CSE 421: Introduction to Algorithms

Network Flow

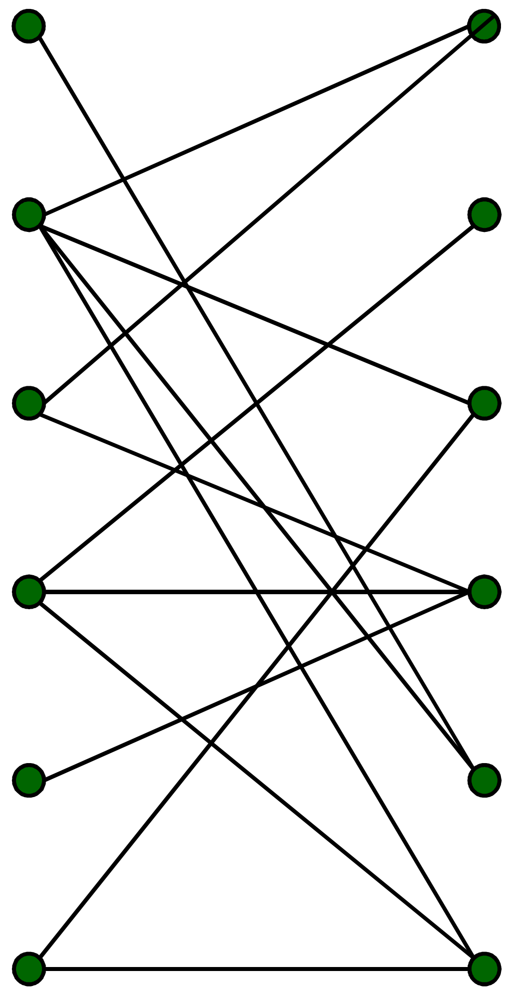
Paul Beame

Bipartite Matching

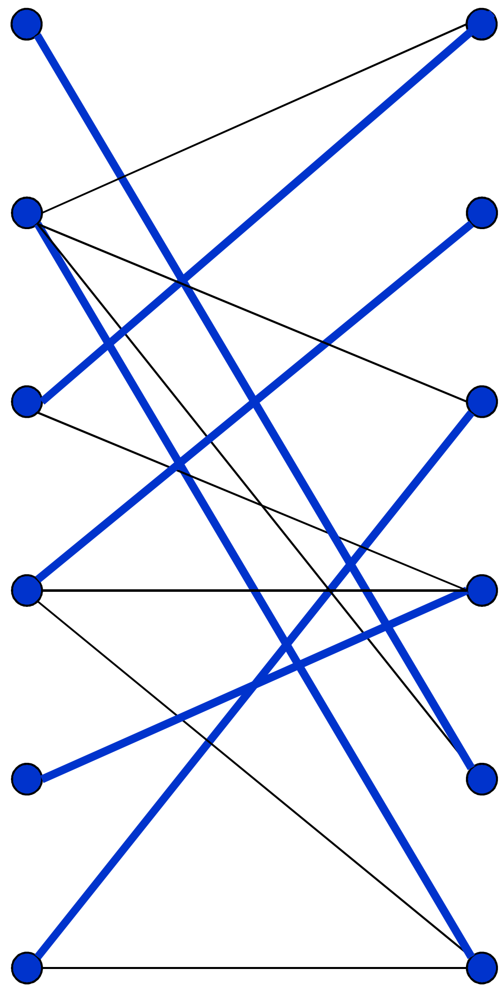
- Given: A bipartite graph $G=(V,E)$
  - $M \subseteq E$ is a matching in G iff no two edges in M share a vertex

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

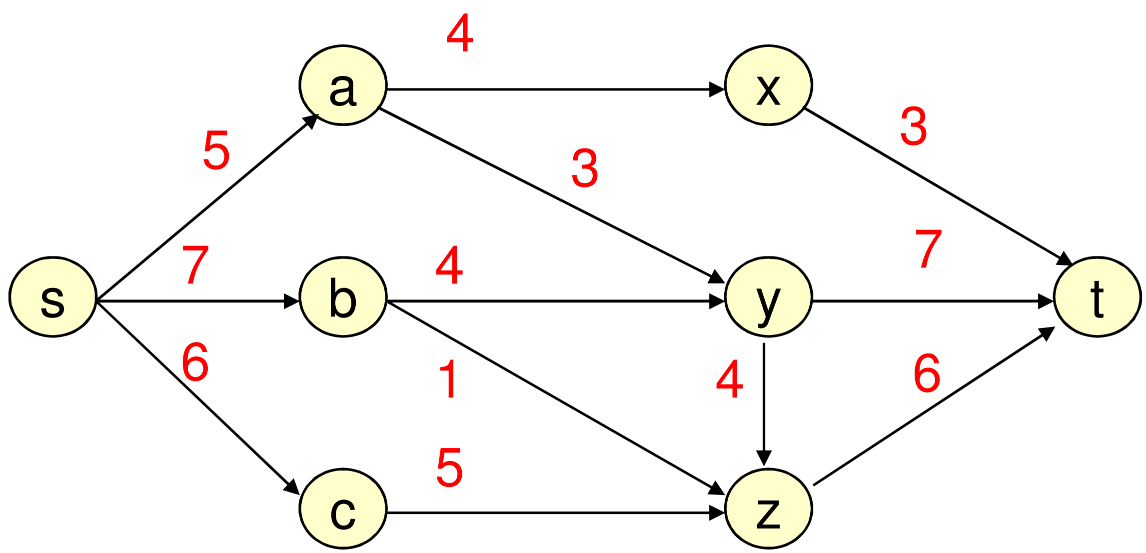
Bipartite Matching



Bipartite Matching

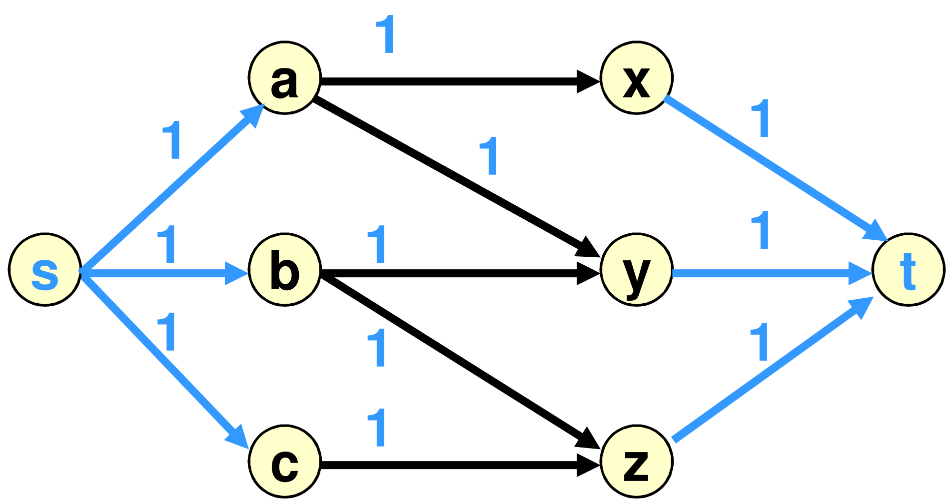


The Network Flow Problem



- How much stuff can flow from s to t?

Bipartite matching as a special case of flow



Net Flow: Formal Definition

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:

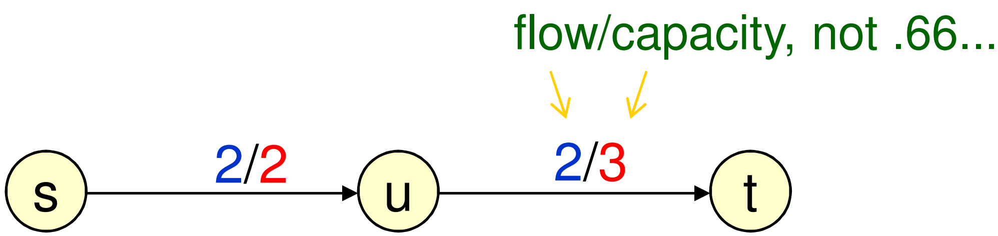
- $0 \leq f(u,v) \leq c(u,v)$ [Capacity Constraint]
- if $u \neq s,t$, i.e. $f^{out}(u)=f^{in}(u)$ [Flow Conservation]

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

Notation:

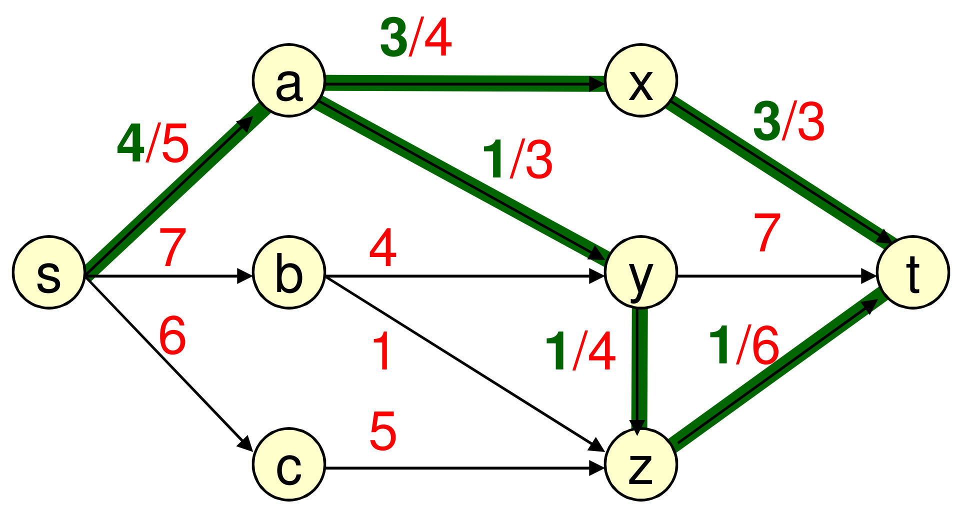
$$f^{in}(v) = \sum_{e=(u,v)\in E} f(u,v) \qquad f^{out}(v) = \sum_{e=(v,w)\in E} f(v,w)$$

Example: A Flow Function



$$f^{in}(u)=f(s,u)=2=f(u,t)=f^{out}(u)$$

Example: A Flow Function



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

Max Flow via a Greedy Alg?

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

Pick such a path, **p**

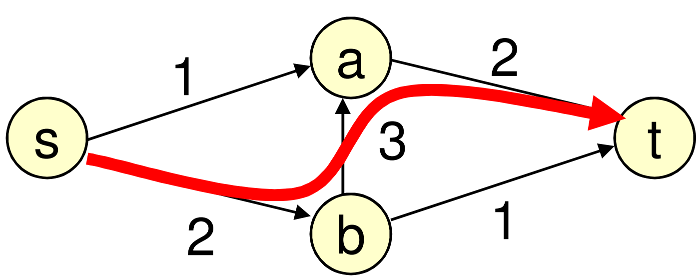
Find **c**, the min capacity of any edge in **p**

Count **c** towards the flow value

Subtract **c** from all capacities on **p**

Delete edges of capacity **0**

- This does NOT always find a max flow:



If pick $s \to b \to a \to t$ first, flow stuck at 2. But flow 3 possible.

A Brief History of Flow

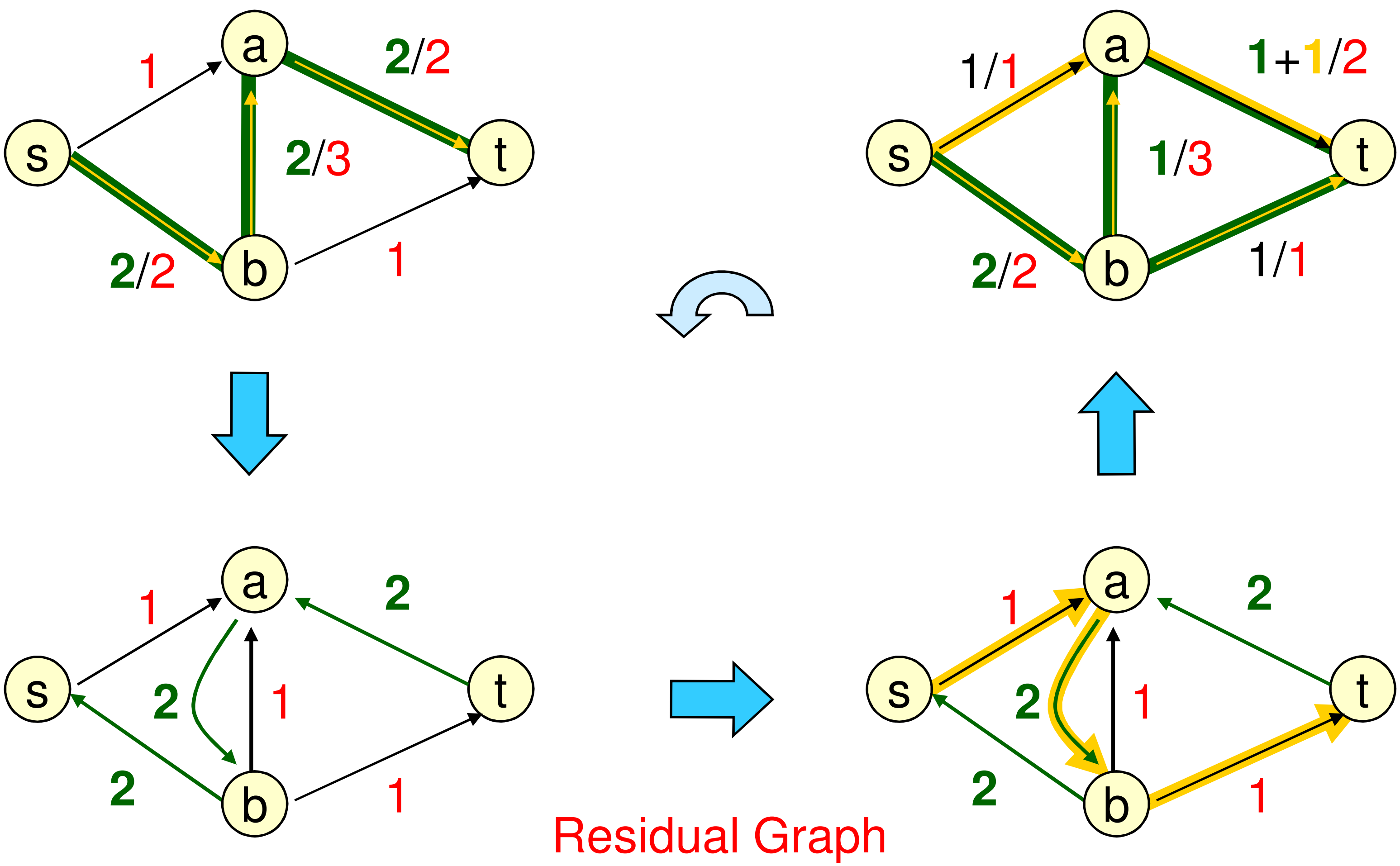
#	year	discoverer(s)	bound
1	1951	Dantzig	$O(n^2mU)$
2	1955	Ford & Fulkerson	$O(nmU)$
3	1970	Dinitz Edmonds & Karp	$O(nm^2)$
4	1970	Dinitz	$O(n^2m)$
5	1972	Edmonds & Karp Dinitz	$O(m^2 \log U)$
6	1973	Dinitz Gabow	$O(nm \log U)$
7	1974	Karzanov	$O(n^3)$
8	1977	Cherkassky	$O(n^2\sqrt{m})$
9	1980	Galil & 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. $O(nm)$

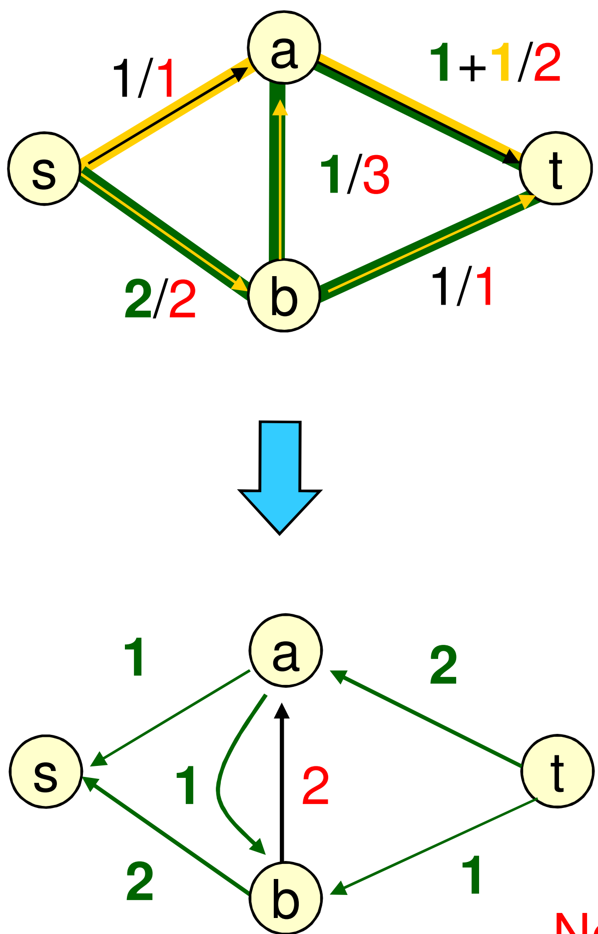
n = # of vertices
m= # of edges
U = Max capacity

Source: Goldberg & Rao, FOCS '97

Greed Revisited: Residual Graph & Augmenting Path



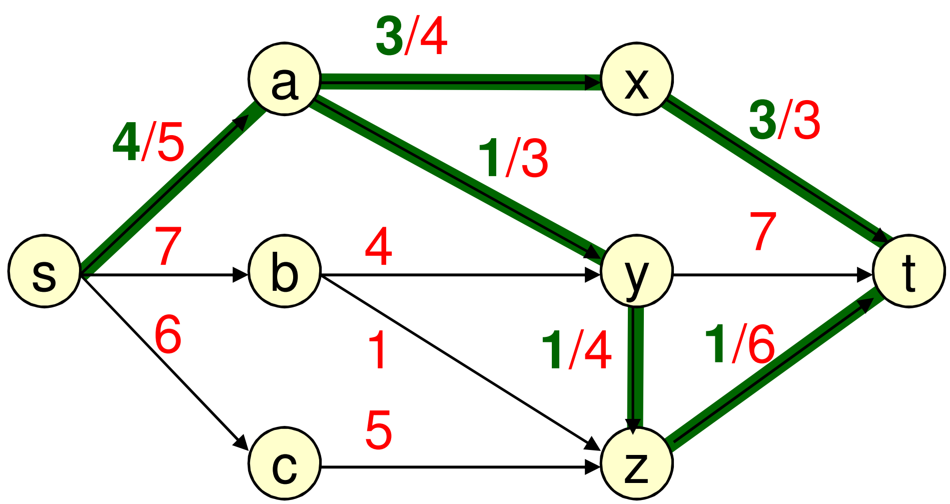
Greed Revisited: An Augmenting Path



New Residual Graph

Residual Capacity

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



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

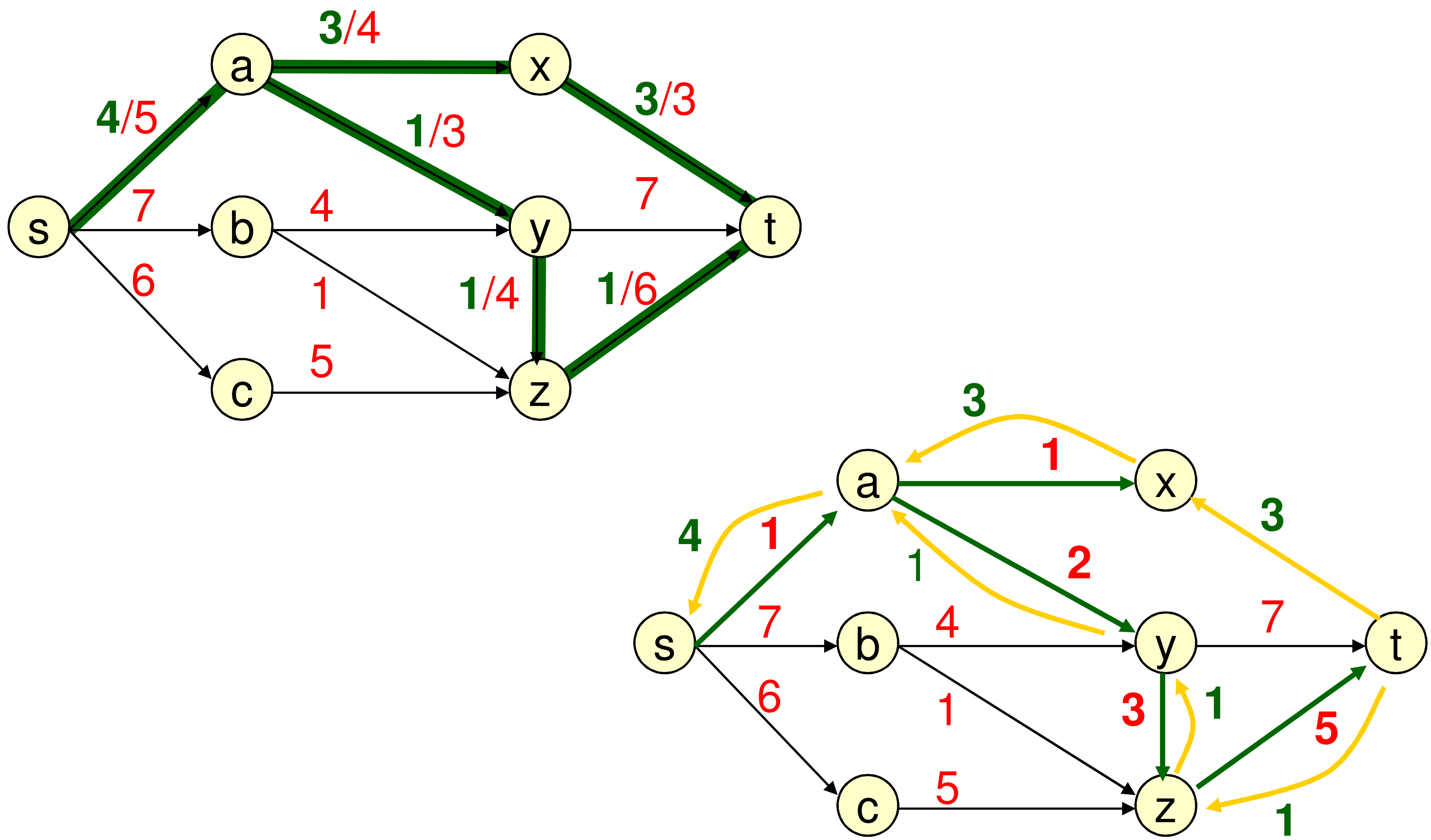
Residual Graph & Augmenting Paths

- The *residual graph* (w.r.t. $\mathbf{f}$) is the graph $\mathbf{G_f = (V, E_f)}$, where

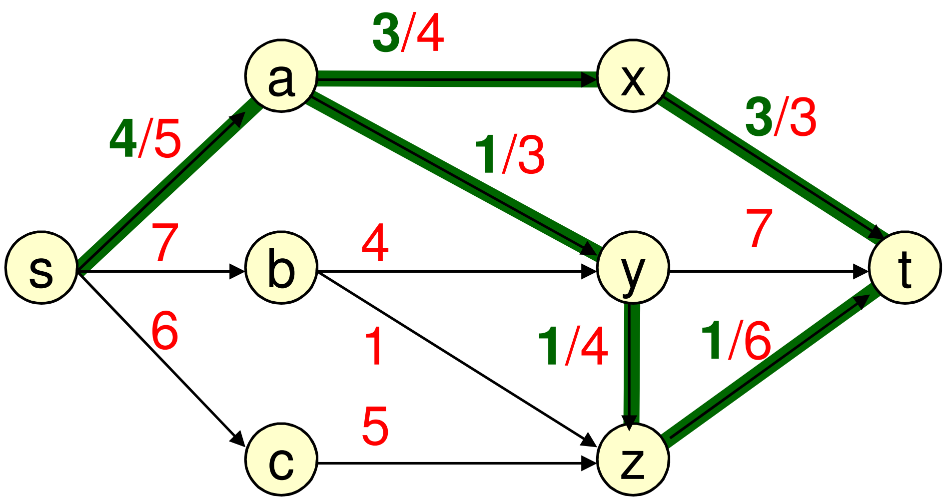
  $$\mathbf{E_f} = \{\, (\mathbf{u},\mathbf{v}) \mid \mathbf{c_f}(\mathbf{u},\mathbf{v}) > 0 \,\}$$

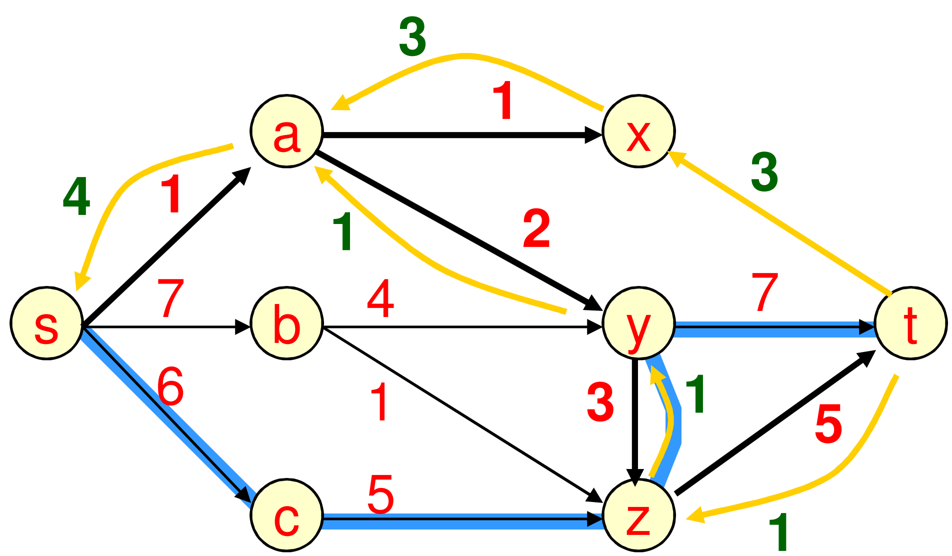
  - Two kinds of edges
    - Forward edges
      - $\mathbf{f(u,v) < c(u,v)}$ so $\mathbf{c_f(u,v) = c(u,v) - f(u,v) > 0}$
    - Backward edges
      - $\mathbf{f(u,v) > 0}$ so $\mathbf{c_f(v,u)} \geq \mathbf{-f(v,u) = f(u,v) > 0}$
- An *augmenting path* (w.r.t. $\mathbf{f}$) is a simple $\mathbf{s} \rightarrow \mathbf{t}$ path in $\mathbf{G_f}$.

A Residual Network



An Augmenting Path





Augmenting A Flow

augment($\mathbf{f}$,$\mathbf{P}$)

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

for each $\mathbf{e} \in \mathbf{P}$

if $\mathbf{e}$ is a forward edge then

increase $\mathbf{f}(\mathbf{e})$ by $\mathbf{c_P}$

else ($\mathbf{e}$ is a backward edge)

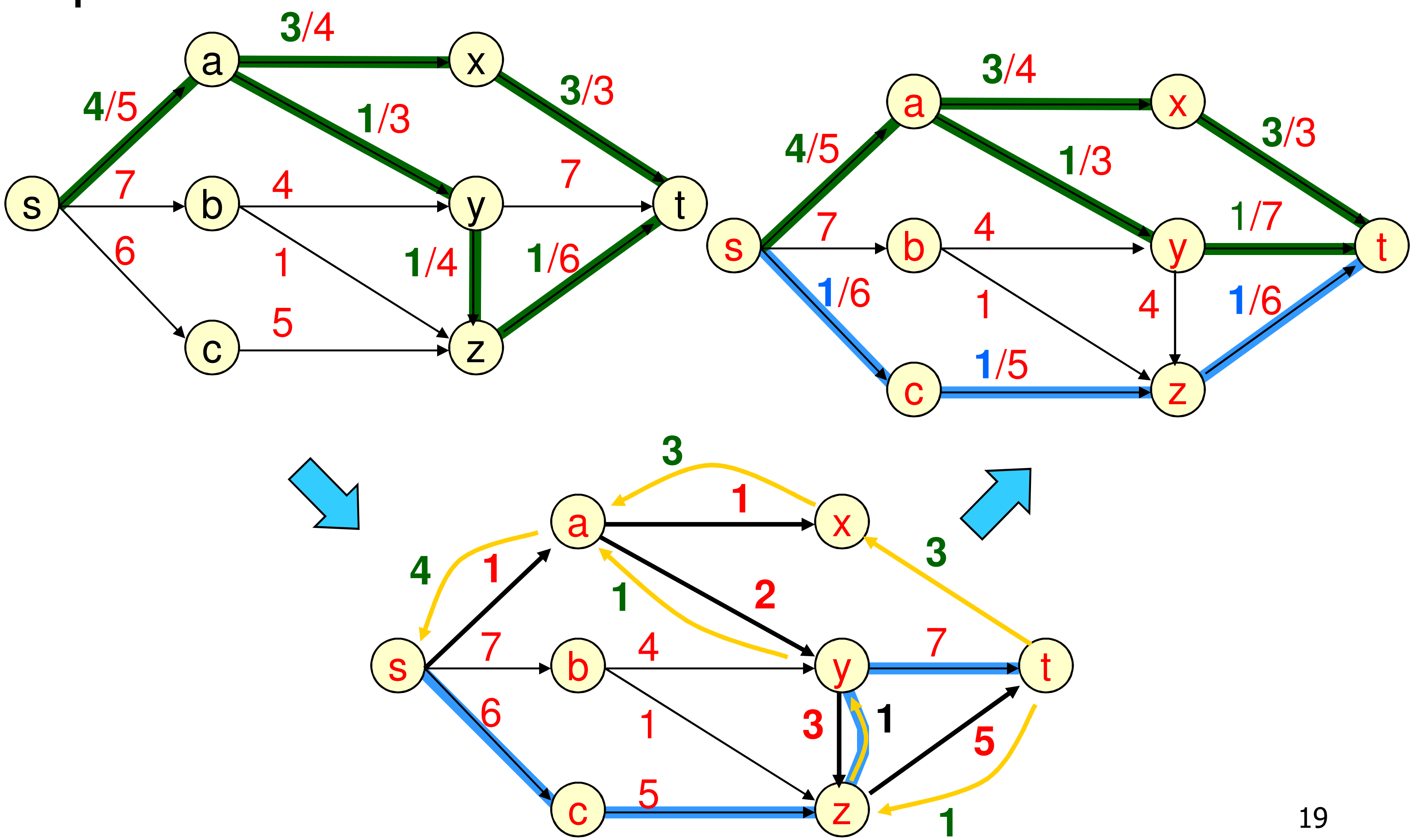
decrease $\mathbf{f}(\mathbf{e})$ by $\mathbf{c_P}$

endif

endfor

return($\mathbf{f}$)

Augmenting A Flow



Claim 7.1

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 7.1

- If (u,v) is a forward edge then $f'(u,v)=f(u,v)+c_P \le f(u,v)+c_f(u,v)$
  $= f(u,v)+c(u,v)-f(u,v)$
  $=c(u,v)$
- If (u,v) is a backward edge then f and f' differ on flow along (v,u) instead of (u,v)
  $f'(v,u)=f(v,u)-c_P \ge f(v,u)-c_f(u,v)$
  $= f(v,u)-f(v,u)=0$
- 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 about Ford-Fulkerson Algorithm

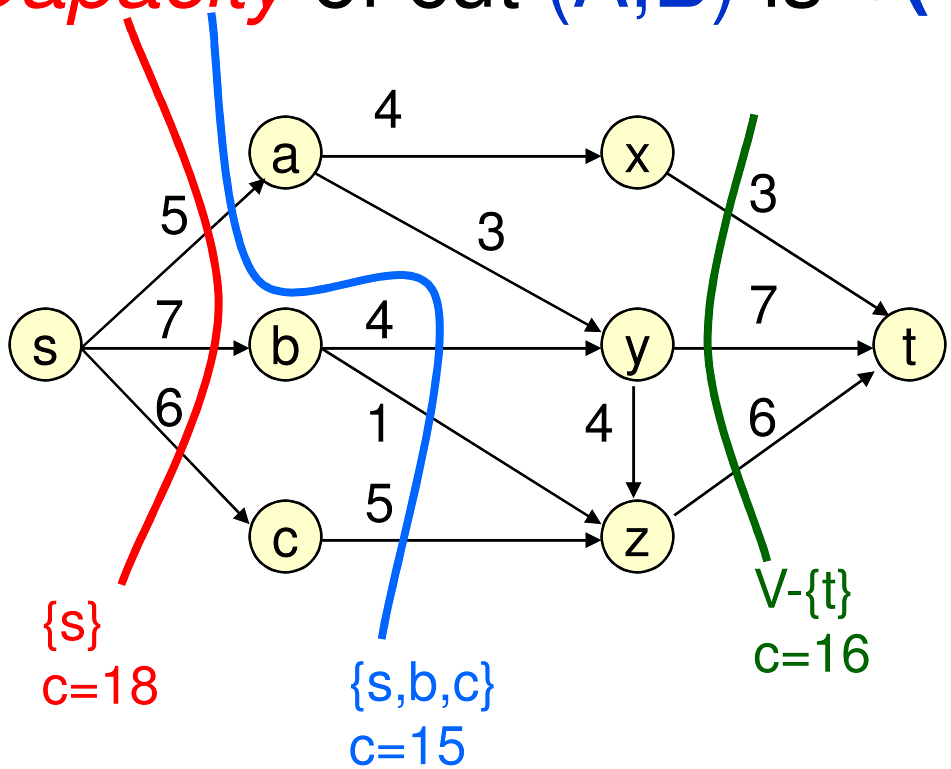
- At every stage the capacities and flow values are always integers (if they start that way)
- The flow value $\nu(\mathbf{f}')=\nu(\mathbf{f})+\mathbf{c_P}>\nu(\mathbf{f})$ for $\mathbf{f}'=\text{augment}(\mathbf{f},\mathbf{P})$
  - Since edges of residual capacity **0** do not appear in the residual graph
- Let $\mathbf{C}=\sum_{(\mathbf{s},\mathbf{u})\in \mathbf{E}} \mathbf{c}(\mathbf{s},\mathbf{u})$
  - $\nu(\mathbf{f})\leq\mathbf{C}$
  - **F-F** does at most **C** rounds of augmentation since flows are integers and increase by at least **1** per step

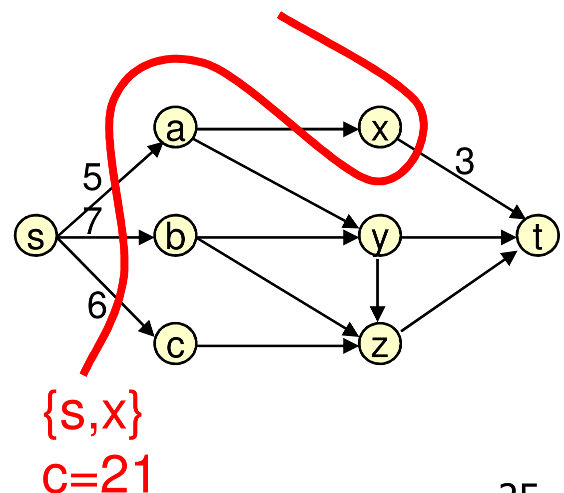
Running Time of Ford-Fulkerson

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

Cuts

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





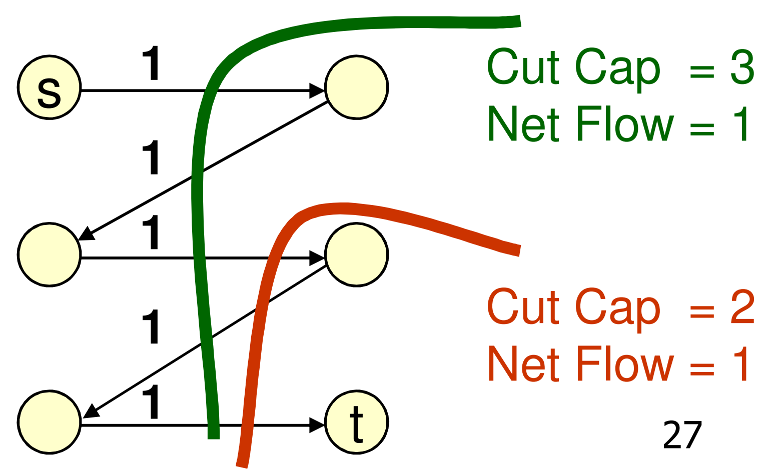
Convenient Definition

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

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

Claims 7.6 and 7.8

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



Proof of Claim 7.6

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

$$= \sum_{v\in A} \left(\sum_{w\in V} f(v,w) - \sum_{u\in V} f(u,v)\right)$$

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

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

since all other vertices have $f^{out}(v)=f^{in}(v)$

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

Proof of Claim 7.8

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

  $\leq f^{out}(A)$

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

  $\leq \sum_{v \in A,\, w \notin A} c(v,w)$

  $\leq \sum_{v \in A,\, w \in B} c(v,w)$

  $= c(A,B)$

Max Flow / Min Cut Theorem

Claim 7.9 For any flow **f**, if G_f has no augmenting path then there is some **s**-**t**-cut (**A**,**B**) such that $\nu(\mathbf{f})=\mathbf{c}(\mathbf{A},\mathbf{B})$ (proof on next slide)

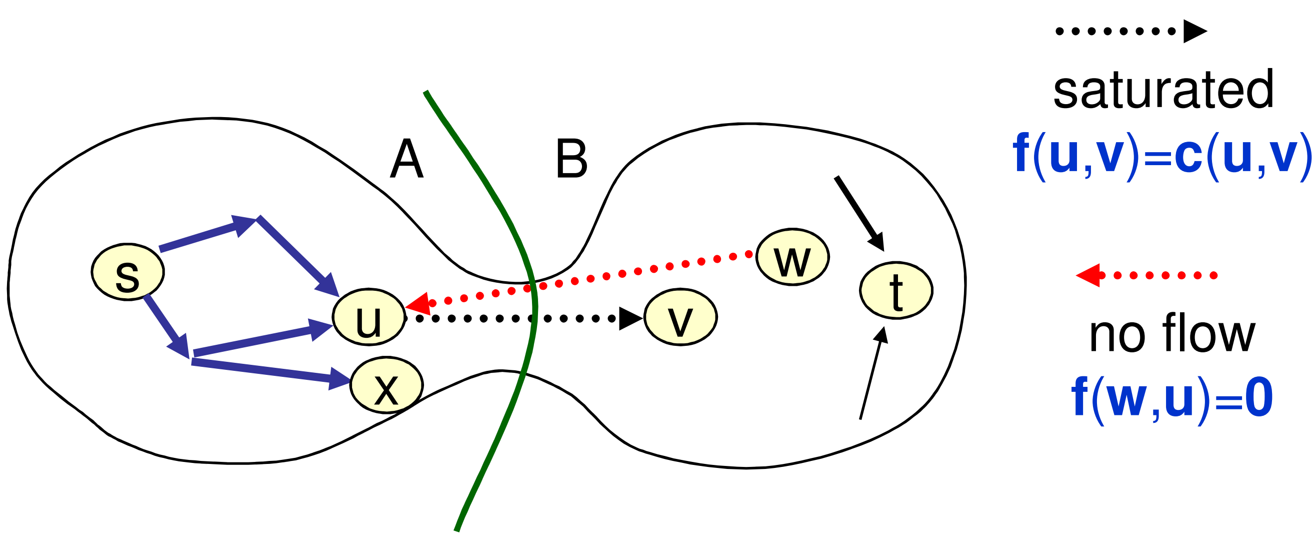
- We know by Claims 7.6 & 7.8 that any flow **f'** satisfies $\nu(\mathbf{f'}) \leq \mathbf{c}(\mathbf{A},\mathbf{B})$ and we know that F-F runs for finite time until it finds a flow **f** satisfying conditions of Claim 7.9
  - Therefore by 7.9 for any flow **f'**, $\nu(\mathbf{f'}) \leq \nu(\mathbf{f})$
- Corollary (1) F-F computes a maximum flow in **G**

  (2) For any graph **G**, the value $\nu(\mathbf{f})$ of a maximum flow = minimum capacity $\mathbf{c}(\mathbf{A},\mathbf{B})$ of any **s**-**t**-cut in **G**

Claim 7.9

Let **A** = { **u** | ∃ an path in $\mathbf{G_f}$ from **s** to **u** }

B = **V** - **A**; $\mathbf{s} \in \mathbf{A}$, $\mathbf{t} \in \mathbf{B}$



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

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

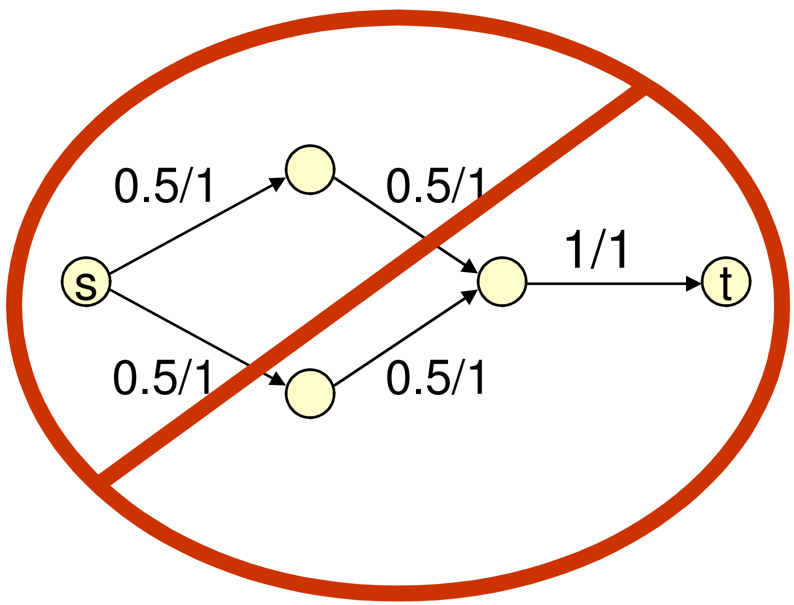
and $\mathbf{f}^{in}(\mathbf{A})=\mathbf{0}$ so

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

Flow Integrality Theorem

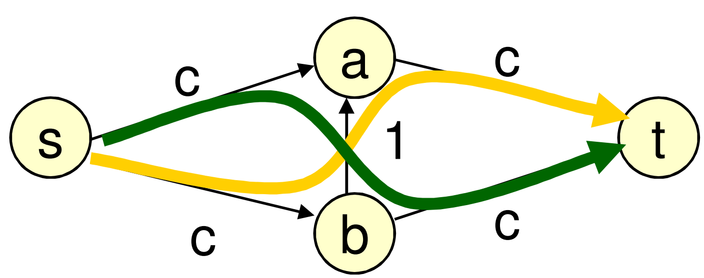
If all capacities are integers

- The 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)



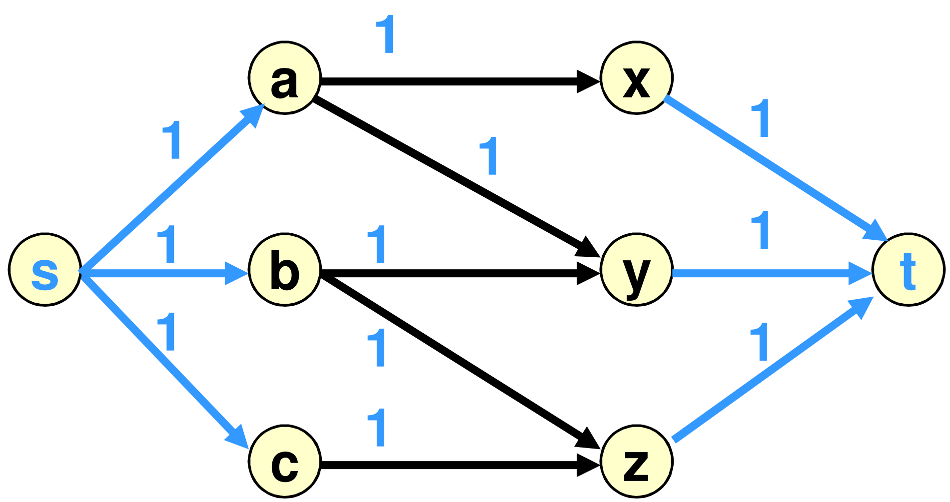
Corollaries & Facts

- 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 = **10^9**, say

Bipartite matching as a special case of flow



Integer flows implies each flow is just a subset of the edges

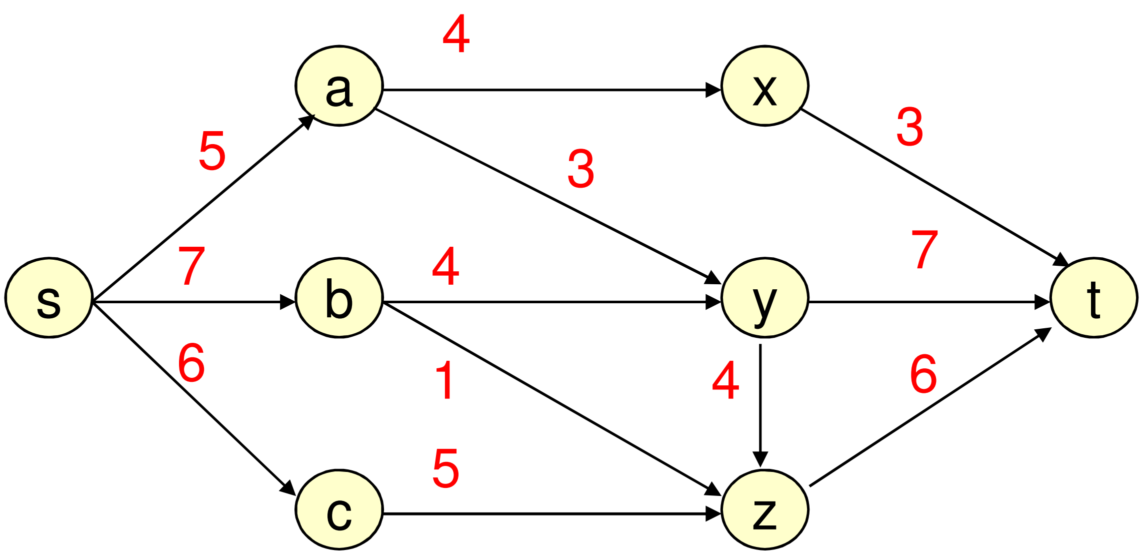
Therefore flow corresponds to a matching

O(**mC**)=O(**nm**) running time

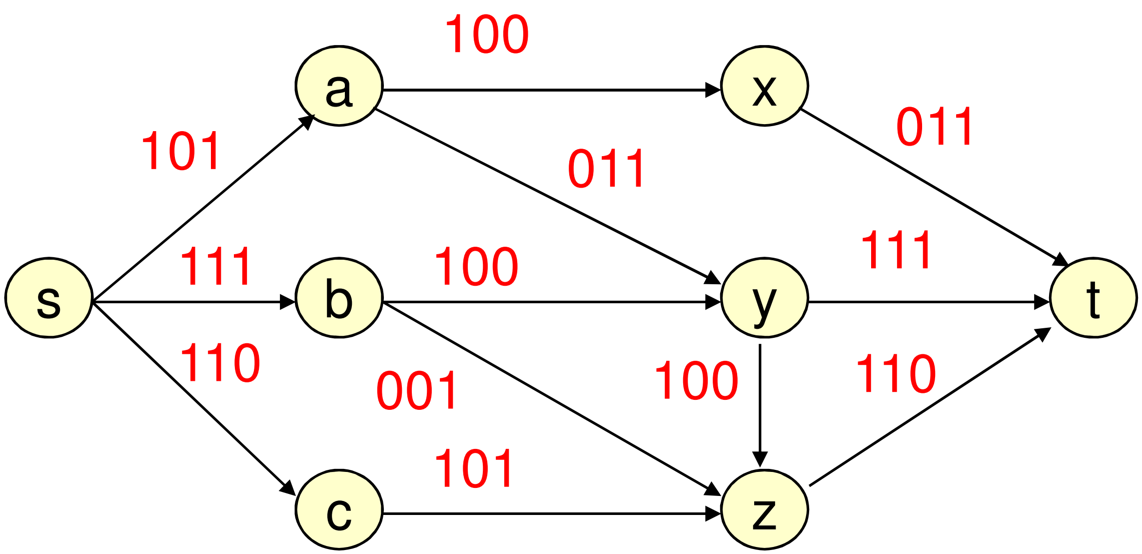
Capacity-scaling algorithm

- General idea:
  - Choose augmenting paths **P** with ‘large’ capacity $\mathbf{c_P}$
  - Can augment flows along a path **P** by any amount $\Delta \le \mathbf{c_P}$
    - Ford-Fulkerson still works
  - Get a flow that is maximum for the high-order bits first and then add more bits later

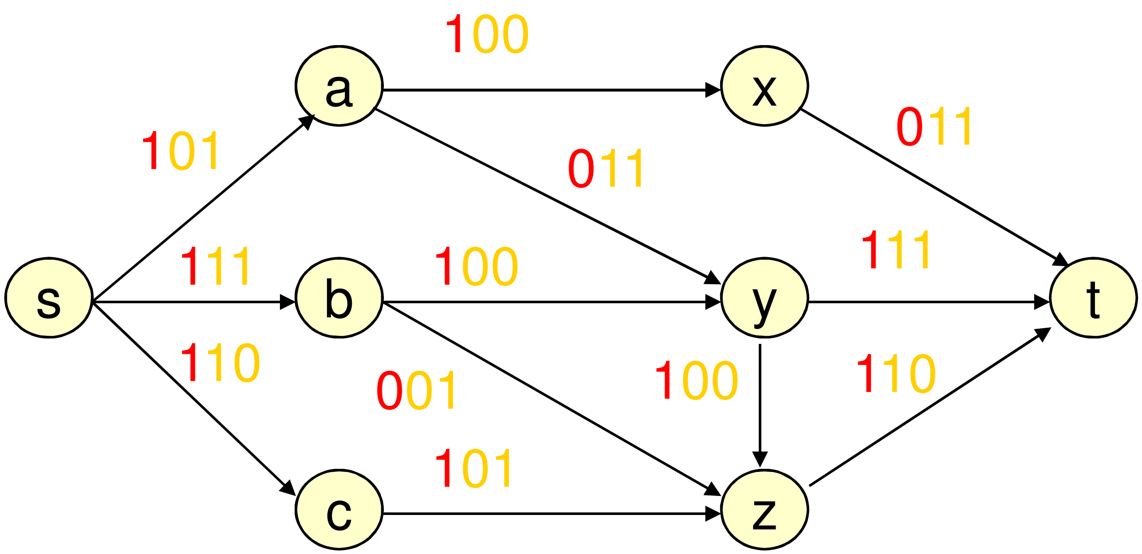
Capacity Scaling



Capacity Scaling

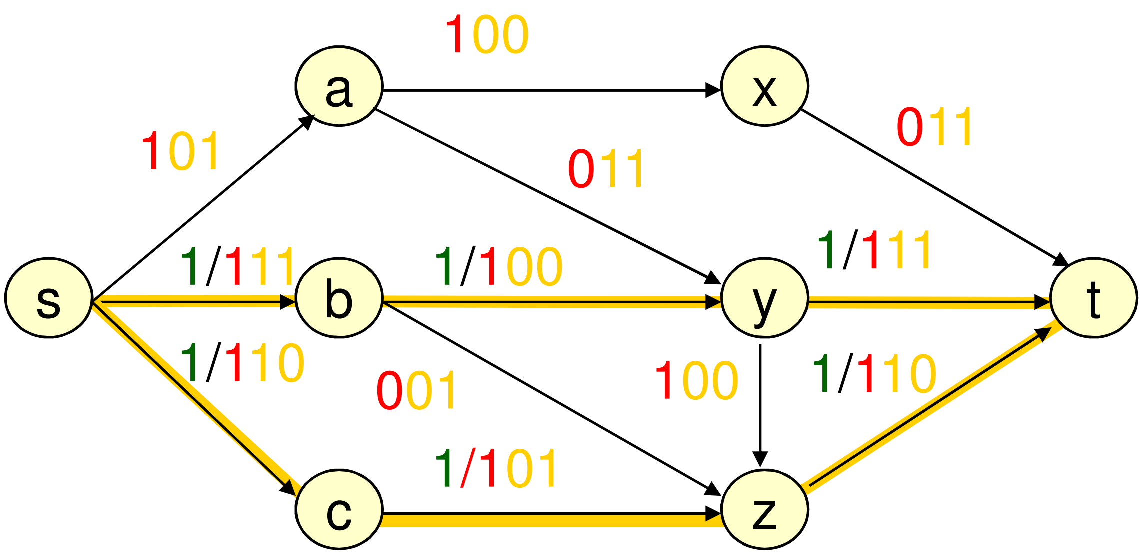


Capacity Scaling Bit 1



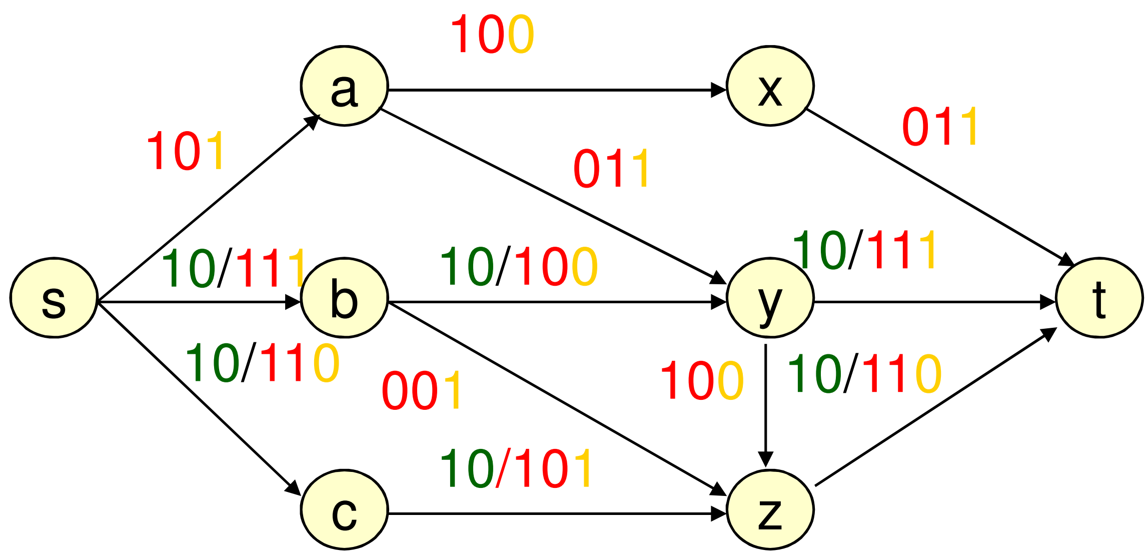
Capacity on each edge is at most **1**
(either **0** or **1** times **Δ=4**)

Capacity Scaling Bit 1



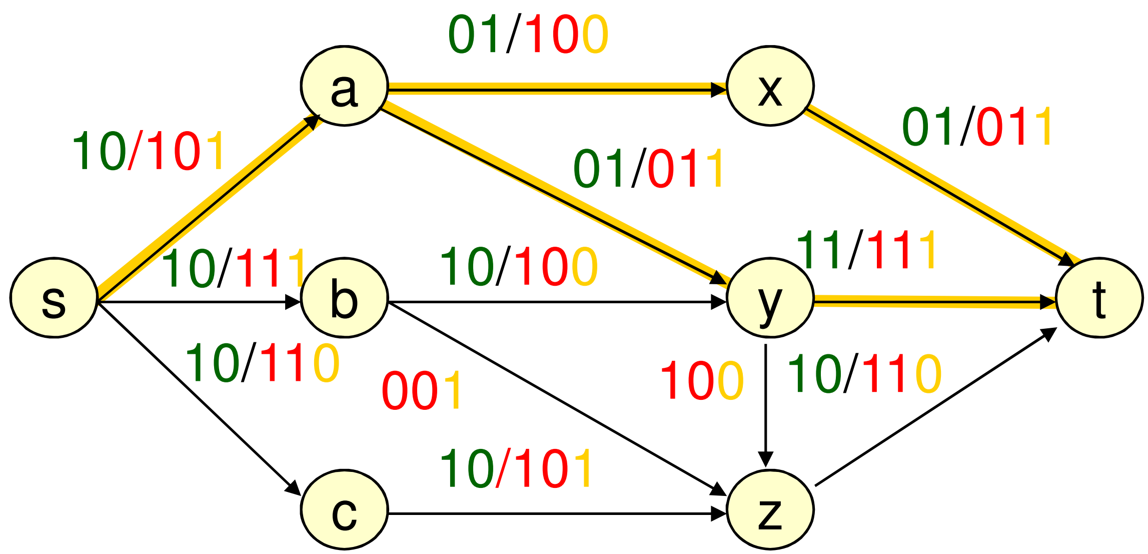
O(**nm**) time

Capacity Scaling Bit 2



Residual capacity across min cut is at most **m**
(either **0** or **1** times **Δ=2**)

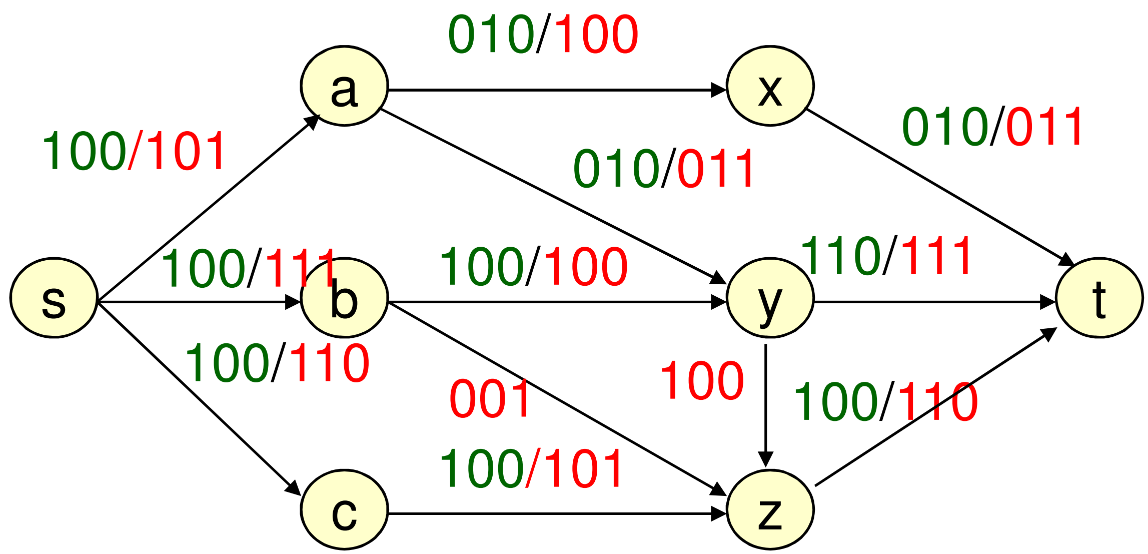
Capacity Scaling Bit 2



Residual capacity across min cut is at most **m**

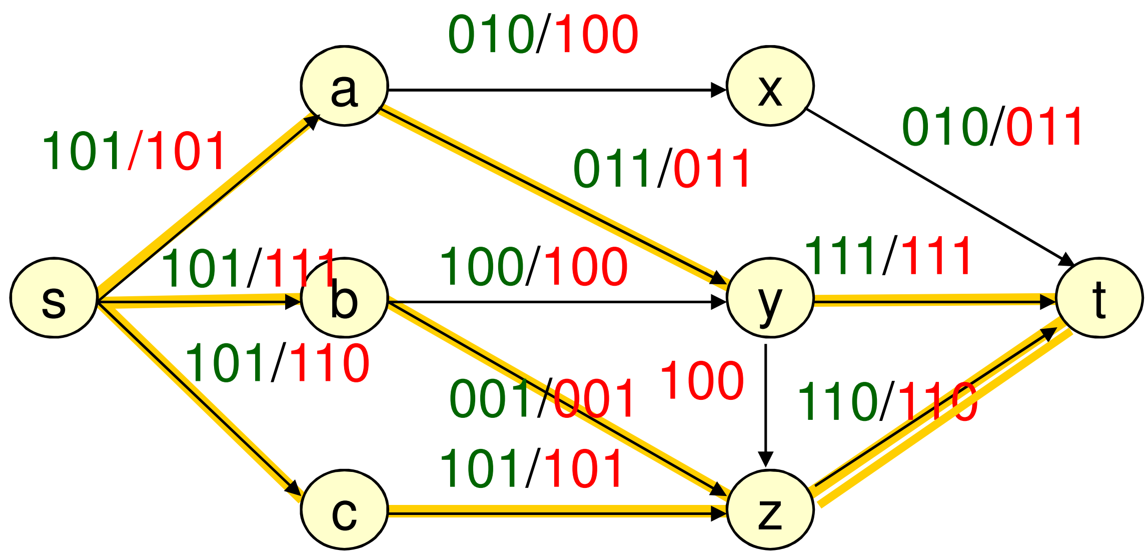
⇒ ≤ **m** augmentations

Capacity Scaling Bit 3



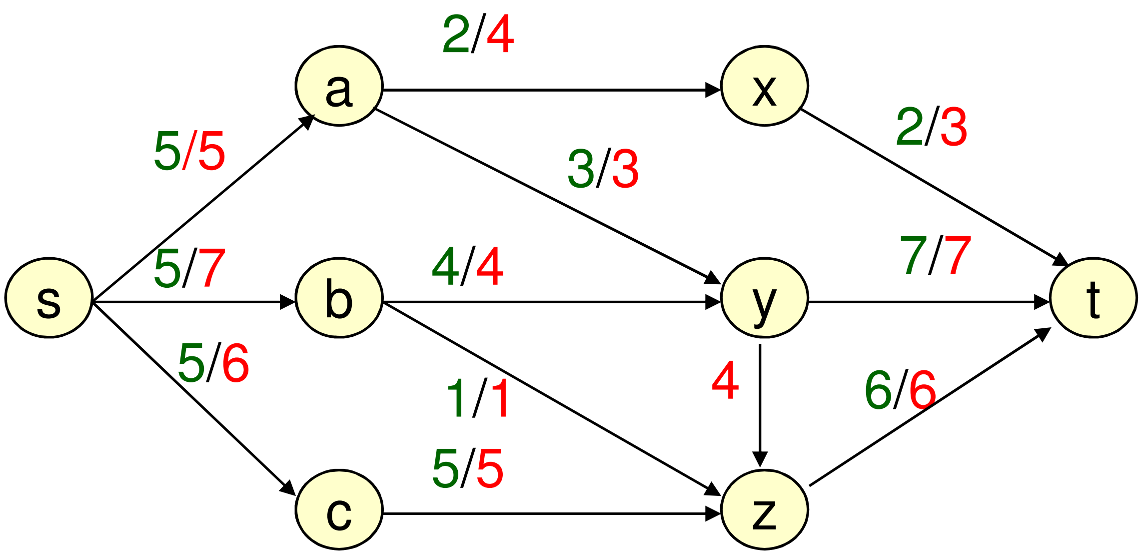
Residual capacity across min cut is at most **m**
(either **0** or **1** times **Δ=1**)

Capacity Scaling Bit 3

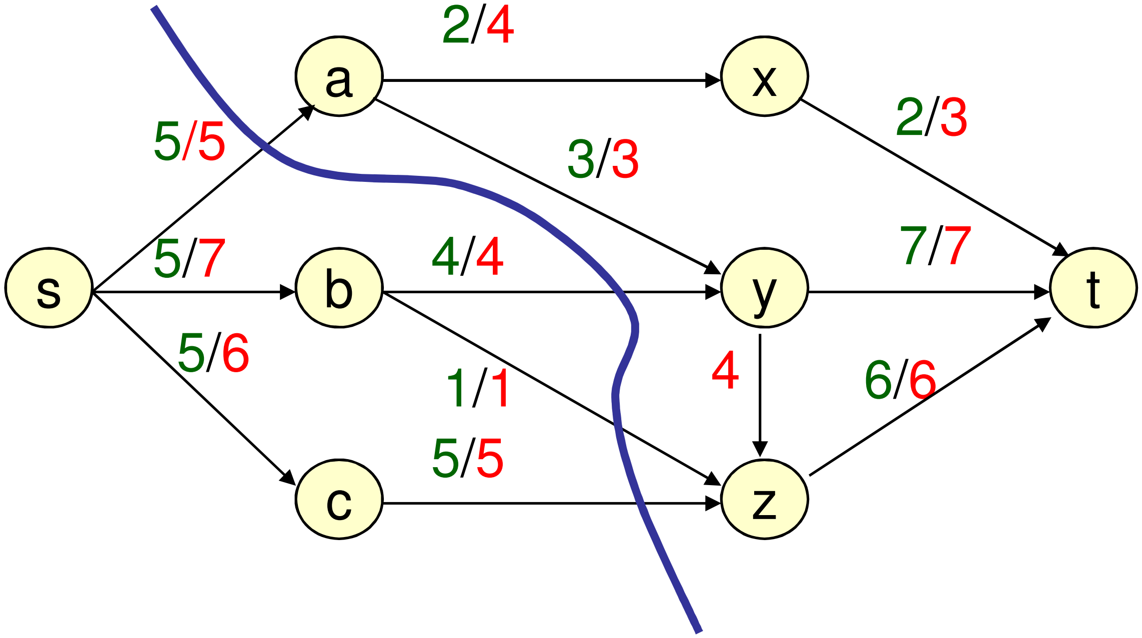


After ≤ **m** augmentations

Capacity Scaling Final



Capacity Scaling Min Cut



Total time for capacity scaling

- $\log_2 \mathbf{U}$ rounds where $\mathbf{U}$ is largest capacity
- At most $\mathbf{m}$ augmentations per round
  - Let $\mathbf{c_i}$ be the capacities used in the $\mathbf{i}^{th}$ round and $\mathbf{f}_i$ be the maxflow found in the $\mathbf{i}^{th}$ round
    - For any edge $(\mathbf{u},\mathbf{v})$, $\mathbf{c_{i+1}}(\mathbf{u},\mathbf{v}) \leq \mathbf{2c_i}(\mathbf{u},\mathbf{v})+\mathbf{1}$
  - $\mathbf{i+1}^{st}$ round starts with flow $\mathbf{f} = \mathbf{2}\ \mathbf{f_i}$
  - Let $(\mathbf{A},\mathbf{B})$ be a min cut from the $\mathbf{i}^{th}$ round
    - $\nu(\mathbf{f_i})=\mathbf{c_i}(\mathbf{A},B)$ so $\nu(\mathbf{f})=\mathbf{2c_i}(\mathbf{A},\mathbf{B})$
  - $\nu(\mathbf{f_{i+1}}) \leq \mathbf{c_{i+1}}(\mathbf{A},\mathbf{B}) \leq \mathbf{2c_i}(\mathbf{A},\mathbf{B})+\mathbf{m} =\nu(\mathbf{f})+\mathbf{m}$
- $O(\mathbf{m})$ time per augmentation
- Total time $O(\mathbf{m}^2 \log \mathbf{U})$

Edmonds-Karp Algorithm

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

- Time: $O(\mathbf{n}\ \mathbf{m}^2)$

BFS/Shortest Path Lemmas

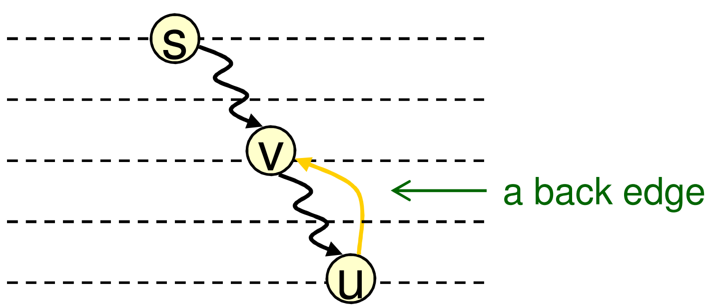
Distance from **s** in $\mathbf{G_f}$ is never reduced by:

- 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

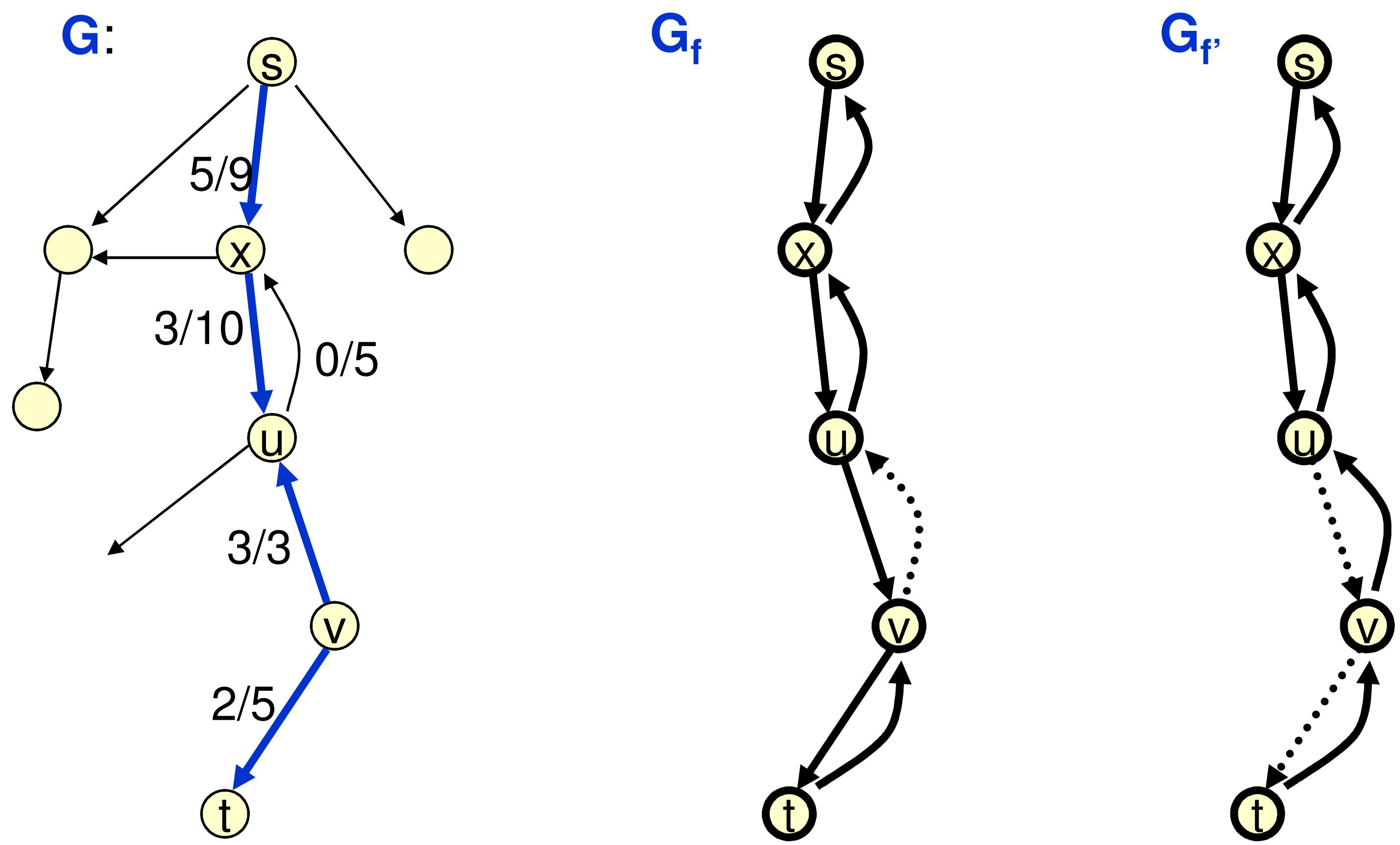


Key Lemma

Let **f** be a flow, **G_f** the residual graph, and **P** a shortest augmenting path. Then no vertex is closer to **s** after augmentation along **P**.

Proof: Augmentation along **P** only deletes forward edges, or adds back edges that go to previous vertices along **P**

Augmentation vs BFS



Theorem

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

Proof:

Call (u,v) critical for augmenting path **P** if it's closest to **s** having min residual capacity

It will disappear from G_f after augmenting along **P**

In order for **(u,v)** to be critical again the **(u,v)** edge must re-appear in G_f but that will only happen when the distance to **u** has increased by **2** (next slide)

It won't be critical again until farther from **s**
so each edge critical at most **n/2** times

Critical Edges in G_f

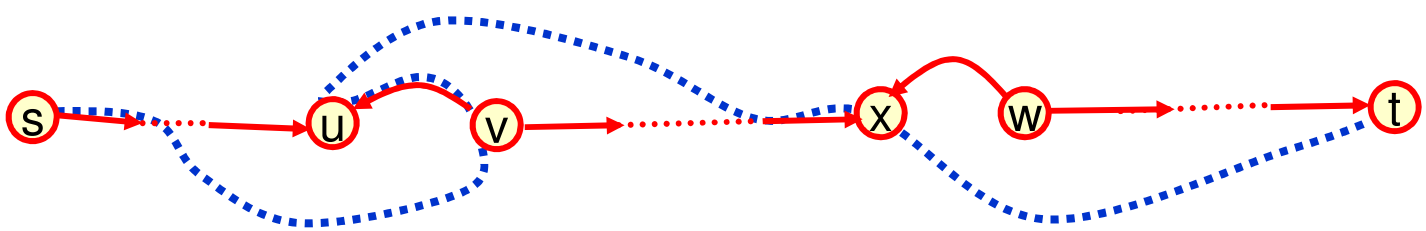
Shortest s-t path **P** in G_f

critical edge

$d_f(s,v)=d_f(s,u)+1$ since this is a shortest path

After augmenting along **P**

For (**u**,**v**) to be critical later for some flow **f'** it must be in $G_{f'}$
so must have augmented along a shortest path containing **(v,u)**



Then we must have $d_{f'}(s,u)=d_{f'}(s,v)+1 \geq d_f(s,v)+1=d_f(s,u)+2$

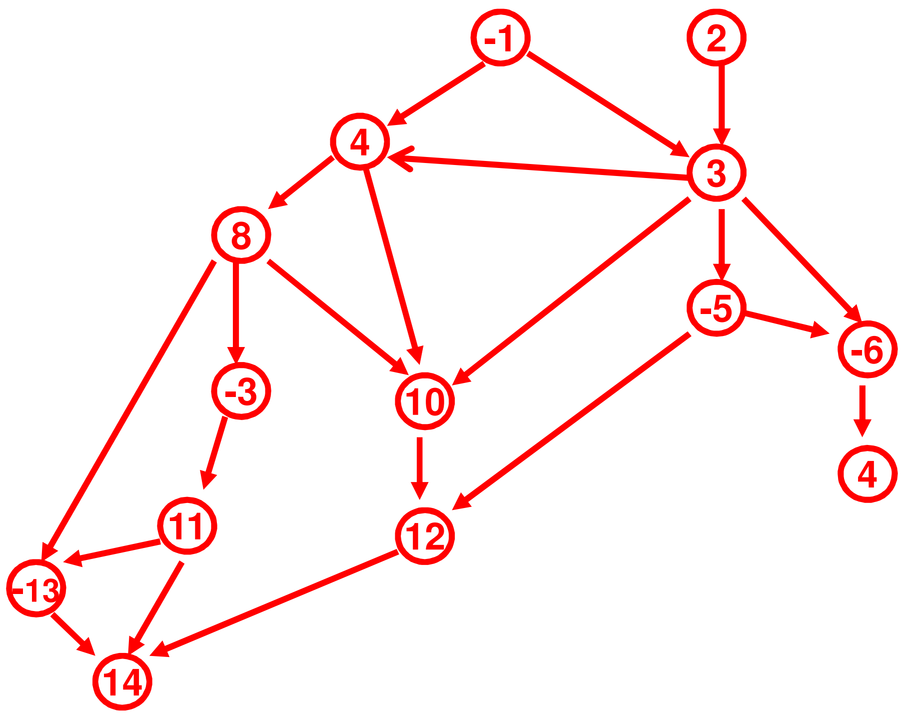
Corollary

- Edmonds-Karp runs in $O(\mathbf{nm^2})$ time

Project Selection
a.k.a. The Strip Mining Problem

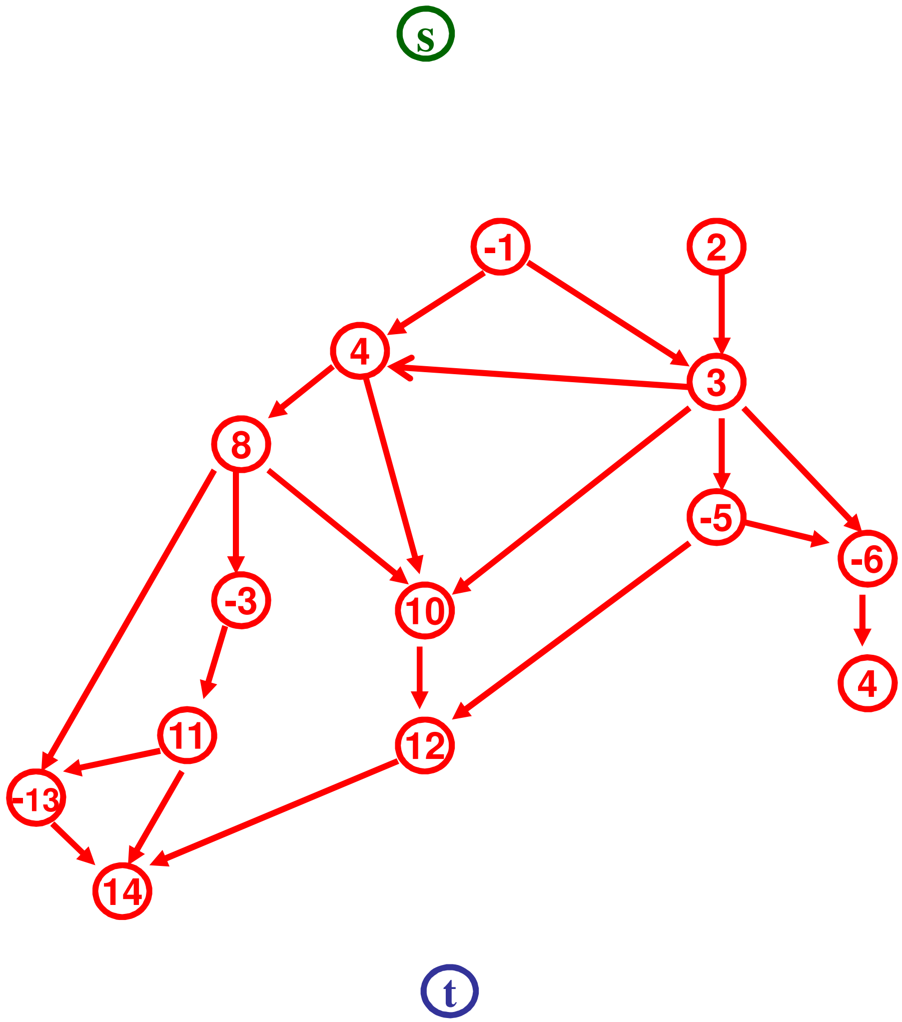
- Given
  - a directed acyclic graph $G=(V,E)$ representing precedence constraints on tasks (a task points to its **predecessors**)
  - a profit value $p(v)$ associated with each task $v \in V$ (may be positive or negative)
- Find
  - a set $A \subseteq V$ of tasks that is closed under predecessors, i.e. if $(u,v) \in E$ and $u \in A$ then $v \in A$, that maximizes $\text{Profit}(A)=\sum_{v \in A} p(v)$

Project Selection Graph



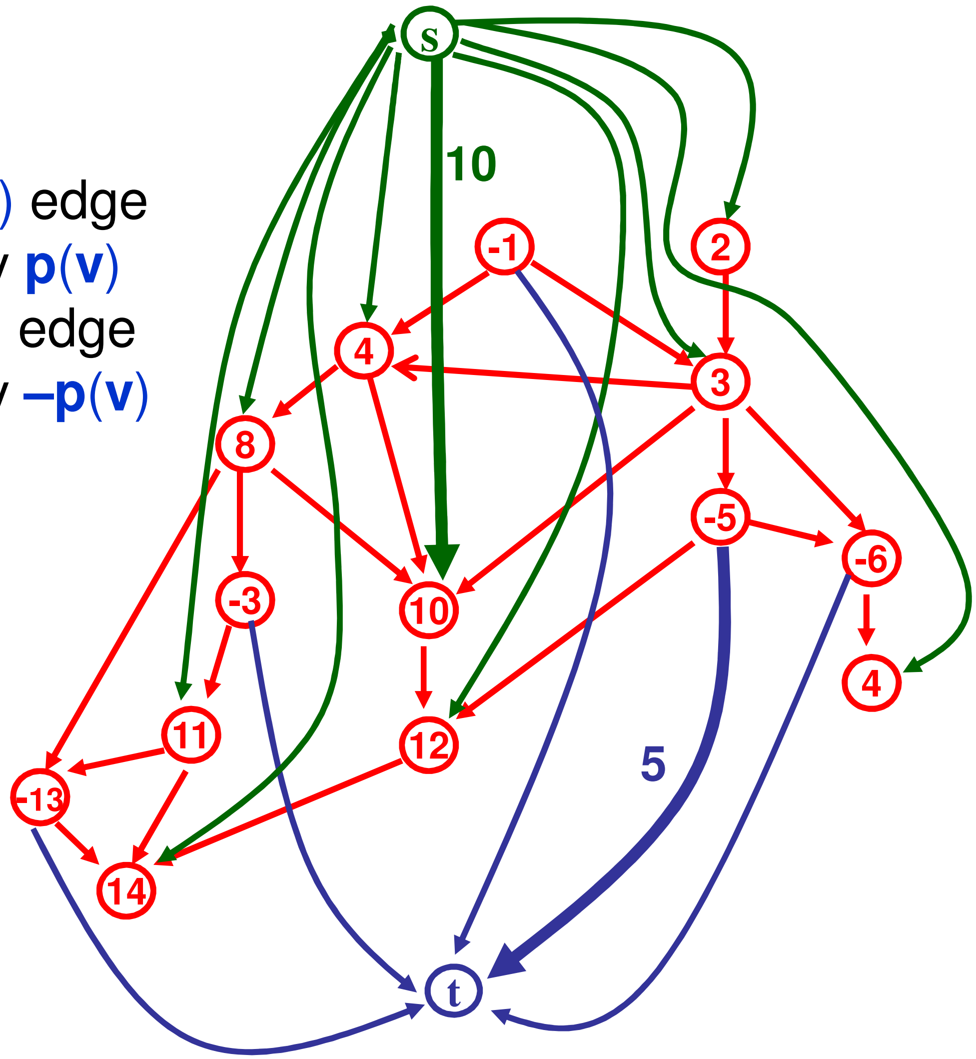
Each task points to its predecessor tasks

Extended Graph



Extended Graph G'

For each vertex **v**
If **p**(**v**)≥**0** add (**s**,**v**) edge
with capacity **p**(**v**)
If **p**(**v**)<**0** add (**v**,**t**) edge
with capacity **–p**(**v**)

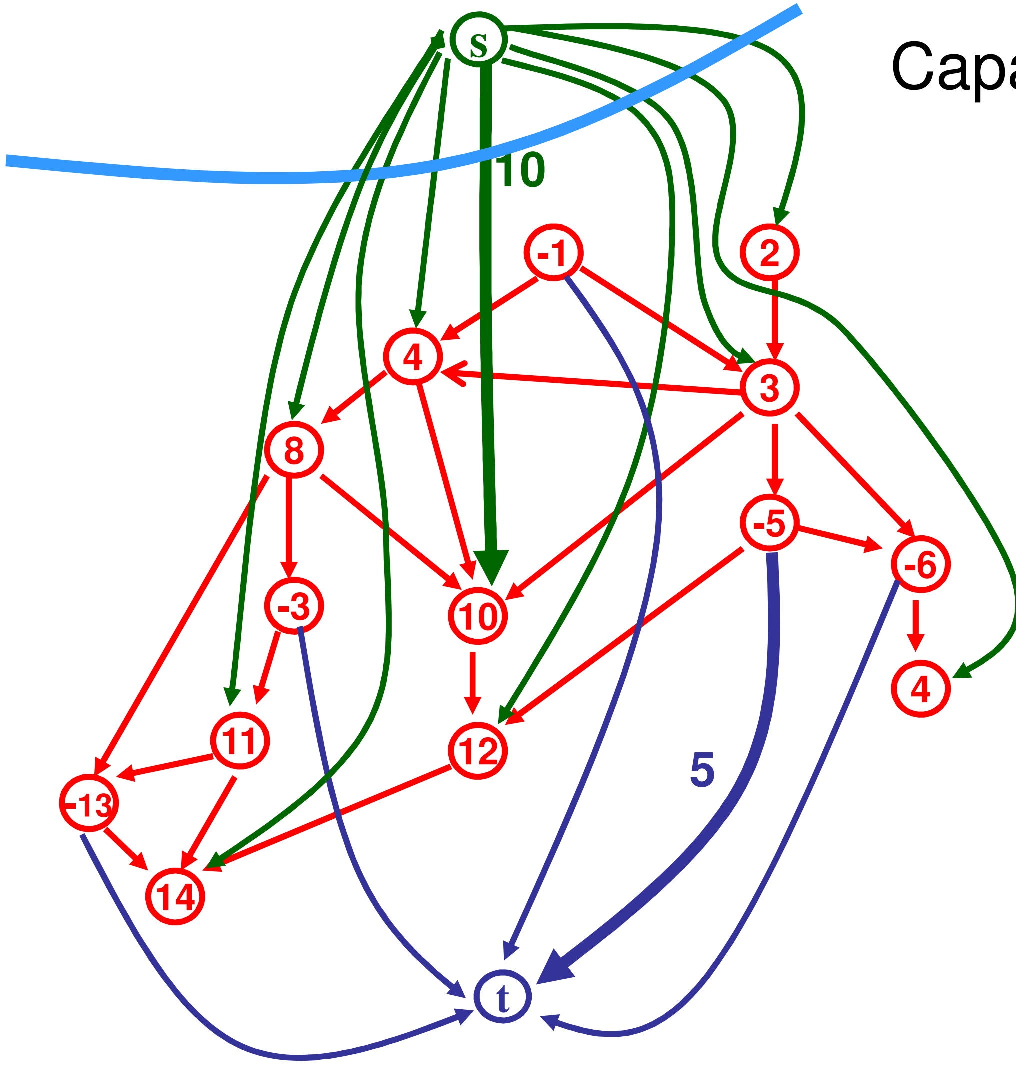


Extended Graph G'

- Want to arrange capacities on edges of **G** so that for minimum **s**-**t**-cut (**S**,**T**) in **G'**, the set **A**=**S**-{**s**}
  - satisfies precedence constraints
  - has maximum possible profit in **G**
- Cut capacity with **S**={**s**} is just $\mathbf{C}=\sum_{\mathbf{v}:\ \mathbf{p(v)}\geq 0} \mathbf{p(v)}$
  - Profit(**A**) ≤ **C** for any set **A**
- To satisfy precedence constraints don't want any original edges of **G** going forward across the minimum cut
  - That would correspond to a task in **A**=**S**-{**s**} that had a predecessor not in **A**=**S**-{**s**}
- Set capacity of each of the edges of **G** to **C+1**
  - The minimum cut has size at most **C**

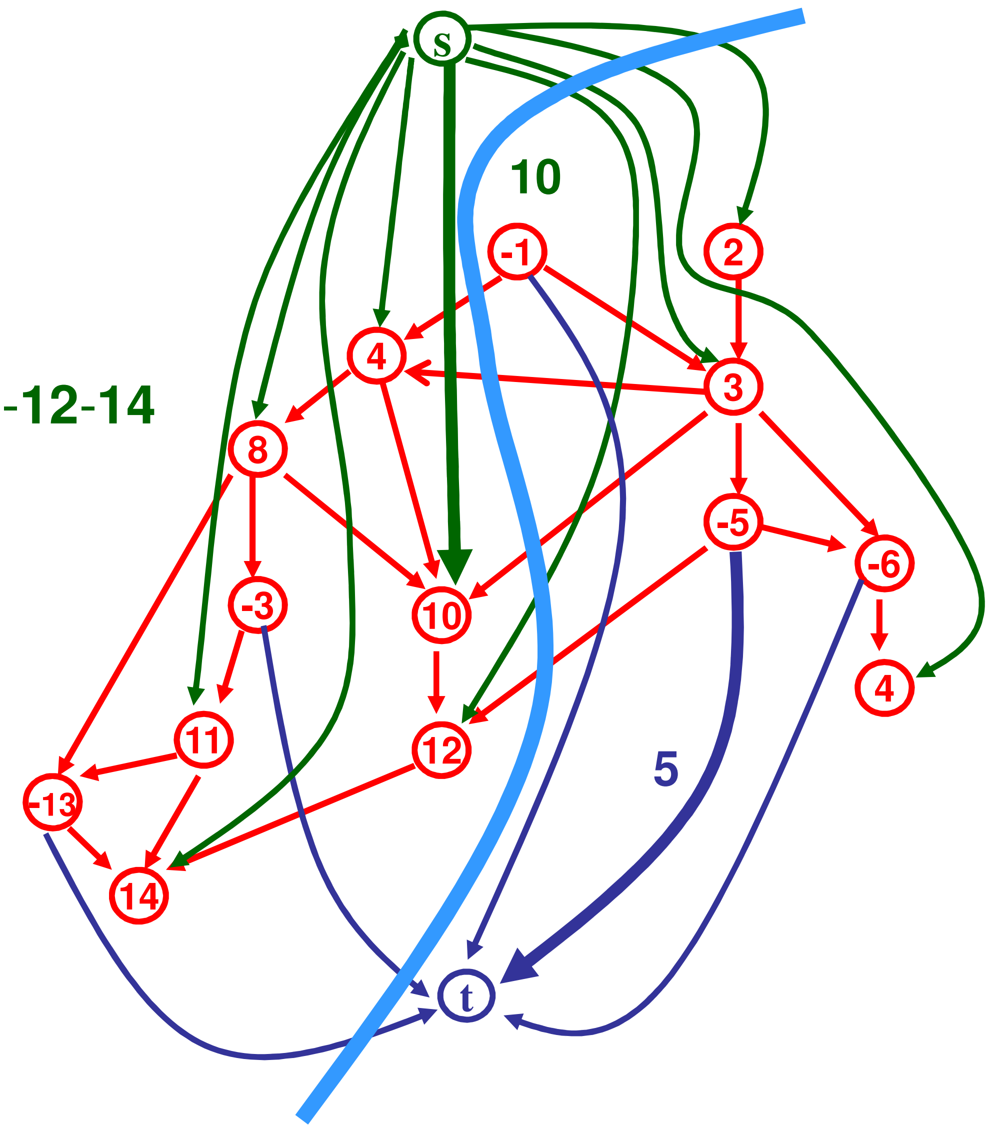
Extended Graph **G'**

Capacity **C**



Extended Graph G'

Cut value
=13+3+2+3+4
=13+3
+C-4-8-10-11-12-14



Project Selection

- **Claim** Any s-t-cut (S,T) in G' such that $A=S-\{s\}$ satisfies precedence constraints has capacity

$$c(S,T)=C - \sum_{v\in A} p(v) = C - \text{Profit}(A)$$

- **Corollary** A minimum cut (S,T) in G' yields an optimal solution $A=S-\{s\}$ to the profit selection problem

- **Algorithm** Compute maximum flow f in G', find the set S of nodes reachable from s in G'_f and return $S-\{s\}$

Proof of Claim

- $A = S - \{s\}$ satisfies precedence constraints
  - No edge of G crosses forward out of A since those edges have capacity $C+1$
  - Only forward edges cut are of the form (v,t) for $v \in A$ or (s,v) for $v \notin A$
  - The (v,t) edges for $v \in A$ contribute
    $$\sum_{v \in A: p(v)<0} -p(v) = -\sum_{v \in A: p(v)<0} p(v)$$
  - The (s,v) edges for $v \notin A$ contribute
    $$\sum_{v \notin A:\ p(v) \geq 0} p(v) = C - \sum_{v \in A:\ p(v) \geq 0} p(v)$$
  - Therefore the total capacity of the cut is
    $$c(S,T) = C - \sum_{v \in A} p(v) = C - \text{Profit}(A)$$