Complexity & Algorithms, Spring '26

Max Flow & Min Cut

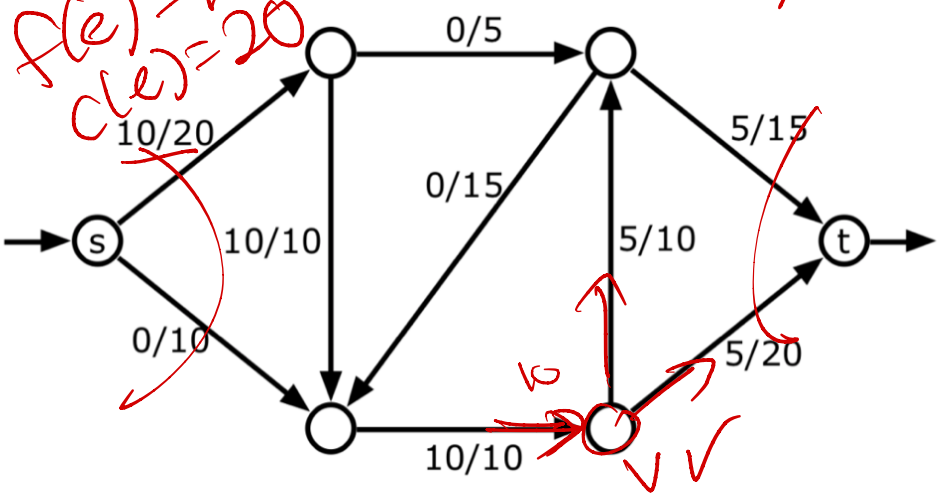
A flow is a function $f: E \to \mathbb{R}^+$, where $f(e)$ is the amount of flow going over edge e

Must satisfy 2 things

Edge constraints: $0 \le f(e) \le c(e)$

Vertex constraints: $\forall v, \sum_{e=u\to v} f(e) = \sum_{e'=v\to x} f(e')$

s, t ✗



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

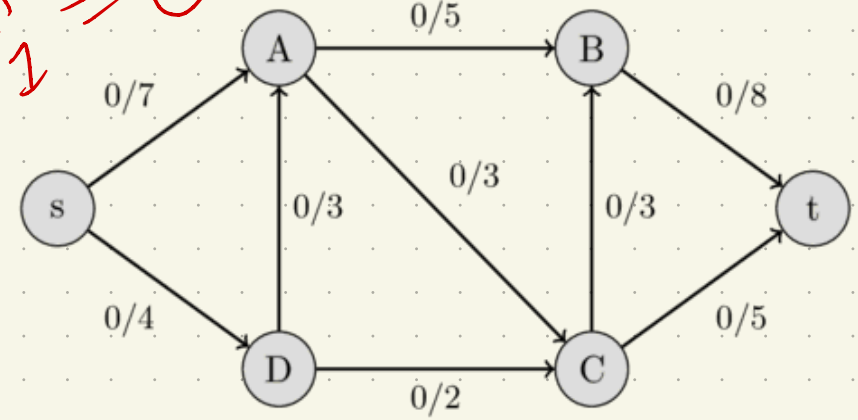
Value(f) =

$$\sum_{e=s\to v} f(e) = \sum_{e'=u\to t} f(e')$$

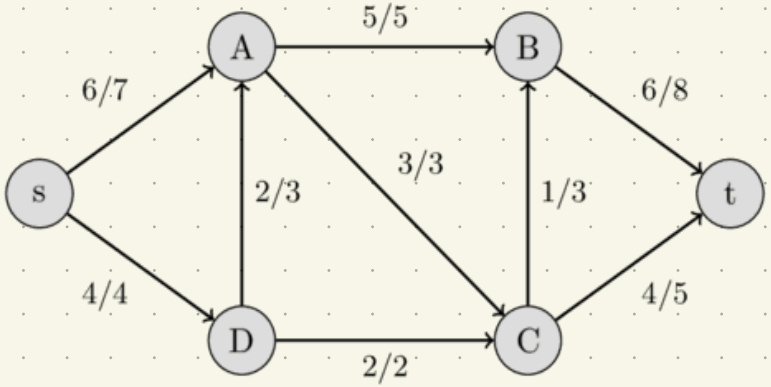
Note: Fix G, with $c: E \to \mathbb{Z}^+$.

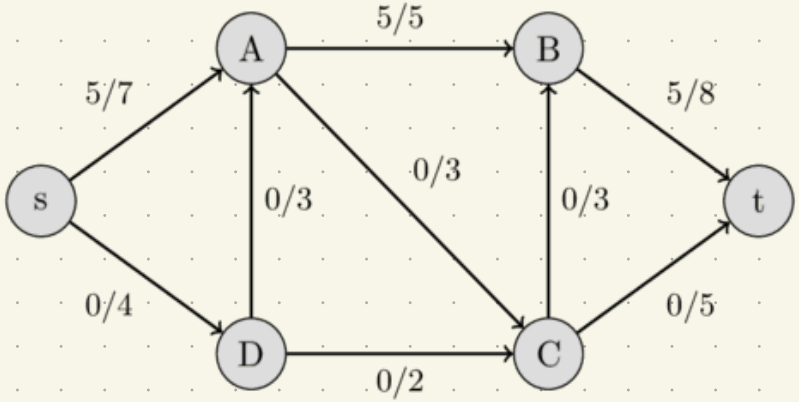
There are many flows:

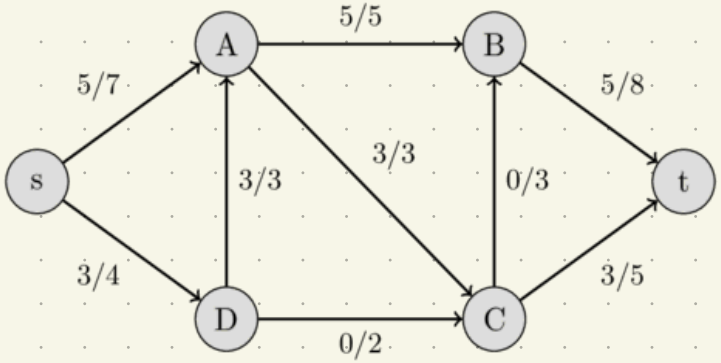
$f_1 = 0$



f_2:





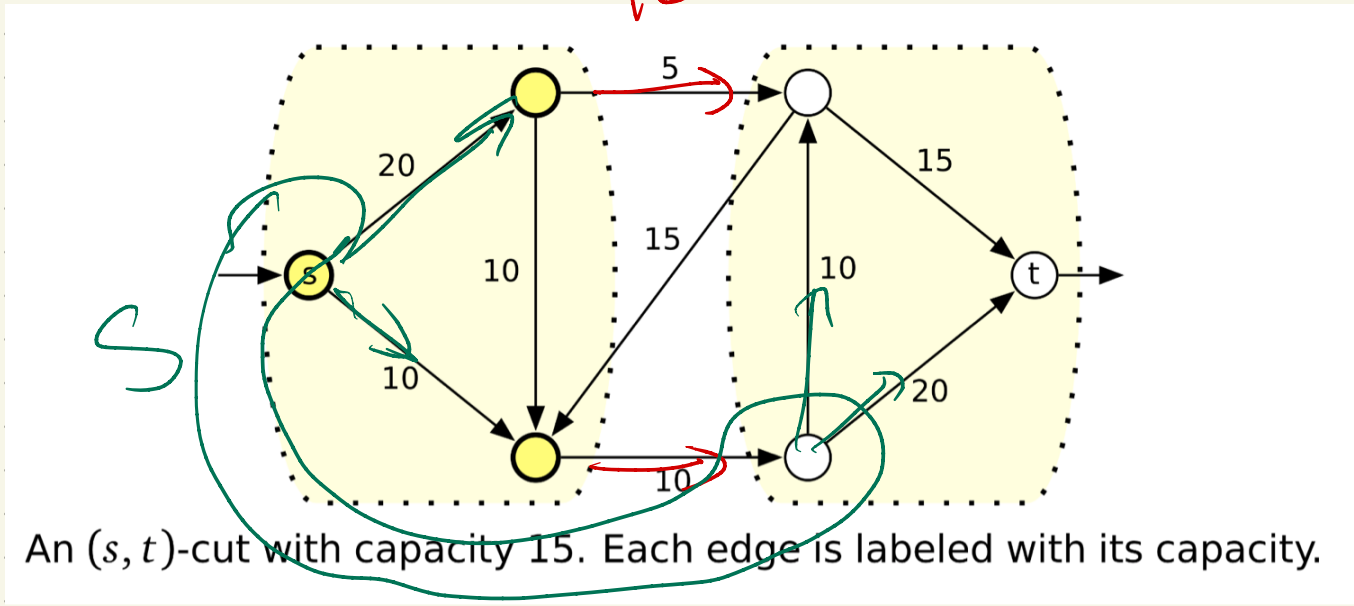


Goal: make it big

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

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

  $S \cup T = V$

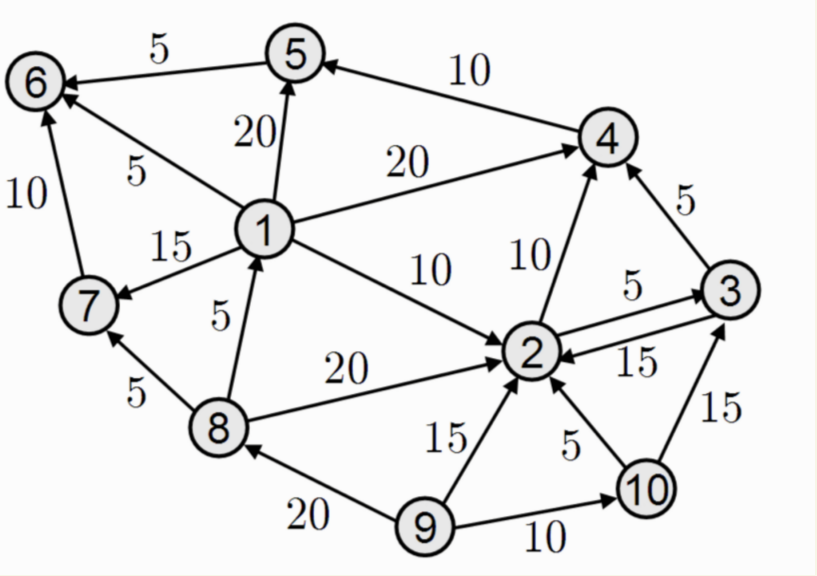


An (s, t)-cut with capacity 15. Each edge is labeled with its capacity.

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

Not unique! 2^{V-2} in total

Cuts: Not always obvious!



Goal: Minimize the value of cut

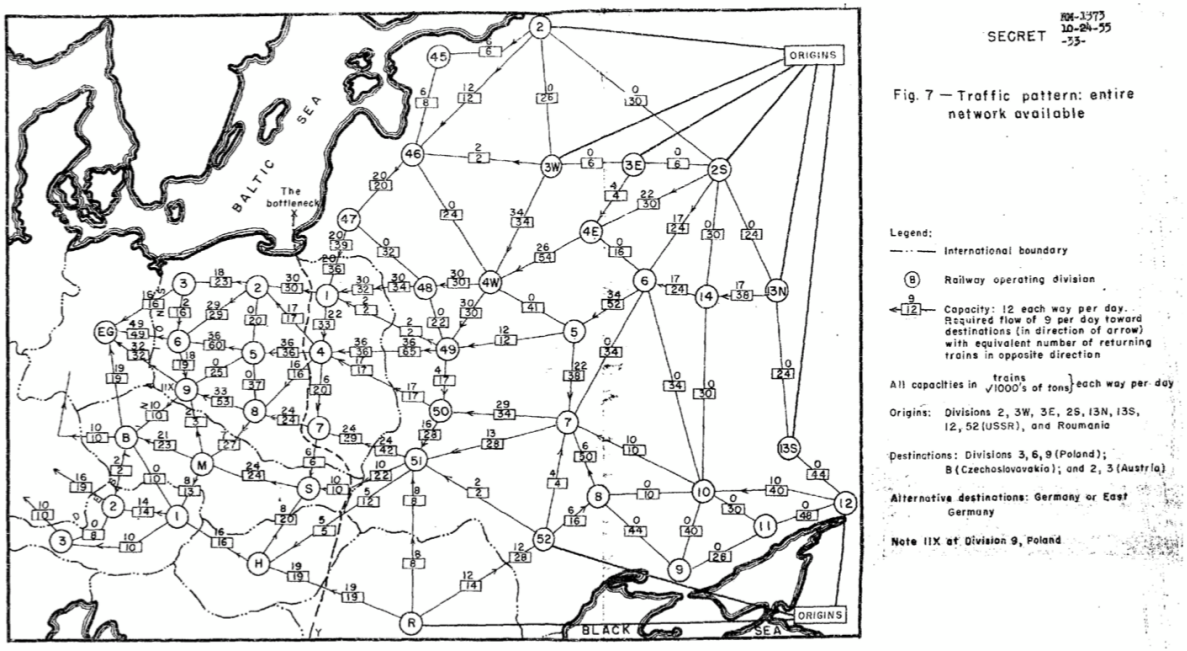


Figure 10.1. Harris and Ross's map of the Warsaw Pact rail network. (See Image Credits at the end of the book.)

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

The max flow value

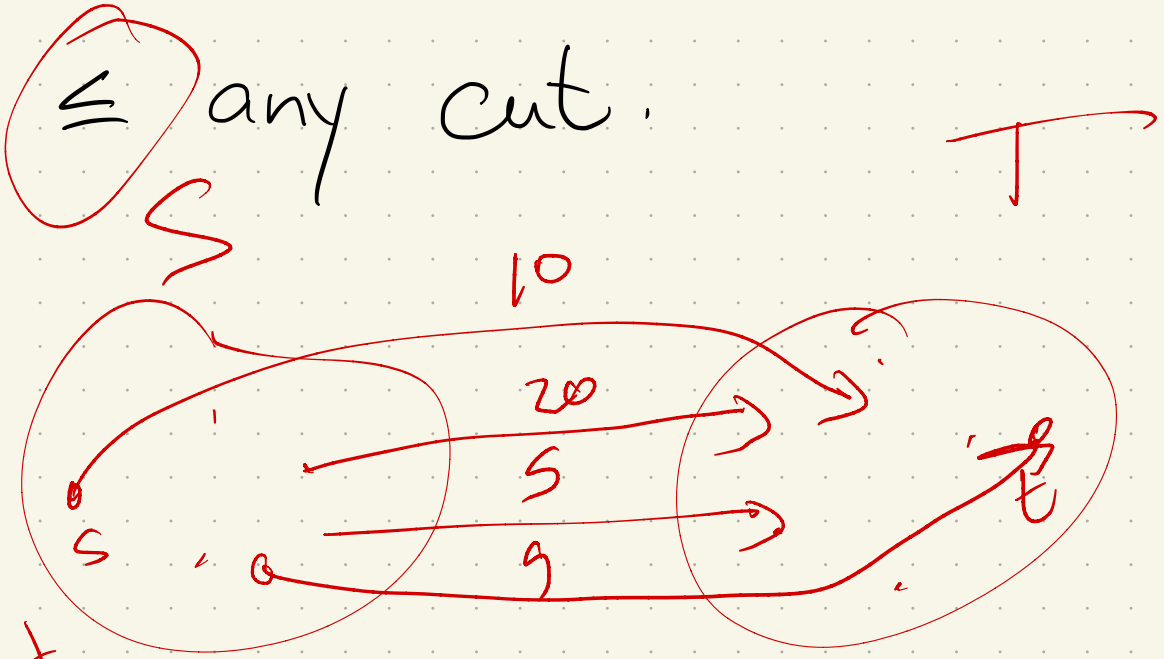
= min cut value

Wow!

One way is easy:

Any flow ≤ any cut.

Why?



any flow ≤ this cut

More formally:

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

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

↳ slower?

More carefully:

Choose flow f + cut (S, T)

$$\text{value}(f) = \partial f(s) = \sum_{s \to u} f(s) - \sum_{u \to s} f(u)$$

$f_{out}(s)$ $\qquad$ $f_{in}(s)$

Then, since flow in = flow out for all $v \neq s$ or t, $f_{in}(v) = f_{out}(v)$

$$= \sum_{v \in S} \partial(v) \quad \text{by dfn of } \partial$$

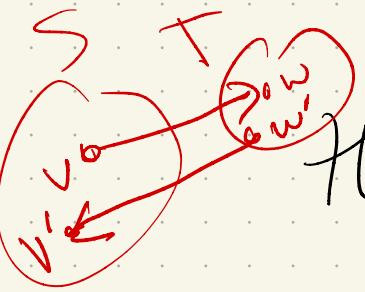
$$= \sum_{v \in S} \left[\sum_{w} f(v \to w) - \sum_{w} f(w \to v) \right]$$

$f_{out}(v) - f_{in}(v)$

& any edge entirely on S side is counted twice ↑

$\Rightarrow$

$$= \sum_{v \in S} \sum_{w \notin S} f(v \to w) - \sum_{v' \in S} \sum_{w' \notin S} f(w' \to v')$$

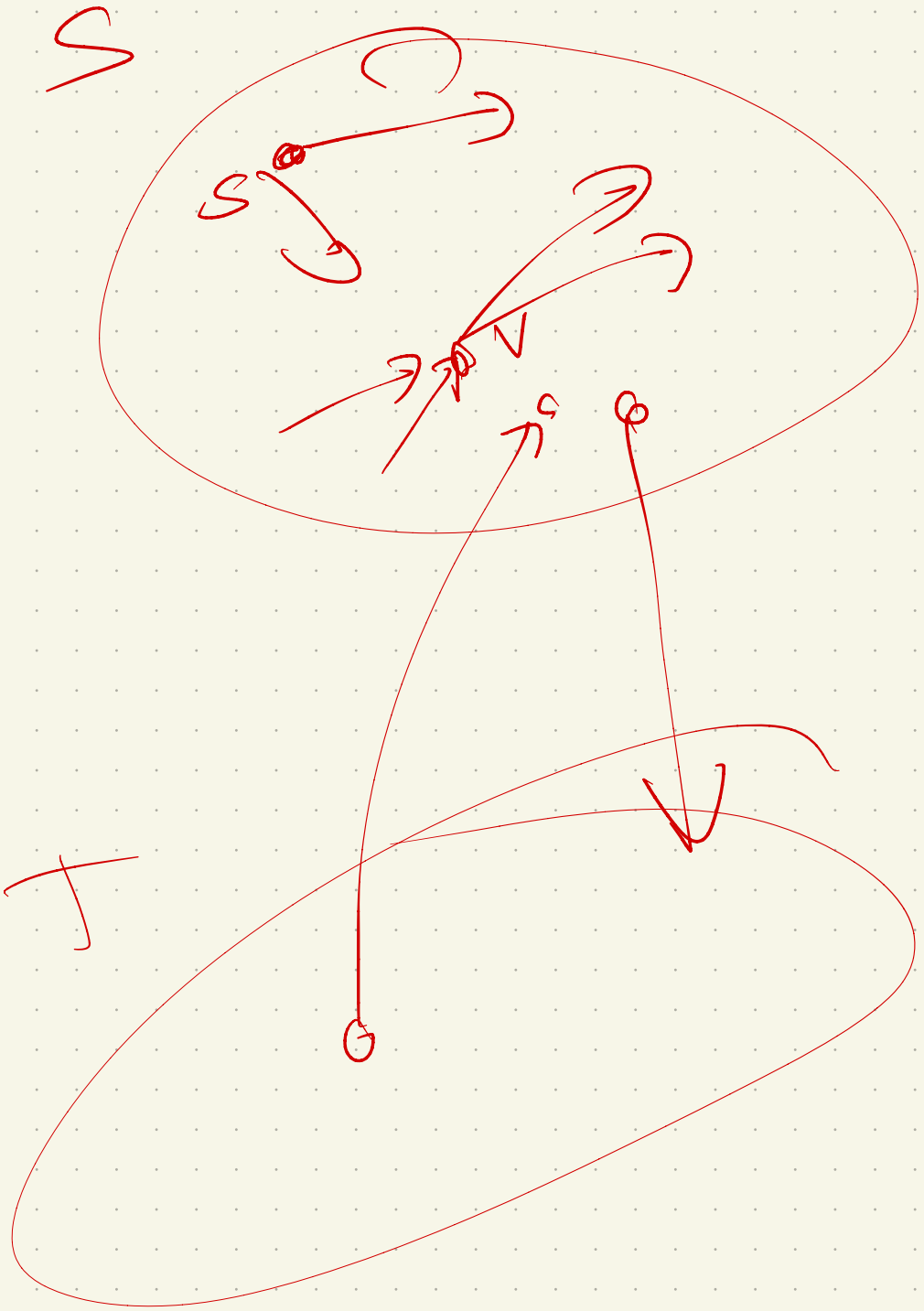


then if $w \notin S$, know $w \in T$
(because it's a cut!)

$$= \sum_{v \in S} \sum_{w \in T} f(v \to w) - \underbrace{\sum_{v \in S} \sum_{w \in T} f(w \to v)}_{\geq 0}$$

so $\leq \sum_{v \in S} \sum_{w \in T} f(v \to w)$ & flow $\leq$ cap

$$\Rightarrow \leq \sum_{v \in S} \sum_{w \in T} c(v \to w)$$



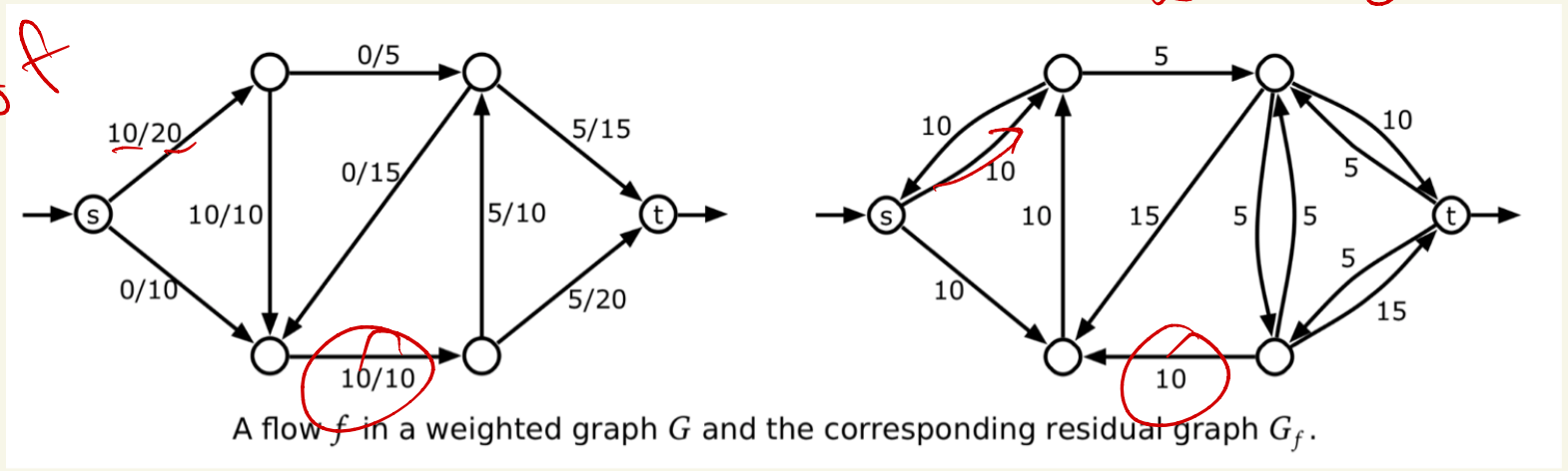
$$f_{out}(S) = \sum_{v \in S} \left(f_{in}(v) - f_{out}(v) \right)$$

Key tool in proof:

for other way

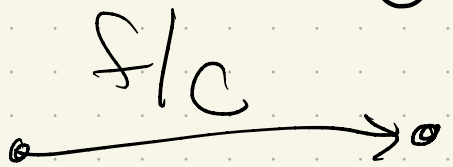
Residual network G_f:

→ G_f based on f.

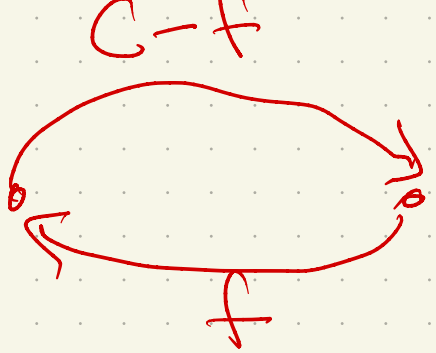


A flow f in a weighted graph G and the corresponding residual graph G_f.

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

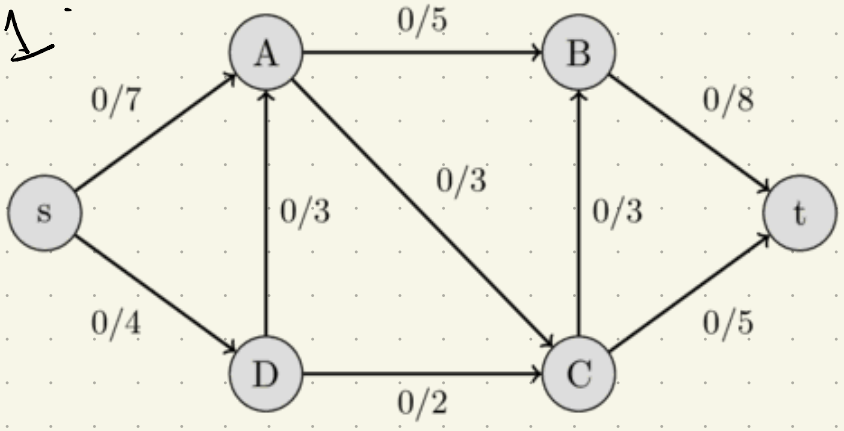


⇒

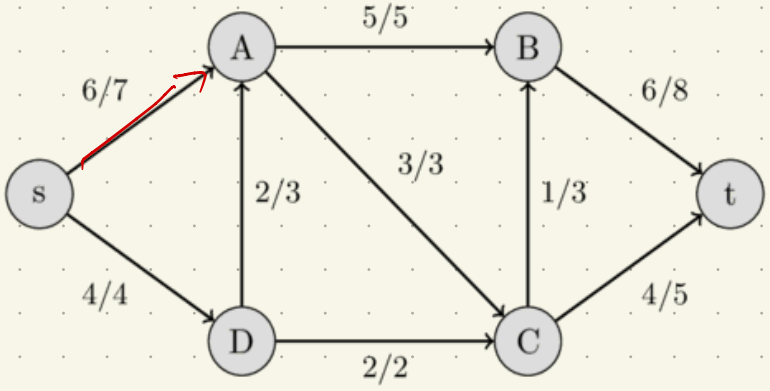


Note: Every f gives a different G_f!

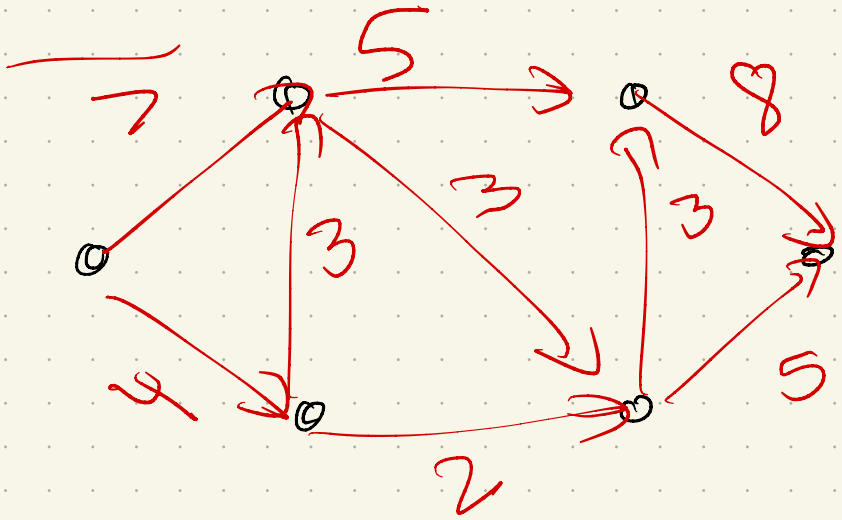
f_1:



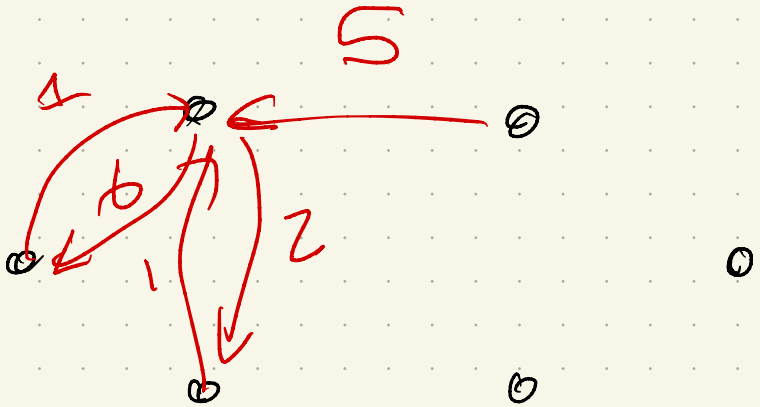
f_2:



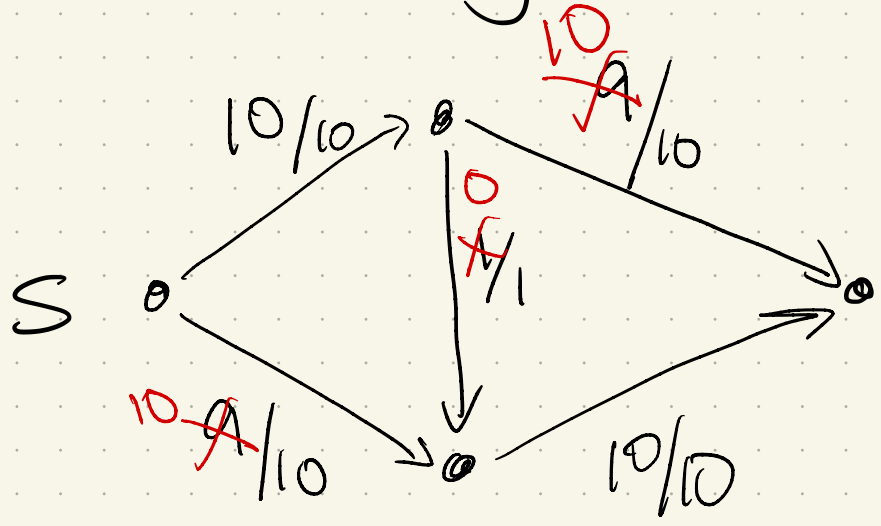
G_{f_1}:

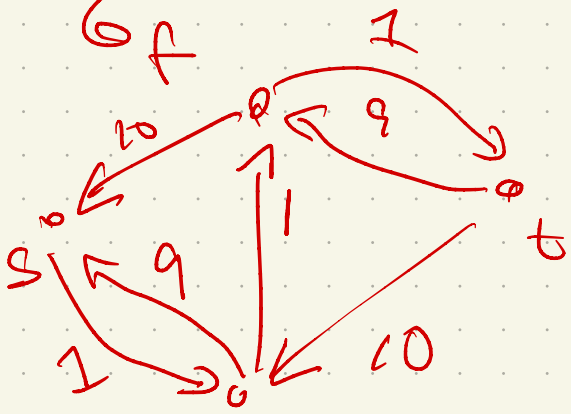


G_{f_2}:



Why can't we just be greedy & push?





Can get "stuck" if we choose wrong initially:

Are there any more flow paths?

No

More formally: Residual network G_f:

Given G & f:

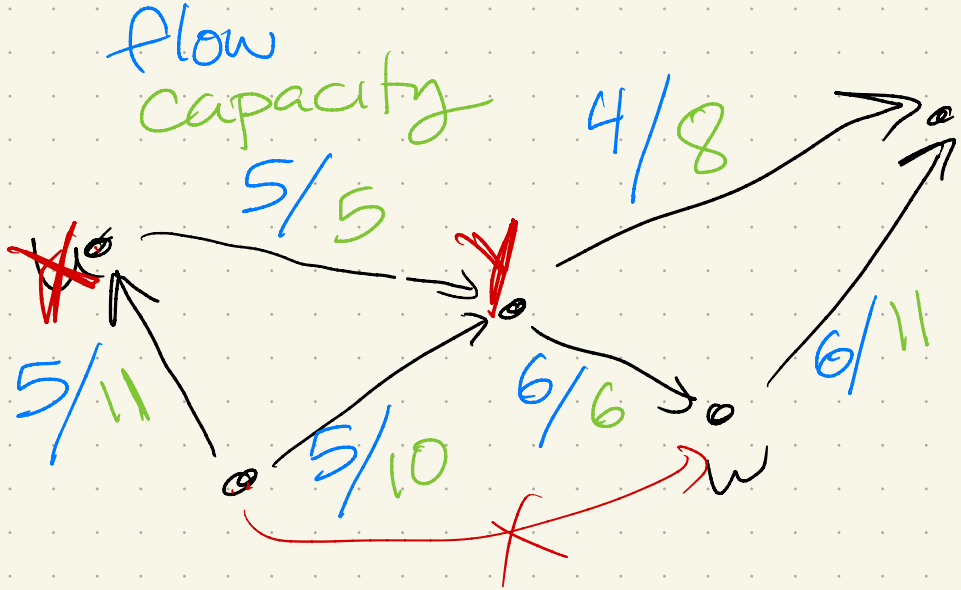
set vertices of G_f = $V(G)$

$$c_f(u \to v) = \begin{cases} c(u \to v) - f(u \to v) & \text{if } u \to v \text{ is in } E \\ f(u \to v) & \text{if } v \to u \text{ is in } E \\ 0 & \text{otherwise (no edge or 0)} \end{cases}$$

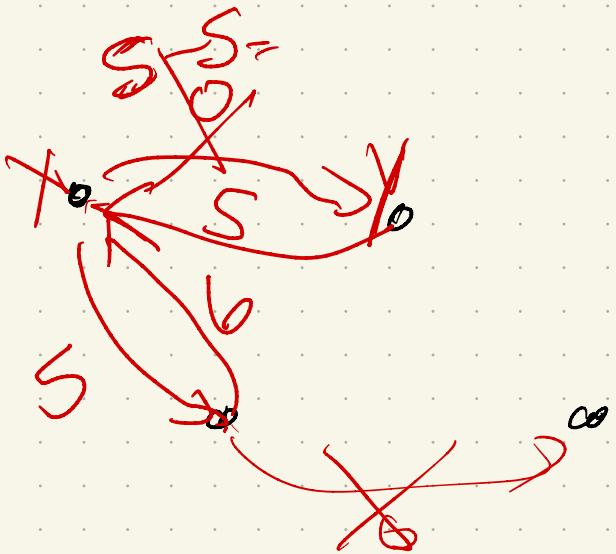
could send more

could "unsend" flow

Ex: G, f

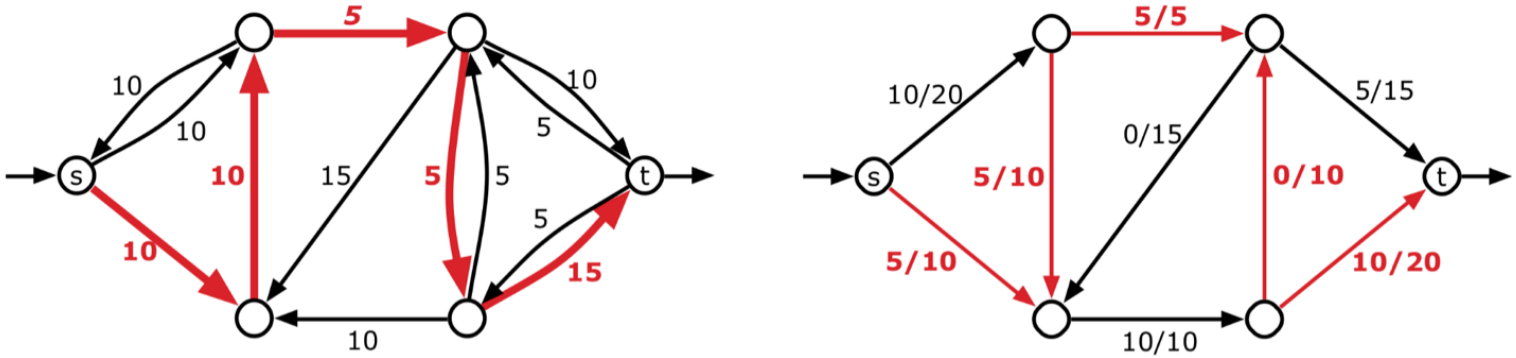


$\Rightarrow$ G_f



Augmenting a path: send more!

G_f



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

Next: Show that can get them equal.

How?

Well, take some flow. Either:

① f is maximum.

If so, find a cut of the same value.

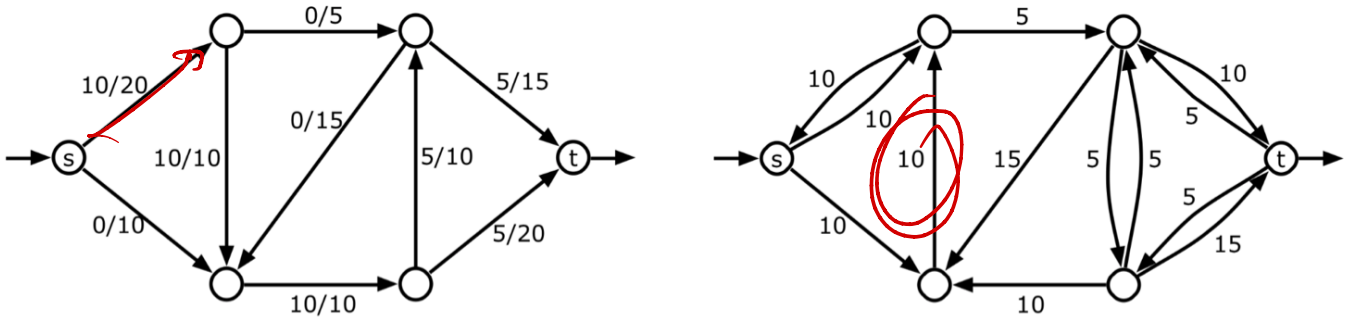
any flow ≤ any cut

② Or, it isn't!

↳ Find a bigger flow.

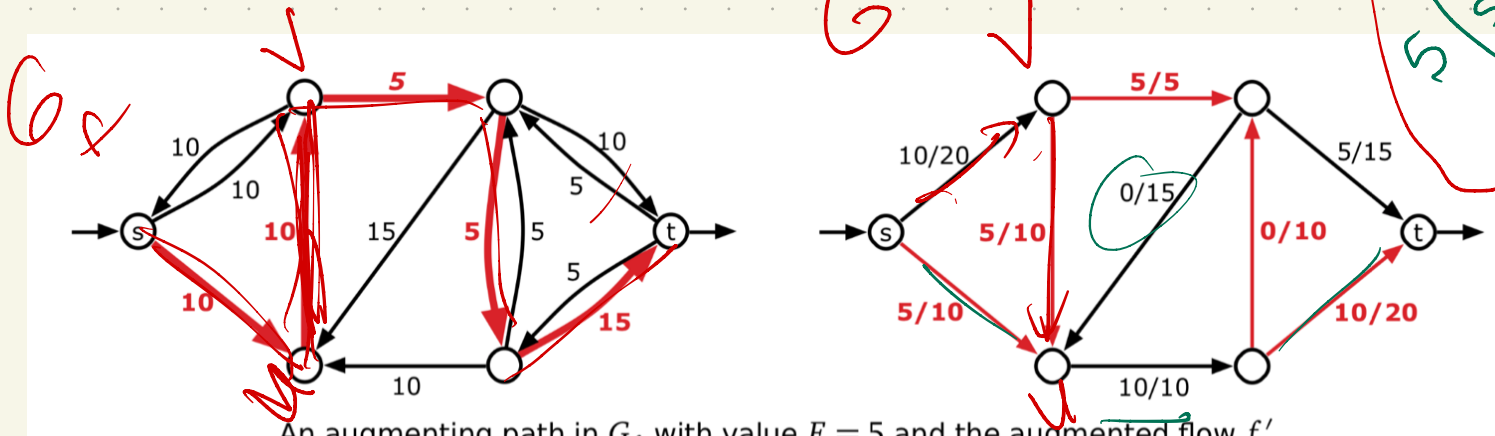
Augmenting a path:

Suppose there is a path in G_f from s to t:

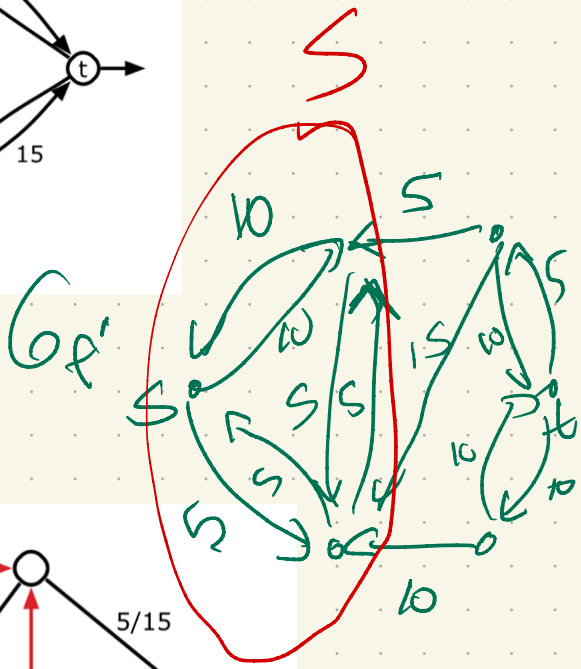


A flow f in a weighted graph G and the corresponding residual graph G_f.

Build f':

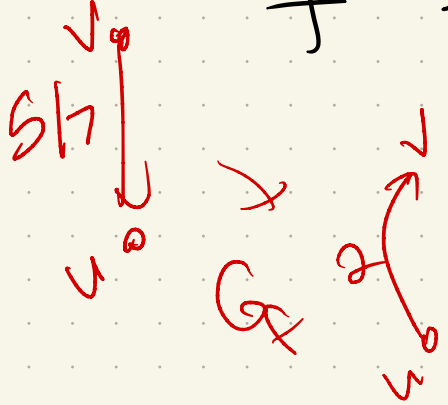


An augmenting path in G_f with value $F = 5$ and the augmented flow f'.



More formally, given path $P \in G_f$ with bottleneck $f = c$:

$$f' = \begin{cases} \text{if } \overrightarrow{uv} \notin P: & \text{unchanged: } f'(\overrightarrow{uv}) = f(\overrightarrow{uv}) \\ \text{if } \overrightarrow{uv} \in P \text{ and } \overrightarrow{uv} \in G_f: & f'(\overrightarrow{uv}) = f(\overrightarrow{uv}) + c \\ \text{if } \overrightarrow{uv} \in P \text{ and } \overrightarrow{vu} \in G: & f'(\overrightarrow{vu}) = f(\overrightarrow{vu}) - c \end{cases}$$



send more

send less "unsend"

Claim: f' is also a feasible flow!

Why? Need edge & vertex constraints:

Edge
- For any $u \to v$ not on augmenting path,

$$f'(u \to v) = f(u \to v) \leq c(u \to v)$$

- For $u \to v$ on augmenting path,

if $u \to v \in G$: took min cap edge on path

$$c_f(u \to v) = c(u \to v) - f(u \to v)$$

$$f'(u \to v) = f(u \to v) + \text{bottleneck}$$

if $v \to u \in G$:

$$c_f(u \to v) = f(v \to u) \leq \text{bottleneck}$$

$s \not\to$ $s \to$

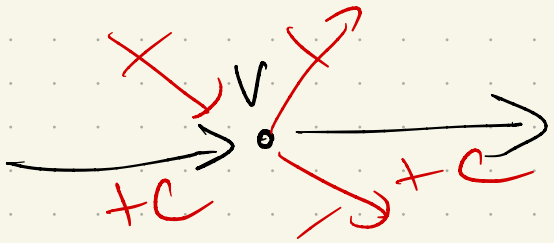
$$f'(v \to u) \geq 0$$

And vertex constraints:

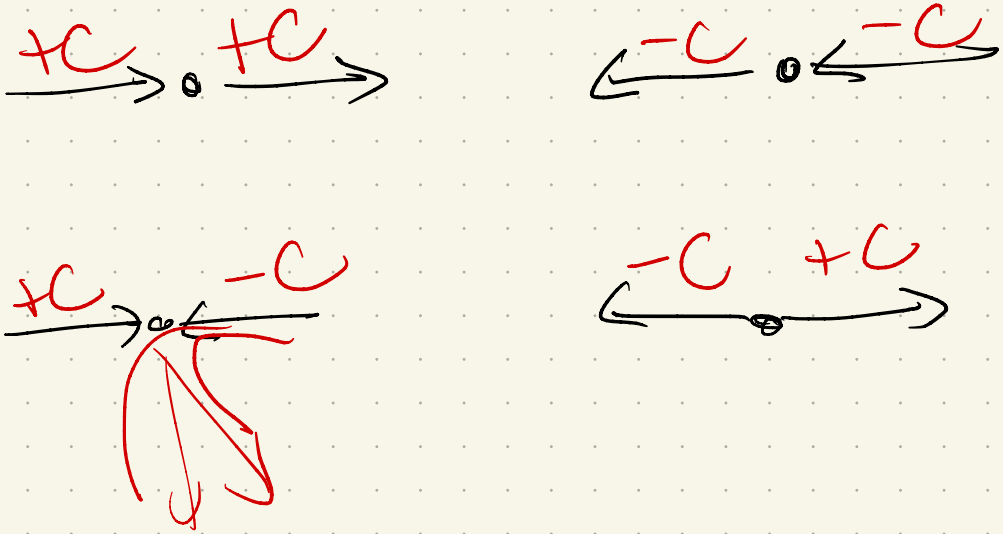
Consider v:

if not on path: ✓

if on path in G_f:



in G:



Claim: If f is a maximum flow, then G_f has no augmenting path

Proof: by contradiction

Assume f is maximum.

Build G_f + find path.

Use this path a bigger flow f'.

$\Rightarrow$ f couldn't be maximum!

□

So: f wasn't a max flow, since f' is larger.

On other hand:

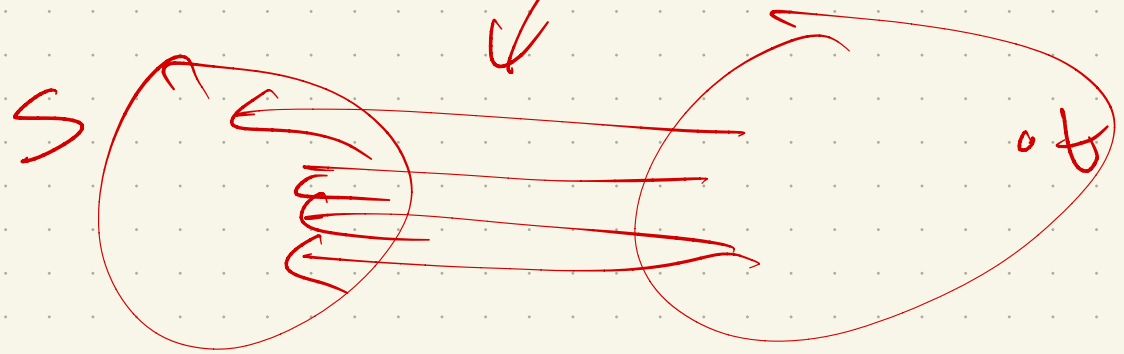
if G_f has no $s \to t$ path, find

$|S|$ = set of vertices that s can reach

Claim: $(S, V-S)$ is a cut.

(& f uses every $S \to V-S$ edge to its max capacity)

Why?



Immediate Algorithm:

Start with $f = 0$.

Build G_f

WFS(G_f, s) ← $O(V+E)$

While $t \neq s$ in same component:) ← $O(f^*(V+E))$, f^* = max flow value

find $s \to t$ path via WFS) $O(V)$

Augement along the path to get f') $O(V)$

$f \leftarrow f'$ ← $O(V)$

Build G_f ← $O(E+V)$

WFS(G_f, s) $O(V+E)$

$O(V+E)$

how many times?

Runtime?

Why all this integrality stuff?

We are assuming each path pushes at least 1 more unit of flow!

Can it be that bad?

Yes:

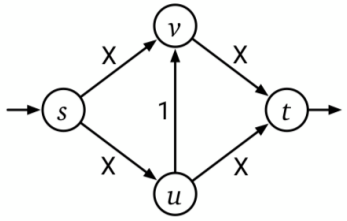


Figure 10.7. Edmonds and Karp's bad example for the Ford-Fulkerson algorithm.

How "big" is f?

(Remember, not part of input!)

Faster versions

This is an active area of research!

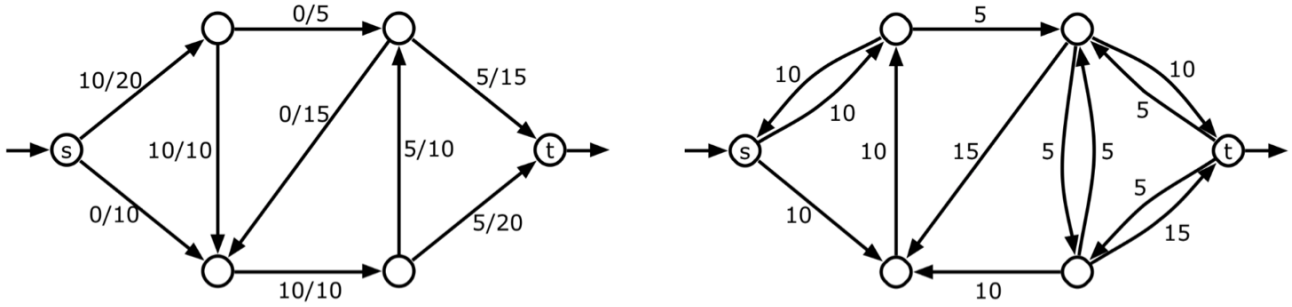
We'll see two faster examples, both (relatively) simple variations on the Ford-Fulkerson algorithm:

1. Edmonds-Karp: choose largest bottleneck edge
   ↳ $O(E^2 \log E \log |f^*|)$
2. shortest augmenting path (fewest edges)
   ↳ $O(VE^2)$

Edmonds-Karp:

Largest bottleneck: how?

Take G_f:



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

Grow a tree from s, adding largest edge out each time

↳ Similar to shortest path intuition.

Runtime:

E-K

~~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 augmenting path~~
$f' \leftarrow$ augment(f, p)
$f \leftarrow f'$
update G_f

return f

Replace:

Let p be the largest bottleneck path

# of repetitions in loop?

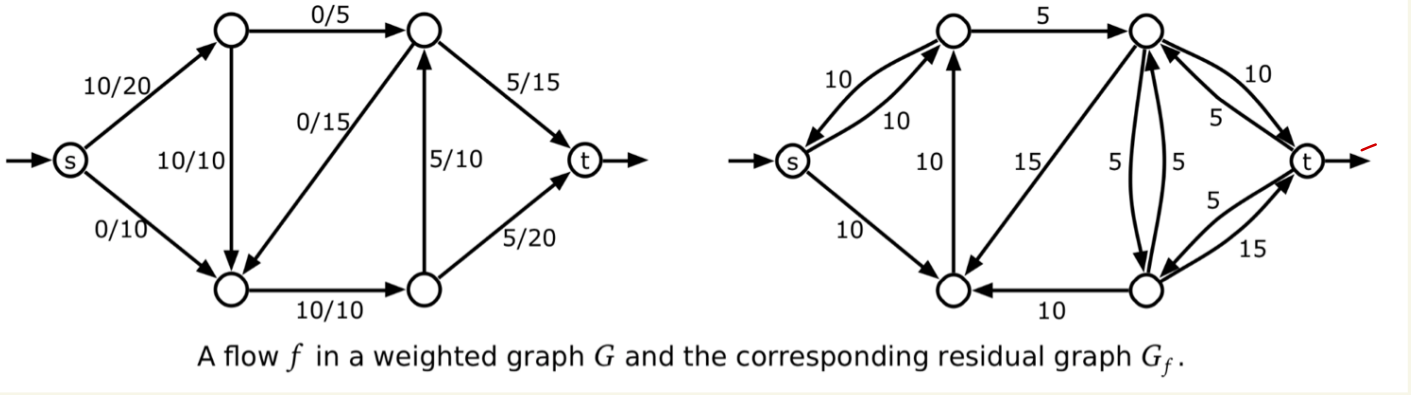
In FF, went down by ≥ 1.

Here? Hopefully better!

Key: look at current flow

loop repetitions:

Consider current flow:



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

Residual graph: Also a flow graph!

Let f' be max flow in G_f.

G_f has f' flow, & at most E paths from s to t

$\Rightarrow$ one of them is big:

$\geq$

So: adds $\geq \frac{f'}{E}$ to flow

$\Rightarrow$ next residual graph will have

$\geq (1 - \frac{1}{E}) f'$ in it.

$\hookrightarrow (1 - \frac{1}{|E|})^{\ell} \cdot f^*$ repetitions

What is ℓ?

Well, $\ell = E \ln f^*$ repetitions,

$$(1 - \tfrac{1}{|E|})^{\ell} f^* < 1$$

Wait: < 1??

E-K

~~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 augmenting path~~
$f' \leftarrow$ augment(f, p)
$f \leftarrow f'$
update G_f

return f

Replace:

Let p be the largest bottleneck path

Find bottleneck:

* # iterations:

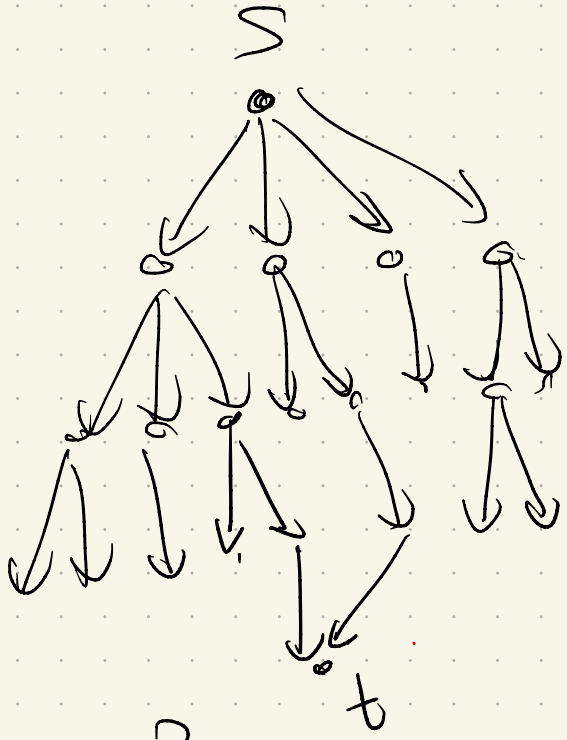
Shortest paths

```
E-K-2(G):
    Let f(e)=0 initially ∀e
    Construct Gf
    while there is s-t path in Gf
        ~~Let p be a simple s-t path~~
        f' ← augment(f, P)
        f ← f'
        update Gf
    return f
```

Let p = shortest s-t path (# edges, not capacities)

Which traversal?

Q: How many times do we need a path? (ie: how many repetitions of the while loop?)



Think of G_f, + the BFS tree rooted at s.

what changes after we push flow?

Let G_0 = initial G_f

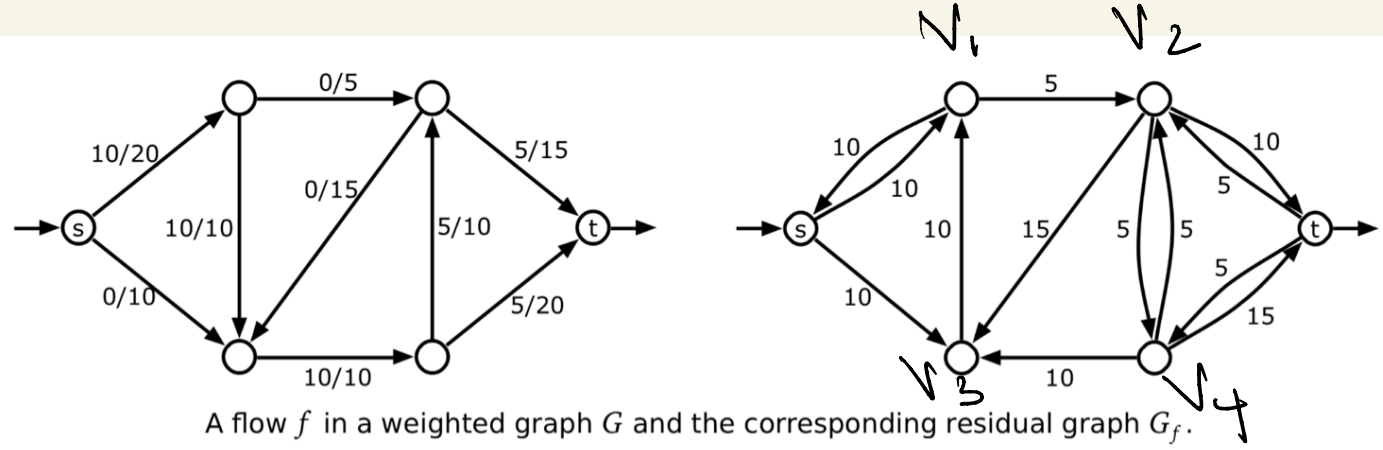
& G_i = residual graph after i repetitions of the loop.

G_i has a BFS tree T_i, so let

$level_i(v)$ = depth in tree

(Note: once s can't reach t, BFS tree: T_i : $\{s\}$

If. en $level(t) = \infty$.)



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

Claim: levels only get get bigger in each round.

proof: induction on level. (fix i)

base case: s (level 0), always stays ≥ 0

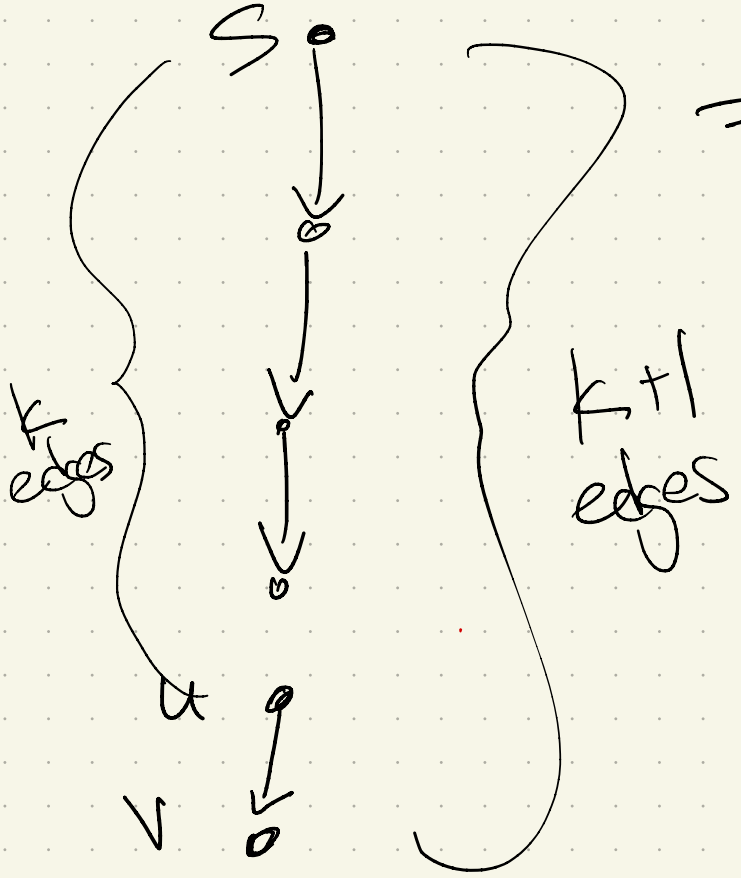
IH: consider levels $\leq k$ in G_i.

for any such u on level $\leq k$,

$$\text{level}_{i-1}(u) \leq \text{level}_i(u)$$

IS: now take v on level k+1 of G_i:

Must be a path $s \rightsquigarrow v$



$\Rightarrow$ take u just before v on path

Now: how did we get this path in G_i?

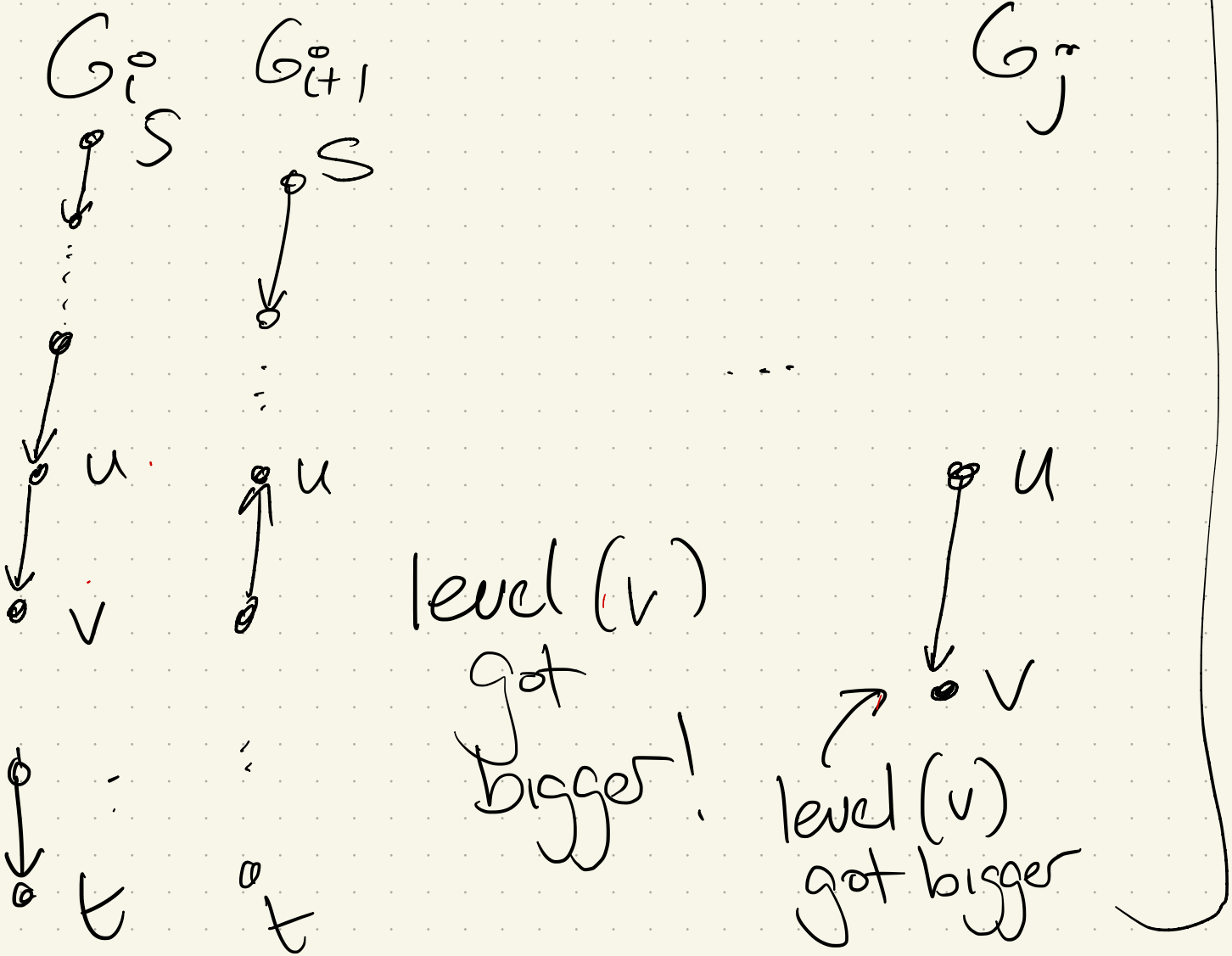
cont:

- $u \to v$ is an edge in G_{i-1}:
  $level_{i-1}(v) \leq level_{i-1}(u) + 1$

- might have come from pushing in G_{i-1}:
  here, it came from pushing a shortest path in the last round,
  so $v \to u$ was used last round.

Then: Edges can disappear & reapper.

How many times?



if level goes up to $t/2$, after $\frac{V}{2}$ rounds

$\Rightarrow$

In each iteration of the loop — some edge disappears!

$\Rightarrow \frac{V}{2} \cdot E$ repetitions

Total:

```
MAXFLOW(G):
    Let f(e)=0 initially ∀e
    Construct Gf

    while there is s-t path in Gf
        ~~let p be a simple s-t path~~
        f' ← augment(f, P)
        f ← f'
        update Gf

    return f
```

Let p = shortest s-t path (# edges, not capacities)

And... not done!

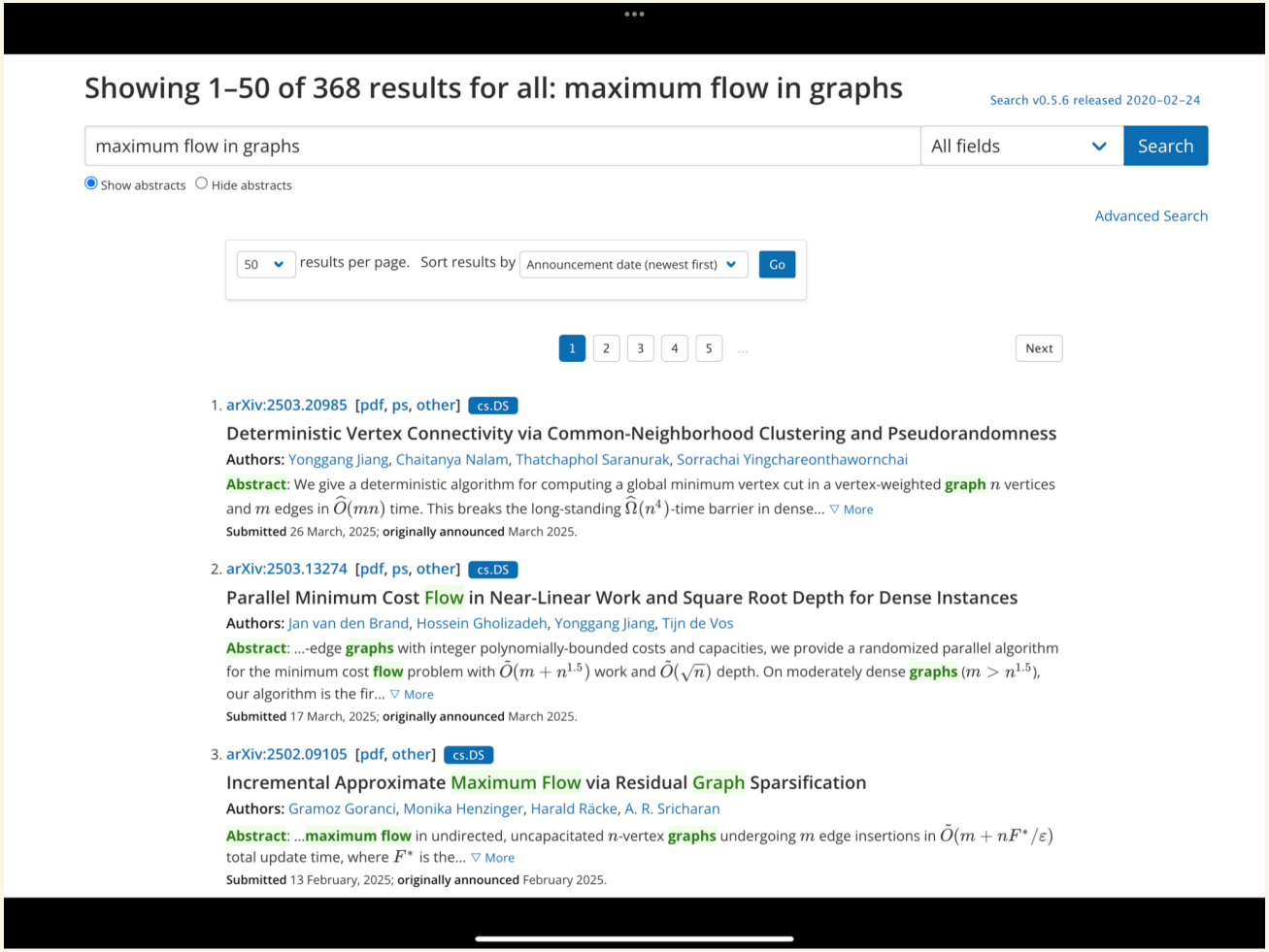
Technique	Direct	With dynamic trees	Source(s)
Blocking flow	$O(V^2E)$	$O(VE \log V)$	[Dinitz; Karzanov; Even and Itai; Sleator and Tarjan]
Network simplex	$O(V^2E)$	$O(VE \log V)$	[Dantzig; Goldfarb and Hao; Goldberg, Grigoriadis, and Tarjan]
Push-relabel (generic)	$O(V^2E)$	—	[Goldberg and Tarjan]
Push-relabel (FIFO)	$O(V^3)$	$O(VE \log(V^2/E))$	[Goldberg and Tarjan]
Push-relabel (highest label)	$O(V^2\sqrt{E})$	—	[Cheriyan and Maheshwari; Tunçel]
Push-relabel-add games	—	$O(VE \log_{E/(V \log V)} V)$	[Cheriyan and Hagerup; King, Rao, and Tarjan]
Pseudoflow	$O(V^2E)$	$O(VE \log V)$	[Hochbaum]
Pseudoflow (highest label)	$O(V^3)$	$O(VE \log(V^2/E))$	[Hochbaum and Orlin]
Incremental BFS	$O(V^2E)$	$O(VE \log(V^2/E))$	[Goldberg, Held, Kaplan, Tarjan, and Werneck]
Compact networks	—	$O(VE)$	[Orlin]

Figure 10.10. Several purely combinatorial maximum-flow algorithms and their running times.

Many use very different techniques

- linear programming
- complex data structures
- not residual graphs

Still active:



Showing 1–50 of 368 results for all: maximum flow in graphs

Search v0.5.6 released 2020-02-24

maximum flow in graphs | All fields | Search

Show abstracts | Hide abstracts

Advanced Search

50 results per page. Sort results by Announcement date (newest first) Go

1 2 3 4 5 … Next

1. arXiv:2503.20985 [pdf, ps, other] cs.DS

Deterministic Vertex Connectivity via Common-Neighborhood Clustering and Pseudorandomness

Authors: Yonggang Jiang, Chaitanya Nalam, Thatchaphol Saranurak, Sorrachai Yingchareonthawornchai

Abstract: We give a deterministic algorithm for computing a global minimum vertex cut in a vertex-weighted graph n vertices and m edges in $\widehat{O}(mn)$ time. This breaks the long-standing $\widehat{\Omega}(n^4)$-time barrier in dense... ▽ More

Submitted 26 March, 2025; originally announced March 2025.

2. arXiv:2503.13274 [pdf, ps, other] cs.DS

Parallel Minimum Cost Flow in Near-Linear Work and Square Root Depth for Dense Instances

Authors: Jan van den Brand, Hossein Gholizadeh, Yonggang Jiang, Tijn de Vos

Abstract: ...-edge graphs with integer polynomially-bounded costs and capacities, we provide a randomized parallel algorithm for the minimum cost flow problem with $\tilde{O}(m + n^{1.5})$ work and $\tilde{O}(\sqrt{n})$ depth. On moderately dense graphs ($m > n^{1.5}$), our algorithm is the fir... ▽ More

Submitted 17 March, 2025; originally announced March 2025.

3. arXiv:2502.09105 [pdf, other] cs.DS

Incremental Approximate Maximum Flow via Residual Graph Sparsification

Authors: Gramoz Goranci, Monika Henzinger, Harald Räcke, A. R. Sricharan

Abstract: ...maximum flow in undirected, uncapacitated n-vertex graphs undergoing m edge insertions in $\tilde{O}(m + nF^*/\varepsilon)$ total update time, where F^* is the... ▽ More

Submitted 13 February, 2025; originally announced February 2025.

Plus work for special classes of graphs: planar, sparse, etc.