CSE 421 Winter 2025
Lecture 17:
Max Flow Running Time

Nathan Brunelle

http://www.cs.uw.edu/421

Flows

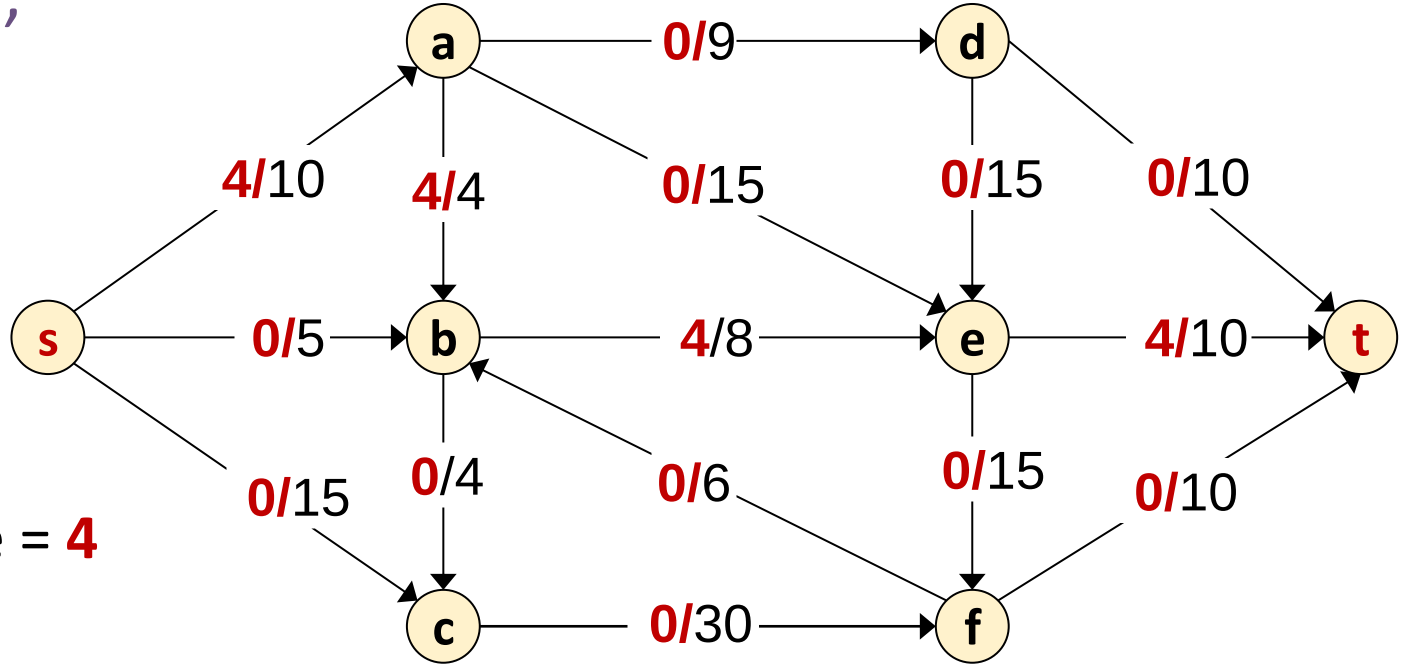
Defn: An s-t flow in a flow network is a function $f\colon E \rightarrow \mathbb{R}$ that satisfies:

- For each $e \in E$: $0 \leq f(e) \leq c(e)$ [capacity constraints]
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [flow conservation]

Defn: The value of flow f,

$$v(f) = \sum_{e \text{ out of } s} f(e)$$

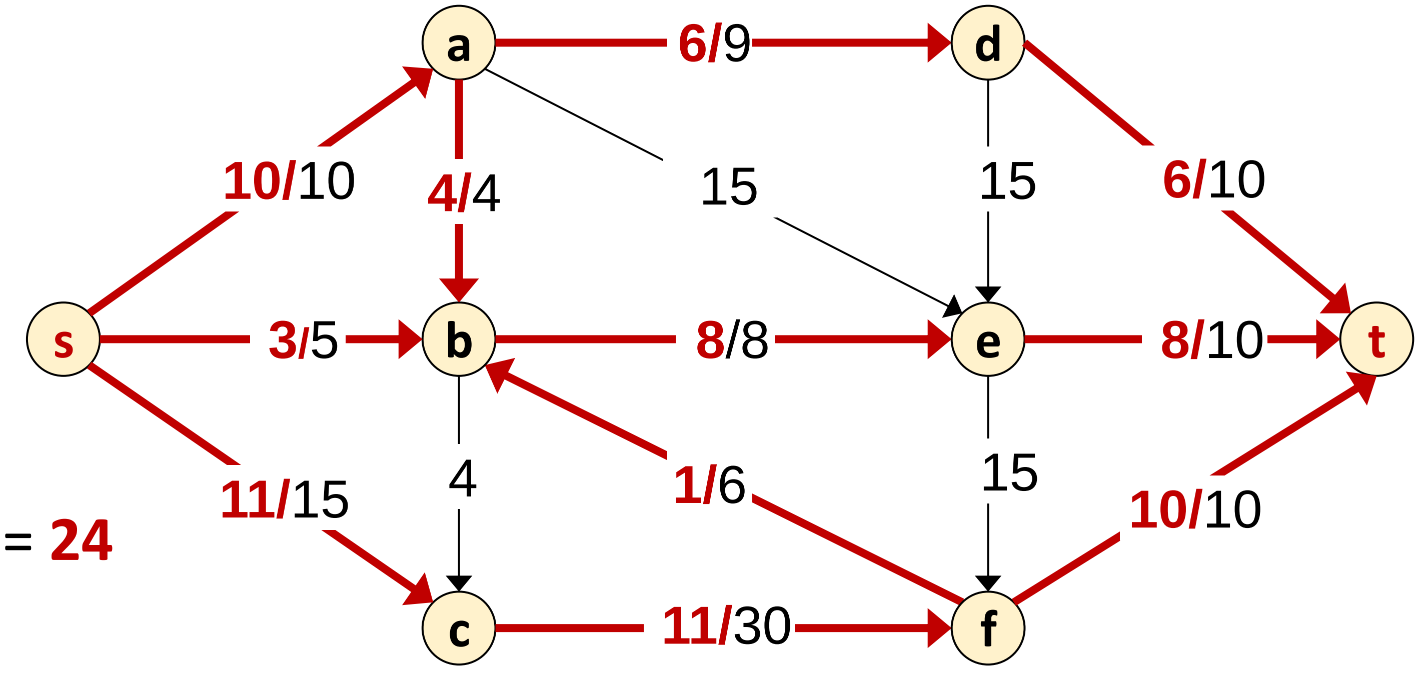
value = 4



Maximum Flow Problem

Given: a flow network

Find: an s-t flow of maximum value



value = **24**

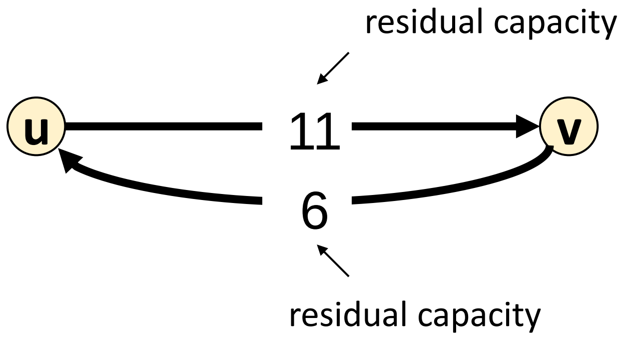
Residual Graphs and Augmenting Paths

Residual edges of two kinds:

- Forward: $e = (u, v)$ with capacity $c_f(e) = c(e) - f(e)$
  - Amount of extra flow we can add along e
- Backward: $e^{\mathrm{R}} = (v, u)$ with capacity $c_f(e) = f(e)$
  - Amount we can reduce/undo flow along e

Residual graph: $G_f = (V, E_f)$.

- Residual edges with residual capacity $c_f(e) > 0$.
- $E_f = \{e : f(e) < c(e)\} \cup \{e^{\mathrm{R}} : f(e) > 0\}$.



Augmenting Path: Any s-t path P in G_f. Let **bottleneck**(P)$= \min_{e \in P} c_f(e)$.

Ford-Fulkerson idea: Repeat "find an augmenting path P and increase flow by **bottleneck**(P)" until none left.

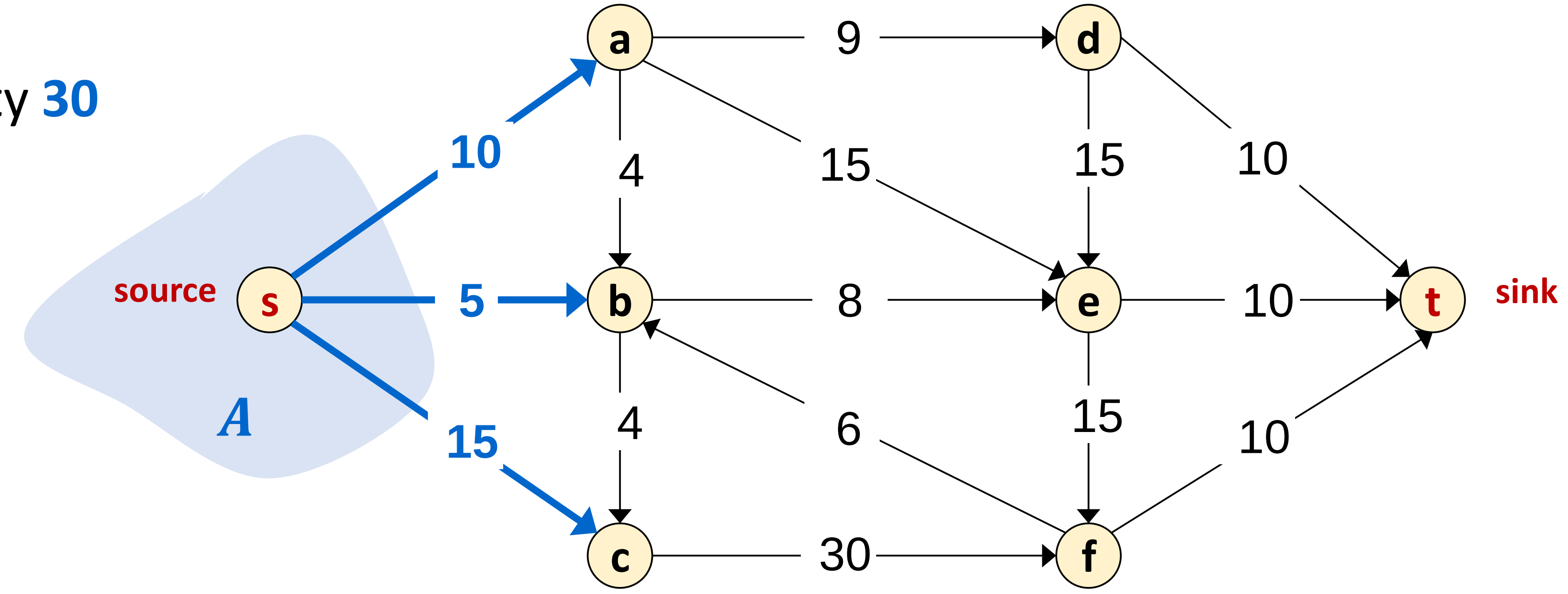
Cuts

Defn: An ***s*-*t* cut** is a partition (A, B) of V with $s \in A$ and $t \in B$.

The **capacity** of cut (A, B) is

$$c(A, B) = \sum_{e \text{ out of } A} c(e)$$

capacity **30**



Minimum Cut Problem

Defn: An s-t cut is a partition (A, B) of V with $s \in A$ and $t \in B$.

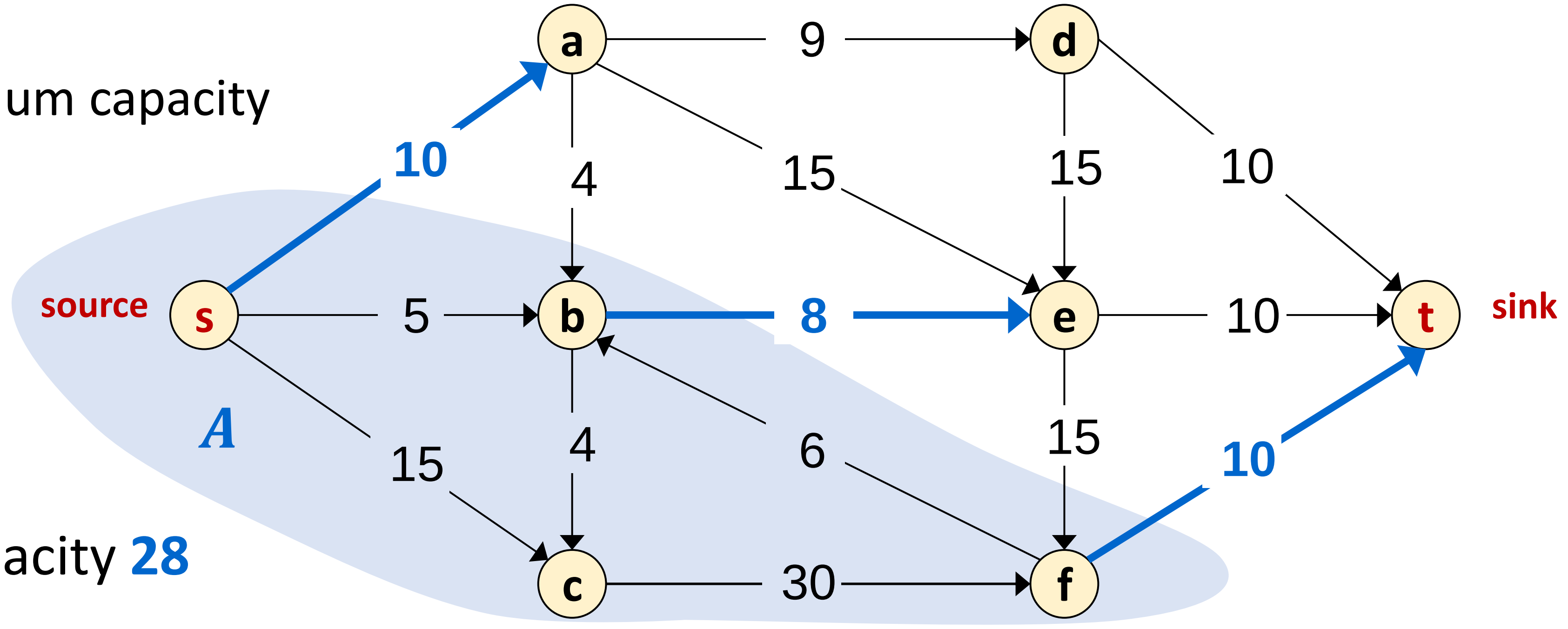
The capacity of cut (A, B) is

$$c(A, B) = \sum_{e \text{ out of } A} c(e)$$

Minimum s-t cut problem:

Given: a flow network

Find: an s-t cut of minimum capacity



capacity **28**

Flows and Cuts

Let $\boldsymbol{f}$ be any $\boldsymbol{s}$-$\boldsymbol{t}$ flow and $(\boldsymbol{A}, \boldsymbol{B})$ be any $\boldsymbol{s}$-$\boldsymbol{t}$ cut:

Flow Value Lemma: The net value of the flow sent across $(\boldsymbol{A}, \boldsymbol{B})$ equals $\boldsymbol{v}(\boldsymbol{f})$.

Intuition: All flow coming from s must eventually reach t, and so must cross that cut

Weak Duality: The value of the flow is at most the capacity of the cut; i.e., $\boldsymbol{v}(\boldsymbol{f}) \leq \boldsymbol{c}(\boldsymbol{A}, \boldsymbol{B})$.

Intuition: Since all flow must cross any cut, any cut's capacity is an upper bound on the flow

Corollary: If $\boldsymbol{v}(\boldsymbol{f}) = \boldsymbol{c}(\boldsymbol{A}, \boldsymbol{B})$ then $\boldsymbol{f}$ is a maximum flow and $(\boldsymbol{A}, \boldsymbol{B})$ is a minimum cut.

Intuition: If we find a cut whose capacity matches the flow, we can't push more flow through that cut because it's already at capacity. We additionally can't find a smaller cut, since that flow was achievable.

Max-Flow Min-Cut Theorem

Augmenting Path Theorem: Flow f is a max flow $\Leftrightarrow$ there are no augmenting paths wrt f

Max-Flow Min-Cut Theorem: The value of the max flow equals the value of the min cut.
[Elias-Feinstein-Shannon 1956, Ford-Fulkerson 1956] **"Maxflow = Mincut"**

Proof: We prove both together by showing that all of these are equivalent:

(i) There is a cut (A, B) such that $v(f) = c(A, B)$.

(ii) Flow f is a max flow.

(iii) There is no augmenting path w.r.t. f.

(i) $\Rightarrow$ (ii): Comes from weak duality lemma.

(ii) $\Rightarrow$ (iii): (by contradiction)
If there is an augmenting path w.r.t. flow f then we can improve f. Therefore f is not a max flow.

(iii) $\Rightarrow$ (i): We will use the residual graph to identify a cut whose capacity matches the flow

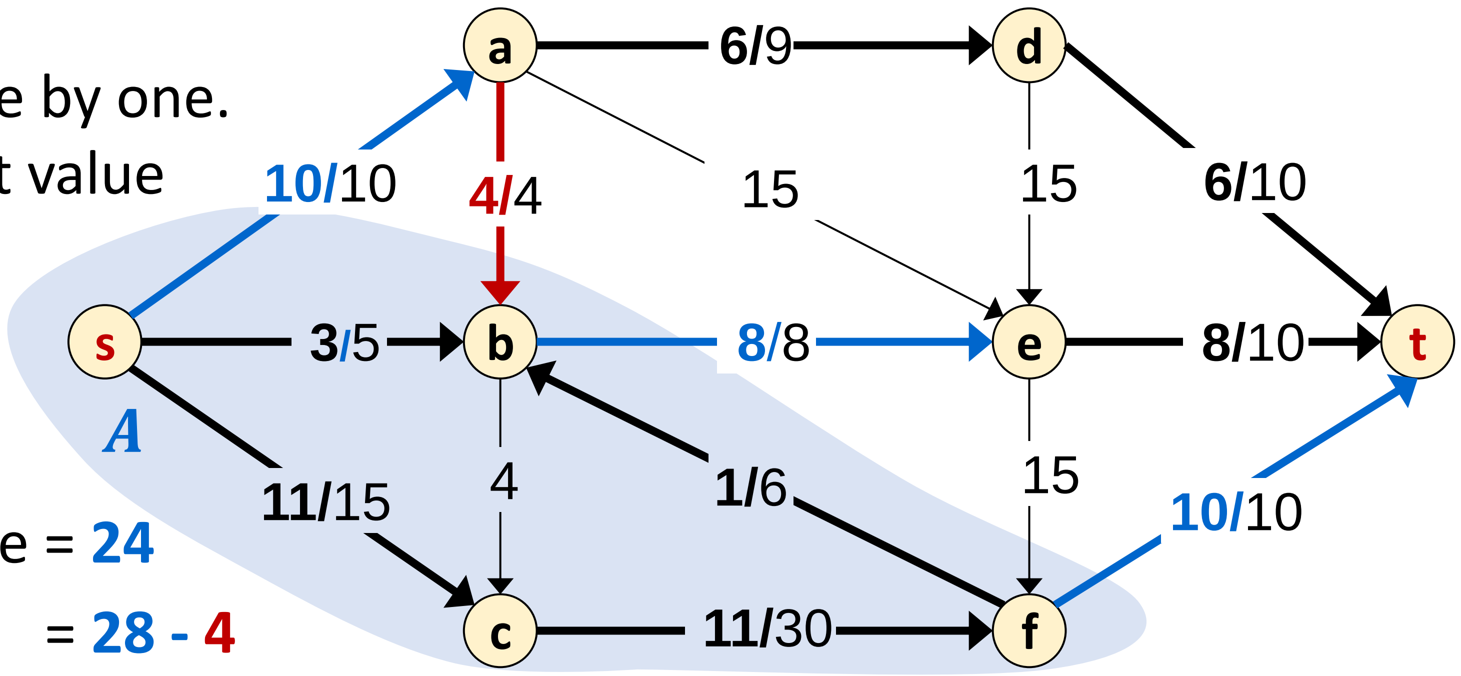
Flow Value Lemma – Idea

Flow Value Lemma: Let f be any s-t flow and (A, B) be any s-t cut. The net value of the flow sent across the cut equals $v(f)$:

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) = v(f)$$

Why is it true?

- Add vertices to s side one by one.
- By flow conservation, net value doesn't change



value = **24**

= **28 - 4**

Flow Value Lemma – Proof

Flow Value Lemma: Let f be any s-t flow and (A, B) be any s-t cut. The net value of the flow sent across the cut equals $v(f)$:

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) = v(f)$$

Proof:

$$v(f) = \sum_{e \text{ out of } s} f(e)$$

$$= \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ into } s} f(e) + \sum_{v \in A - \{s\}} \left[\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ into } v} f(e) \right]$$

$$= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$

$= \mathbf{0}$. No edges into s since it is a source

$= \mathbf{0}$ by flow conservation.

Contributions from internal edges of A cancel.

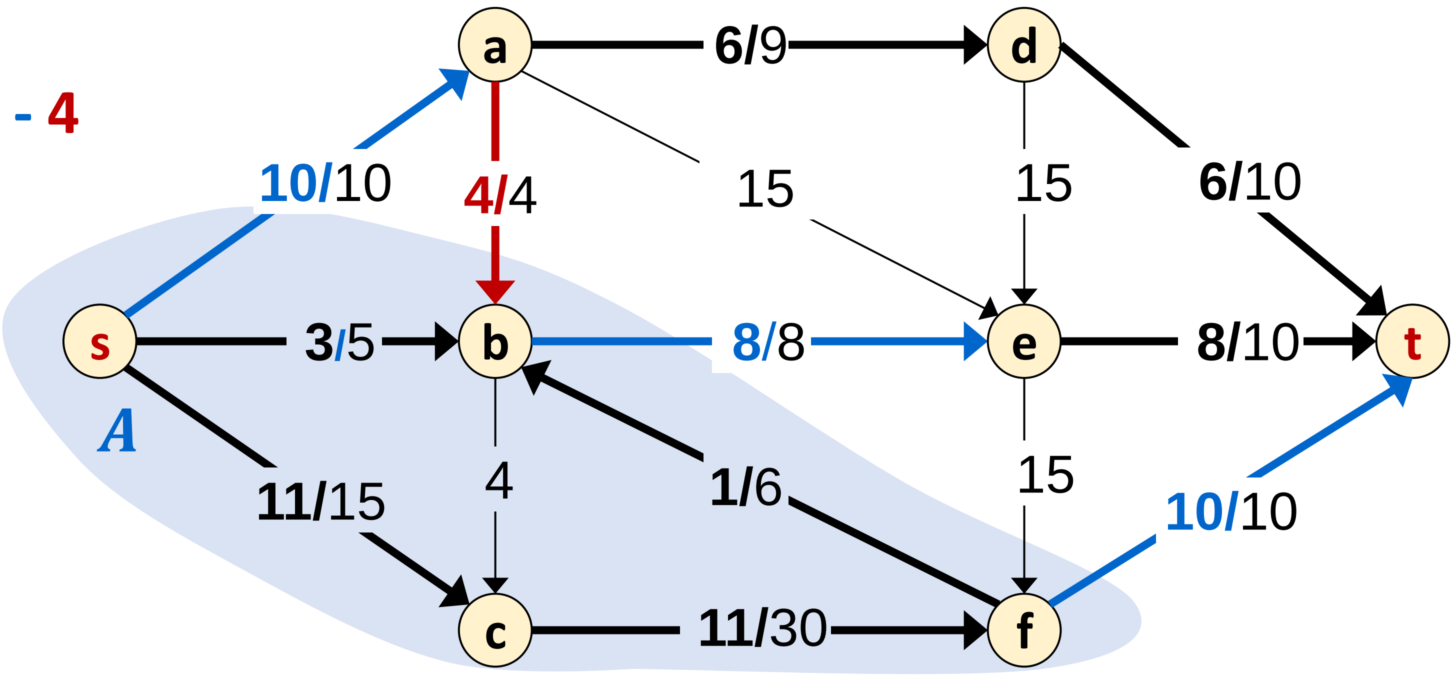
Weak Duality - Idea

(i) ⇒ (ii)

Weak Duality: Let f be any s-t flow and (A, B) be any s-t cut. The value of the flow is at most the capacity of the cut; i.e., $v(f) \leq c(A, B)$:

Value of flow = **24** = **28 - 4**

Capacity of cut = **28**



Weak Duality - Proof

(i) $\Rightarrow$ (ii)

Weak Duality: Let f be any s-t flow and (A, B) be any s-t cut. The value of the flow is at most the capacity of the cut; i.e., $v(f) \leq c(A, B)$.

Proof:

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$

$$\leq \sum_{e \text{ out of } A} f(e) \qquad \text{since } f(e) \geq 0$$

$$\leq \sum_{e \text{ out of } A} c(e) \qquad \text{since } f(e) \leq c(e)$$

$$= c(A, B)$$

■

Proof of Max-Flow Min-Cut Theorem

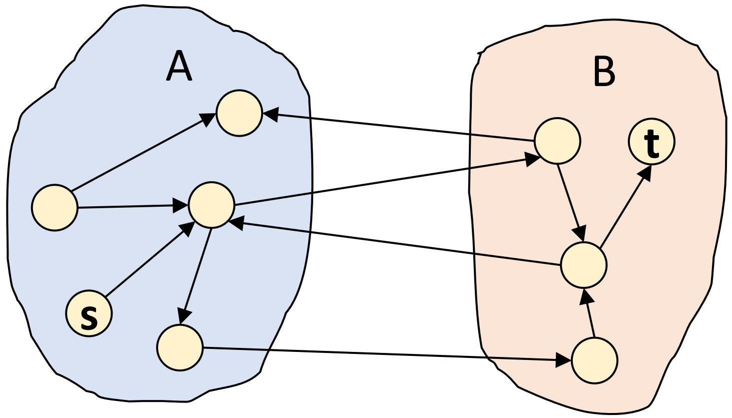
(iii) ⇒ (i):

Claim: If there is no augmenting path w.r.t. f, there is a cut (A, B) s.t. $v(f) = c(A, B)$.

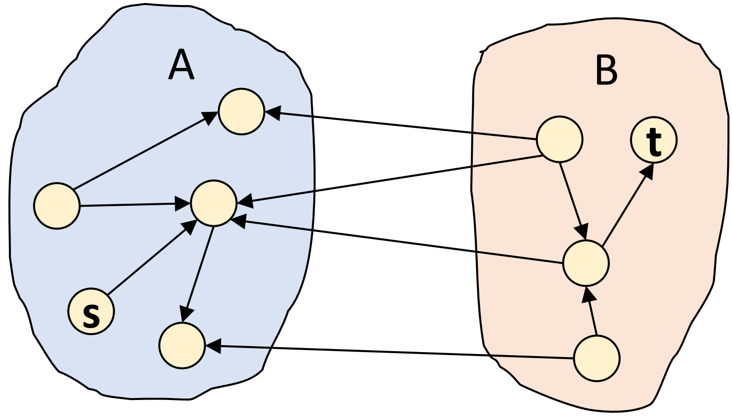
Proof of Claim: Let f be a flow with no augmenting paths.

Let A be the set of vertices reachable from s in residual graph G_f.

- By definition of A, $s \in A$.
- Since no augmenting path (s-t path in G_f), $t \notin A$.



original network



residual graph

Proof: Identifying the Min Cut

(iii) ⇒ (i):

Claim: If there is no augmenting path w.r.t. f, there is a cut (A, B) s.t. $v(f) = c(A, B)$.

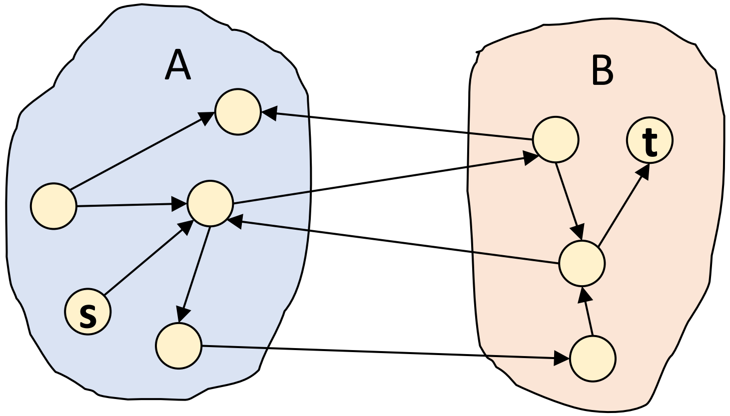
Proof of Claim: Let f be a flow with no augmenting paths.

Let A be the set of vertices reachable from s in residual graph G_f.

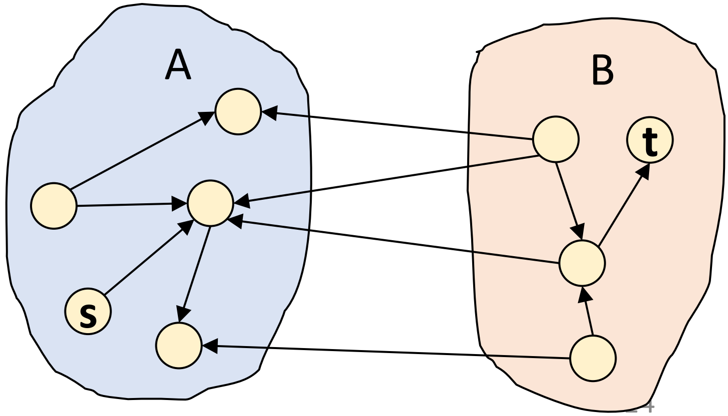
- By definition of A, $s \in A$.
- Since no augmenting path (s-t path in G_f), $t \notin A$.

Then

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) \qquad \text{(by Flow-Value Lemma)}$$



original network



residual graph

Identifying the Min Cut: No Inflow

<u>(iii) ⇒ (i):</u>

Claim: If there is no augmenting path w.r.t. f, there is a cut (A, B) s.t. $v(f) = c(A, B)$.

Proof of Claim: Let f be a flow with no augmenting paths.

Let A be the set of vertices reachable from s in residual graph G_f.

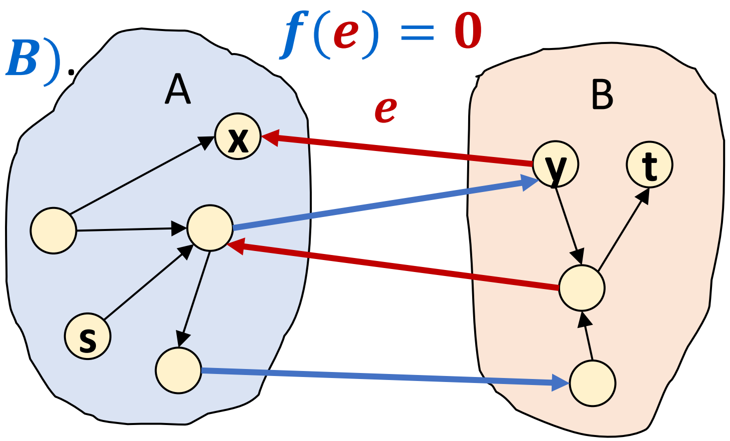
- By definition of A, $s \in A$.
- Since no augmenting path (s-t path in G_f), $t \notin A$.

Then

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$

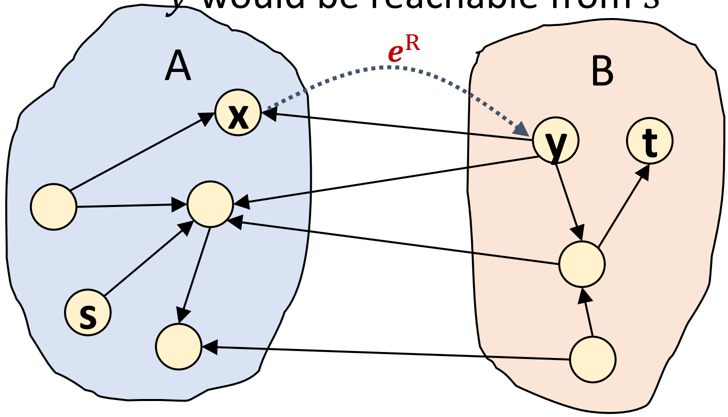
$$= \sum_{e \text{ out of } A} f(e)$$

(**By contradiction:** If an edge going into A had flow then the backward edge would be in the residual graph, so the edge should not cross the cut)



original network

e^R can't exist because then y would be reachable from s



residual graph

Identifying the Min Cut: Saturated Outflow

(iii) ⇒ (i):

Claim: If there is no augmenting path w.r.t. $\boldsymbol{f}$, there is a cut $(\boldsymbol{A}, \boldsymbol{B})$ s.t. $\boldsymbol{v}(\boldsymbol{f}) = \boldsymbol{c}(\boldsymbol{A}, \boldsymbol{B})$.

Proof of Claim: Let $\boldsymbol{f}$ be a flow with no augmenting paths.

Let $\boldsymbol{A}$ be the set of vertices reachable from $\boldsymbol{s}$ in residual graph $\boldsymbol{G_f}$.

- By definition of $\boldsymbol{A}$, $\boldsymbol{s} \in \boldsymbol{A}$.
- Since no augmenting path ($\boldsymbol{s}$-$\boldsymbol{t}$ path in $\boldsymbol{G_f}$), $\boldsymbol{t} \notin \boldsymbol{A}$.

Then

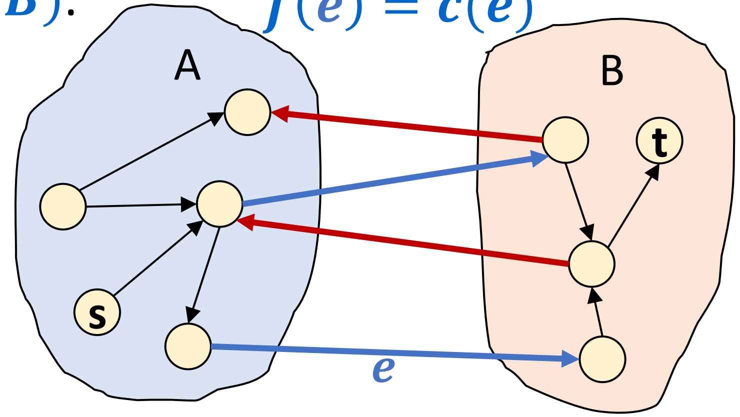
$$\boldsymbol{v}(\boldsymbol{f}) = \sum_{\boldsymbol{e} \text{ out of } \boldsymbol{A}} \boldsymbol{f}(\boldsymbol{e}) - \sum_{\boldsymbol{e} \text{ into } \boldsymbol{A}} \boldsymbol{f}(\boldsymbol{e})$$

$$= \sum_{\boldsymbol{e} \text{ out of } \boldsymbol{A}} \boldsymbol{f}(\boldsymbol{e})$$

$$= \sum_{\boldsymbol{e} \text{ out of } \boldsymbol{A}} \boldsymbol{c}(\boldsymbol{e})$$

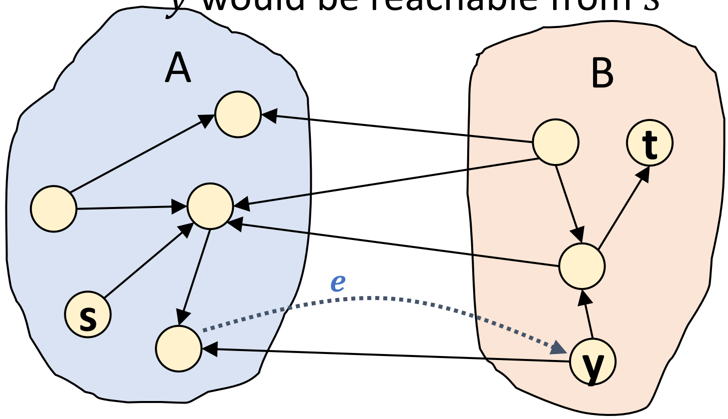
(**By contradiction:** If an edge going out of A had unused capacity then the forward edge would be in the residual graph, so the edge should not cross the cut)

"$\boldsymbol{e}$ is saturated"
No unused capacity on $\boldsymbol{e}$
$\boldsymbol{f}(\boldsymbol{e}) = \boldsymbol{c}(\boldsymbol{e})$



original network

e^R can't exist because then y would be reachable from s



residual graph

Identifying the Min Cut: Conclusion

(iii) ⇒ (i):

Claim: If there is no augmenting path w.r.t. f, there is a cut (A, B) s.t. $v(f) = c(A, B)$.

Proof of Claim: Let f be a flow with no augmenting paths.

Let A be the set of vertices reachable from s in residual graph G_f.

- By definition of A, $s \in A$.
- Since no augmenting path (s-t path in G_f), $t \notin A$.

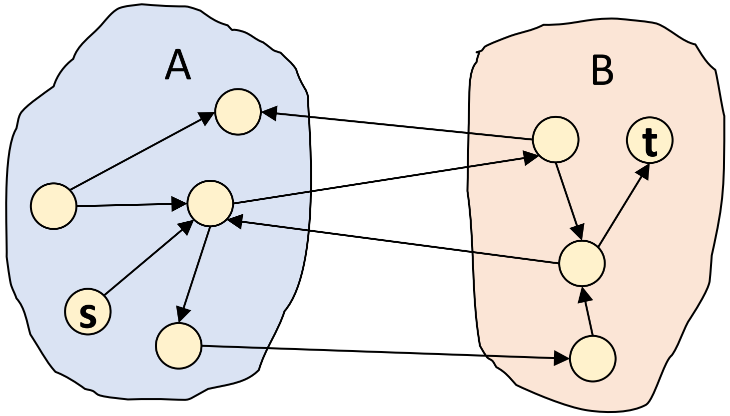
Then

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$

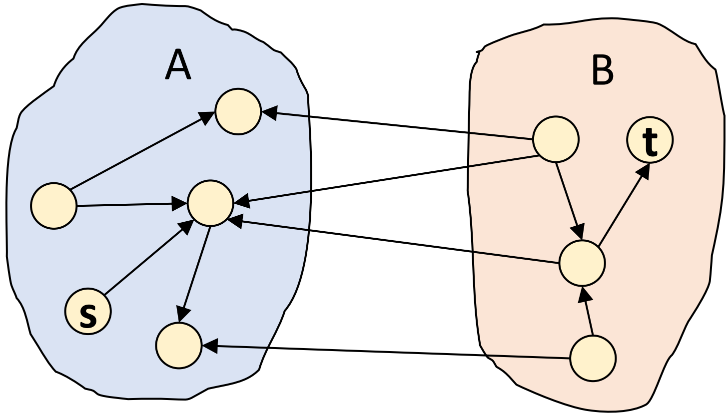
$$= \sum_{e \text{ out of } A} f(e)$$

$$= \sum_{e \text{ out of } A} c(e) \quad = c(A, B)$$

(by Definition)



original network



residual graph

Fork Fulkerson Algorithm

```
FordFulkerson(G, s, t, c){
    for each e ∈ E{
        set f(e) = 0
    }
    calculate residual graph G_f
    while G_f has an s − t path P{
        augment(f, c, P)
        update G_f
    }
    return f
}
```

```
augment(f, c, P){
    b = bottleneck(P)
    for each e ∈ P{
        f(e) += b
        f(e^R) −= b
    }
    return f
}
```

MaxFlow/MinCut & Ford-Fulkerson Algorithm

Augmenting Path Theorem: Flow $\boldsymbol{f}$ is a max flow $\Leftrightarrow$ there are no augmenting paths wrt $\boldsymbol{f}$

Max-Flow Min-Cut Theorem: The value of the max flow equals the value of the min cut.
[Elias-Feinstein-Shannon 1956, Ford-Fulkerson 1956] **"MaxFlow = MinCut"**

Flow Integrality Theorem: If all capacities are integers then there is a maximum flow with all-integer flow values.

Ford-Fulkerson Algorithm: $O(\boldsymbol{m})$ per iteration. With integer capacities each at most $\boldsymbol{C}$ need at most **MaxFlow** $< \boldsymbol{nC}$ iterations for a total of $O(\boldsymbol{mnC})$ time.

19

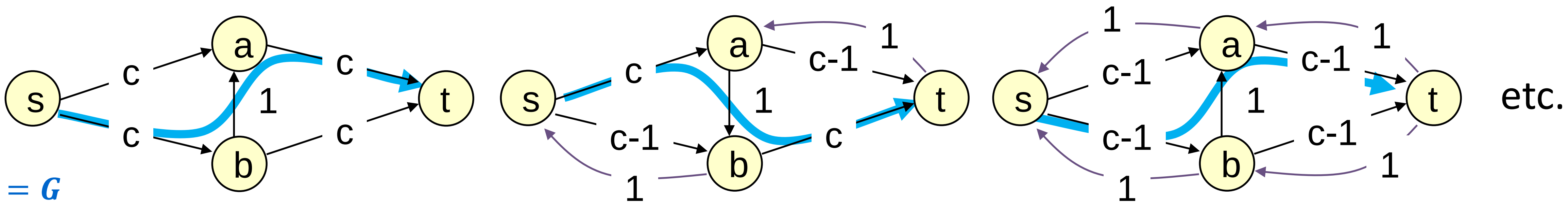
Ford-Fulkerson Efficiency

Worst case runtime $O(mnC)$ with integer capacities $\leq C$.

- $O(m)$ time per iteration.
- At most nC iterations.
- This is “pseudo-polynomial” running time.

- May take exponential time, even with integer capacities:

c $= 10^9$, say



$G_f = G$

Polynomial-Time Variant of Ford-Fulkerson

Use care when selecting augmenting paths.

- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.
- If capacities are irrational, algorithm not guaranteed to terminate!

Goal: Choose augmenting paths so that:

- Can find augmenting paths efficiently.
- Few iterations.

Choose augmenting paths with fewest number of edges. [Edmonds-Karp 1972 , Dinitz 1970]

- Just run BFS to find an augmenting path!

Edmonds-Karp Algorithm (Ford-Fulkerson with BFS)

Use Breadth First Search as the search algorithm to find an s-t path in G_f.

- Using any shortest augmenting path

Theorem: Ford-Fulkerson using BFS terminates in $O(m^2 n)$ time. [Edmonds-Karp, Dinitz]

"One of the most obvious ways to implement Ford-Fulkerson is always polynomial time"

Why might this be good intuitively?

- Longer augmenting paths involve more edges so may be more likely to hit a low residual capacity one which would limit the amount of flow improvement.

The proof uses a completely different idea…

Edmonds-Karp Algorithm (Ford-Fulkerson with BFS)

Analysis Focus:

For any edge e that could be in the residual graph G_f, (either an edge in G or its reverse)

count # of iterations that e is the first bottleneck edge on the augmenting path chosen by the algorithm.

Claim: This can't happen in more than $n/2$ iterations.

Proof: Write $e = (u, v)$.

Show that each time it happens, the distance from s to u in the residual graph G_f is at least 2 more than it was the last time.

This would be enough since the distance is $< n$
(or infinite and hence u isn't reachable) so this can happen at most $n/2$ times.

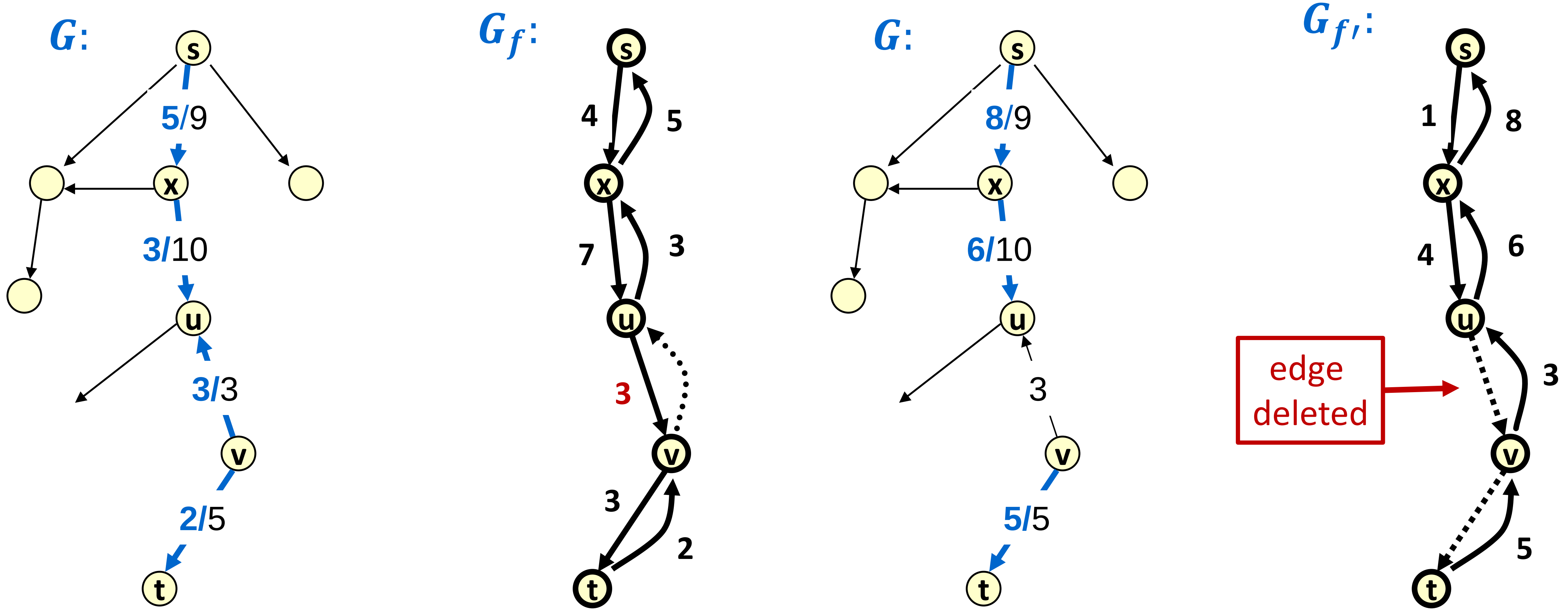
Distances in the Residual Graph

Key Lemma: Let $\boldsymbol{f}$ be a flow, $\boldsymbol{G_f}$ the residual graph, and $\boldsymbol{P}$ be a shortest augmenting path. No vertex is closer to $\boldsymbol{s}$ in the residual graph after augmenting along $\boldsymbol{P}$.

Proof: Augmenting along $\boldsymbol{P}$ can only change the edges in $\boldsymbol{G_f}$ by either:

1. Deleting a forward edge
   - Deleting any edge can never reduce distances
2. Add a backward edge $(\boldsymbol{v}, \boldsymbol{u})$ that is the reverse of an edge $(\boldsymbol{u}, \boldsymbol{v})$ of $\boldsymbol{P}$
   - Since $\boldsymbol{P}$ was a shortest path in $\boldsymbol{G_f}$, the distance from $\boldsymbol{s}$ to $\boldsymbol{v}$ in $\boldsymbol{G_f}$ is already more than the distance from $\boldsymbol{s}$ to $\boldsymbol{u}$. Using the new backward edge $(\boldsymbol{v}, \boldsymbol{u})$ to get to $\boldsymbol{u}$ would be an even longer path to $\boldsymbol{u}$ so it is never on a shortest path to any node in the new residual graph.

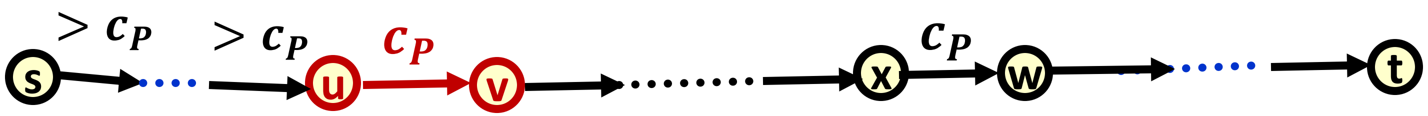
Augmentation vs BFS



First Bottleneck Edges in G_f

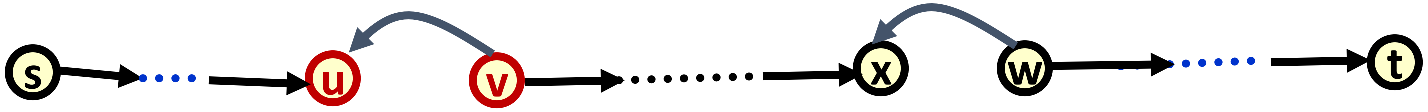
Shortest s-t path P in G_f

Write $c_P = \text{bottleneck}(P)$



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

After augmenting along P, edge (u,v) disappears; but will have edge (v,u)



distance is ≥ 2 larger than before

For (u,v) to be a first bottleneck edge later, it must get added back to the residual graph by augmenting along a shortest path P' containing (v,u) in $G_{f'}$ for some flow f'

Since P' is shortest $d_{f'}(s,u) = d_{f'}(s,v) + 1 \geq d_f(s,v) + 1 = d_f(s,u) + 2$

The next time that (u,v) is first bottleneck edge is even later so distance is at least as large!

Edmonds-Karp Algorithm (Ford-Fulkerson with BFS)

Analysis Focus:

For any edge $\boldsymbol{e}$ that could be in the residual graph $\boldsymbol{G_f}$, (either an edge in $\boldsymbol{G}$ or its reverse)
count # of iterations that $\boldsymbol{e}$ is the first bottleneck edge on the augmenting path chosen by the algorithm.

Claim: This can't happen in more than $\boldsymbol{n/2}$ iterations

Claim ⇒ Theorem:

Only $\boldsymbol{2m}$ edges and $O(\boldsymbol{m})$ time per iteration so $O(\boldsymbol{m^2 n})$ time overall.

History & State of the Art for MaxFlow Algorithms

#	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	$O(m^{3/2} \log(n^2/m) \log U)$ $O(n^{2/3} m \log(n^2/m) \log U)$

Source: Goldberg & Rao, FOCS '97

21	**2013**	**Orlin**	$O(mn)$
22	**2014**	**Lee & Sidford**	$m\sqrt{n} \log^{O(1)} n \log U$
23	**2016**	**Madry**	$m^{10/7} U^{1/7} \log^{O(1)} n$
24	**2021**	**Gao, Liu, & Peng**	$m^{3/2-1/328} \log^{O(1)} n \log U$
25	**2022**	**van den Brand et al.**	$m^{3/2-1/58} \log^{O(1)} n \log U$
26	**2022**	**Chen et al.**	$m^{1+o(1)} \log U$

Tables use U instead of C for the upper bound on capacities

Methods:

Augmenting Paths – increase flow to capacity

Preflow-Push – decrease flow to get flow conservation

Linear Programming – randomized, high probability of optimality

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