

Algorithms - Spring 2025

Flows + Cuts



## Recap

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

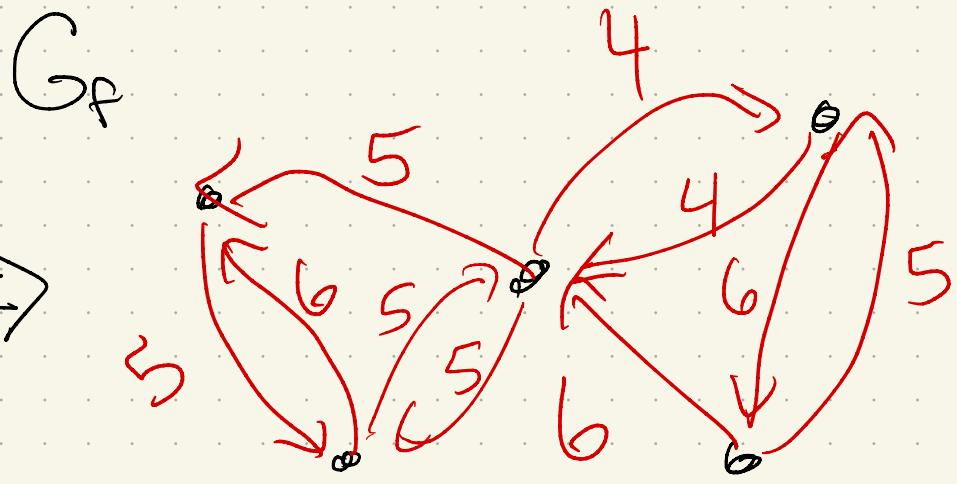
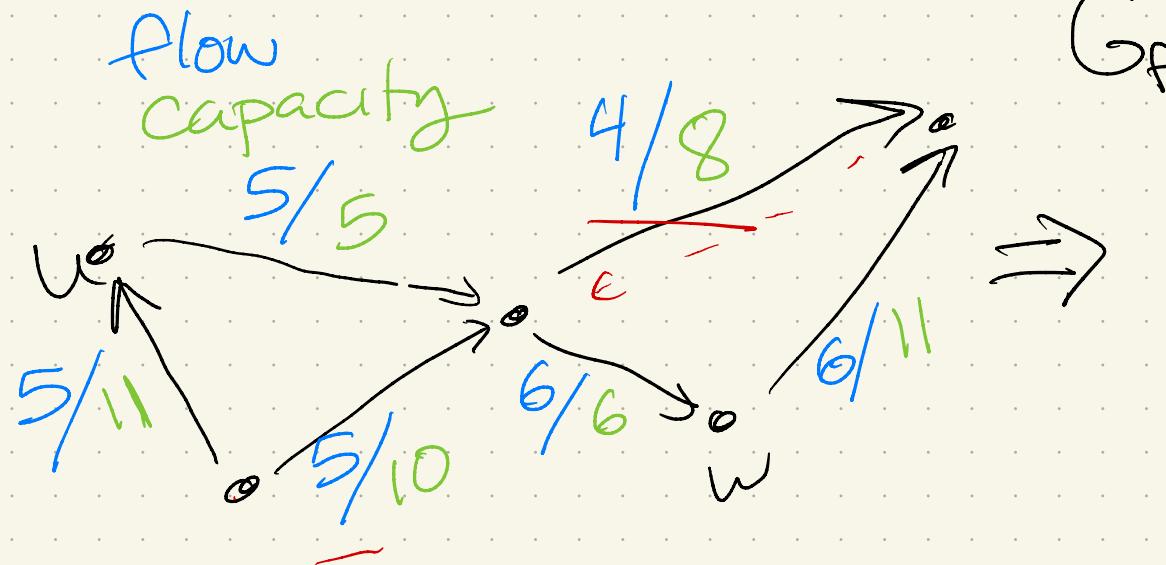
More formally: Residual network  $G_f$ :

Given  $G$  &  $f$ :

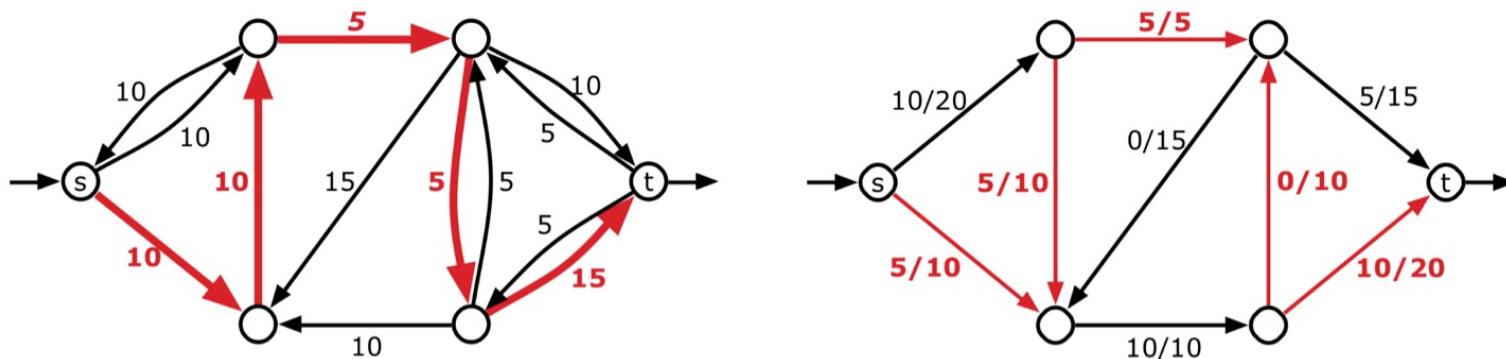
$$C_f(u \rightarrow v) = \begin{cases} c(u \rightarrow v) - f(u \rightarrow v) & \text{if } u \rightarrow v \\ f(u \rightarrow v) & \text{if } v \rightarrow u \text{ is in } E \\ \text{no edge (or } 0\text{)} & \text{otherwise} \end{cases}$$

*if  $u \rightarrow v$   
is in  $E$*   
*~~if reverse  
edge~~*

Ex:  $G, f$



# 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

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

bottle  
neck  
eye

$f'$ .

→ Then  $f$  was not  
maximum

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

On other hand:

If  $G_f$  has no  $s \rightarrow t$  path, find

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

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

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

Why?

# Immediate Algorithm:

Start with  $f = \emptyset$ .

Build  $G_f$

WFS( $G_f, s$ )

While  $t \neq s$  in same component:

find  $s \rightarrow t$  path via WFS

Augment along the path to  
get  $f'$

$f \leftarrow f'$

Build  $G_f$

WFS( $G_f, s$ )

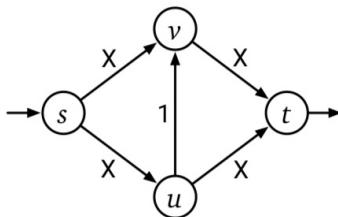
Runtime?

Why all this integrality stuff?

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

Can it be that bad?

Yes:



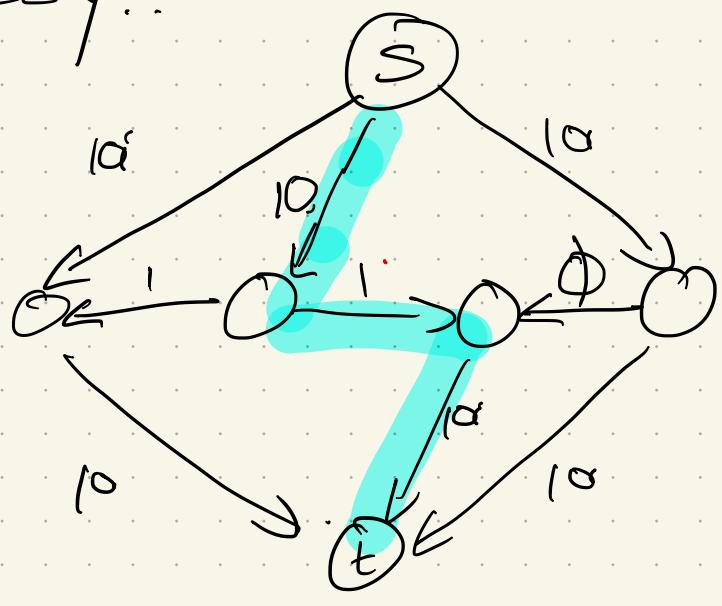
**Figure 10.7.** Edmonds and Karp's bad example for the Ford-Fulkerson algorithm.

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??

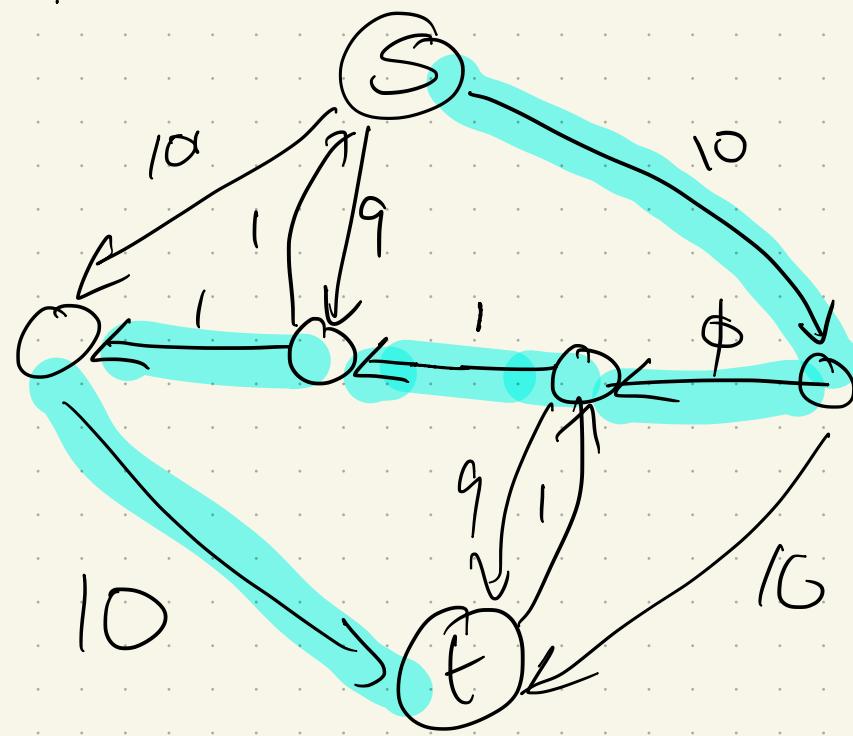
Simple:

$$1 - \phi = \phi^2$$

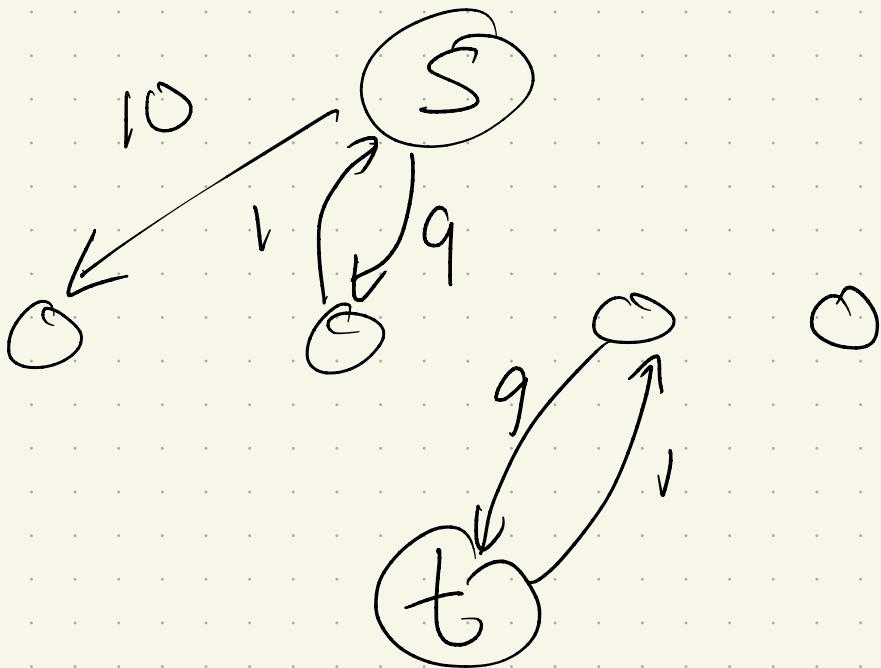
Gf:   ○   ○   ○   ○

(t)

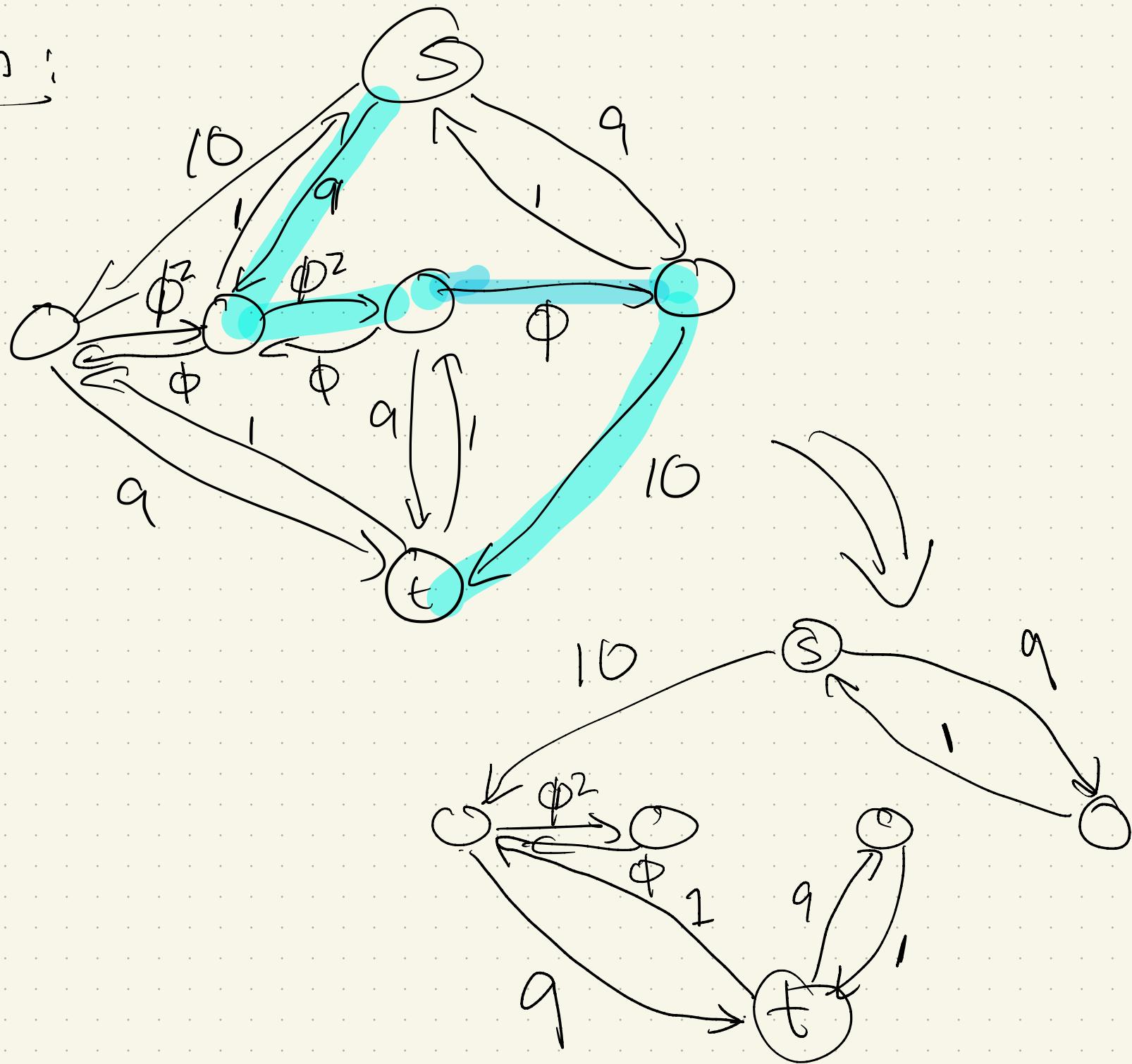
Next path:



↙  
New  $G_F$ :



Then:



Continue to push:

Ends with:

$$\phi, 0, \text{ and } 1-\phi = \phi^2$$

Repeat:

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

then

.

etc.

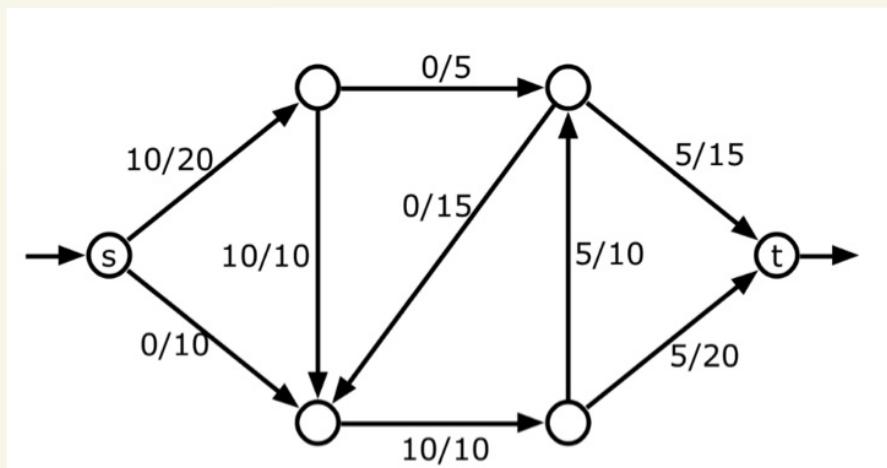
etc.

But, max flow = 21

# 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!

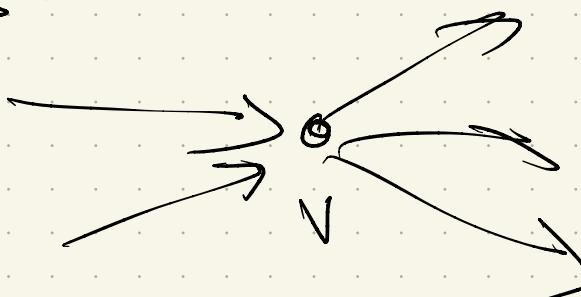
Formalizing:  
Linear combinations of flows  
result in flows:  
(ignoring capacities...)

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

$$f(u \rightarrow v) = 5$$

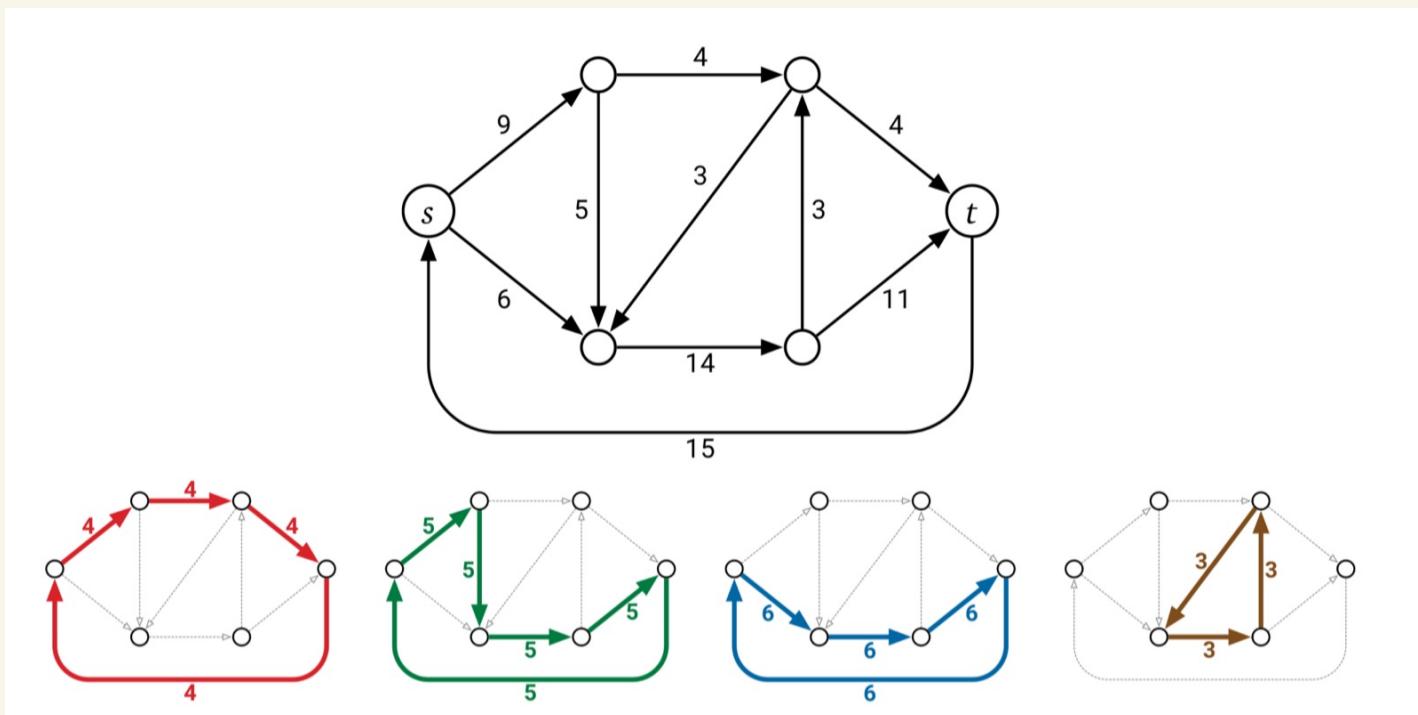
$$u \xrightarrow{0} v$$
$$g(u \rightarrow v) = 1$$

vertex constraints:-



Flow decomposition:

Add an edge  $t \rightarrow s$ , + can decompose a flow into cycles:



S.t. "Sum" of cycles = flow.

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

$$\hookrightarrow O(E^2 \log E \log f^*)$$

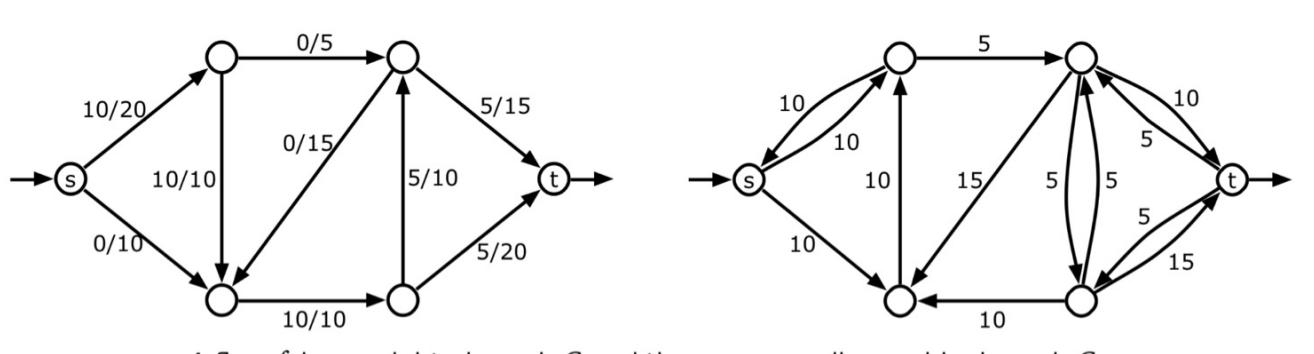
- ② shortest augmenting path (fewest edges)

$$\hookrightarrow O(VE^2)$$

Edmonds-Karp:

Largest bottleneck: how?

Take  $G_f$ :



Grow a tree from  $s$ , adding largest edge out each time  
↳ Similar to HW!

Runtime:

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~~  
 $f' \leftarrow \text{augment}(f, p)$

update  $G_f$

return  $f$

Replace:

Let  $p$  be the  
largest bottleneck  
path

# of repetitions in loop?

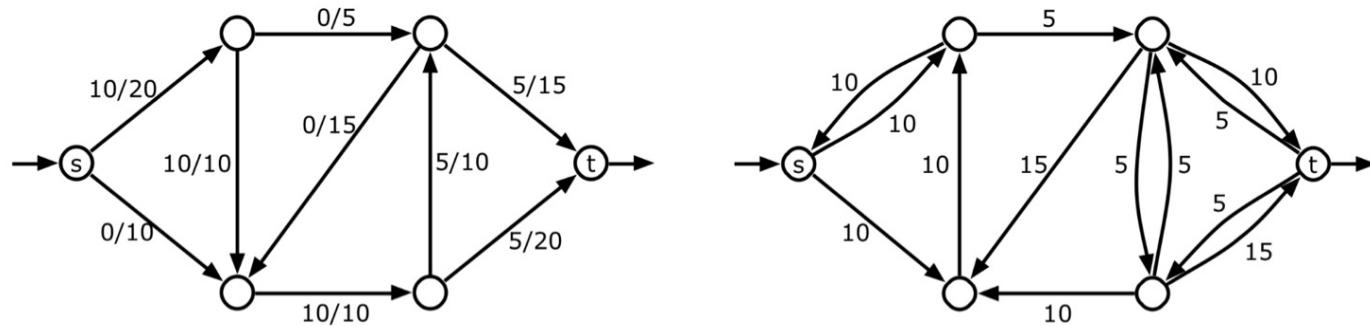
In FF, went down by  $\geq 1$ .

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:



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

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

$\hookrightarrow (1 - \frac{1}{E})^l f^*$  repetitions

What is  $l$ ?

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

$$(1 - \frac{1}{E})^l f^* < 1$$

Wait:  $< 1$ ??

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~~  
 $f' \leftarrow \text{augment}(f, p)$   
 $f \leftarrow f'$   
update  $G_f$

return  $f$

Replace:

Let  $p$  be the  
largest bottleneck  
path

find bottleneck:

+ # iterations:

# Shortest paths

Let  $P = \text{shortest}$

MAX Flow ( $G$ ):

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

while there is  $s-t$  path in  $G_f$

~~let  $P$  be a simple  $s-t$  path~~

$f' \leftarrow \text{augment}(f, P)$

$f \leftarrow f'$

update  $G_f$

return  $f$

$s-t$  path

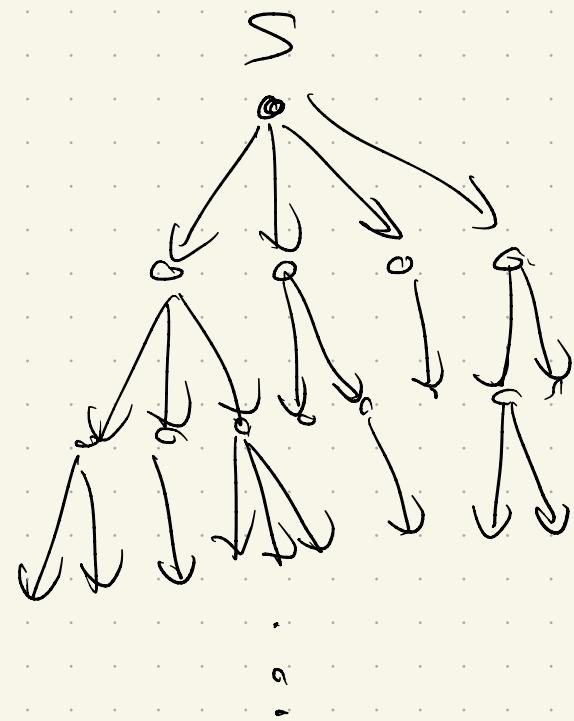
(# edges, not  
capacities)

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  $s_0$ .



What changes after we push flow?

Let  $G_0$  = initial  $G_f$

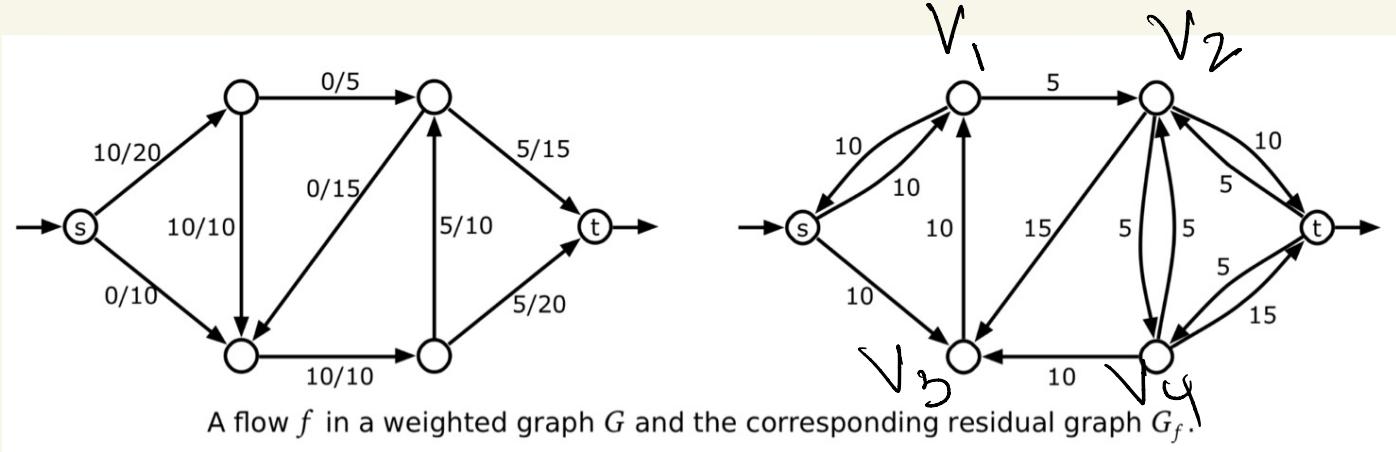
&  $G_i$  = residual graph after  $i$  repetitions of the loop.

$G_i$  has a BFS tree  $T_i$ , so let

$$\text{level}_i(v) = \text{depth in tree}$$

(Note: once  $s$  can't reach  $t$ , BFS tree:  
then  $\text{level}(t) = \infty$ )

•  $S$



Claim: levels only get bigger  
in each round.

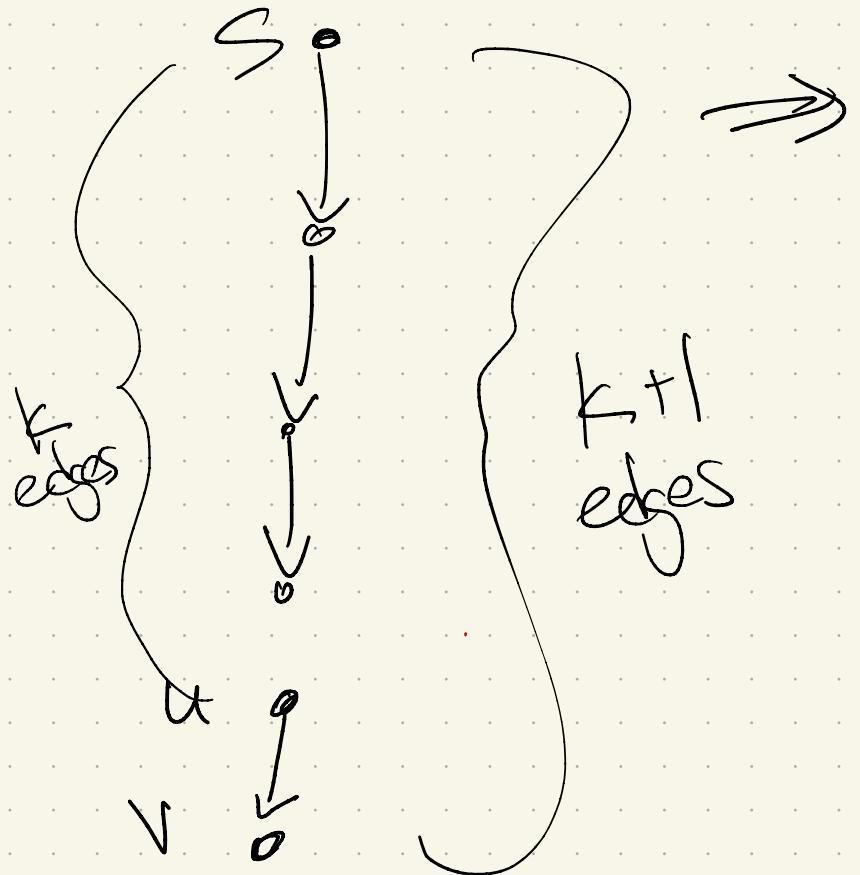
Proof: induction on level. (fix  $i$ )

base case:  $S$  (level 0), always stays  $\geq 0$

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  $s \rightsquigarrow v$



take  $u$  just before  
 $v$  on path

Now: how did we get this  
path in  $G_i$ ?

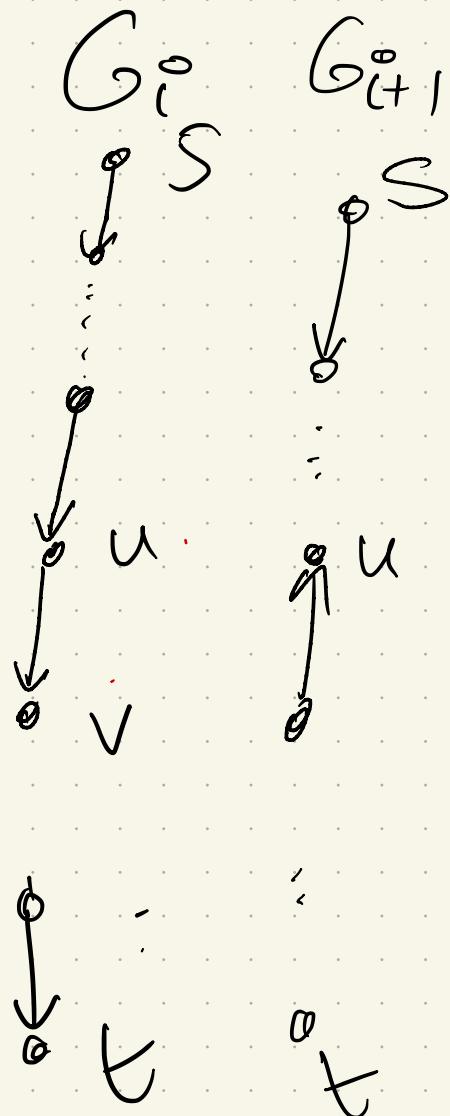
Cont:

- $u \rightarrow v$  is an edge in  $G_{i-1}^{\circ}$   
 $\text{level}_{i-1}(v) \leq \text{level}_{i-1}(u) + 1$

- might have come from pushing in  $G_i$ :  
here, it came from pushing a shortest path in the last round.  
so  $v \rightarrow u$  was used last round.

Then: Edges can disappear & reappear.

How many times?



level ( $v$ )

got

bigger!

level ( $v$ )  
got bigger

$G_j$

$u$

$v$

If level  
goes up  
+2, after  
 $\frac{V}{2}$  rounds



In each iteration of the loop —  
some edge disappears!

⇒  $\frac{V}{2} \cdot E$  repetitions

Total:

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  $s$ - $t$  path~~

$f' \leftarrow \text{augment}(f, P)$

$f \leftarrow f'$   
update  $G_f$

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  $\tilde{O}(mn)$  time. This breaks the long-standing  $\tilde{\Omega}(n^4)$ -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 first... ▾ 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  $\tilde{O}(m + nF^*/\epsilon)$  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.