Algorithms - Spring 2025

Flows + Cuts

Recap

- No class next Monday, and office hours will be Wed: 1-3
- HW - due tonight
- Next HW: flows

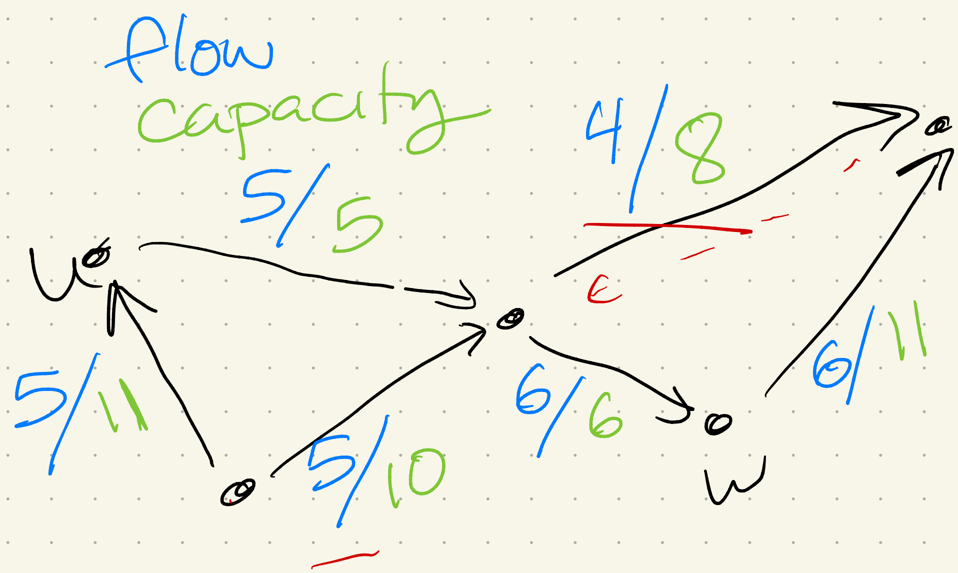
More formally: Residual network G_f:

Given G & f:

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

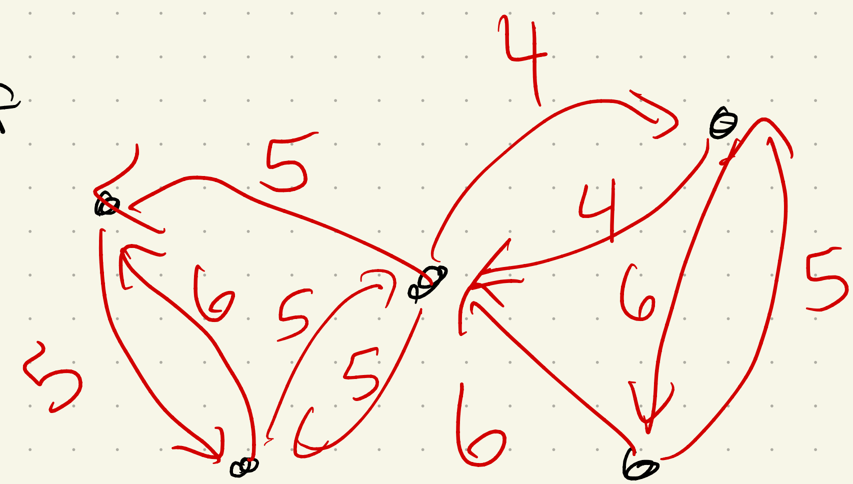
if reverse edge

Ex: G, f



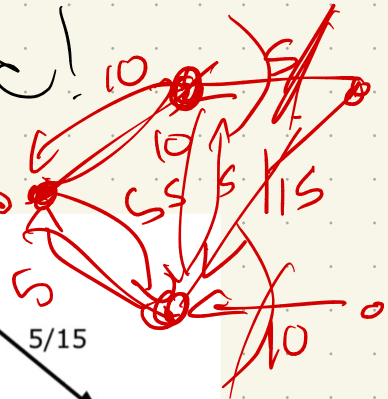
$\Rightarrow$

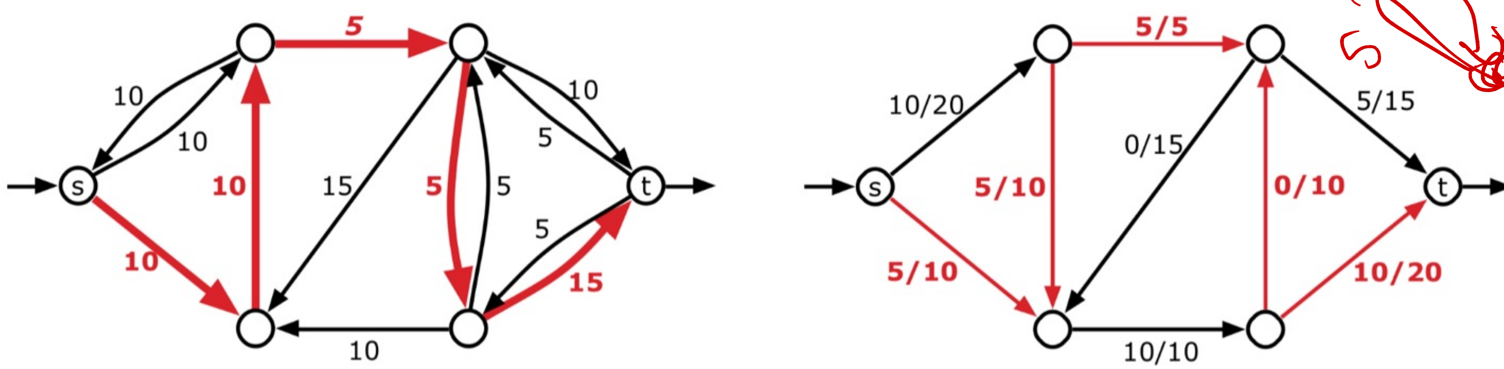
G_f



Augmenting a path: send more!

BFS→



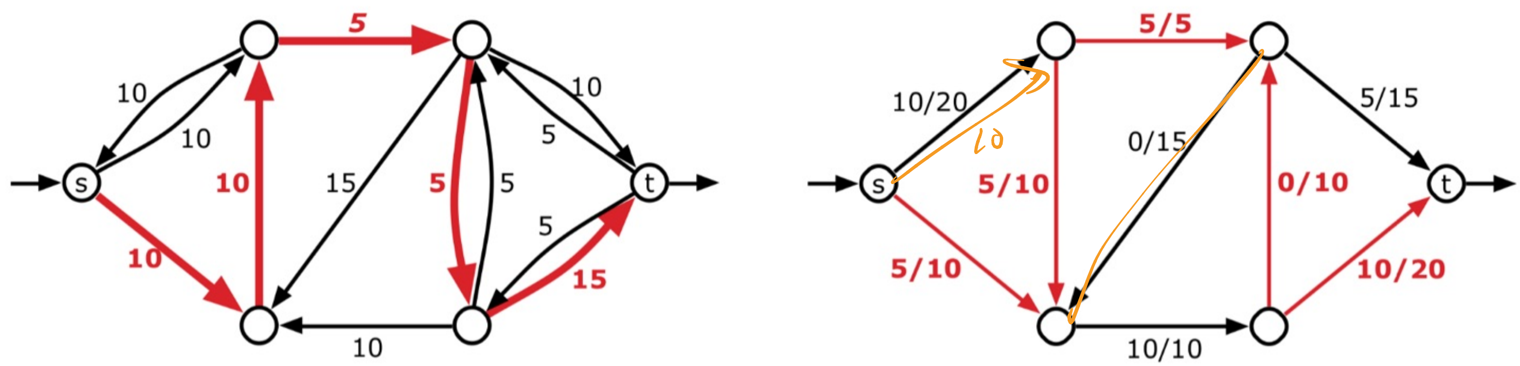


An augmenting path in G_f with value $F = 5$ and the augmented flow f'.

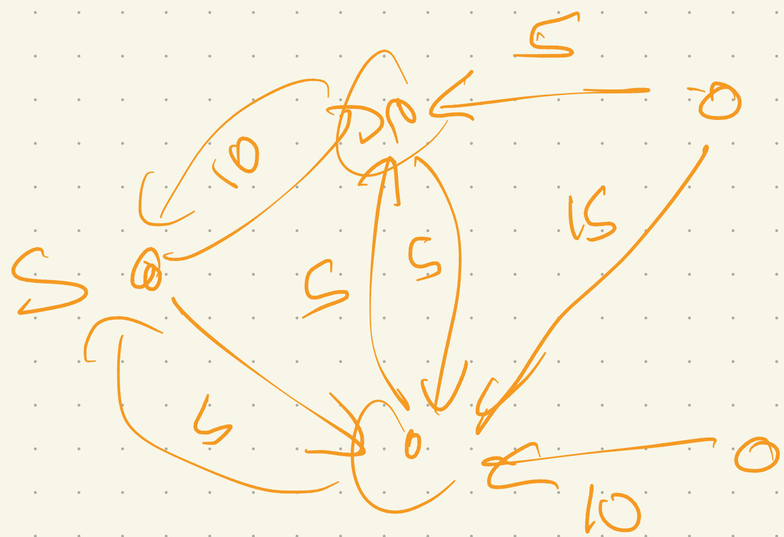
This is just an s-t path in G_f.

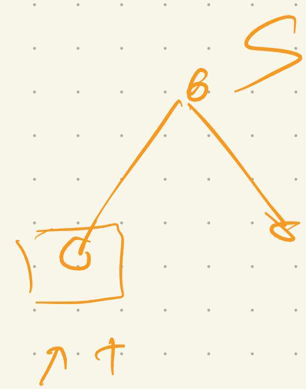
Then, find min capacity edge on that path

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



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





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'!

+ bottle neck edge

$\Rightarrow$ then f was not maximum

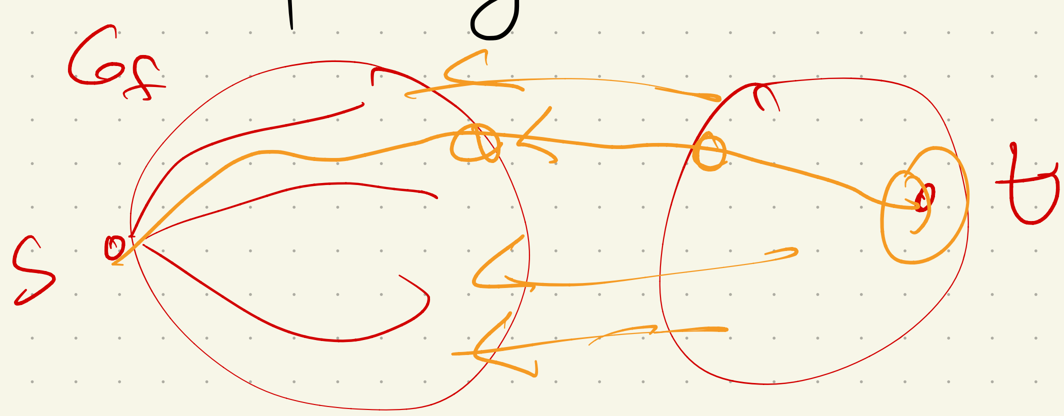
<u>So:</u> f wasn't a max flow, since f' is larger.

On other hand:

if G_f has no $s \to t$ path, find $|S|$ = set of vertices that s can reach

<u>Claim:</u> $(S, V-S)$ is a cut. (& f uses every $S \to V-S$ edge to its max capacity)

Why?



Immediate Algorithm: $(V+E) \cdot f^*$ ← max flow

Start with $f = 0$.

Build G_f ← $V+E \geq O(E)$

WFS(G_f, s) ← are s & t connected

While t & s in same component: ← every time, flow increases ← $V+E$

find $s \to t$ path via WFS

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

$f \leftarrow f'$ ←

Build G_f ← $V+E$

WFS(G_f, s) ← $V+E$

$\boxed{\text{Runtime?}}$

loop repeats $\leq f^*$ times

Why all this integrality stuff?

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

Can it be that bad?

Yes:

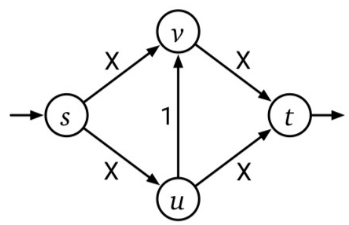


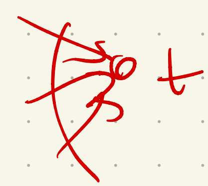
Figure 10.7. Edmonds and Karp's bad example for the Ford-Fulkerson algorithm.

+1 each round if bottleneck is that cap = 1 edge

How "big" is f^*?

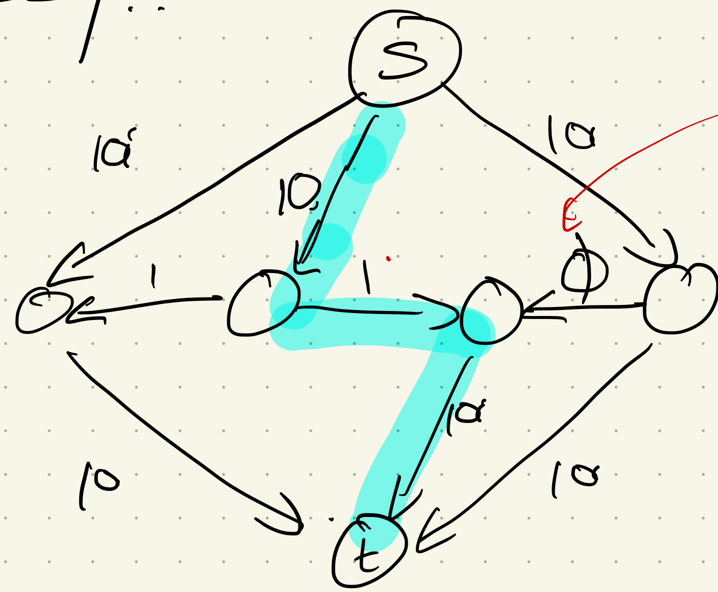
(Remember, not part of input!)

$$f^* \le \sum_{\substack{\text{all edges} \\ u\to v}} c(u\to v)$$



What if it's <u>not</u> integers?

Messy!!



The key:

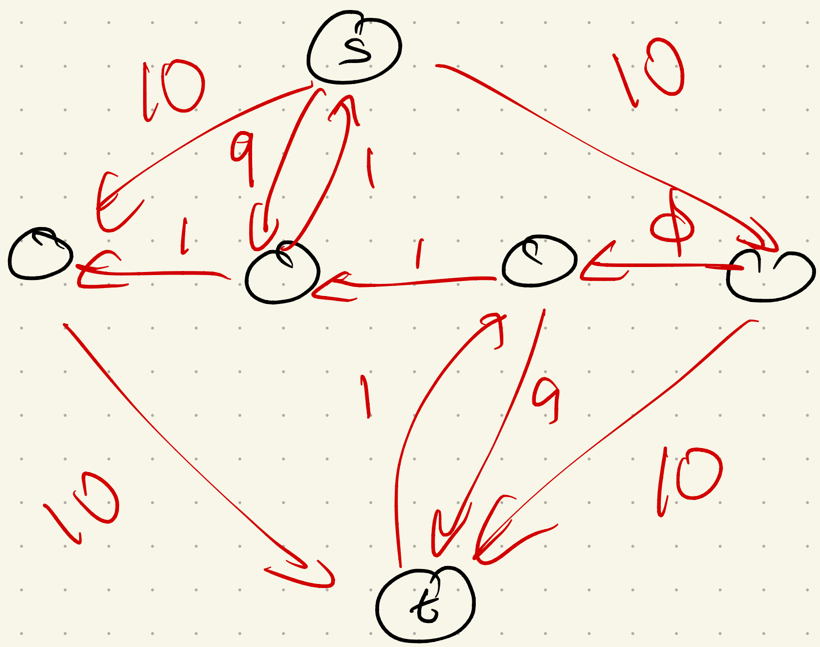
$\phi = \frac{1 \cancel{+} \sqrt{5}}{2}$ <1

WHY??

Simple:

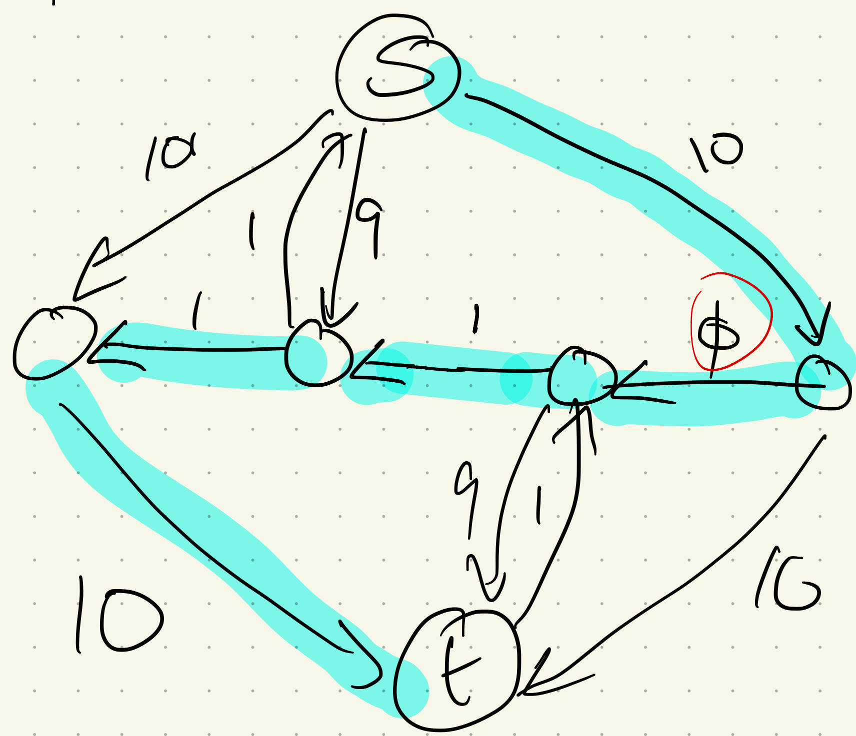
$\boxed{1 - \phi = \phi^2}$

G_f:

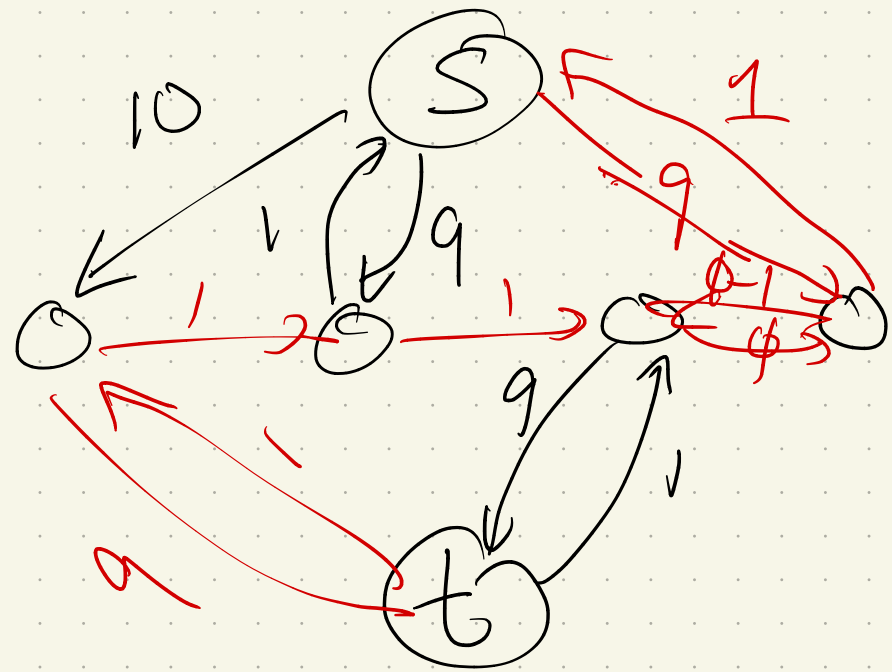


Flow of val = 1

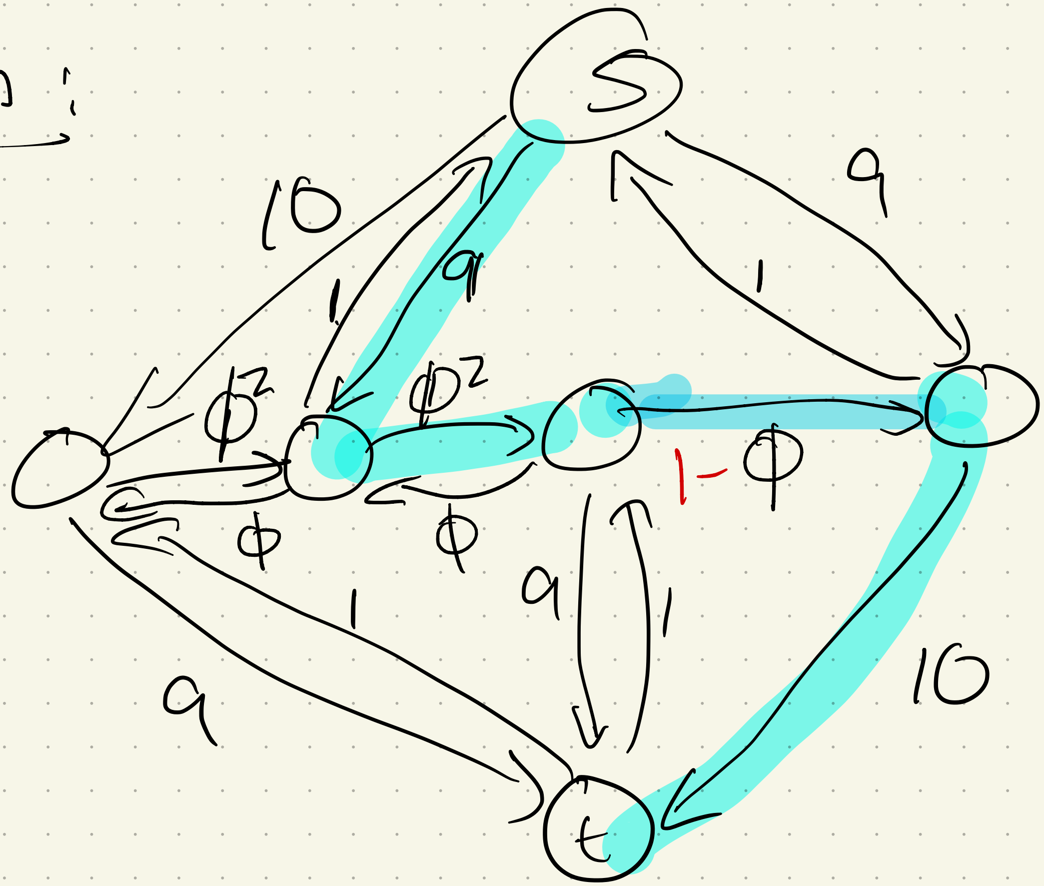
Next path:

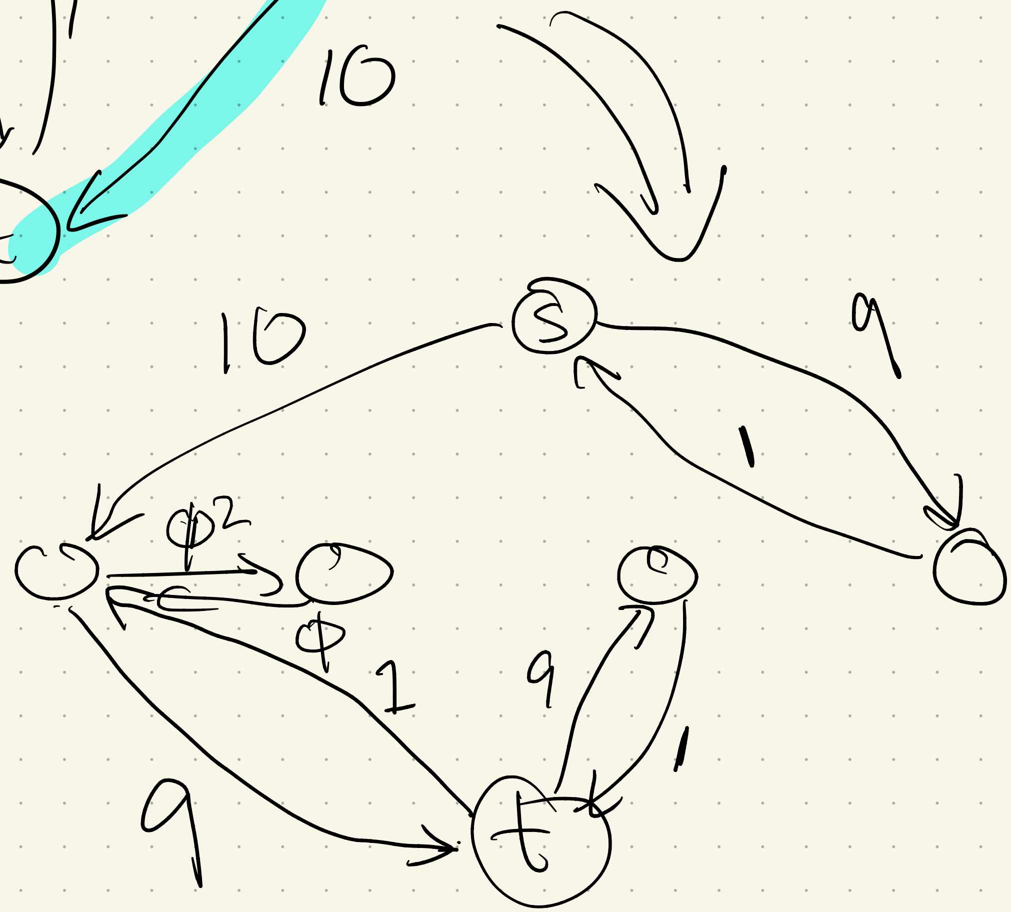


New G_f:



Then:





Continue to push:

assume integers!

Ends with:

ϕ, 0, and $\boxed{1-\phi = \phi^2}$

Repeat:

- $\phi^2, 0, \phi^3$ $\quad (1-\phi)\phi$

then

-

etc ⋮

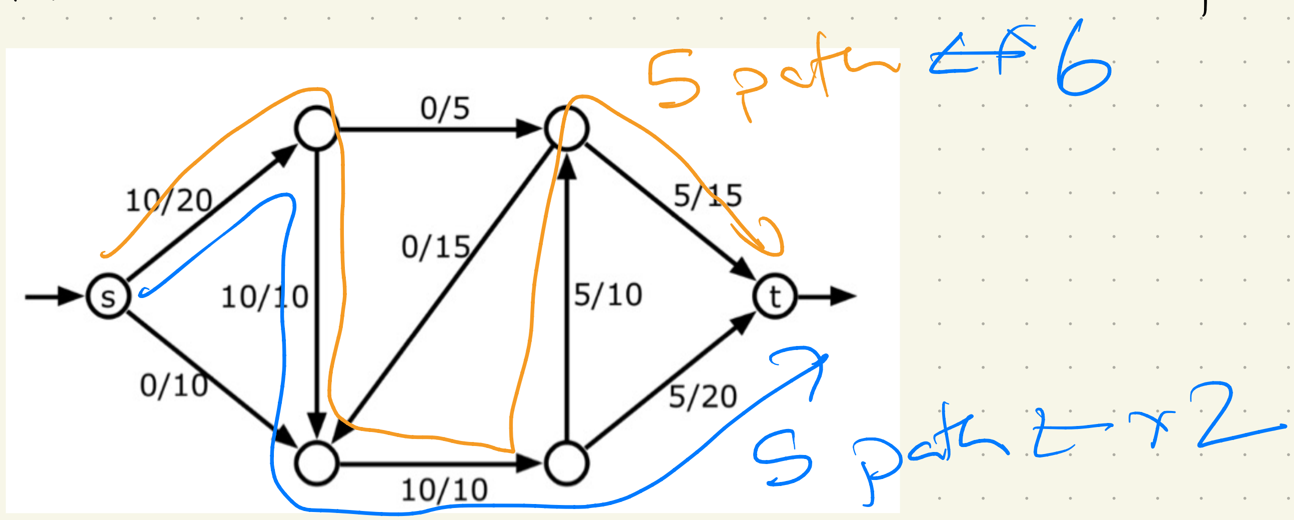
ϕ^k

But: max flow = <u>21</u>

General framework: Decomposing flows

Given a valid flow, we have an intuition that it "follows" paths:

But: it's a function! (no paths)



That said, this intuition does hold in some sense:

Formalizing:

Linear combinations of flows result in flows!

(ignoring capacities...)

$$h(u\to v) = \alpha\cdot f(u\to v) + \beta g(u\to v)$$

$f(u\to v) = 5$

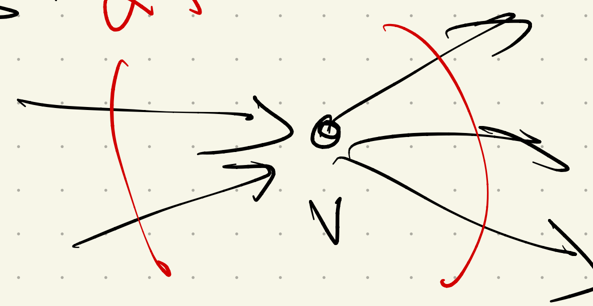
$u \longrightarrow v$

$g(u\to v) = 1$

$h = 2\cdot 5 + 3\cdot 1$
$= 13$

vertex constraints: $\alpha\cdot f + \beta\cdot g$

$\alpha f + \beta f$

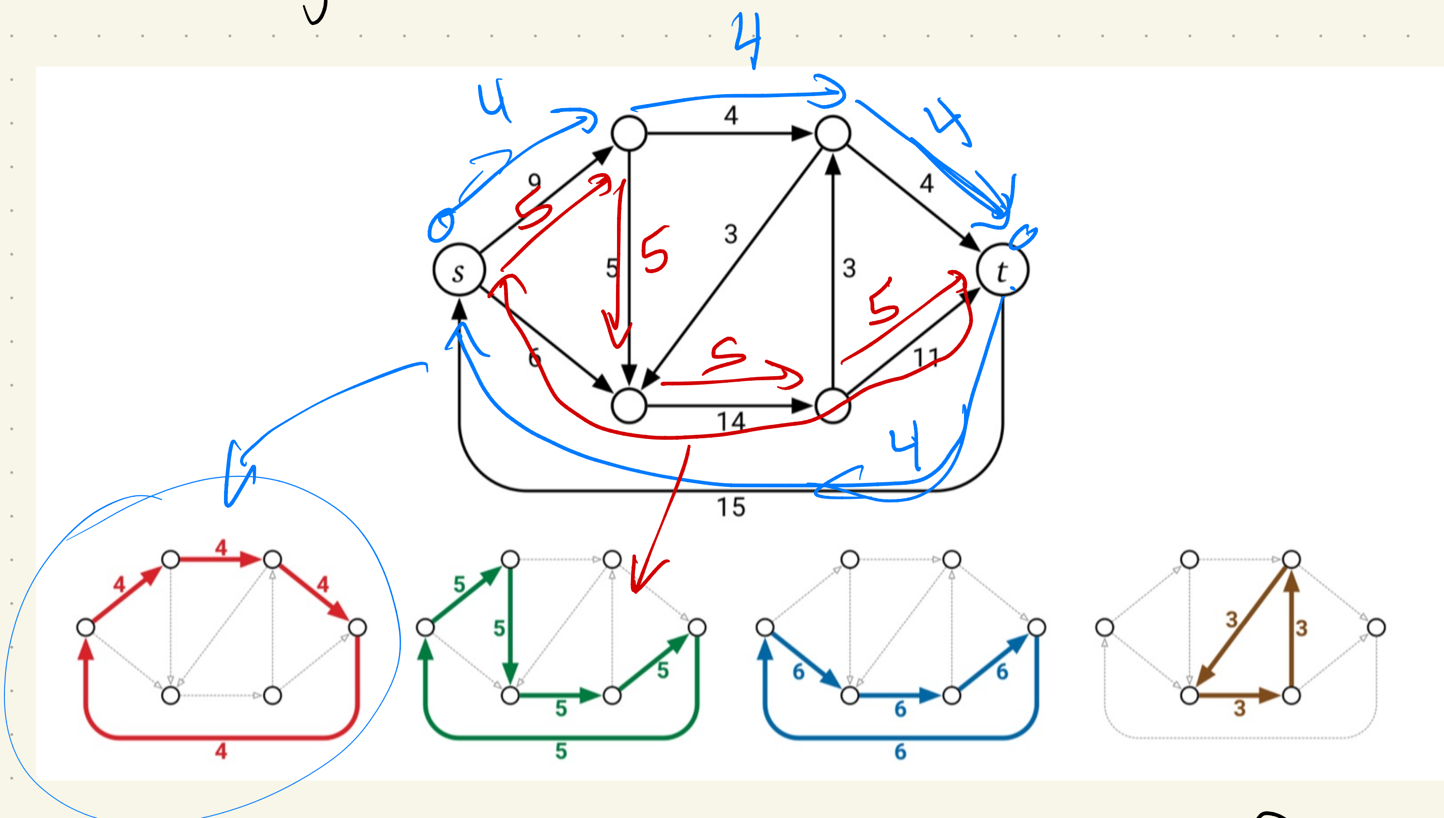


all OK

Flow decomposition: flow in = flow out

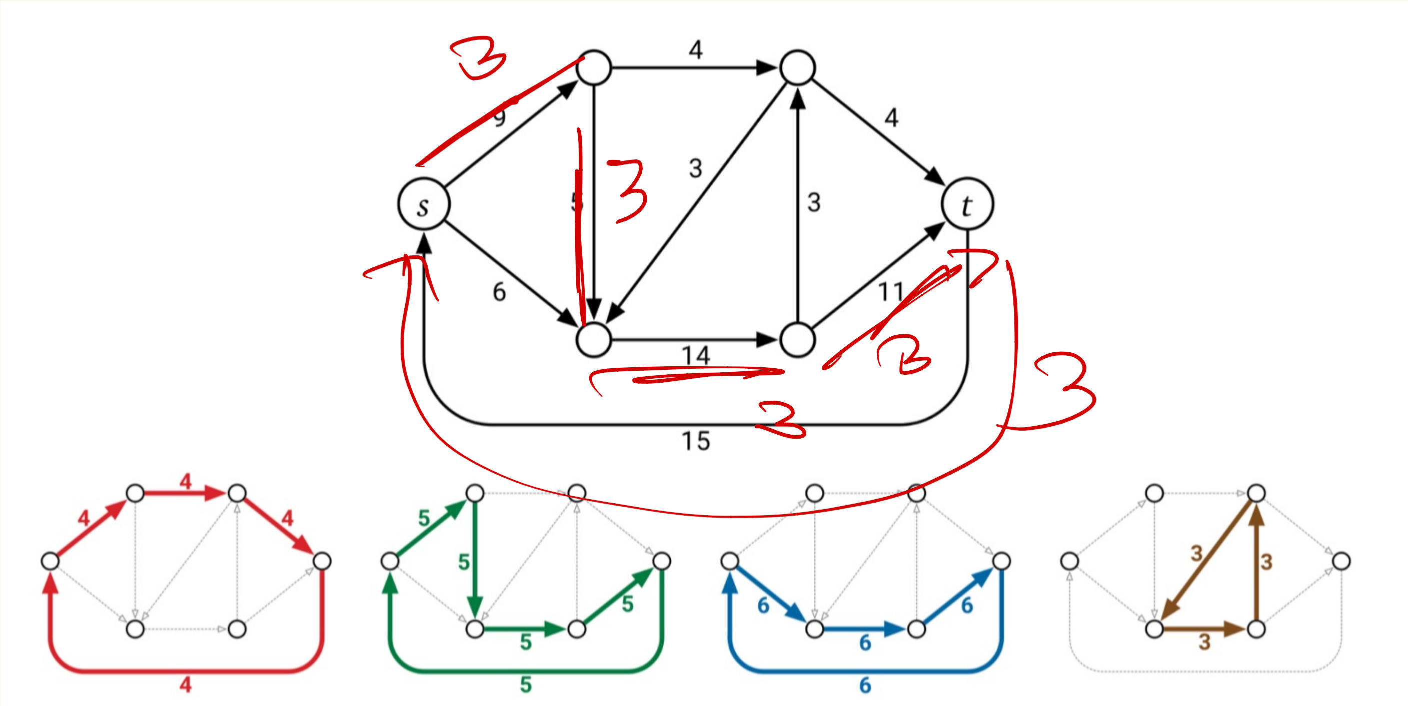
Add an edge $t \to s$, + con

decompose a flow into cycles:



s.t. "sum" of cycles = flow.

Not unique:



Why care?

Suggests general strategy in FF of "find a path & push" will always get you to best flow (if integer capacity).

Let's see how it works...

Faster versions

This is an active area of research!

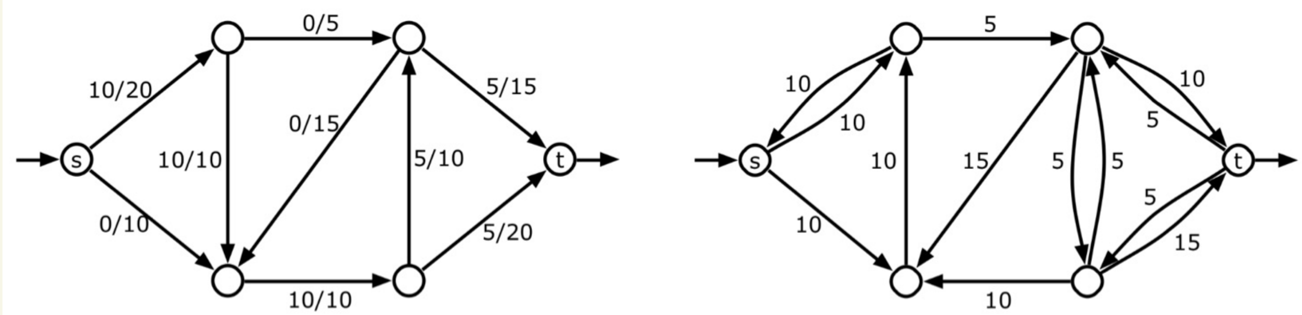
We'll see two faster examples, both (relatively) simple variations on the Ford-Fulkerson algorithm:

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 HW!

Runtime:

variant of MST: $E \log V$

E-K

~~MAXFLOW~~(G):

Let $f(e)=0$ initially $\forall e$
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

$V+E$
$=O(E)$

return f

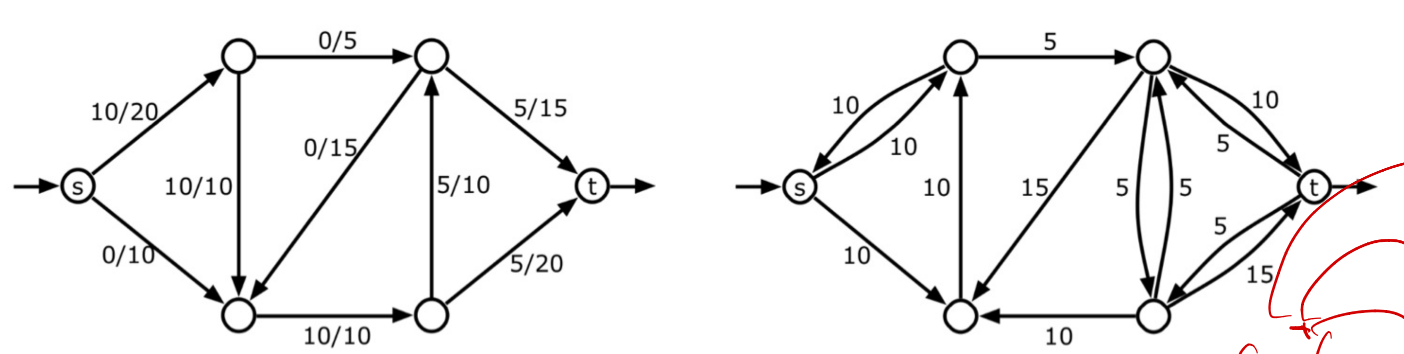
Replace: $E \log V$

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.

total is f'

Residual graph: Also a flow graph!

Let f' be max flow in G_f.

$\leq E$

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

$\Rightarrow$ One of them is big:

$\geq f'/E$

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

$f' - \frac{f'}{E}$ next time

$\Rightarrow$ next residual graph will have $\geq \left(1-\frac{1}{E}\right)f'$ in it.

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

What is ℓ?

Well, $\boxed{\ell = E \cdot \ln f^*}$ repetitions,

$\left(1-\frac{1}{|E|}\right)^{\ell} f^* < 1$

Wait: <1??

must hit 0, since integer valued

$$f' \longrightarrow \left(1-\frac{1}{E}\right)f'$$

$$\downarrow$$

$$\left(1-\frac{1}{E}\right)^2 f'$$

e

$$\downarrow$$

$$\left(1-\frac{1}{E}\right)^3 f'$$

$\vdots$ smaller

E-K

~~MAXFLOW~~(G):

Let $f(e)=0$ initially $\forall e$
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

Inside of loop: find bottleneck: $E \log V + (V+E)$

$\times$ # iterations: $E \ln f$

$\Rightarrow O(E^2 \log V \ln f)$

Shortest paths ← min # of edges

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

~~MAX FLOW~~ (G):

Let f(e)=0 initially ∀e
Construct G_f

while there is s-t path in G_f
~~Let p be a simple s-t path~~
V+E
f' ← augment (f, P)
f ← f'
update G_f

return f

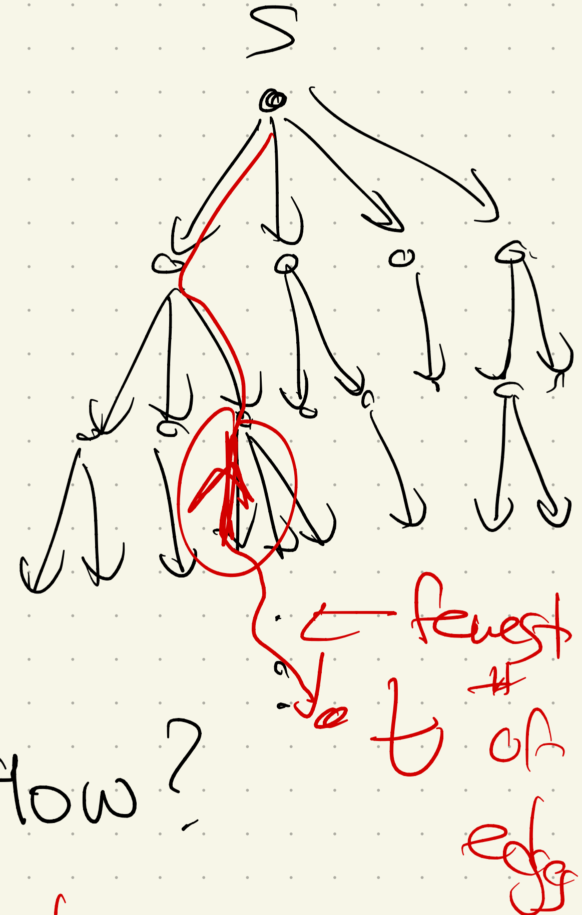
V+E

Which traversal? $\boxed{\text{BFS}}$

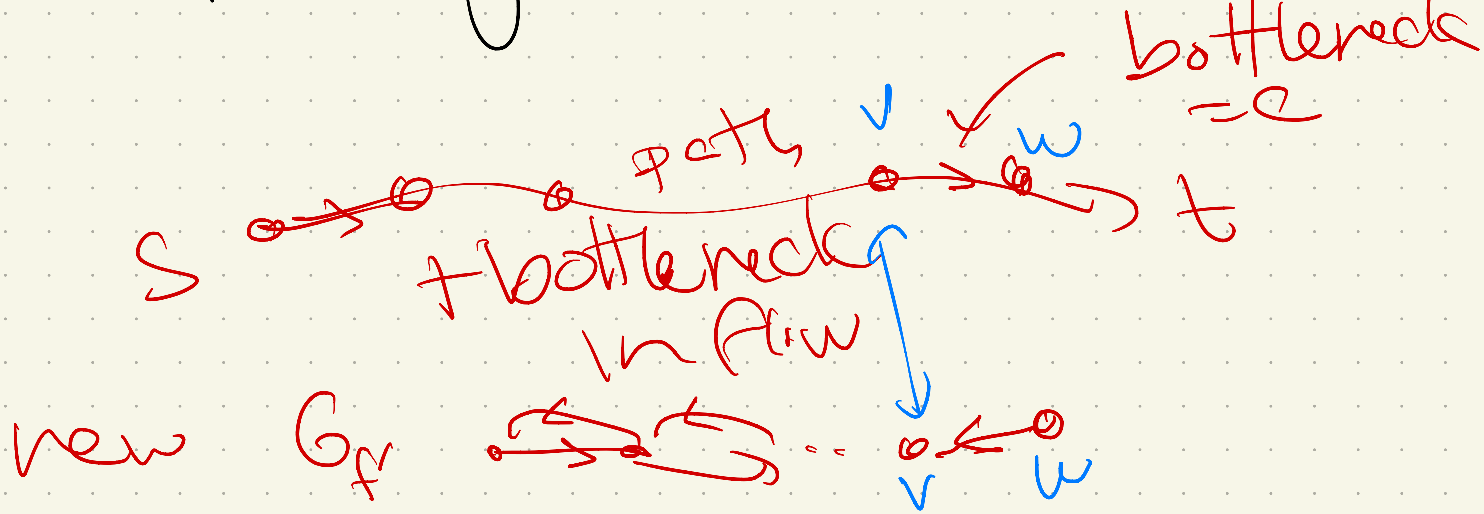
So inside of loop = O(V+E)

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

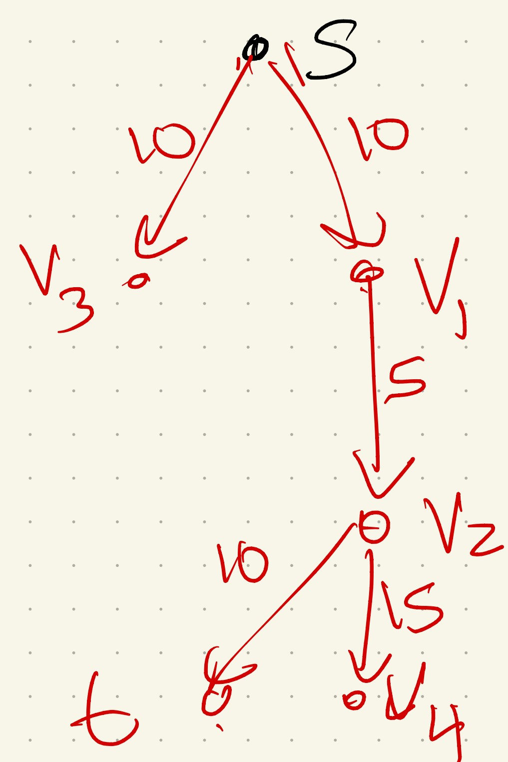
& G_i = residual graph after i repetitions of the loop.

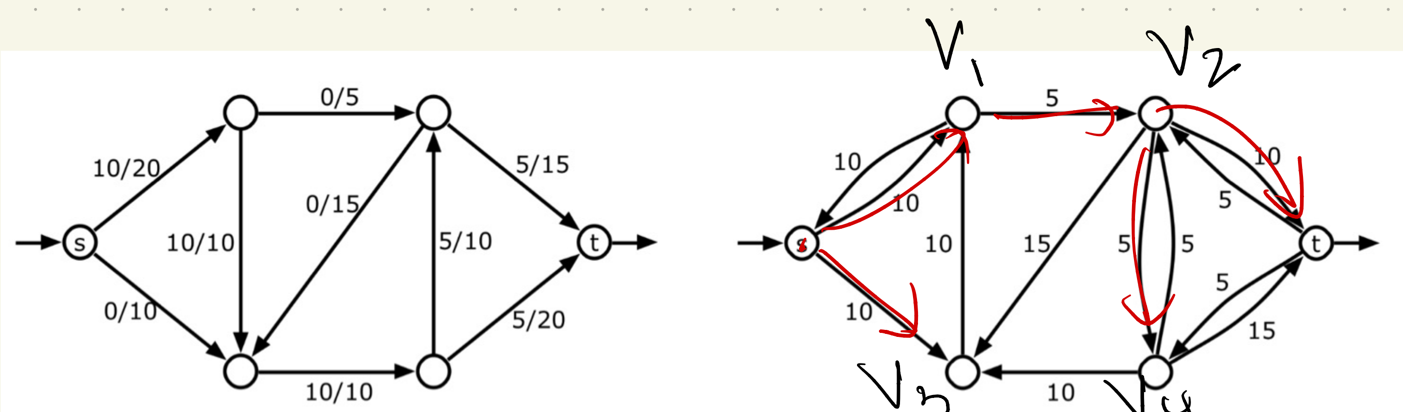
G_i has a BFS tree T_i, so let

$level_i(v)$ = depth in tree

(Note: once s can't reach t, then $level(t) = \infty$.)

BFS tree:





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)

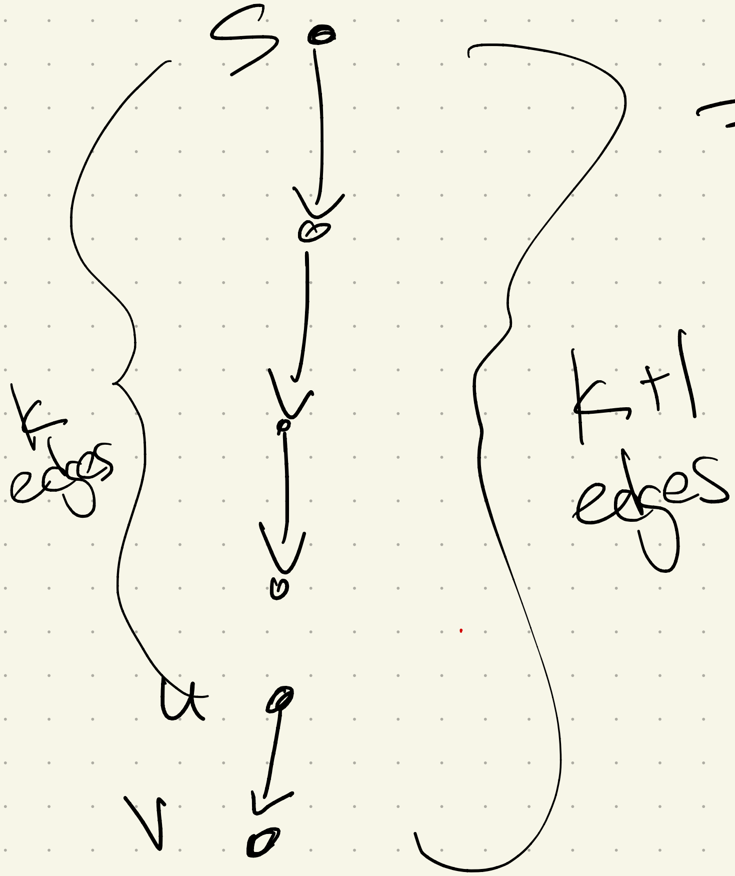
base case: s (level 0), always stays ≥ 0

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?

<u>cont:</u>

- $u \to v$ is an edge in G_{i-1}:

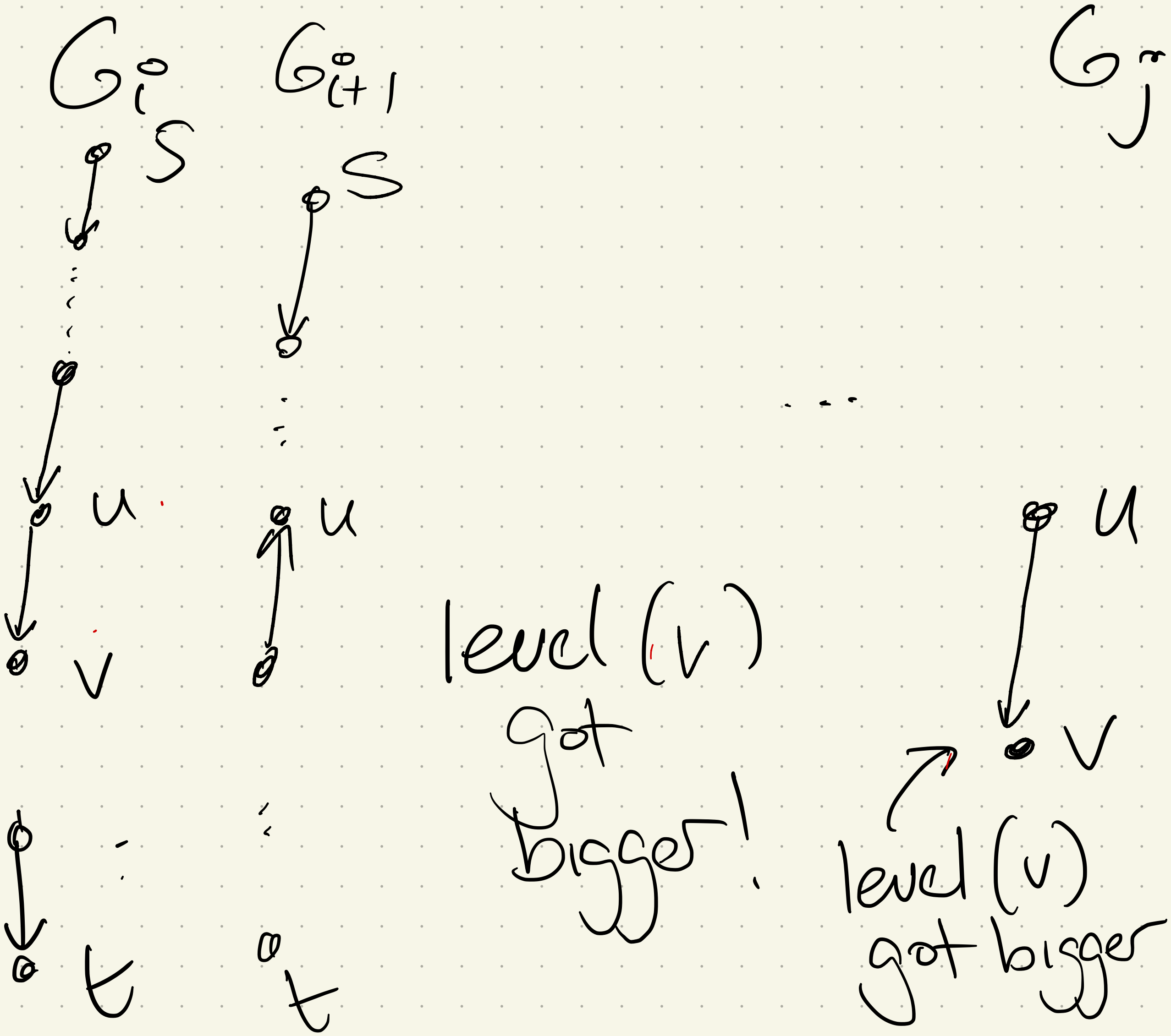
  $$\text{level}_{i-1}(v) \leq \text{level}_{i-1}(u) + 1$$

- might have come from pushing in G_{i-1}: here, it came from pushing a shortest path in the <u>last</u> round,

  so $v \to u$ was used last round.

Then: Edges can disappear & reapper.

How many times?



if level goes up +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 G_f
    while there is s-t path in G_f
        ~~Let p be a simple s-t path~~
        f' ← augment(f, P)
        f ← f'
        update G_f
    return f
```

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

$V \cdot E \cdot (V+E)$

$= VE^2$

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 <u>very</u> different techniques

- linear programming
- complex data structures
- <u>not</u> 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.