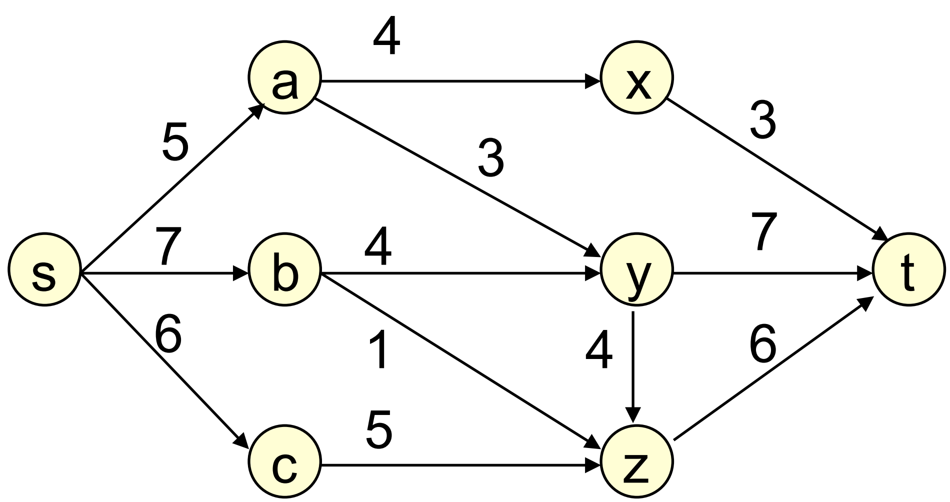
# CSE 421
# Introduction to Algorithms

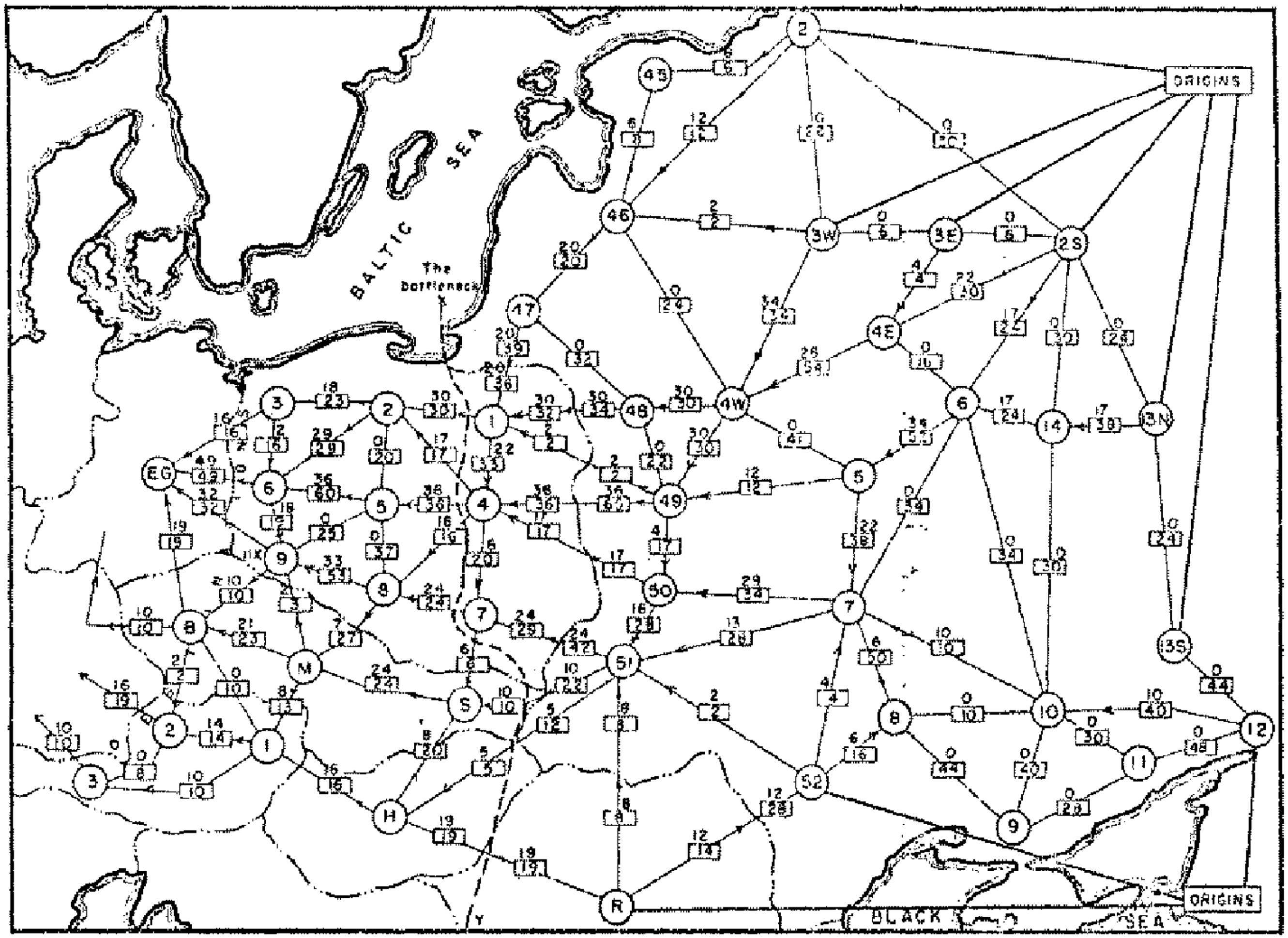
## The Network Flow Problem

# The Network Flow Problem




How much stuff can flow from *s* to *t*?

# Soviet Rail Network, 1955




Reference: *On the history of the transportation and maximum flow problems*.
Alexander Schrijver in Math Programming, 91: 3, 2002.

# Net Flow: Formal Definition

Given:

A digraph $G = (V,E)$

Two vertices $s,t$ in $V$
($s$ = source, $t$ = sink)

A *capacity* $c(u,v) \geq 0$
for each $(u,v) \in E$
(and $c(u,v) = 0$ for all non-edges $(u,v)$)

Find:

A *flow function* $f$: $V \times V \rightarrow R$ s.t., for all $u,v$:

- $f(u,v) \leq c(u,v)$ [Capacity Constraint]
- $f(u,v) = -f(v,u)$ [Skew Symmetry]
- if $u \neq s,t$, $f(u,V) = 0$ [Flow Conservation]

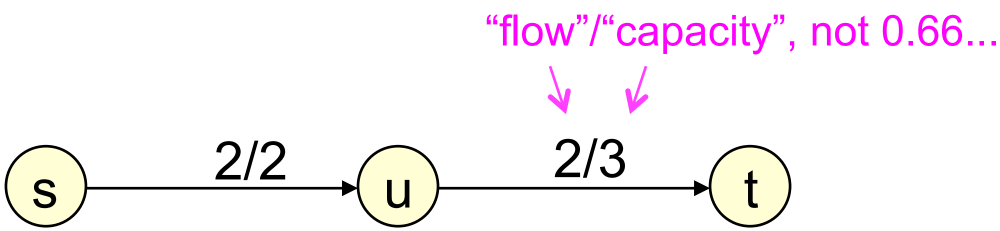
Maximizing total flow $|f| = f(s,V)$

Notation:

$$f(X,Y) = \sum_{x \in X} \sum_{y \in Y} f(x,y)$$

(technically, not quite the same definition as in the book…)

# Example: A Flow Function



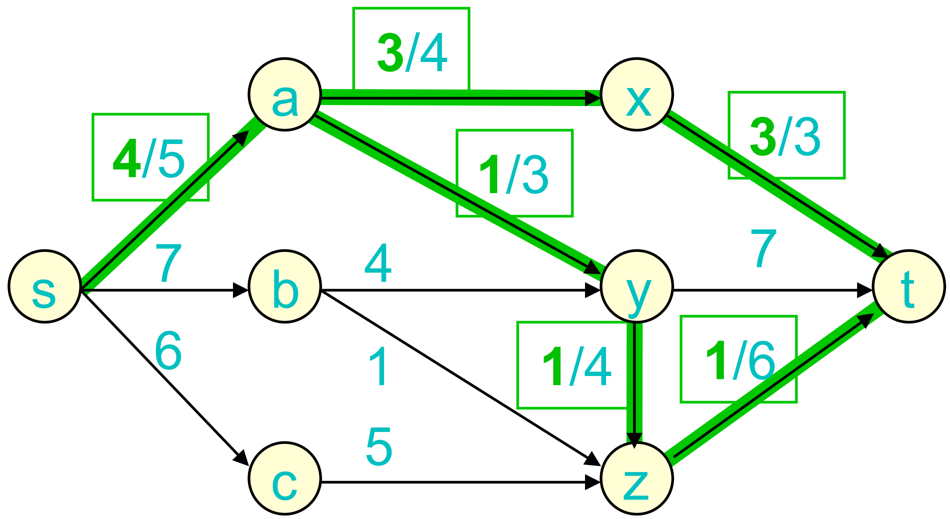

$f(s,u) = f(u,t) = 2$

$f(u,s) = f(t,u) = -2$ *(Why?)*

$f(s,t) = -f(t,s) = 0$ *(In every flow function for this G. Why?)*

$$f(u,V) = \sum_{v \in V} f(u,v) = f(u,s) + f(u,t) = -2 + 2 = 0$$

# Example: A Flow Function



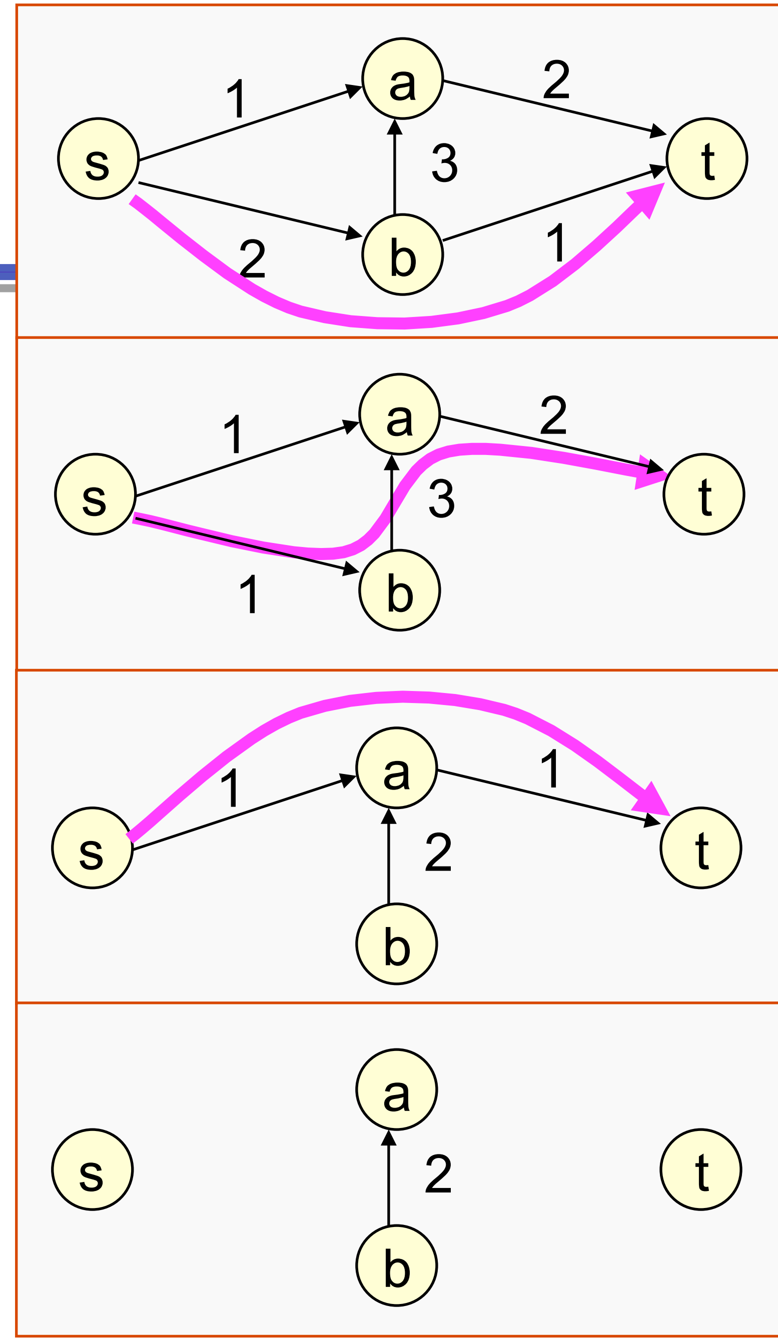

Not shown: $f(u,v)$ if $\leq 0$

Note: max flow $\geq 4$ since $f$ is a flow, $|f| = 4$

# Max Flow via a Greedy Alg?

While there is an $s \rightarrow t$ path in $G$

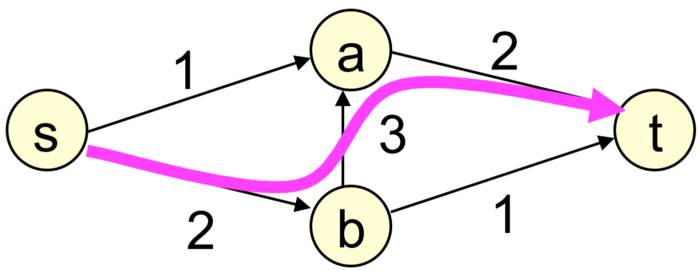
- Pick such a path, $p$
- Find $c_p$, the min capacity of any edge in $p$
- Subtract $c_p$ from all capacities on $p$
- Delete edges of capacity 0

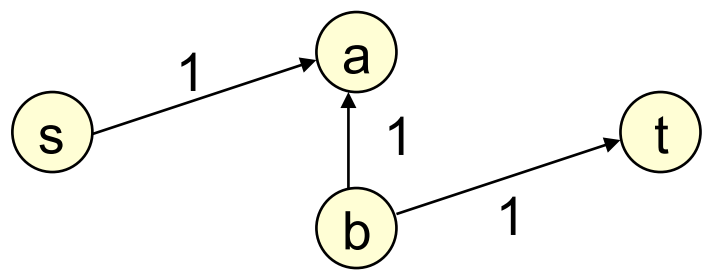


# Max Flow via a Greedy Alg?

This does NOT always find a max flow:

If you pick $s \rightarrow b \rightarrow a \rightarrow t$ first,




Flow stuck at 2, but 3 possible (above).

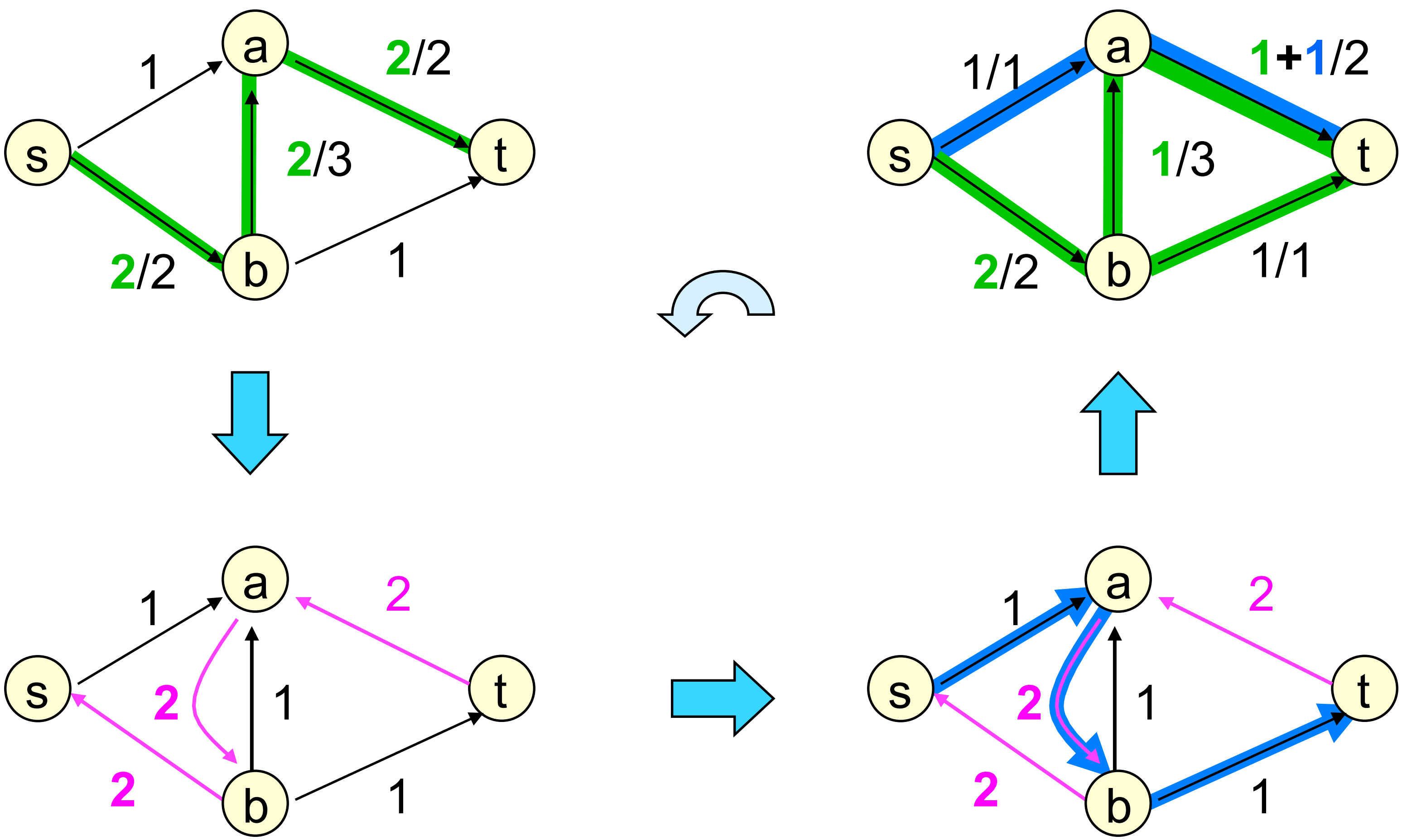
# A Brief History of Flow

| # | Year | Discoverer(s) | Bound |
|---|---|---|---|
| 1 | 1951 | Dantzig | $O(n^2 mC)$ |
| 2 | 1955 | Ford & Fulkerson | $O(nmC)$ |
| 3 | 1970 | Dinitz; Edmonds & Karp | $O(nm^2)$ |
| 4 | 1970 | Dinitz | $O(n^2 m)$ |
| 5 | 1972 | Edmonds & Karp; Dinitz | $O(m^2 \log C)$ |
| 6 | 1973 | Dinitz;Gabow | $O(nm \log C)$ |
| 7 | 1974 | Karzanov | $O(n^3)$ |
| 8 | 1977 | Cherkassky | $O(n^2 \operatorname{sqrt}(m))$ |
| 9 | 1980 | Galil & Naamad | $O(nm \log^2 n)$ |
| 10 | 1983 | Sleator & Tarjan | $O(nm \log n)$ |
| 11 | 1986 | Goldberg &Tarjan | $O(nm \log (n^2/m))$ |
| 12 | 1987 | Ahuja & Orlin | $O(nm + n^2 \log C)$ |
| 13 | 1987 | Ahuja et al. | $O(nm \log(n \operatorname{sqrt}(\log C)/(m+2))$ |
| 14 | 1989 | Cheriyan & Hagerup | $E(nm + n^2 \log^2 n)$ |
| 15 | 1990 | Cheriyan et al. | $O(n^3/\log n)$ |
| 16 | 1990 | Alon | $O(nm + n^{8/3} \log n)$ |
| 17 | 1992 | King et al. | $O(nm + n^{2+\varepsilon})$ |
| 18 | 1993 | Phillips & Westbrook | $O(nm(\log_{m/n} n + \log^{2+\varepsilon} n)$ |
| 19 | 1994 | King et al. | $O(nm(\log_{m/(n \log n)} n)$ |
| 20 | 1997 | Goldberg & Rao | $O(m^{3/2} \log(n^2/m) \log C)$ ; $O(n^{2/3} m \log(n^2/m) \log C)$ |
| … | … | … | … |

n = # of vertices
m= # of edges
C = Max capacity

Source: Goldberg & Rao, FOCS '97

# Greed Revisited



# Residual Capacity

The *residual capacity* (w.r.t. $f$) of $(u,v)$ is

$c_f(u,v) = c(u,v) - f(u,v)$

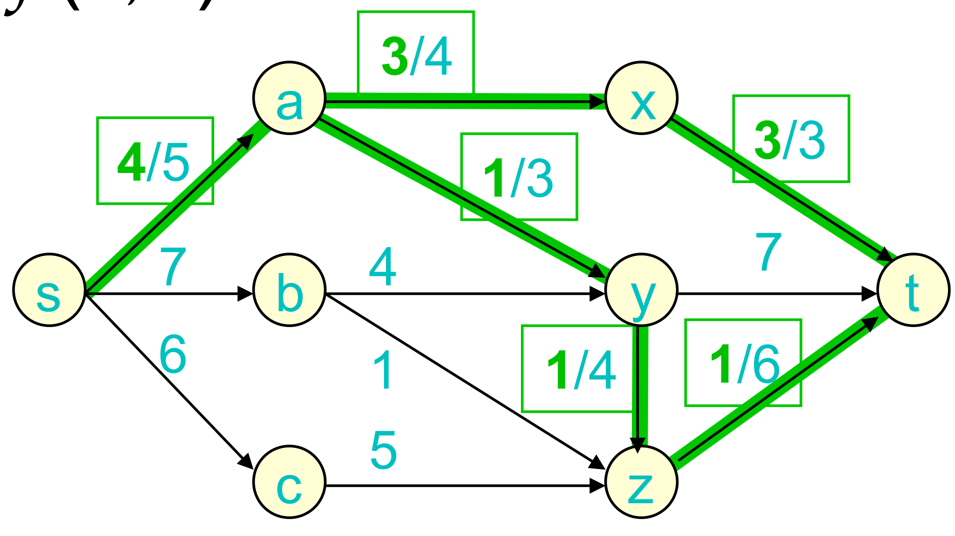
E.g.:

$c_f(s,b) = 7;$

$c_f(a,x) = 1;$

$c_f(x,a) = 3;$

$c_f(x,t) = 0$ (*a* saturated edge)



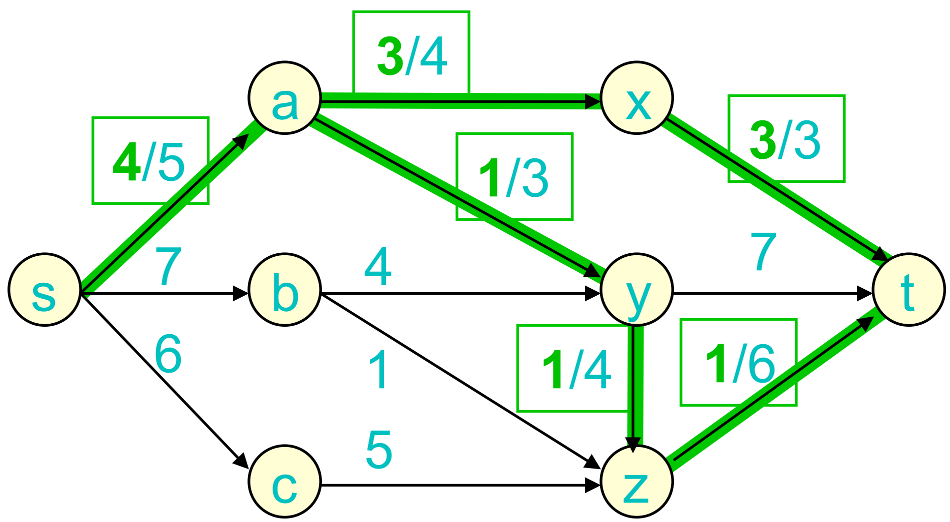
# Residual Networks & Augmenting Paths

The *residual network* (w.r.t. $f$ ) is the graph $G_f = (V, E_f)$, where

$$E_f = \{ (u,v) \mid c_f(u,v) > 0 \}$$

An *augmenting path* (w.r.t. $f$ ) is a simple $s \rightarrow t$ path in $G_f$

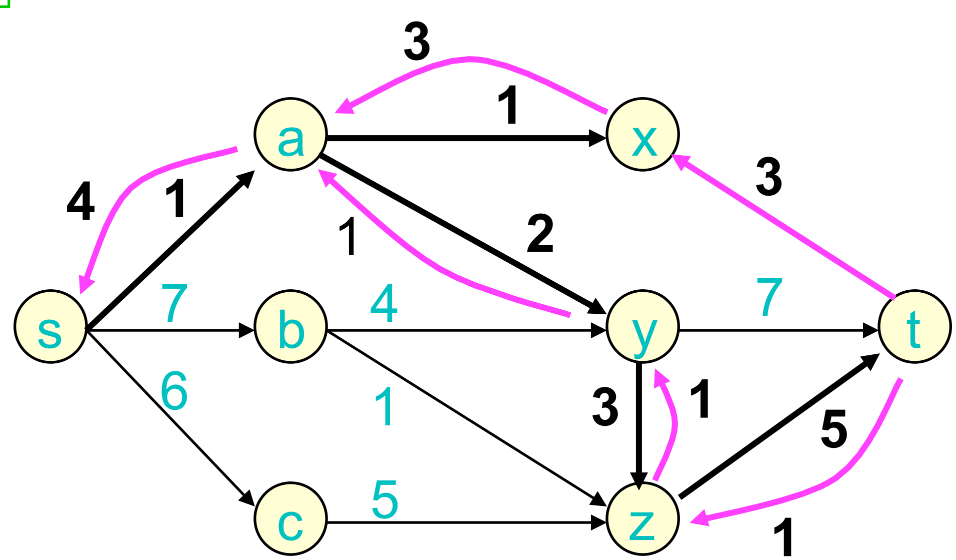
# A Residual Network



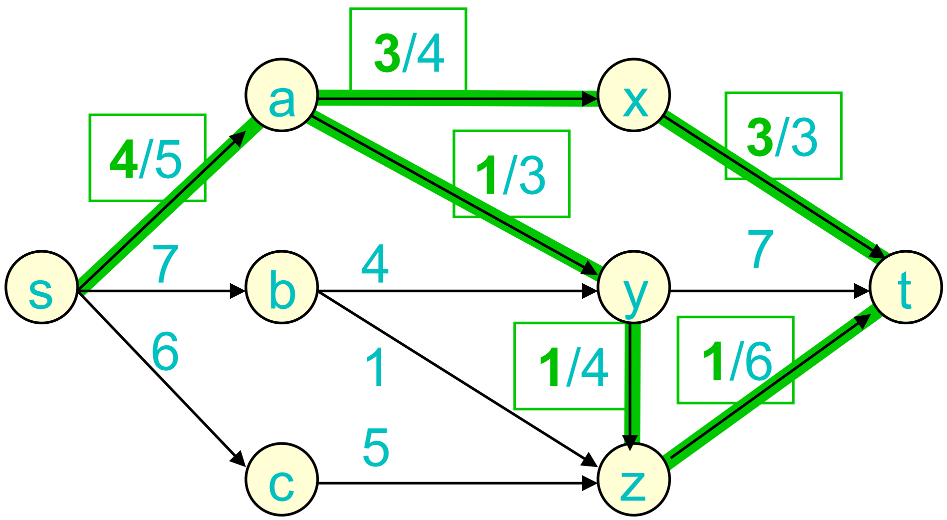

*residual network*: the graph

$G_f = (V, E_f)$, where

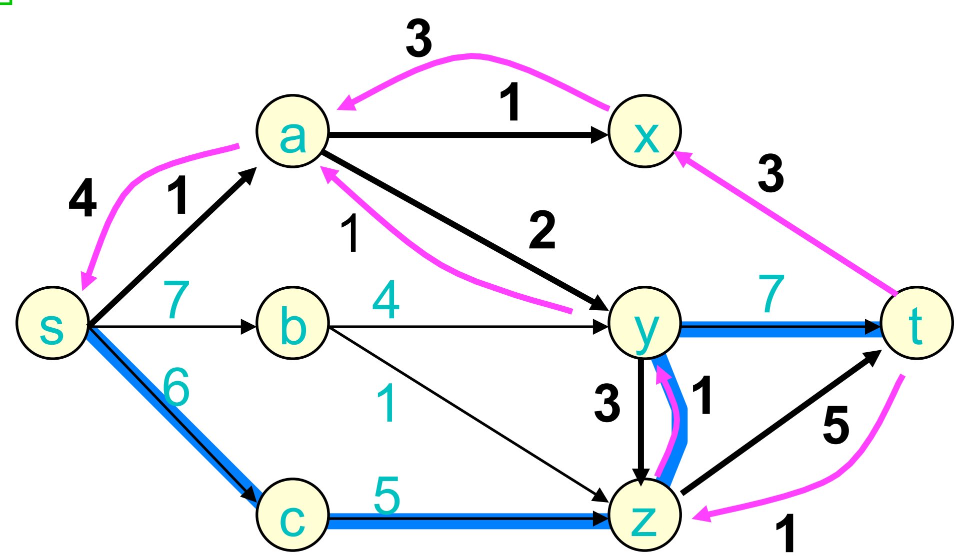
$E_f = \{ (u,v) \mid c_f(u,v) > 0 \}$



# An Augmenting Path




*augmenting path:*
a simple $s \to t$ path in $G_f$

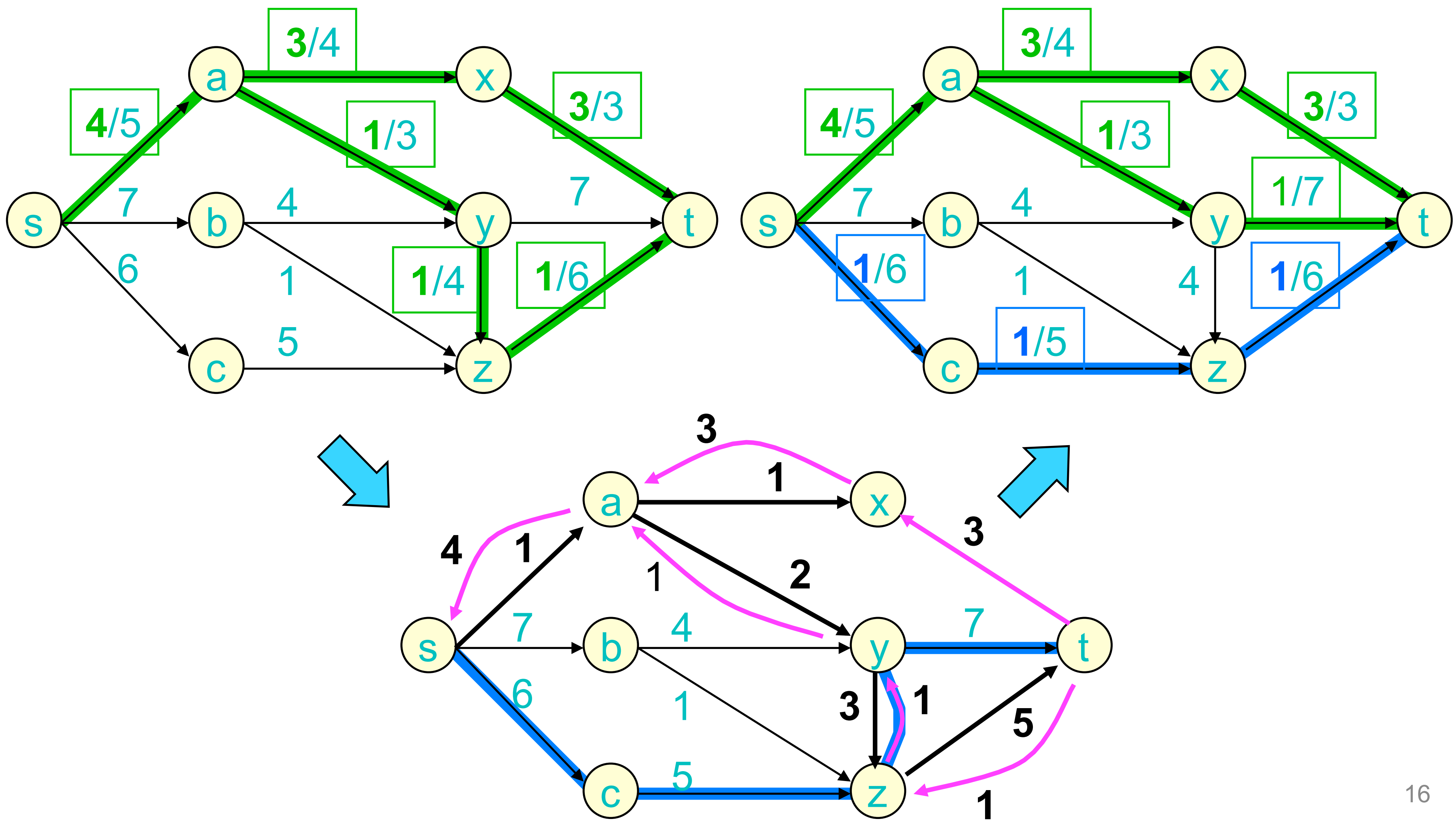


# Lemma 1

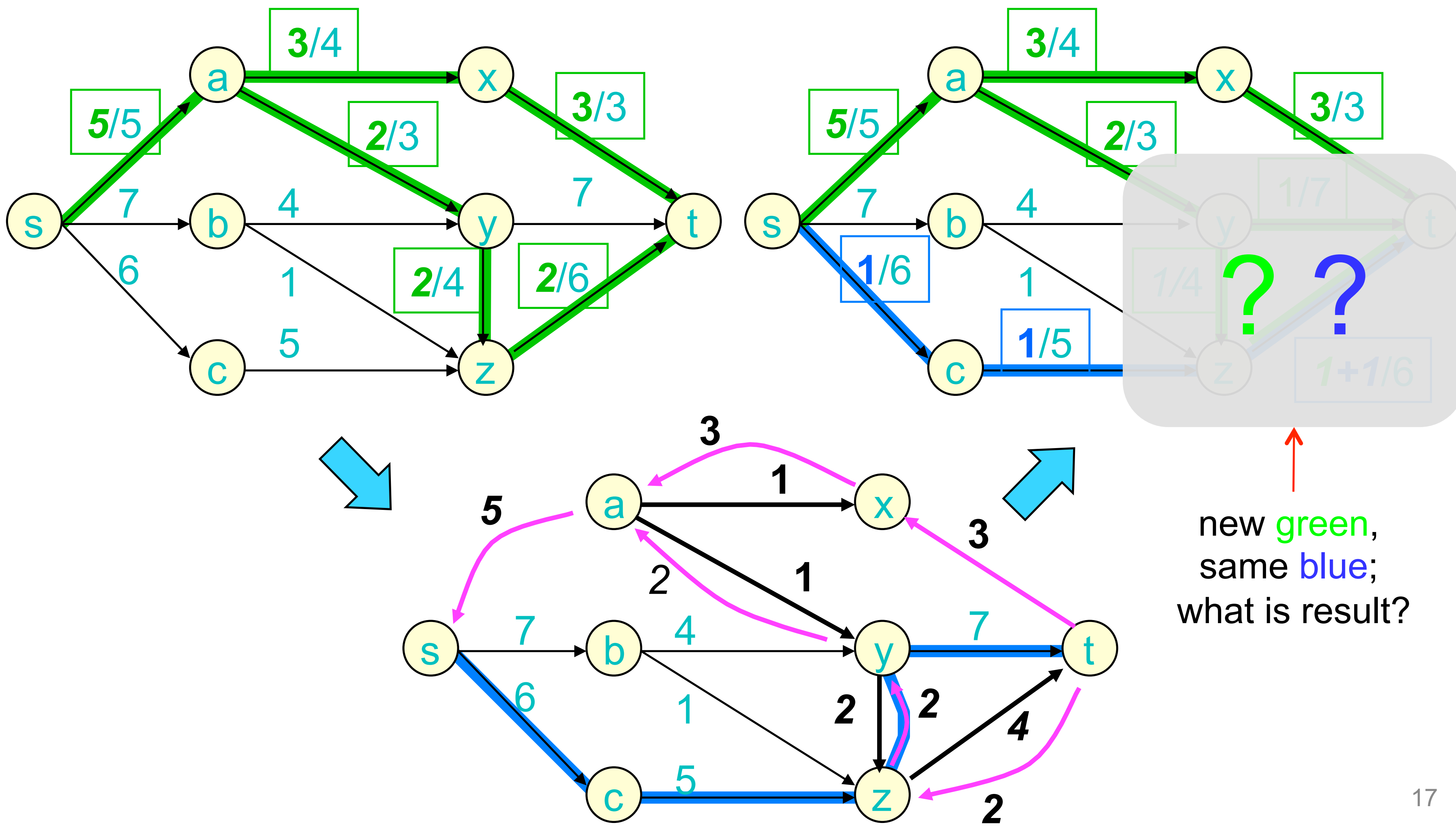
If $f$ admits an augmenting path $p$, then $f$ is not maximal.

Proof: “obvious” -- augment along $p$ by $c_p$, the min residual capacity of $p$’s edges.

# Augmenting A Flow



# Augmenting A Flow



# Lemma 1':
# Augmented Flows are Flows

If $f$ is a flow & $p$ an augmenting path of capacity $c_p$, then $f'$ is also a valid flow, where

$$\boxed{f'(u,v) = \begin{cases} f(u,v) + c_p, & \text{if } (u,v) \text{ in path } p \\ f(u,v) - c_p, & \text{if } (v,u) \text{ in path } p \\ f(u,v), & \text{otherwise} \end{cases}}$$

Proof:

a) Flow conservation – easy
b) Skew symmetry – easy
c) Capacity constraints – pretty easy; next slides

# Lma 1': Augmented Flows are Flows

$$f'(u,v) = \begin{cases} f(u,v) + c_p, & \text{if } (u,v) \text{ in path } p \\ f(u,v) - c_p, & \text{if } (v,u) \text{ in path } p \\ f(u,v), & \text{otherwise} \end{cases}$$

$f$ a flow & $p$ an aug path of cap $c_p$, then $f'$ also a valid flow.

Proof (Capacity constraints):

$(u,v)$, $(v,u)$ not on path: no change

$(u,v)$ on path:

$$\begin{aligned} f'(u,v) &= f(u,v) + c_p \\ &\le f(u,v) + c_f(u,v) \\ &= f(u,v) + c(u,v) - f(u,v) \\ &= c(u,v) \end{aligned}$$

$$\begin{aligned} f'(v,u) &= f(v,u) - c_p \\ &< f(v,u) \\ &\le c(v,u) \end{aligned}$$

*QED*

Residual Capacity:

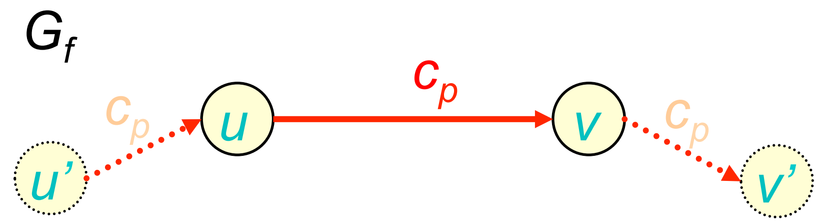
$0 < c_p \le c_f(u,v) = c(u,v) - f(u,v)$

Cap Constraints:

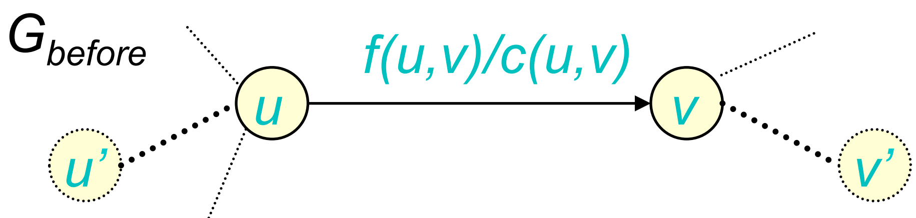
$-c(v,u) \le f(u,v) \le c(u,v)$

# Lemma 1' Example – Case 1

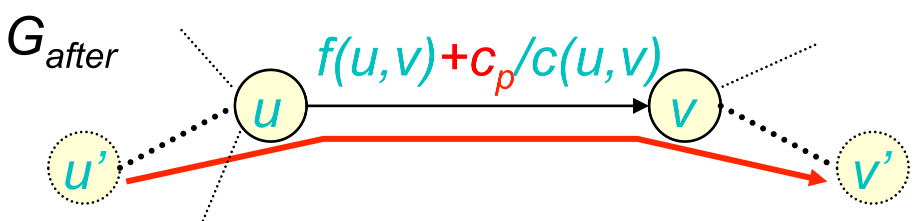
Let *(u,v)* be any edge in augmenting path. Note $c_f(u,v) = c(u,v) - f(u,v) \geq c_p > 0$




Case 1: $f(u,v) \geq 0$:

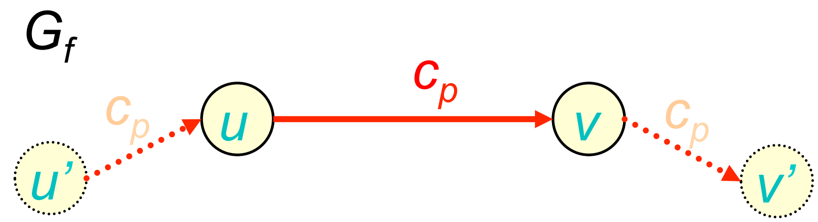



Add forward flow



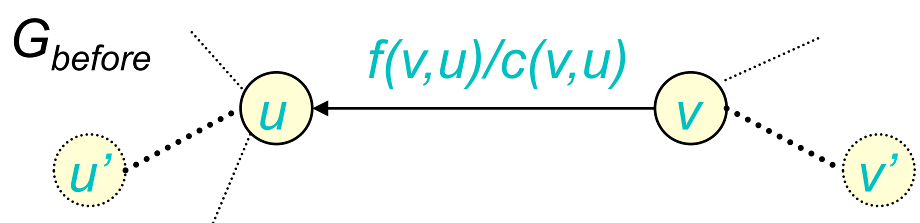
# Lemma 1' Example – Case 2

Let $(u,v)$ be any edge in augmenting path. Note $c_f(u,v) = c(u,v) - f(u,v) \geq c_p > 0$

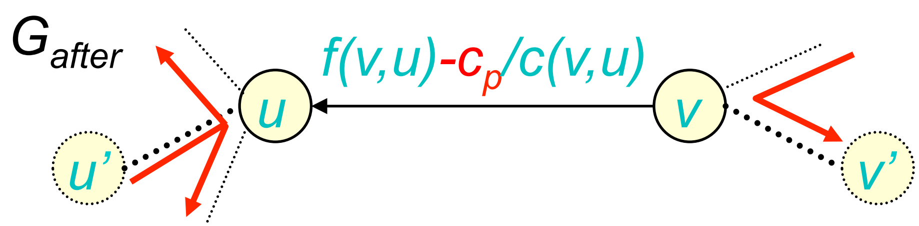



Case 2: $f(u,v) \leq -c_p$:

$f(v,u) = -f(u,v) \geq c_p$



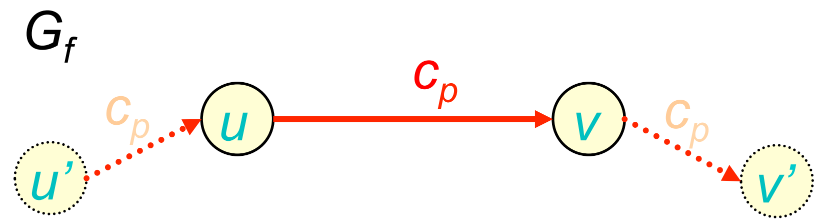

Cancel/redirect reverse flow



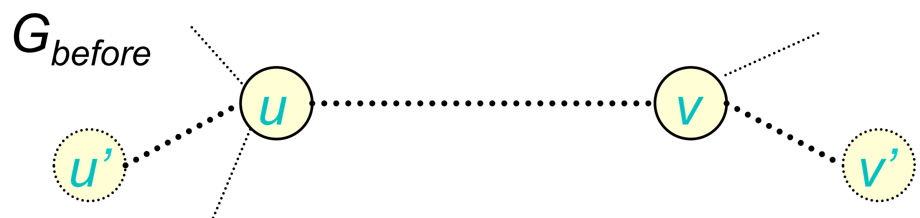
# Lemma 1' Example – Case 3

Let $(u,v)$ be any edge in augmenting path. Note

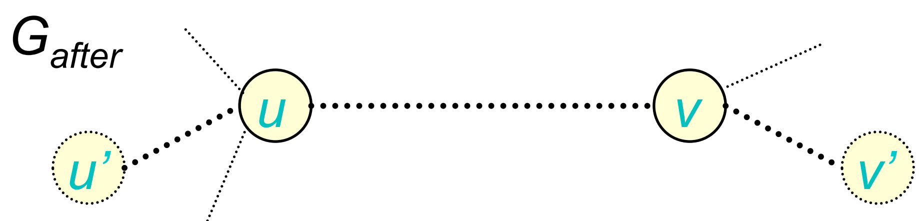
$c_f(u,v) = c(u,v) – f(u,v) \geq c_p > 0$




Case 3: $-c_p < f(u,v) < 0$:




**???**




[E.g., $c_p = 8$, $f(u,v) = -5$]

# Lemma 1' Example – Case 3

Let $(u,v)$ be any edge in augmenting path. Note

$c_f(u,v) = c(u,v) - f(u,v) \geq c_p > 0$

Case 3: $-c_p < f(u,v) < 0$

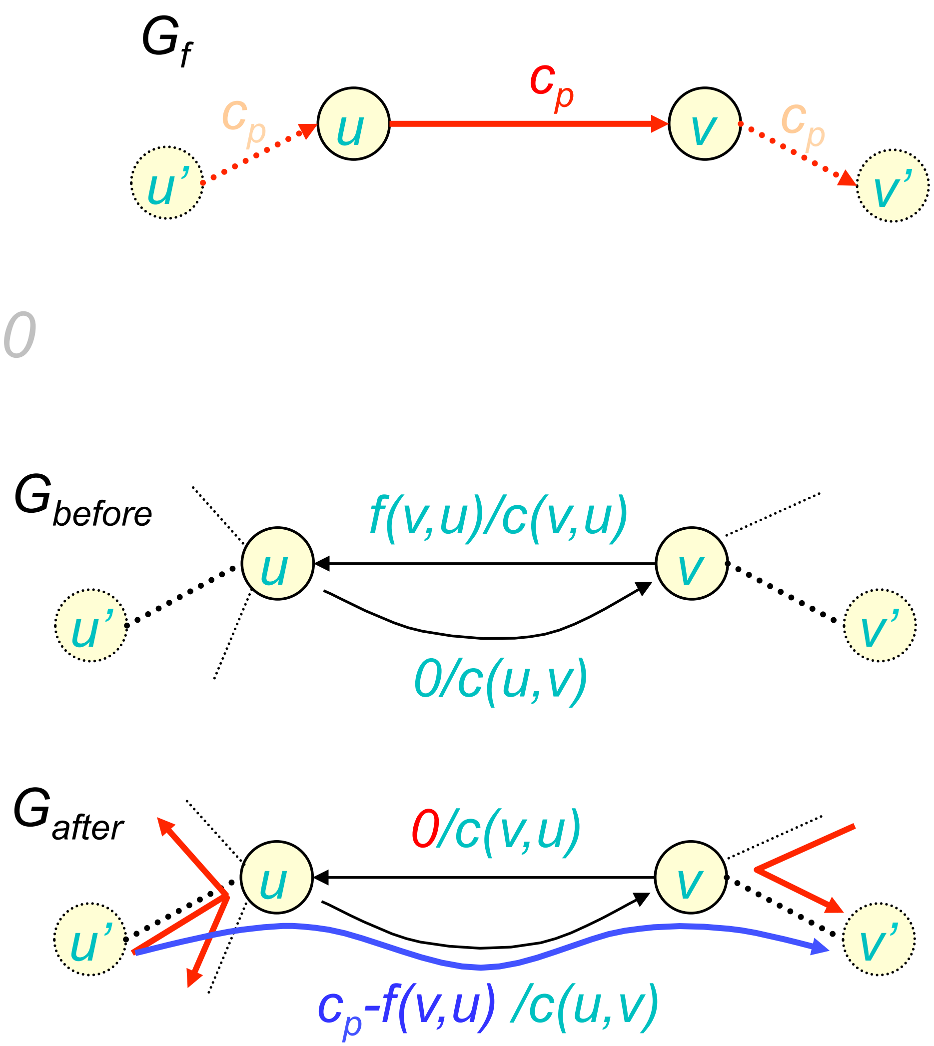
$c_p > f(v,u) > 0$:

Both:

cancel/redirect reverse flow

*and*

add forward flow



# Ford-Fulkerson Method

While $G_f$ has an augmenting path,
augment

Questions:

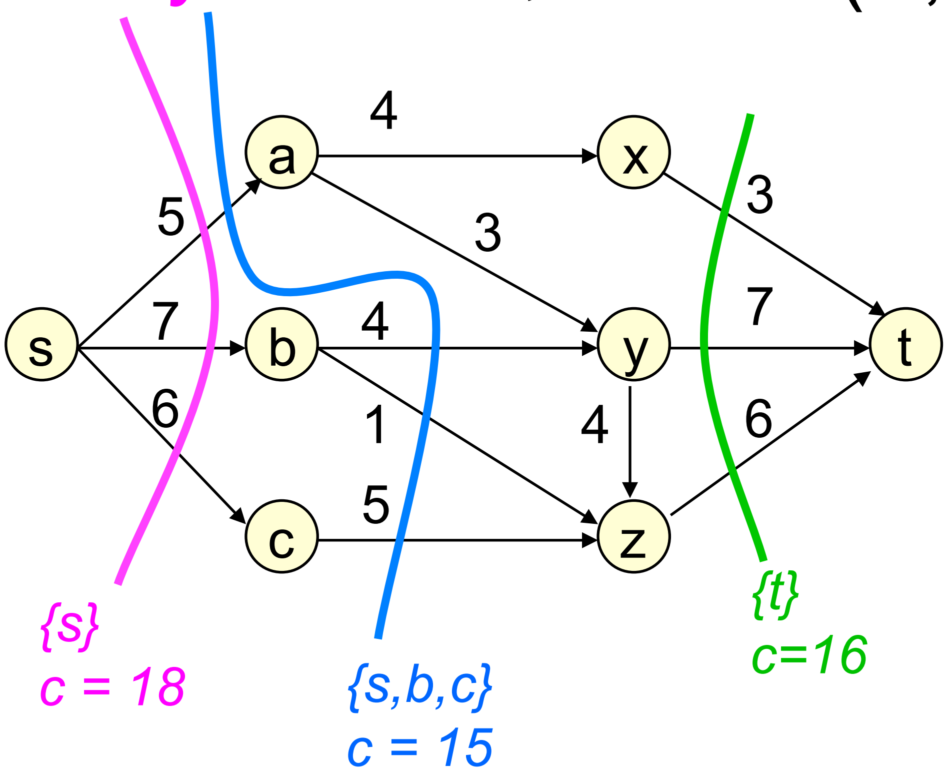
- » Does it halt?
- » Does it find a maximum flow?
- » How fast?

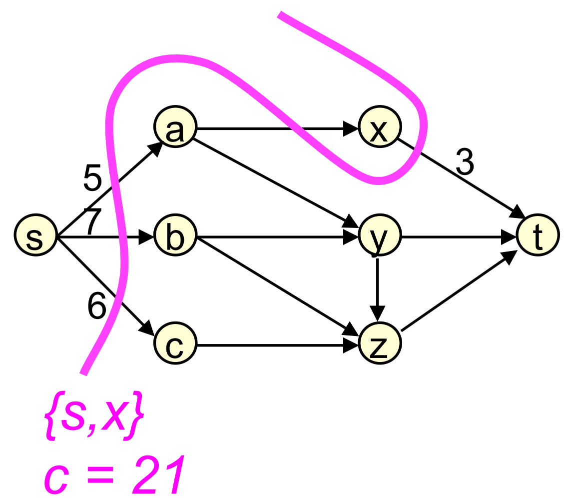
# Cuts

A partition $S, T$ of $V$ is a *cut* if $s \in S$, $t \in T$.

*Capacity* of cut $S, T$ is $$c(S,T) = \sum_{\substack{u \in S \\ v \in T}} c(u,v)$$

sum of caps of edges *from S to T*



# Lemma 2

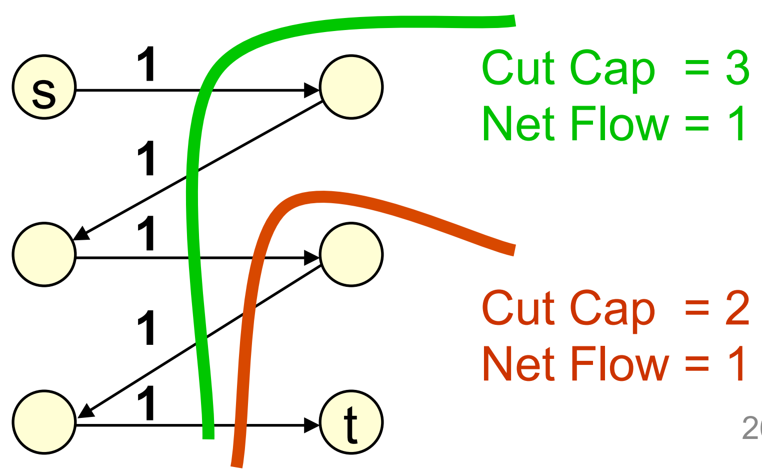
For any flow $f$ and any cut $S, T$,

the net flow across the cut equals the total flow, i.e., $|f| = f(S,T)$, and

the net flow across the cut cannot exceed the capacity of the cut, i.e. $f(S,T) \le c(S,T)$

Corollary:

Max flow ≤ Min cut



# Lemma 2

For any flow *f* and any cut *S, T,*

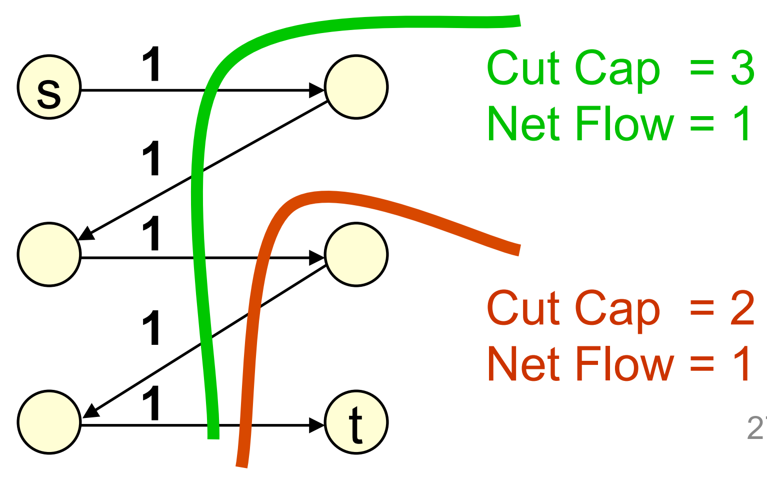
net flow across cut = total flow ≤ cut capacity

Proof:

Track a flow unit. Starts at *s*, ends at *t*.

crosses cut an odd # of times; net = 1.

Last crossing uses a forward edge totaled in *C(S, T)*



# Max Flow / Min Cut Theorem

For any flow $f$, the following are equivalent

(1) $|\, f \,| = c(S,T)$ for some cut $S,T$ (a min cut)

(2) $f$ is a maximum flow

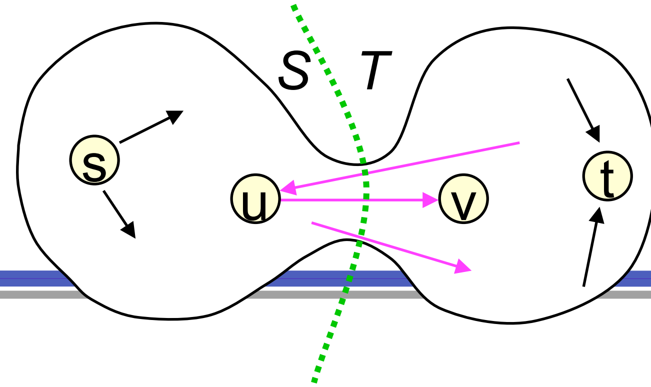
(3) $f$ admits no augmenting path

Proof:

(1) $\Rightarrow$ (2): corollary to lemma 2

(2) $\Rightarrow$ (3): contrapositive of lemma 1

# (3) ⇒ (1)
## (no aug) ⇒ (cut)




Idea: where's the bottleneck

$S = \{ u \mid \exists$ an augmenting path wrt $f$ from $s$ to $u$ $\}$

$T = V - S;\ \ s \in S,\ t \in T$

For any $(u,v)$ in $S \times T$, $\exists$ an augmenting path from $s$ to $u$, but not to $v$.

$\therefore$ $(u,v)$ has 0 residual capacity:

$(u,v) \in E \Rightarrow$ saturated $\qquad f(u,v) = c(u,v)$

$(v,u) \in E \Rightarrow$ no flow $\qquad f(u,v) = 0 = -f(v,u)$

This is true for every edge crossing the cut, i.e.

$$|f| = f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) =$$

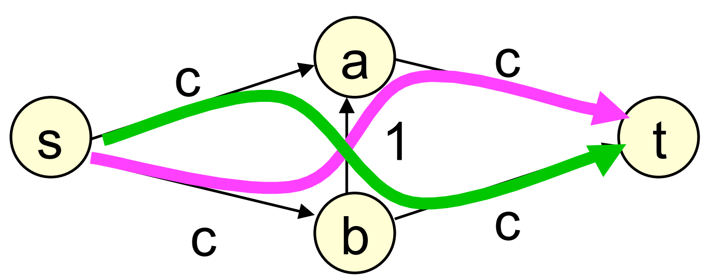
$$\sum_{u \in S, v \in T, (u,v) \in E} f(u,v) = \sum_{u \in S, v \in T, (u,v) \in E} c(u,v) = c(S,T)$$

# Corollaries & Facts

If Ford-Fulkerson terminates, then it's found a max flow.

It will terminate if *c(e)* integer or rational (but may not if they're irrational).

However, may take exponential time, even with integer capacities:




$c = 10^{99}$, say

# How to Make it Faster

Several ways. Three important ones:

- Edmonds-Karp ‘70; Dinitz ‘70
  - 1st “strongly” poly time alg. (next) $T = O(nm^2)$
- “Scaling” [Edmonds-Karp, ‘72; Dinitz ’72]
  - do *largest* edges first; see text, and below.
  - if C = max capacity, $T = O(m^2 \log C)$
- Preflow-Push [Goldberg, Tarjan ‘86]
  - see text $T = O(n^3)$

# Edmonds-Karp-Dinitz '70 Algorithm

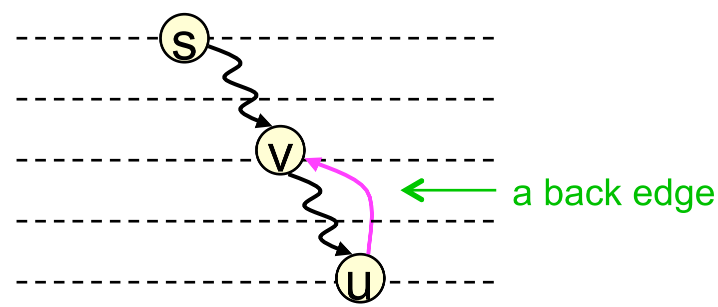
Use a shortest augmenting path
(via Breadth First Search in residual graph)

Time: $O(n\ m^2)$

# BFS/Shortest Path Lemmas

Distance from s is never reduced by:

- Deleting an edge
  proof: no new (hence no shorter) path created
- Adding an edge (*u*,*v*), provided *v* is nearer than *u*
  proof: BFS is unchanged, since *v* visited before (*u*,*v*) examined

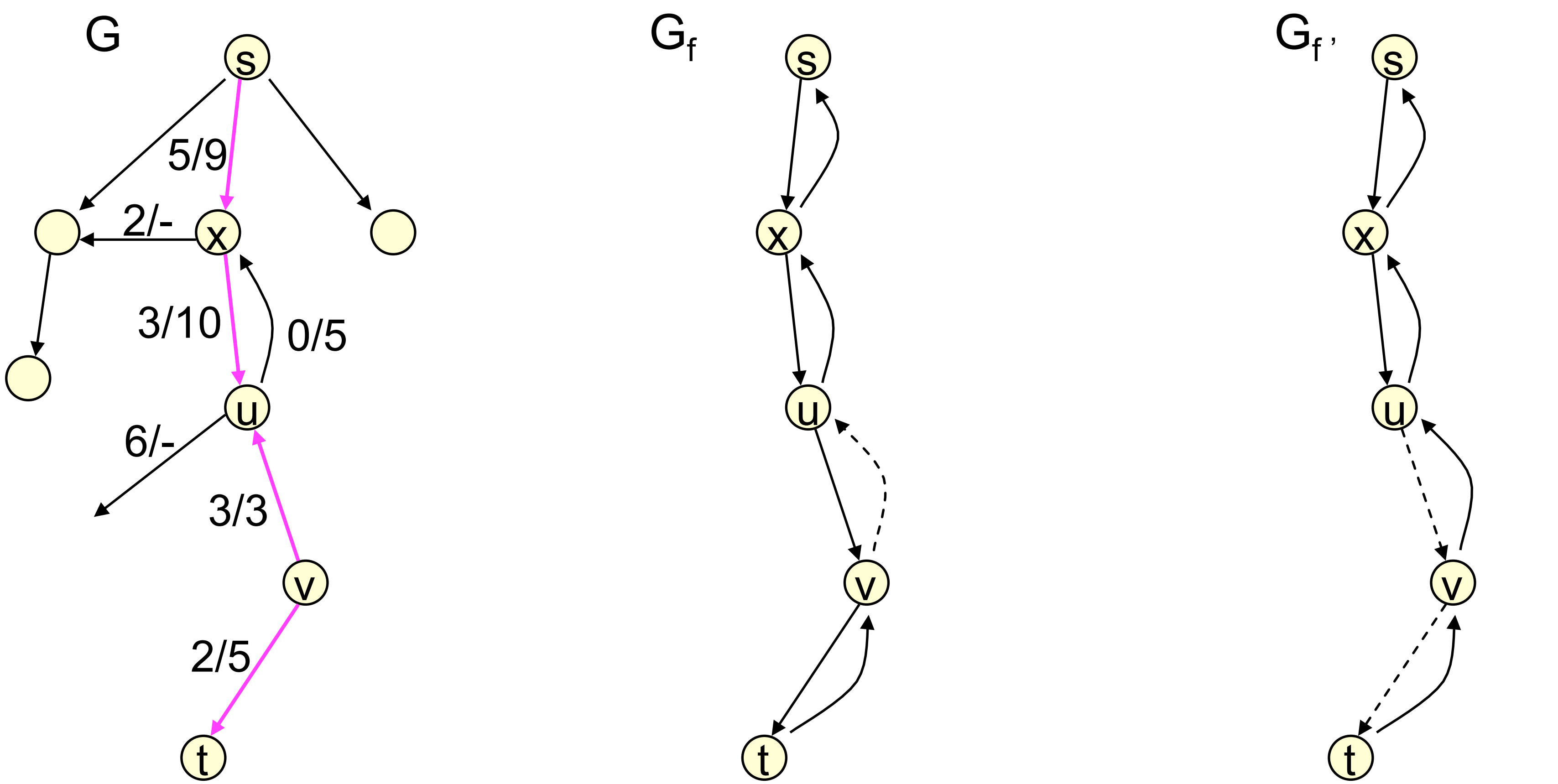


# Lemma 3

Let $f$ be a flow, $G_f$ the residual graph, and $p$ a shortest augmenting path. Then no vertex is closer to $s$ in the new residual graph $G_{f+p}$ after augmentation along $p$.

Proof: Augmentation only deletes edges, adds back edges

# Augmentation vs BFS



# Theorem 2

The Edmonds-Karp-Dinitiz Algorithm performs O(mn) flow augmentations

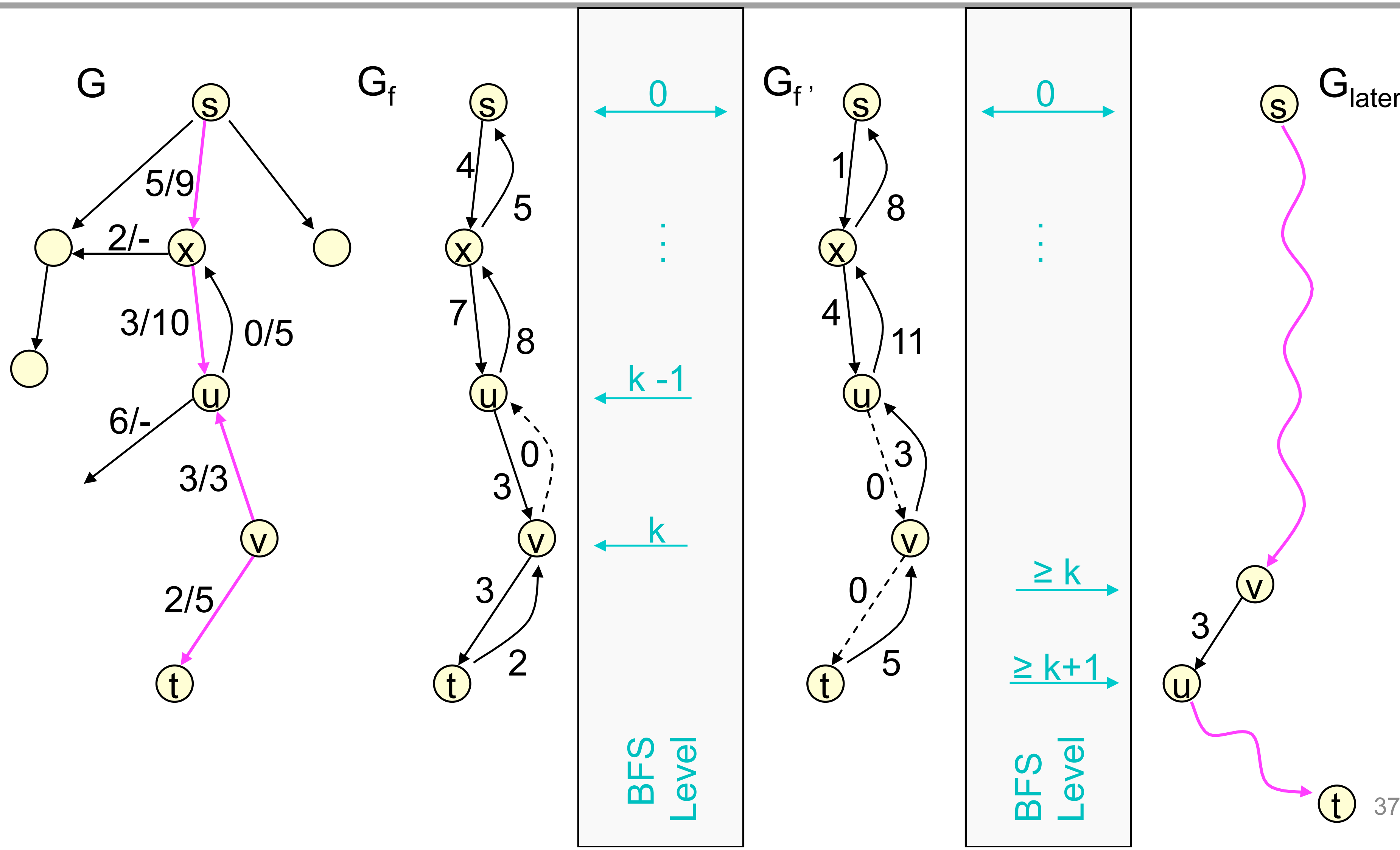
Proof:

{*u*,*v*} is critical on augmenting path *p* if it's closest to *s* having min residual capacity.

Won't be critical again until farther from *s*.

So each edge critical at most *n* times.

# Augmentation vs BFS Level

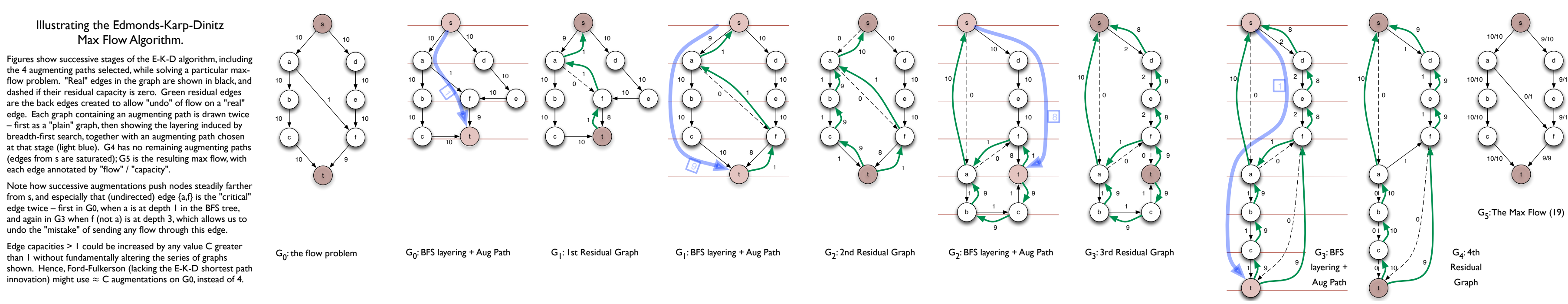


# Corollary

Edmonds-Karp-Dinitz runs in $O(nm^2)$

# Example

See "[Edmonds-Karp-Dinitz Example]" on course web page



Illustrating the Edmonds-Karp-Dinitz Max Flow Algorithm.

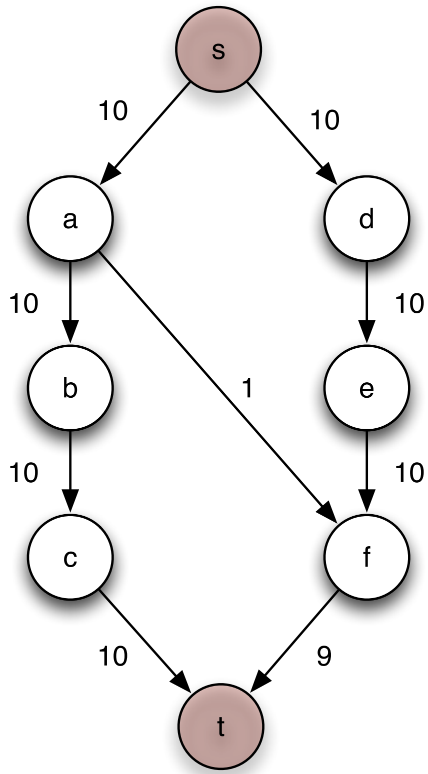
Figures show successive stages of the E-K-D algorithm, including the 4 augmenting paths selected, while solving a particular max-flow problem. "Real" edges in the graph are shown in black, and dashed if their residual capacity is zero. Green residual edges are the back edges created to allow "undo" of flow on a "real" edge. Each graph containing an augmenting path is drawn twice – first as a "plain" graph, then showing the layering induced by breadth-first search, together with an augmenting path chosen at that stage (light blue). G4 has no remaining augmenting paths (edges from s are saturated); G5 is the resulting max flow, with each edge annotated by "flow" / "capacity".

Note how successive augmentations push nodes steadily farther from s, and especially that (undirected) edge {a,f} is the "critical" edge twice – first in G0, when a is at depth 1 in the BFS tree, and again in G3 when f (not a) is at depth 3, which allows us to undo the "mistake" of sending any flow through this edge.

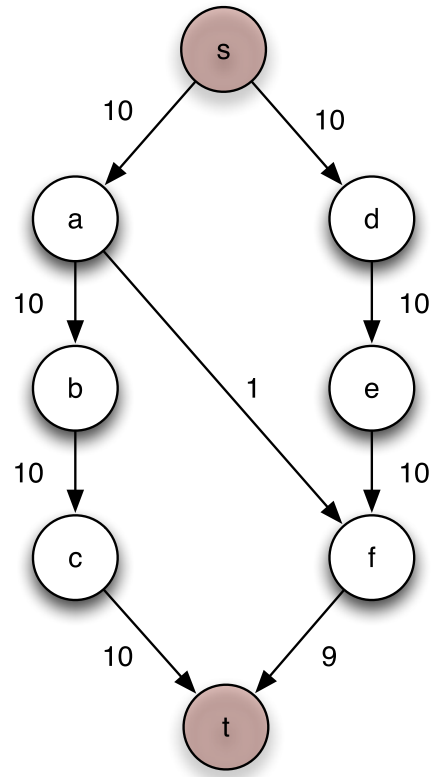
Edge capacities > 1 could be increased by any value C greater than 1 without fundamentally altering the series of graphs shown. Hence, Ford-Fulkerson (lacking the E-K-D shortest path innovation) might use ≈ C augmentations on G0, instead of 4.



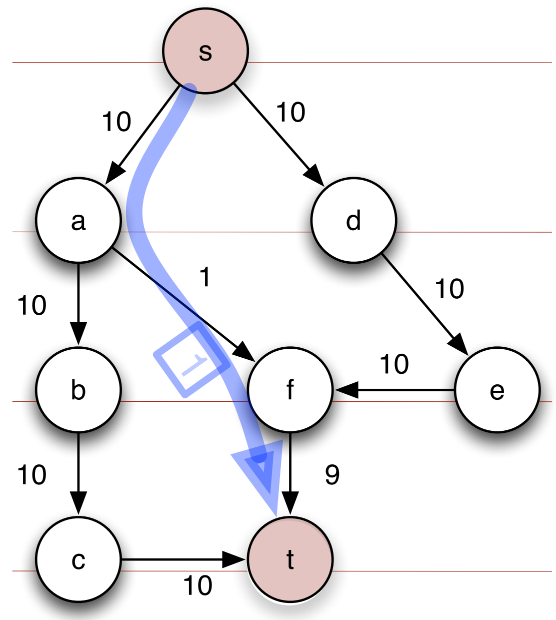
(file)



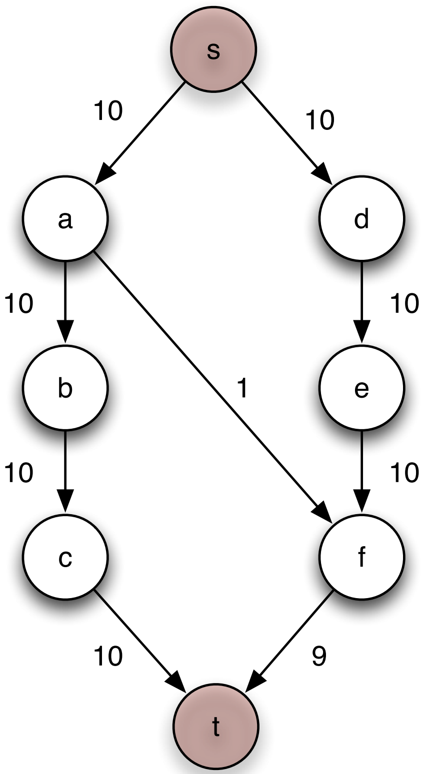

$G_0$: the flow problem



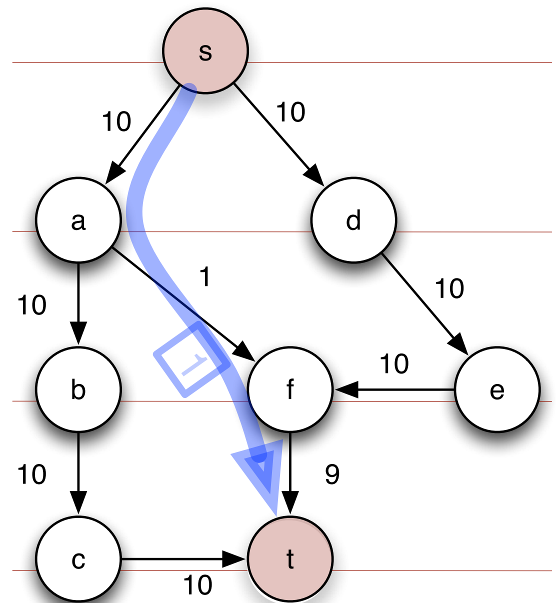

$G_0$: the flow problem



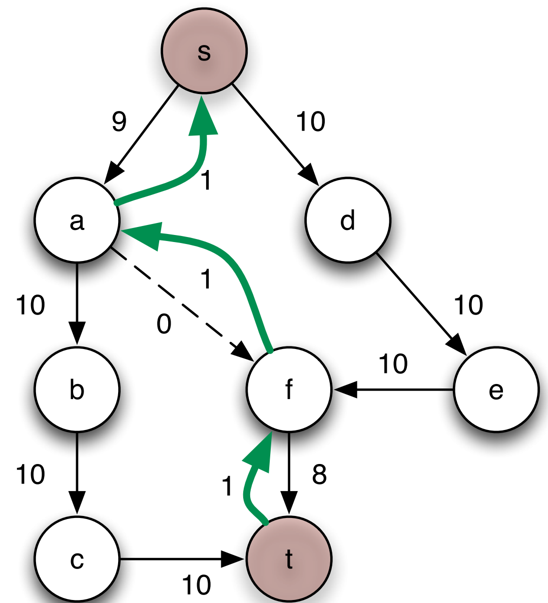

$G_0$: BFS layering + Aug Path



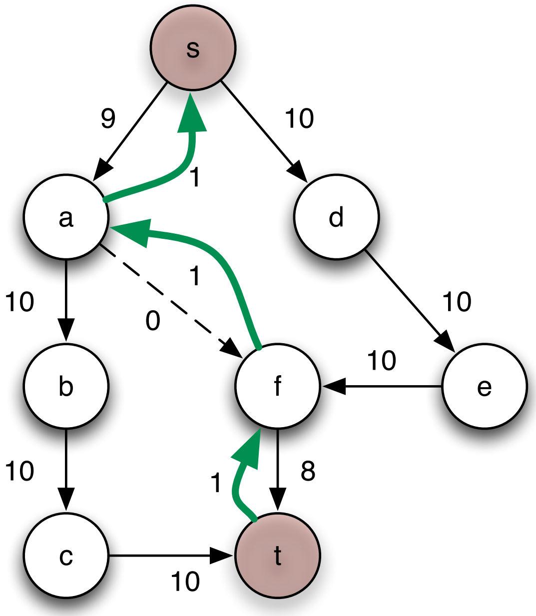

$G_0$: the flow problem



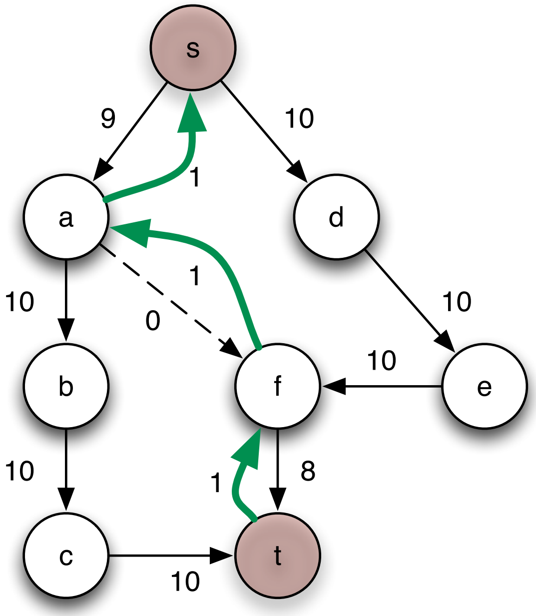

$G_0$: BFS layering + Aug Path



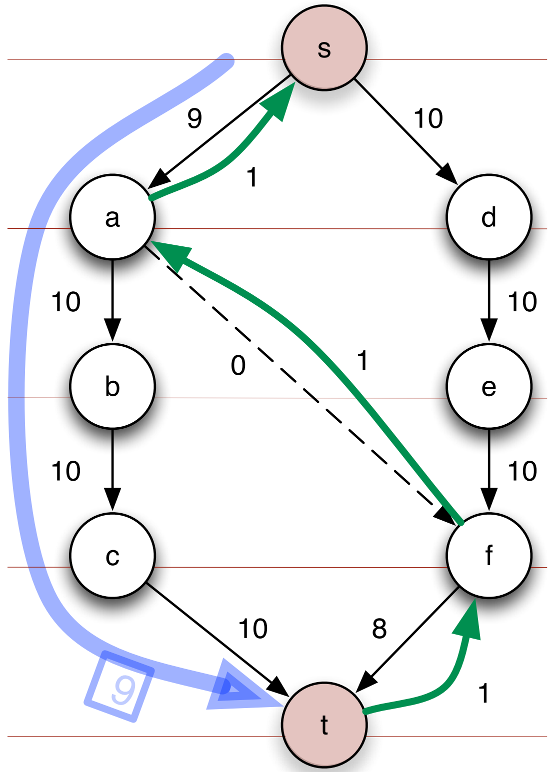

$G_1$: 1st Residual Graph



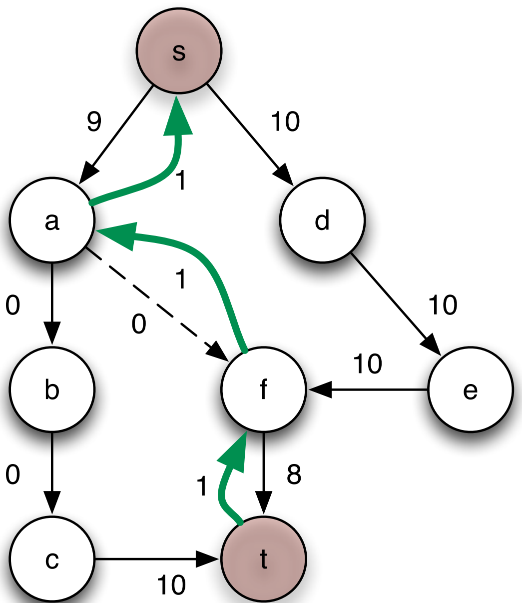

$G_1$: 1st Residual Graph



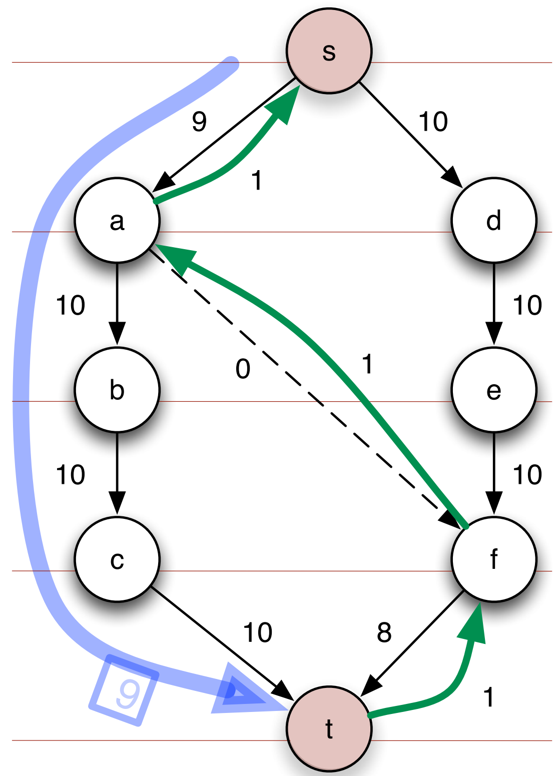

$G_1$: 1st Residual Graph



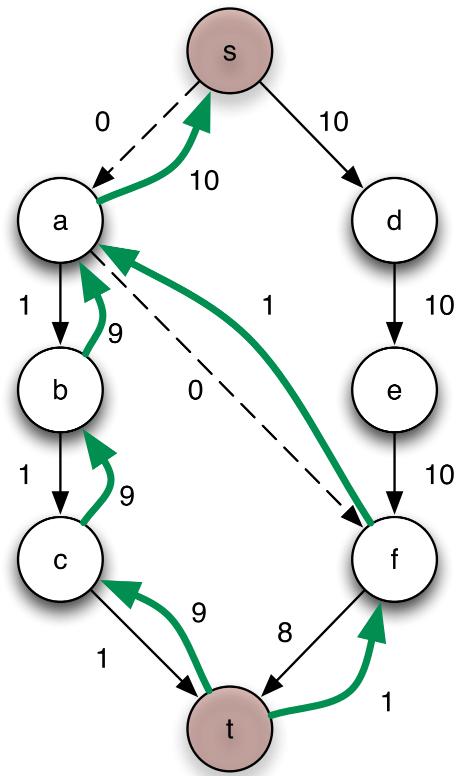

$G_1$: BFS layering + Aug Path



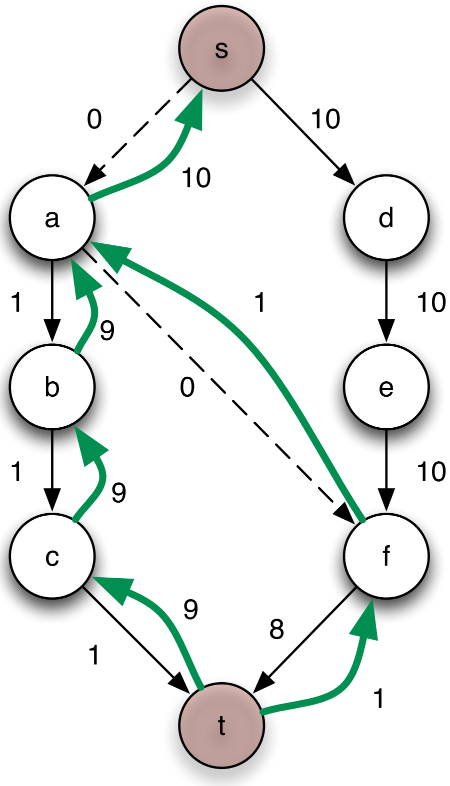

$G_1$: 1st Residual Graph



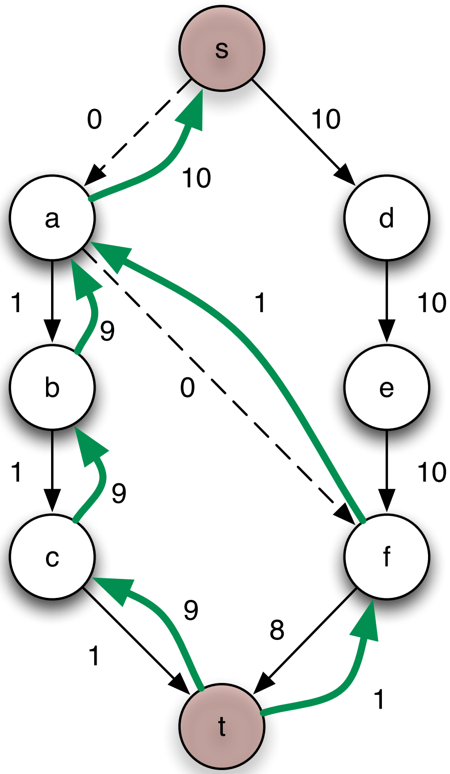

$G_1$: BFS layering + Aug Path



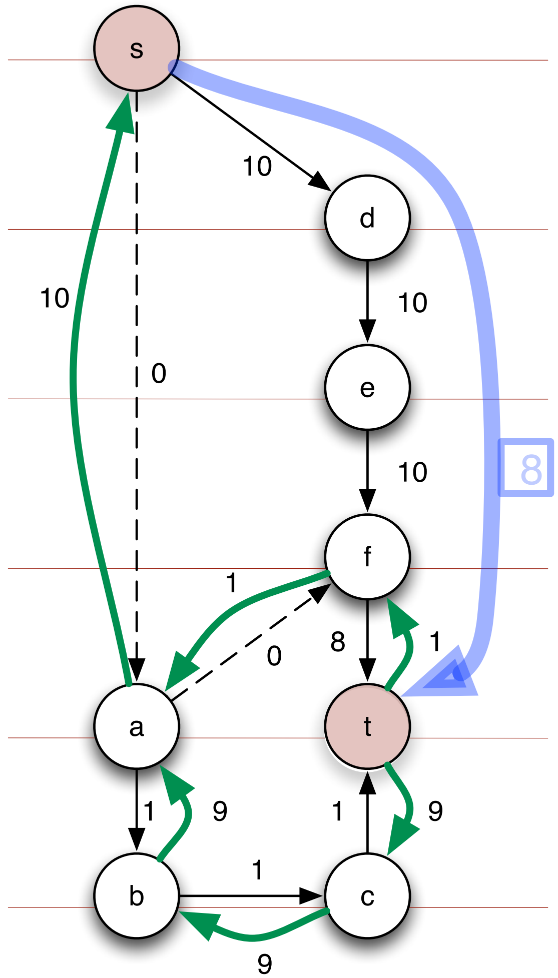

$G_2$: 2nd Residual Graph



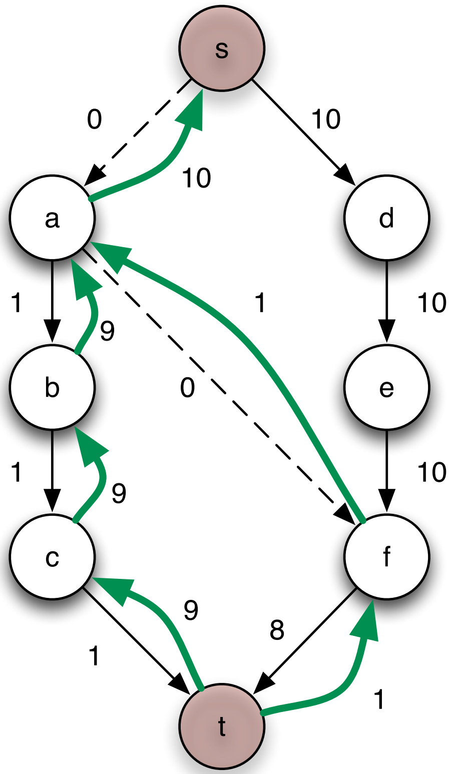

$G_2$: 2nd Residual Graph

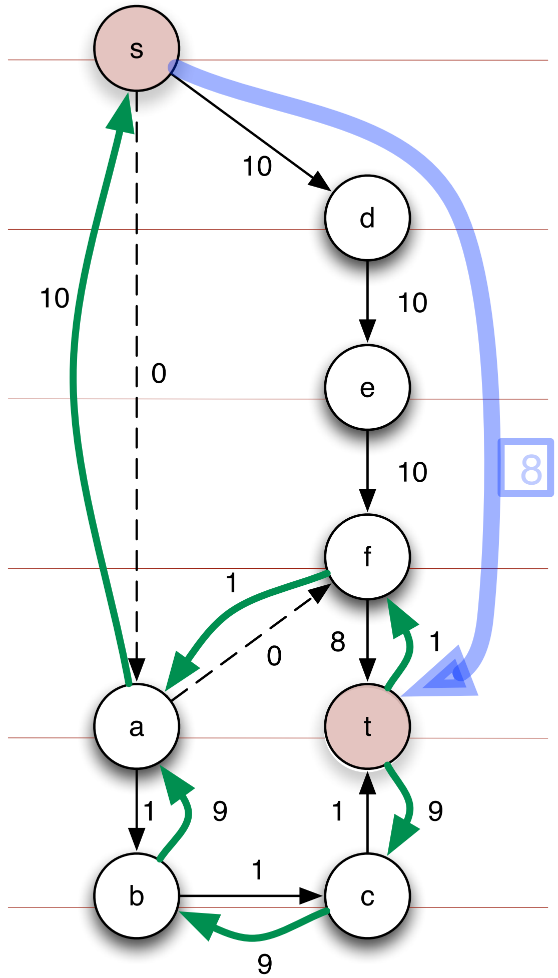


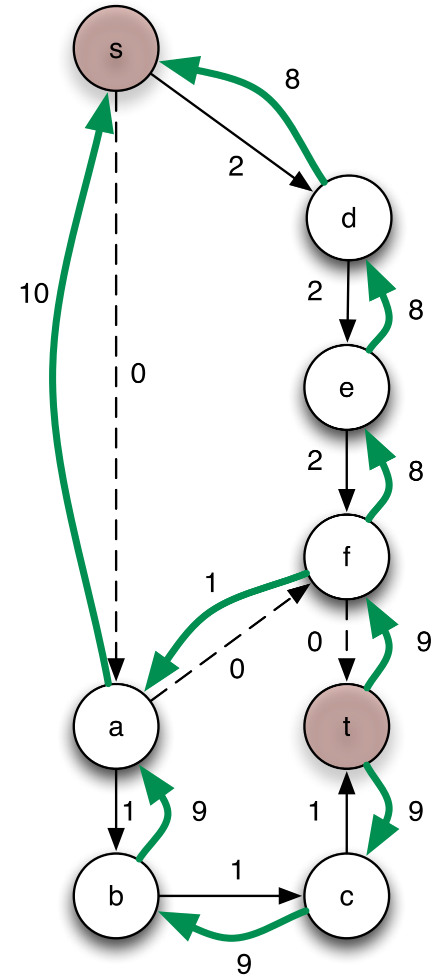

$G_2$: 2nd Residual Graph




$G_2$: BFS layering + Aug Path



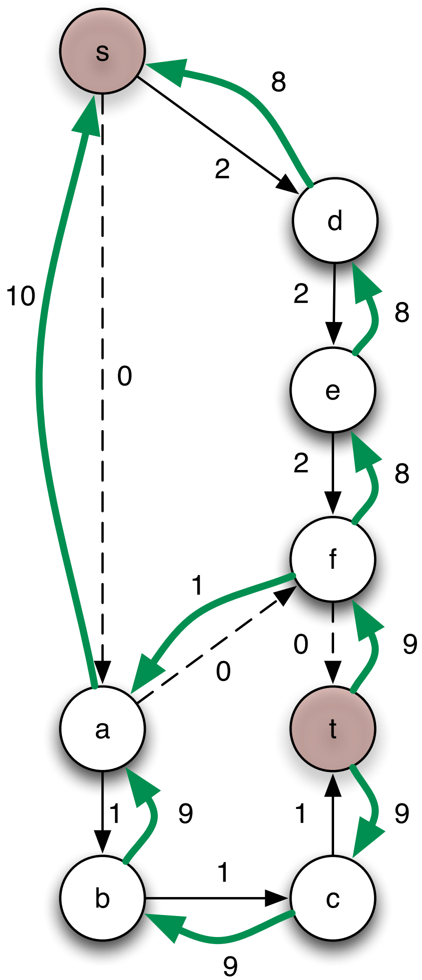



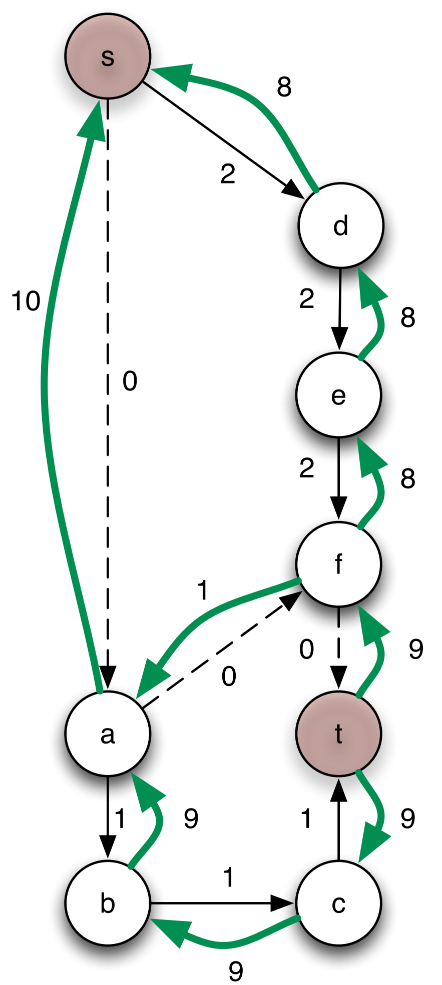

$G_2$: 2nd Residual Graph

$G_2$: BFS layering + Aug Path

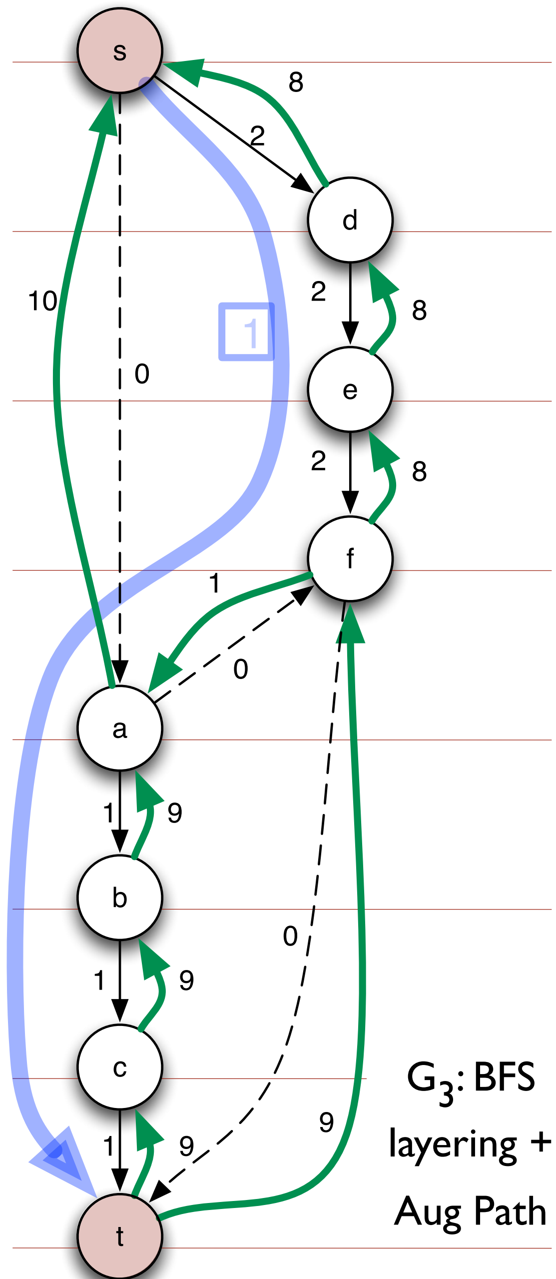
$G_3$: 3rd Residual Graph



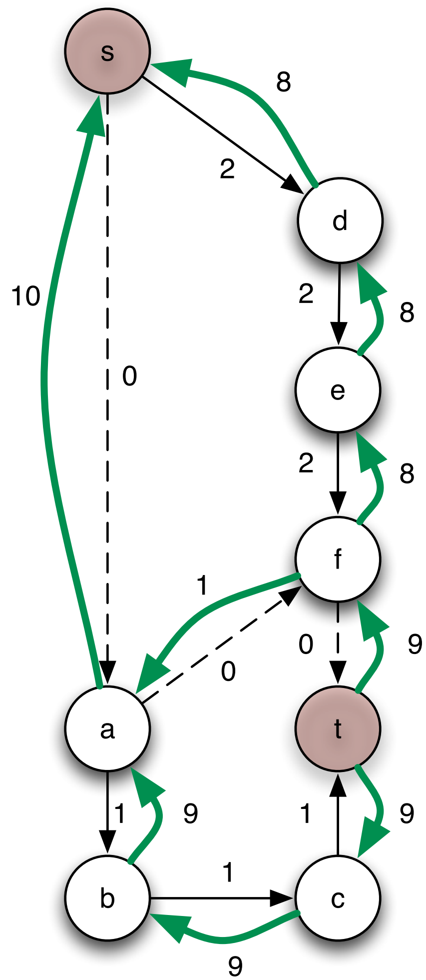

$G_3$: 3rd Residual Graph



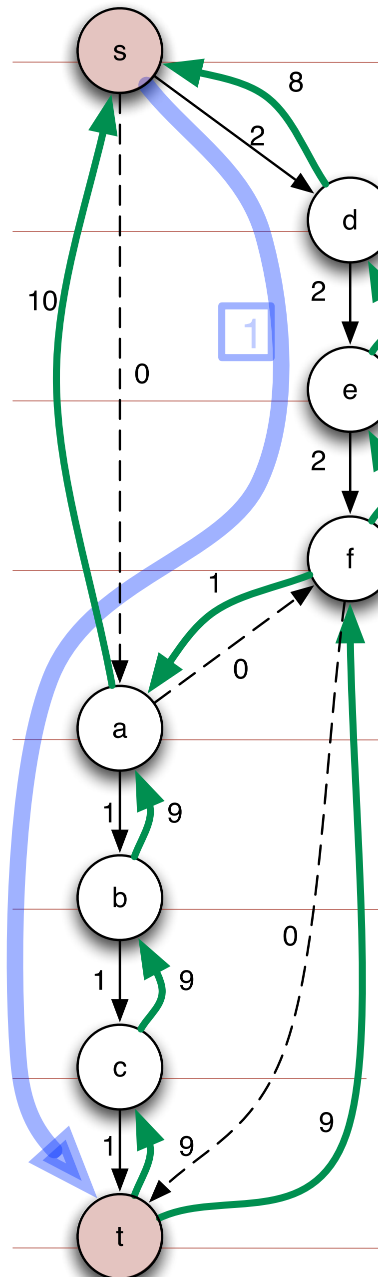

$G_3$: 3rd Residual Graph



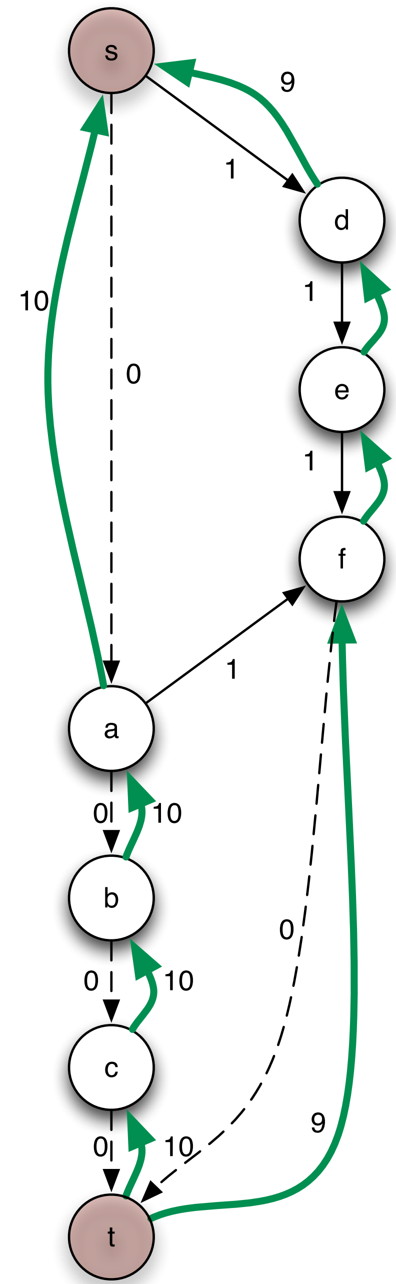

$G_3$: BFS layering + Aug Path



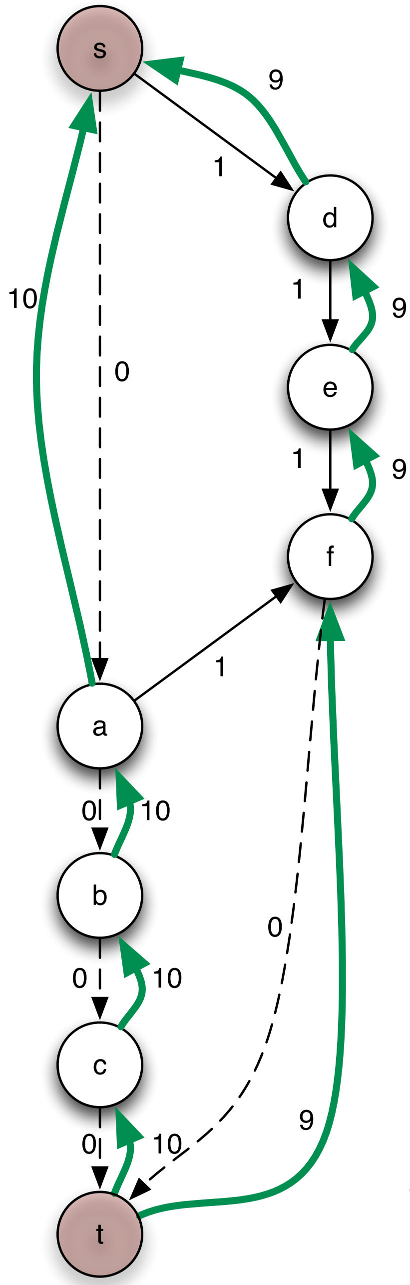

$G_3$: 3rd Residual Graph



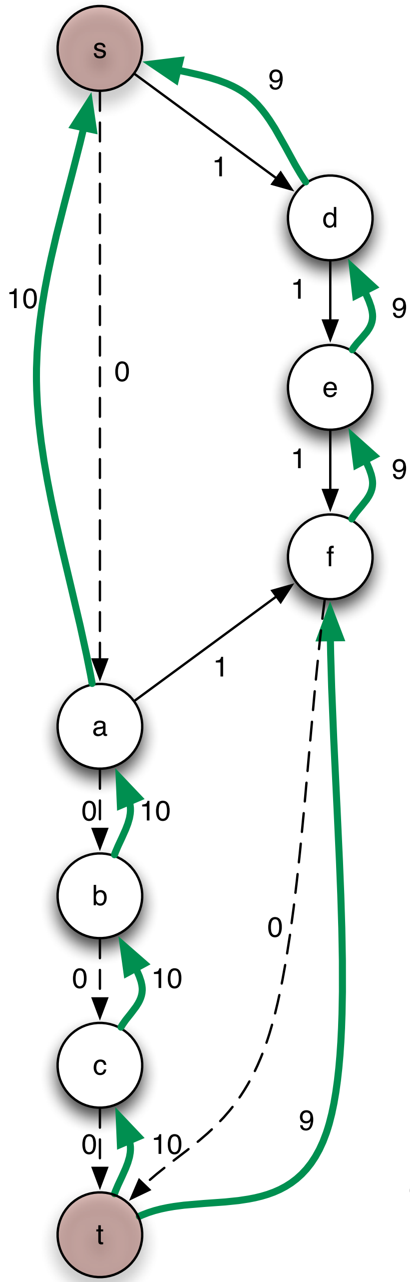

$G_3$: BFS layering + Aug Path



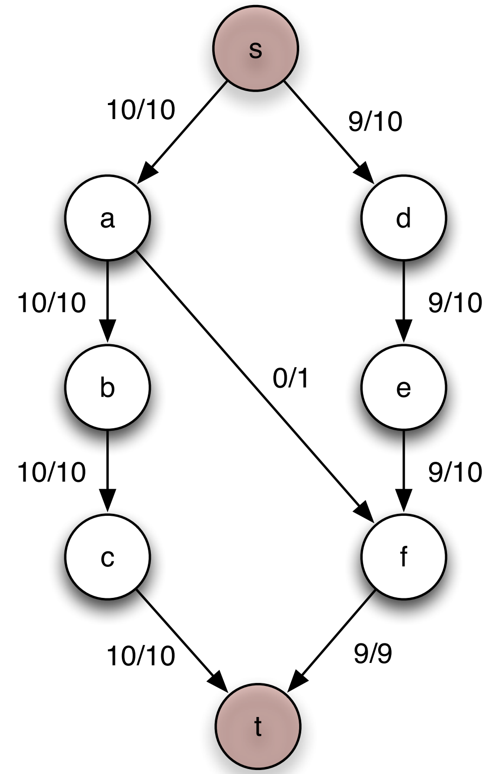

$G_4$: $4^{th}$ residual graph




$G_4$: $4^{th}$
residual
graph




$G_4$: $4^{th}$ residual graph




$G_5$:The Max Flow (19)

# Flow Applications

# Applications of Max Flow

Many!

Most look nothing like flow, at least superficially, but are deeply connected

Several interesting examples in 7.5-7.13

(7.8-7.11, 7.13 are optional, but interesting. Airline scheduling and image segmentation are especially recommended.)

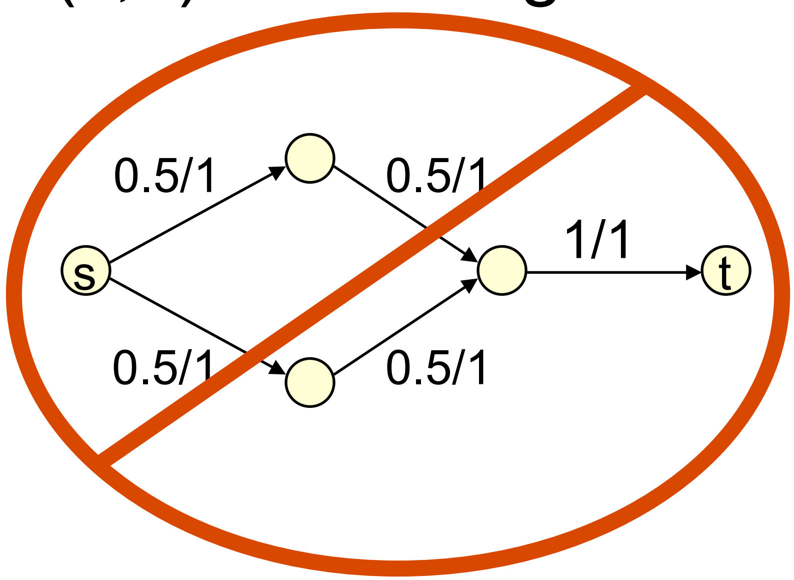
A few more in following slides

# Flow Integrality Theorem

Useful facts: If all capacities are integers

» Some max flow has an integer value

» Ford-Fulkerson method finds a max flow in which $f(u,v)$ is an integer for all edges $(u,v)$




A valid flow,
but unnecessary

# 7.6: Disjoint Paths

Given a digraph with designated nodes *s,t*, are there *k* edge-disjoint paths from *s* to *t* ?

You might try depth-first search; you might fail…

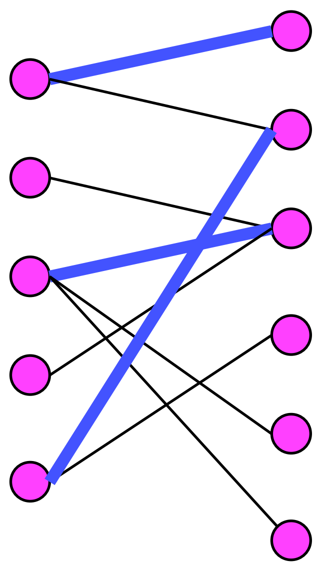
Instead:"edge caps=1, is max flow ≥ *k*?" Success!

Max-flow/min-cut also implies max number of edge disjoint paths = min number of edges whose removal separates *s* from *t*.

Many variants: node-disjoint, undirected, …

See 7.6

# 7.5: Bipartite Maximum Matching



Bipartite Graphs:

$G = (V,E)$

$V = L \cup R \;\; (L \cap R = \emptyset)$

$E \subseteq L \times R$

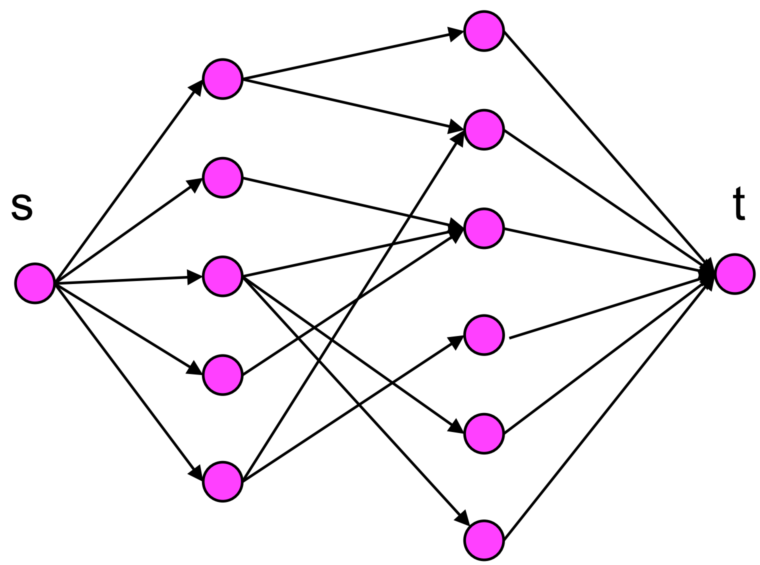
Matching:

A set of edges $M \subseteq E$ such that no two edges touch a common vertex

Problem:

Find a max size matching $M$

# Reducing Matching to Flow



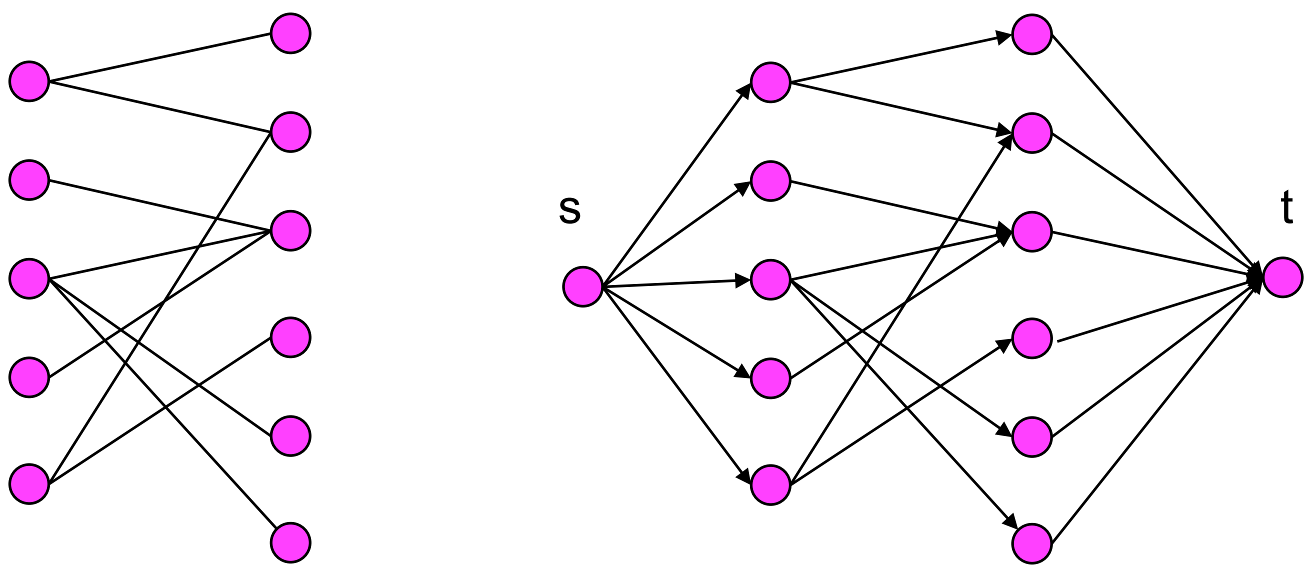

Given bipartite $G$, build flow network $N$ as follows:

- Add source $s$, sink $t$
- Add edges $s \rightarrow L$
- Add edges $R \rightarrow t$
- All edge capacities 1

**Theorem:**
Max flow iff
max matching

# Reducing Matching to Flow

**Theorem:** Max matching size = max flow value




$M \rightarrow f$ ? Easy – send flow only through $M$

$f \rightarrow M$ ? Flow Integrality Thm, + cap constraints

# Notes on Matching

Max Flow Algorithm is probably overly general here

But most direct matching algorithms use "augmenting path"-type ideas similar to that in max flow – See text (& homework?)

Time $mn^{1/2}$ possible via Edmonds-Karp

# 7.12 Baseball Elimination

Some slides by Kevin Wayne

# Baseball Elimination

| Team $i$ | Wins $w_i$ | Losses $l_i$ | To play $g_i$ | Against = $g_{ij}$ | | | |
|---|---|---|---|---|---|---|---|
| | | | | Atl | Phi | NY | Mon |
| Atlanta | 83 | 71 | 8 | - | 1 | 6 | 1 |
| Philly | 80 | 79 | 3 | 1 | - | 0 | 2 |
| New York | 78 | 78 | 6 | 6 | 0 | - | 0 |
| Montreal | 77 | 82 | 3 | 1 | 2 | 0 | - |

Which teams have a chance of finishing the season with most wins?

- » Montreal eliminated since it can finish with at most 80 wins, but Atlanta already has 83.
- » $w_i + g_i < w_j \;\Rightarrow$ team *i* eliminated.
- » Only reason sports writers appear to be aware of.
- » Sufficient, but not necessary!

# Baseball Elimination

| Team $i$ | Wins $w_i$ | Losses $l_i$ | To play $g_i$ | Against = $g_{ij}$ | | | |
|---|---|---|---|---|---|---|---|
| | | | | Atl | Phi | NY | Mon |
| Atlanta | 83 | 71 | 8 | - | 1 | 6 | 1 |
| Philly | 80 | 79 | 3 | 1 | - | 0 | 2 |
| New York | 78 | 78 | 6 | 6 | 0 | - | 0 |
| Montreal | 77 | 82 | 3 | 1 | 2 | 0 | - |

Which teams have a chance of finishing the season with most wins?

- » Philly can win 83, but still eliminated . . .
- » If Atlanta loses a game, then some other team wins one.

Remark. Depends on *both* how many games already won and left to play, *and* on which opponents.

# Baseball Elimination

Baseball elimination problem.

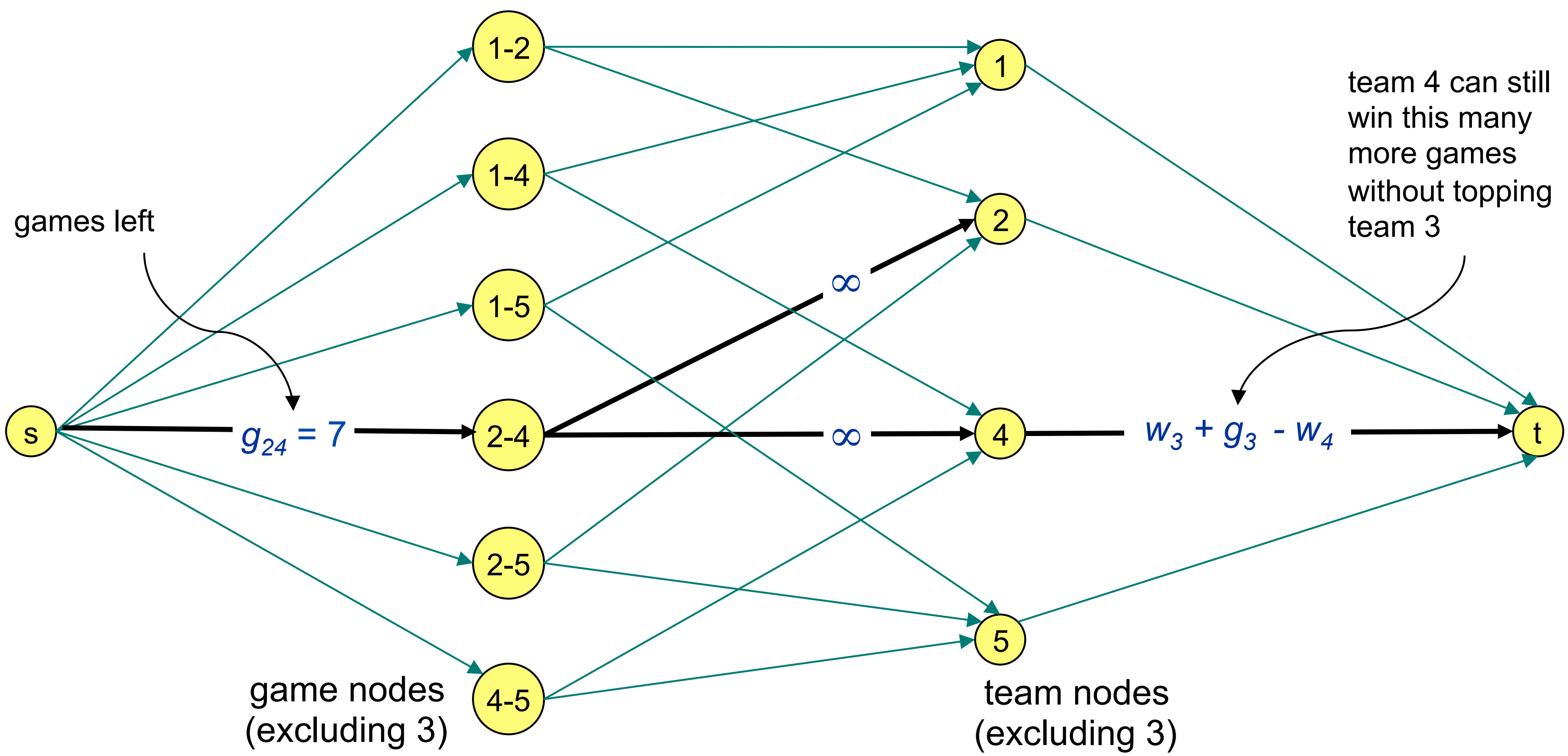
- Set of teams $S$.
- Distinguished team $s \in S$.
- Team $x$ has won $w_x$ games already.
- Teams $x$ and $y$ play each other $g_{xy}$ additional times.
- Is there any outcome of the remaining games in which team $s$ finishes with the most (or tied for the most) wins?

# Baseball Elimination: Max Flow Formulation

Can team 3 finish with most wins?

Assume team 3 wins all remaining games $\Rightarrow$ $w_3 + g_3$ wins.

Divvy remaining games so that all teams have $\leq$ $w_3 + g_3$ wins.

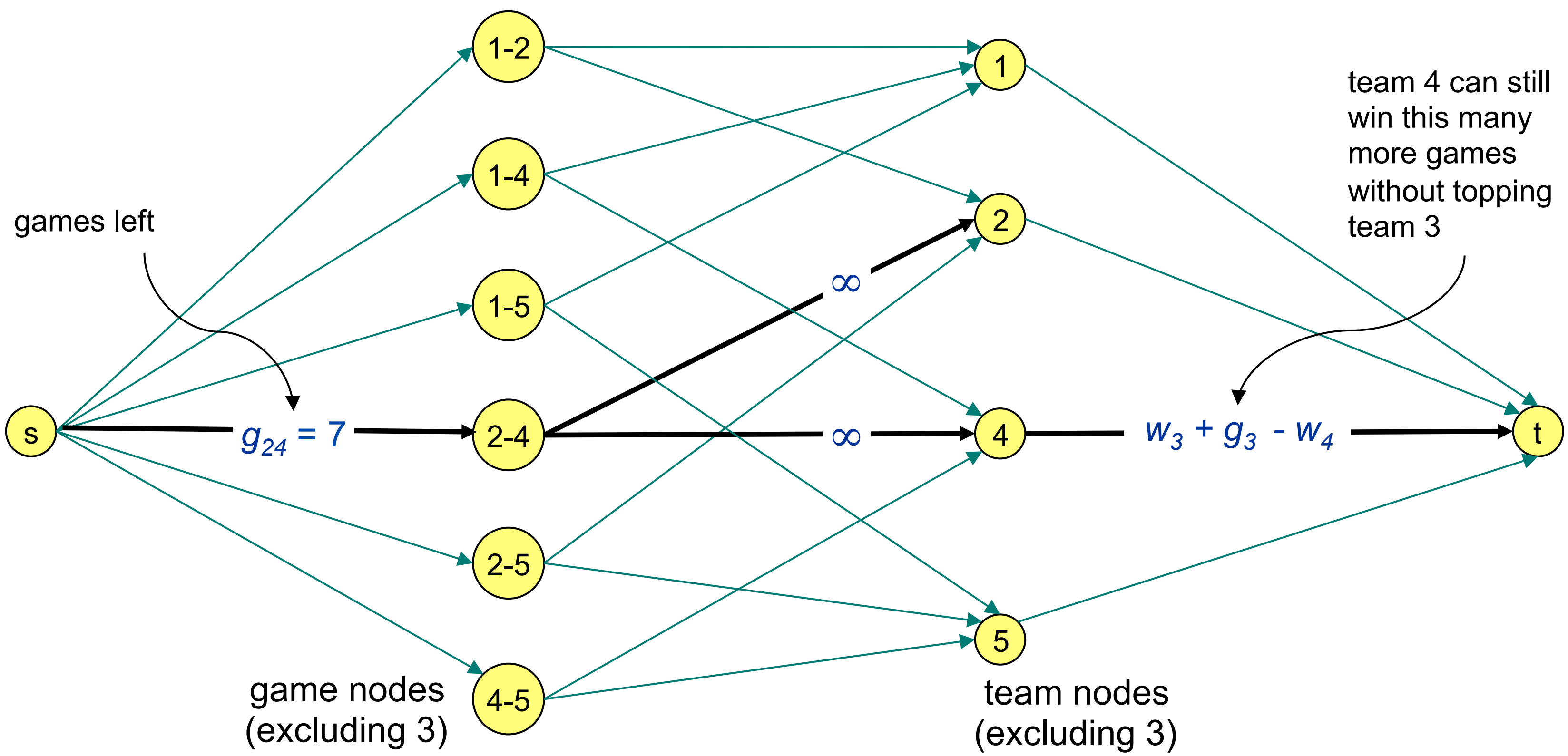


# Baseball Elimination: As Max Flow

Integrality ⇒ each remaining *x : y* game added to # wins for *x* or *y*.

Capacity on (*x, t*) edges ensure no team wins too many games.

In max flow, unsaturated source edge = unplayed game; if played, (either) winner would push ahead of team 3



# Baseball Elimination: Explanation for Sports Writers

| Team $i$ | Wins $w_i$ | Losses $l_i$ | To play $g_i$ | Against = $g_{ij}$ | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | NY | Bal | Bos | Tor | Det |
| NY | 75 | 59 | 28 | - | 3 | 8 | 7 | 3 |
| Baltimore | 71 | 63 | 28 | 3 | - | 2 | 7 | 4 |
| Boston | 69 | 66 | 27 | 8 | 2 | - | 0 | 0 |
| Toronto | 63 | 72 | 27 | 7 | 7 | 0 | - | - |
| Detroit | 49 | 86 | 27 | 3 | 4 | 0 | 0 | - |

AL East: August 30, 1996

Which teams have a chance of finishing the season with most wins?

Detroit could finish season with 49 + 27 = 76 wins.

# Baseball Elimination: Explanation for Sports Writers

| Team $i$ | Wins $w_i$ | Losses $l_i$ | To play $g_i$ | Against = $g_{ij}$ | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | NY | Bal | Bos | Tor | Det |
| NY | 75 | 59 | 28 | - | 3 | 8 | 7 | 3 |
| Baltimore | 71 | 63 | 28 | 3 | - | 2 | 7 | 4 |
| Boston | 69 | 66 | 27 | 8 | 2 | - | 0 | 0 |
| Toronto | 63 | 72 | 27 | 7 | 7 | 0 | - | - |
| Detroit | 49 | 86 | 27 | 3 | 4 | 0 | 0 | - |

AL East: August 30, 1996

Which teams could finish the season with most wins?

- Detroit could finish season with 49 + 27 = 76 wins.

Certificate of elimination. R = {NY, Bal, Bos, Tor}

- Have already won w(R) = 278 games.
- Must win at least r(R) = 27 more.
- Average team in R wins at least 305/4 > 76 games.

# Baseball Elimination: Explanation for Sports Writers

*Certificate of elimination*

$$T \subseteq S, \quad w(T) := \overbrace{\sum_{i \in T} w_i}^{\#\text{ wins}}, \quad g(T) := \overbrace{\sum_{\{x,y\} \subseteq T} g_{xy}}^{\#\text{ remaining games}},$$

If $\overbrace{\dfrac{w(T)+g(T)}{|T|}}^{\text{LB on avg \# games won}} > w_z + g_z$ then $z$ eliminated (by subset $T$).

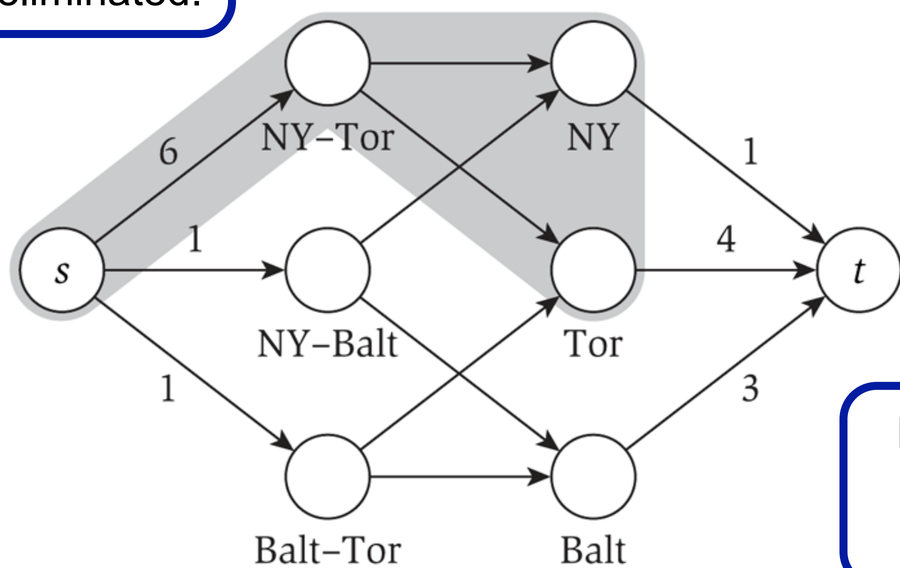
Theorem. [Hoffman-Rivlin 1967] Team $z$ is eliminated iff there exists a subset $T^*$ that eliminates $z$.

Proof idea. Let $T^*$ = teams on source side of min cut.

| | *w* | *l* | *g* | NY | Balt | Tor | Bos |
|---|---|---|---|---|---|---|---|
| NY | 90 | | 11 | - | 1 | 6 | 4 |
| Baltimore | 88 | | 6 | 1 | - | 1 | 4 |
| Toronto | 87 | | 10 | 6 | 1 | - | 4 |
| Boston | 79 | | 12 | 4 | 4 | 4 | - |

$g^* = 1+6+1 = 8$

( 90 + 87 + 6 ) / 2 > 91,
so the set T = {NY, Tor}
proves Boston is eliminated.



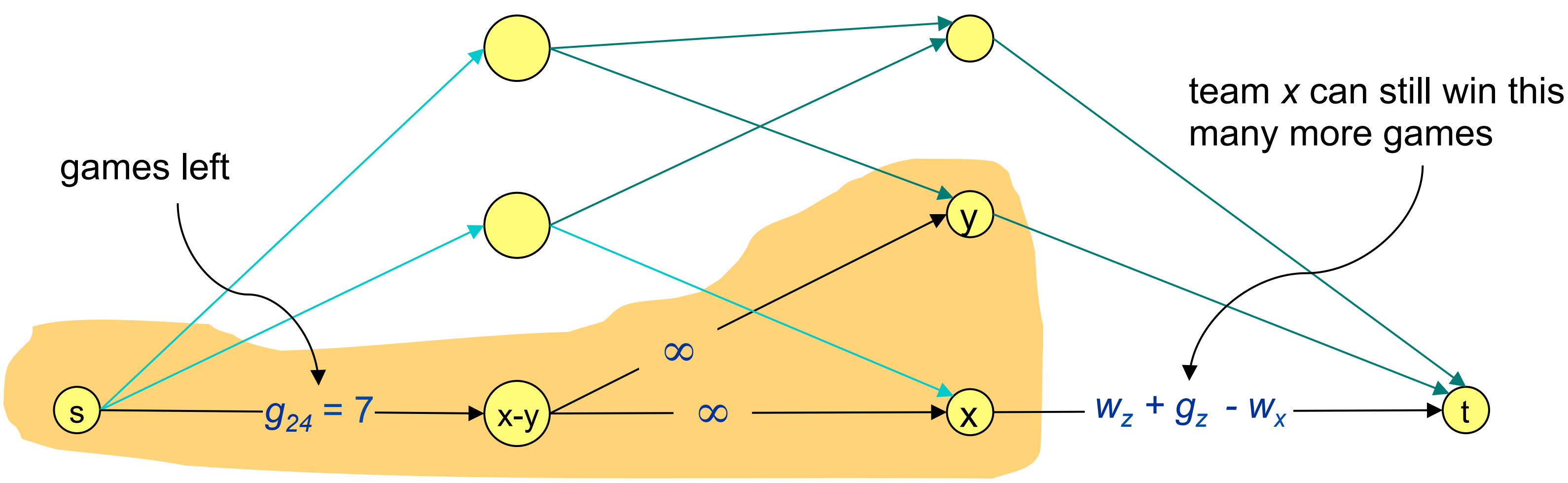

Note: *T* = {NY,Tor, Balt} is NOT a certificate, since (90+88+87+8)/3 = 91

Fig 7.21 Min cut ⇒ no flow of value $g^*$, so Boston eliminated.

# Baseball Elimination: Explanation for Sports Writers

Pf of theorem.

- Use max flow formulation, and consider min cut $(A, B)$.
- Define $T^*$ = team nodes on source side of min cut.
- Observe $x\text{-}y \in A$ iff both $x \in T^*$ and $y \in T^*$.
  - infinite capacity edges ensure if $x\text{-}y \in A$ then $x \in A$ and $y \in A$
  - if $x \in A$ and $y \in A$ but $x\text{-}y \notin T^*$, then adding $x\text{-}y$ to $A$ decreases capacity of cut



# Baseball Elimination: Explanation for Sports Writers

Pf of theorem.

Use max flow formulation, and consider min cut (*A, B*).

Define *T** = team nodes on source side of min cut.

Observe *x-y* $\in$ *A* iff both *x* $\in$ *T** and *y* $\in$ *T**.

$$g(S - \{z\}) \;>\; cap(A,\, B)$$

$$= \overbrace{g(S - \{z\}) - g(T^*)}^{\text{capacity of game edges leaving A}} + \overbrace{\sum_{x \in T^*} (w_z + g_z - w_x)}^{\text{capacity of team edges leaving A}}$$

$$= g(S - \{z\}) - g(T^*) - w(T^*) + |T^*|(w_z + g_z)$$

Rearranging:

$$w_z + g_z \;<\; \frac{w(T^*) + g(T^*)}{|T^*|}$$

# Matching & Baseball: Key Points

Can (sometimes) take problems that seemingly have *nothing* to do with flow & reduce them to a flow problem

How? Build a clever network; map allocation of *stuff* in original problem (match edges; wins) to allocation of *flow* in network. Clever edge capacities constrain solution to mimic original problem in some way. Integrality useful.

# Matching & Baseball: Key Points

Furthermore, in the baseball example, min cut can be translated into a succinct *certificate* or *proof* of some property that is much more transparent than "see, I ran max-flow and it says flow must be less than $g^*$".

These examples suggest why max flow is so important – *it's a very general tool used in many other algorithms*.