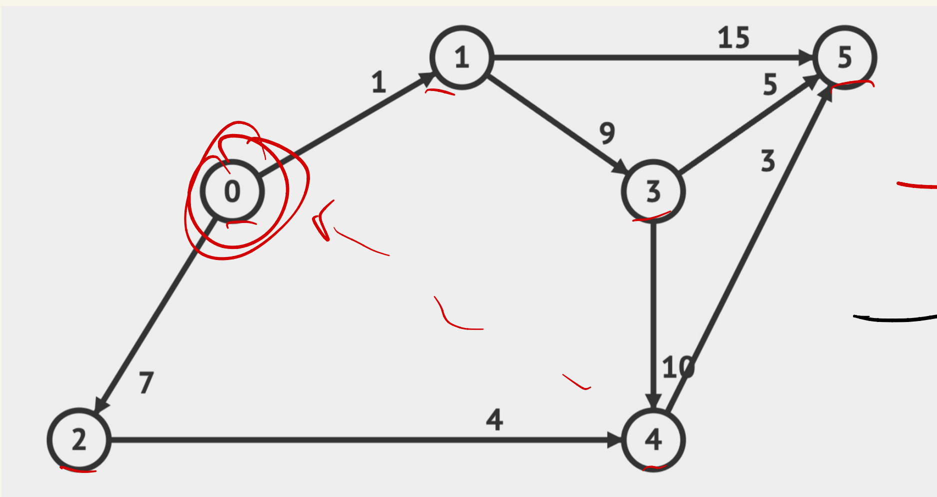
MSSPs
Intro to Flows

Recap

- Readings + HW next week

All Multiple Source Shortest paths

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



dest:

source:

	0	1	2	3	4	5
0	0	1	7	10	11	14
1	∞	0	∞	9	19	15
2	∞		0			
3	∞			0		
4	∞				0	
5	∞	∞	∞	∞	∞	0

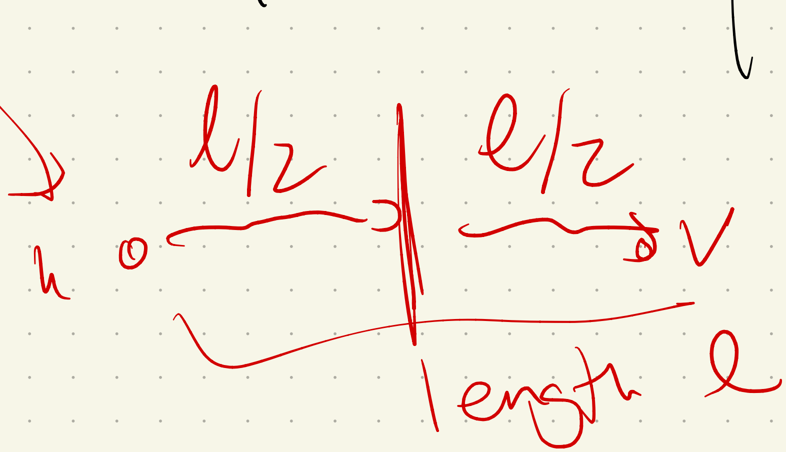
Lookup: $O(1)$:)

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	X	$O(V^3 \log V)$
Fisher: divide + conquer		$O(V^3 \log V)$

$O(V^3 \log V)$

$V \cdot (BF)$



Can we get better?

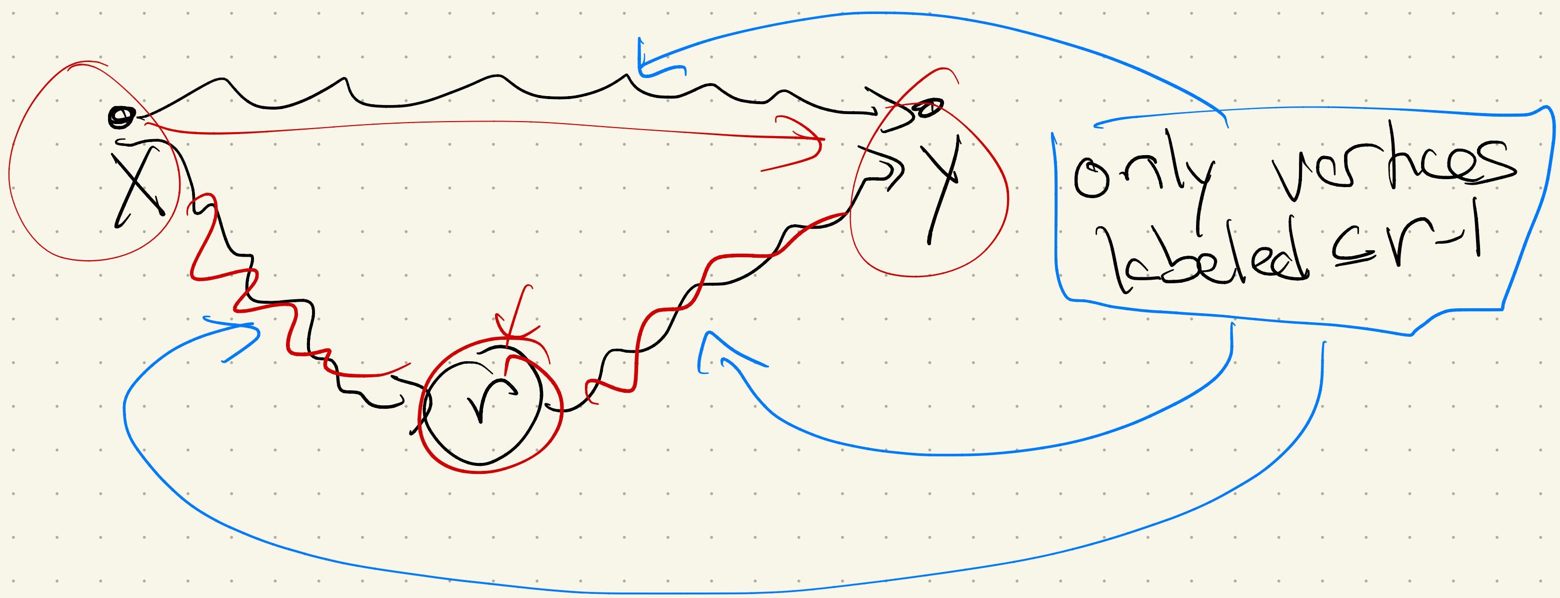
Floyd-Warshall: instead of path length

order vertices $1, \ldots, V$

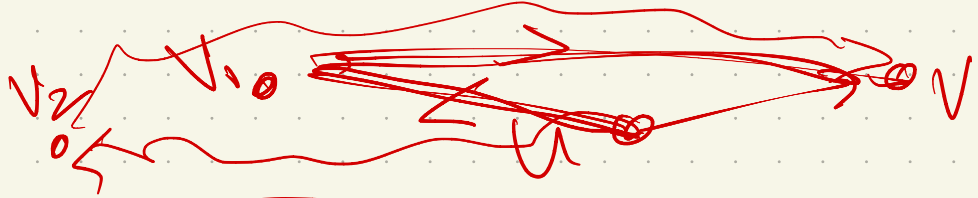
Let $d(x, y, r) =$

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

Then:



Recursion:



$v \circ \infty \circ y$

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

don't use vertex r

use vertex r

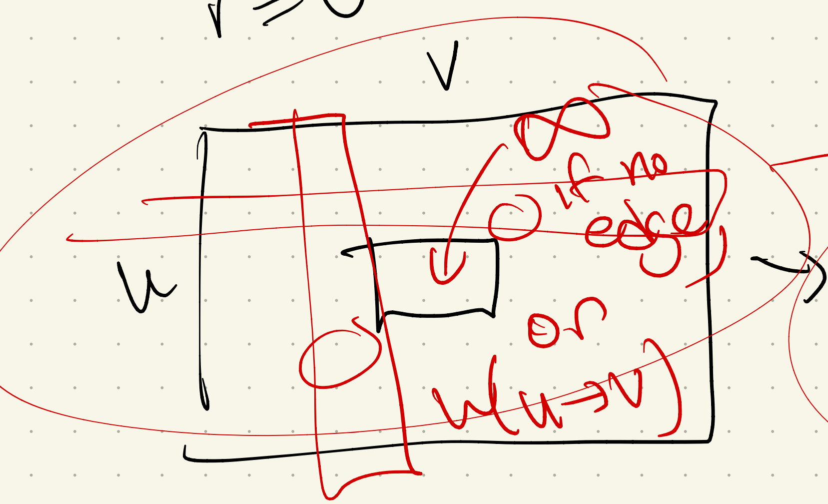
So: for $r \leftarrow 1$ to V

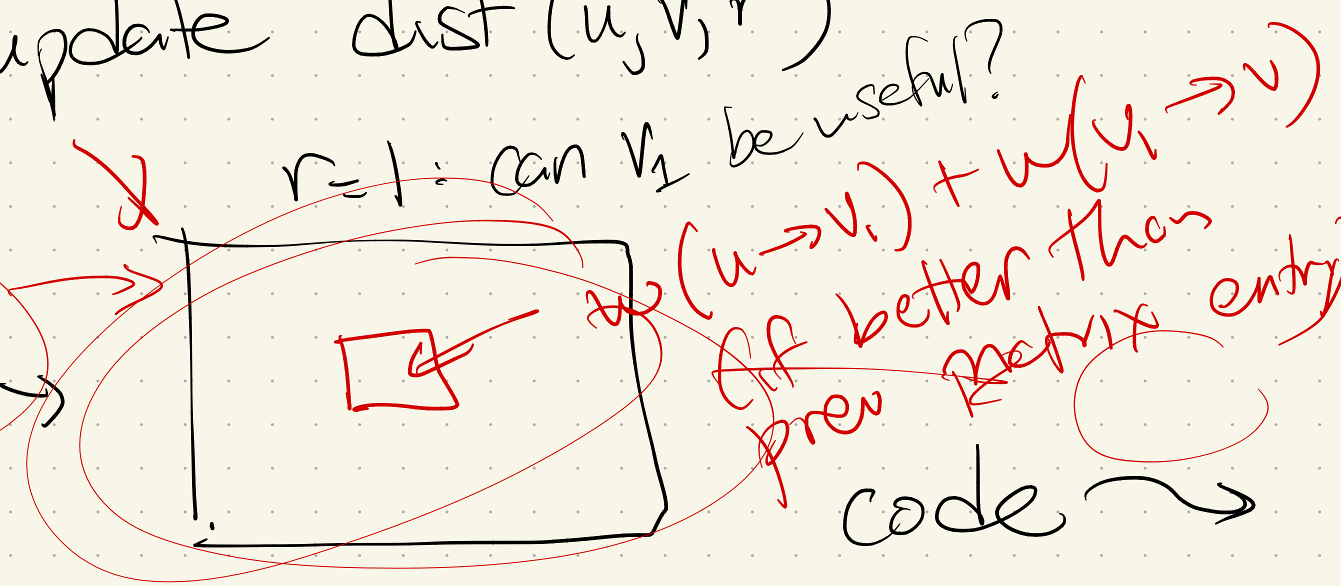
for all $u, v \in G$

update $dist(u, v, r)$

$r = 0$

$r = 1$: can v_1 be useful?





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]
```

(or ∞ if no edge u→v)

Runtime: $O(V^3)$

Space: V^3

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]
```

$O(V^2)$ space

Special cases

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

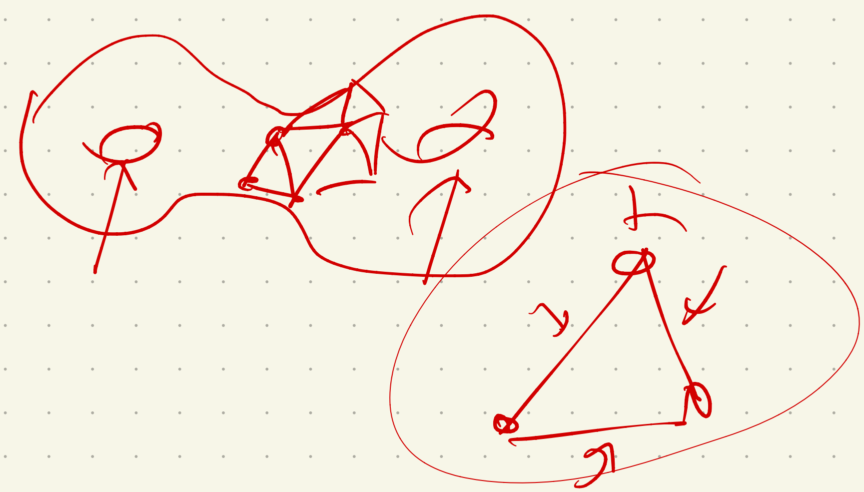
Conjecture: no $O(V^{3-\varepsilon})$ algorithm

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

Meshes (graphs embedded in 3d):

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

genus



Many others...

Ch 10: Flows + Cuts

Motivation:

← railroad

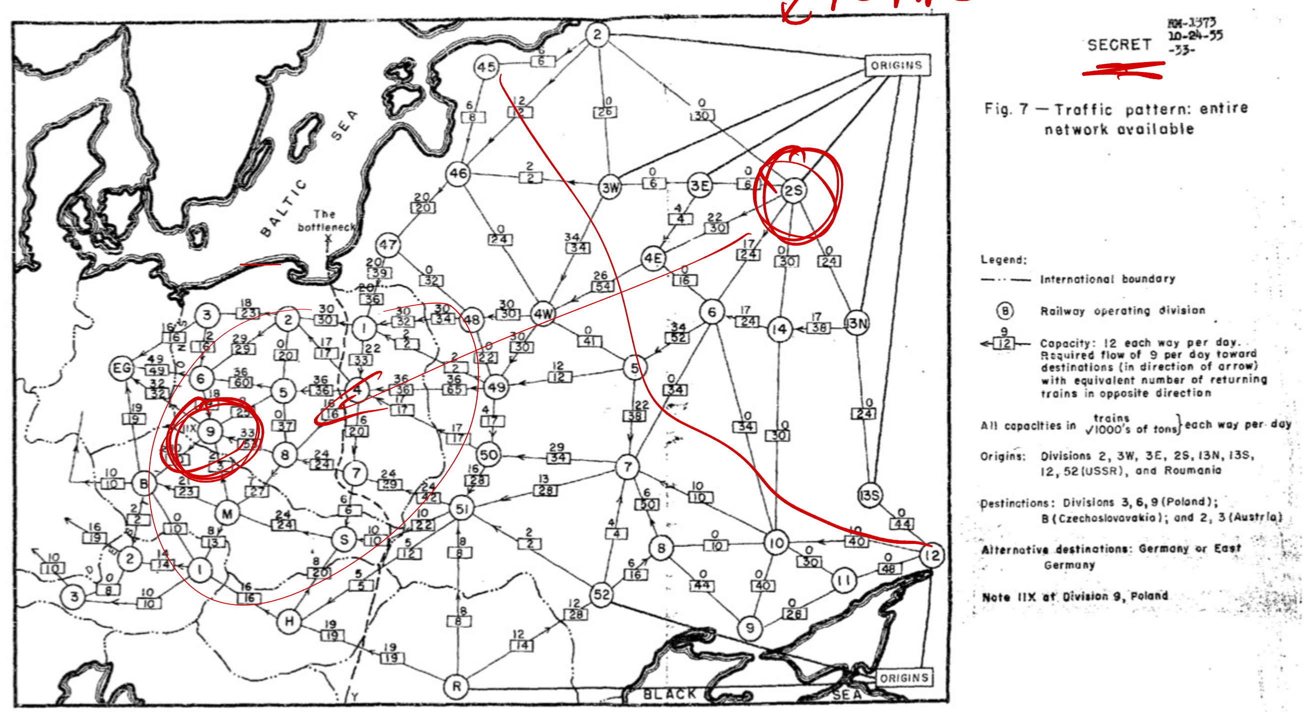


Figure 10.1. Harris and Ross's map of the Warsaw Pact rail network. (See Image Credits at the end of the book.)

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

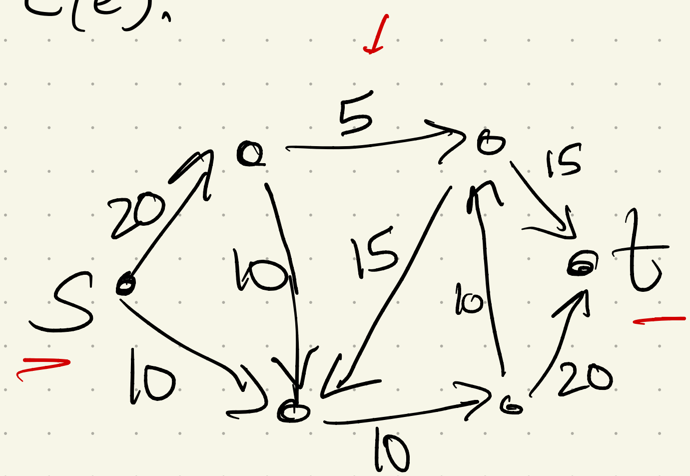
Question: How can I separate u from v with "least cost"?

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 $c(e) \in \mathbb{Z}^+$

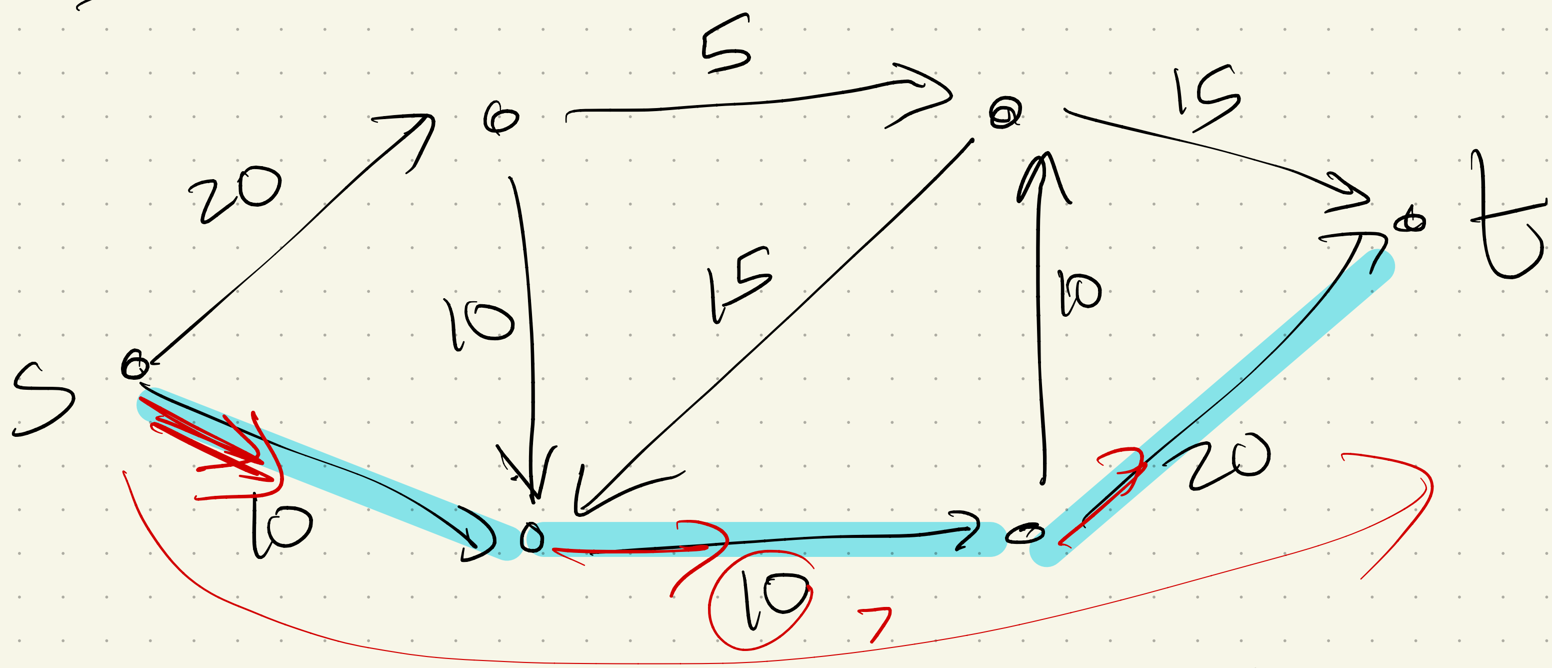


Max flow: Find most I can send from s to t without exceeding edge capacities.

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

Aside:

Not path length:



Consider a path s⇝t:

length: 10+10+20=40

Flow: can send 10 along it

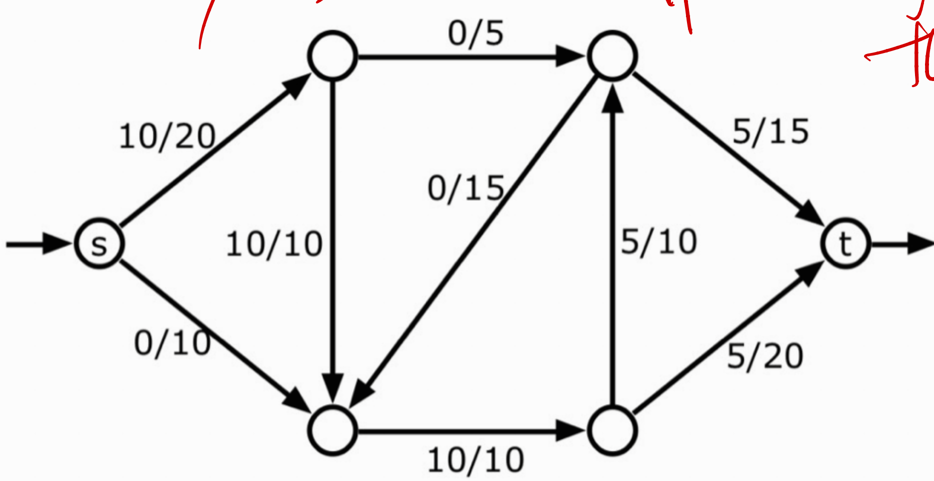
Formalizing flow:

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: $0 \leq f(e) \leq c(e)$ (don't overflow edge)
- Vertex constraints: $\sum_{v \neq s \text{ or } t} \text{flow into } v = \sum \text{flow out}$

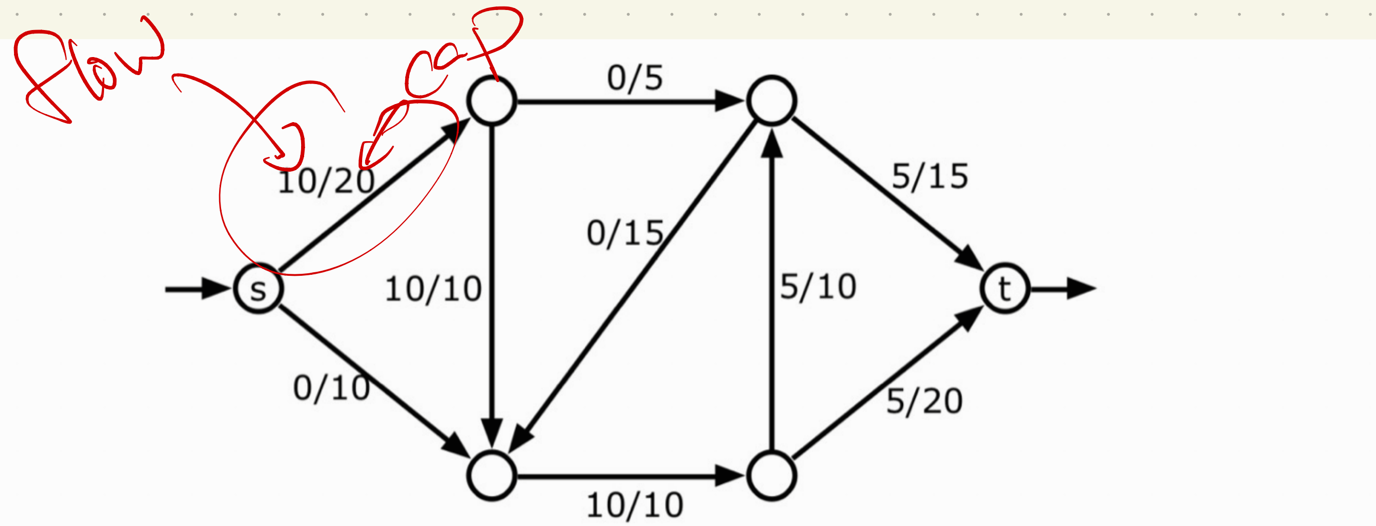
only s can ship out, & no other vertex other than t can store



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

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

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: $\frac{f(e)}{c(e)}$ in figures

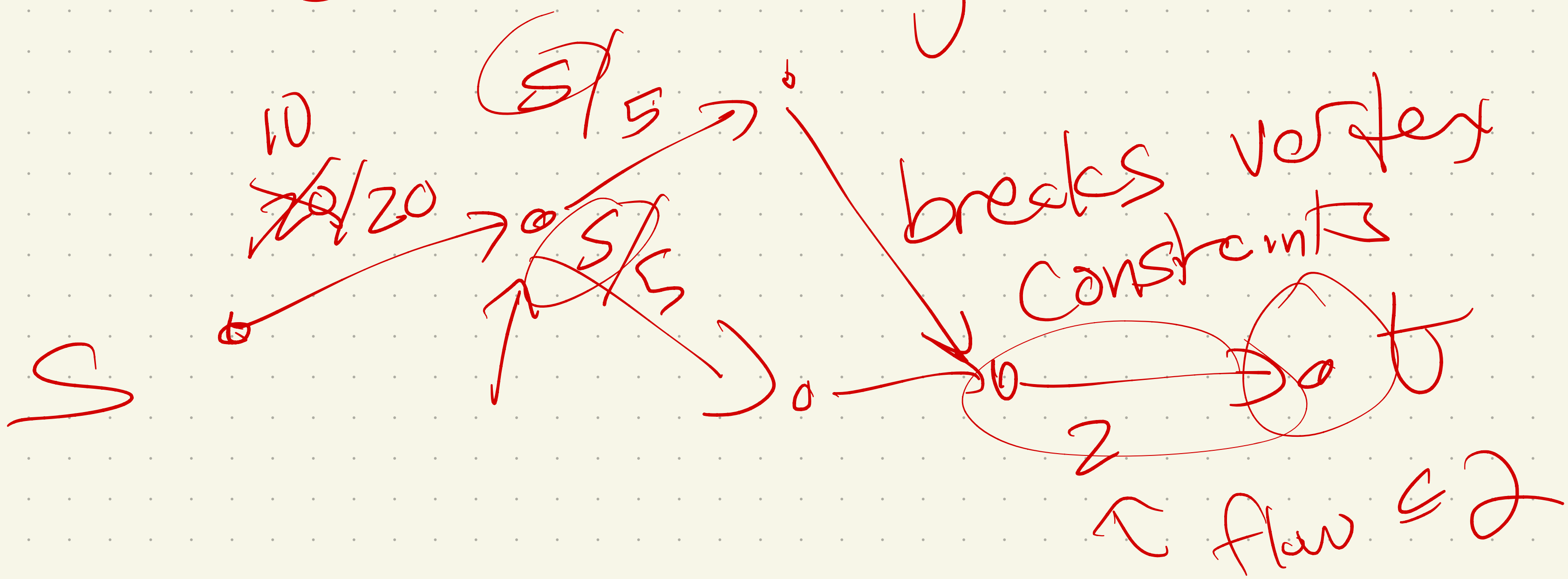
edge constraints $\Rightarrow 0 \leq \frac{f}{c} \leq 1$

Possible flows:

$f(e) = 0 \quad \forall e \in E$ ←

"0-flow"

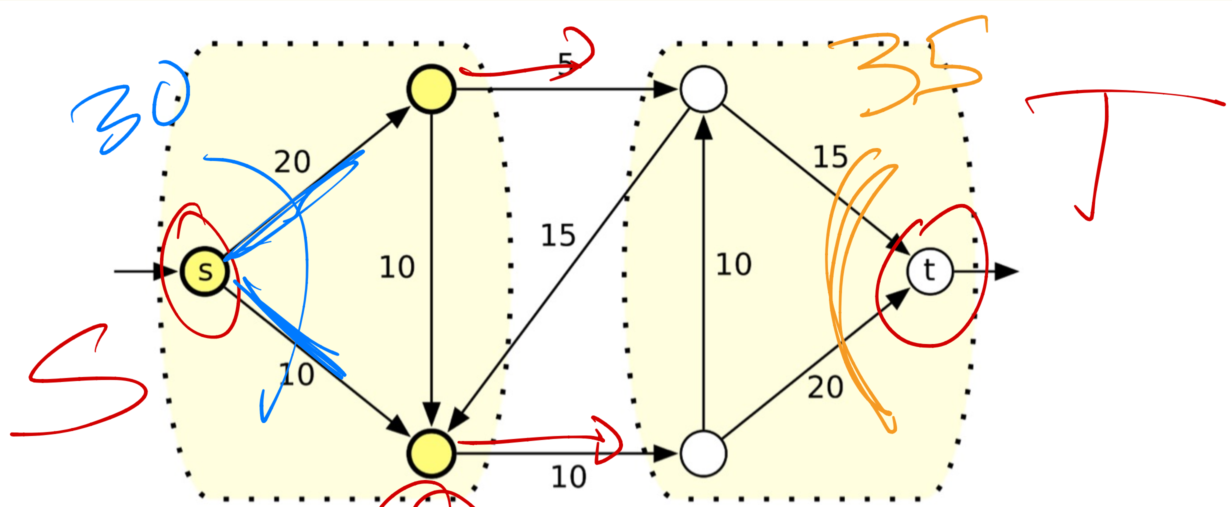
Not: set $f(e) = c(e)$ on all edges



Formalizing Cuts

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

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



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

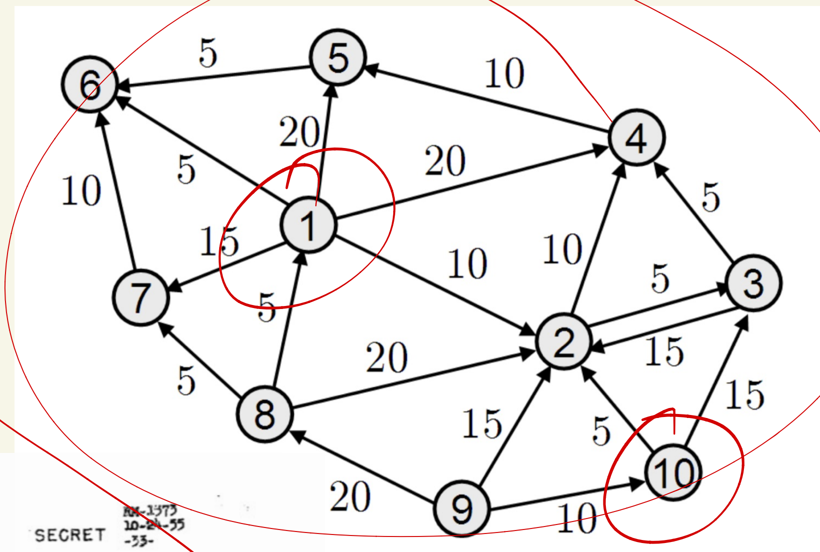
S, T is a partition of V

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

Min Cuts: not always so obvious!

There are many s-t cuts.

Finding any cut? easy



S
v_i
T
S_0
t

2^{V-2} cuts

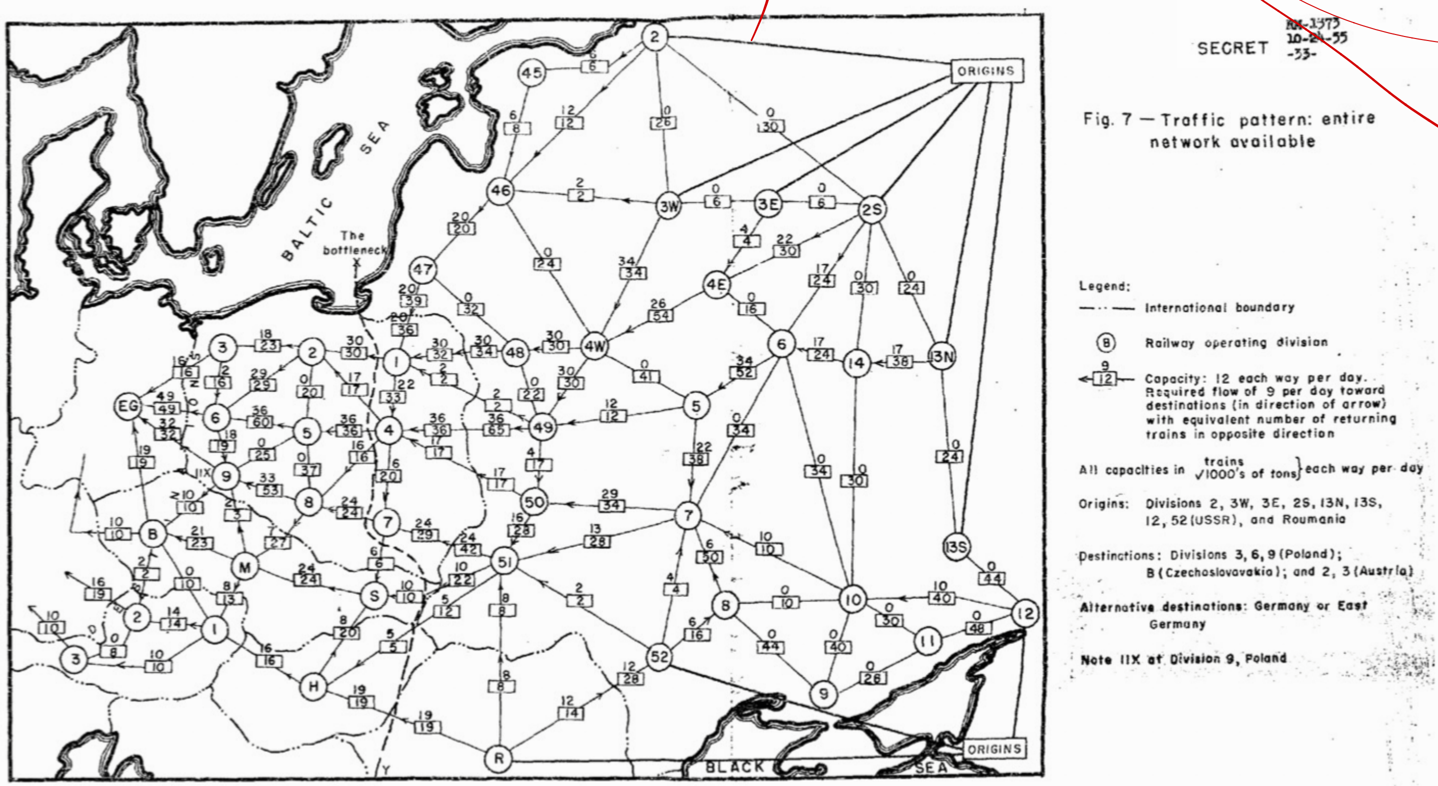


Figure 10.1. Harris and Ross's map of the Warsaw Pact rail network. (See Image Credits at the end of the book.)

Intuitively, these are connected: any flow ≤ any cut

Consider any cut:

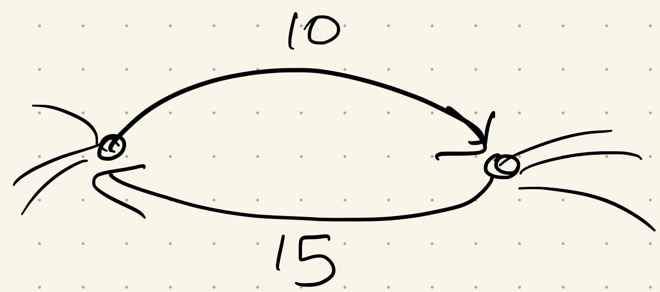
cap of cut: edges from S to T

Any flow must use these edges

Amount of flow can't exeed cap. of edges in cut.

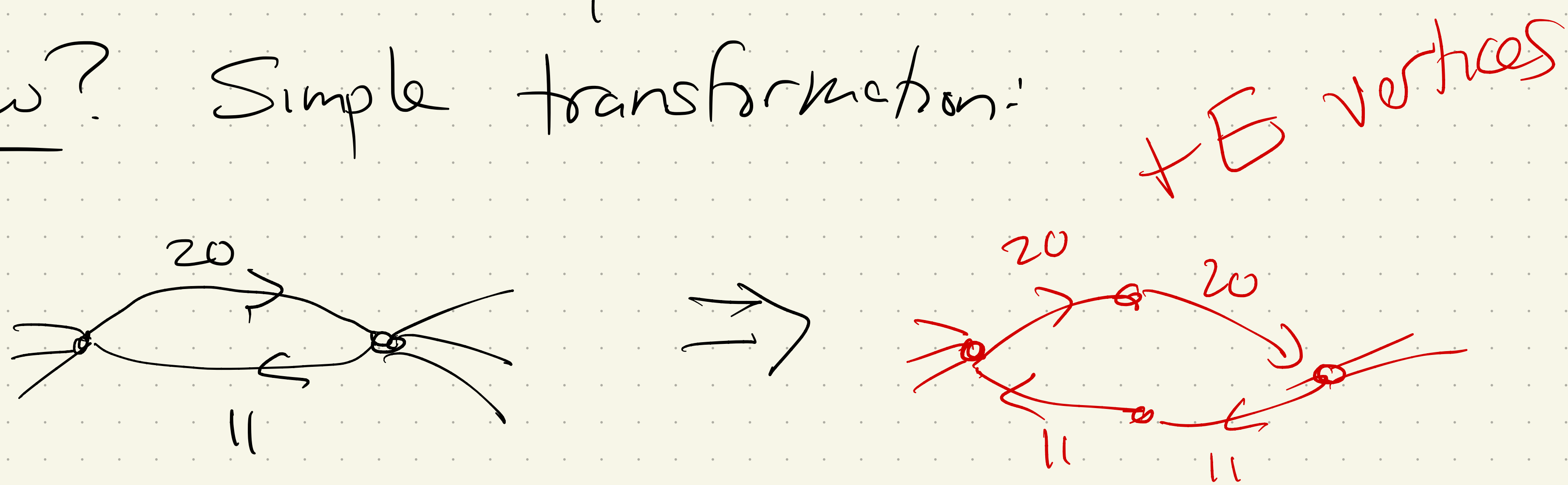
Note: We'll assume every pair of vertices has at most one edge.

So no:



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

How? Simple transformation:



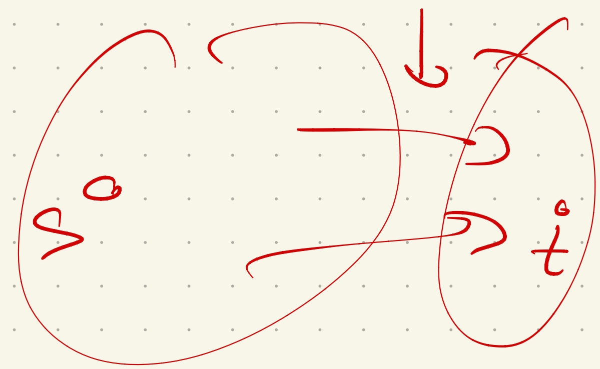
Thm: (Ford-Fulkerson '54, Elias-Feinstein-Shannon '56)

The max flow value = min cut value

Wow!

One way is easy:

Any flow ≤ any cut.



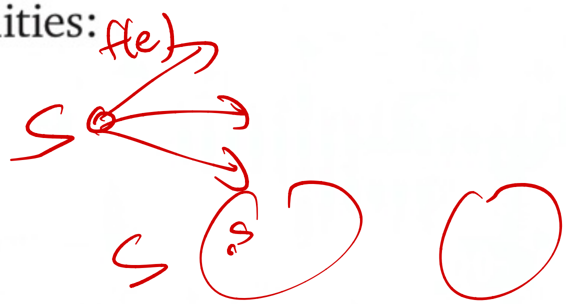
Why?

Can exceed edges out of S + into T

More formally:

any flow ≤ any cut

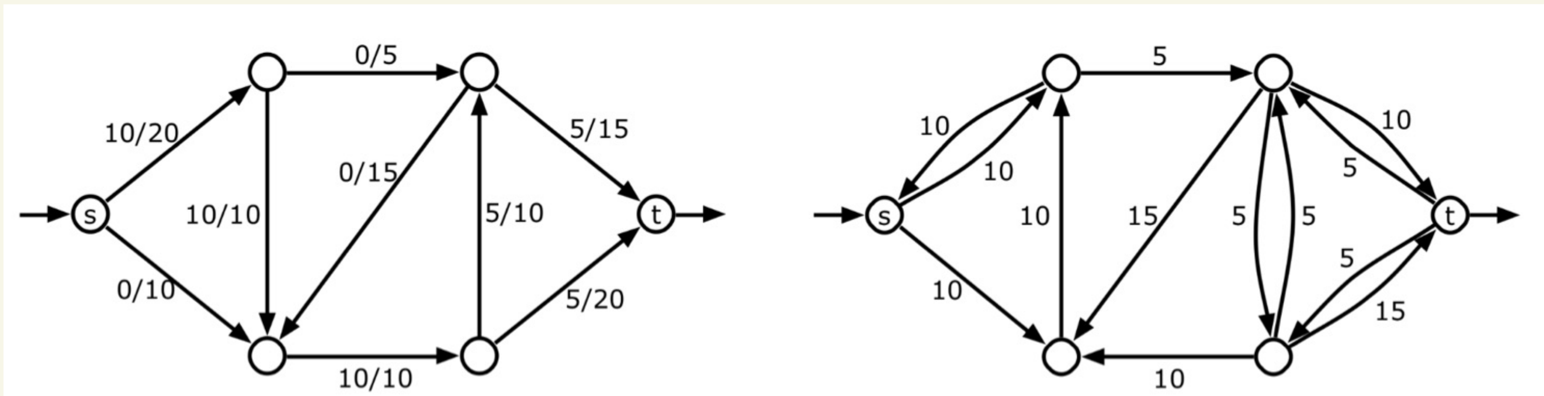
Proof: Choose your favorite flow f and your favorite cut (S, T), and then follow the bouncing inequalities:



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

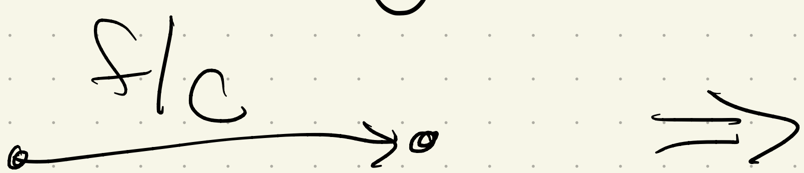
Key tool in proof:

Residual network G_f:

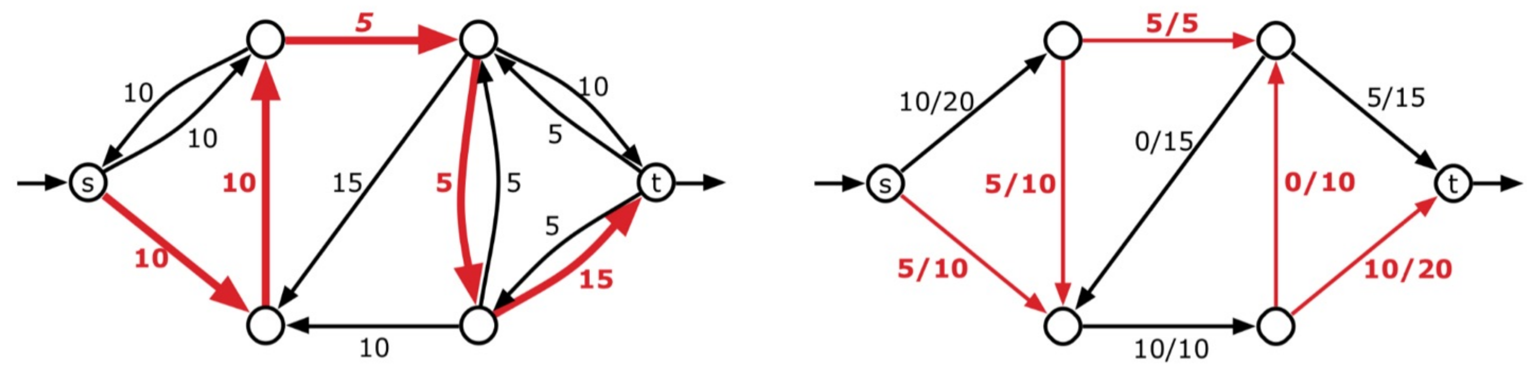


A flow f in a weighted graph G and the corresponding residual graph G_f.

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



Augmenting a path:



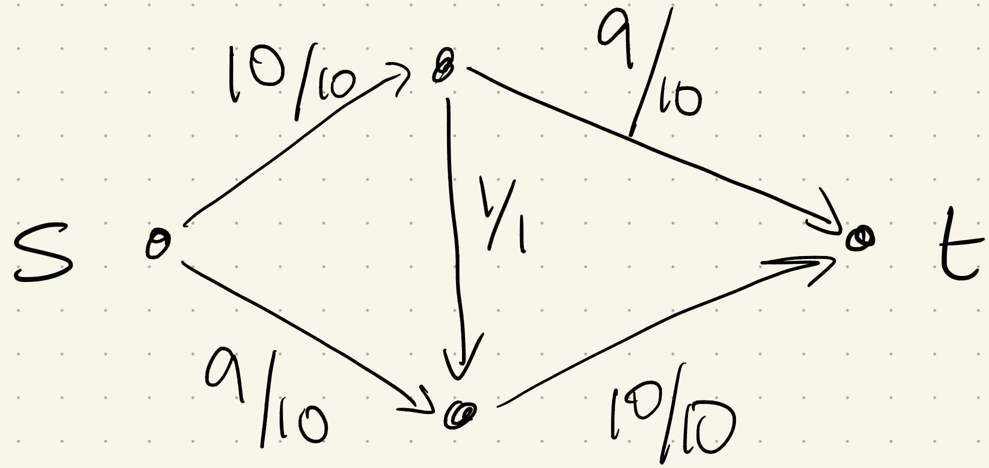
An augmenting path in G_f with value $F = 5$ and the augmented flow f'.

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.