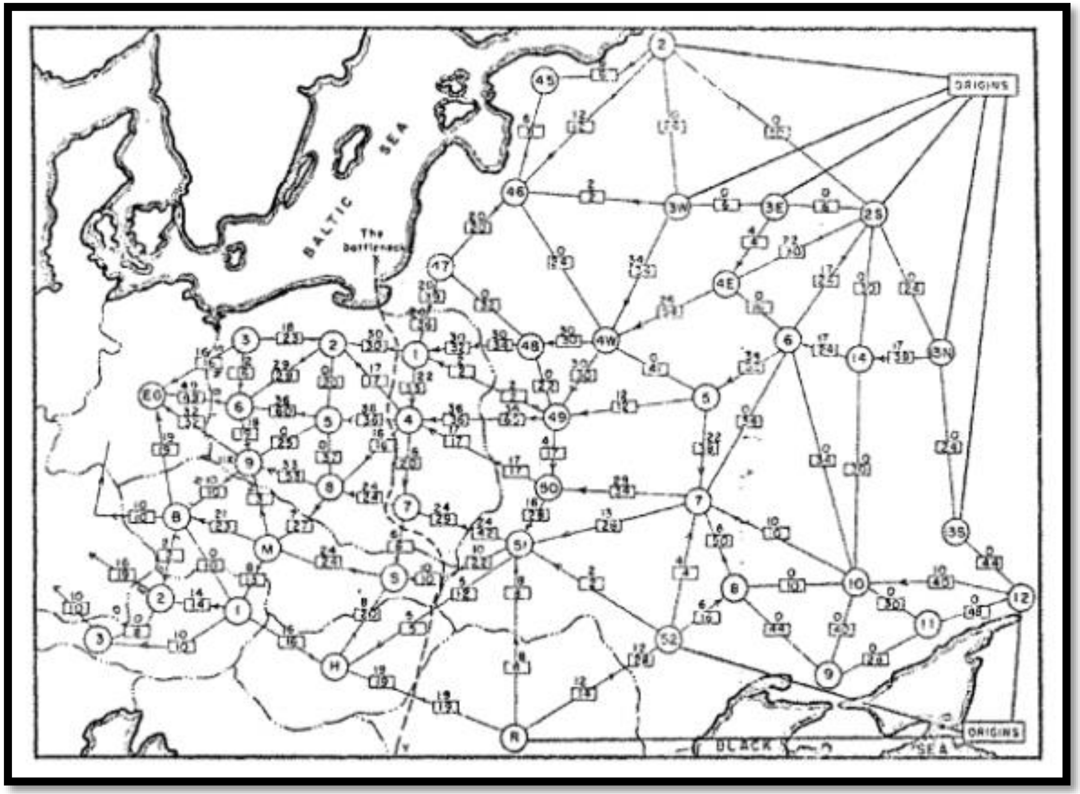
CSE 421: Algorithms

Winter 2014

Lecture 18: Network flow

Reading:
Sections 6.6-6.10

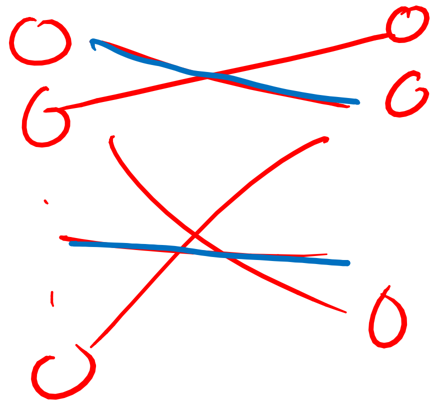


bipartite matching

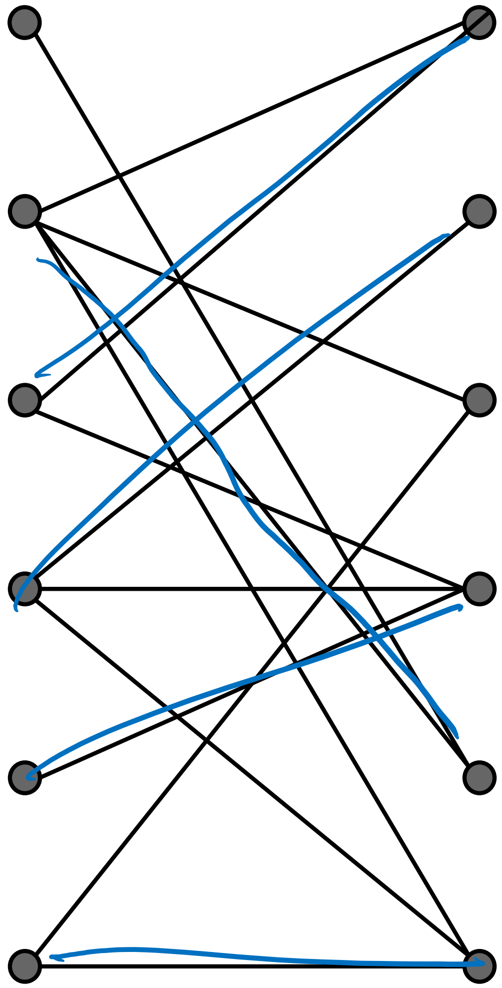
Given: A bipartite graph G=(V,E)

Def: M ⊆ E is a matching in G iff no two edges in M share a vertex

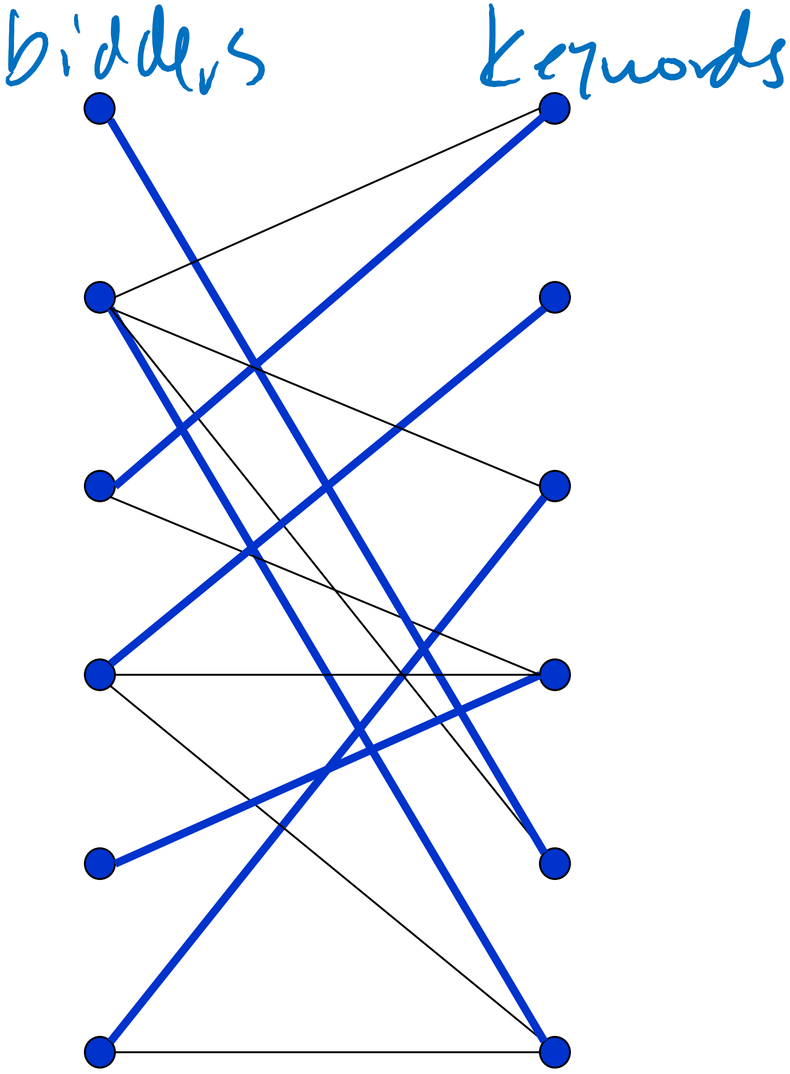
Goal: Find a matching M in G of maximum possible size



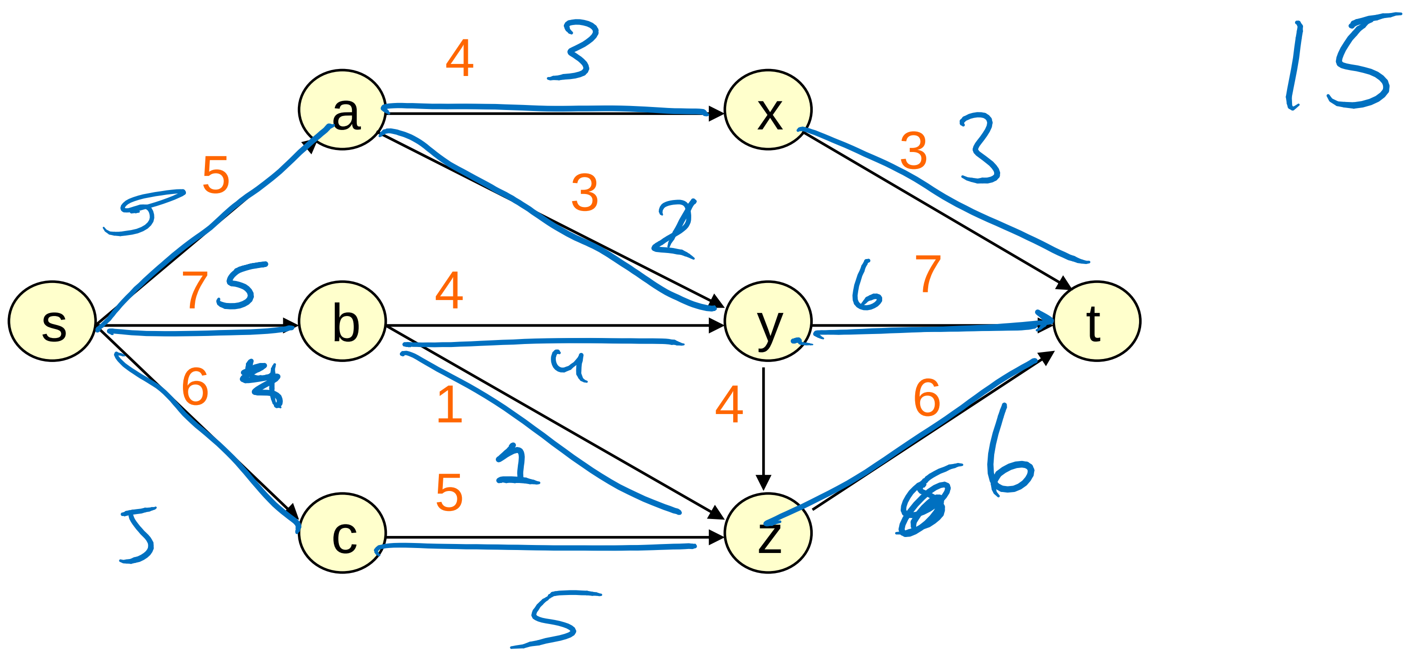
bipartite matching



bipartite matching

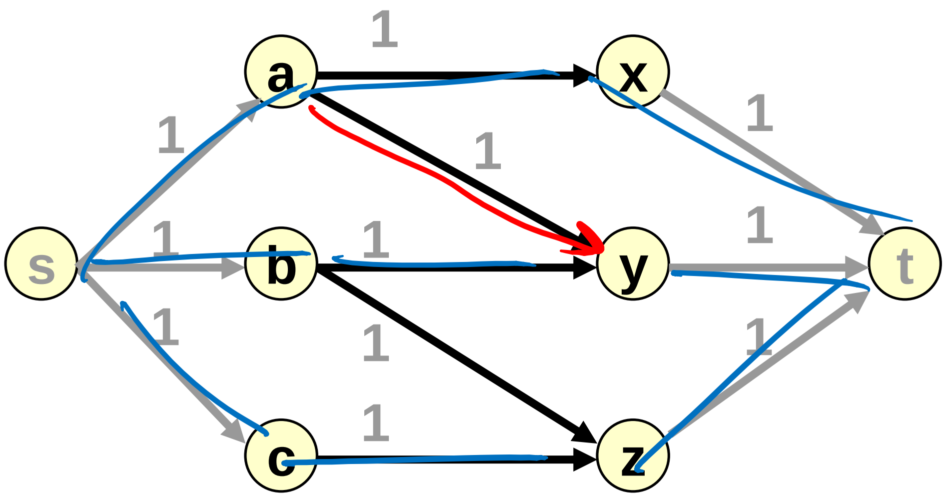


the network flow problem



How much stuff can flow from s to t ?

bipartite matching as a special case



bipartite matching as a special case

$f^{out}(s) \stackrel{?}{=} f^{in}(t)$

Given:

A digraph $G = (V,E)$

Two vertices s,t in V (source & 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* $f: E \to R$ s.t. for all u,v:

- $0 \leq f(u,v) \leq c(u,v)$

[Capacity Constraint]

- if $u \neq s,t$, we have $f^{out}(u)=f^{in}(u)$

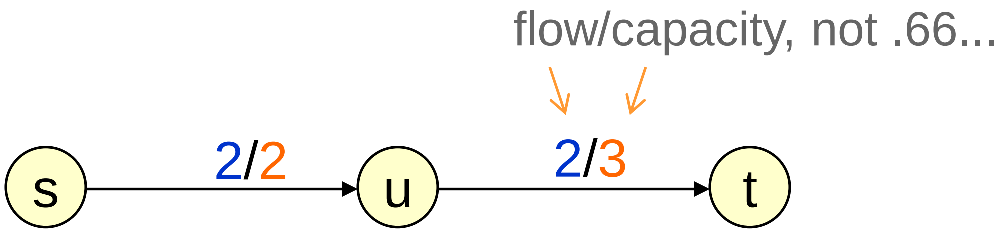
[Flow Conservation]

Maximizing total flow $n(f) = f^{out}(s)$

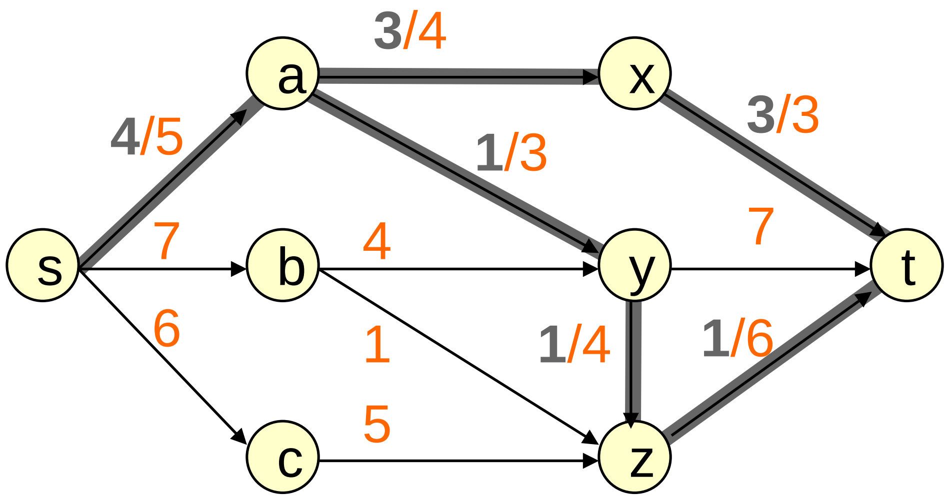
Notation:

$$f^{in}(v) = \sum_{e=(u,v)\in E} f(u,v) \qquad f^{out}(v) = \sum_{e=(v,w)\in E} f(v,w)$$

example: a flow function



example: a flow function



- Not shown: $f(u,v)$ if $= 0$
- Note: max flow ≥ 4 since
  f is a flow function, with $\nu(f) = 4$

greedy algorithm?

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

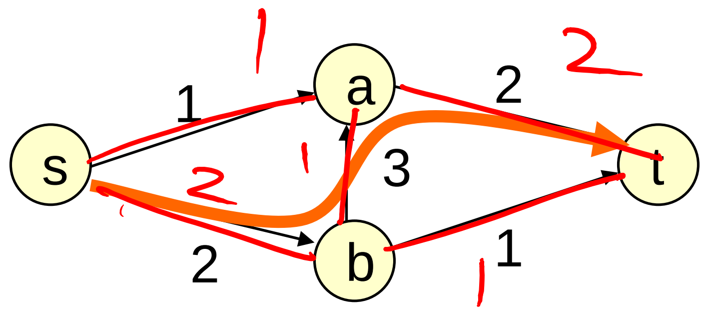
Pick such a path, p

Find c, the min capacity of any edge in p

Subtract c from all capacities on p

Delete edges of capacity 0

- This does NOT always find a max flow:



If pick $s \rightarrow b \rightarrow a \rightarrow t$ first, flow stuck at 2.

But flow 3 possible.

a brief history of flow

#	year	discoverer(s)	bound
1	1951	Dantzig	$O(n^2 mU)$
2	1955	Ford & Fulkerson	$O(nmU)$
3	1970	Dinitz Edmonds & Karp	$O(nm^2)$
4	1970	Dinitz	$O(n^2 m)$
5	1972	Edmonds & Karp Dinitz	$O(m^2 \log U)$
6	1973	Dinitz Gabow	$O(nm \log U)$
7	1974	Karzanov	$O(n^3)$
8	1977	Cherkassky	$O(n^2 \sqrt{m})$
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	$O(nm + n^2 \log U)$
13	1987	Ahuja et al.	$O(nm \log(n\sqrt{\log U}/(m+2)))$
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.	$O(nm + n^{2+\epsilon})$
18	1993	Phillips & Westbrook	$O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$
19	1994	King et al.	$O(nm \log_{m/(n \log n)} n)$
20	1997	Goldberg & Rao	$O(m^{3/2} \log(n^2/m) \log U)$ $O(n^{2/3} m \log(n^2/m) \log U)$

2012 Orlin + King et al. $O(nm)$

$O(n^3)$ . . .

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

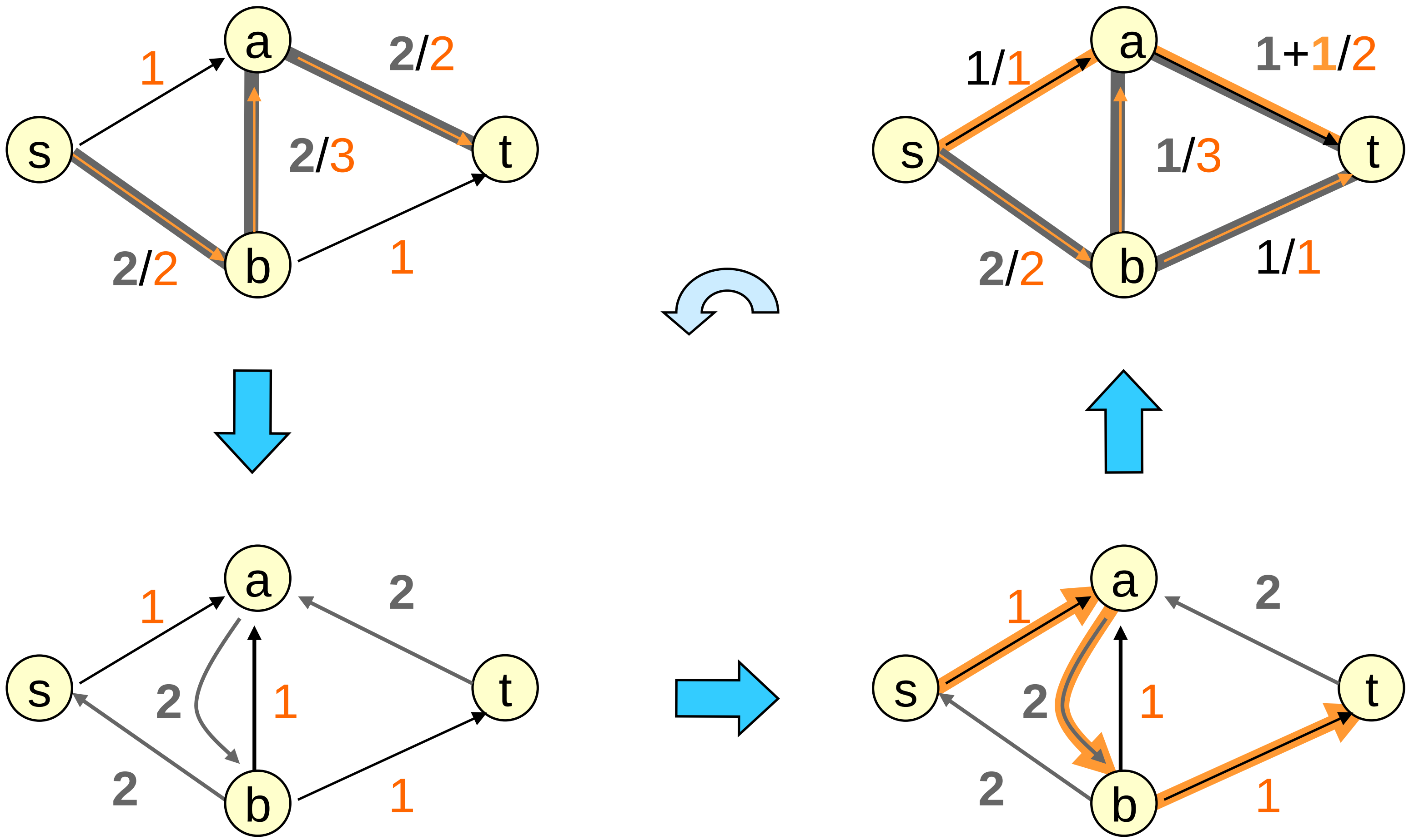
Source: Goldberg & Rao, FOCS '97

dream:

$O(m \log n)$

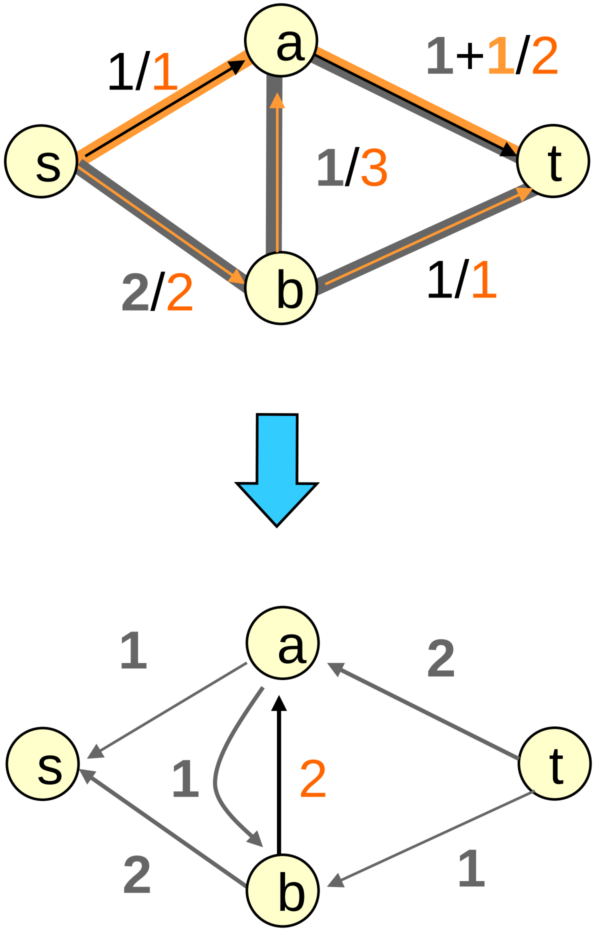
$O(m^{1.2})$

greed revisited: augmenting paths



Residual Graph

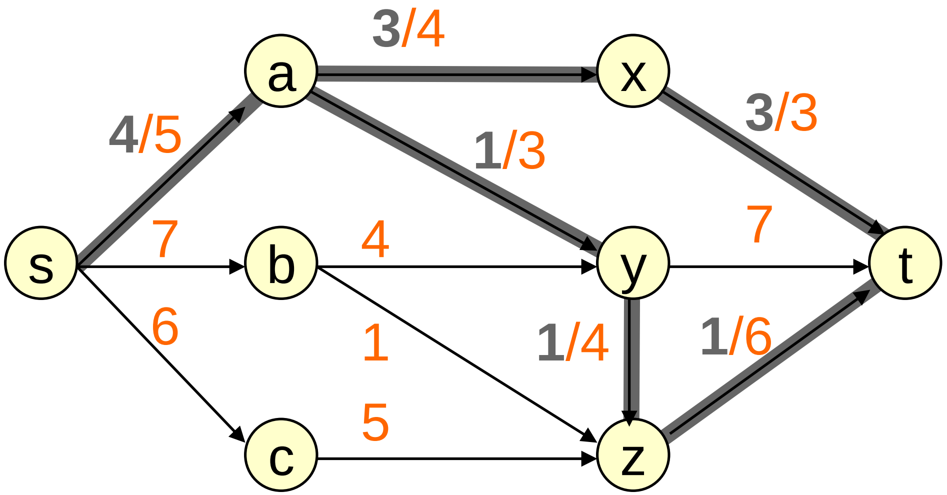
greed revisited: augmenting paths



New Residual Graph

residual capacity

- The residual capacity (w.r.t. f) of (u,v) is $c_f(u,v) = c(u,v) - f(u,v)$ if $f(u,v) \leq c(u,v)$ and $c_f(u,v)=f(v,u)$ if $f(v,u) > 0$



- e.g. $c_f(s,b)=7$; $c_f(a,x) = 1$; $c_f(x,a) = 3$

residual graph & augmenting paths

- The **residual graph** (w.r.t. **f**) is the graph

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

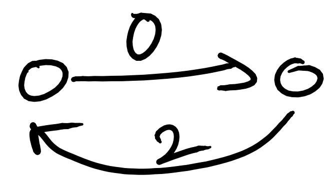
  – Two kinds of edges

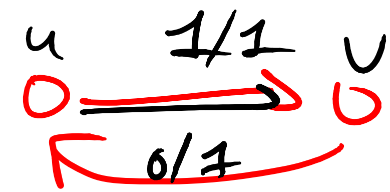
    Forward edges

      $f(u,v) < c(u,v)$ so $c_f(u,v) = c(u,v) - f(u,v) > 0$

    Backward edges

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

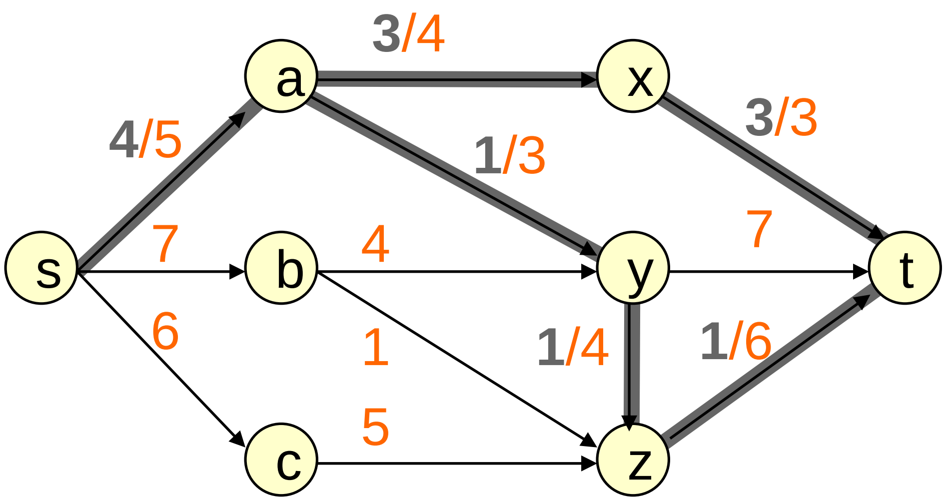


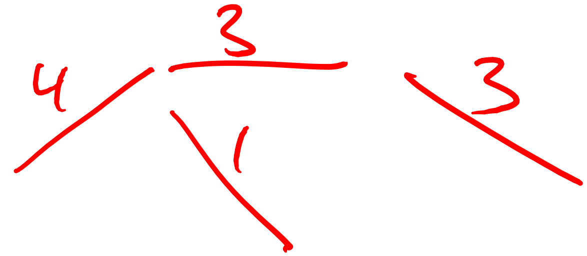


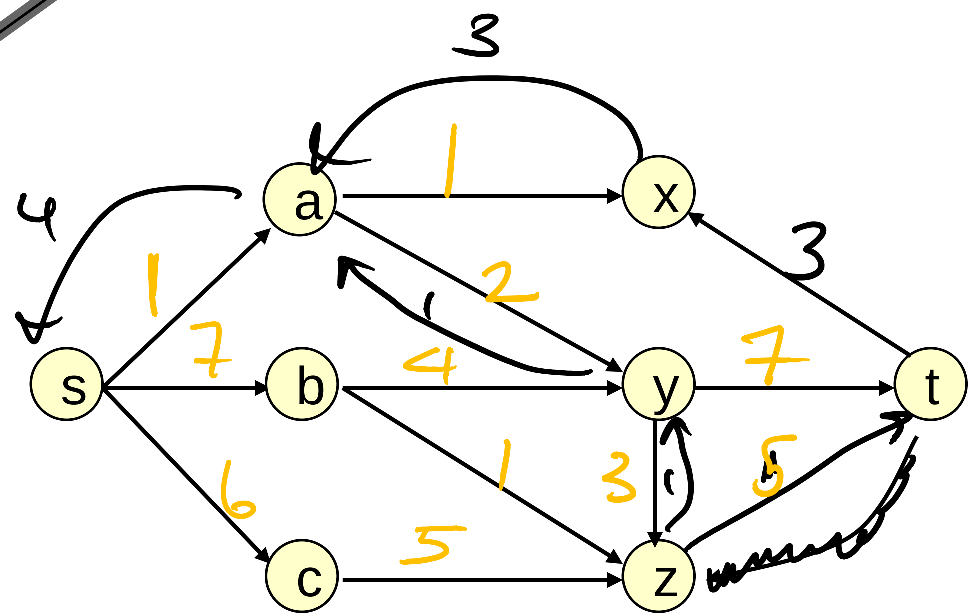
$c_f(v,u) = c(v,u) + f(v,u)$

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

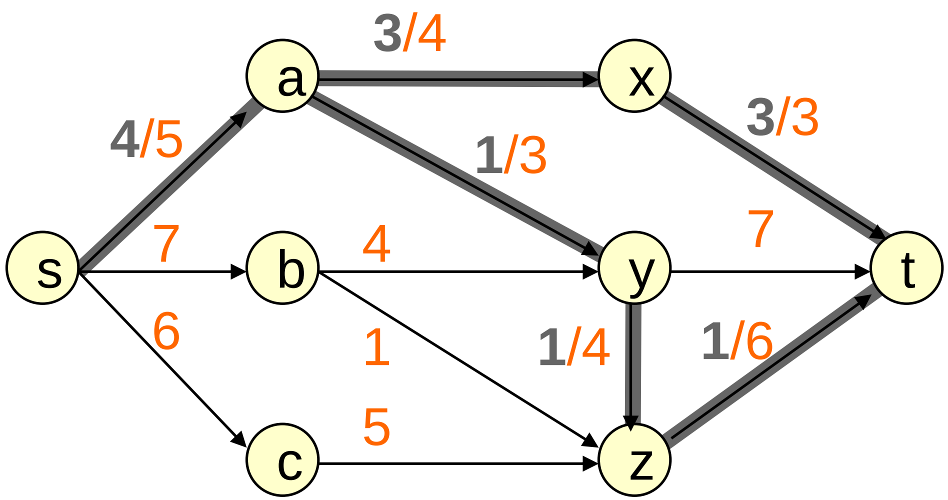
a residual network

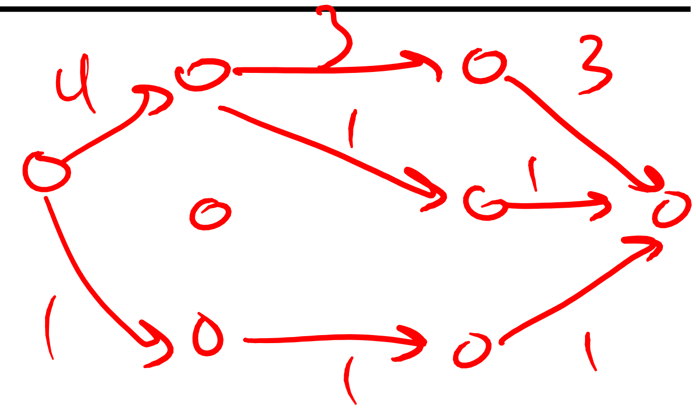


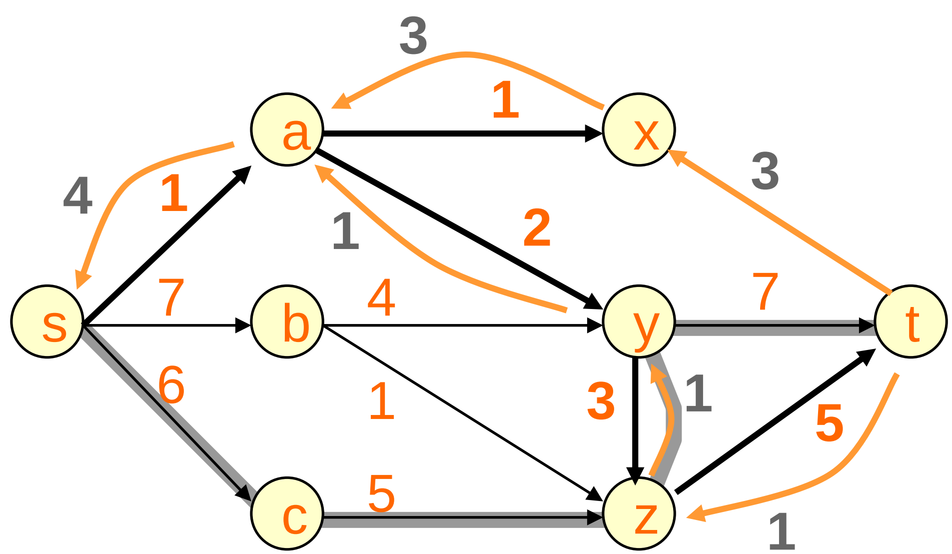




an augmenting path







augmenting a flow along a path

augment(**f**,**P**)

$c_P \leftarrow \min_{(u,v)\in P} c_f(u,v)$ "bottleneck(P)"

for each $e \in P$

if **e** is a forward edge then

increase **f**(**e**) by c_P

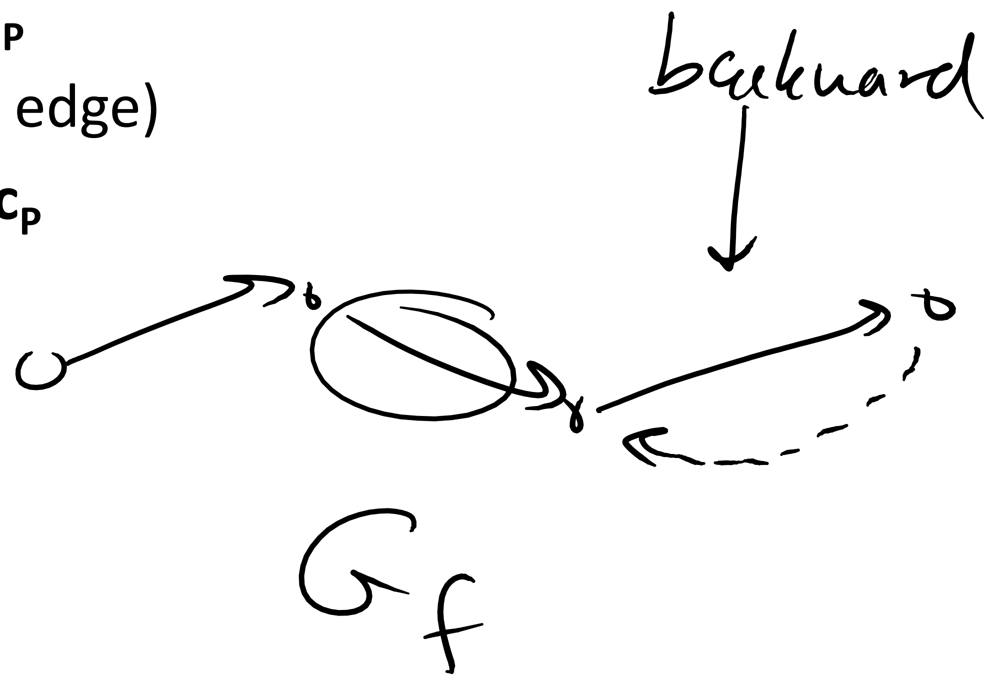
else (**e** is a backward edge)

decrease **f**(**e**) by c_P

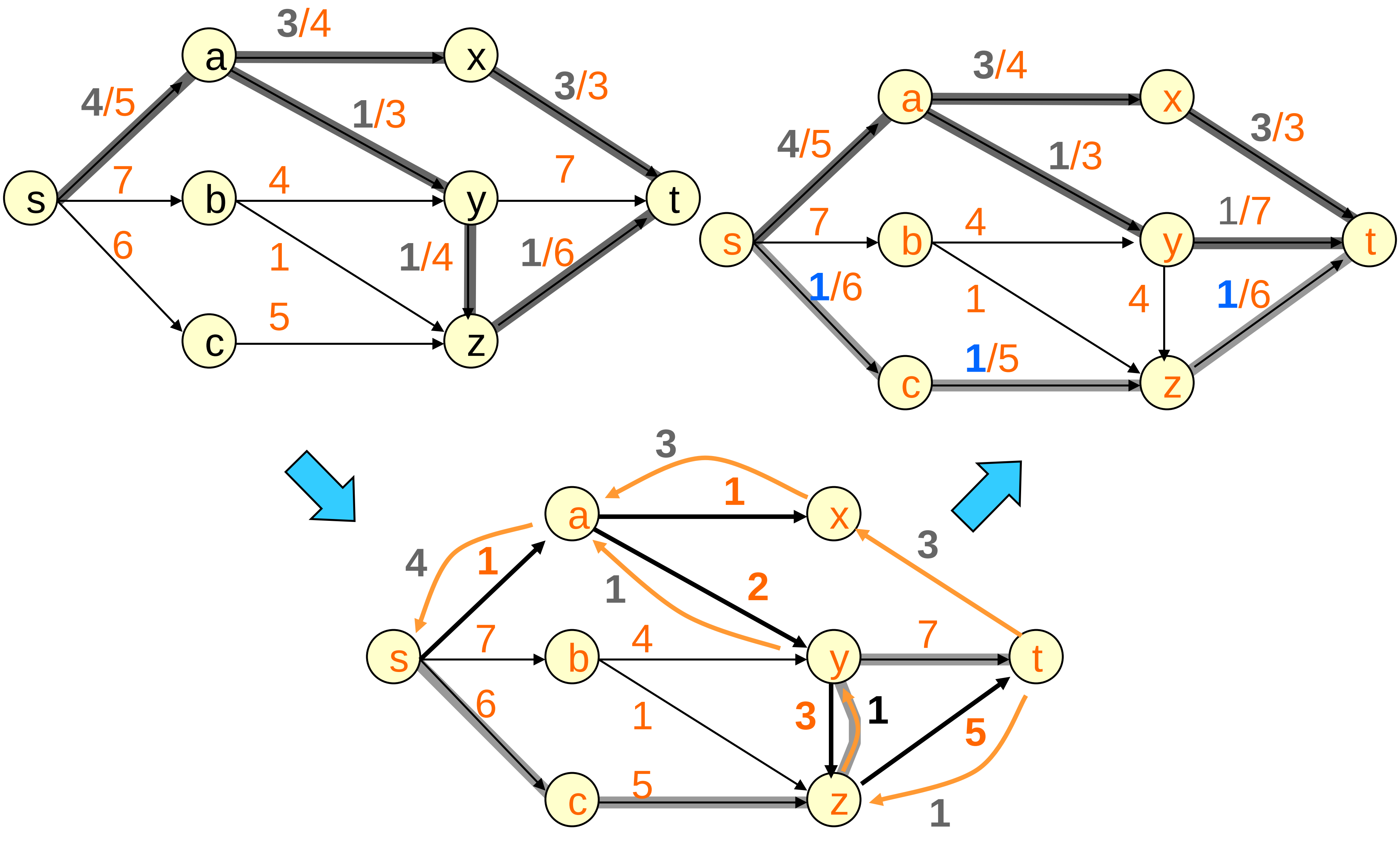
endif

endfor

return(**f**)



augmenting a flow



claim

If G_f has an augmenting path P, then the function f'=augment(f,P) is a legal flow.

Proof:

f' and f differ only on the edges of P so only need to consider such edges (u,v)

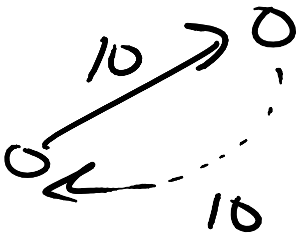
proof of claim

- If (u,v) is a forward edge then

$$f'(u,v) = f(u,v)+c_P \leq f(u,v)+c_f(u,v)$$
$$= f(u,v)+c(u,v)-f(u,v)$$
$$= c(u,v)$$

- If (u,v) is a backward edge then f and f' differ on flow along (v,u) instead of (u,v)

$$f'(v,u) = f(v,u) - c_P \geq f(v,u)- c_f(u,v)$$
$$= f(v,u)-f(v,u) = 0$$



- Other conditions like flow conservation still met

Ford-Fulkerson method

Start with $f = 0$ for every edge

While G_f has an augmenting path, augment.

Questions:

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

observations

- At every stage the capacities and flow values are always integers (if they start that way)
- The flow value $\nu(f') = \nu(f) + c_P > \nu(f)$ for

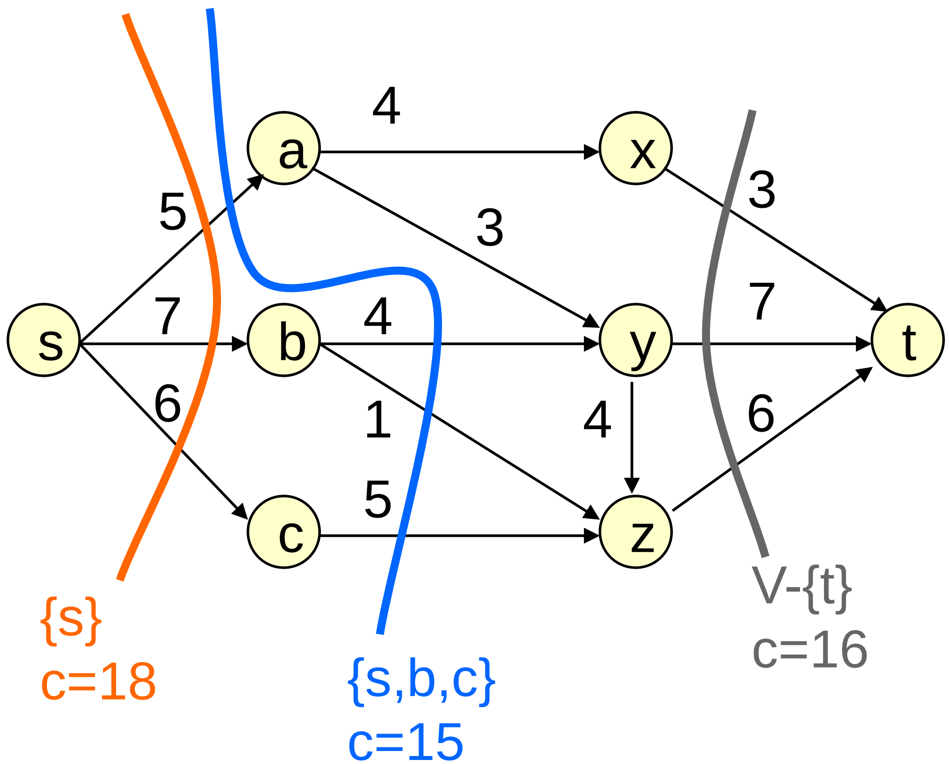
  $f' = \text{augment}(f,P)$
  - Since edges of residual capacity 0 do not appear in the residual graph
- Let $C = \sum_{(s,u)\in E} c(s,u)$
  - $\nu(f) \le C$
  - F-F does at most C rounds of augmentation since flows are integers and increase by at least 1 per step

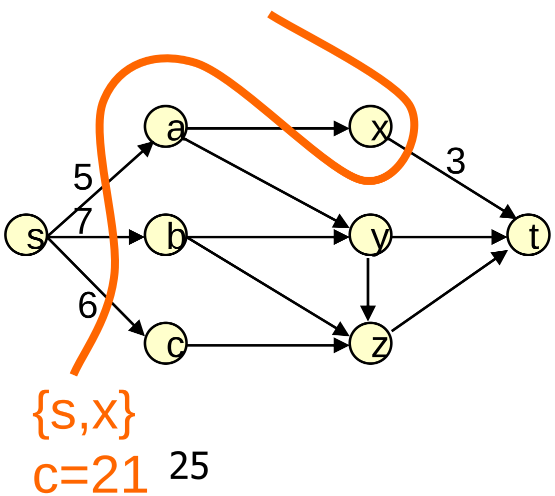
running time

- For $f = 0$, $G_f = G$
- Finding an augmenting path in G_f is graph search $O(n+m)=O(m)$ time
- Augmenting and updating G_f is $O(n)$ time
- Total $O(mC)$ time
- Does is find a maximum flow?
  - Need to show that for every flow f that isn't maximum G_f contains an s-t-path

cuts

- A partition (A,B) of V is an *s-t-cut* if
  s∈A, t∈B
- Capacity of cut (A,B) is $c(A,B) = \sum_{\substack{u \in A \\ v \in B}} c(u,v)$





convenient definitions

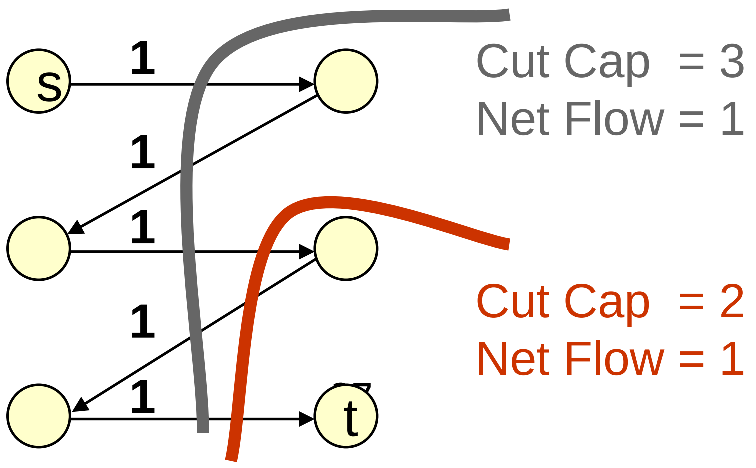
- $f^{out}(A) = \sum_{v \in A,\ w \notin A} f(v,w)$

- $f^{in}(A) = \sum_{v \in A,\ u \notin A} f(u,v)$

claims

- For any flow f and any cut (A,B),
  - the net flow across the cut equals the total flow: $\nu(f) = f^{out}(A)-f^{in}(A)$, and
  - the net flow across the cut cannot exceed the capacity of the cut: $f^{out}(A)-f^{in}(A) \leq c(A,B)$
- Corollary:

  Max flow $\leq$ Min cut



proof of claim

- Consider a set **A** with $s \in A$, $t \notin A$
- $f^{out}(A) - f^{in}(A) = \sum_{v \in A,\ w \notin A} f(v,w) - \sum_{v \in A,\ u \notin A} f(u,v)$
- We can add flow values for edges with both endpoints in **A** to **both** sums and they would cancel out so

- $f^{out}(A) - f^{in}(A) =$

- $\nu(f) = f^{out}(s)$ and $f^{in}(s)=0$

proof of claim

$$\begin{aligned}
\nu(f) &= f^{out}(A) - f^{in}(A) \\
&\leq f^{out}(A) \\
&= \sum_{v \in A,\, w \notin A} f(v,w) \\
&\leq \sum_{v \in A,\, w \notin A} c(v,w) \\
&\leq \sum_{v \in A,\, w \in B} c(v,w) \\
&= c(A,B)
\end{aligned}$$

max flow/min cut theorem

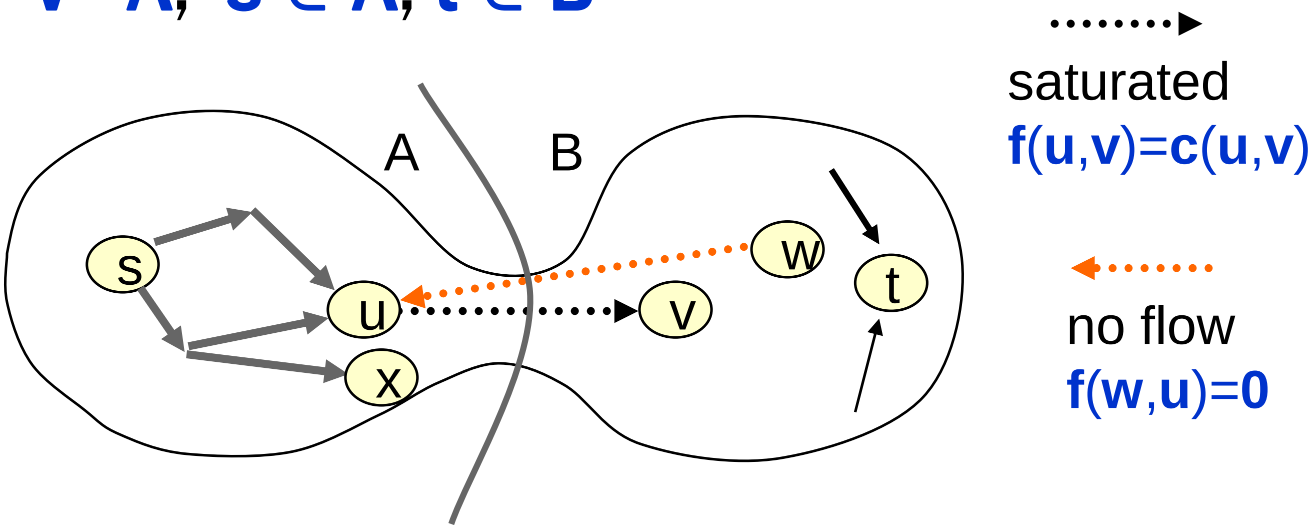
Theorem: For any flow f, if G_f has no augmenting path then there is some s-t-cut (A,B) such that $\nu(f)=c(A,B)$ (proof on next slide)

- We know by previous claims that any flow f' satisfies $\nu(f') \leq c(A,B)$ and we know that F-F runs for finite time until it finds a flow f satisfying conditions of the theorem Therefore for any flow f', $\nu(f') \leq \nu(f)$
- **Corollary:**
  – (1) F-F computes a maximum flow in G
  – (2) For any graph G, the value $n(f)$ of a maximum flow = minimum capacity $c(A,B)$ of any s-t-cut in G

proof of the theorem

Let $A = \{ u \mid \exists$ a path in G_f from s to $u \}$

$B = V - A$; $s \in A$, $t \in B$



This is true for **every** edge crossing the cut:

$\nu(f) = f^{out}(A) - f^{in}(A) = c(A,B)$ and $f^{in}(A) = 0$ hence

$$f^{out}(A) = \sum_{\substack{u \in A \\ v \in B}} f(u,v) = \sum_{\substack{u \in A \\ v \in B}} c(u,v) = c(A,B)$$