Algorithms - Spring '25

Flows:
Ford-Fulkerson

More formally:

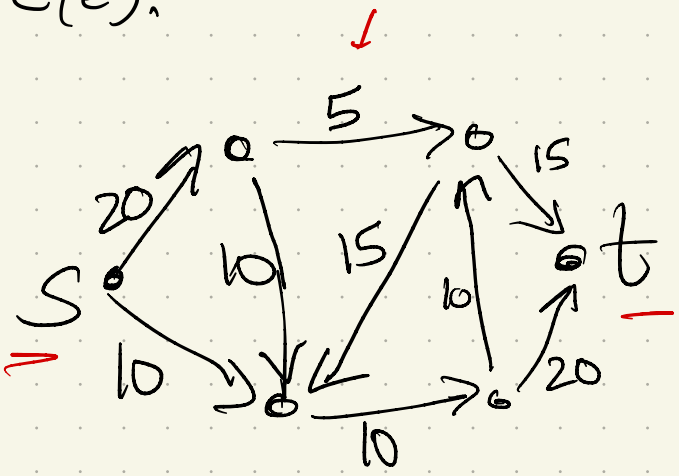
Given a directed graph with two designated vertices, s and t.

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

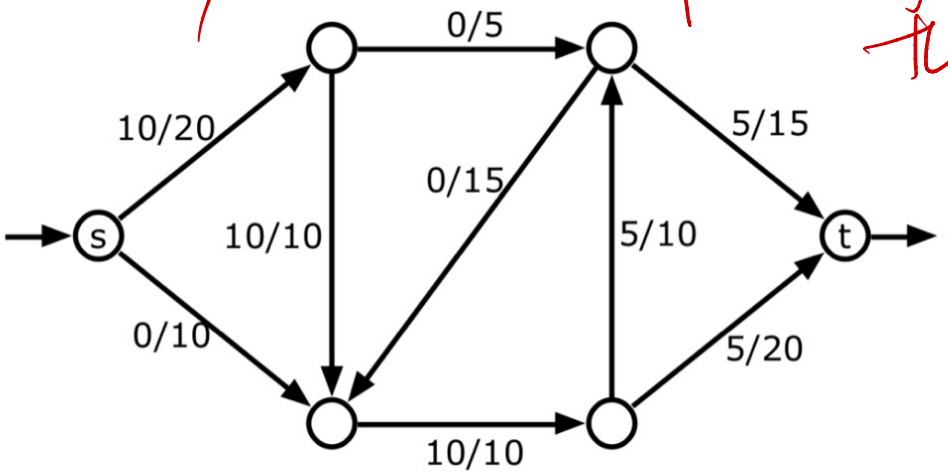
Formalizing flow:

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

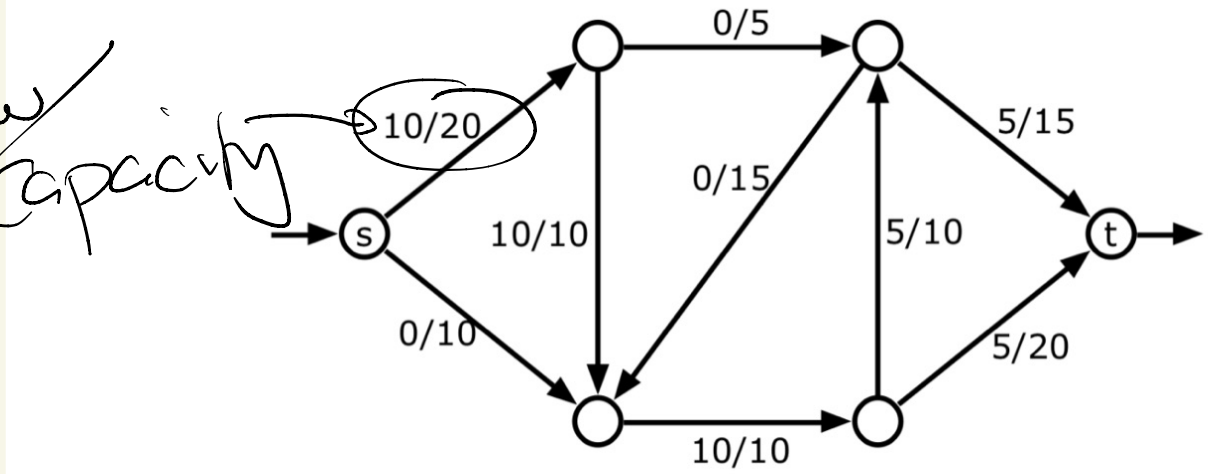
only s can ship out & no other vertex can store flow, then t can



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.

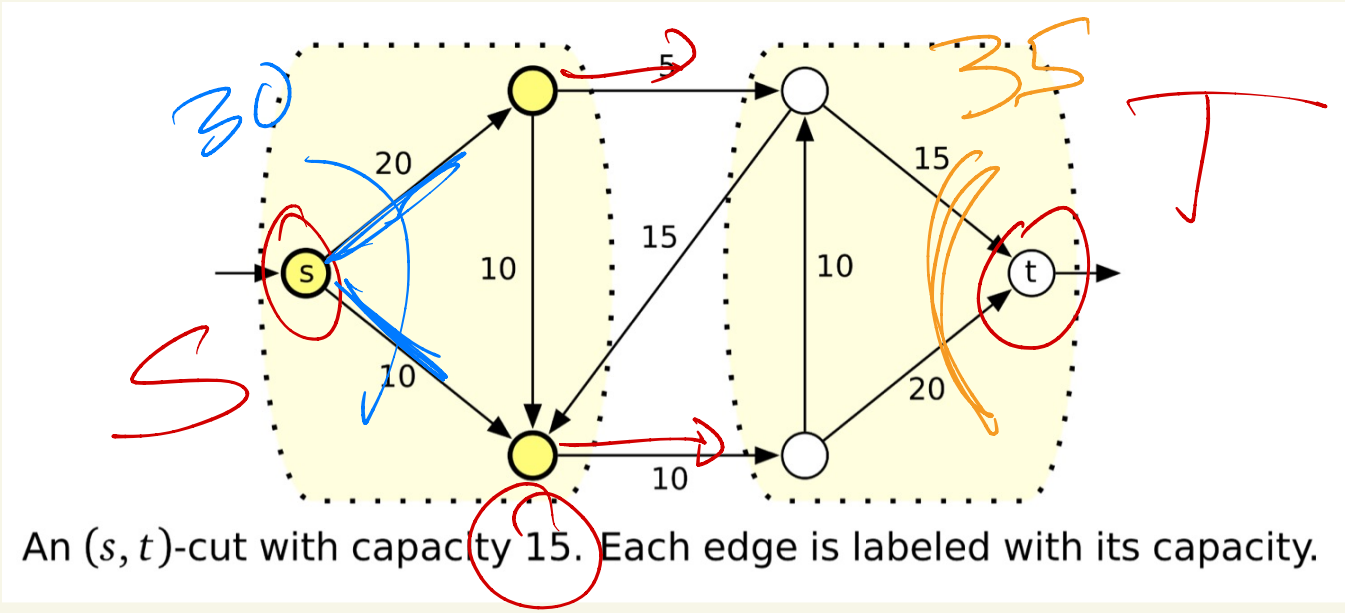
A flow is a function on edges!
(so are capacities)
Assume both are positive!
(so vertex constraints make sense)

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.

S, T is a partition of V

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

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

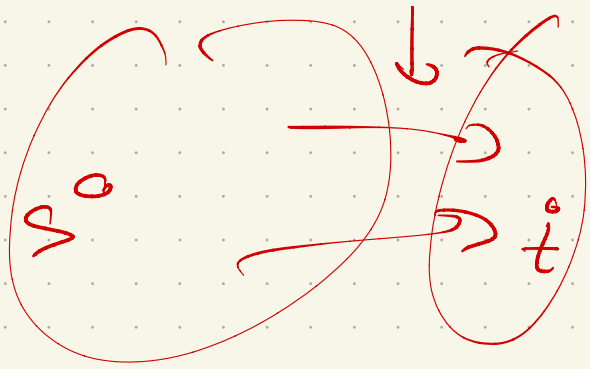
The max flow value

= min cut value

Wow!

One way is easy:

Any flow ≤ any cut.

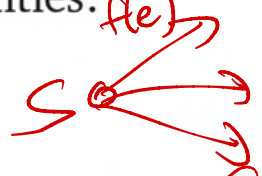


Why? Can exceed edges out of S ↓ into T

More formally:

any flow ≤ any cut

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



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

↳ slower...

More carefully:

Choose flow f + cut (S, T) $\overset{\text{O}}{=}$

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

then, since flow in = flow out for all $v \neq s, t$,

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

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

& 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!)

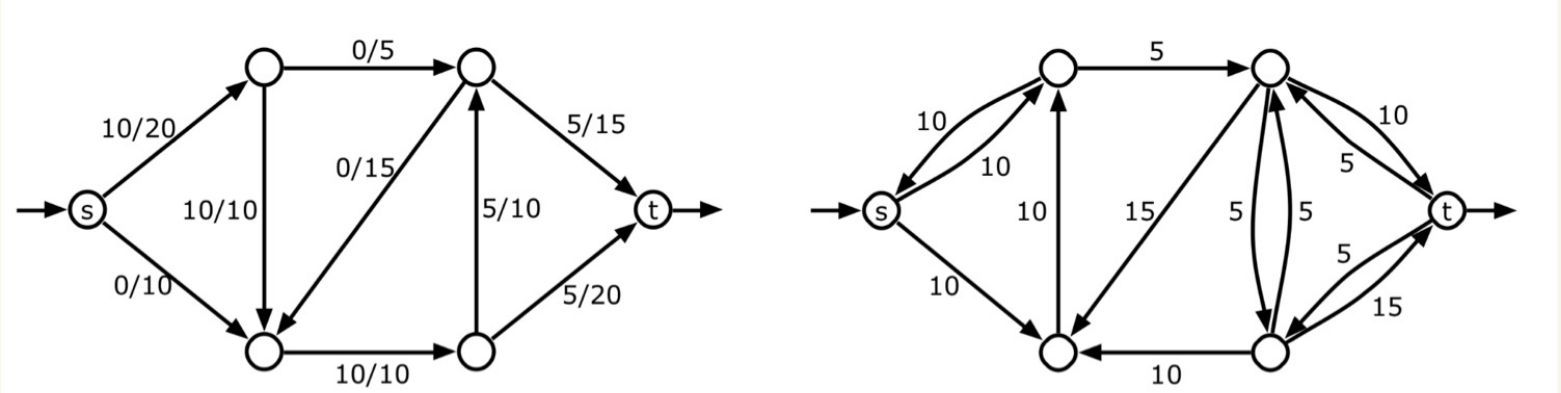
$$= \sum_{v \in S} \sum_{w \in T} f(v \to w) - \underbrace{\sum_{v \in S} \sum_{w \in T} f(w \to v)}_{\geq 0}$$

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

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

Key tool in proof:

Residual network G_f:

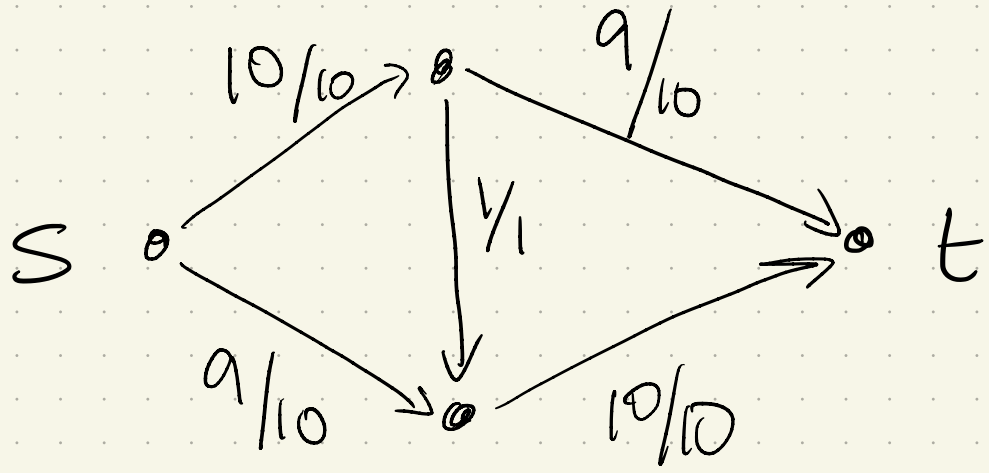


A flow f in a weighted graph G and the corresponding residual graph G_f.

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

f/c ⟹

Why can't we just be greedy & push?



Can get "stuck" if we choose wrong initially:

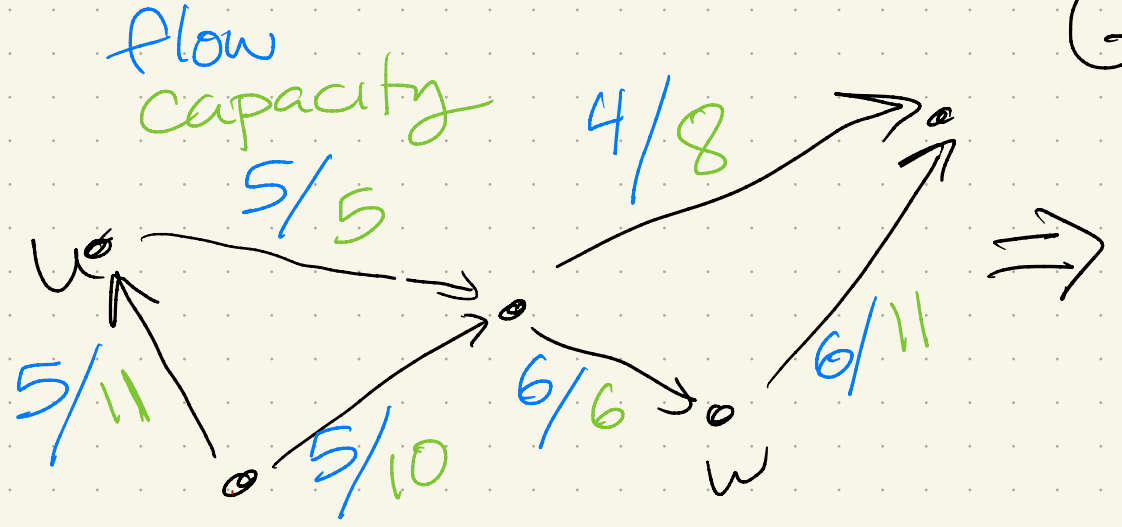
Are there any more flow paths?

More formally: Residual network G_f:

Given G & f:

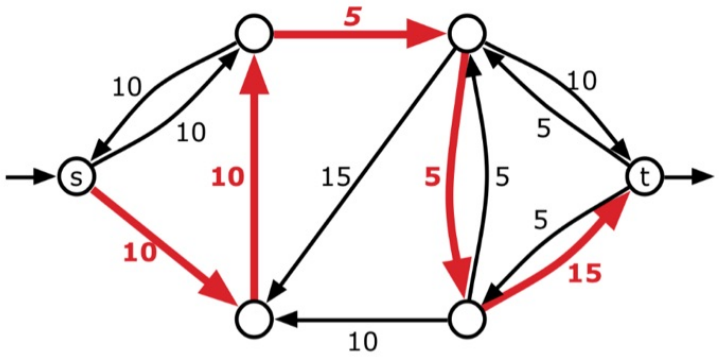
$$c_f(u \to v) = \begin{cases} c(u \to v) - f(u \to v) & \text{if } u \to v \text{ is in } E \\ f(u \to v) & \text{if } v \to u \text{ is in } E \\ \text{no edge (or 0)} & \text{otherwise} \end{cases}$$

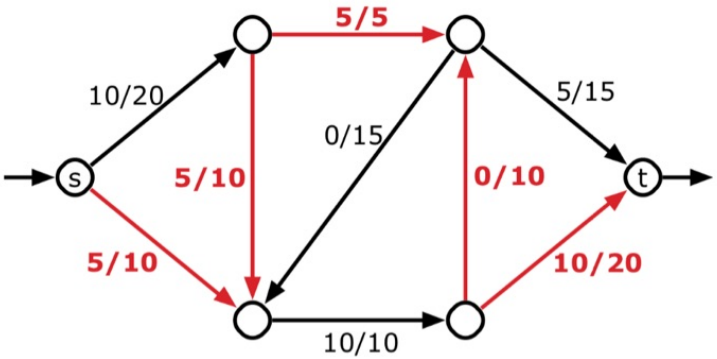
Ex: G, f



G_f

Augmenting a path: send more!





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?

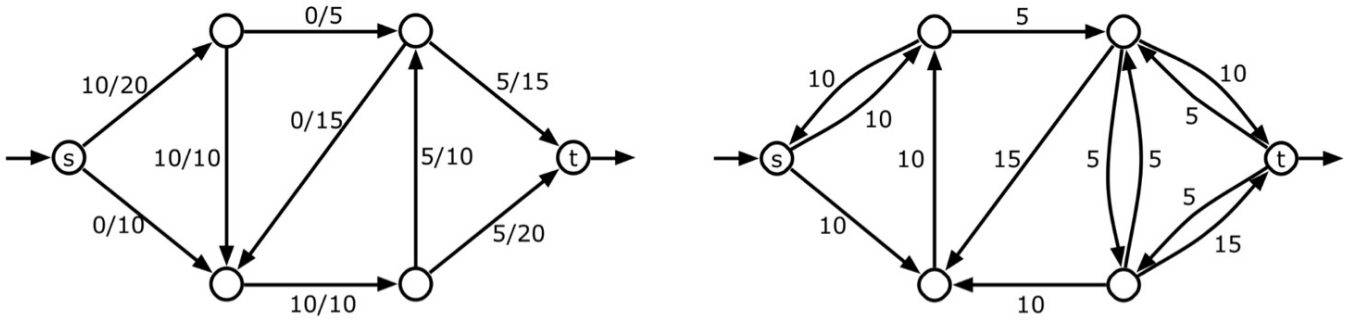
Well, take some flow. Either:

① f is maximum.
If so, find a cut of the same value.

② Or, it isn't!
↳ Find a bigger flow.

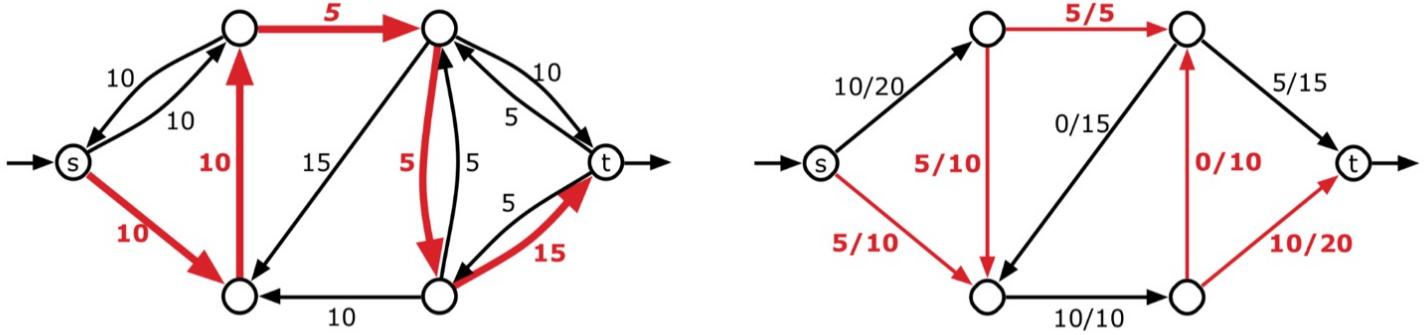
Augmenting a path:

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 $F = c$:

$$f' = \begin{cases} \text{if } \overrightarrow{uv} \notin P: \\ \text{if } \overrightarrow{uv} \in P \text{ and } \overrightarrow{uv} \in G: \\ \text{if } \overrightarrow{uv} \in P \text{ and } \overrightarrow{vu} \in G: \end{cases}$$

Claim: f' is also a feasible flow!

Why? Need edge + vertex constraints:

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

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

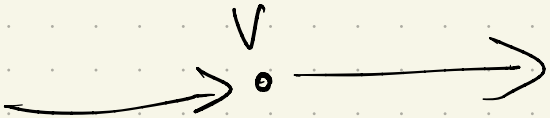
  if $u \to v \in G$:

  if $v \to u \in G$:

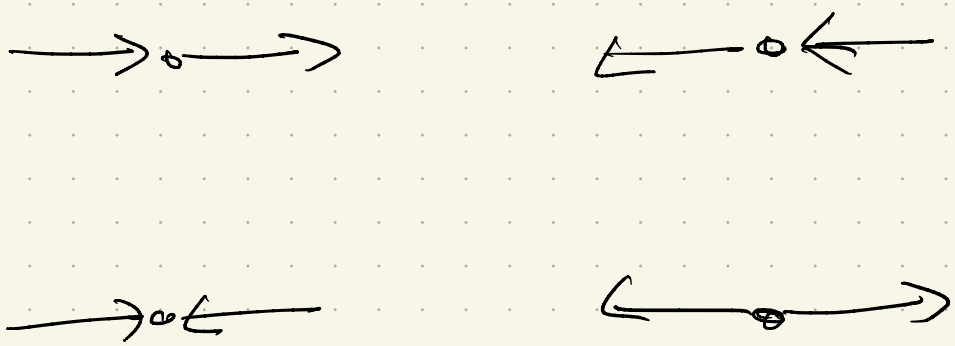
And vertex constraints:

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'.

$\Rightarrow$

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)

While t & s in same component:

find $s \to t$ path via WFS

Augement along the path to get f'

$f \leftarrow f'$

Build G_f

WFS(G_f, s)

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:

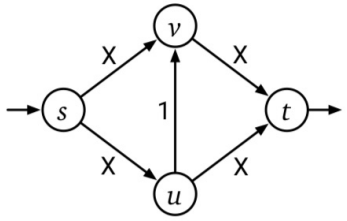


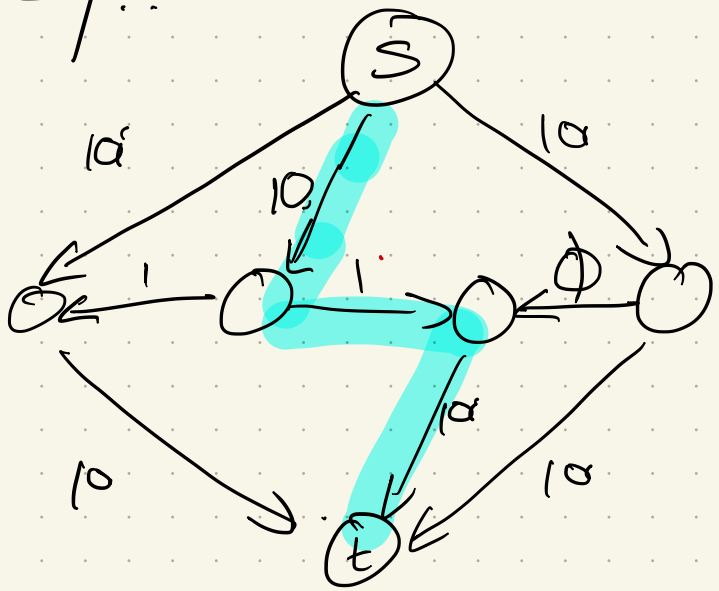
Figure 10.7. Edmonds and Karp's bad example for the Ford-Fulkerson algorithm.

How "big" is f?

(Remember, not part of input!)

What if it's not integers?

Messy!!



The key:

$\phi = \frac{1+\sqrt{5}}{2}$

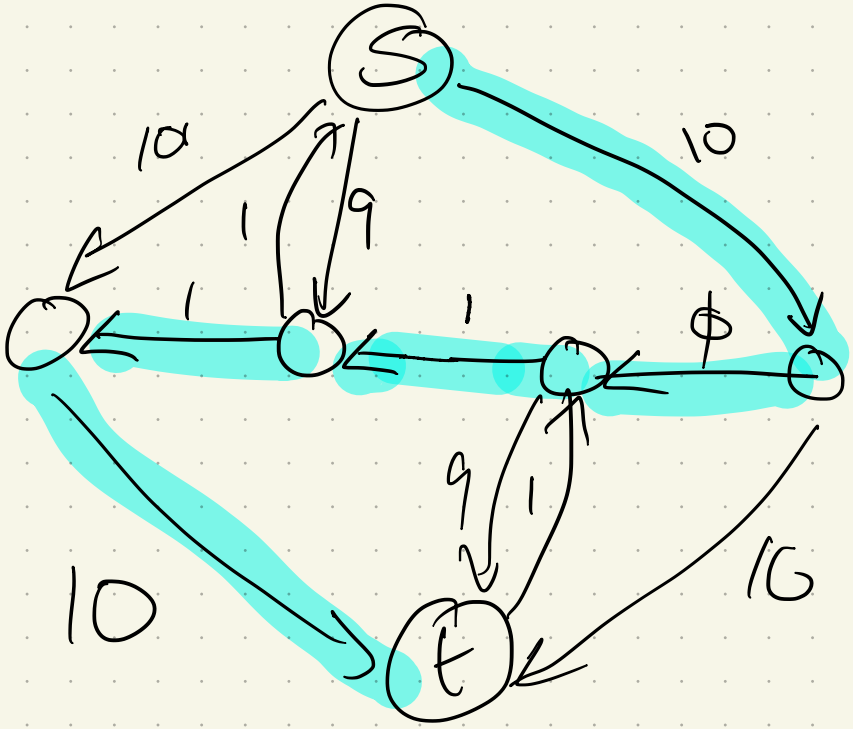
WHY??

Simple:

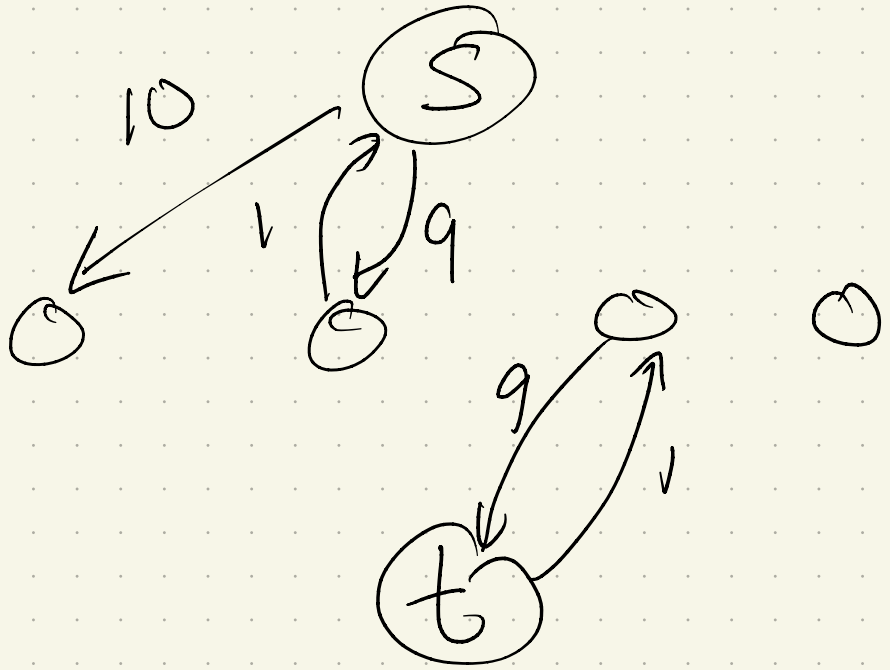
$1 - \phi = \phi^2$

G_f:

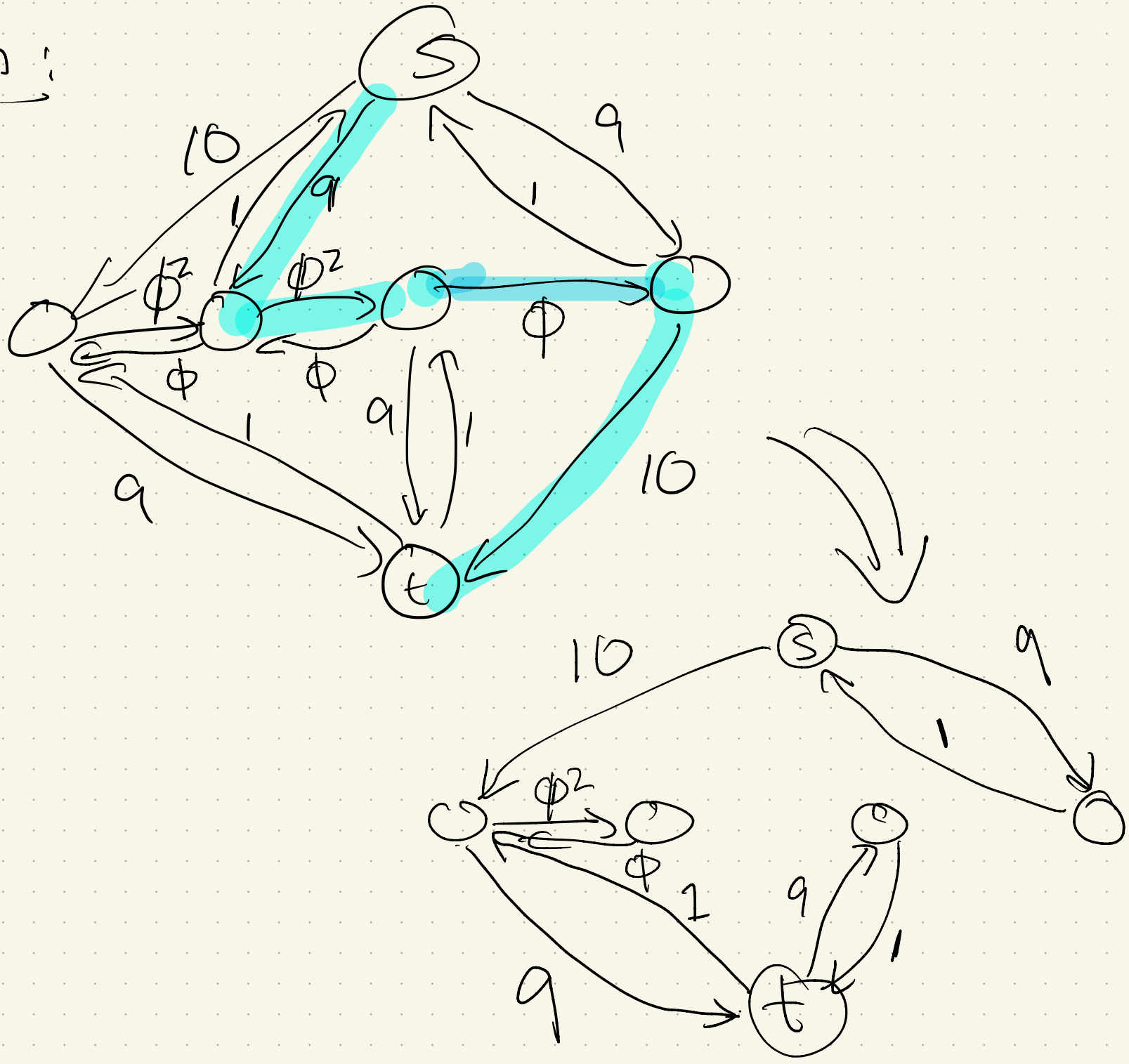
Next path:



New G_f:



Then:



Continue to push:

Ends with:

$\phi, 0,$ and $1-\phi = \phi^2$

Repeat:

- $\phi^2, 0, \phi^3$

then

.

.

etc :

But max flow = 21

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
   ↳ $O(VE^2)$

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: