Portland State University

W'21 CS 584/684
Algorithm Design & Analysis

Fang Song

Lecture 13

- Amortized analysis
- Network flow

Recap: minimum spanning tree algorithms

Prim's algorithm

- Start with some node s. Grow a tree T from s outward. Add v to T such that $w(u, v)$ cheapest and $u \in T$.
- Correctness follows by cut property.
- Implementation by priority queue: $O((m + n)\log n)$.

Kruskal's algorithm

- Start with $T = \emptyset$. Insert edges in ascending order of weights, unless it creates a cycle.
- Implementation by disjoint-set (Union-Find) data structure: $O(m \log m + n \log n)$.

An excursion to data structures & amortized analysis

Disjoint-set data structure

- **Goal. Three operations on a collection of disjoint sets.**
  - Make-set(x): create a singleton set containing x.
  - Find-set(x): return the “name” of the unique set containing x.
  - Union(x, y): merge the sets containing x and y respectively.

- **Performance parameters.**
  - k : number of calls to the three operations.
  - n: number of elements.

Simple implementation by an array

- **Array $Component[x]$: name of the set containing x.**
  - FIND(x): $O(1)$.
  - `UNION`(x, y): $\Theta(n)$ update all nodes in sets containing x and y.

- **Some improvement**
  - Maintain the list of elements in each set.
  - Choose the name for the union to be the name of the larger set [so changes are fewer].
  - ☹ `UNION`(x, y): still $\Theta(n)$ in the worst-case.

But this rarely happens …
Can we refine the analysis?

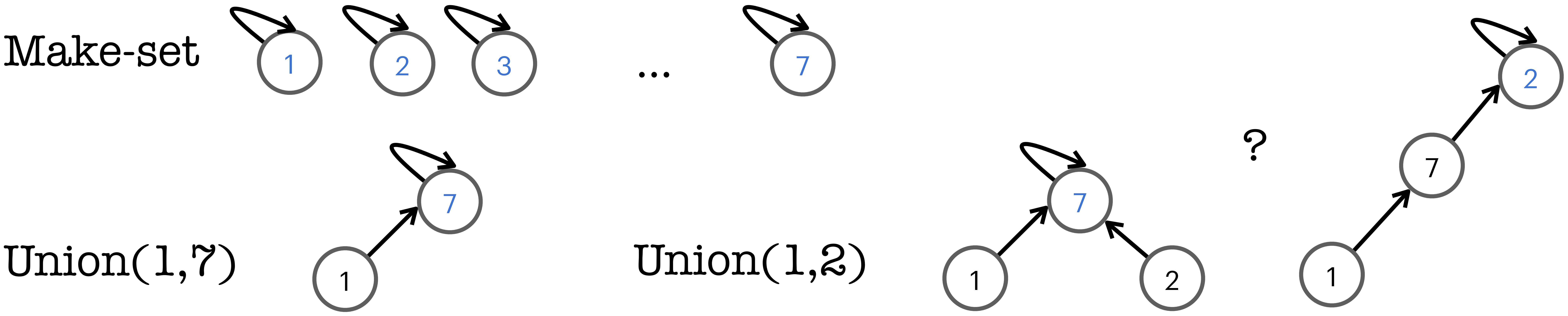
Amortized analysis

- **Determine worst-case running time of a sequence of k data structure operations.**
  - Standard (worst-case) analysis can be too pessimistic if the only way to encounter an expensive operation is when there were lots of previous cheap operations.

- **Theorem. A sequence of k *Union* costs $O(k \log k)$. [contrast w. $O(k^2)$]**
- **Proof. [Aggregate method]**
  - Start from singletons. After k unions, at most $2k$ nodes involved.
  - Any $Component[x]$ changes only when merged with a larger set, i.e., change of name implies doubling of the set size.
  - ➔ For any x, # changes at most $\log_2(2k)$.
  - ➔ $O(k \log k)$ for a sequence of k `Unions` [i.e., each has amortized cost $O(\log k)$].

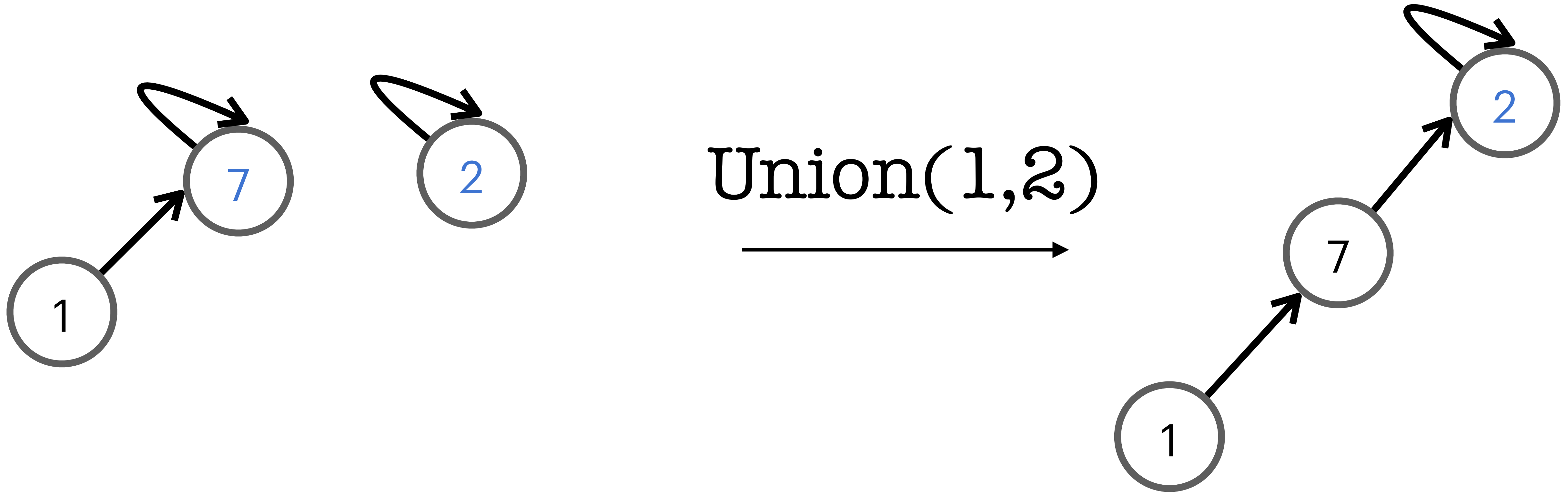
Parent-link representation

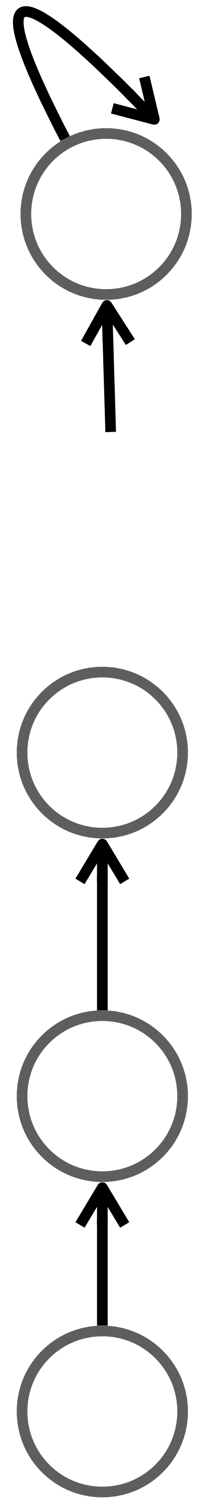
- **Represent each set as a tree**
  - Each element has an explicit parent pointer in the tree.
  - The root (points to itself) serves as the "name".
  - `FIND(x)`: find the root of the tree containing x.
  - `UNION(x, y)`: merge trees containing x and y.



Naive linking

- **Link root of first tree to the root of second tree.**

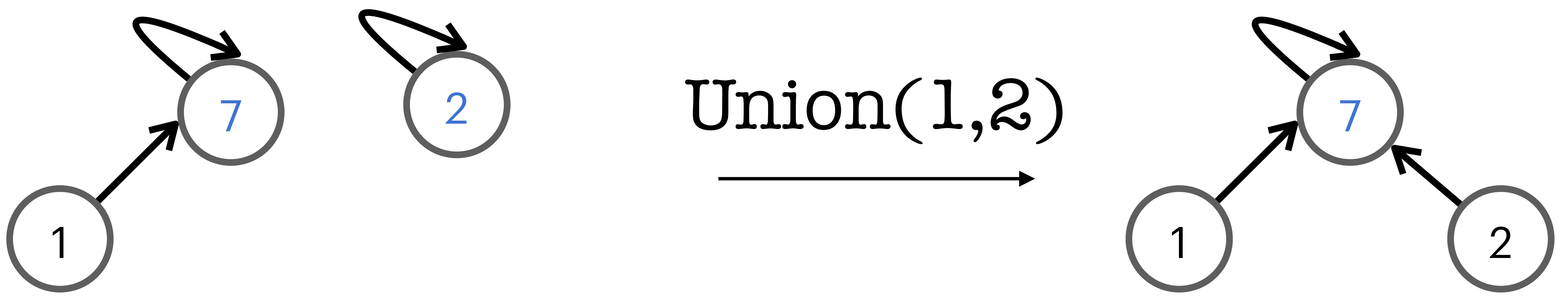




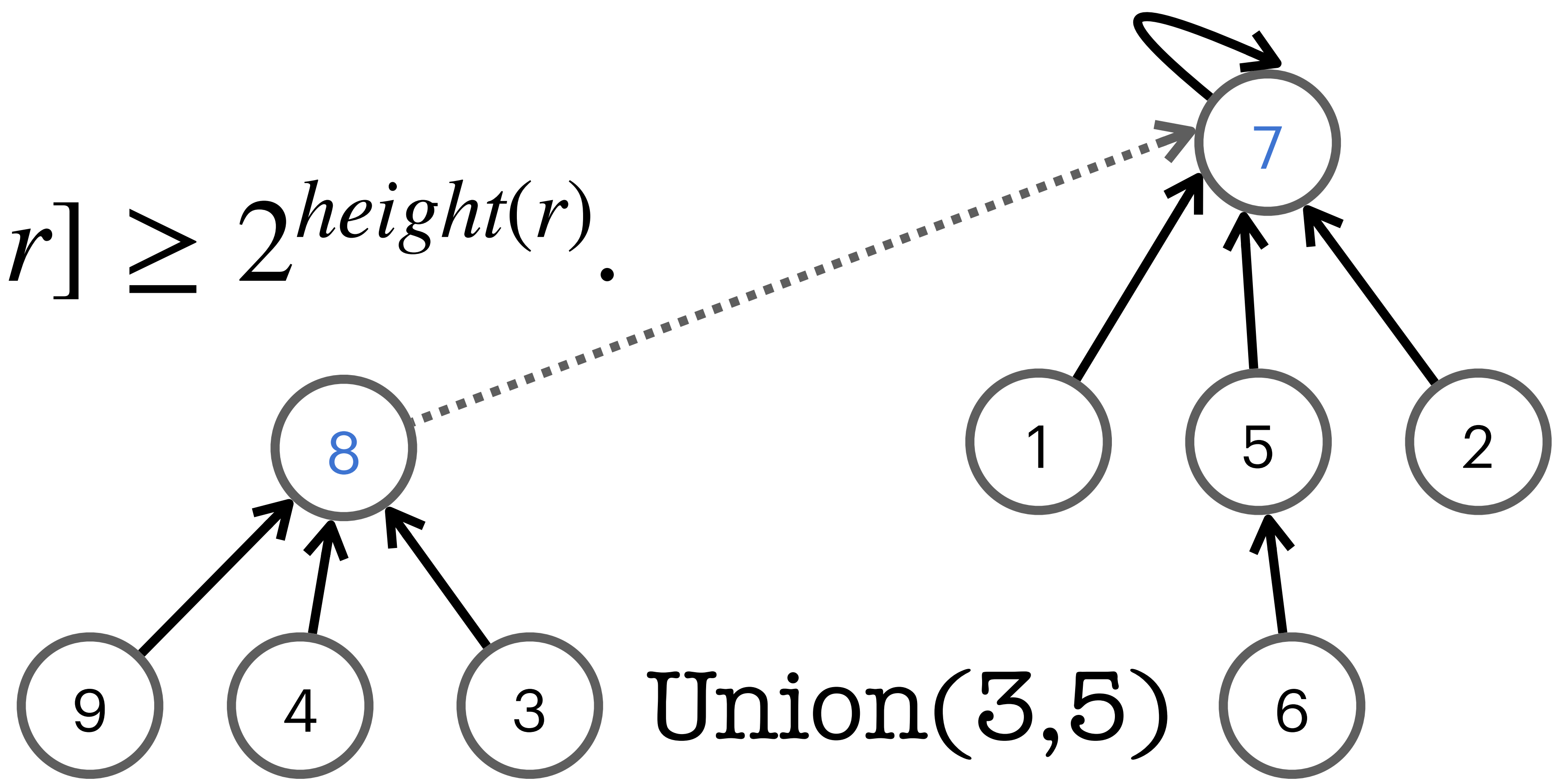
- **Observation. A** Union **can take $\Theta(n)$ in the worst case.**
  - Find root of this tree: determined by the height of the tree.

Link-by-size

- Maintain a tree size (# of nodes in the set) for each root note; link smaller tree to larger.



- **Observation.** Union takes $O(\log n)$ in the worst case.
- **Proof.** [NB. Time $\propto$ height]
  - Show by induction: for every root node r, $size[r] \geq 2^{height(r)}$.
  - $\Rightarrow$ height $\leq \log n$.



Disjoint-set summary

	Array/naive linking	Link-by-size (balanced tree)	Link-by-size w/ path-compressing
Find	$\Theta(1)$	$\Theta(\log n)$	$\Theta(\log n)$
Union	$\Theta(n)$	$\Theta(\log n)$	$\Theta(\log n)$
Amoretized	$\Theta(k \log k)$	$\Theta(k \log k)$	$\Theta(k\alpha(k))$

$\alpha(n)$: inverse Ackermann function;
≤ 4 for any practical cases.

Network flow

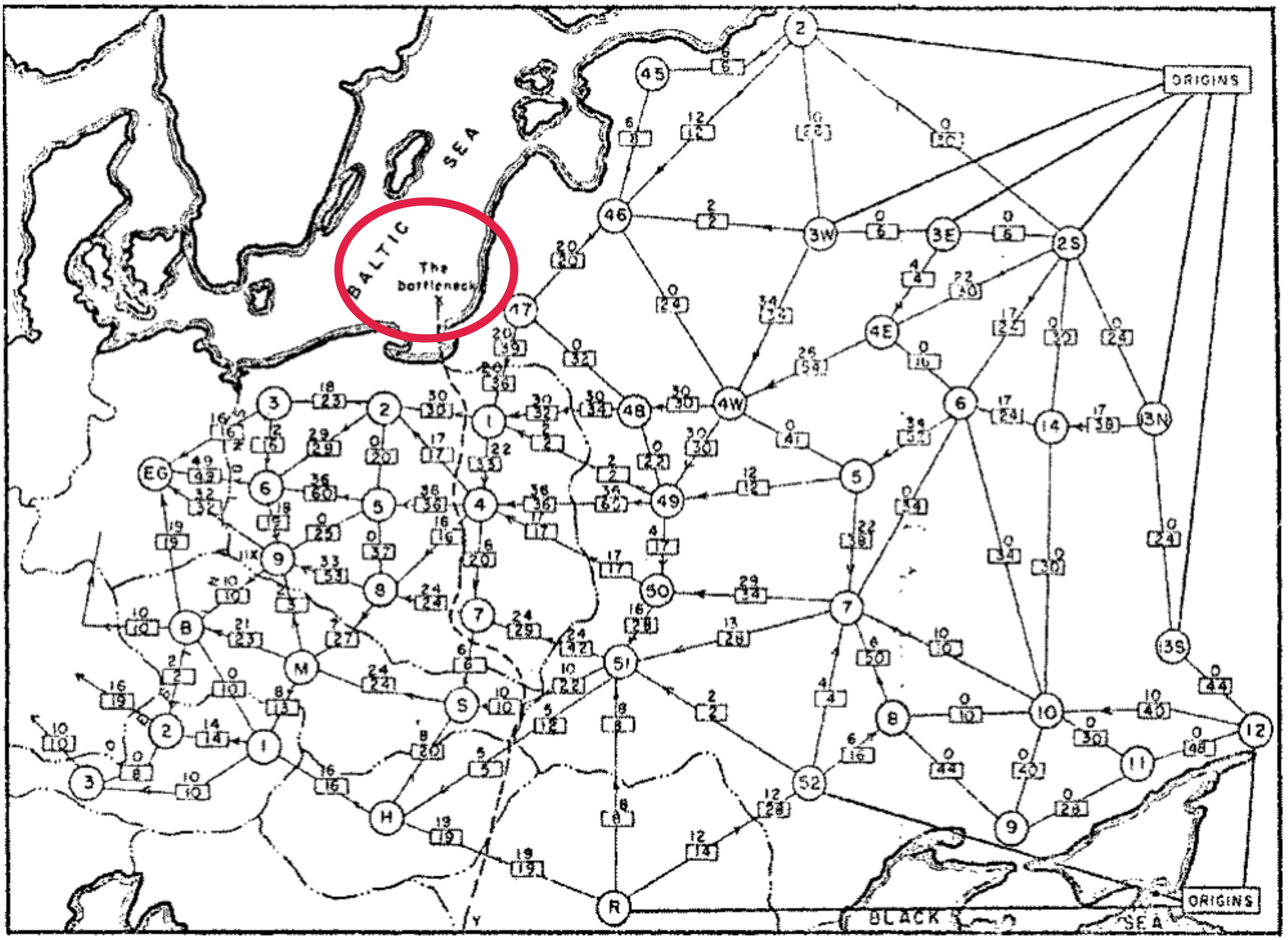


Figure 2
From Harris and Ross [1955]: Schematic diagram of the railway network of the Western Soviet Union and Eastern European countries, with a maximum flow of value 163,000 tons from Russia to Eastern Europe, and a cut of capacity 163,000 tons indicated as "The bottleneck".

Schrijver, Alexander. "On the history of the transportation and maximum flow problems." Mathematical Programming 91.3 (2002): 437-445.

- **Soviet Rail Network 1955**

1. What is the maximum amount of stuff that could be moved from USSR into Europe?

2. What is the cheapest way to disrupt the network by blowing up train tracks (i.e., the bottleneck)?

Maximum flow and minimum cut

Max flow and min cut

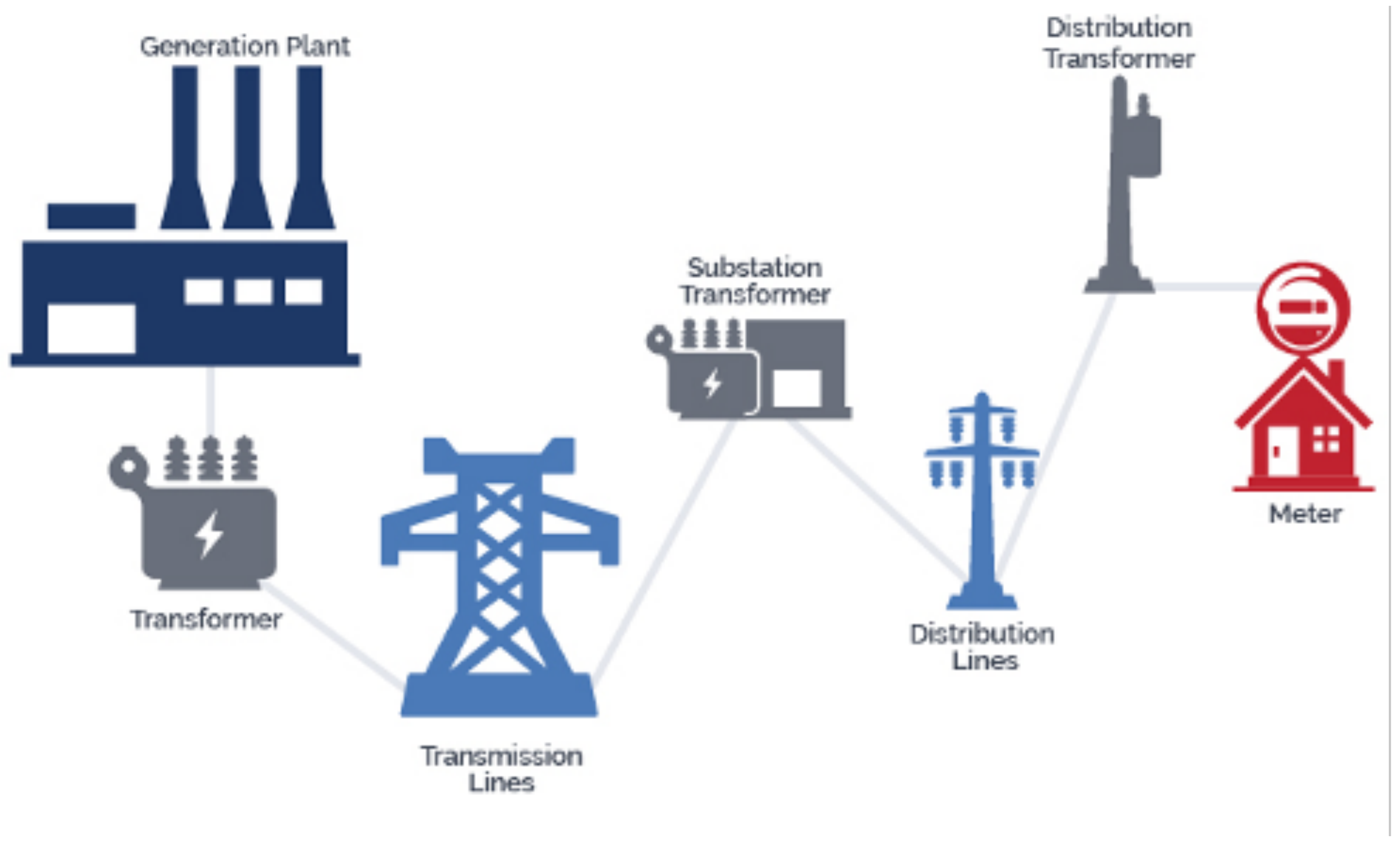
- Two very rich algorithmic problems.
- Cornerstone in combinatorial optimization.
- Beautiful mathematical duality.

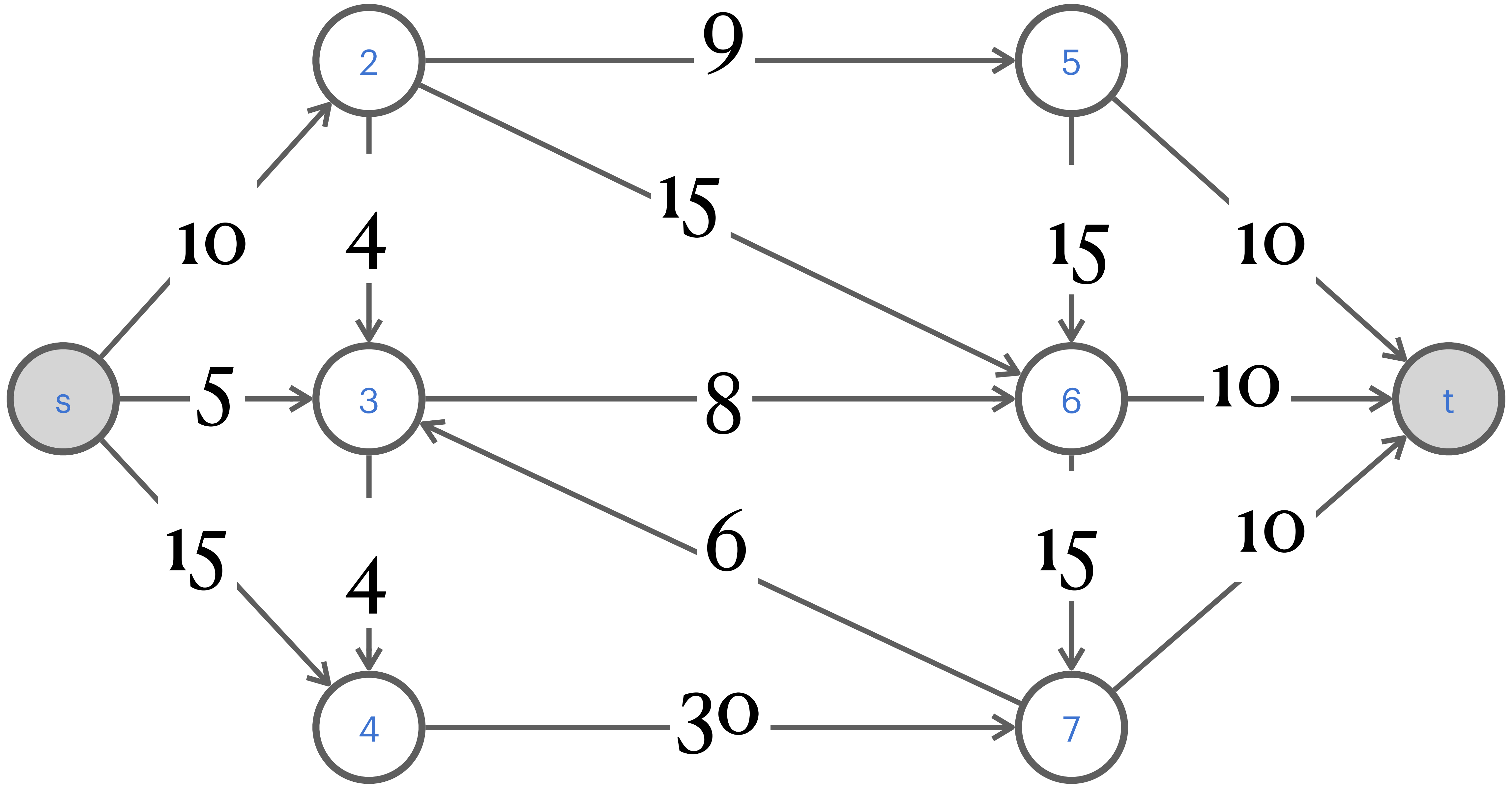
Applications (by reductions)

- Data mining.
- Airline scheduling.
- Bipartite matching, stable matching.
- Image segmentation, clustering, multi-camera scene reconstruction.
- Network intrusion detection, Data privacy.

Flow network

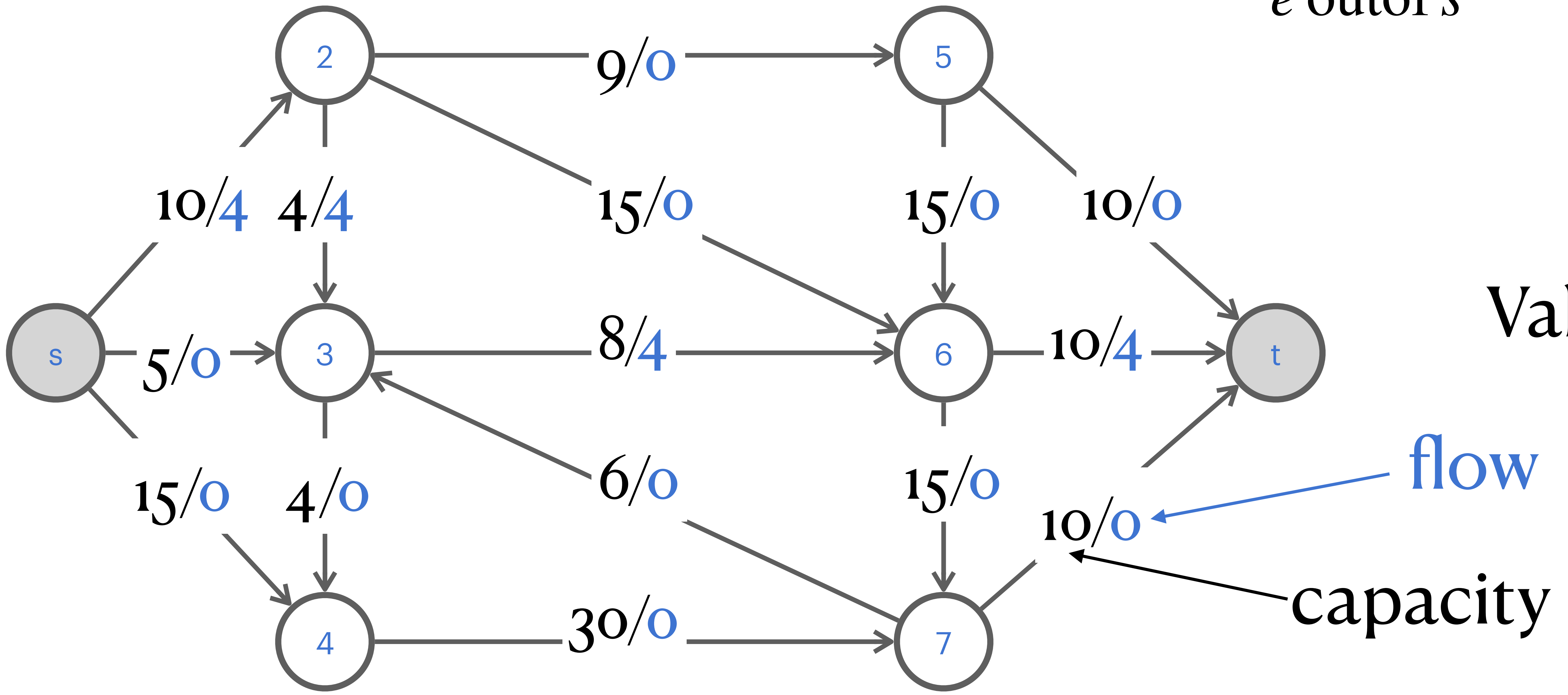
- **Abstraction for material flowing through the edges.**
  - $G = (V, E)$ directed graph, no parallel edges.
  - Two distinguished nodes: s = source, t = sink.
  - $c(e)$: capacity of edge e, $\forall e \in E$.





Flows

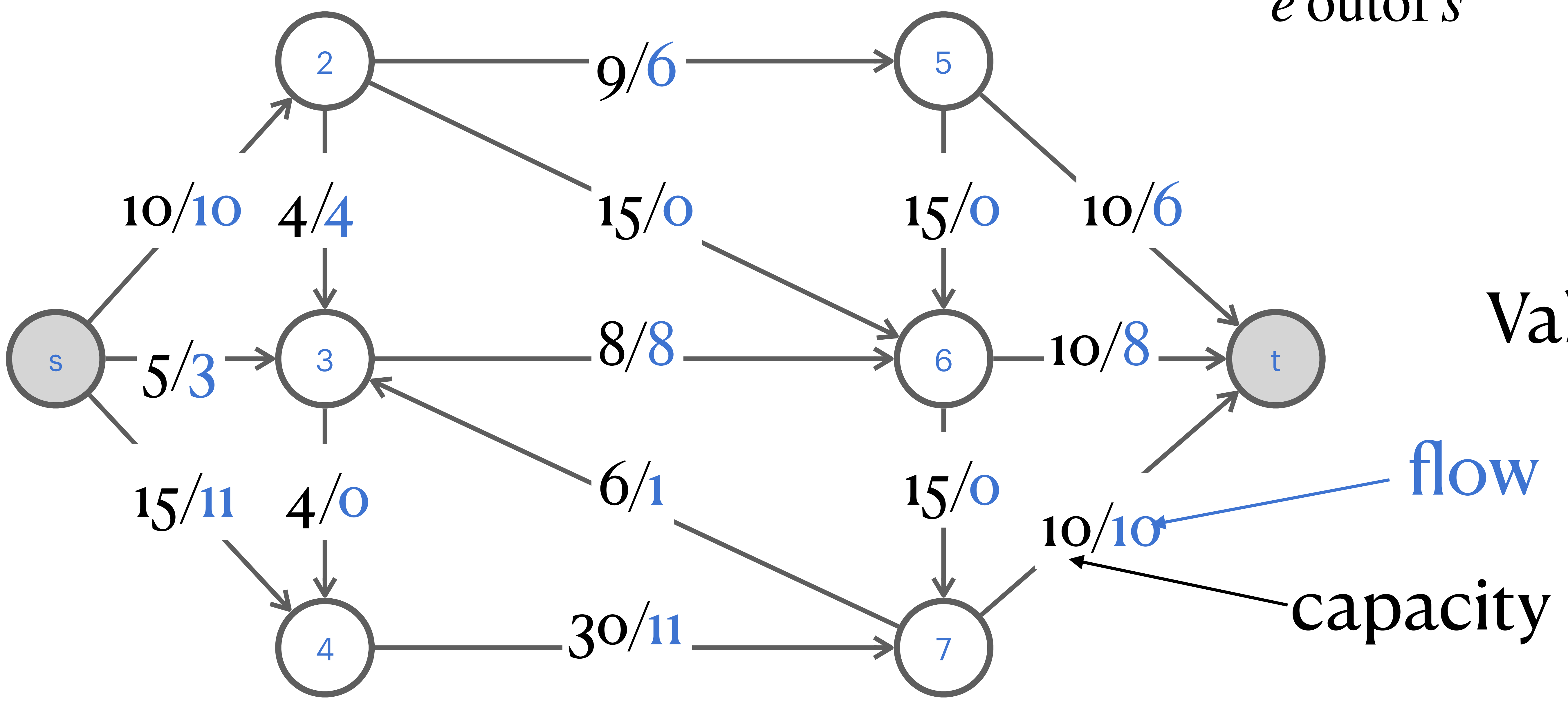
- **Definition.** **An** $s - t$ **flow is a function** $f : E \to \mathbb{R}^+$ **satisfying**
  - [Capacity] $\forall e \in E : 0 \leq f(e) \leq c(e)$.
  - [Conservation] $\forall v \in V - \{s, t\}$: $\displaystyle\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$
- **Definition.** **The value of a flow** f **is** $v(f) := \displaystyle\sum_{e \text{ outof } s} f(e)$



Value $v(f) = 4 + 0 + 0 = 4$

Flows, cont'd

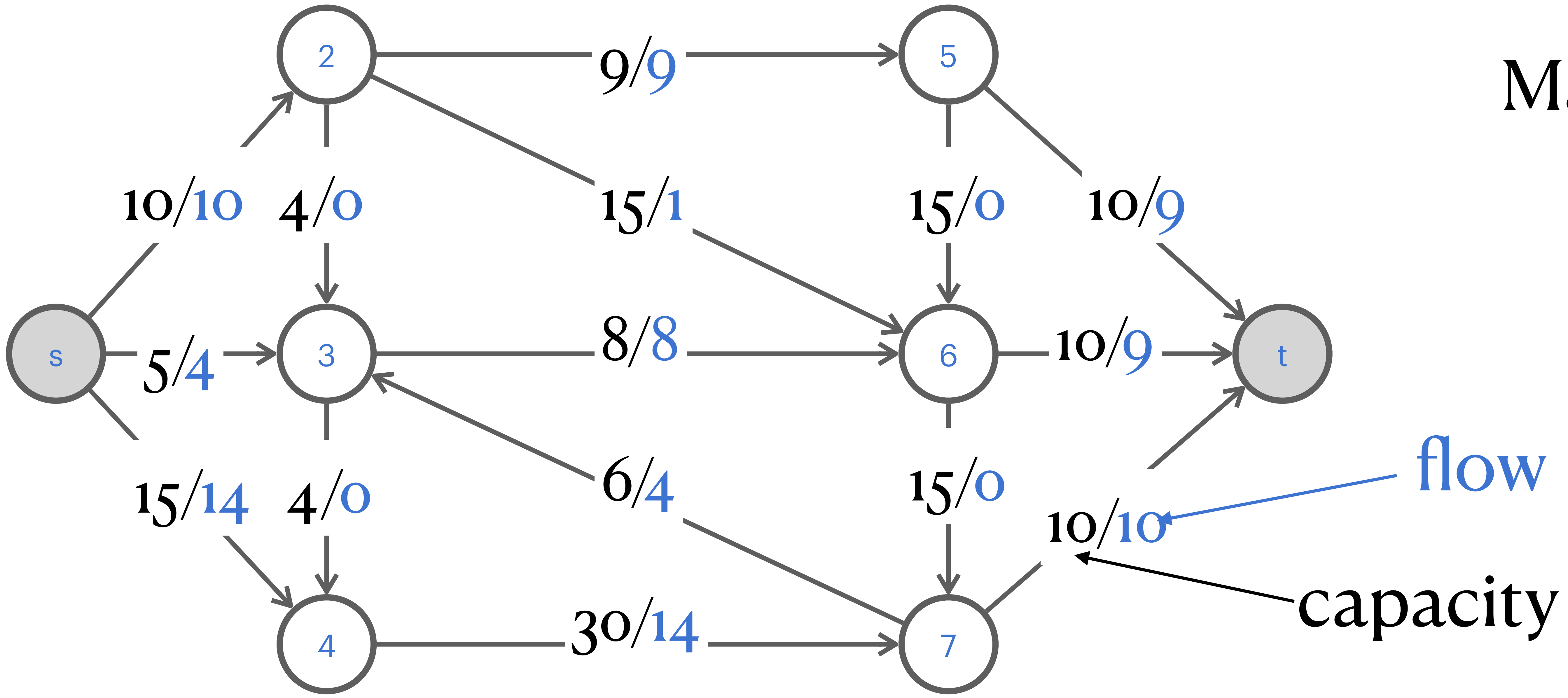
- **Definition. An $s - t$ flow is a function $f : E \to \mathbb{R}^+$ satisfying**
  - [Capacity] $\forall e \in E : 0 \leq f(e) \leq c(e)$.
  - [Conservation] $\forall v \in V - \{s, t\}$: $\displaystyle\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$
- **Definition. The value of a flow f is** $v(f) := \displaystyle\sum_{e \text{ outof } s} f(e)$



Value $v(f) = 10 + 3 + 11 = 24$

Maximum Flow problem

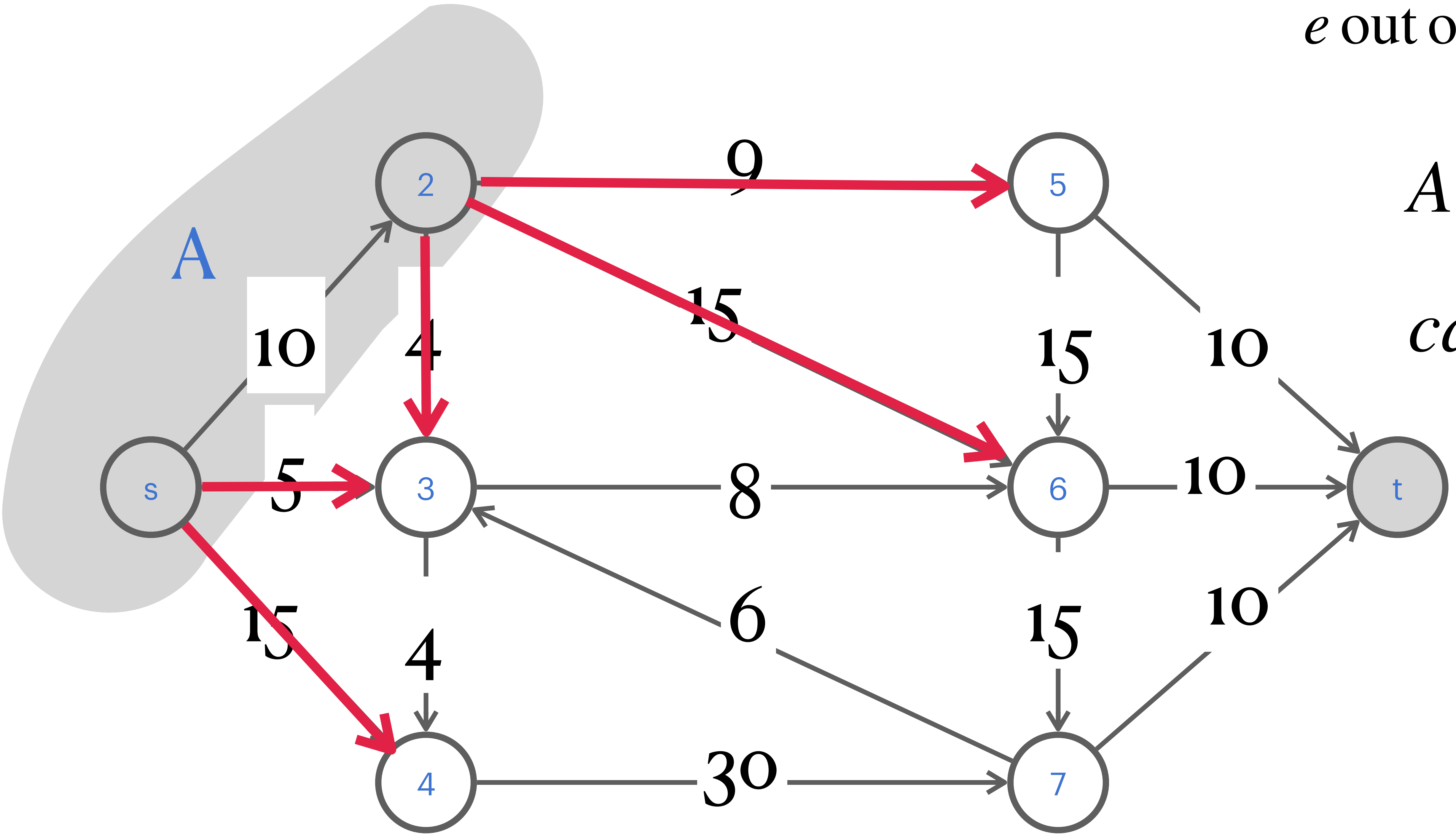
- **Find $s-t$ flow of maximum value.**
  - N.B. It has to be a valid flow, i.e., satisfying both Capacity and Conservation constraints.



Max flow $v(f) = 28$

Cuts

- Recall. A cut is a subset of vertices.
- Def. $s-t$ cut: $(A, B = V - A)$ partition of V with $s \in A$ and $t \in B$.
- Def. Capacity of cut (A, B): $cap(A, B) = \sum_{e \text{ out of } A} c(e)$

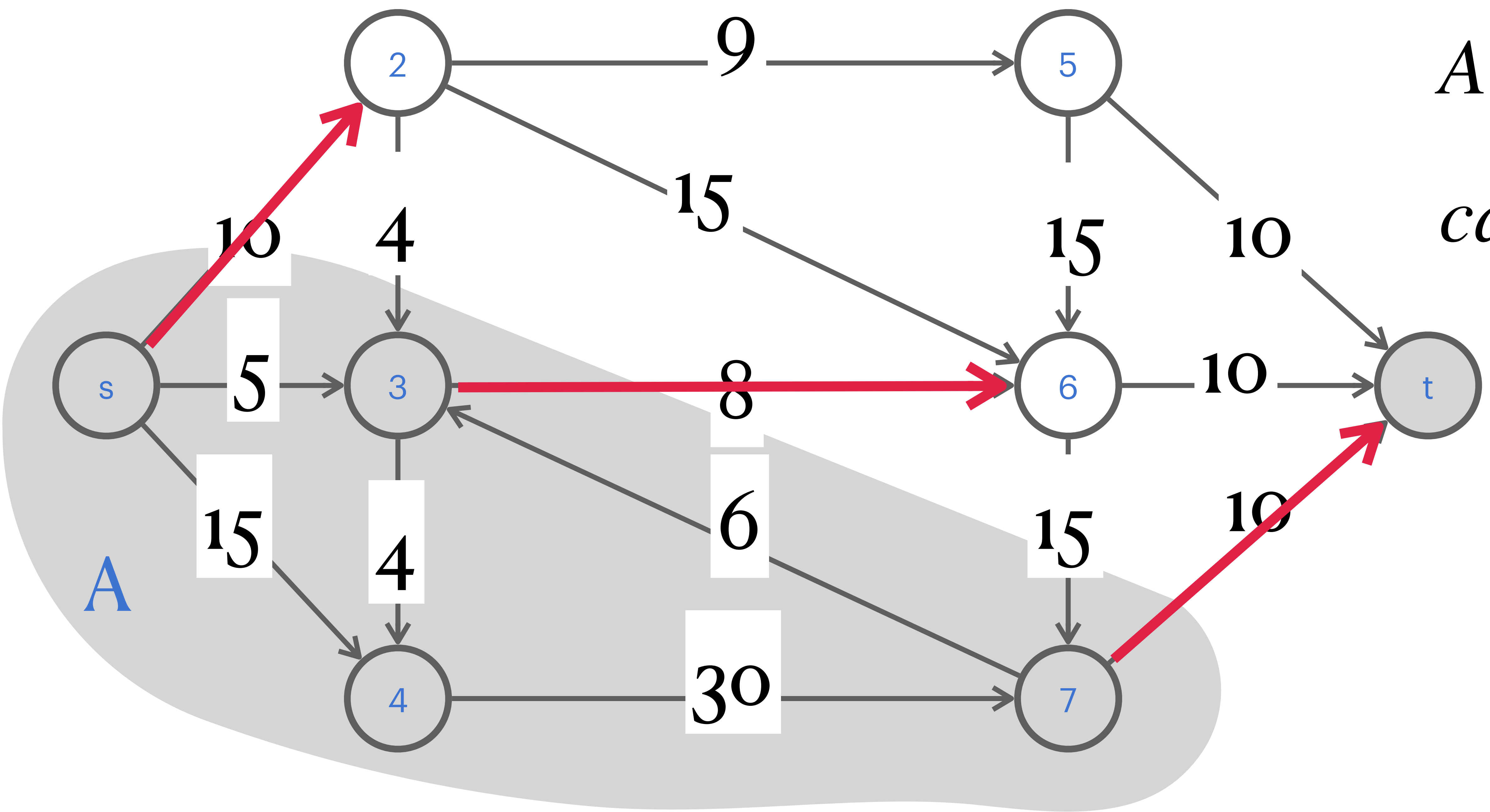


$A = \{s,2\}, B = \{3,4,\ldots,t\}$

$cap(A, B) = 9 + 15 + 4 + 5 + 15 = 48$

Cuts, cont'd

- **Recall.** A cut is a subset of vertices.
- **Def.** $s - t$ cut: $(A, B = V - A)$ partition of V with $s \in A$ and $t \in B$.
- **Def.** Capacity of cut (A, B): $cap(A, B) = \sum_{e \text{ out of } A} c(e)$

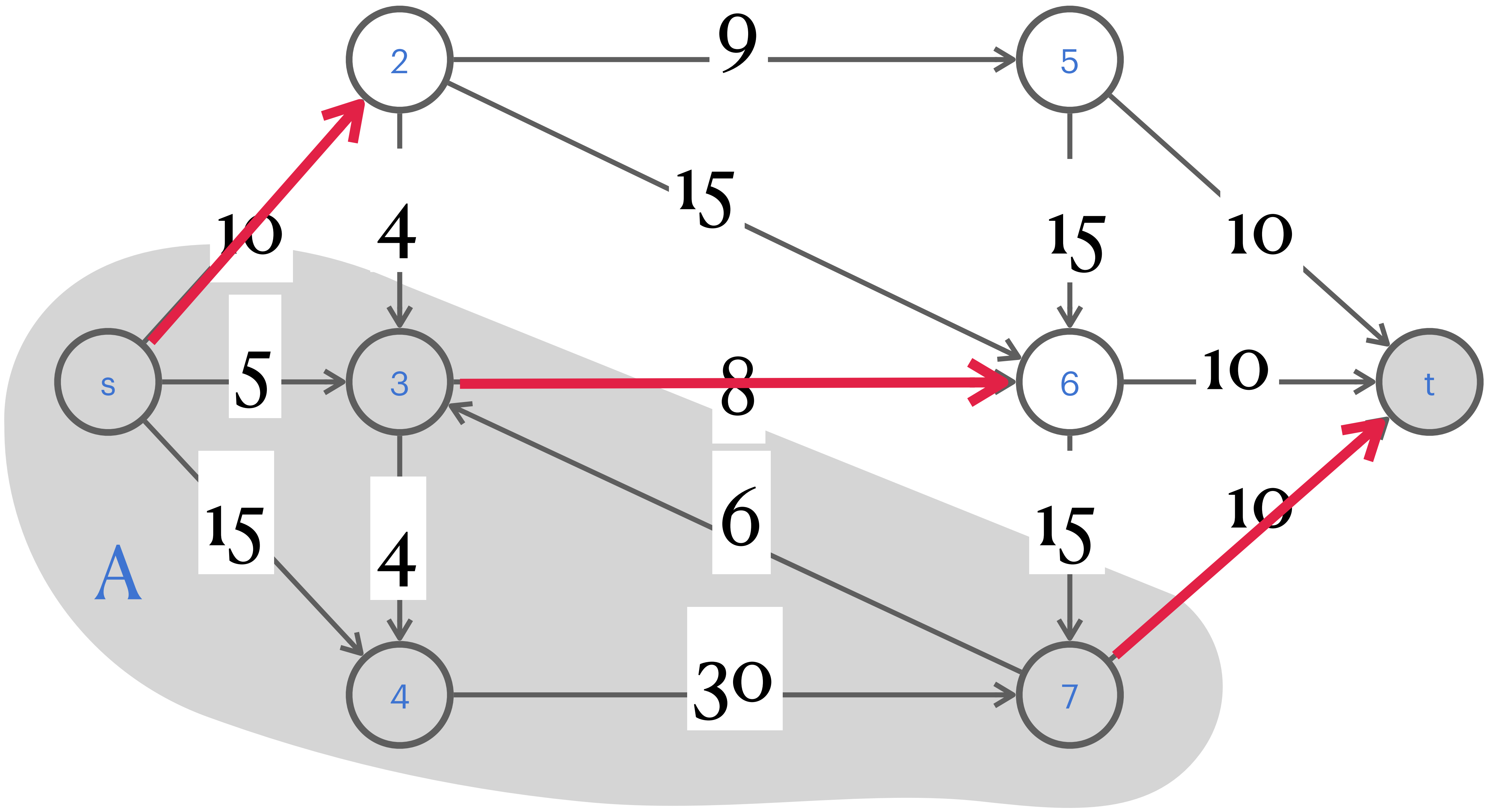


$A = \{s,3,4,7\}, B = \{2,5,6,t\}$

$cap(A, B) = 10 + 8 + 10 = 28$

Minimum cut problem

- Find $s - t$ cut of minimum capacity value.



Min cut $cap(A, B) = 28$

Max flow Min cut

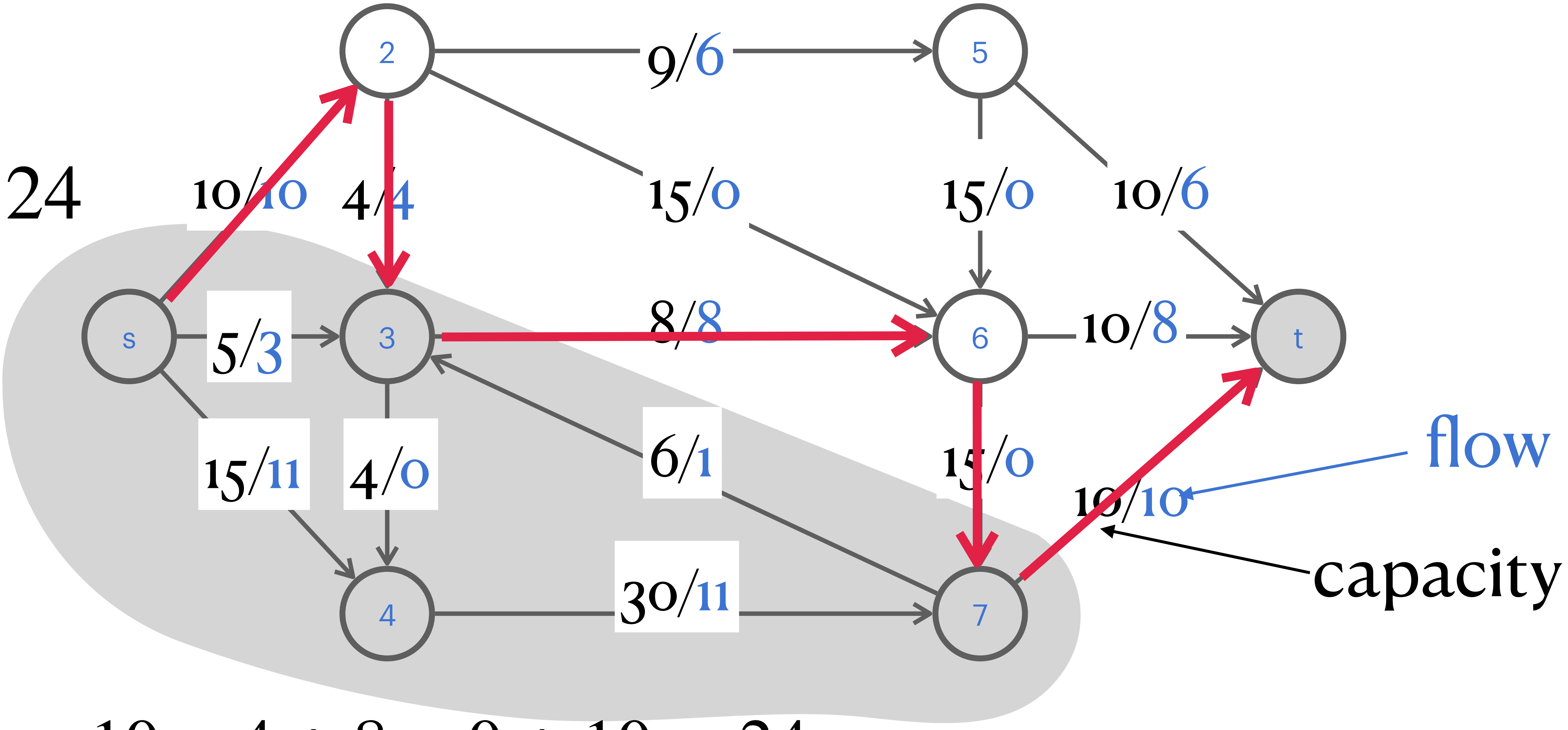
How to they relate?

Flow value lemma

- **Flow-value lemma.** Let f be any flow, and let (A, B) be any $s - t$ cut. Then the net flow across the cut is equal to the amount leaving s (i.e., value of flow).

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

$v(f) = 24$



Net flow $= 10 - 4 + 8 - 0 + 10 = 24$

Flow value lemma: proof

- **Flow-value lemma**. Let f be any flow, and let (A, B) be any $s-t$ cut.

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

- **Proof.**

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

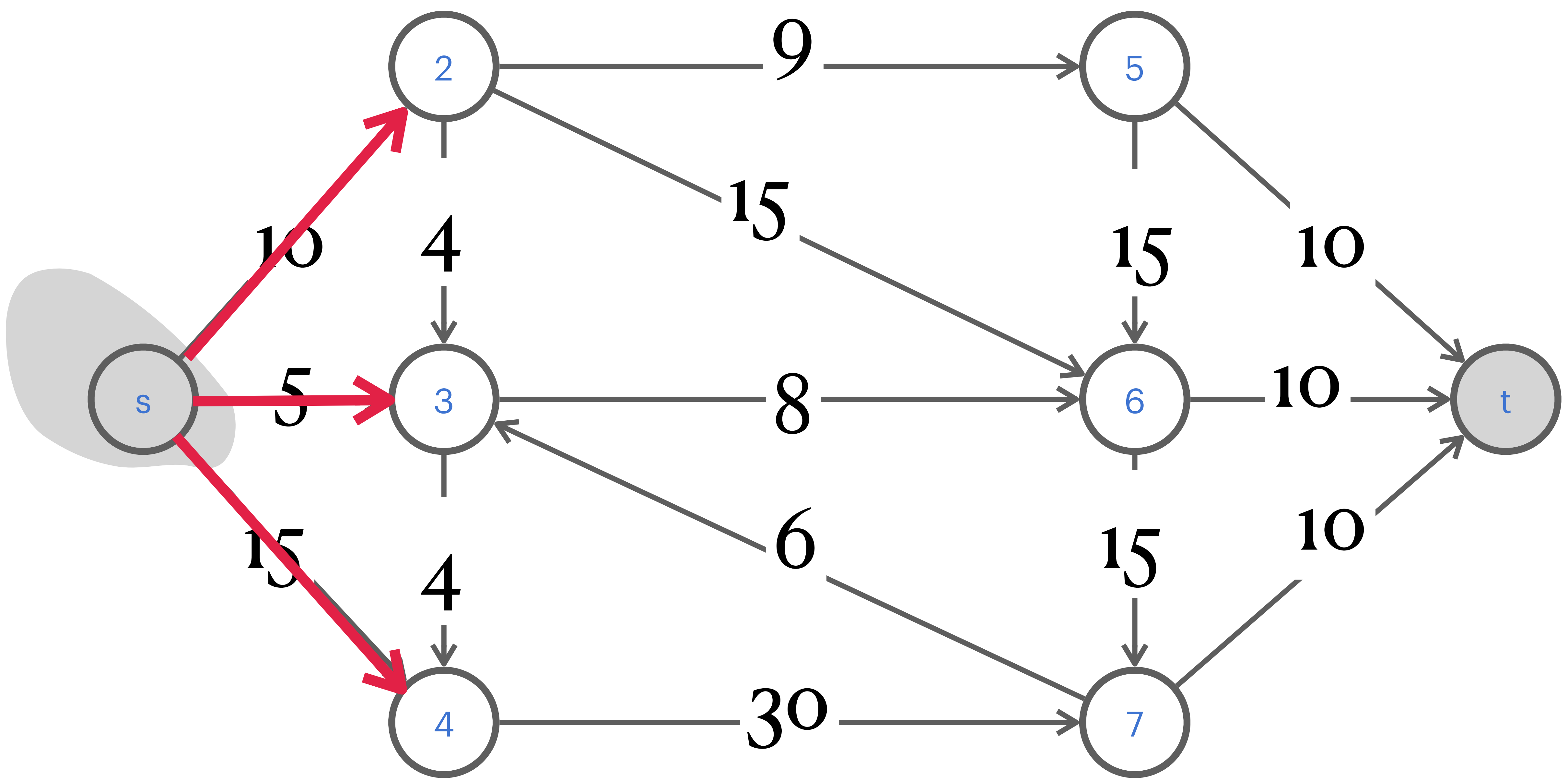
$$= \sum_{v \in A} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ into } v} f(e) \right)$$ // all but $v = s$ produce 0 by conservation

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

Weak duality

- Weak duality. Let f be any flow, and let (A, B) be any $s - t$ cut. Then the value of the flow is at most the capacity of the cut.

$$v(f) \le cap(A, B)$$



$$A = \{s\}, cap(A, B) = 30$$

$$\Rightarrow \text{any flow, value } \le 30$$

Weak duality: proof

- **Weak duality.** Let f be any flow, and let (A, B) be any $s - t$ cut.

$$v(f) \leq cap(A, B)$$

- **Proof.**

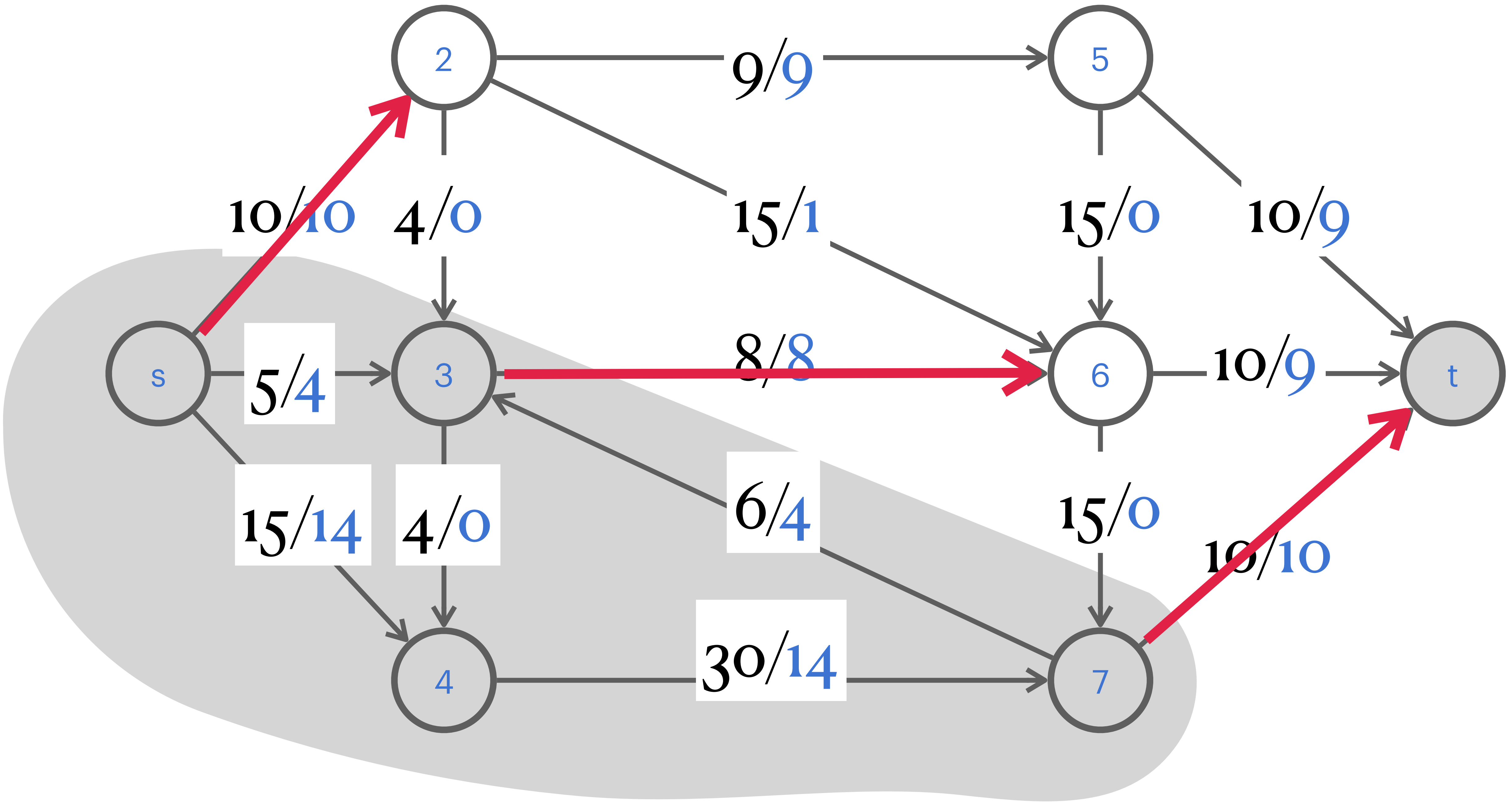
$$\begin{aligned} v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) && \text{// flow-value lemma} \\ &\leq \sum_{e \text{ out of } A} f(e) \\ &\leq \sum_{e \text{ out of } A} c(e) && \text{// capacity constraint} \\ &= cap(A, B) && \text{// definition of capacity} \end{aligned}$$

When does equality hold?

1. No flow coming into A
2. Flows saturate outgoing edges

Weak duality ⇒ certificate of optimality

- **Corollary of weak duality.** Let f be any flow, and let (A, B) be any $s - t$ cut. If $v(f) = cap(A, B)$, then f is a max flow, and (A, B) a min cut.



Value of flow = 28

Cut capacity = 28

⇒ value of flow ≤ 28

When does equality hold?

1. No flow coming into A
2. Flows saturate outgoing edges

Max-flow Min-cut theorem

Theorem. Value of max flow = capacity of min cut.

[Strong duality]

MAXIMAL FLOW THROUGH A NETWORK

L. R. FORD, JR. AND D. R. FULKERSON

ON THE MAX FLOW MIN CUT THEOREM OF NETWORKS

G. B. Dantzig

D. R. Fulkerson

P-826

April 15, 1955

1956 IRE TRANSACTIONS ON INFORMATION THEORY 117

A Note on the Maximum Flow Through a Network*

P. ELIAS†, A. FEINSTEIN‡, AND C. E. SHANNON§

Stay tuned for an elegant proof next time!

Scratch