

CSE 421

Introduction to Algorithms

Lecture 15: Network Flow

Announcements

Midterm Reminder:


- Date:
 - Next Wednesday, November 8, 6:00 – 7:30 pm in this room
 - Exam designed for a regular class time-slot but this includes extra time to finish.
- Coverage:
 - Up to the end of last Thursday's section on Dynamic Programming
- Sample midterm for practice problems and length coming soon.
 - Will include "summary sheet" available to you on the midterm.
- This week's section will focus on review problems. *(not necessarily exam ready)*
- Zoom review session for Q&A on Tuesday Nov 7 at 4:30 pm.

Maximum Flow and Minimum Cut

Max flow and min cut:

- Two very rich algorithmic problems.
- Cornerstone problems in combinatorial optimization.
- Beautiful mathematical duality.

Nontrivial applications / reductions:

- Data mining.
- Project selection.
- Airline scheduling.
- Bipartite matching.
- Baseball elimination. 
- Image segmentation.
- Network connectivity.
- Strip mining.
- Network reliability.
- Distributed computing.
- Egalitarian stable matching.
- Security of statistical data.
- Network intrusion detection.
- Multi-camera scene reconstruction.
- many many more ...

Origins of Max Flow and Min Cut Problems

Max Flow problem formulation:

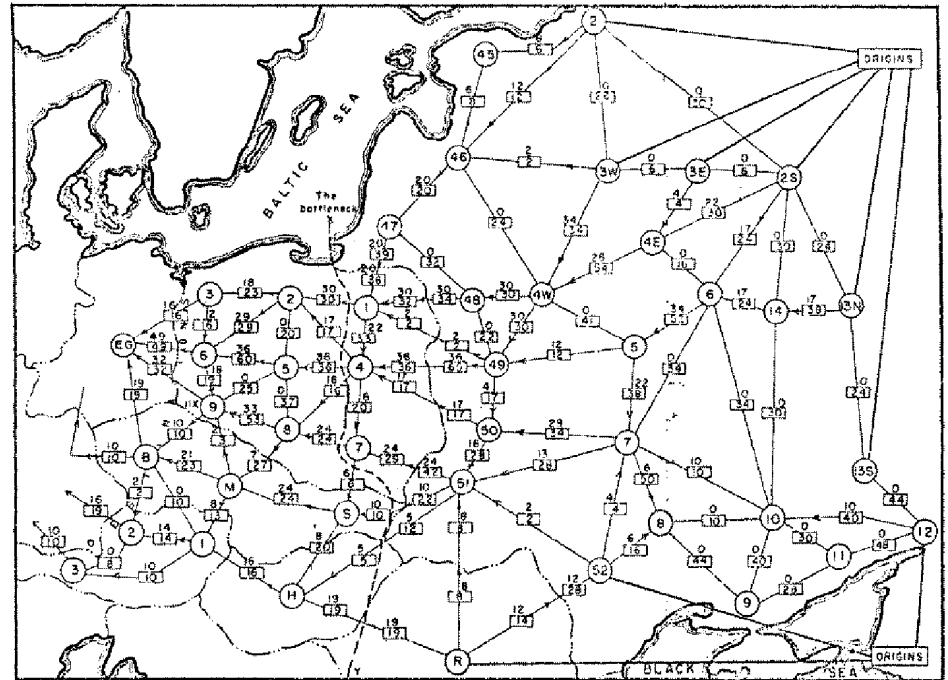
- [Tolstoy 1930] Rail transportation planning for the Soviet Union

Min Cut problem formulation:

- Cold War: US military planners want to find a way to cripple Soviet supply routes
- [Harris 1954] Secret RAND corp report for US Air Force

[Ford-Fulkerson 1955] Problems are equivalent

Soviet Rail Network 1955



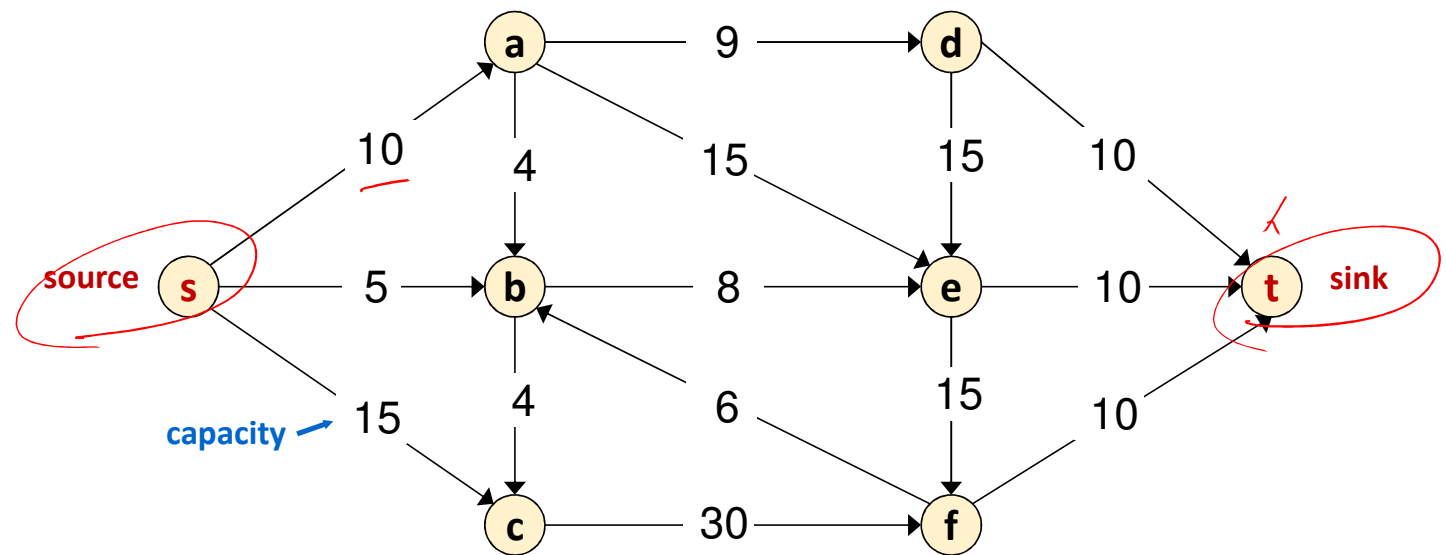
Reference: *On the history of the transportation and maximum flow problems.*
Alexander Schrijver in Math Programming, 91: 3, 2002.

Minimum Cut Problem

Flow Network

Flow network:

- Abstraction for material *flowing* through the edges.
- $G = (V, E)$ directed graph, no parallel edges.
- Two distinguished nodes: $s = \text{source}$, $t = \text{sink}$.
- $c(e) = \text{capacity of edge } e \geq 0$.



Cuts

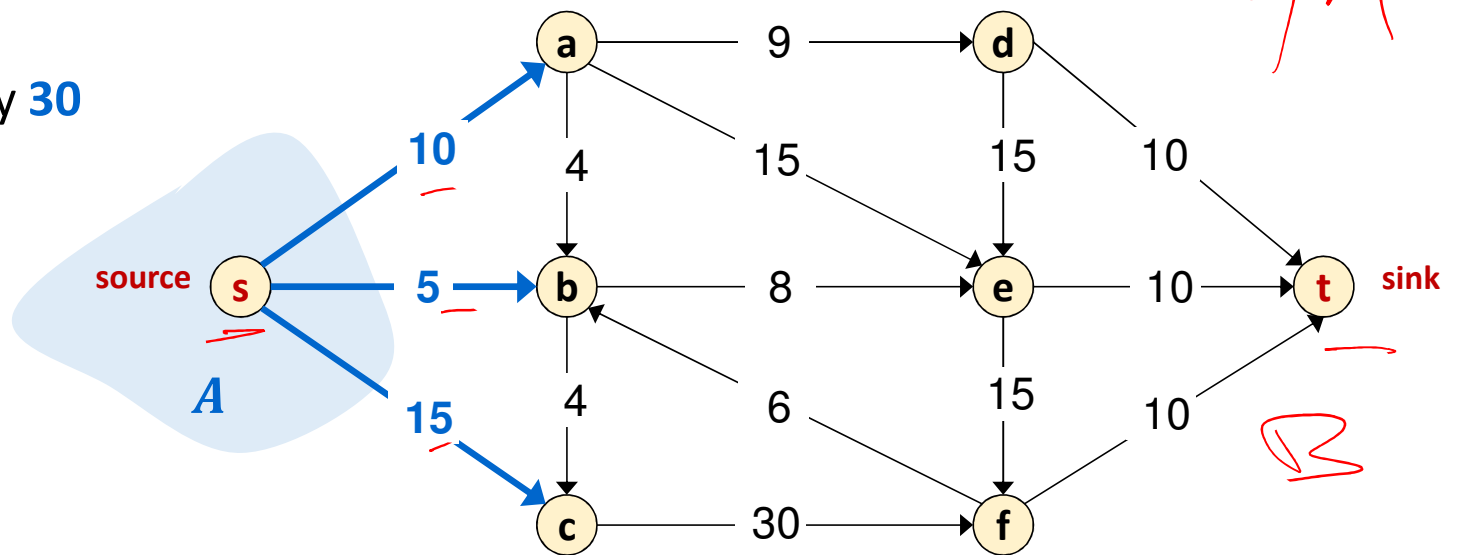
Defn: An **s-t cut** is a partition (A, B) of V with $s \in A$ and $t \in B$.

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

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

s ∈ A
t ∉ A

capacity **30**

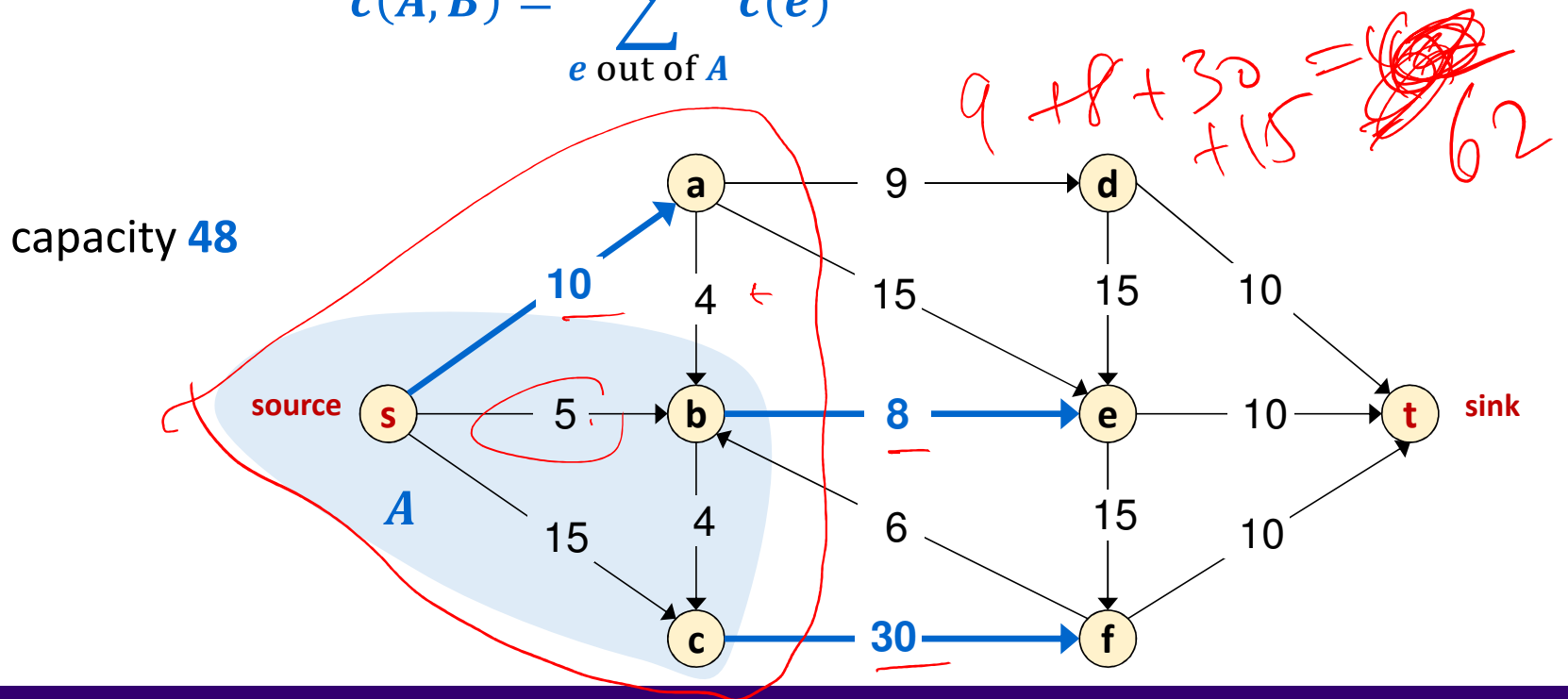


Cuts

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

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



Cuts

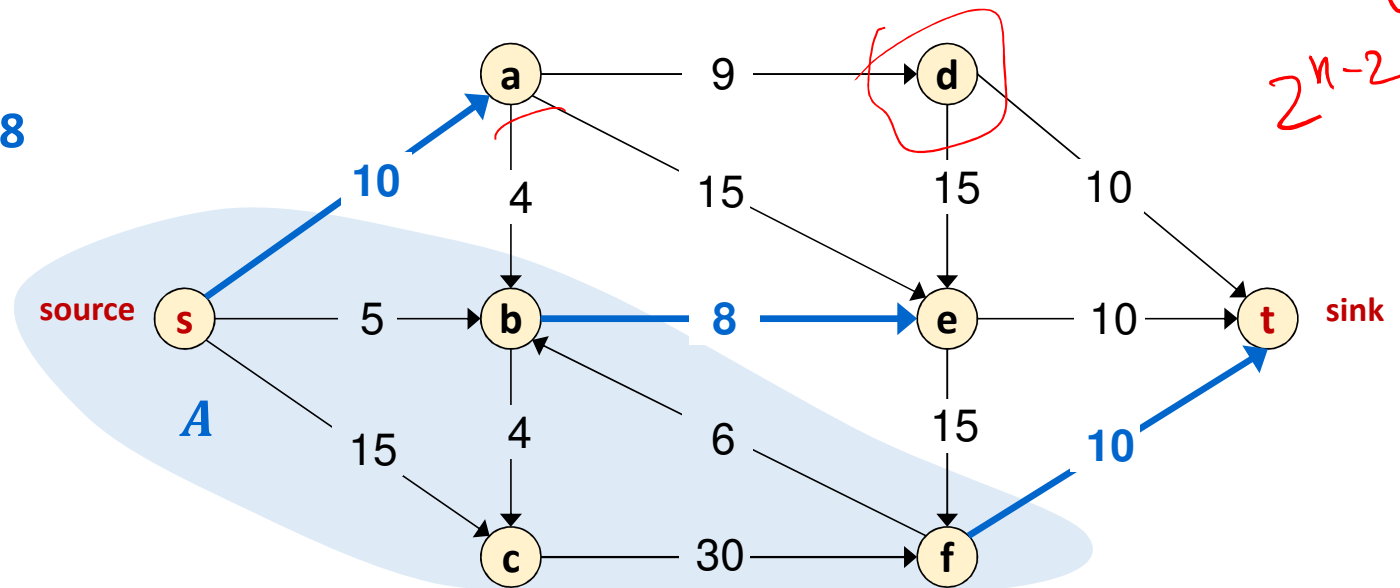
Minimum s-t cut problem:

Given: a flow network

Find: an **s-t** cut of minimum capacity

Brute force
try all sets A
that include s
but not t
 2^{n-2} sets

capacity **28**



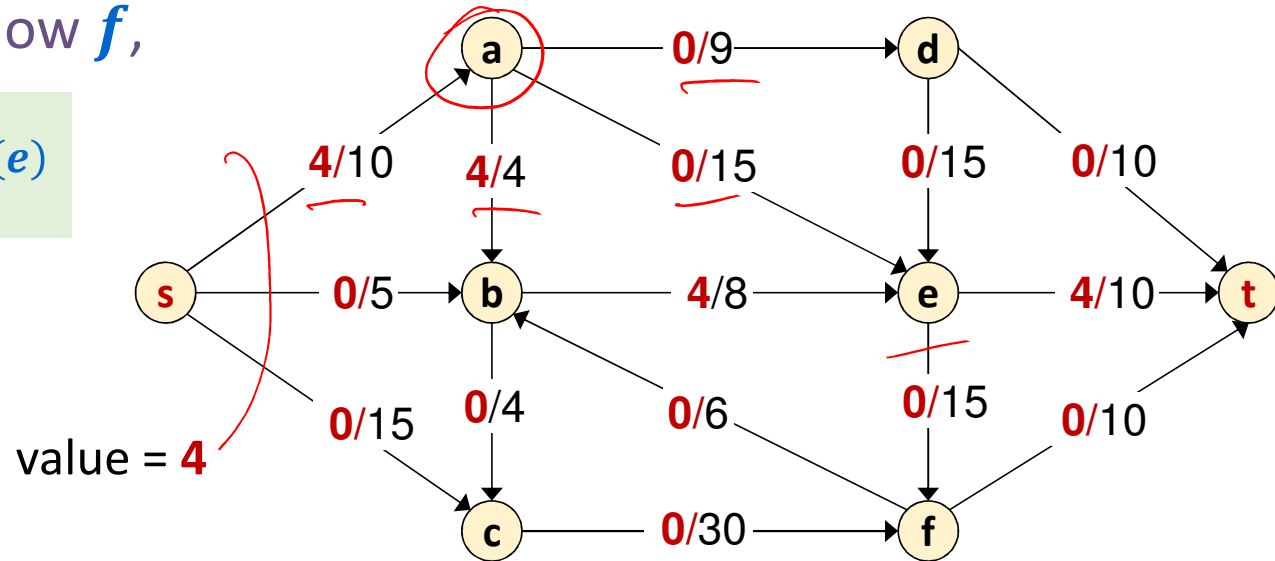
Flows

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

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



Flows

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

- For each $e \in E$: $0 \leq f(e) \leq c(e)$ [capacity constraints]

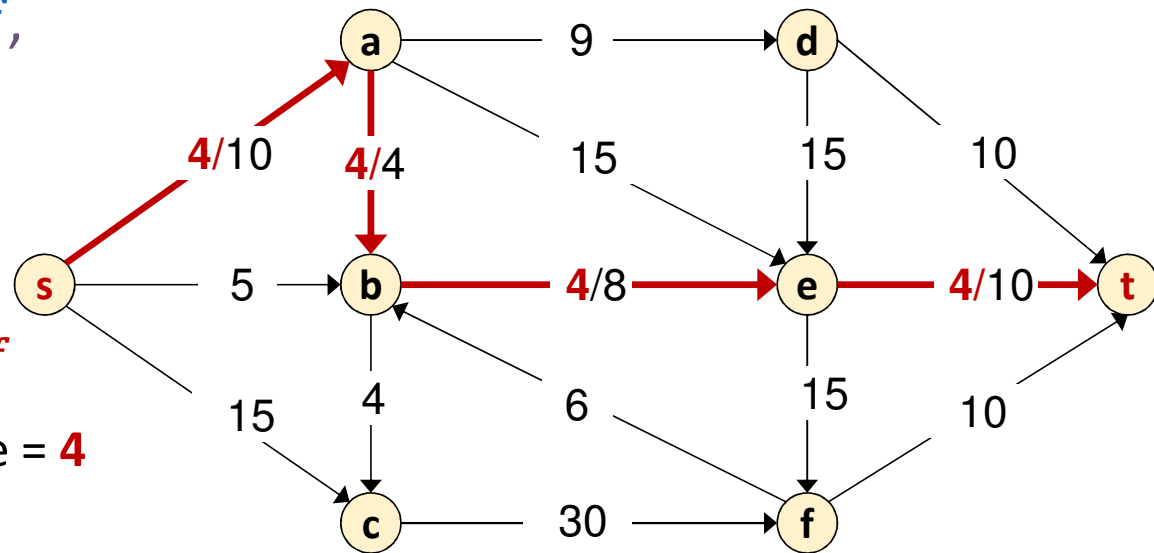
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [flow conservation]

Defn: The **value** of flow f ,

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

Only show non-zero values of f

value = 4



Flows

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

- For each $e \in E$: $0 \leq f(e) \leq c(e)$ [capacity constraints]

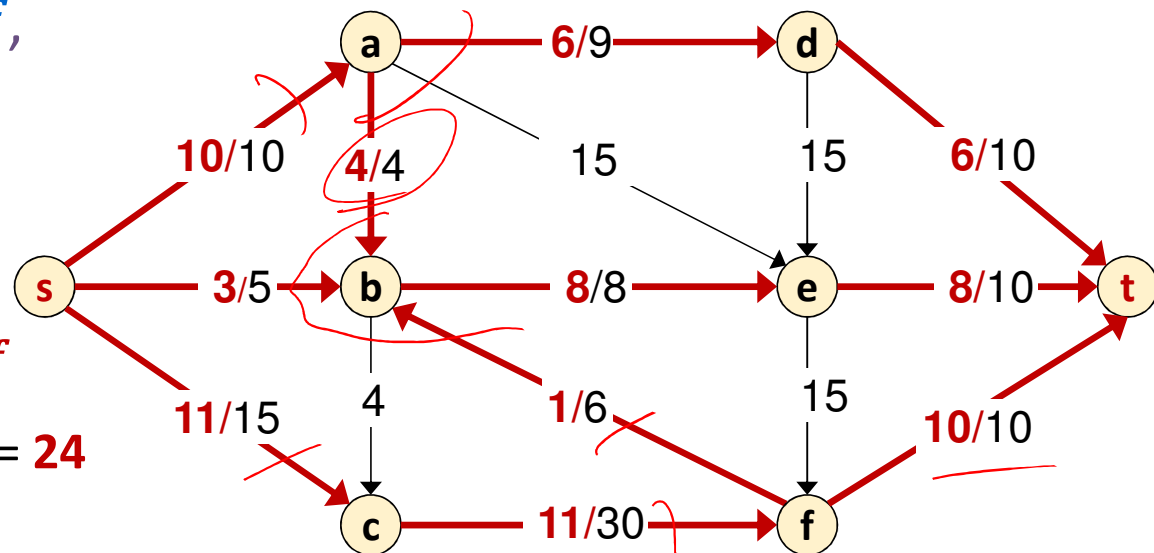
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [flow conservation]

Defn: The **value** of flow f ,

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

Only show non-zero values of f

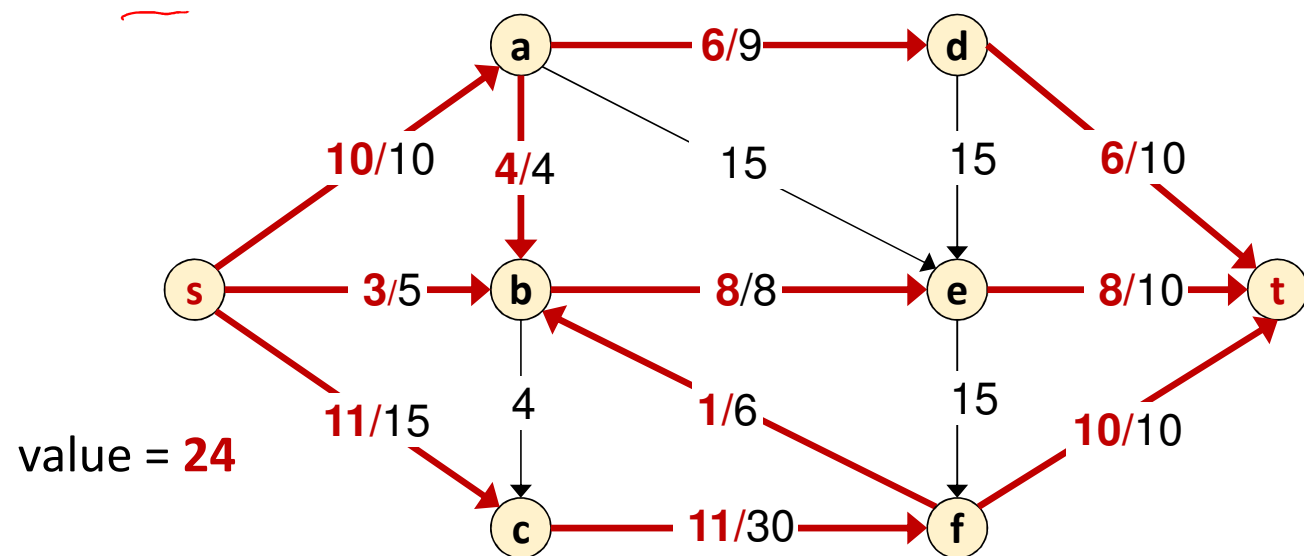
value = **24**



Maximum Flow Problem

Given: a flow network

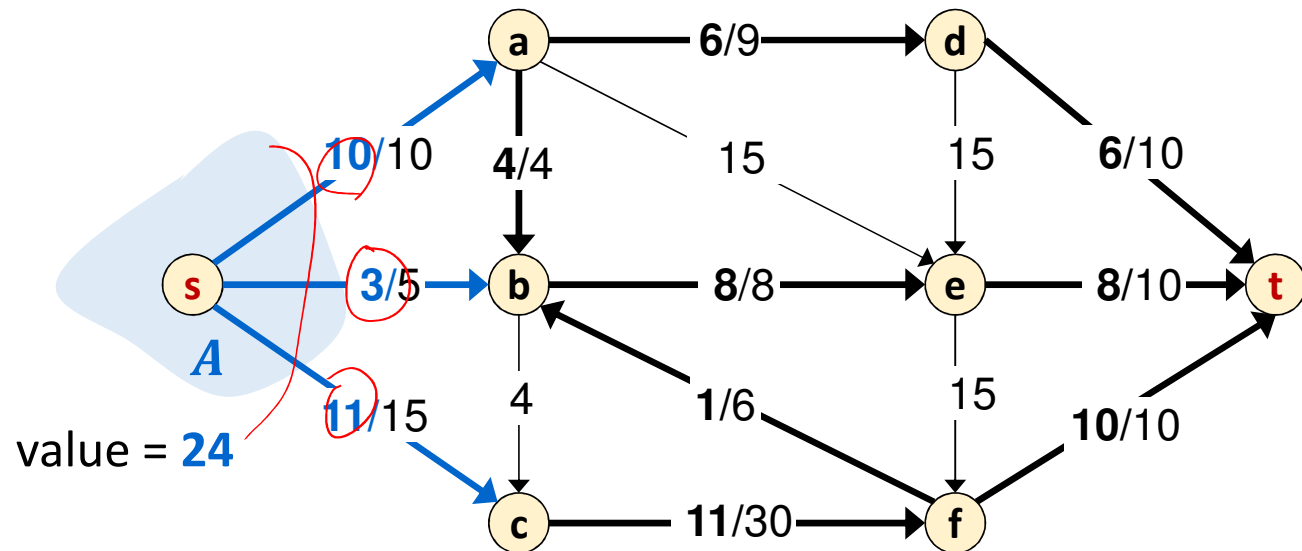
Find: an s - t flow of maximum value



Maximum Flow Problem

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 $v(f)$:

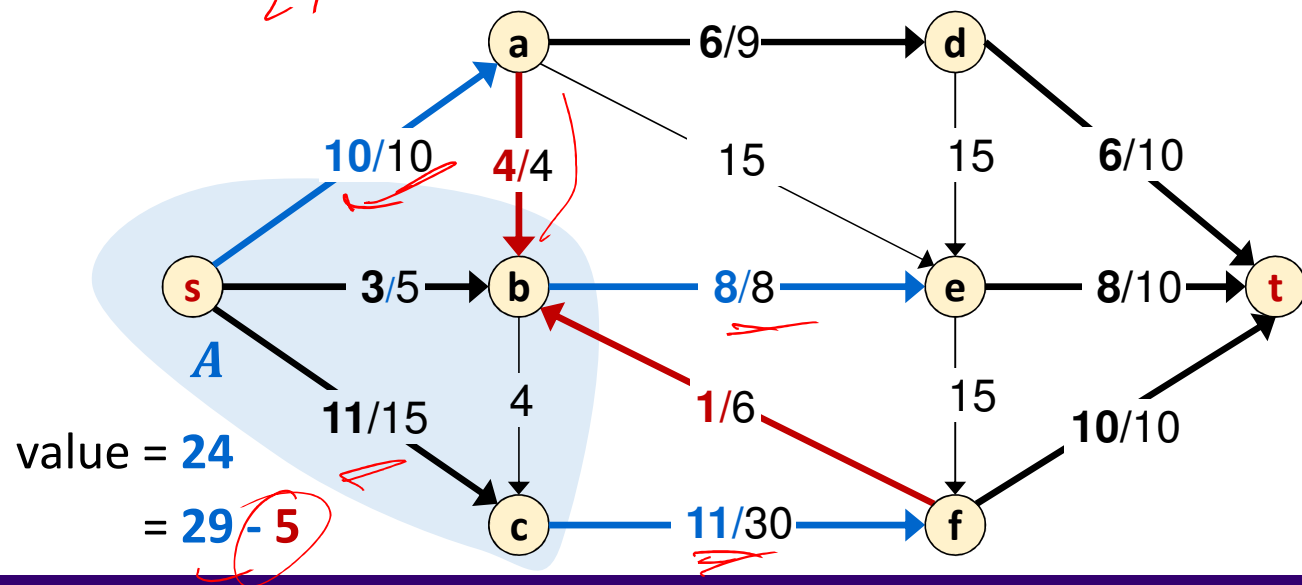
$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) = v(f)$$



Maximum Flow Problem

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 $v(f)$:

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



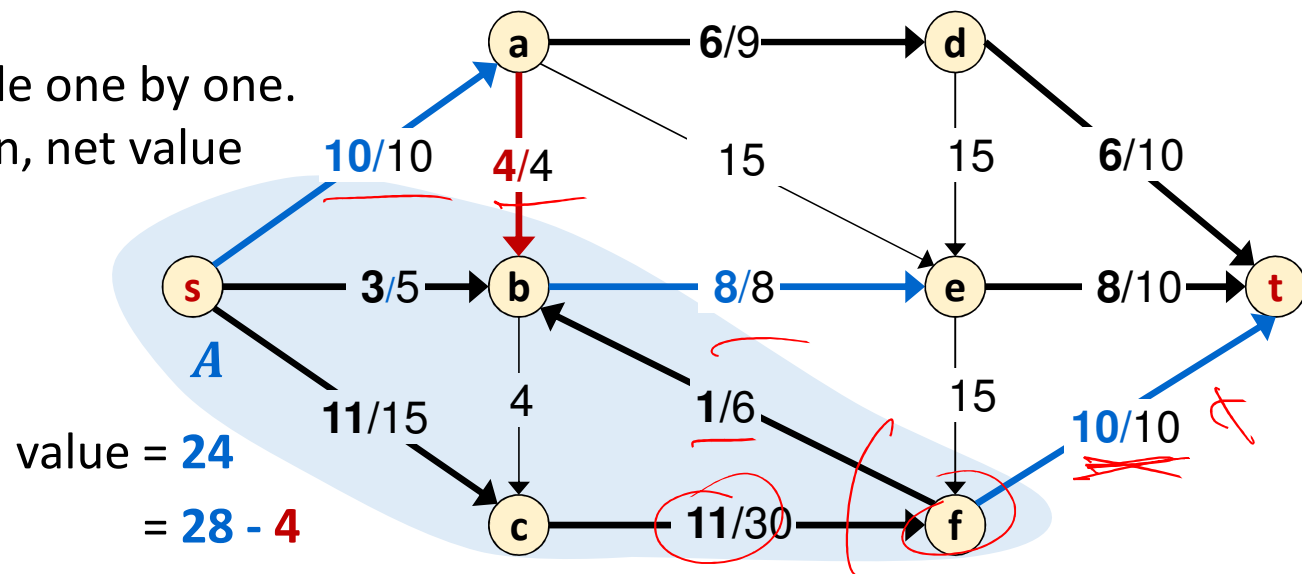
Maximum Flow Problem

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



Maximum Flow Problem

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:

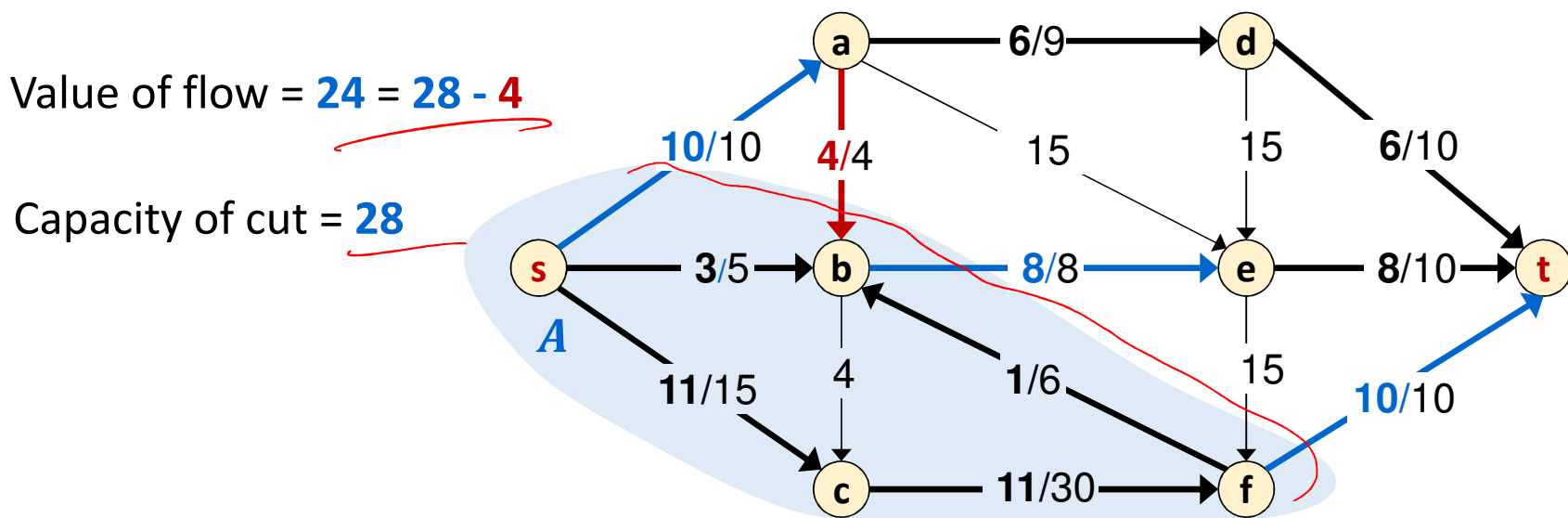
$$\begin{aligned}
 v(f) &= \sum_{e \text{ out of } s} f(e) && = 0. \text{ No edges into } s \text{ since it is a source} \\
 &= \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ into } s} f(e) + \sum_{v \in A - \{s\}} \left[\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ into } v} f(e) \right] \\
 &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) && = 0 \text{ by flow conservation.}
 \end{aligned}$$

Contributions from internal edges of A cancel.



Flows and Cuts

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., $v(f) \leq c(A, B)$:



Flows and Cuts

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:

$$\begin{aligned} v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) \\ &\leq \sum_{e \text{ out of } A} f(e) && \text{since } f(e) \geq 0 \\ &\leq \sum_{e \text{ out of } A} c(e) && \text{since } f(e) \leq c(e) \\ &= c(A, B) \end{aligned}$$

Handwritten red annotations: A circle around the second term in the first line with a question mark and a zero, and a checkmark under the final result.



Certificate of Optimality

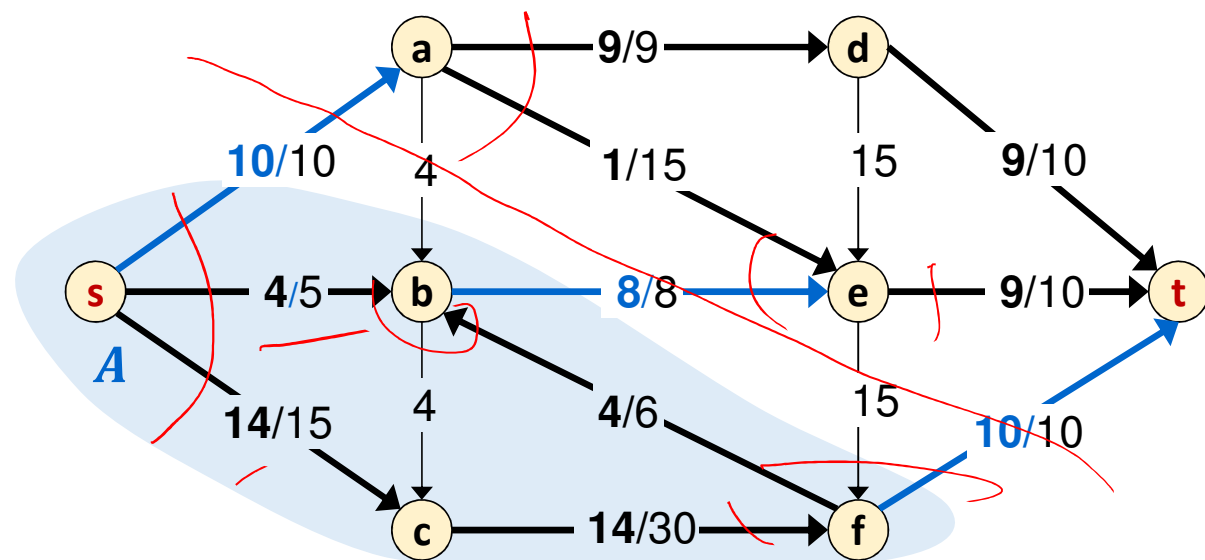
Corollary: Let f be any s - t flow and (A, B) be any s - t cut.

If $v(f) = c(A, B)$ then f is a max flow and (A, B) is a min cut.

Value of flow = **28**

Capacity of cut = **28**

Both are optimal!



Towards a Max Flow Algorithm

What about the following greedy algorithm?

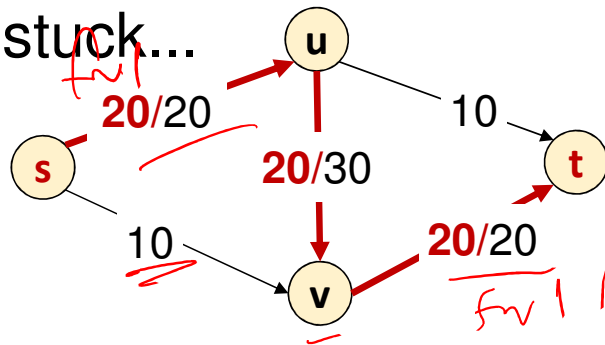
- Start with $f(e) = 0$ for all edges $e \in E$.
- While there is an s - t path P where each edge has $f(e) < c(e)$.
 - “Augment” flow along P ; that is:
 - Let $\alpha = \min_{e \in P} (c(e) - f(e))$
 - Add α to flow on every edge e along path P . (Adds α to $v(f)$.)

Towards a Max Flow Algorithm

What about the following greedy algorithm?

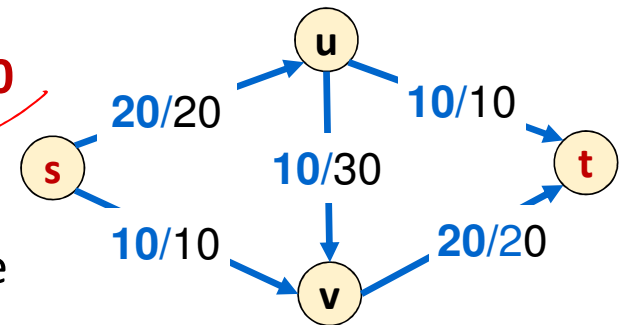
- Start with $f(e) = 0$ for all edges $e \in E$.
- While there is an s - t path P where each edge has $f(e) < c(e)$.
 - “Augment” flow along P ; that is:
 - Let $\alpha = \min_{e \in P} (c(e) - f(e))$
 - Add α to flow on every edge e along path P . (Adds α to $v(f)$.)

Can get stuck...



Has flow value **20**
and no path P

but **30** is possible

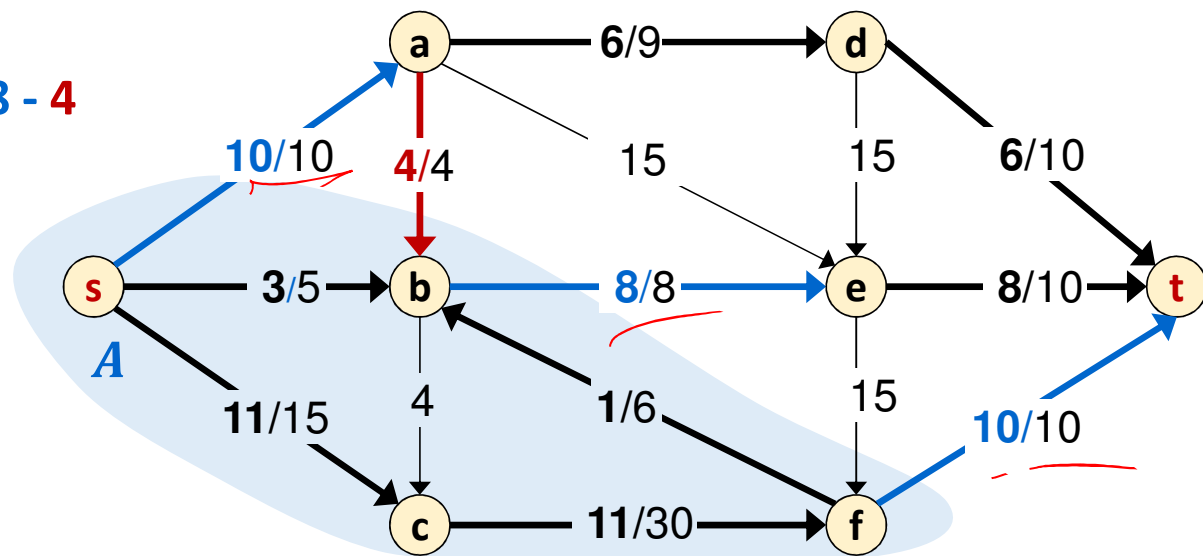


Another Stuck Example

On every $s-t$ path there is some edge with $f(e) = c(e)$:

Value of flow = $24 = 28 - 4$

Capacity of cut = 28



Flows and cuts so far

Let f be any s - t flow and (A, B) be any s - t cut:

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

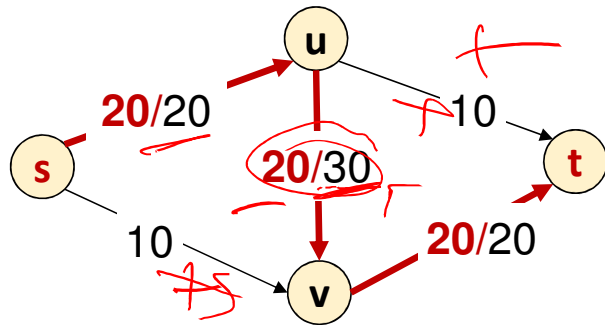
Weak Duality: The value of the flow is at most the capacity of the cut;
i.e., $v(f) \leq c(A, B)$.

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

Augmenting along paths using a greedy algorithm can get stuck.

Next idea: Ford-Fulkerson Algorithm, which applies greedy ideas to a “residual graph” that lets us reverse prior flow decisions from the basic greedy approach to get optimal results!

Greed Revisited: Residual Graph & Augmenting Paths



The only way we could route more flow from s to t would be to reduce the flow from u to v to make room for that amount of extra flow from s to v . But to conserve flow we also would need to increase the flow from u to t by that same amount.

Suppose that we took this flow f as a baseline, what changes could each edge handle?

- We could add up to 10 units along sv or ut or uv
- We could reduce by up to 20 units from su or uv or vt

This gives us a **residual graph** G_f of possible changes where we draw reducing as “sending back”.

