Introduction to Network Flows

Admin

- No Problem Set This week
  - We just took (are taking?) an exam, and you’ve earned a break

Story So Far

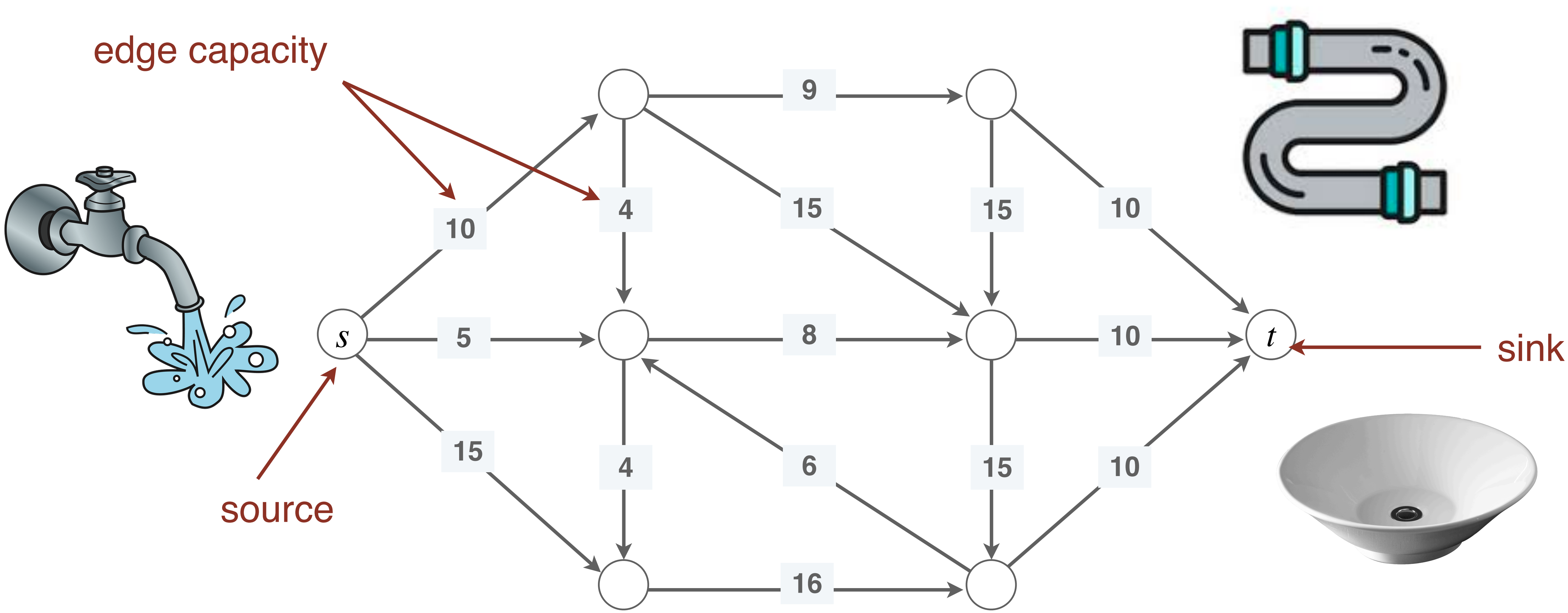
- Algorithmic design paradigms:
  - **Greedy**: often simplest algorithms to design, but only work for certain limited class of optimization problems
    - A good initial thought for most problems but rarely optimal
  - **Divide and Conquer**
    - Solving a problem by breaking it down into smaller subproblems and (often) combining results
  - **Dynamic programming**
    - Recursion with memoization: avoiding repeated work
    - Trade space (memoization structure representation) for time

New Algorithmic Paradigm

- **Network flows** model a variety of optimization problems
- These optimization problems look complicated with lots of constraints
  - At first they may 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** (have we seen any reductions before?)
- We'll 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 positive integers**
    - Will revisit this assumption & what happens otherwise
- Directed edge (u, v) written as $u \rightarrow v$
- For simplifying expositions, we will sometimes write $c(u \rightarrow v) = 0$ when $(u, v) \notin E$

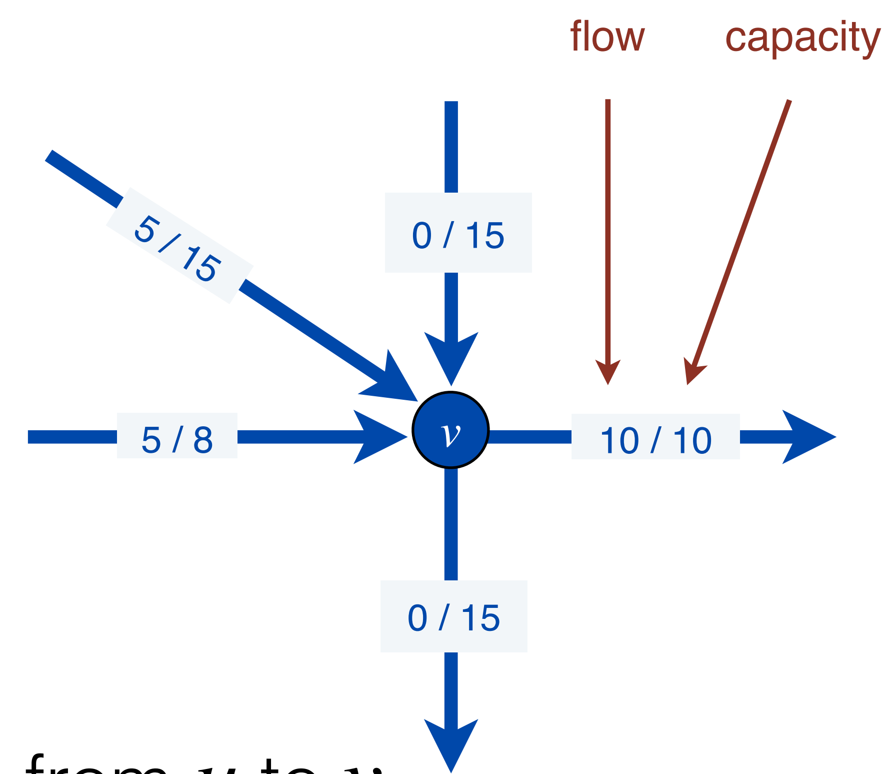
What's a Flow?

- 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

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

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

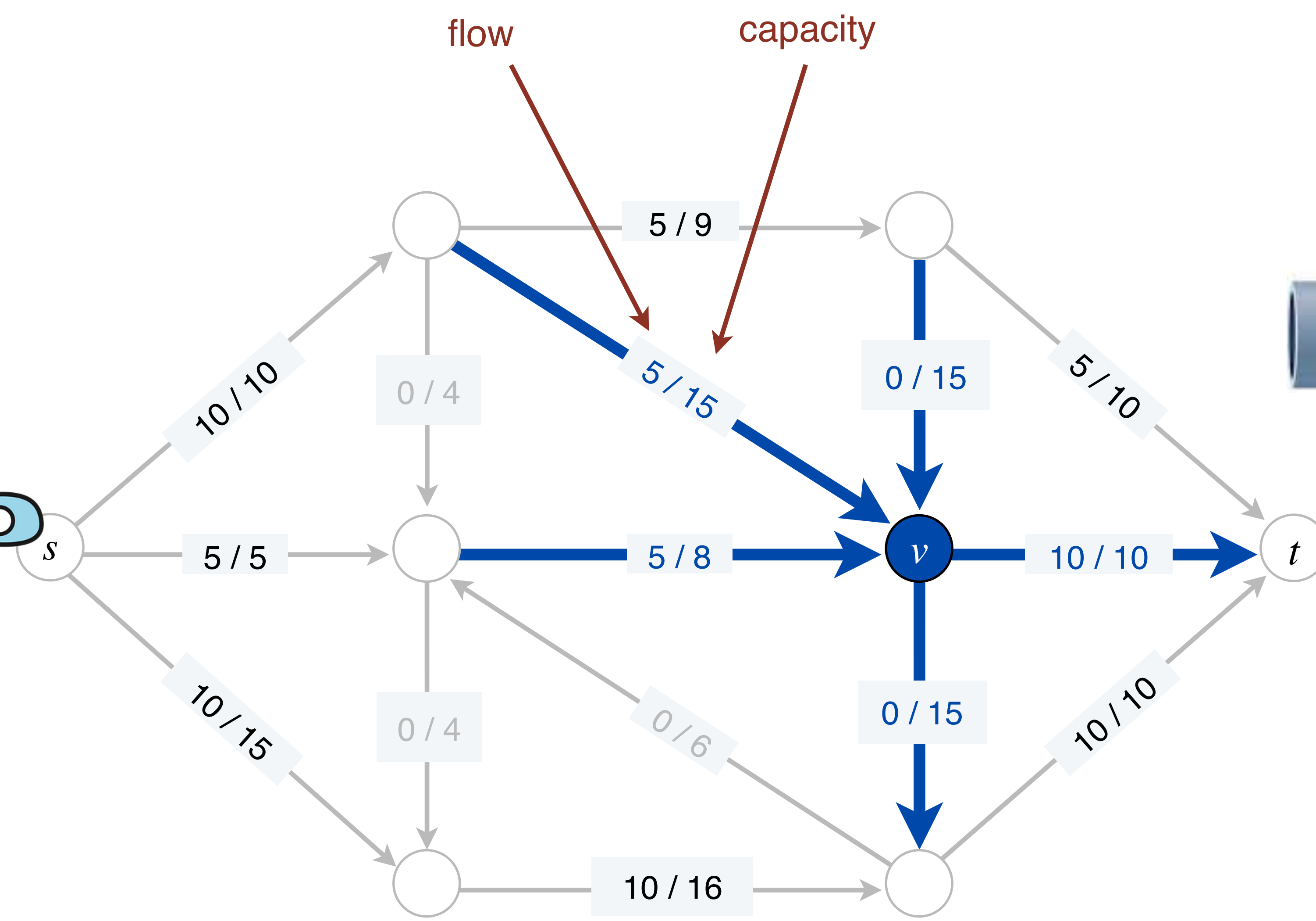


- To simplify, $f(u \to v) = 0$ if there is no edge from u to v

Feasible Flow

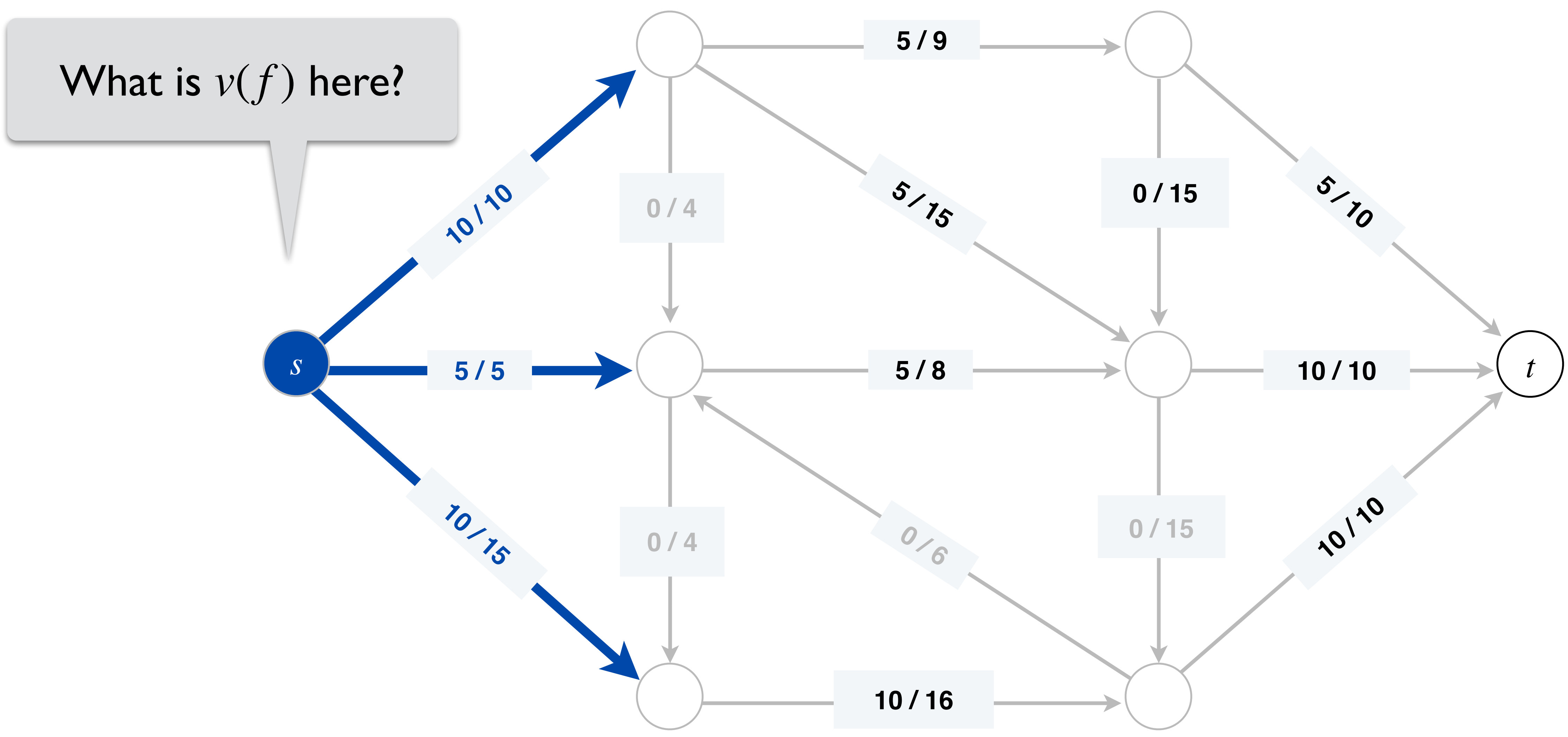
- And second, a feasible flow must satisfy the capacity constraints of the network, that is,

[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 $f_{out}(s)$.

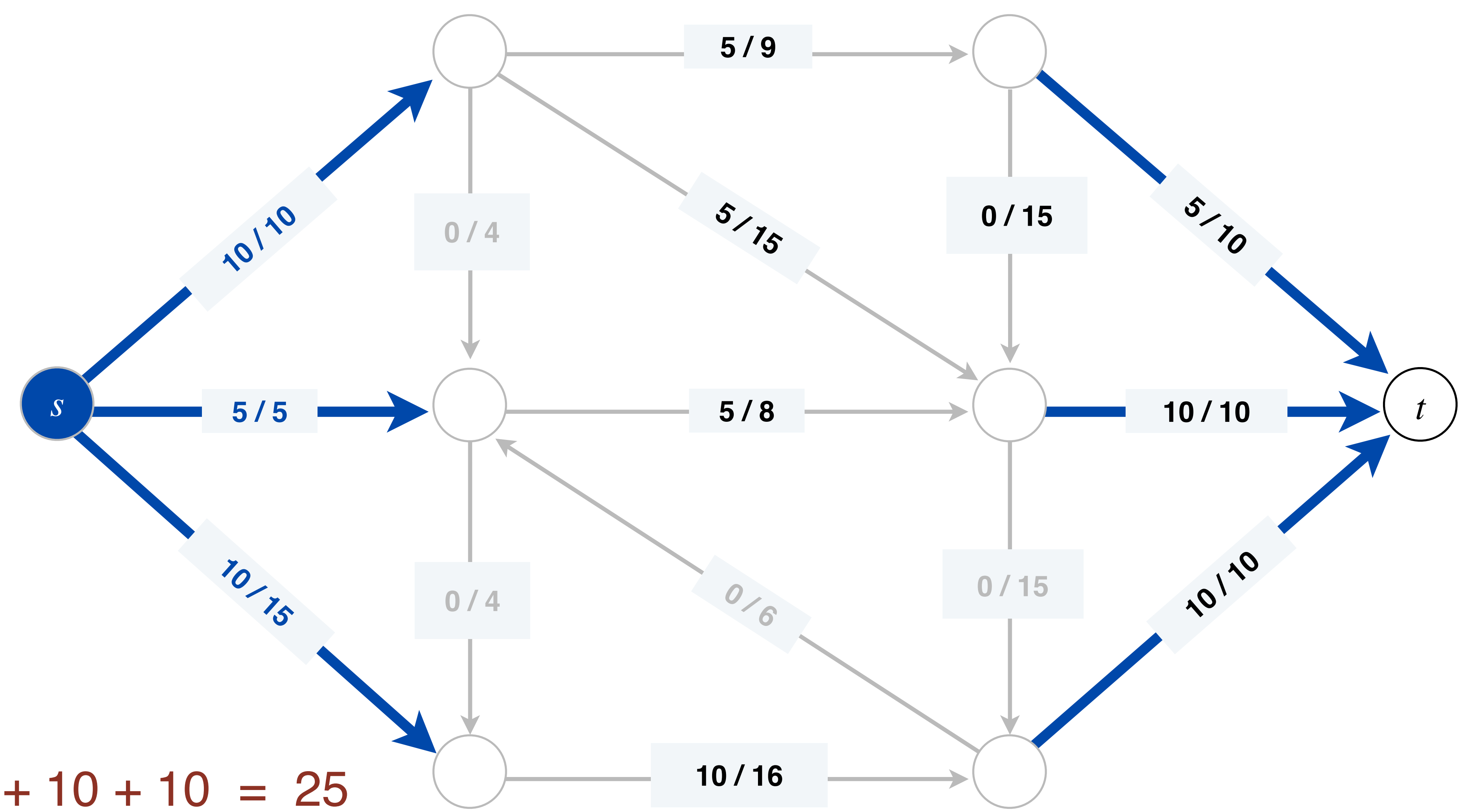


$v(f) = 5 + 10 + 10 = 25$

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?

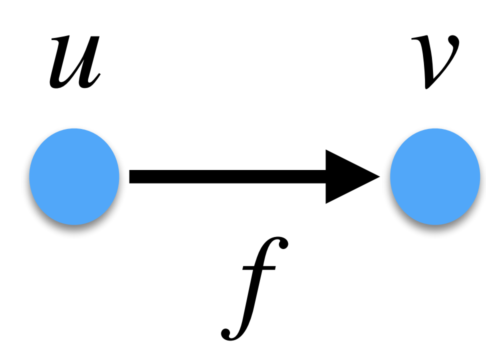


value = 5 + 10 + 10 = 25

Value of a Flow

Lemma. $f_{out}(s) = f_{in}(t)$

Proof. Let $f(E) = \sum_{e \in E} f(e)$

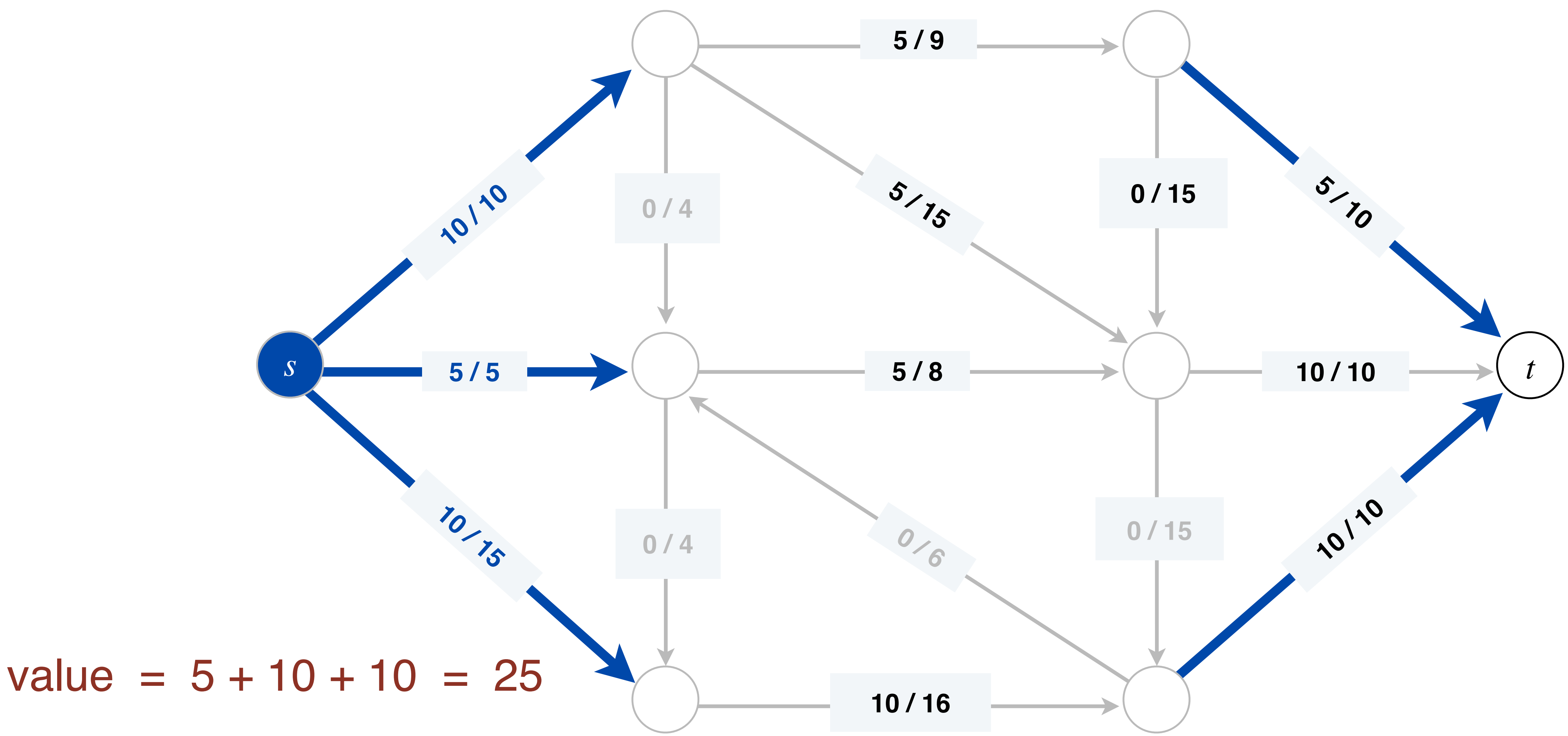


- Then, $\sum_{v \in V} f_{in}(v) = f(E) = \sum_{v \in V} f_{out}(v)$
- For every $v \neq s, t$ flow conservation implies $f_{in}(v) = f_{out}(v)$
- Thus all terms cancel out on both sides except

  $f_{in}(s) + f_{in}(t) = f_{out}(s) + f_{out}(t)$
- But $f_{in}(s) = f_{out}(t) = 0$ $\blacksquare$

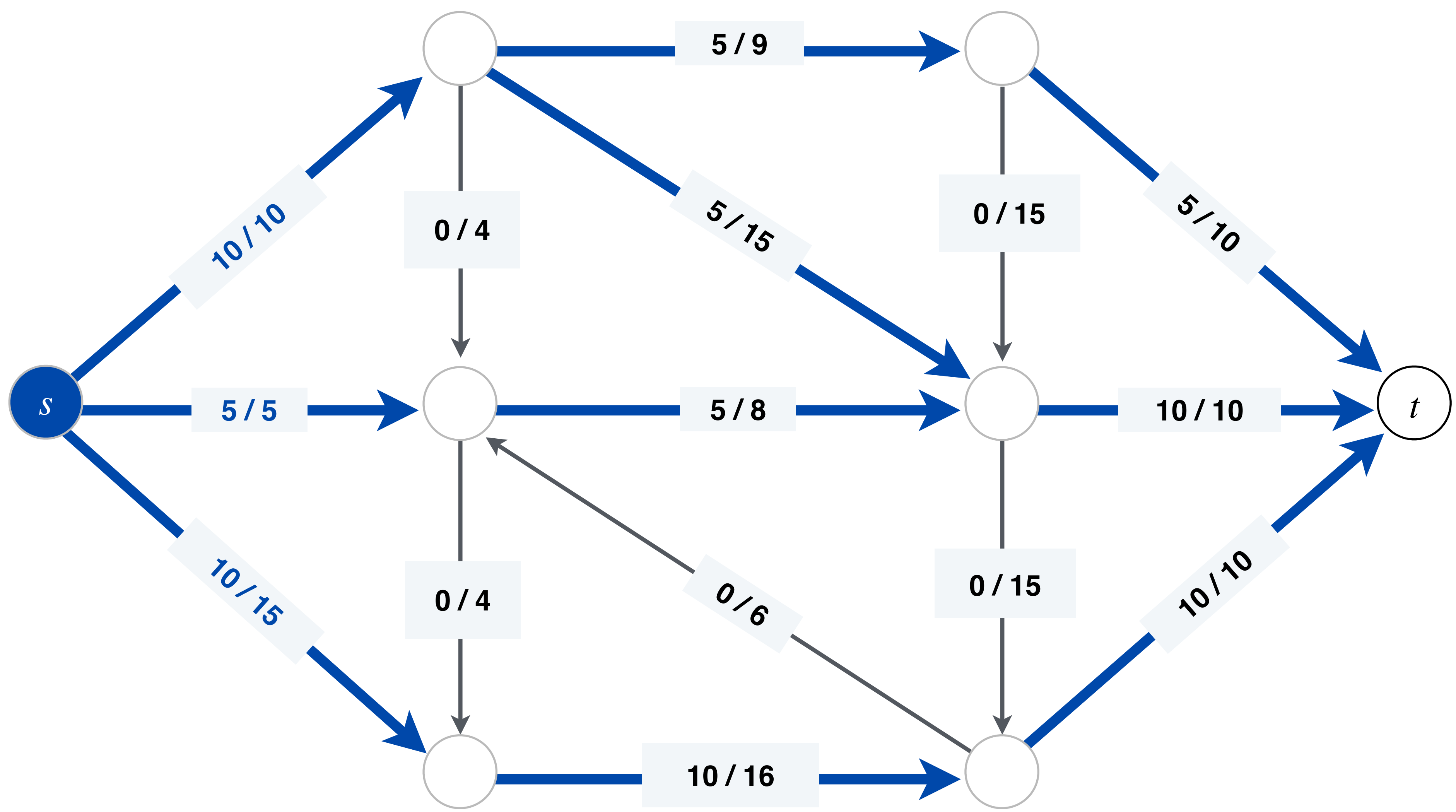
Value of a Flow

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



Max-Flow Problem

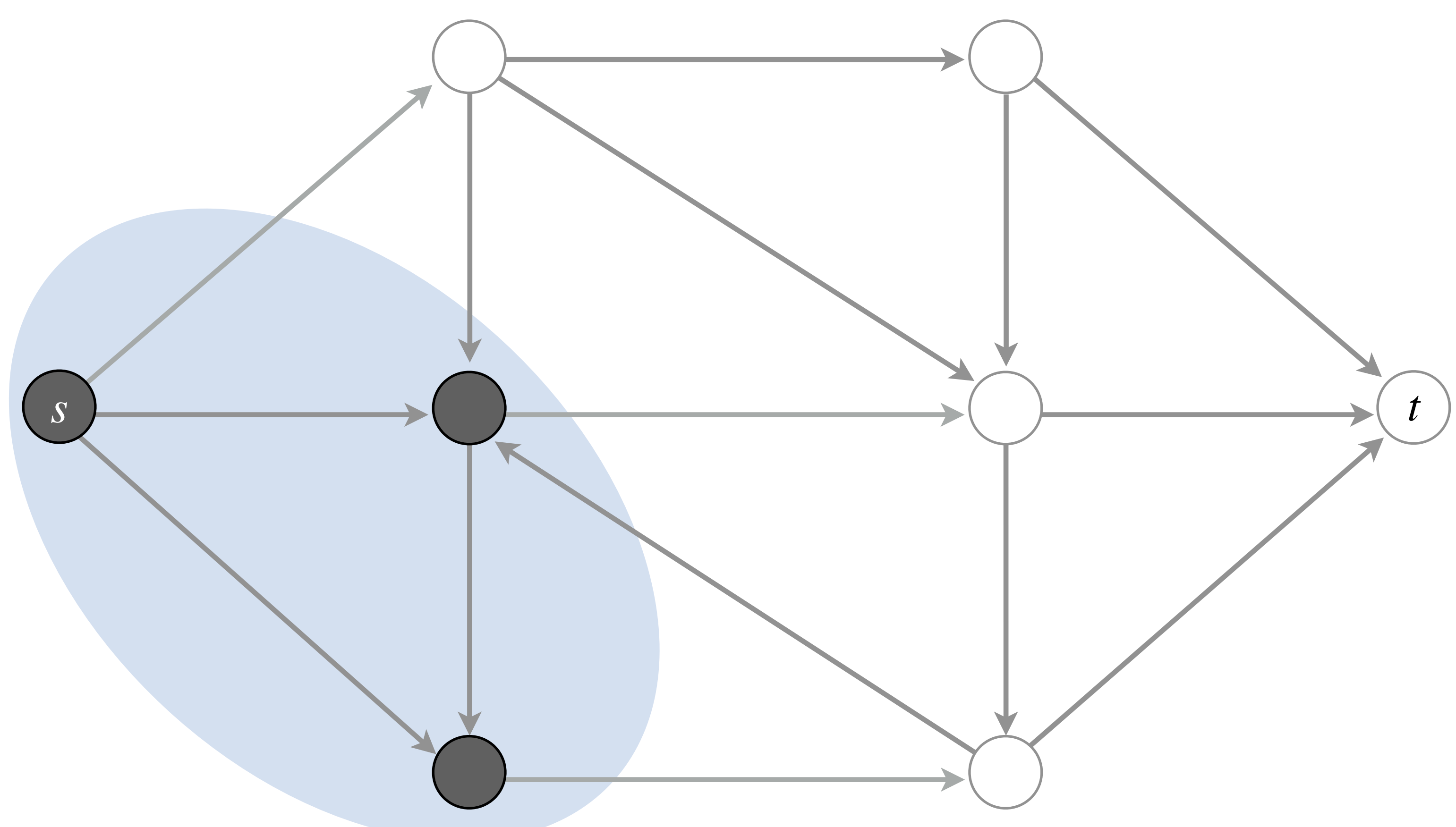
- **Problem**. Given an s-t flow network, find a feasible s-t flow of **maximum** value.



Minimum Cut Problem

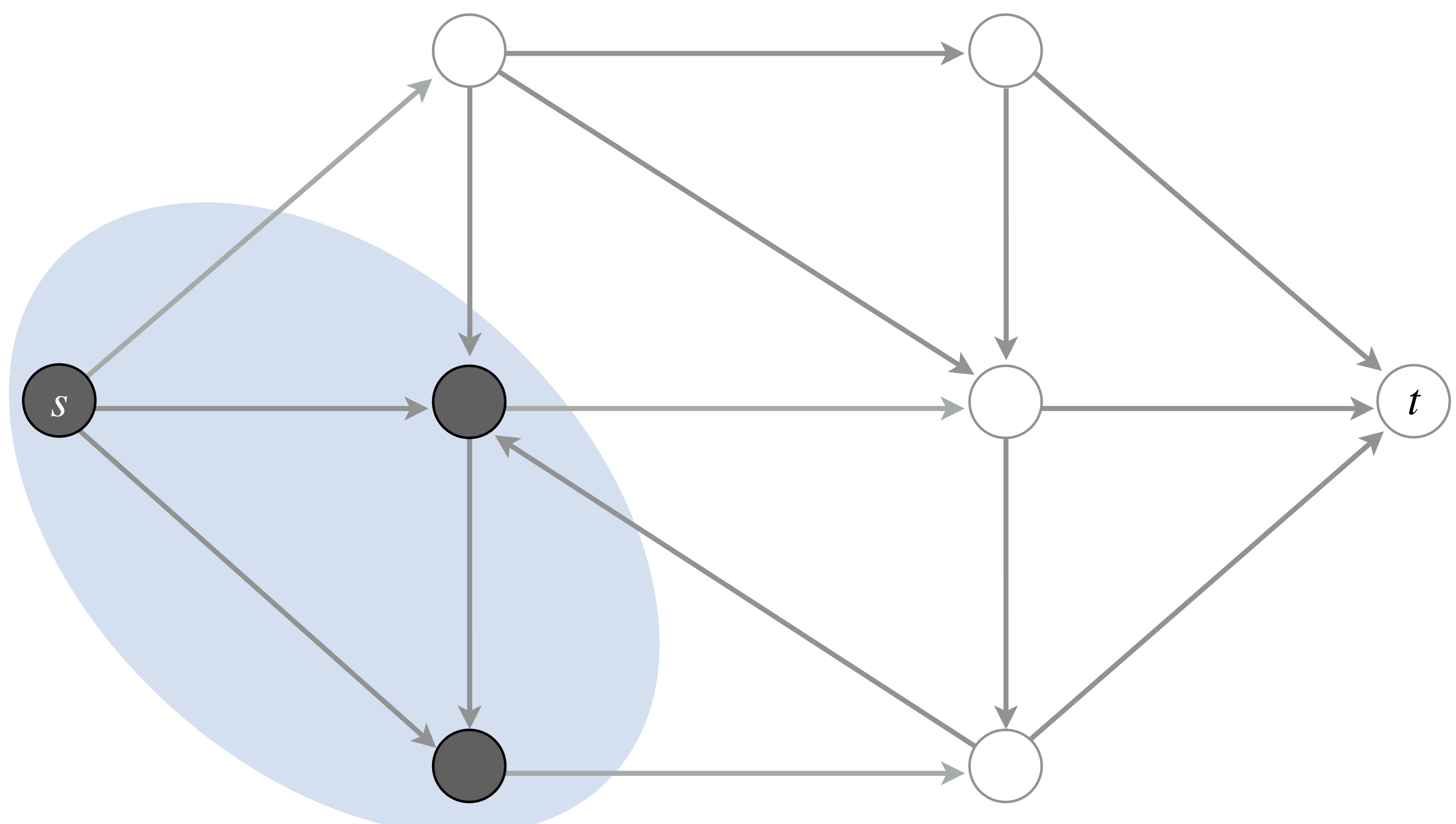
Cuts are Back!

- Cuts in graphs played a key 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 $S \cup T = V$, $S \cap T = \emptyset$ and S, T are non-empty.
- **Definition**. An (s, t)-*cut* is a cut (S, T) s.t. $s \in S$ and $t \in T$.



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?

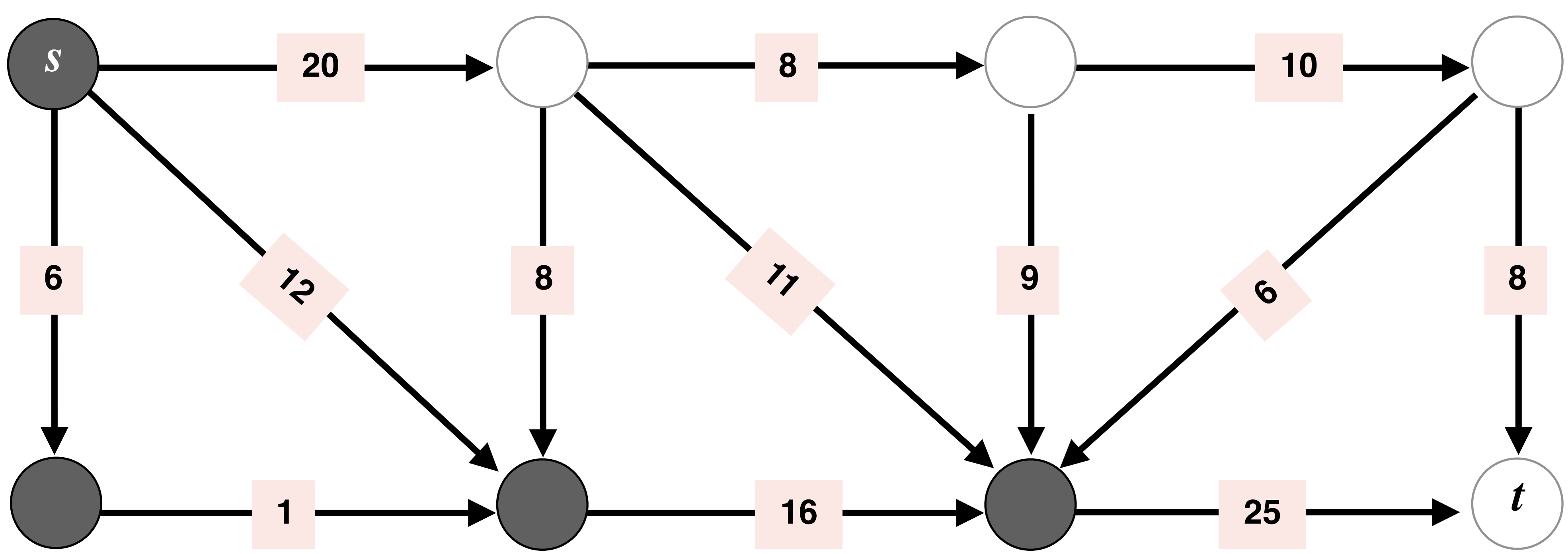
$$c(S,T) = \sum_{v \in S, w \in T} c(v \to w)$$

A. 11 (20 + 25 – 8 – 11 – 9 – 6)

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

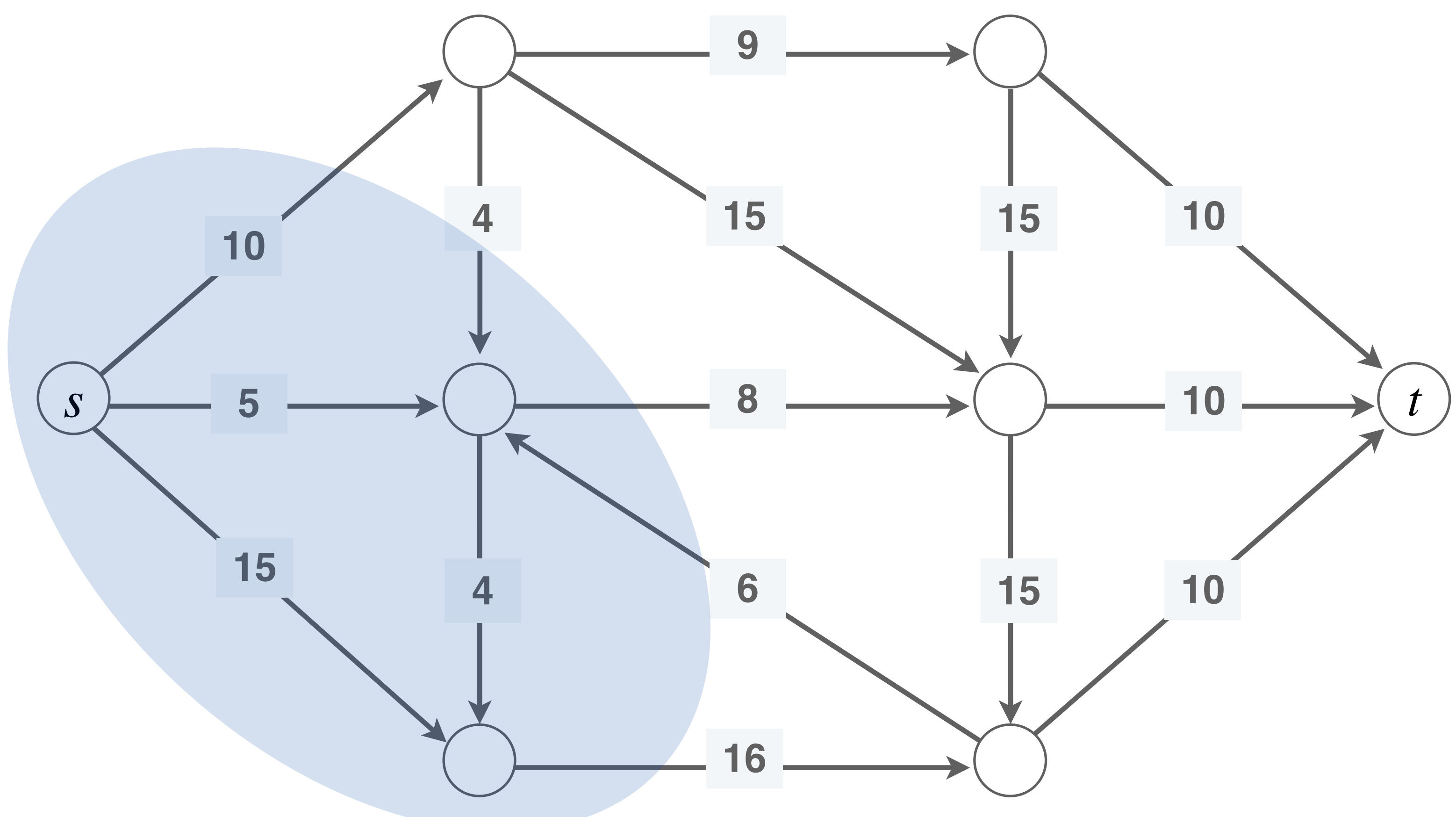
C. 45 (20 + 25)

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



Min Cut Problem

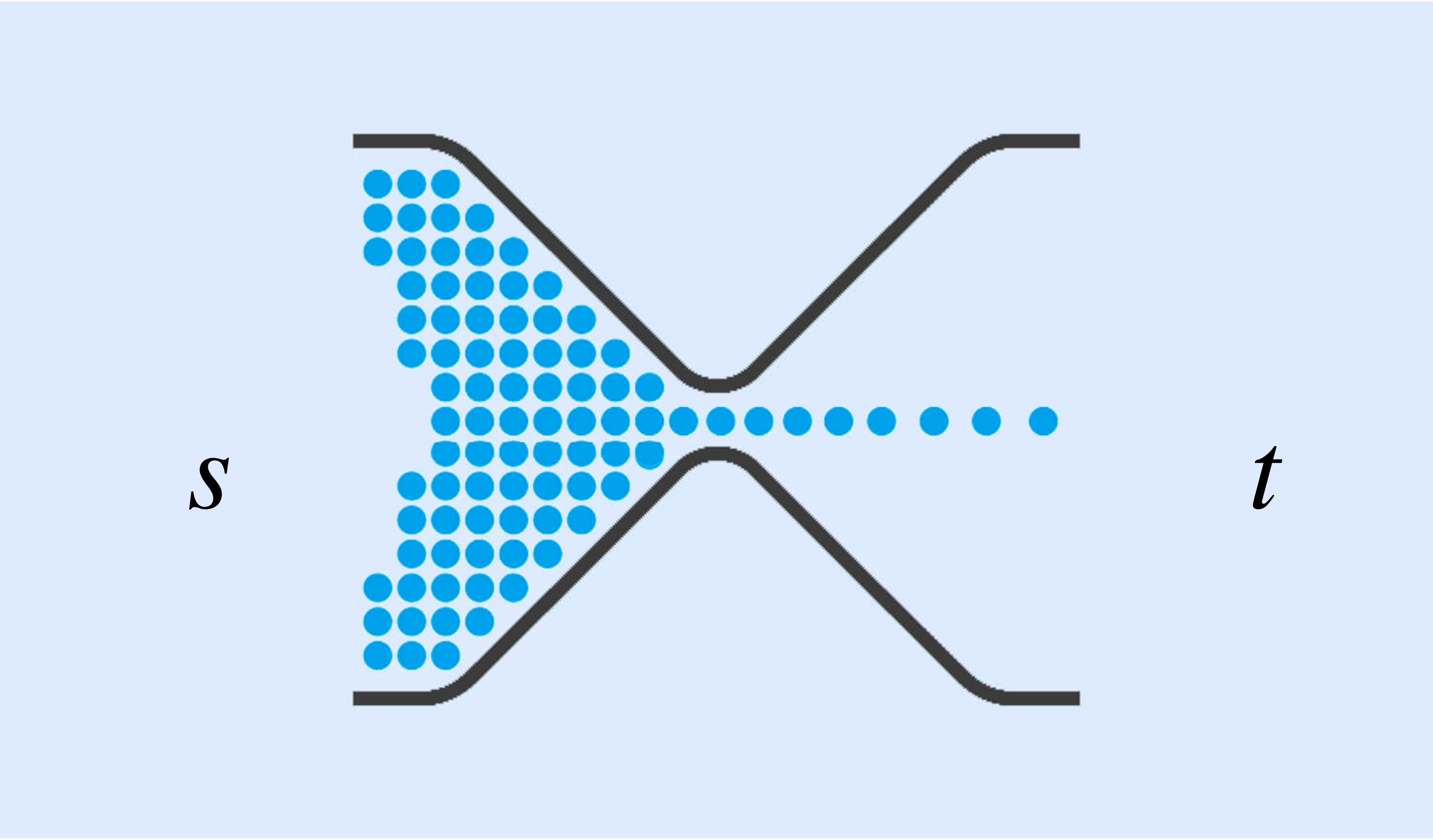
- **Problem**. Given an s-t flow network, find an s-t cut of **minimum** capacity.



Relationship between Flows and Cuts

Flows and Cuts

- Cuts represent "**bottlenecks**" in a flow network
- For any (s, t)-cut, all flow needs to “exit” S to get to t
- We will formalize this intuition



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)