Introduction to Network Flows

Admin

- Assignment 6 will be released today afternoon
    - More practice with dynamic programming
    - Shortest path with negative weights
- Some network flow questions (topic we'll start today)
- Due next week (Wed 11 pm, April 14)
    - In a week & a half
    - Utilize office & TA hours this week as well as next

Health Days Coming Up!

You are here

05Apr Intro to Network Flows Reading: KT §7.1–7.2 | E §10.1–10.2 Assignment 6 out	06Apr	07Apr Ford-Fulkerson Algorithm Reading: KT §7.2–7.3 | E §10.3	08Apr	09Apr Flow Applications Reading: KT §7.2–7.3 | E §10.3
12Apr P vs NP and NP-hardness Reading: KT §7.6 | E §11	13Apr	14Apr Problem Reductions Reading: KT §8.1, 8.3 | E §12.1–12.5 Assignment 7 out Assignment 6 due	15Apr	16Apr NP-hard Reductions Reading: KT §8.1, 8.3 | E §12.1–12.5
19Apr Intractability Wrap Up Reading: Reading: KT §8.2, 8.4 | E §12.6–12.8	20Apr	21Apr Health Day	22Apr Health Day	23Apr Probability Review 1 Reading: KT §8.5–8.7; | E §12.6–12.8
26Apr Probability Review 2 Reading: See Glow Syllabus	27Apr	28Apr Randomized Analyses Reading: See Glow Syllabus Assignment 8 out Assignment 7 due	29Apr	30Apr Karger's Min Cut & Rand QuickSort Reading: See GLOW Syllabus

Rest and sunshine is here!

Story So Far

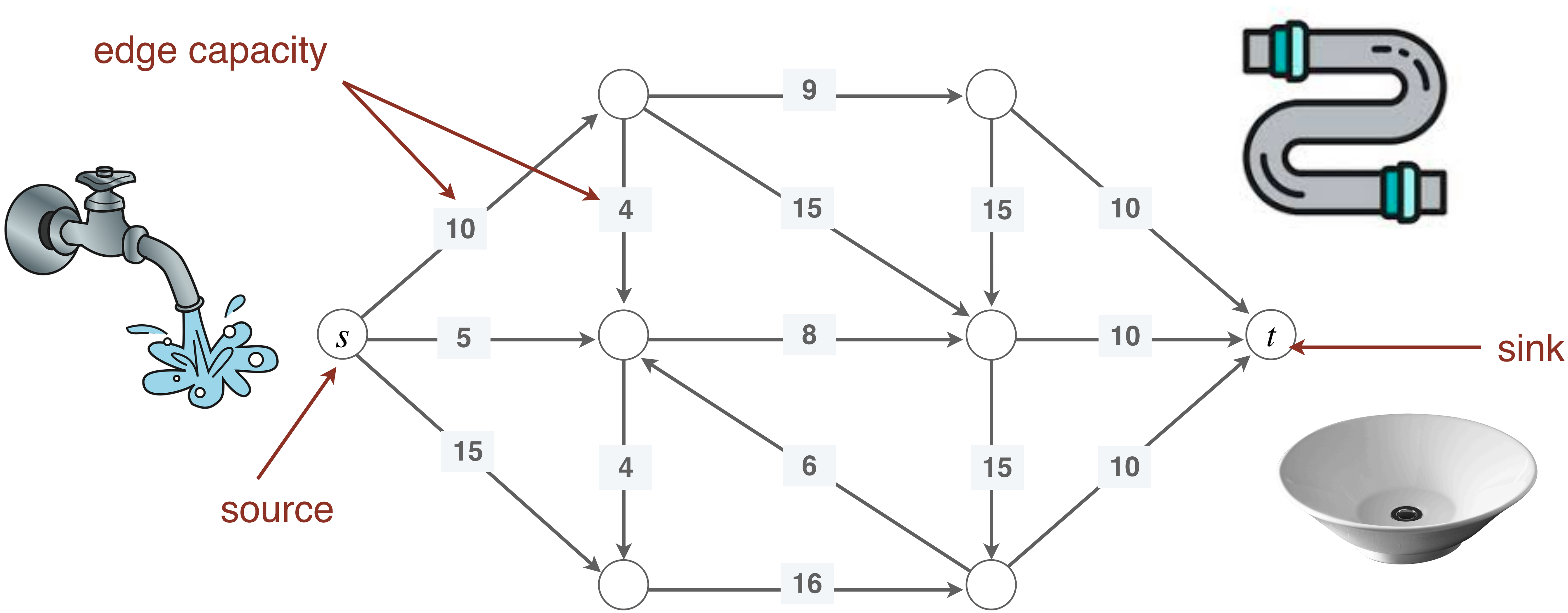
- Algorithmic design paradigms:
  - **Greedy**: simplest to design but works only for certain limited class of optimization problems
    - A good starting point for most problems but rarely optimal
  - **Divide and Conquer**
    - Solving a problem by breaking it down into smaller subproblems and recursing
  - **Dynamic programming**
    - Recursion with memoization: avoiding repeated work
    - Trading off space for time

New Algorithmic Paradigm

- **Network flows** model a variety of optimization problems
- These optimization problems look complicated with lots of constraints; on the face of it seem to have nothing to do with networks or flows
- Very powerful problem solving frameworks
- We'll focus on the concept of **problem reductions**
  - Problem **A** reduces to **B** if a solution to **B** leads to a solution to **A**
- Learn how to prove that our reductions are correct

What’s a Flow Network?

- A flow network is a directed graph $G = (V, E)$ with a
  - A **source** is a vertex s with in degree 0
  - A **sink** is a vertex t with out degree 0
  - Each edge $e \in E$ has **edge capacity** $c(e) > 0$



Assumptions

- Assume that each node v is on some s-t path, that is, $s \rightsquigarrow v \rightsquigarrow t$ exists, for any vertex $v \in V$
    - Implies G is connected and $m \geq n - 1$
- Assume **capacities are integers**
    - Will revisit this assumption and what happens if its not
- Directed edge (u, v) written as $u \to v$
- For simplifying expositions, we will sometimes write $c(u \to v) = 0$ when $(u, v) \notin E$

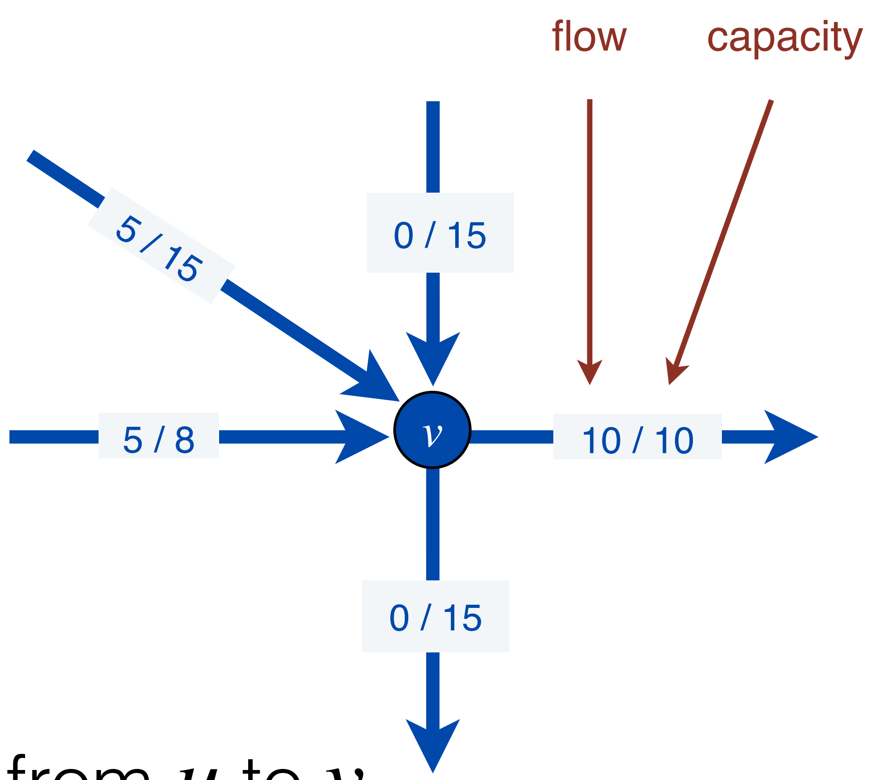
What's a Flow?

- Given a flow network, an (s, t)**-flow** or just **flow** (if source s and sink t are clear from context) $f : E \to \mathbb{Z}^+$ satisfies the following two constraints:

- **[Flow conservation]** $f_{in}(v) = f_{out}(v)$, for $v \neq s, t$ where

$$f_{in}(v) = \sum_u f(u \to v)$$

$$f_{out}(v) = \sum_w f(v \to w)$$

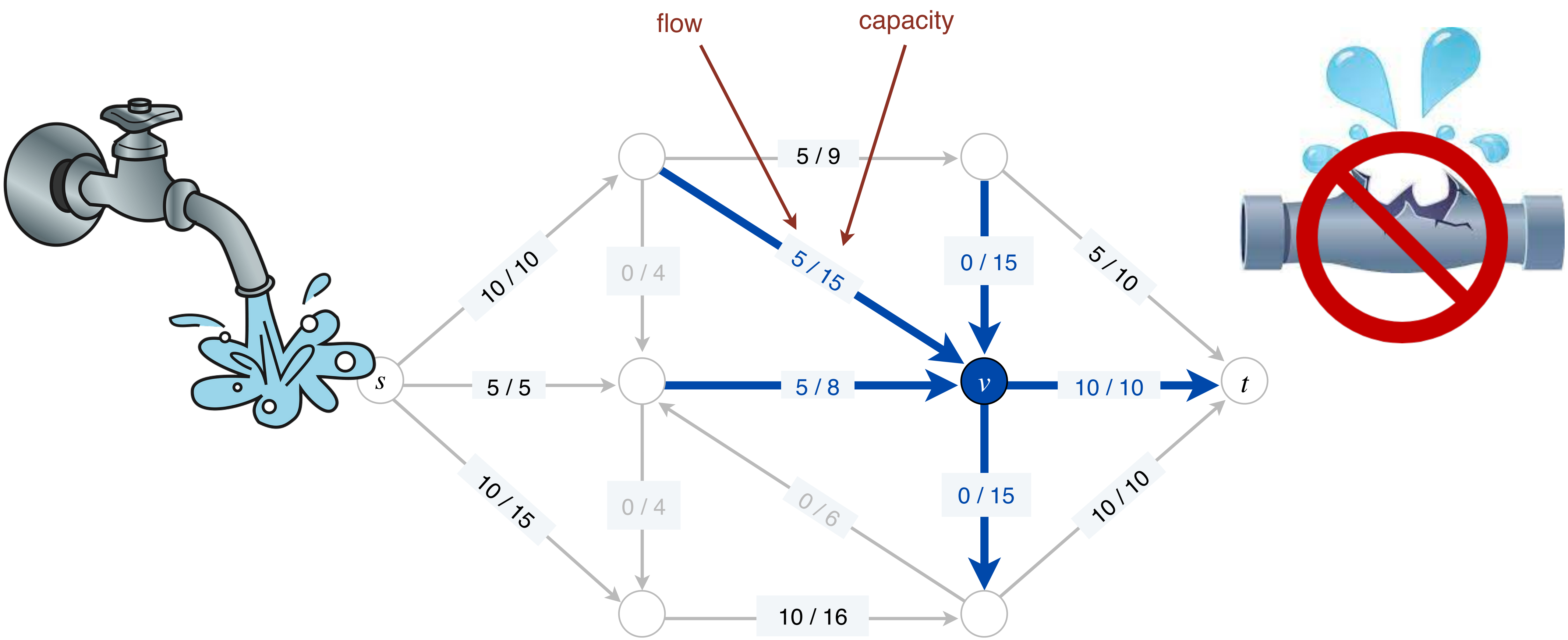


- To simplify, $f(u \to v) = 0$ if there is no edge from u to v

Feasible Flow

- And second, a feasible flow must satisfy the capacity constraints of the network, that is,

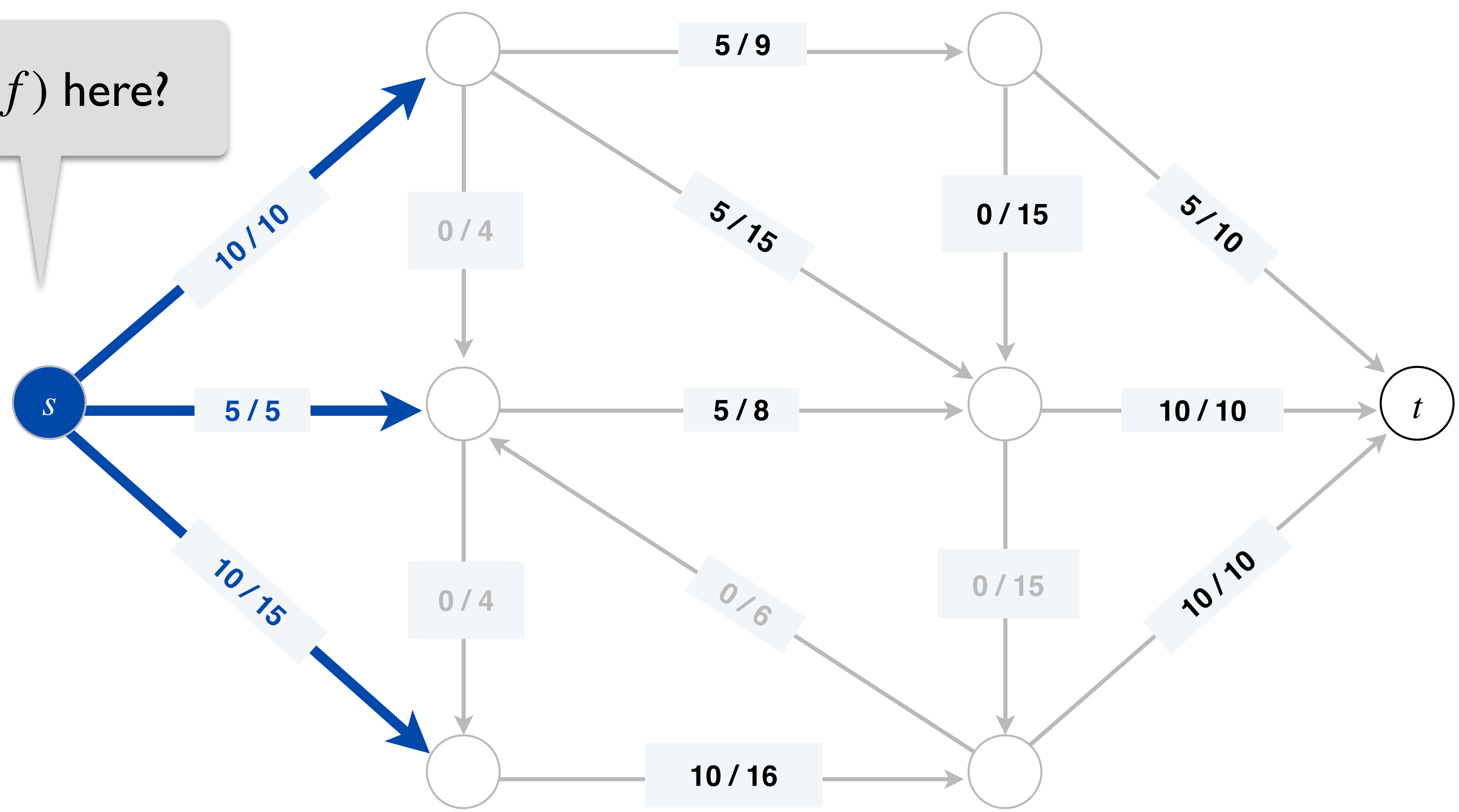
[Capacity constraint] for each $e \in E$, $0 \leq f(e) \leq c(e)$



Value of a Flow

- **Definition.** The **value** of a flow f, written $v(f)$, is $f_{out}(s)$.

What is $v(f)$ here?

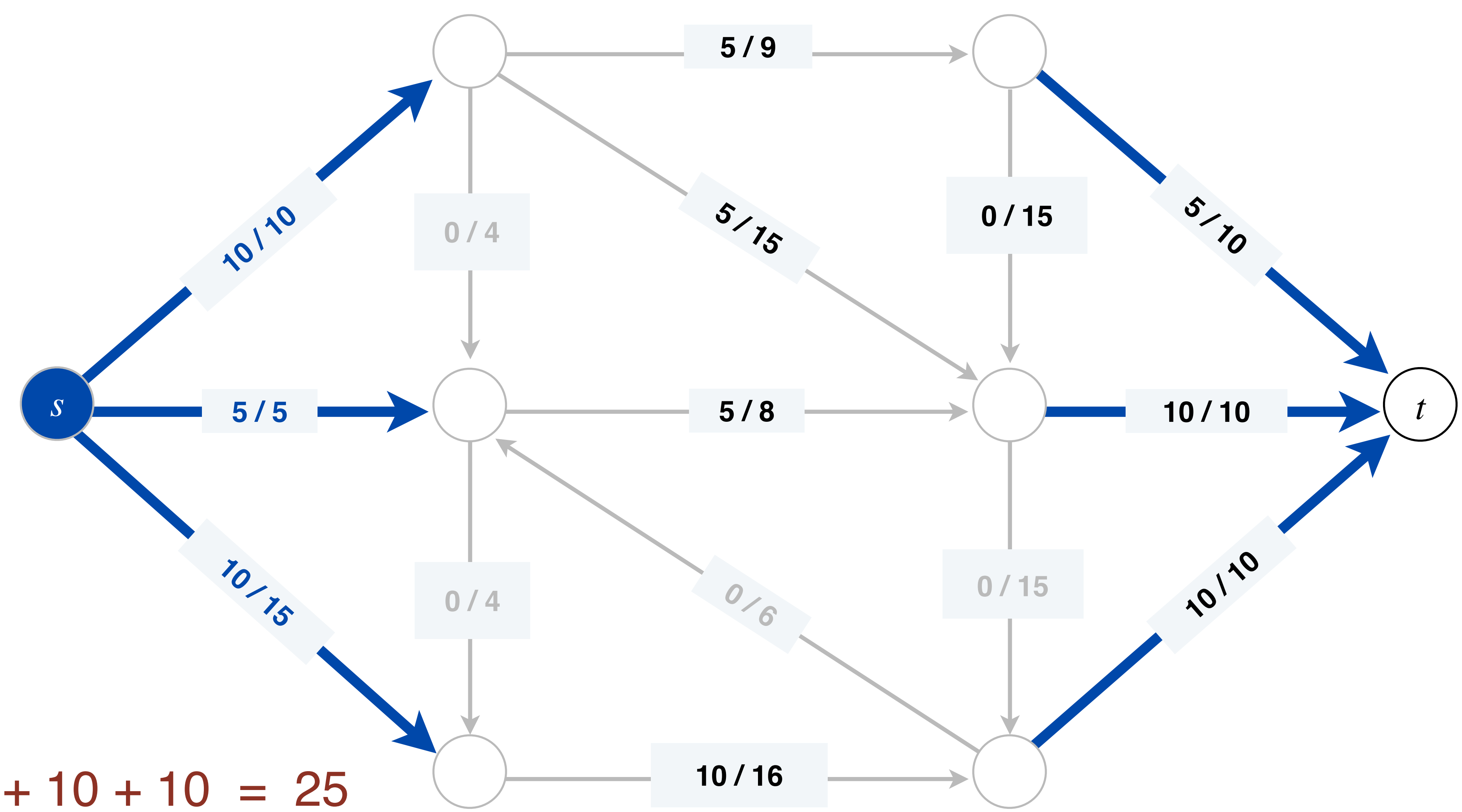


$v(f) = 5 + 10 + 10 = 25$

Value of a Flow

- **Definition.** The **value** of a flow f, written $v(f)$, is $f_{out}(s)$.
- **Lemma**. $f_{out}(s) = f_{in}(t)$

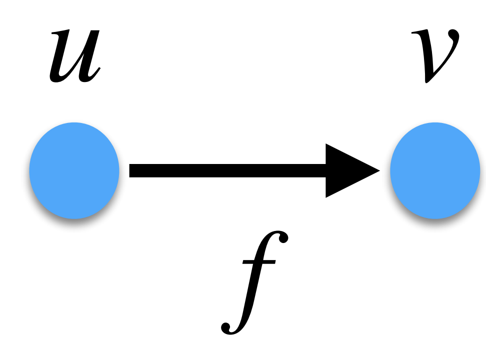
Intuitively, why do you think this is true?



value = 5 + 10 + 10 = 25

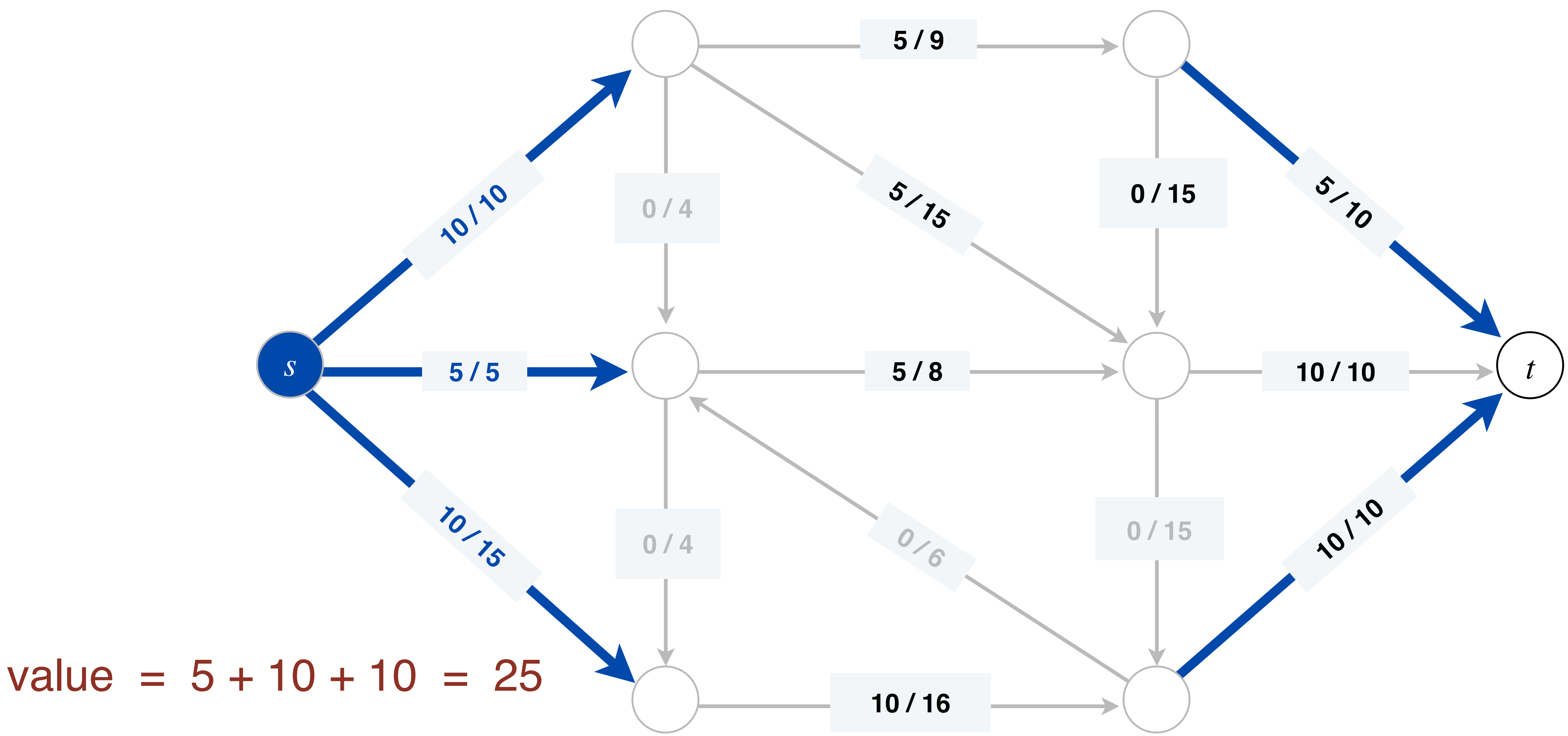
Value of a Flow

- **Lemma**. $f_{out}(s) = f_{in}(t)$
- **Proof**. Let $f(E) = \sum_{e \in E} f(e)$
  - Then, $\sum_{v \in V} f_{in}(v) = f(E) = \sum_{v \in V} f_{out}(v)$
  - For every $v \neq s, t$ flow conversation implies $f_{in}(v) = f_{out}(v)$
  - Thus all terms cancel out on both sides except $f_{in}(s) + f_{in}(t) = f_{out}(s) + f_{out}(t)$
  - But $f_{in}(s) = f_{out}(t) = 0$ $\blacksquare$



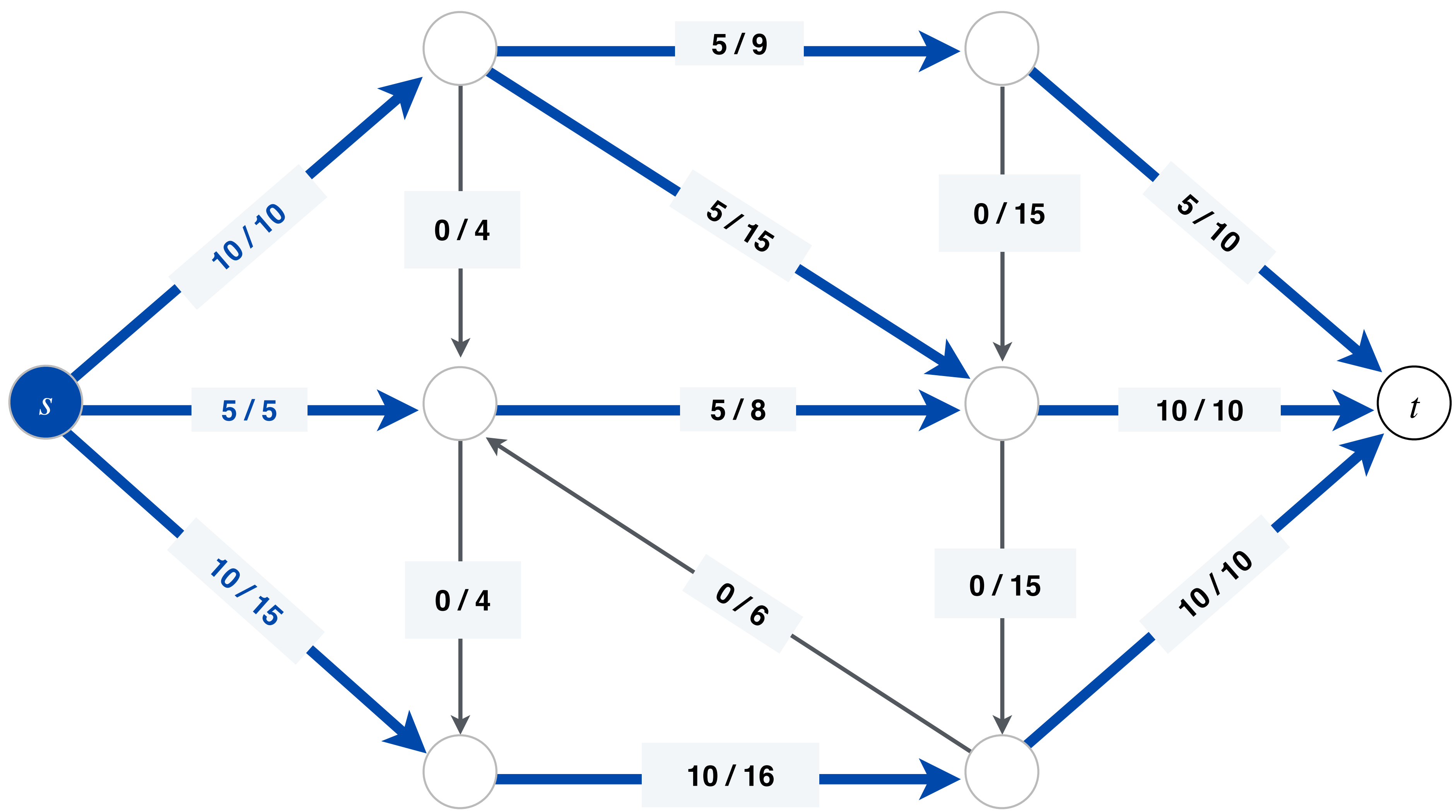
Value of a Flow

- **Lemma**. $f_{out}(s) = f_{in}(t)$
- **Corollary.** $v(f) = f_{in}(t)$.



Max-Flow Problem

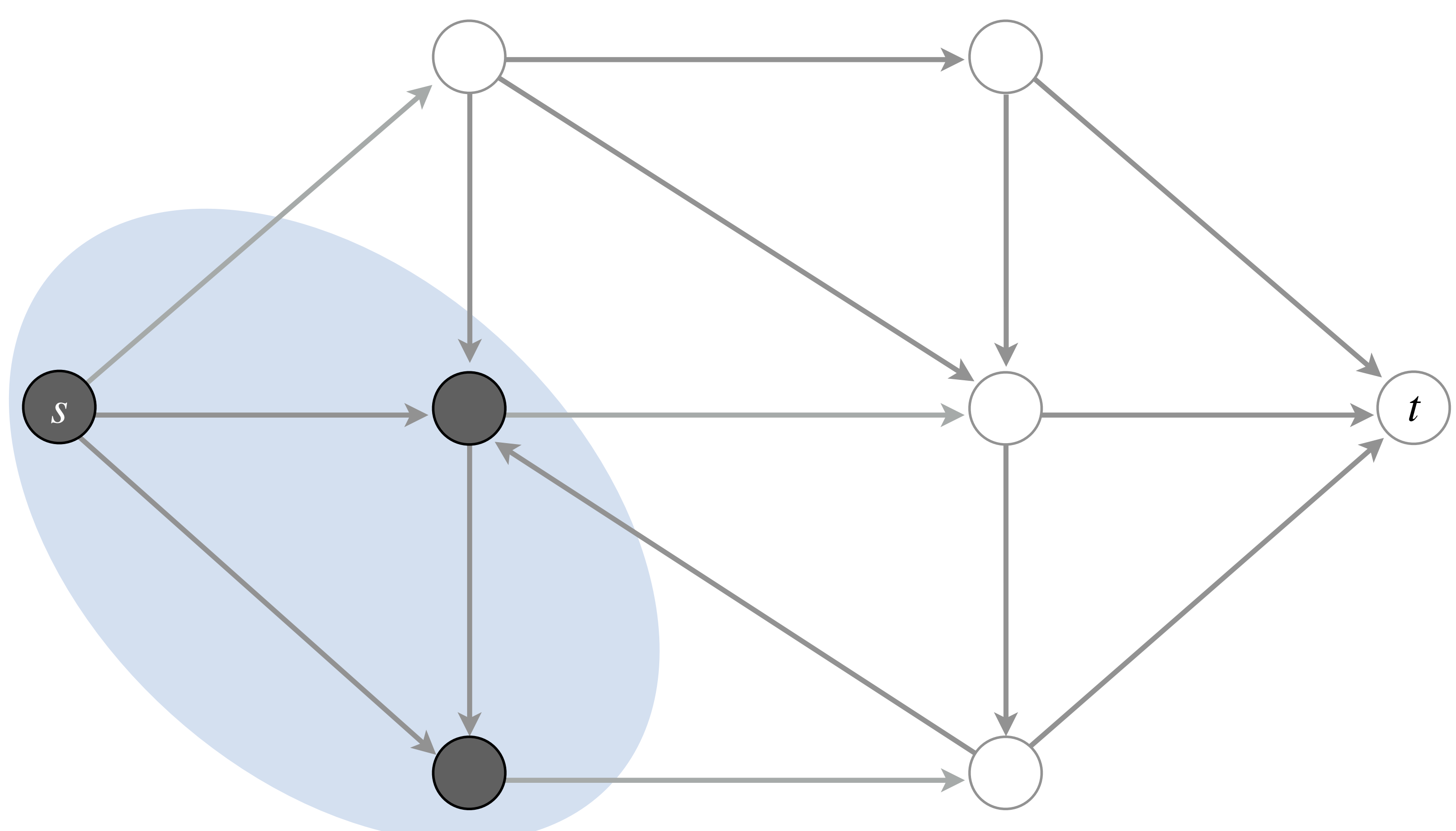
- **Problem**. Given an s-t flow network, find a feasible s-t flow of **maximum** value.



Minimum Cut Problem

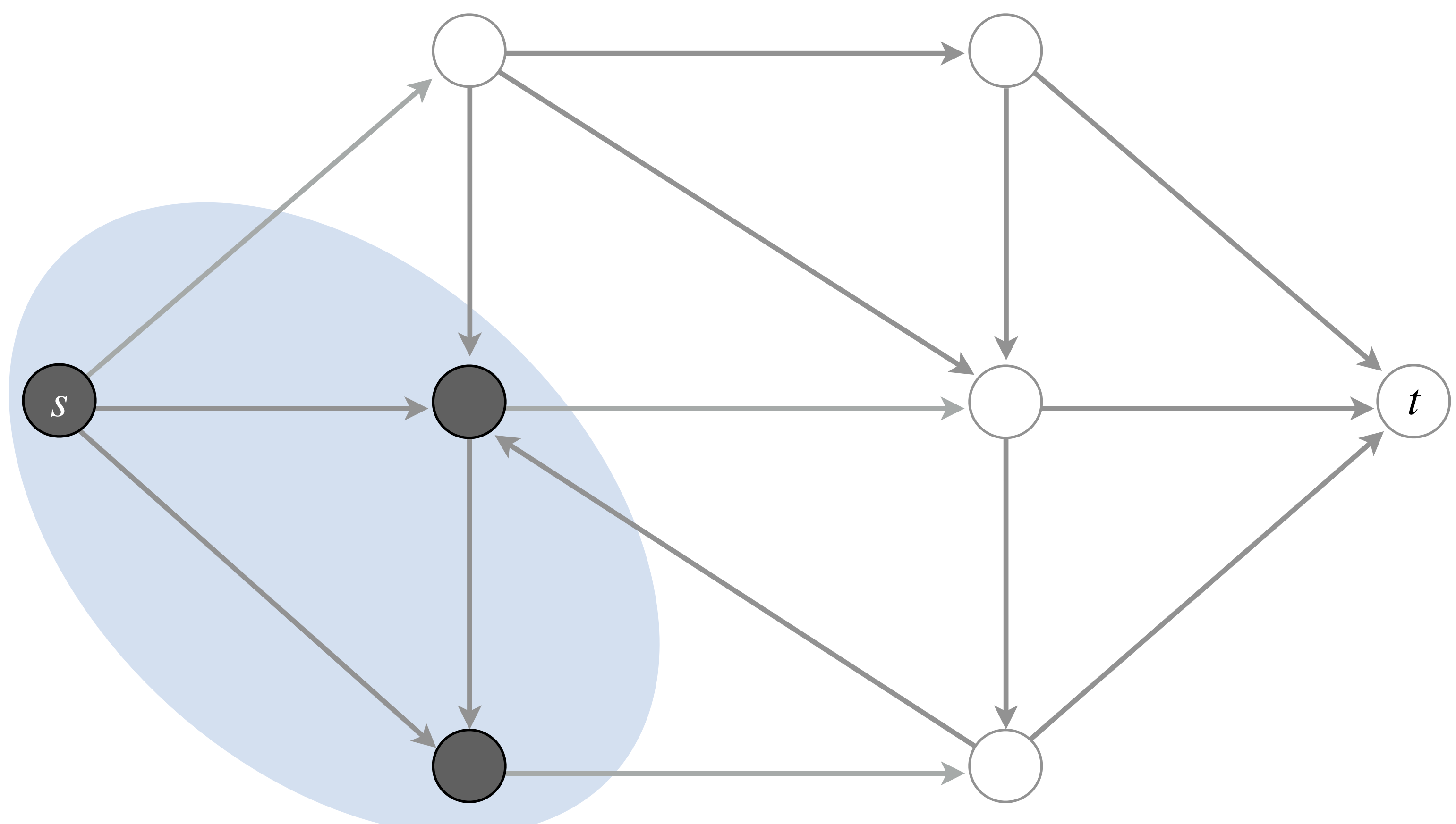
Cuts are Back!

- Cuts in graphs played a lead role when we were designing algorithms for MSTs
- What is the definition of a cut?



Cuts in Flow Networks

- **Recall**. A cut (S, T) in a graph is a partition of vertices such that $S \cup T = V$, $S \cap T = \varnothing$ and S, T are non-empty.
- **Definition**. An (s, t)-*cut* is a cut (S, T) s.t. $s \in S$ and $t \in T$.



Cut Capacity

- **Recall**. A cut (S, T) in a graph is a partition of vertices such that $S \cup T = V$, $S \cap T = \emptyset$ and S, T are non-empty.
- **Definition**. An (s, t)-*cut* is a cut (S, T) s.t. $s \in S$ and $t \in T$.
- **Capacity** of a (s, t)-cut (S, T) is the sum of the capacities of edges leaving S:
  - $$c(S, T) = \sum_{v \in S, w \in T} c(v \to w)$$

Quick Quiz

Question. What is the capacity of the s-t given by grey and white nodes?

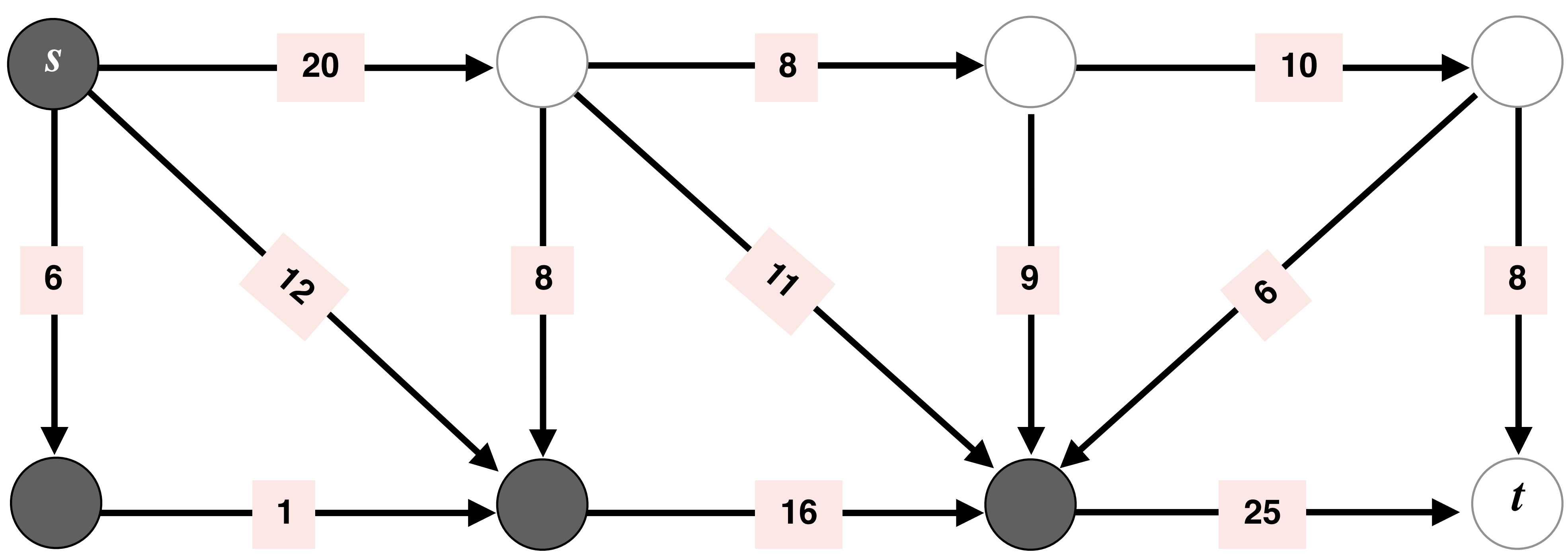
$$c(S,T) = \sum_{v \in S, w \in T} c(v \to w)$$

A. 11 (20 + 25 – 8 – 11 – 9 – 6)

B. 34 (8 + 11 + 9 + 6)

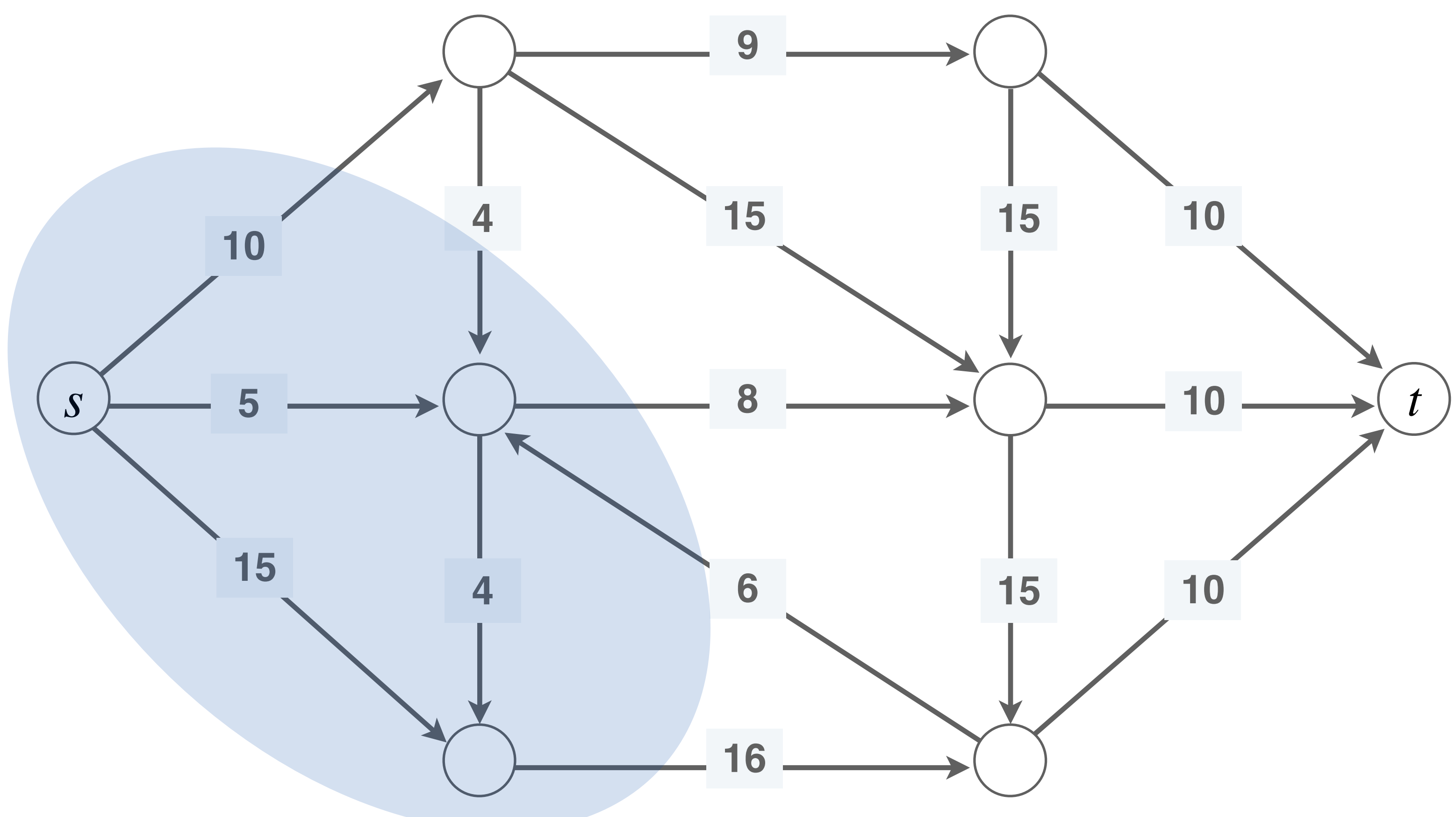
C. 45 (20 + 25)

D. 79 (20 + 25 + 8 + 11 + 9 + 6)



Min Cut Problem

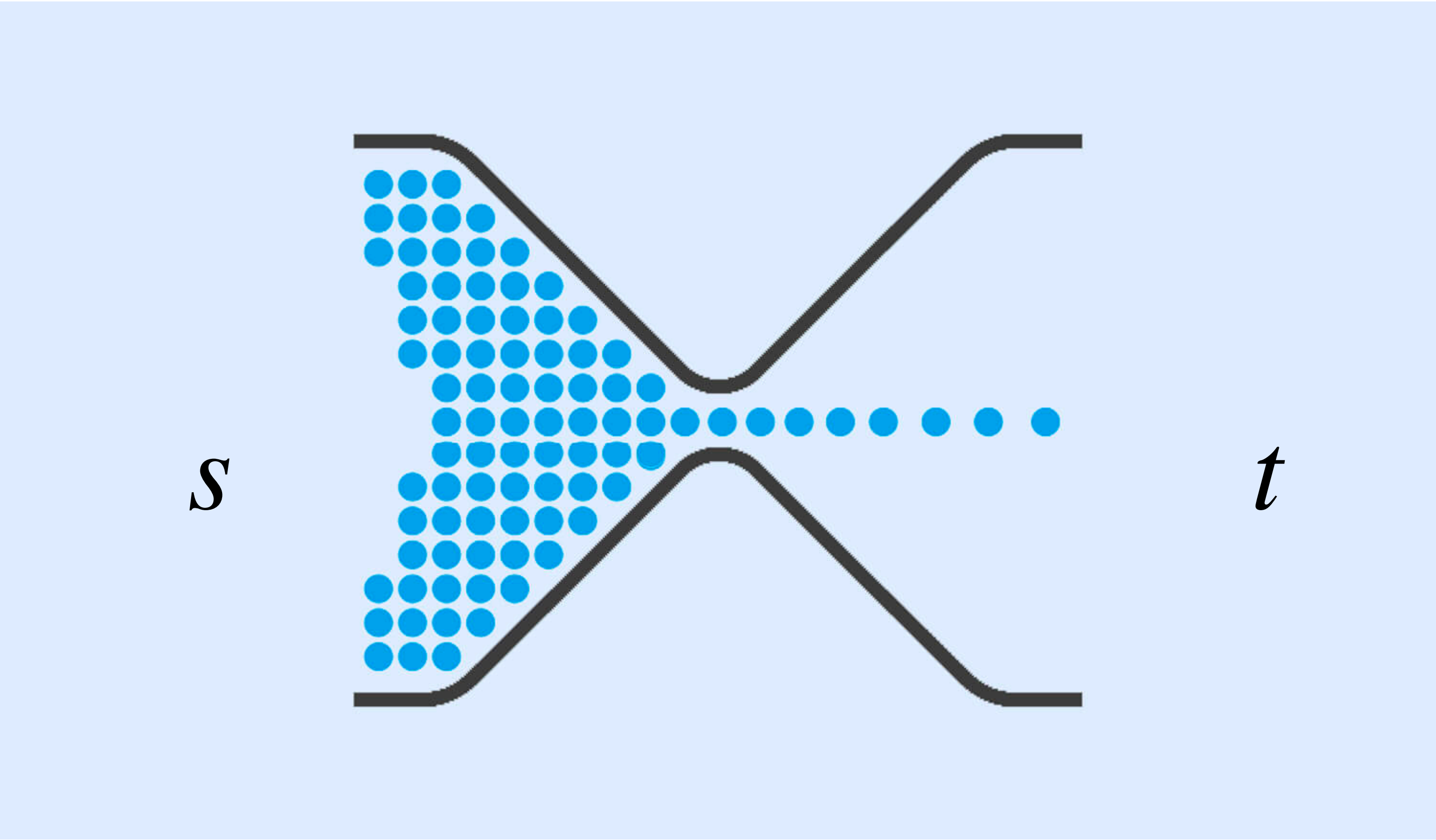
- **Problem**. Given an s-t flow network, find an s-t cut of **minimum** capacity.



Relationship between Flows and Cuts

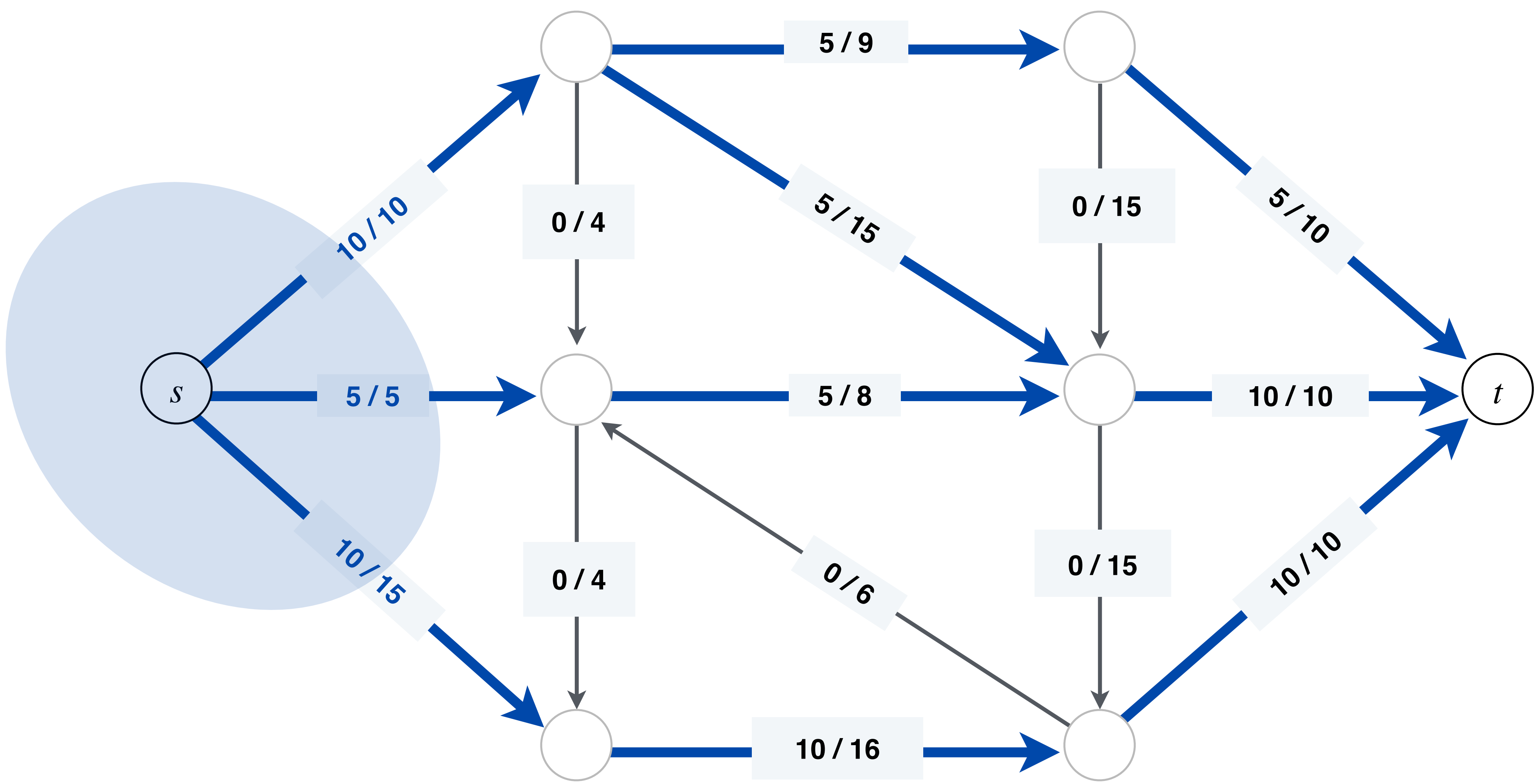
Flows and Cuts

- Cuts represent "**bottlenecks**" in a flow network
- For any cut, our flow needs to “get out” of that cut on its route from s to t
- Let us formalize this intuition



Flows and Cuts

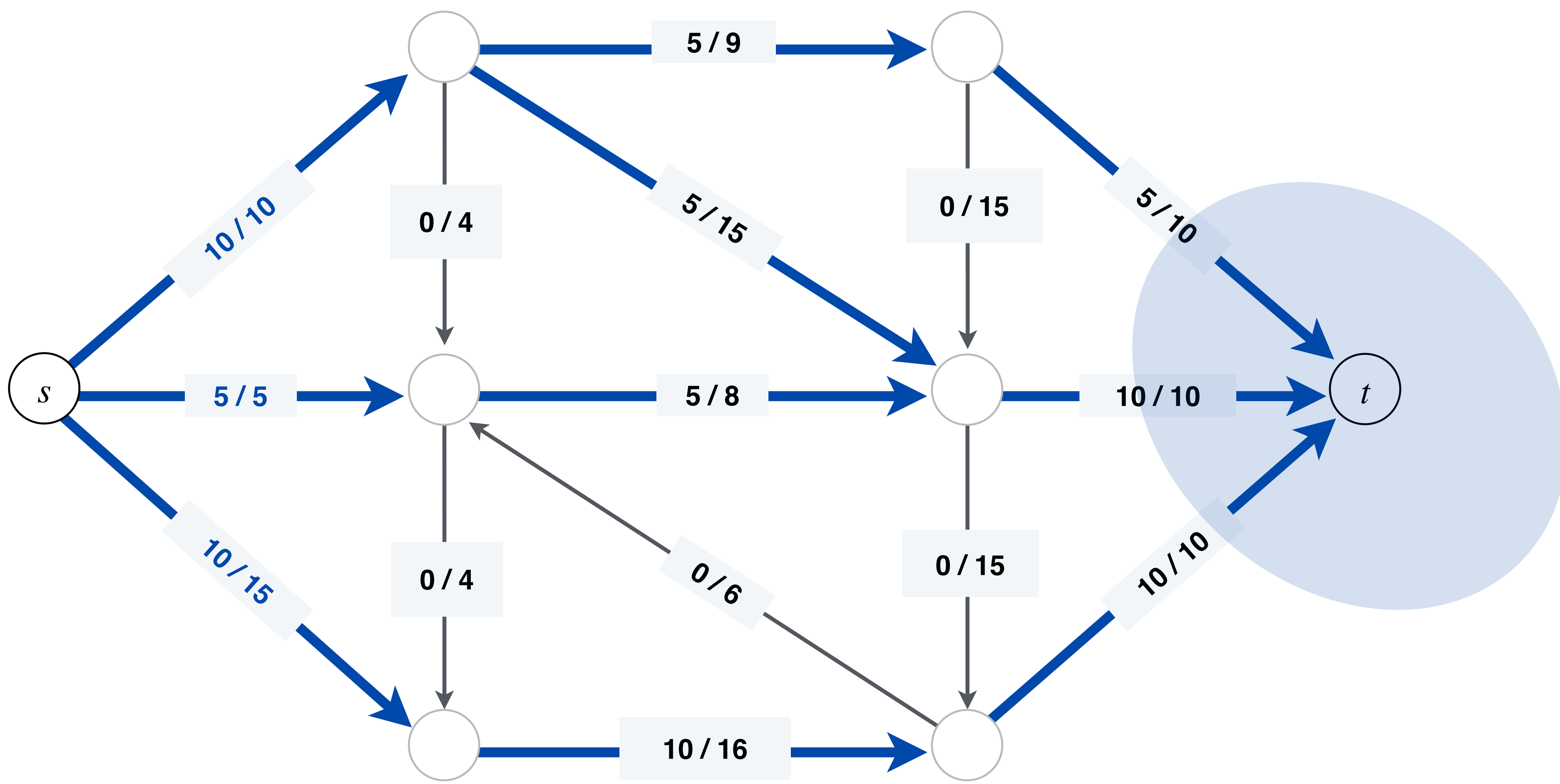
- **Claim**. Let f be **any** s-t flow and (S, T) be **any** s-t cut then $v(f) \leq c(S, T)$
- There are two s-t cuts for which this is easy to see, which ones?



Flows and Cuts

- **Claim**. Let f be **any** s-t flow and (S, T) be **any** s-t cut then $v(f) \leq c(S, T)$

- There are two s-t cuts for which this is easy to see, which ones?



Flows and Cuts

- To prove this for any cut, we first relate the flow value in a network to the net flow leaving a cut
- **Lemma**. For any feasible (s, t)-flow f on $G = (V, E)$ and any (s, t)-cut , $v(f) = f_{out}(S) - f_{in}(S)$, where
  - $f_{out}(S) = \sum_{v \in S, w \in T} f(v \to w)$ (sum of flow 'leaving' S)
  - $f_{in}(S) = \sum_{v \in S, w \in T} f(w \to v)$ (sum of flow 'entering' S)
  - Note: $f_{out}(S) = f_{in}(T)$ and $f_{in}(S) = f_{out}(T)$

Flows and Cuts

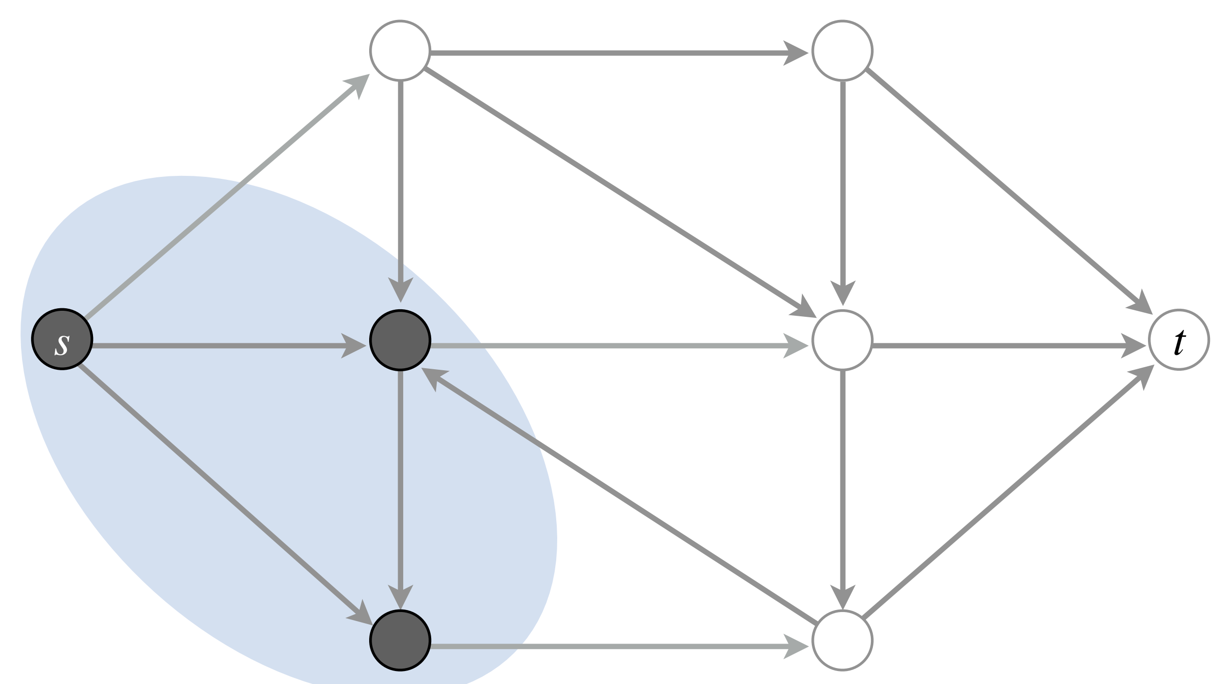
Proof. $f_{out}(S) - f_{in}(S)$

$$= \sum_{v \in S, w \in T} f(v \to w) - \sum_{v \in S, u \in T} f(u \to v) \quad \text{[by definition]}$$

Adding zero terms

$$= \left[\sum_{v,w \in S} f(v \to w) - \sum_{v,u \in S} f(u \to v) \right] + \sum_{v \in S, w \in T} f(v \to w) - \sum_{v \in S, u \in T} f(u \to v)$$

These are the same sum: they sum the flow of all edges with both vertices in S



Flows and Cuts

Proof. $f_{out}(S) - f_{in}(S)$

$$= \left[\sum_{v,w\in S} f(v\to w) - \sum_{v,u\in S} f(u\to v)\right] + \sum_{v\in S,w\in T} f(v\to w) - \sum_{v\in S,u\in T} f(u\to v)$$

Rearranging terms

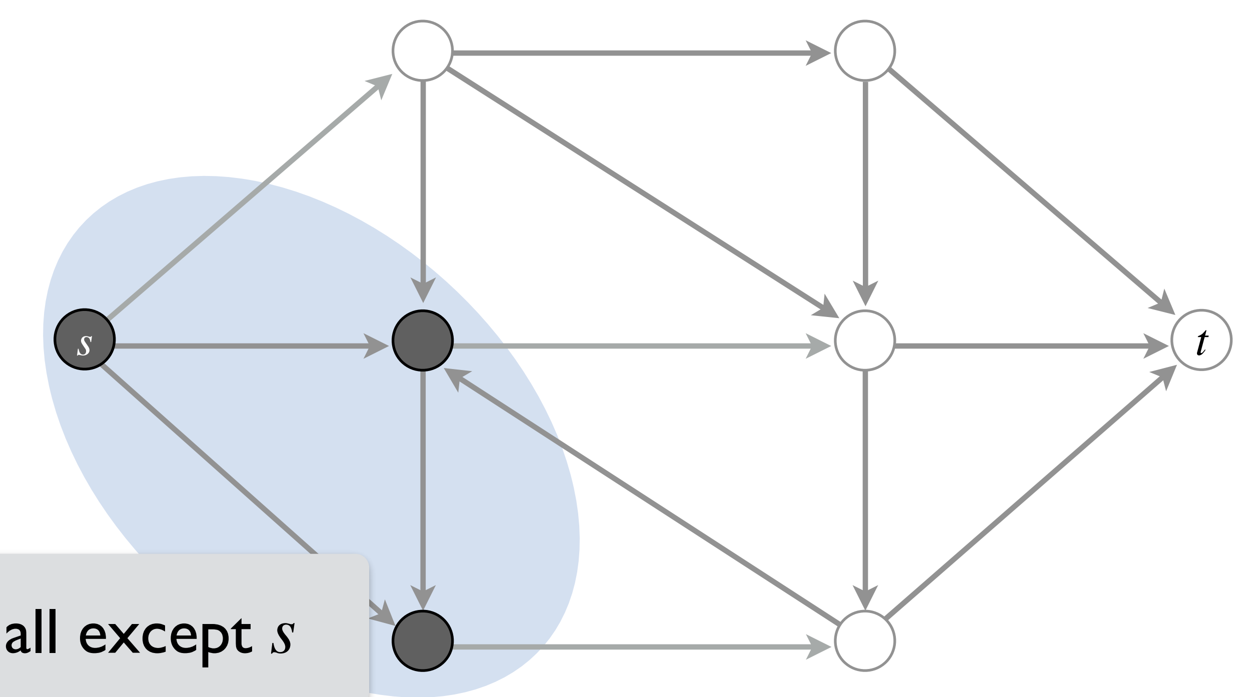
$$= \sum_{v,w\in S} f(v\to w) + \sum_{v\in S,w\in T} f(v\to w) - \sum_{v,u\in S} f(u\to v) - \sum_{v\in S,u\in T} f(u\to v)$$

$$= \sum_{v\in S}\Big(\sum_{w} f(v\to w) - \sum_{u} f(u\to v)\Big)$$

$$= \sum_{v\in S} f_{out}(v) - f_{in}(v)$$

Cancels out for all except s

$$= f_{out}(s) = v(f) \quad \blacksquare$$



Flows and Cuts

- We use this result to prove that the value of a flow cannot exceed the capacity of any cut in the network
- **Claim**. Let f be any s-t flow and (S, T) be any s-t cut then $v(f) \leq c(S, T)$
- **Proof.** $v(f) = f_{out}(S) - f_{in}(S)$

$$\leq f_{out}(S) = \sum_{v \in S, w \in T} f(v \to w)$$

$$\leq \sum_{v \in S, w \in T} c(v, w) = c(S, T)$$

When is $v(f) = c(S, T)$?

$f_{in}(S) = 0,\ f_{out}(S) = c(S, T)$

Max-Flow & Min-Cut

- Suppose the $c_{\min}$ is the capacity of the minimum cut in a network
- What can we say about the feasible flow we can send through it
    - cannot be more than $c_{\min}$
- In fact, whenever we find any s-t flow f and any s-t cut (S, T) such that, $v(f) = c(S, T)$ we can conclude that:
    - f is the maximum flow, and,
    - (S, T) is the minimum cut
- The question now is, given any flow network with min cut $c_{\min}$, is it always possible to route a feasible s-t flow f with $v(f) = c_{\min}$

Max-Flow Min-Cut Theorem

- A beautiful, powerful relationship between these two problems in given by the following theorem
- **Theorem**. Given any flow network G, there exists a feasible (s, t)-flow f and a (s, t)-cut (S, T) such that,

$$v(f) = c(S, T)$$

- Informally, in a flow network, the max-flow = min-cut
- This will guide our algorithm design for finding max flow
- (Will prove this theorem by construction in a bit.)

Network Flow History

- In 1950s, US military researchers Harris and Ross wrote a classified report about the rail network linking Soviet Union and Eastern Europe
    - Vertices were the geographic regions
    - Edges were railway links between the regions
    - Edge weights were the rate at which material could be shipped from one region to next
- Ross and Harris determined:
    - Maximum amount of stuff that could be moved from Russia to Europe **(max flow)**
    - Cheapest way to disrupt the network by removing rail links **(min cut)**

Network Flow History

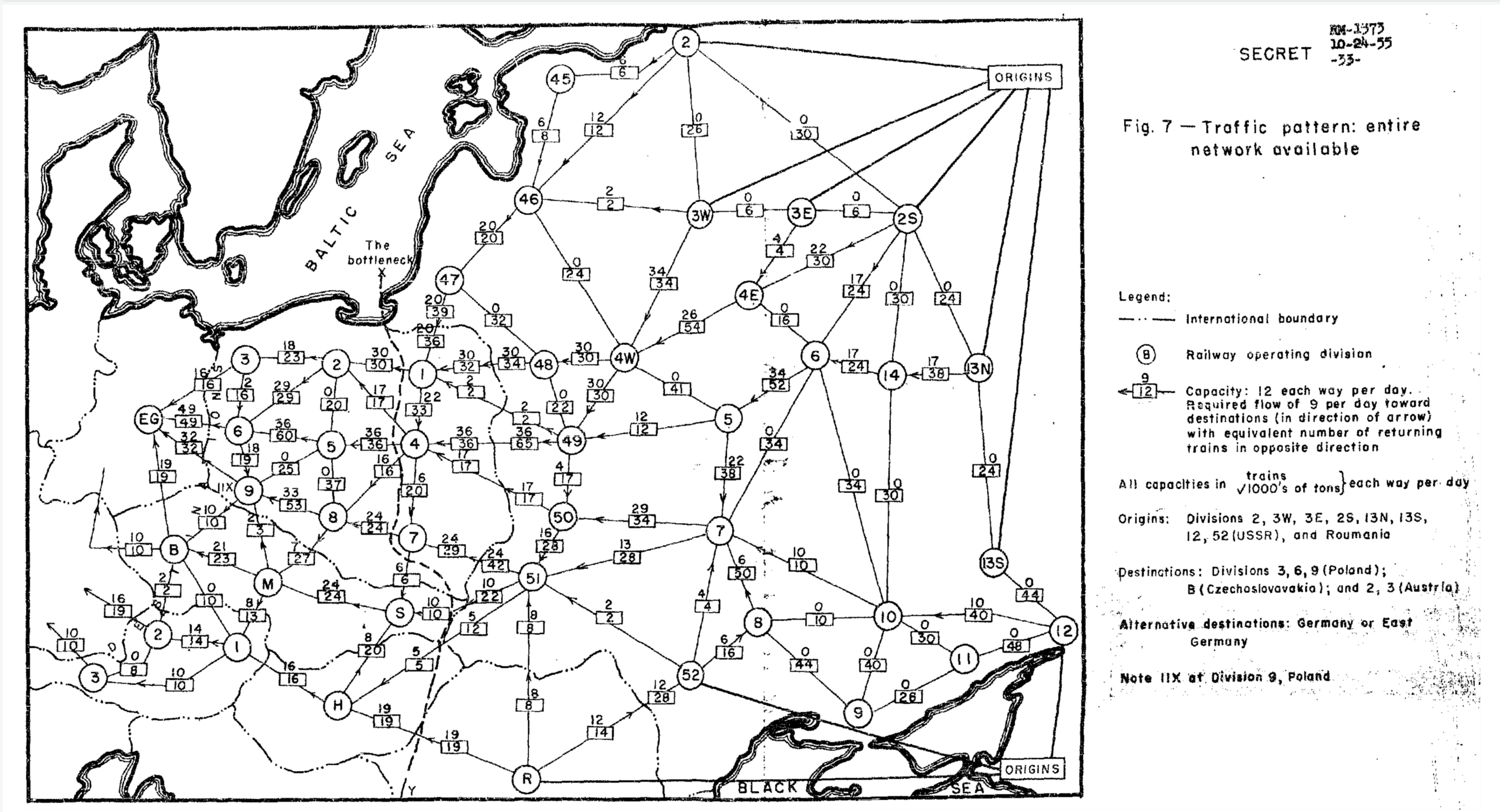


Image Credits: — Jeff Erickson's book and T[homas] E. Harris and F[rank] S. Ross. Fundamentals of a method for evaluating rail net capacities. The RAND Corporation, Research Memorandum RM-1517, October 24, 1955. United States Government work in the public domain. http://www.dtic.mil/dtic/tr/fulltext/u2/093458.pdf

Acknowledgments

- Some of the material in these slides are taken from
  - Kleinberg Tardos Slides by Kevin Wayne (https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/04GreedyAlgorithmsI.pdf)
  - Jeff Erickson's Algorithms Book (http://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf)