https://algs4.cs.princeton.edu

6.4 Maximum Flow

- *introduction*
- *Ford–Fulkerson algorithm*
- *maxflow–mincut theorem*
- *analysis of running time*
- *Java implementation (see textbook)*
- *applications*

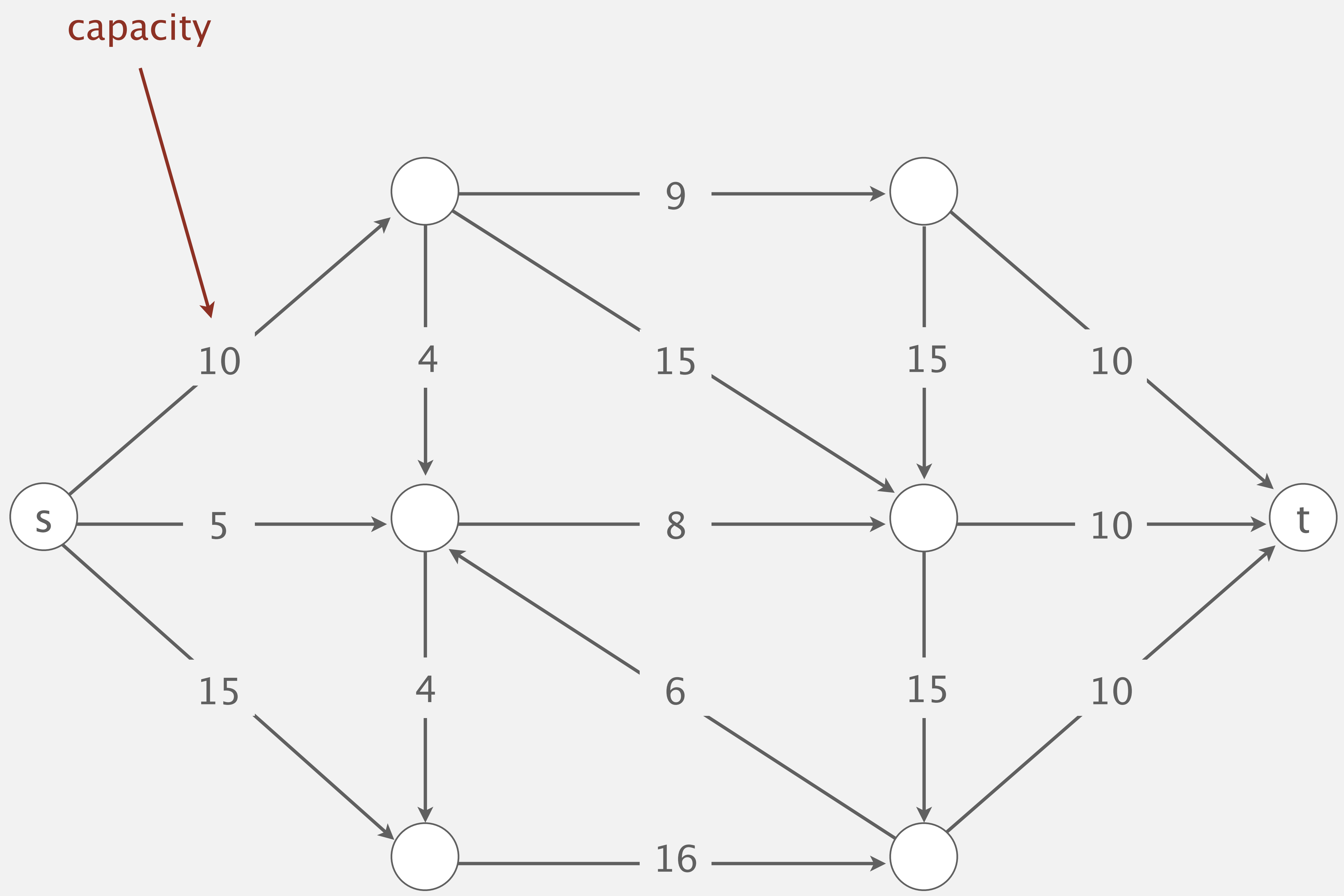
https://algs4.cs.princeton.edu

6.4 Maximum Flow

- *introduction*
- *Ford–Fulkerson algorithm*
- *maxflow–mincut theorem*
- *analysis of running time*
- *Java implementation*
- *applications*

Mincut problem

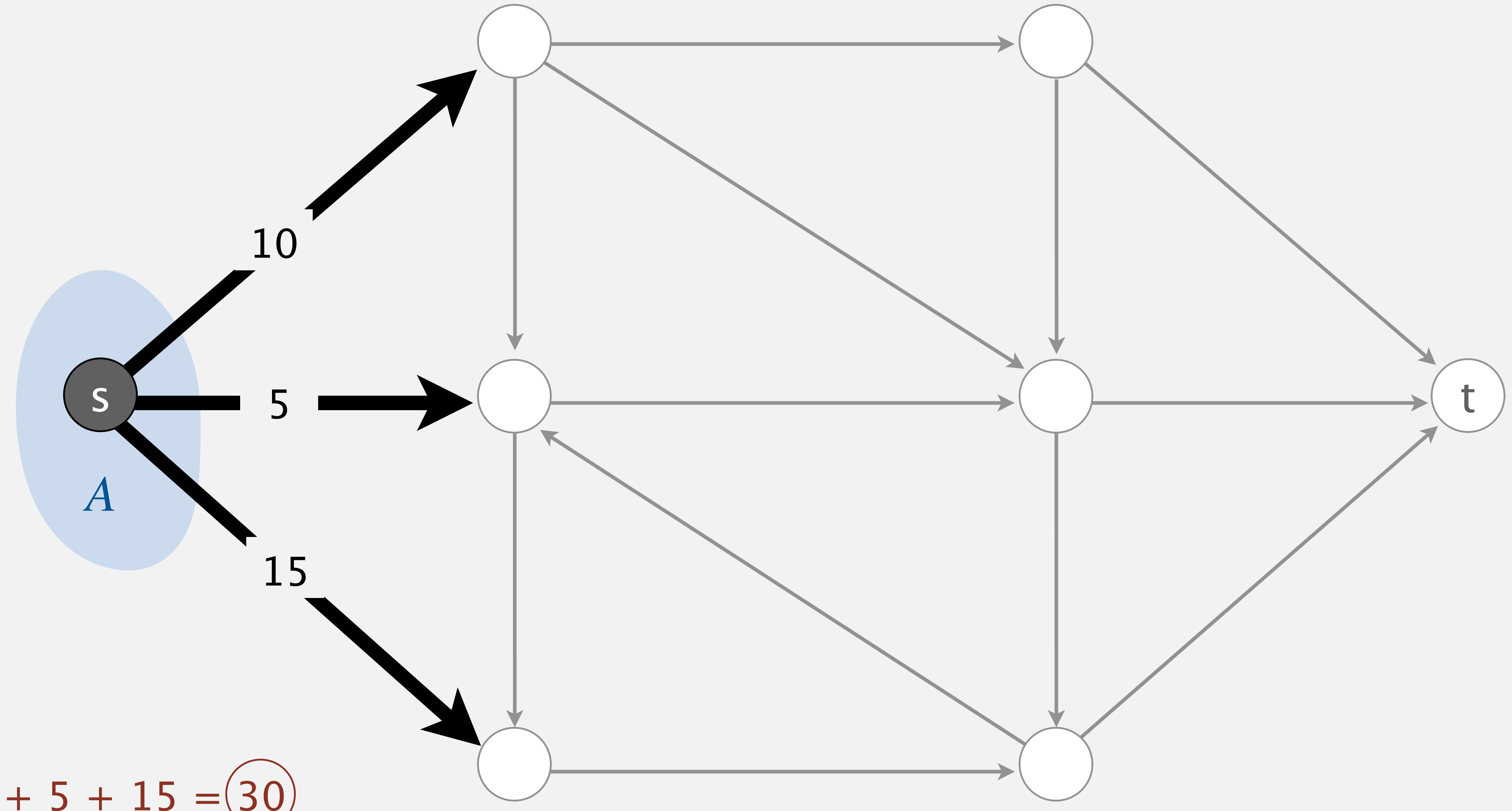
Input. A digraph with positive edge weights, source vertex s, and target vertex t.



Mincut problem

Def. A *st*-cut (cut) is a partition of the vertices into two disjoint sets, with s in one set A and t in the other set B.

Def. Its capacity is the sum of the capacities of the edges from A to B.

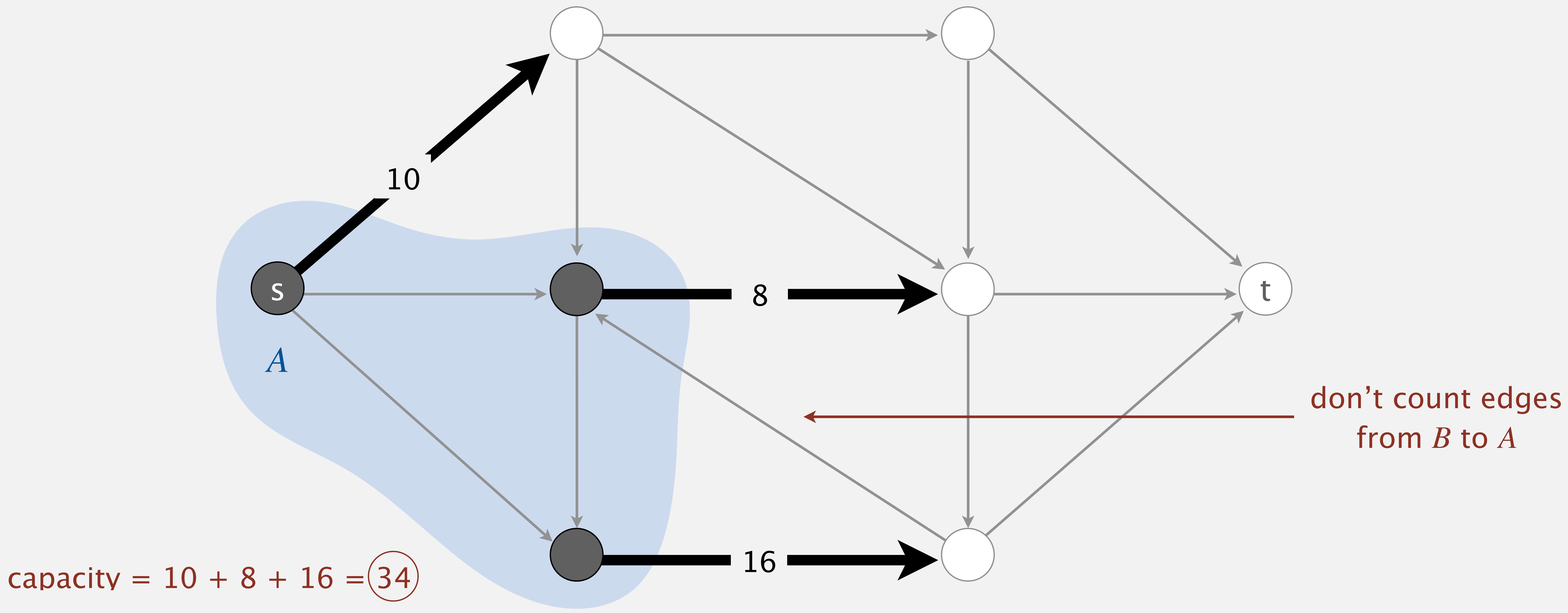


capacity = 10 + 5 + 15 = 30

Mincut problem

Def. A *st*-cut (cut) is a partition of the vertices into two disjoint sets, with s in one set A and t in the other set B.

Def. Its capacity is the sum of the capacities of the edges from A to B.

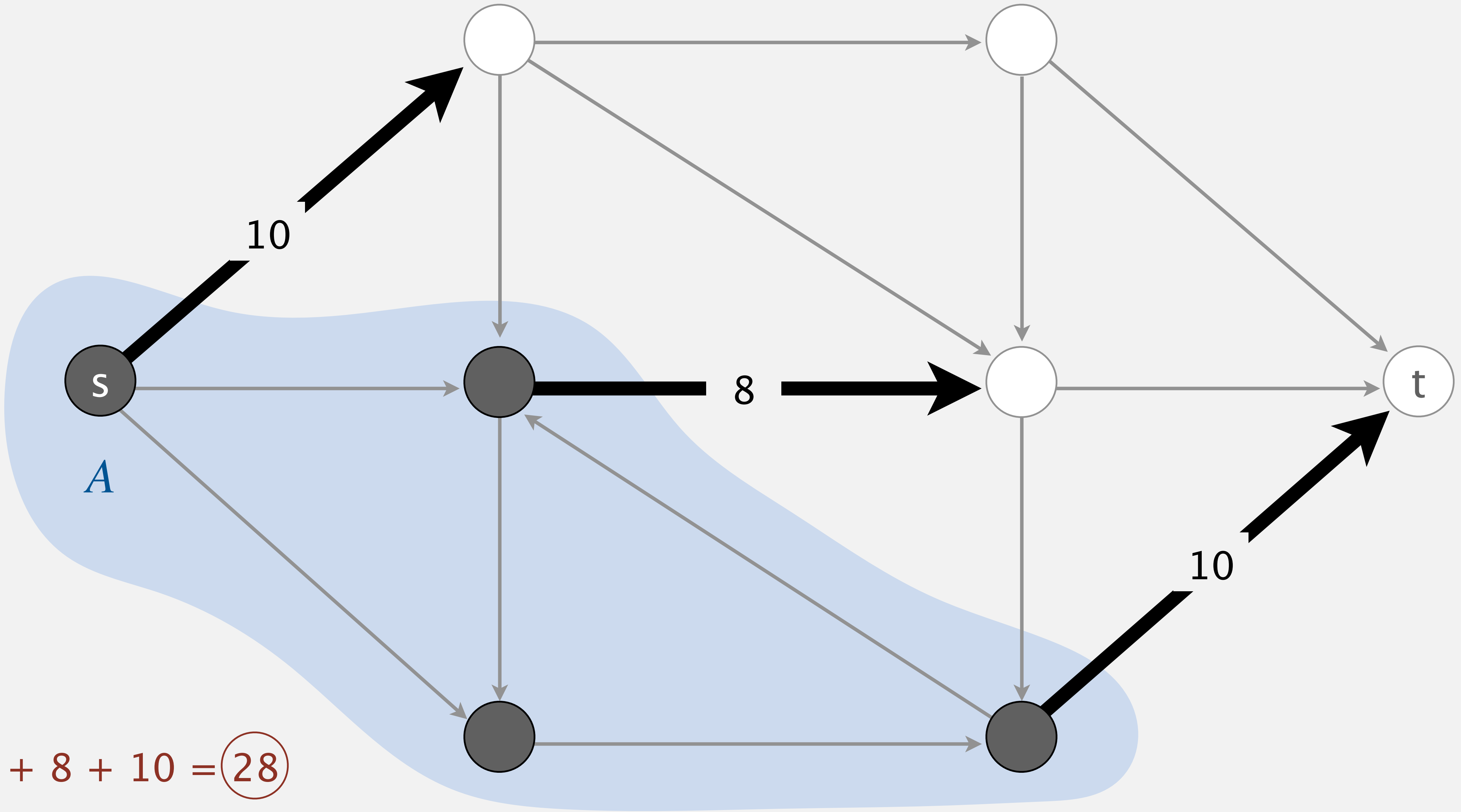


Mincut problem

Def. A st-cut (cut) is a partition of the vertices into two disjoint sets, with s in one set A and t in the other set B.

Def. Its capacity is the sum of the capacities of the edges from A to B.

Minimum st-cut (mincut) problem. Find a cut of minimum capacity.



capacity = 10 + 8 + 10 = 28

Maxflow: quiz 1

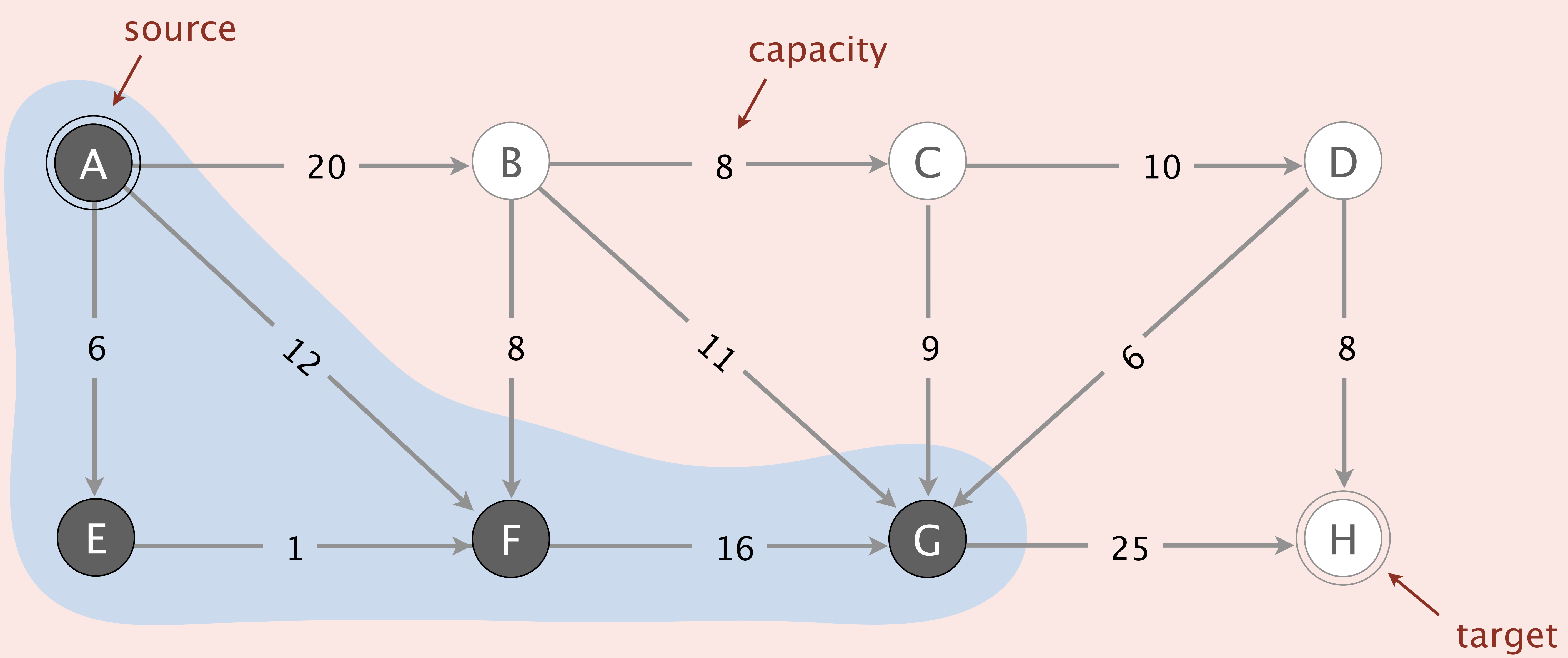
What is the capacity of the cut $\{A, E, F, G\}$?

A. 11 (20 + 25 − 8 − 11 − 9 − 6)

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

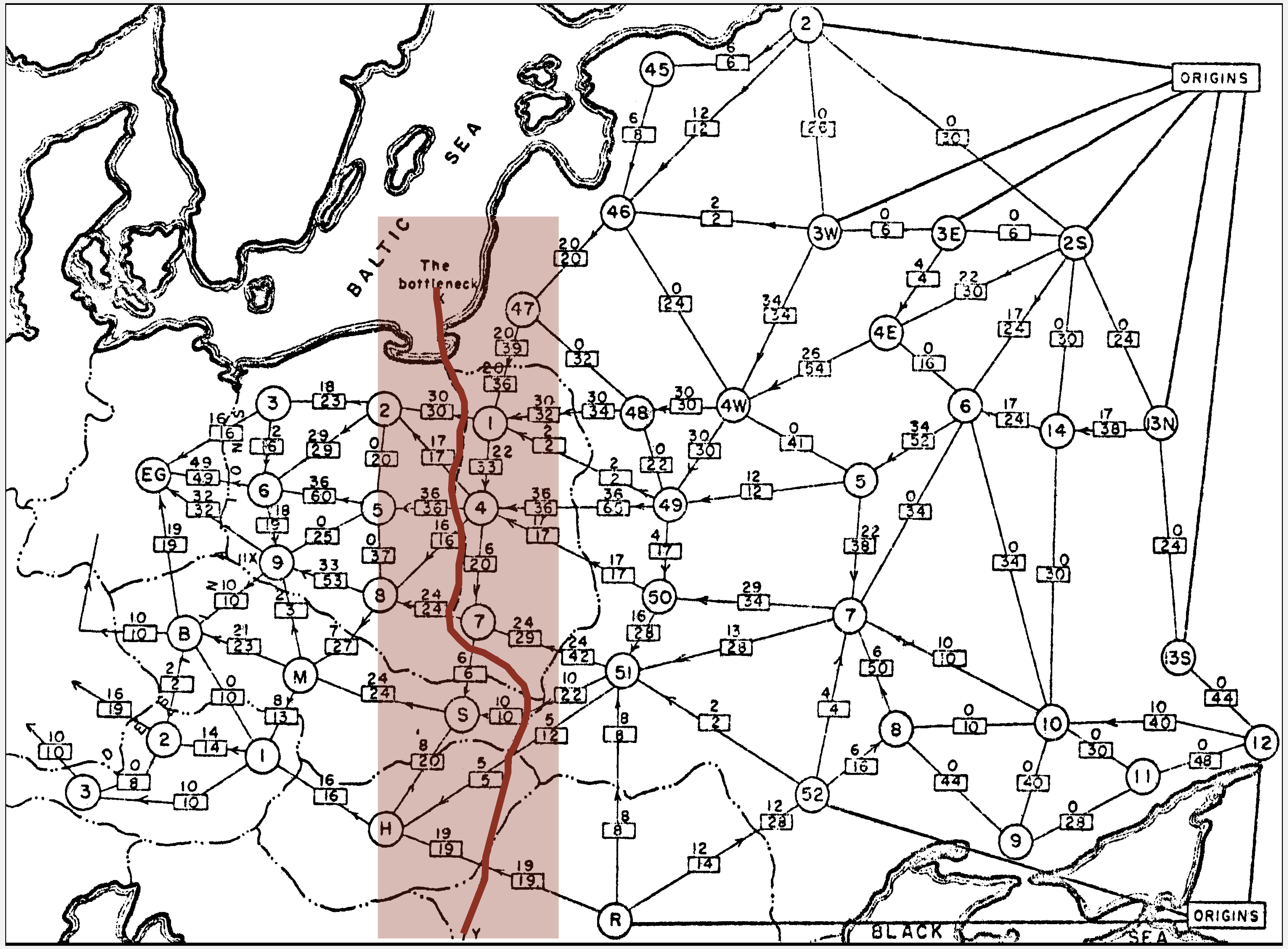
C. 45 (20 + 25)

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



Mincut application (RAND 1950s)

“Free world” goal. Disrupt rail network (if Cold War turns into real war).



rail network connecting Soviet Union with Eastern European countries

(map declassified by Pentagon in 1999)

Maxflow problem

Though maximum flow algorithms have a long history, revolutionary progress is still being made.

BY ANDREW V. GOLDBERG AND ROBERT E. TARJAN

Efficient Maximum Flow Algorithms

gorithms in more detail. We restrict ourselves to basic maximum flow algorithms and do not cover interesting special cases (such as undirected graphs, planar graphs, and bipartite matchings) or generalizations (such as minimum-cost and multi-commodity flow problems).

Before formally defining the maximum flow and the minimum cut problems, we give a simple example of each problem: For the maximum flow example, suppose we have a graph that represents an oil pipeline network from an oil well to an oil depot. Each arc has a capacity, or maximum number of liters per second that can flow through the corresponding pipe. The goal is to find the maximum number of liters per second (maximum flow) that can be shipped from well to depot. For the minimum cut problem, we want to find the set of pipes of the smallest total capacity such that removing the pipes disconnects the oil well from the oil depot (minimum cut).

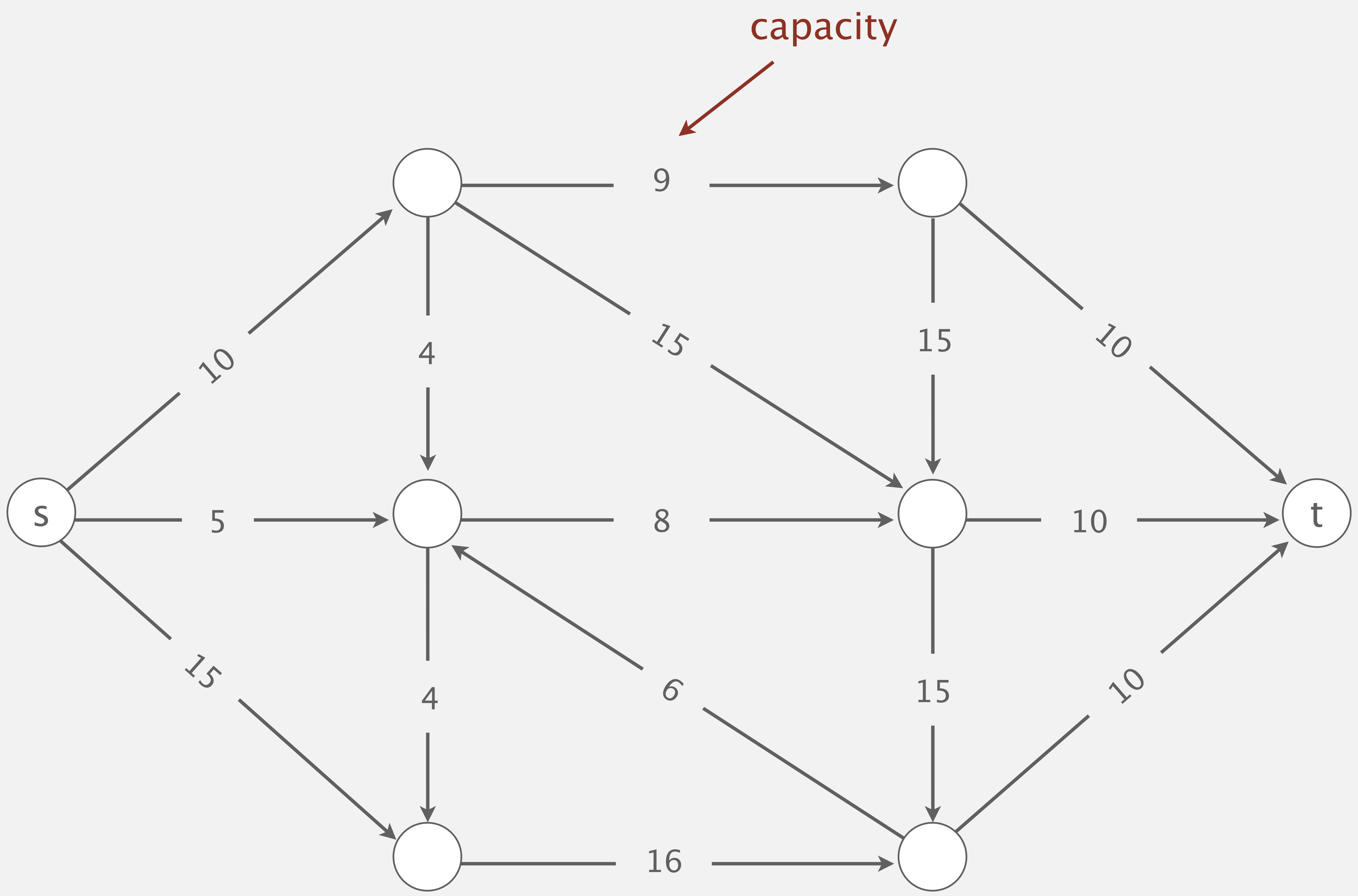
The maximum flow, minimum cut

Efficient Maximum Flow Algorithms by Andrew Goldberg and Bob Tarjan

https://vimeo.com/100774435

Maxflow problem

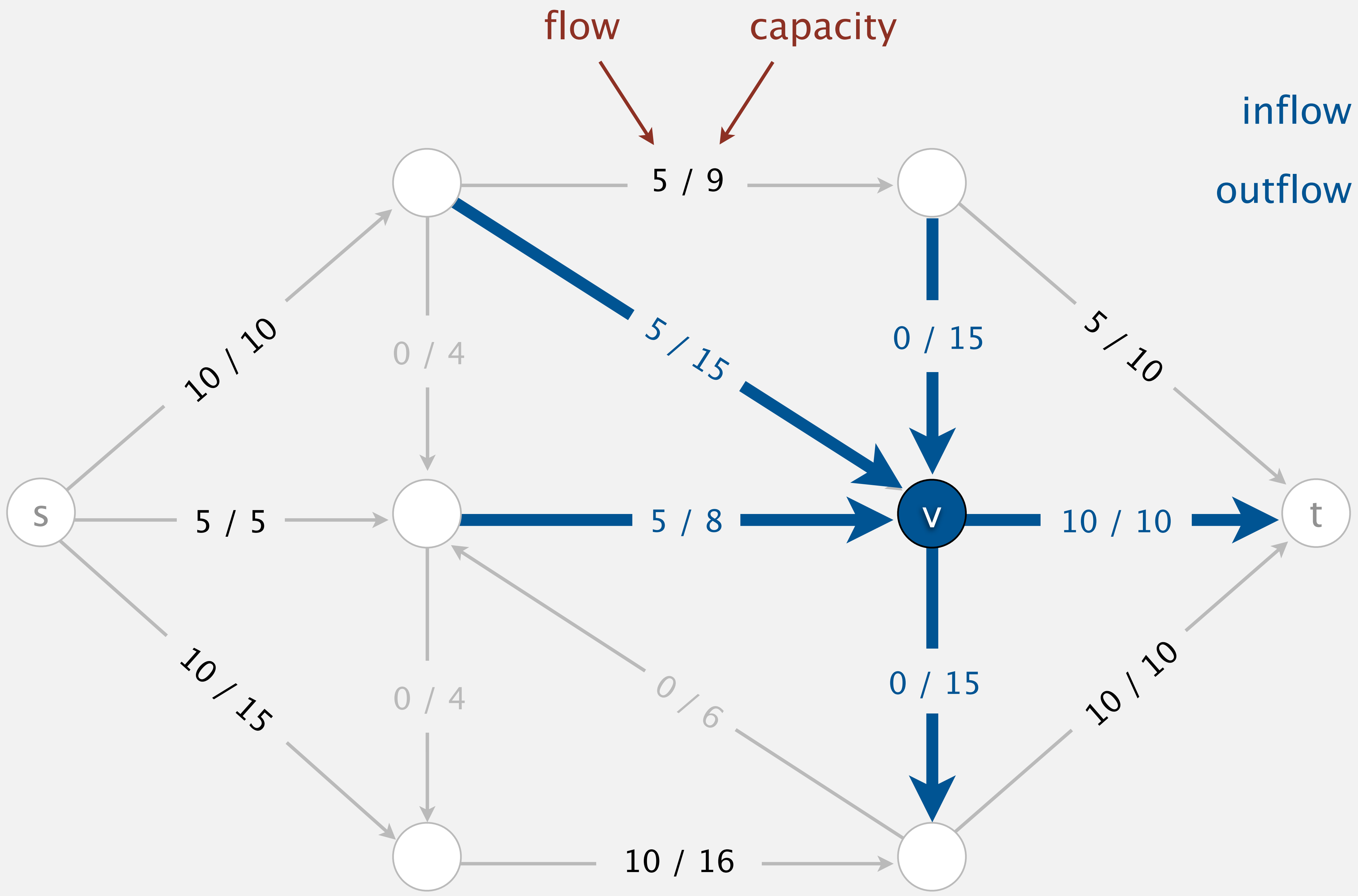
Input. A digraph with positive edge weights, source vertex s, and target vertex t.



Maxflow problem

Def. An st–flow (flow) is an assignment of values to the edges such that:

- Capacity constraints: $0 \leq$ edge's flow $\leq$ edge's capacity.
- Flow conservation constraints: inflow = outflow at every vertex (except s and t).



inflow at v = $5 + 5 + 0 = 10$

outflow at v = $10 + 0 = 10$

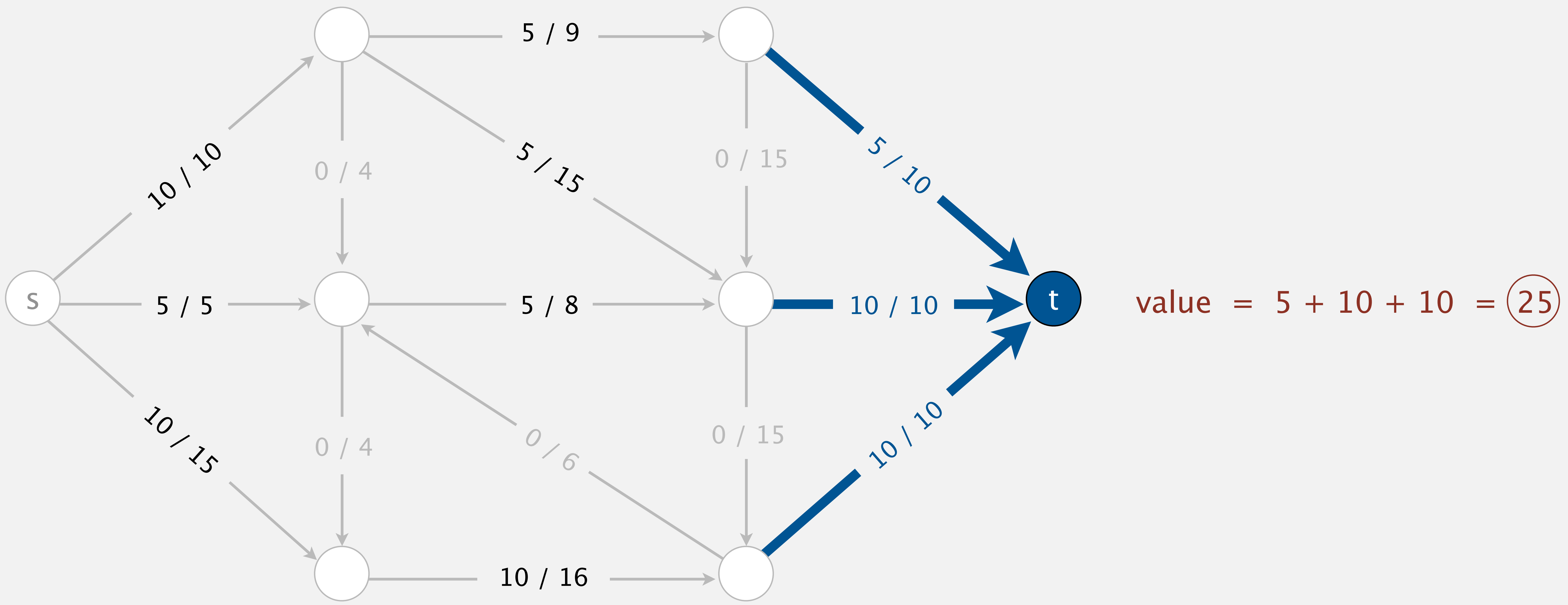
Maxflow problem

Def. An *st*-flow (flow) is an assignment of values to the edges such that:

- Capacity constraints: $0 \leq$ edge's flow $\leq$ edge's capacity.
- Flow conservation constraints: inflow = outflow at every vertex (except s and t).

Def. The value of a flow is the inflow at t.

we assume no edges incident to s or from t



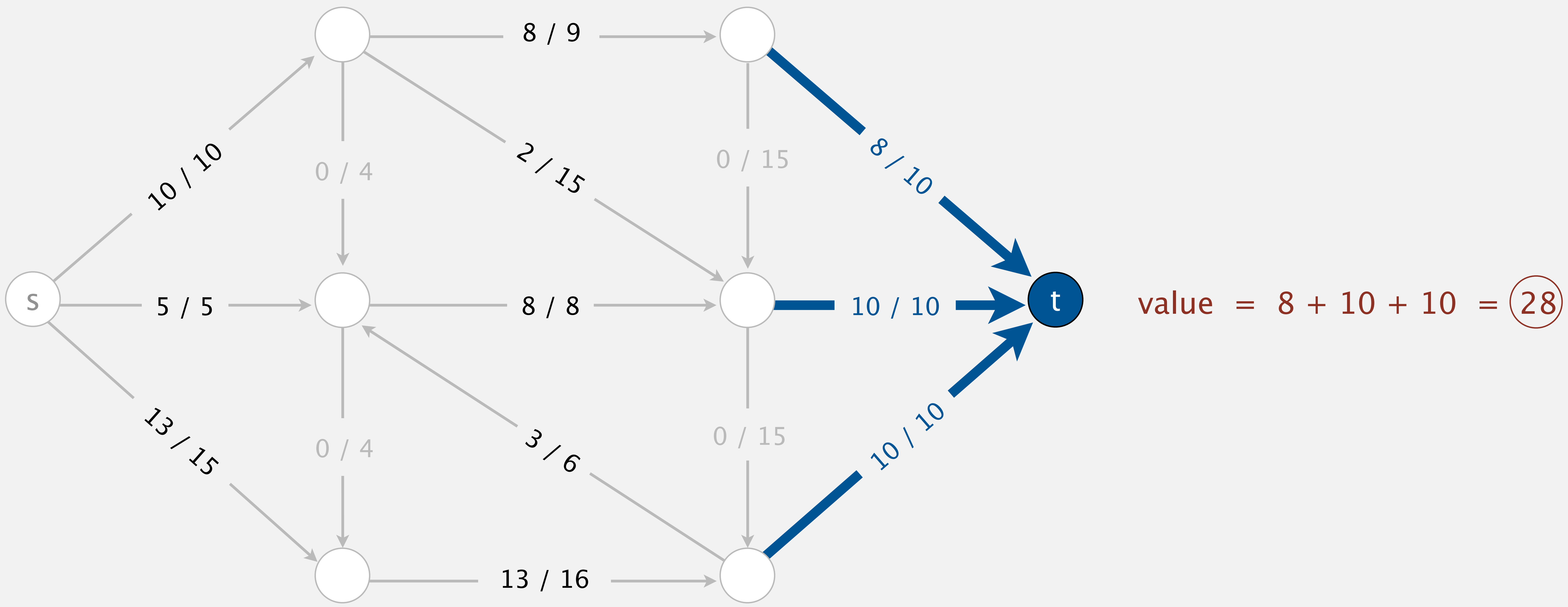
Maxflow problem

Def. An *st*-flow (flow) is an assignment of values to the edges such that:

- Capacity constraints: $0 \leq$ edge's flow $\leq$ edge's capacity.
- Flow conservation constraints: inflow = outflow at every vertex (except s and t).

Def. The value of a flow is the inflow at t.

Maximum st-flow (maxflow) problem. Find a flow of maximum value.

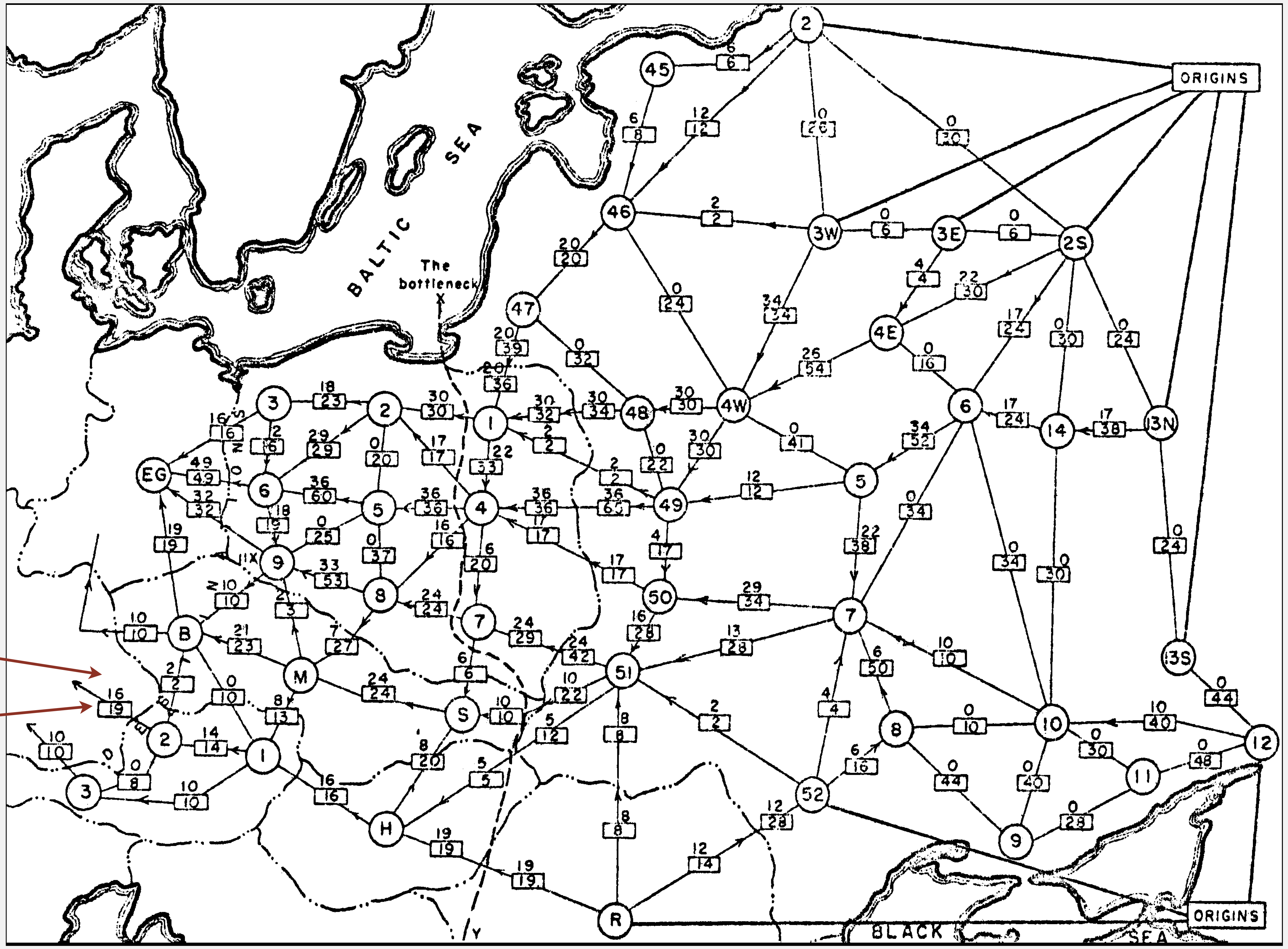


Maxflow application (Tolstoǐ 1930s)

Soviet Union goal. Maximize flow of supplies to Eastern Europe.

flow

capacity



rail network connecting Soviet Union with Eastern European countries

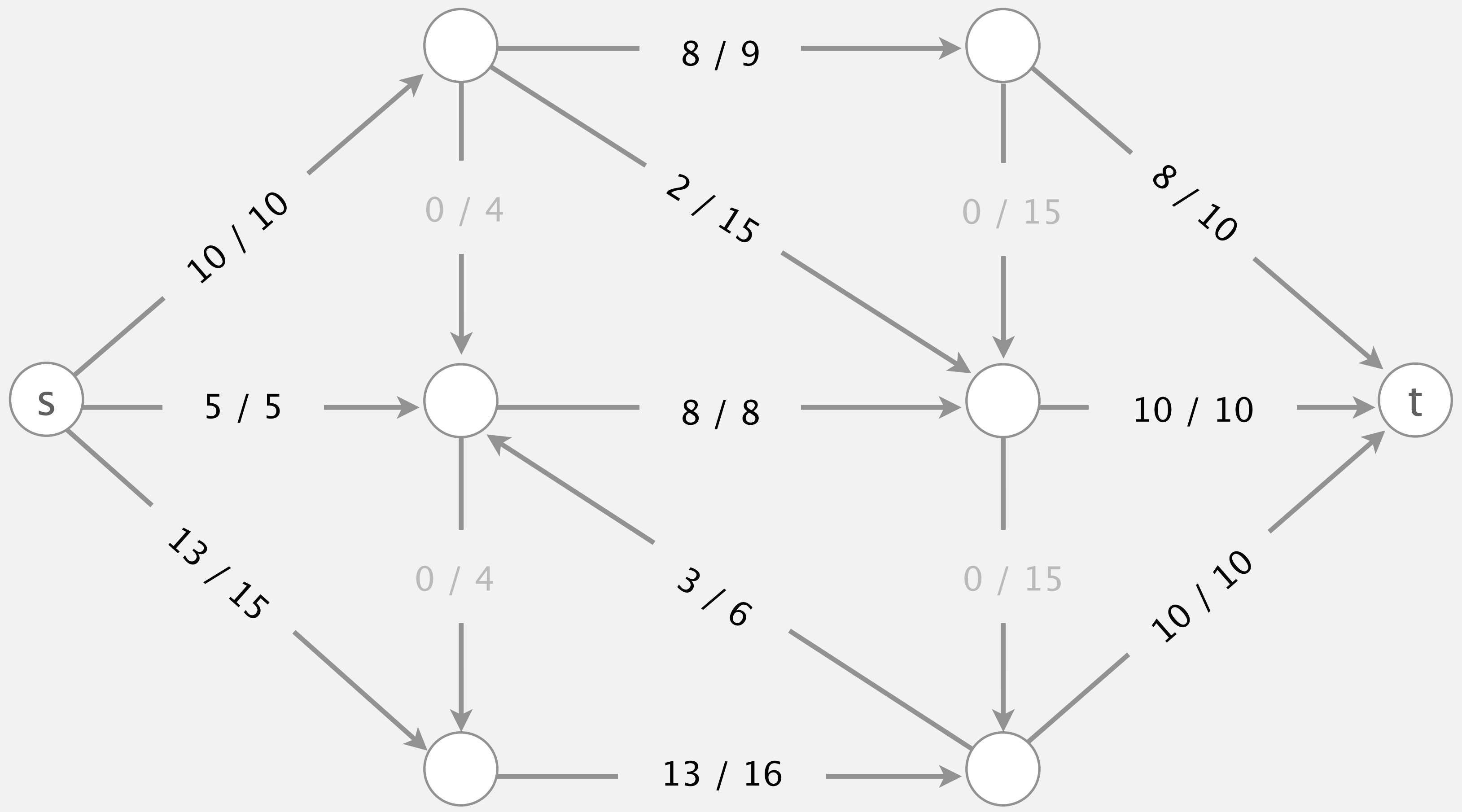
(map declassified by Pentagon in 1999)

Summary

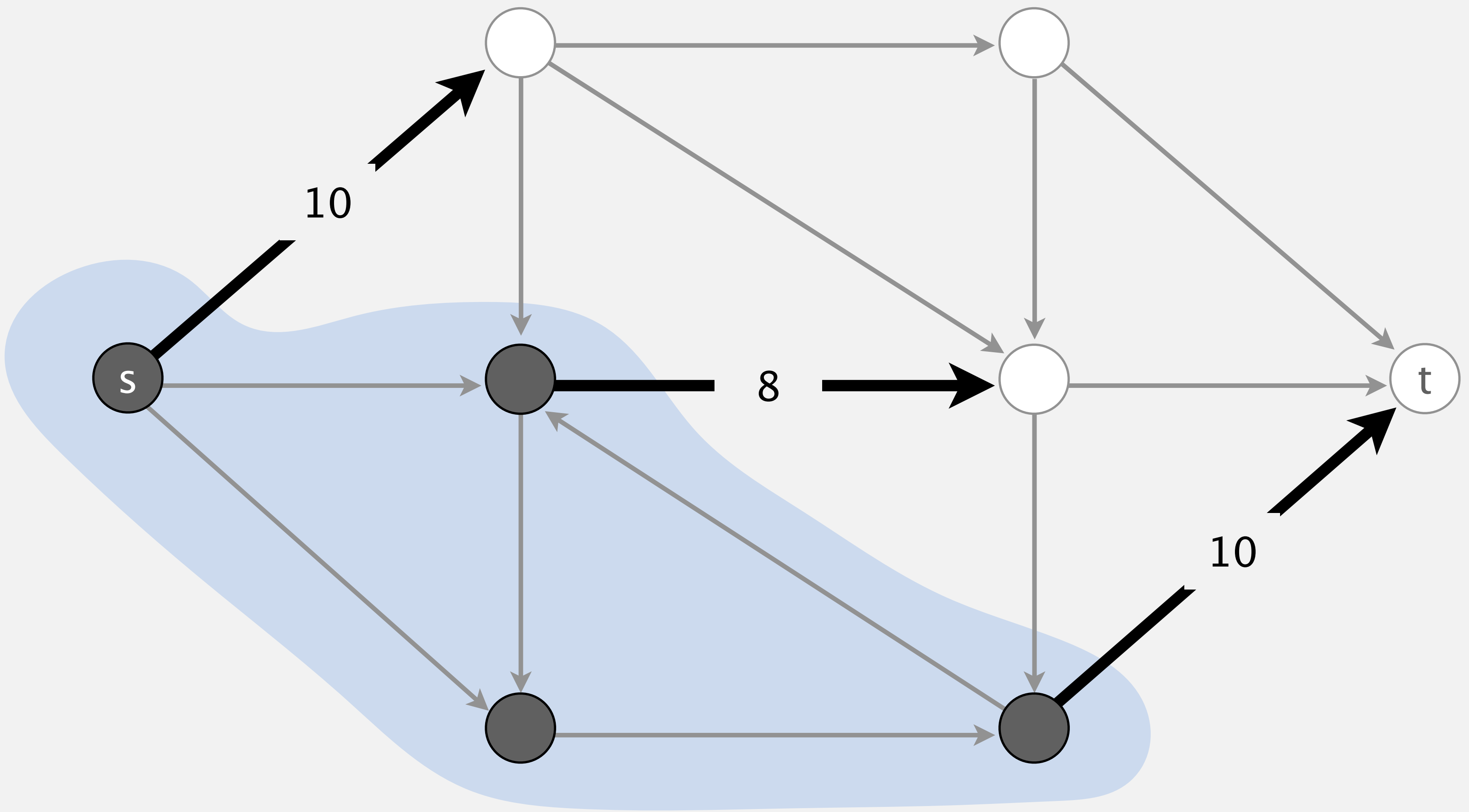
Input. A digraph with positive edge weights, source vertex s, and target vertex t.

Mincut problem. Find a cut of minimum capacity.

Maxflow problem. Find a flow of maximum value.



value of flow = 28



capacity of cut = 28

Remarkable fact. These two problems are dual! [stay tuned]

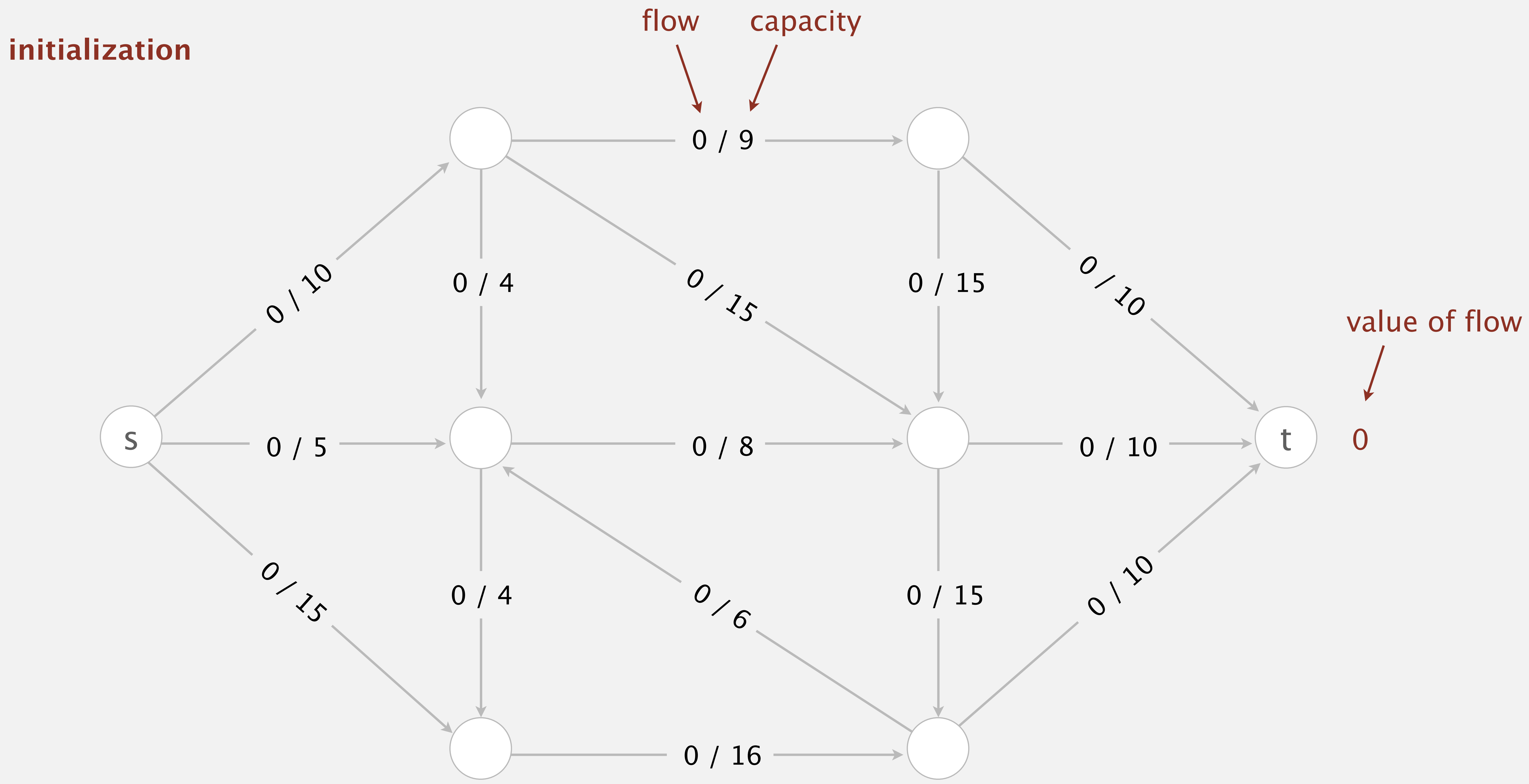
https://algs4.cs.princeton.edu

6.4 Maximum Flow

- introduction
- **Ford–Fulkerson algorithm**
- maxflow–mincut theorem
- analysis of running time
- Java implementation
- applications

Ford–Fulkerson algorithm demo

Initialization. Start with 0 flow.



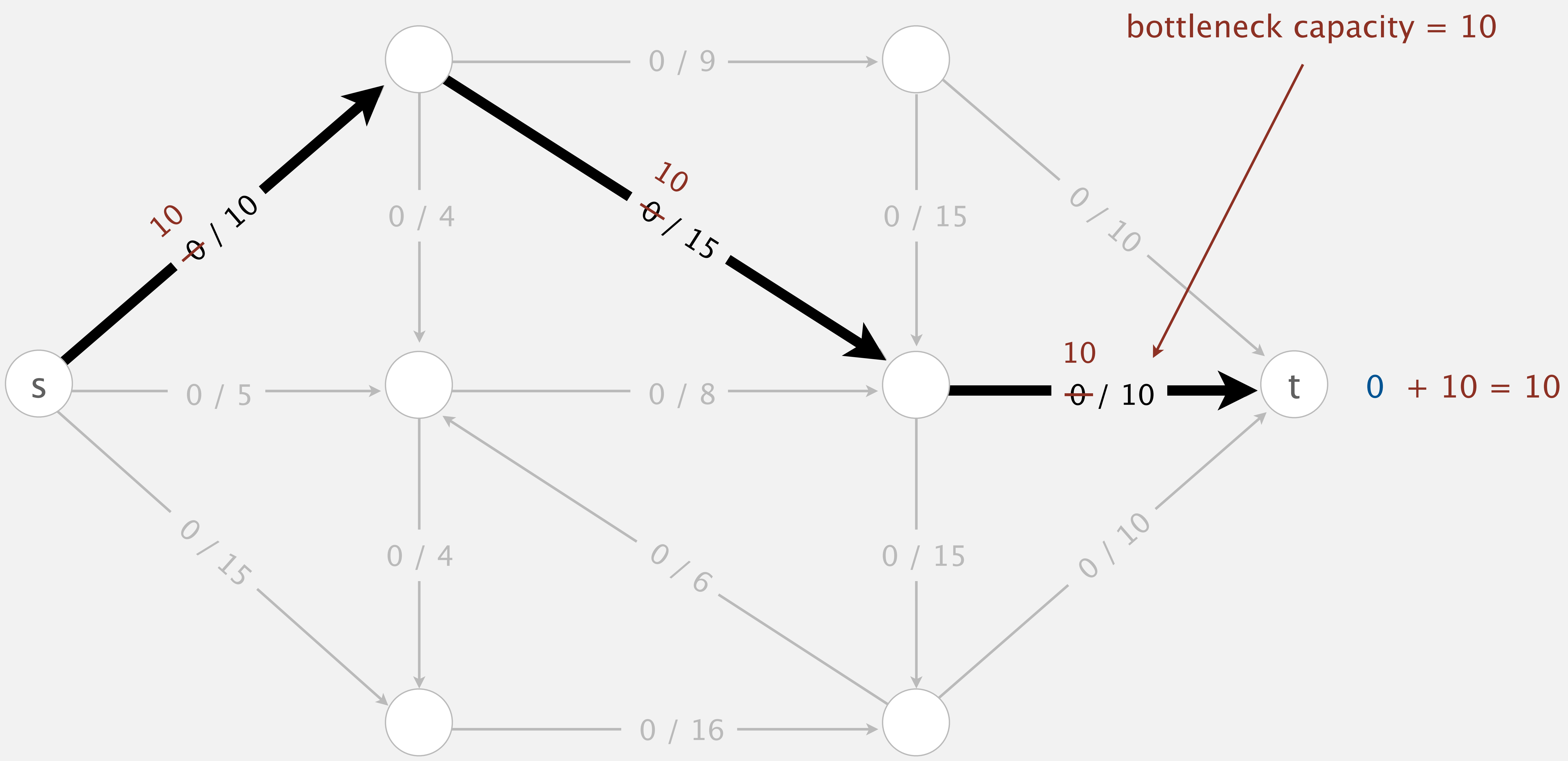
Ford–Fulkerson algorithm demo

Augmenting path. Find an undirected path from s to t such that:

- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).

impact: increases value of flow
(maintains capacity and flow conservation constraints)

1st augmenting path

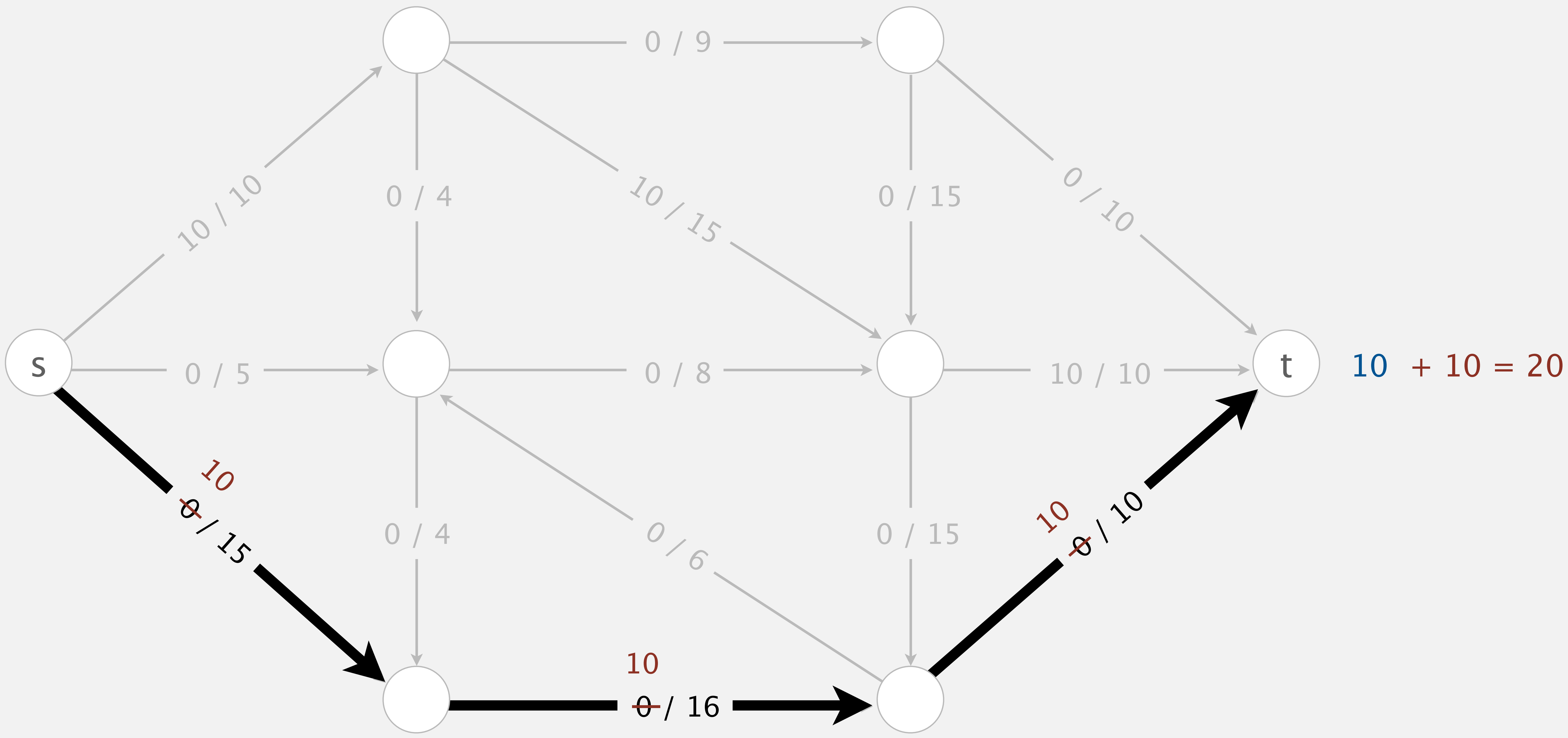


Ford–Fulkerson algorithm demo

Augmenting path. Find an undirected path from s to t such that:

- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).

2nd augmenting path



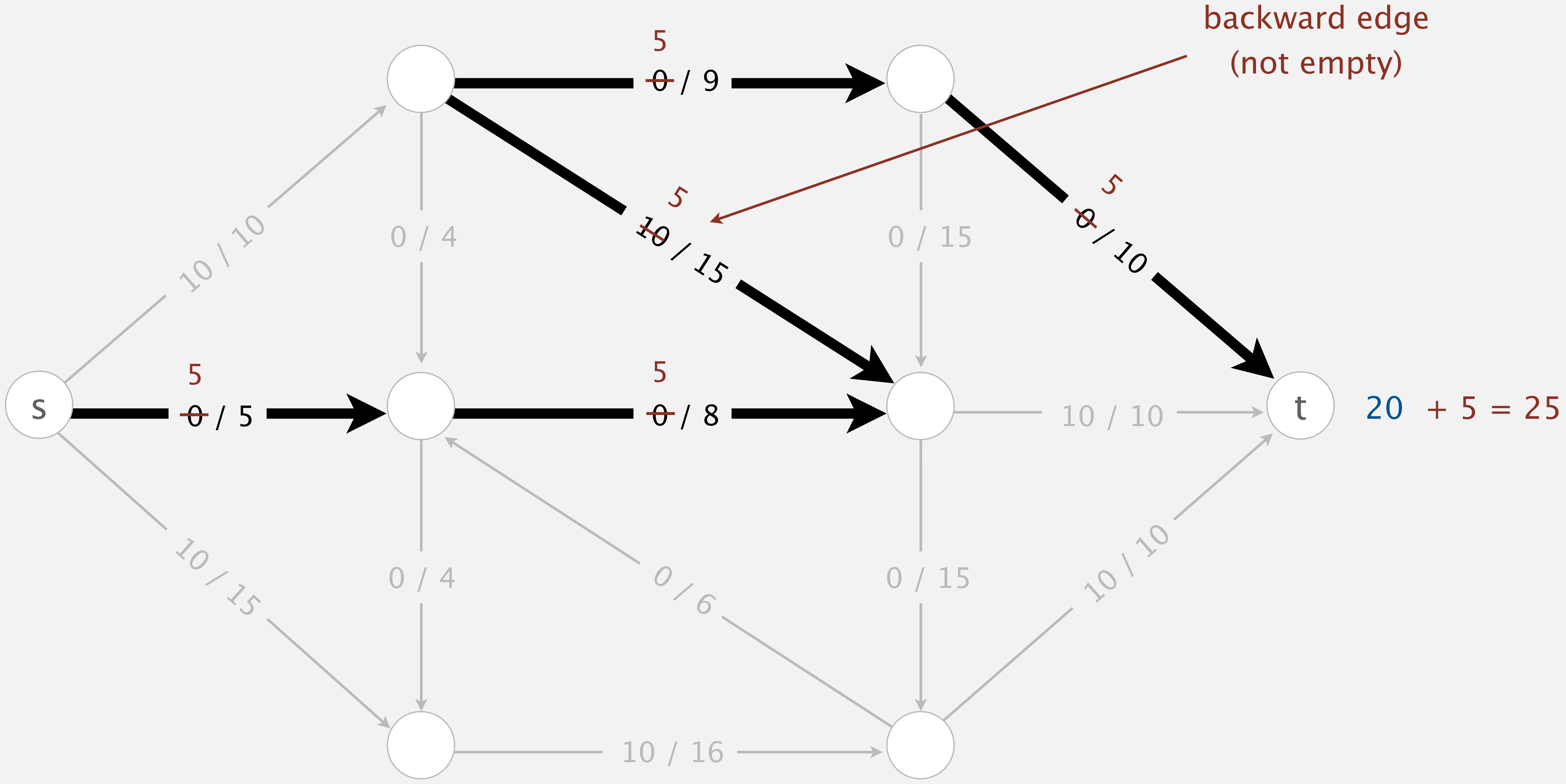
Ford–Fulkerson algorithm demo

Augmenting path. Find an undirected path from s to t such that:

- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).

impact: increases value of flow; maintains flow conservation

3rd augmenting path

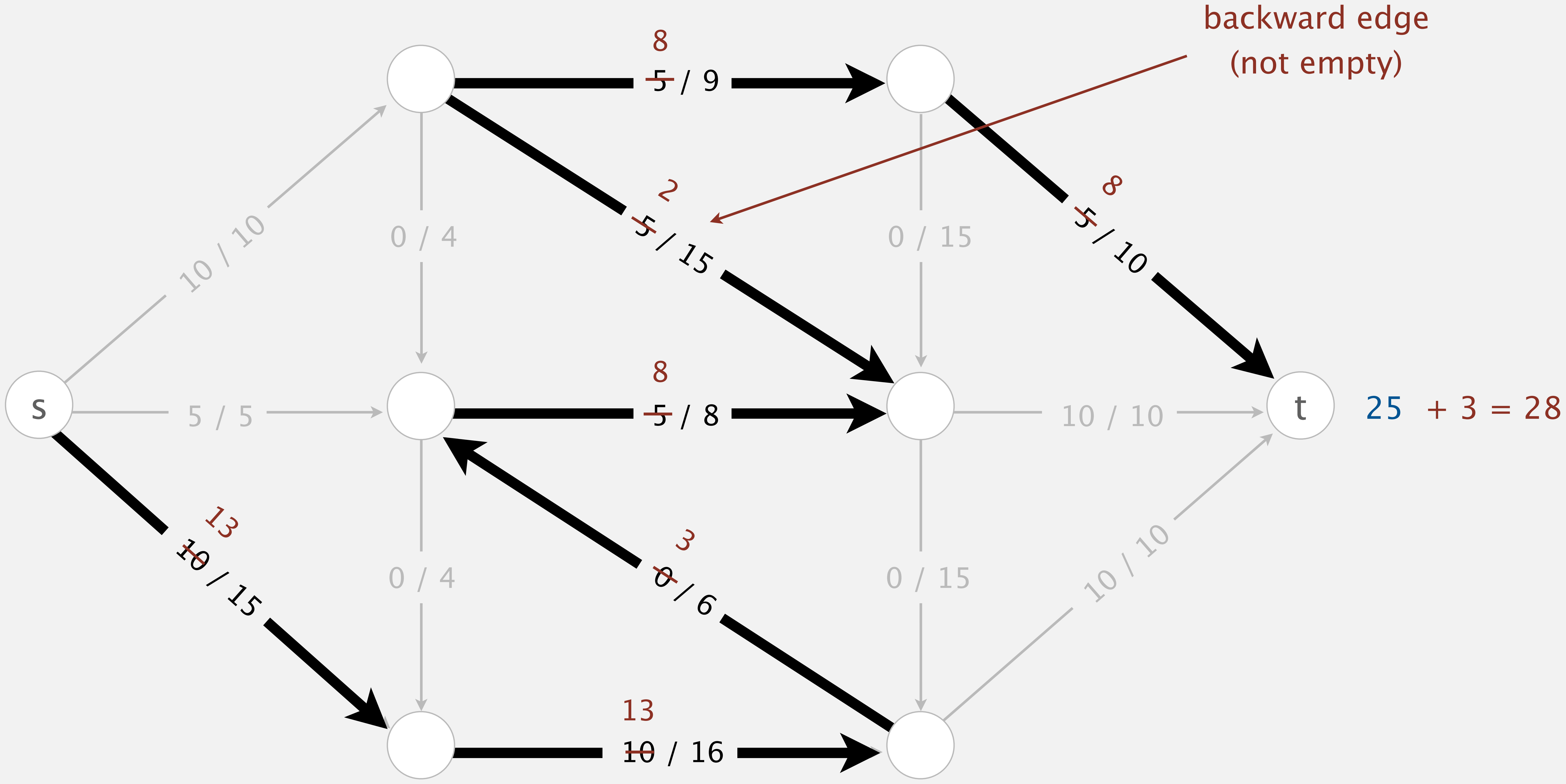


Ford–Fulkerson algorithm demo

Augmenting path. Find an undirected path from s to t such that:

- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).

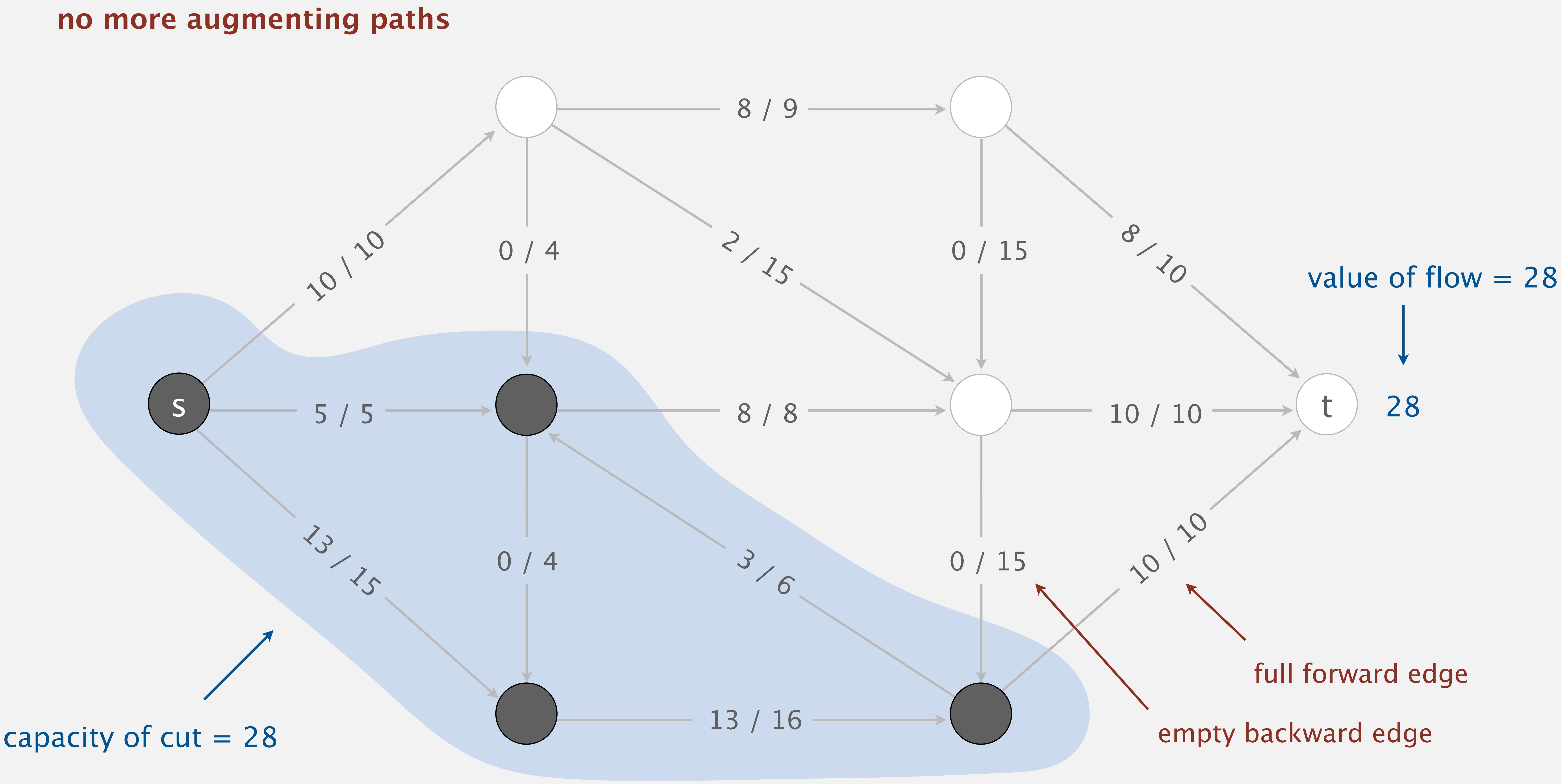
4th augmenting path



Ford–Fulkerson algorithm demo

Termination. All paths from s to t are blocked by either a

- Full forward edge.
- Empty backward edge.



Maxflow: quiz 2

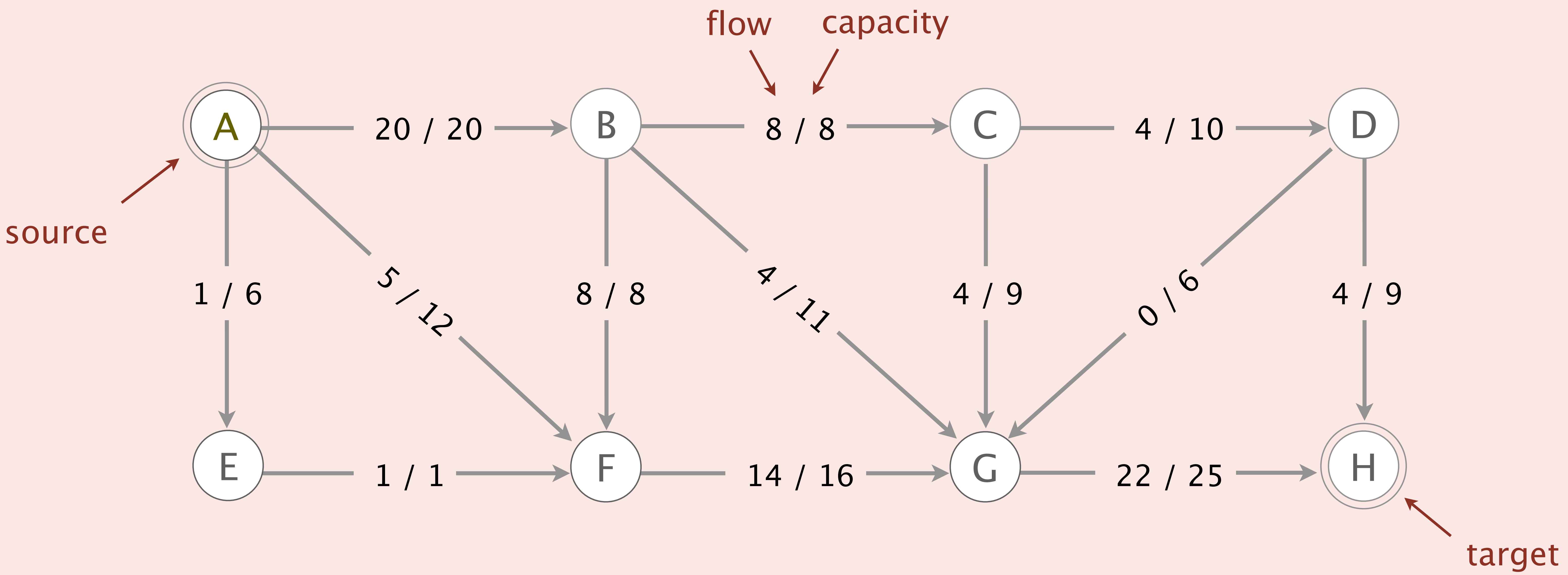
Which is an augmenting path?

A. $A \rightarrow F \rightarrow G \rightarrow D \rightarrow H$

B. $A \rightarrow F \rightarrow B \rightarrow G \rightarrow C \rightarrow D \rightarrow H$

C. Both A and B.

D. Neither A nor B.



Maxflow: quiz 3

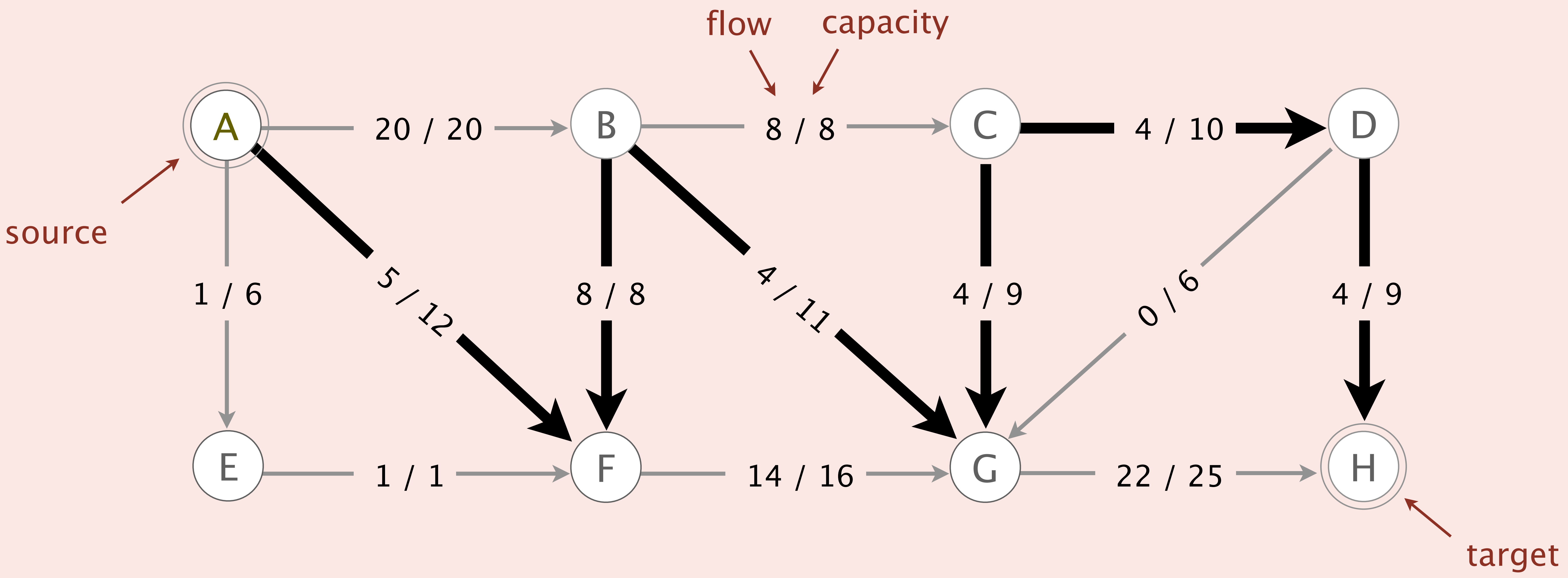
What is the bottleneck capacity of the augmenting path $A \to F \to B \to G \to C \to D \to H$?

A. 4

B. 5

C. 6

D. 7



Ford–Fulkerson algorithm

Ford–Fulkerson algorithm

Start with 0 flow.

While there exists an augmenting path:

- **find an augmenting path P**
- **compute bottleneck capacity of P**
- **update flow on P by bottleneck capacity**

Fundamental questions.

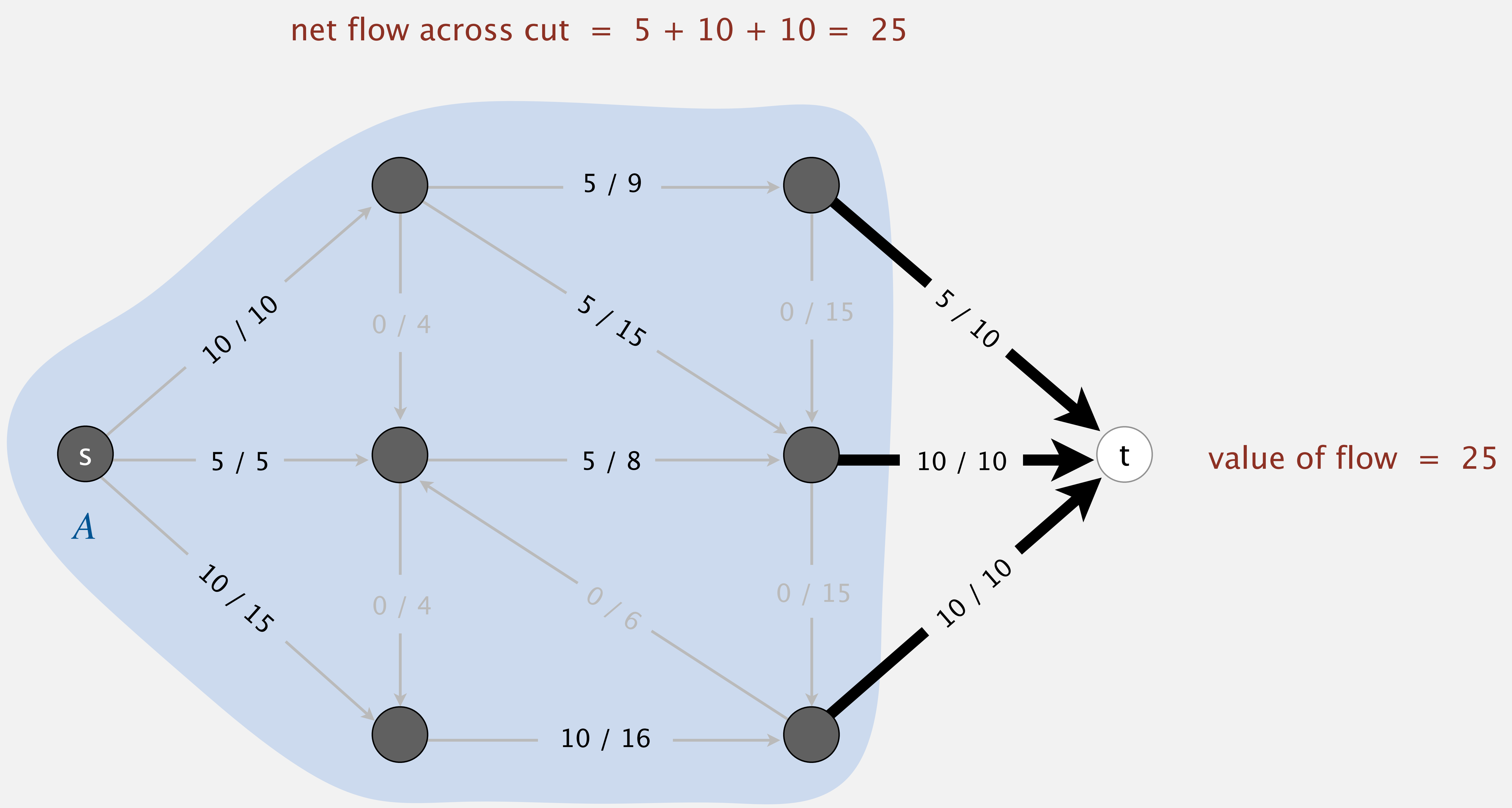
- How to find an augmenting path?
- How many augmenting paths?
- Guaranteed to compute a maxflow?
- Given a maxflow, how to compute a mincut?

https://algs4.cs.princeton.edu

6.4 Maximum Flow

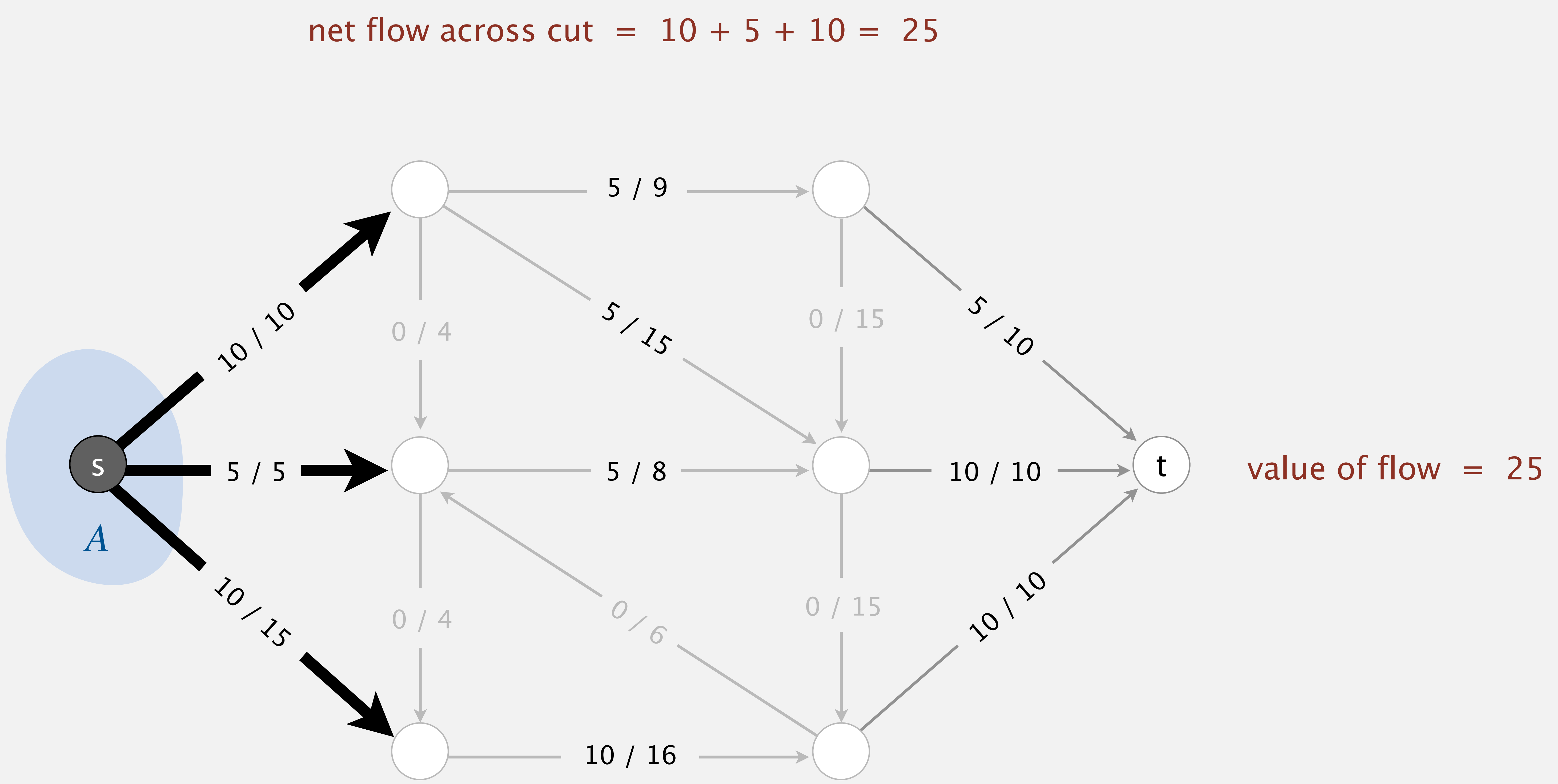
Relationship between flows and cuts

Def. Given a flow f, the net flow across a cut (A, B) is the sum of the flows on its edges from A to B minus the sum of the flows on its edges from B to A.



Relationship between flows and cuts

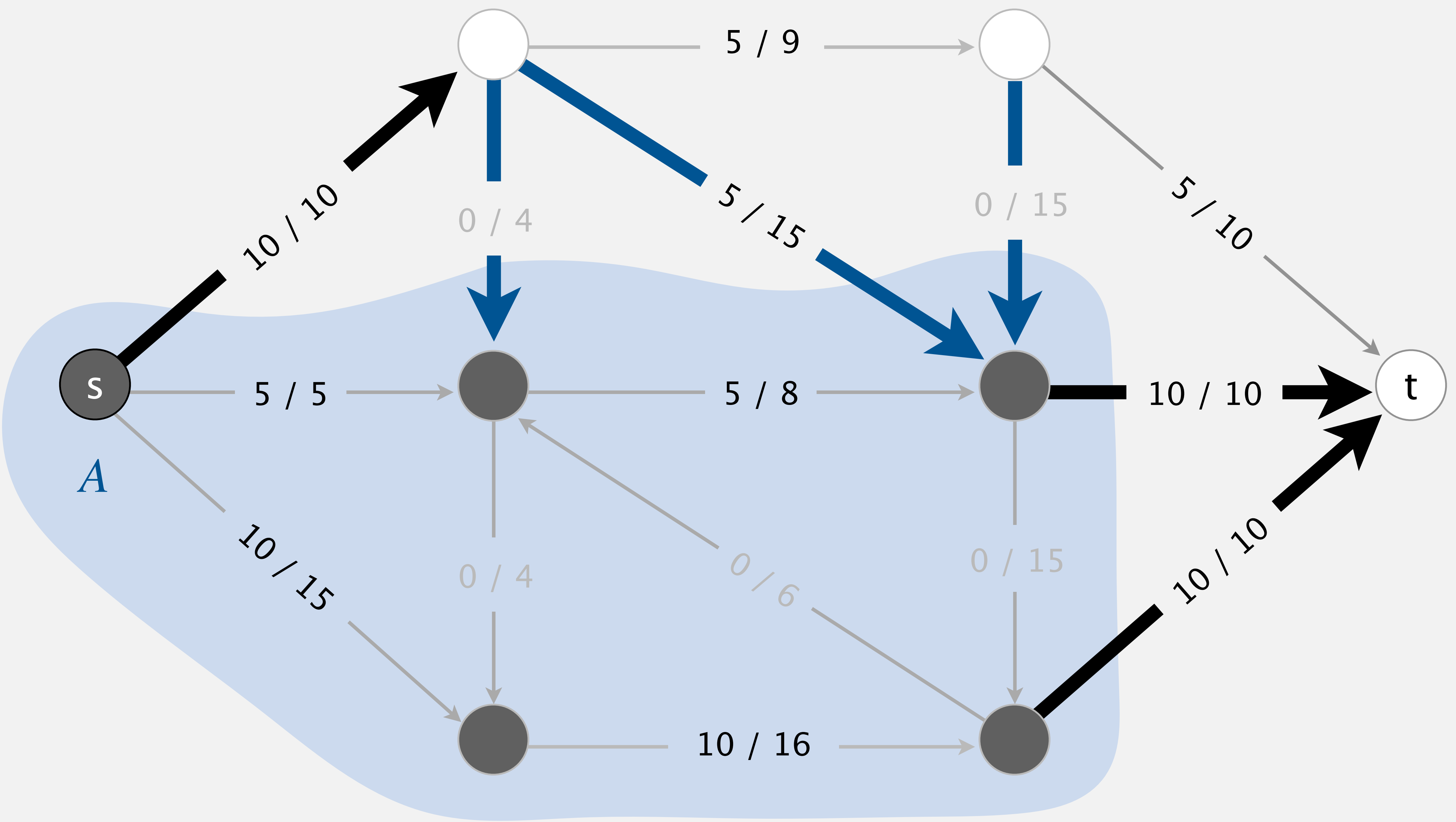
Def. Given a flow f, the net flow across a cut (A, B) is the sum of the flows on its edges from A to B minus the sum of the flows on its edges from B to A.



Relationship between flows and cuts

Def. Given a flow f, the net flow across a cut (A, B) is the sum of the flows on its edges from A to B minus the sum of the flows on its edges from B to A.

net flow across cut = (10 + 10 + 10) − (0 + 5 + 0) = 25



value of flow = 25

Maxflow: quiz 4

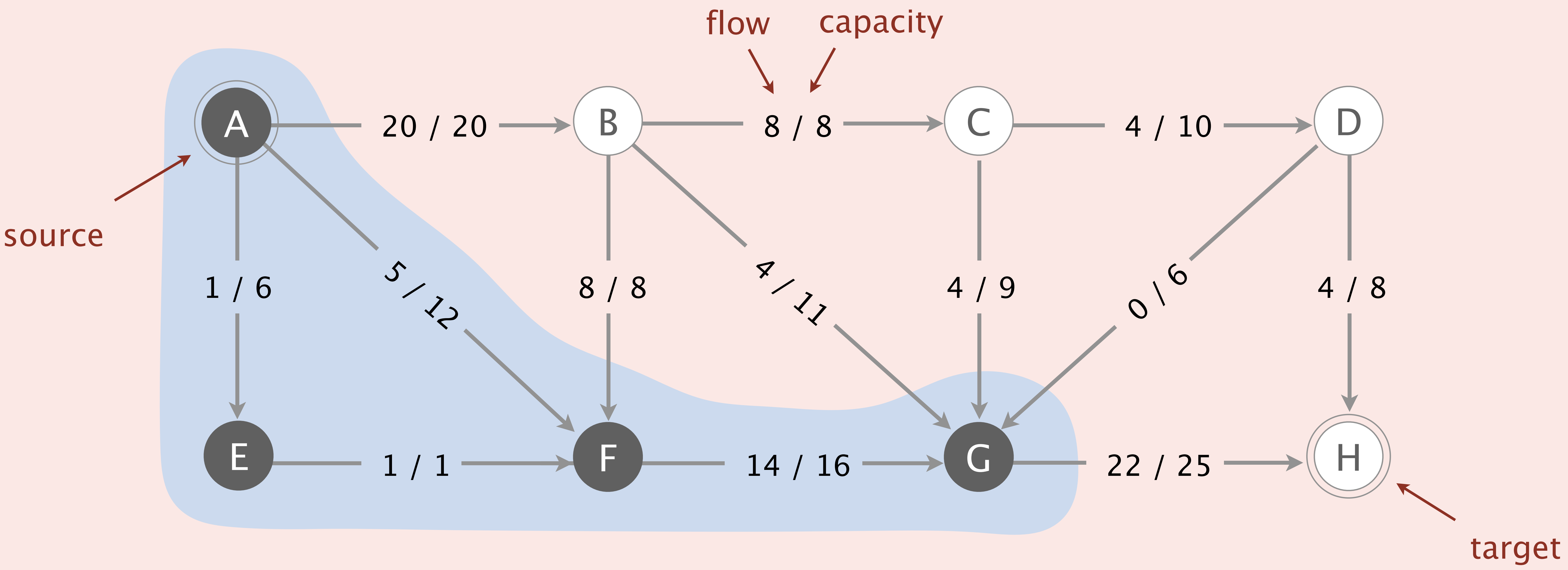
Given the flow f below, what is the net flow across the cut $\{A, E, F, G\}$?

A. 11 (20 + 25 – 8 – 11 – 9 – 6)

B. 26 (20 + 22 – 8 – 4 – 4 – 0)

C. 42 (20 + 22)

D. 45 (20 + 25)



Relationship between flows and cuts

Flow-value lemma. Let f be any flow and let (A, B) be any cut.
Then, the net flow across the cut (A, B) equals the value of the flow f.

Intuition. Conservation of flow.

Pf. By induction on the number of vertices in B.

- Base case: $B = \{ t \}$.
- Induction step: remains true when moving any vertex v from A to B
  (because of flow conservation constraint for vertex v)

Corollary. Outflow from s = inflow to t = value of flow.

we assume no edges incident to s or from t

Relationship between flows and cuts

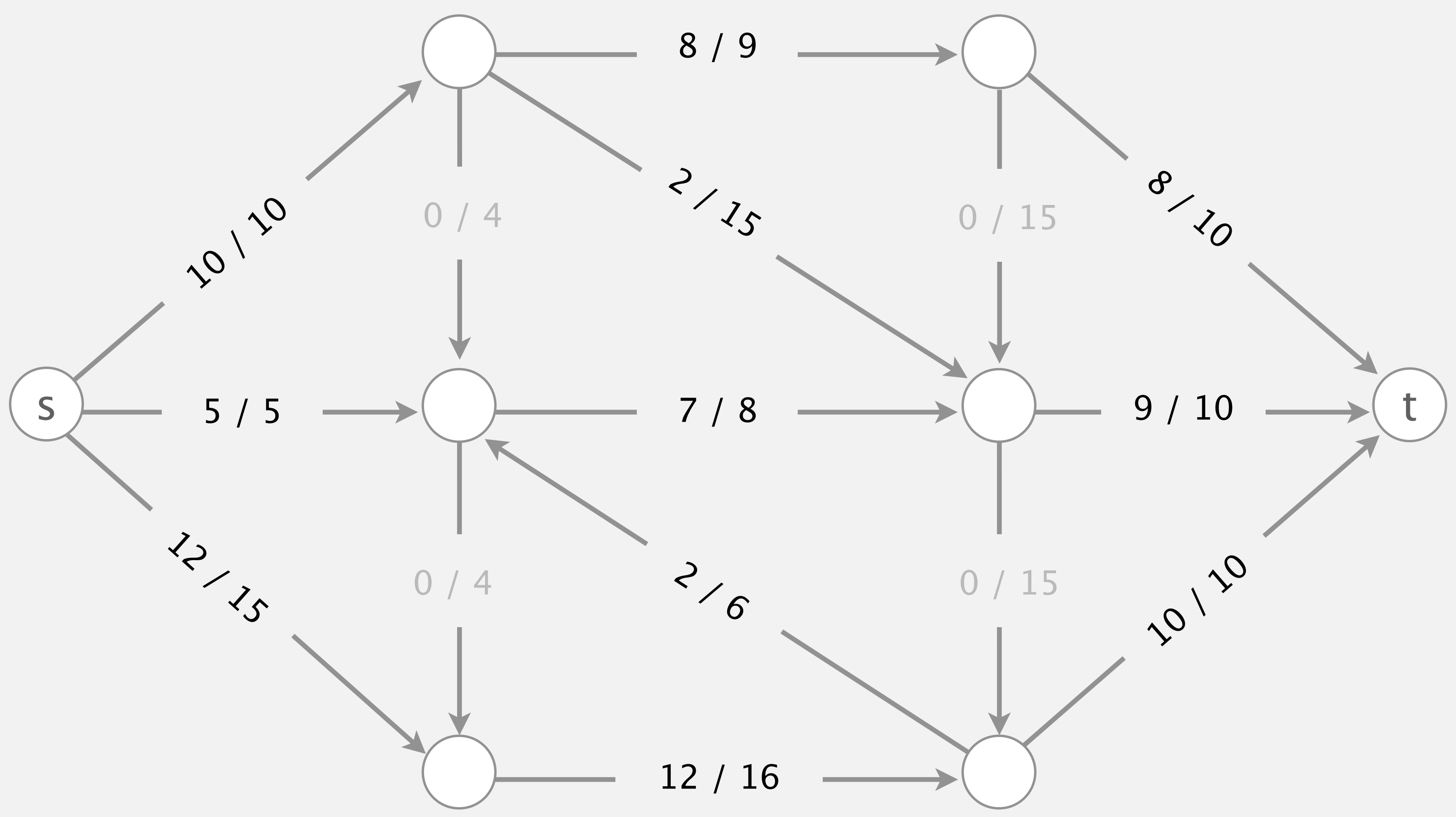
Weak duality. Let f be any flow and let (A, B) be any cut.
Then, the value of flow f $\leq$ the capacity of cut (A, B).

Pf. Value of flow f $=$ net flow across cut (A, B) $\leq$ capacity of cut (A, B).

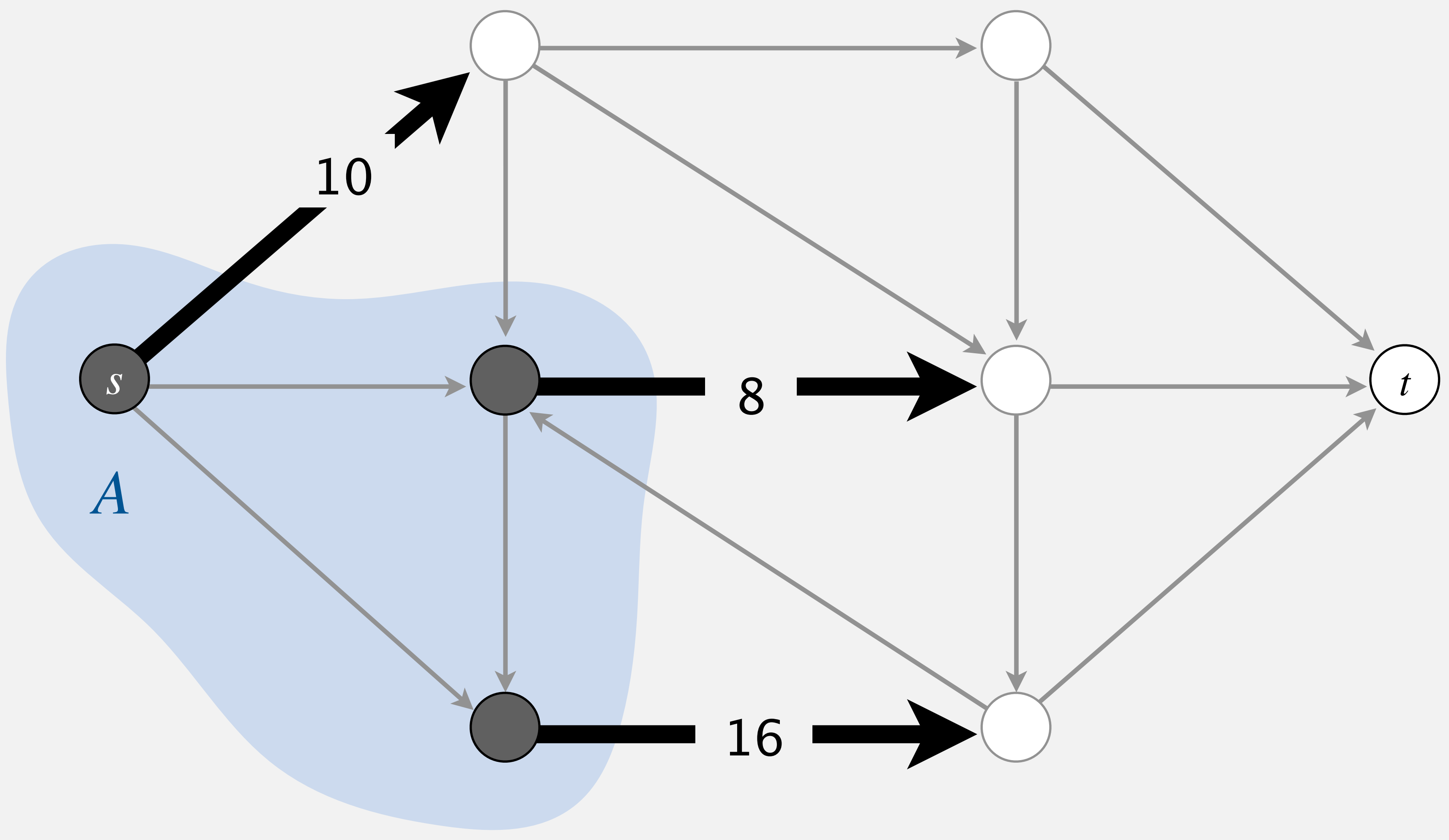
flow-value lemma

flow on each edge from A to B bounded by capacity

Equivalent. Value of maxflow $\leq$ capacity of mincut.



value of flow $f = 27$



capacity of cut $(A, B) = 34$

Maxflow–mincut theorem

Maxflow–mincut theorem. Value of the maxflow = capacity of mincut. ⟵ "strong duality"

Augmenting path theorem. A flow f is a maxflow if and only if no augmenting paths.

Pf. For any flow f, the following three conditions are equivalent:

i. Flow f is a maxflow.

ii. There is no augmenting path with respect to flow f.

iii. There exists a cut whose capacity equals the value of flow f.

[i ⇒ ii] We prove contrapositive: ~ii ⇒ ~i.

- Suppose that there is an augmenting path with respect to flow f.
- Can improve f by sending flow along this path.
- Thus, f is not a maxflow. ▪

Maxflow–mincut theorem

Maxflow–mincut theorem. Value of the maxflow = capacity of mincut.

Augmenting path theorem. A flow f is a maxflow if and only if no augmenting paths.

Pf. For any flow f, the following three conditions are equivalent:

i. Flow f is a maxflow.

ii. There is no augmenting path with respect to flow f.

iii. There exists a cut whose capacity equals the value of flow f.

[iii ⇒ i]

- Let (A, B) be a cut whose capacity equals the value of flow f.
- Then, the value of any flow f' $\underset{\text{weak duality}}{\leq}$ capacity of (A, B) $\underset{\text{by assumption}}{=}$ value of f.
- Thus, f is a maxflow. ▪

Maxflow-mincut theorem

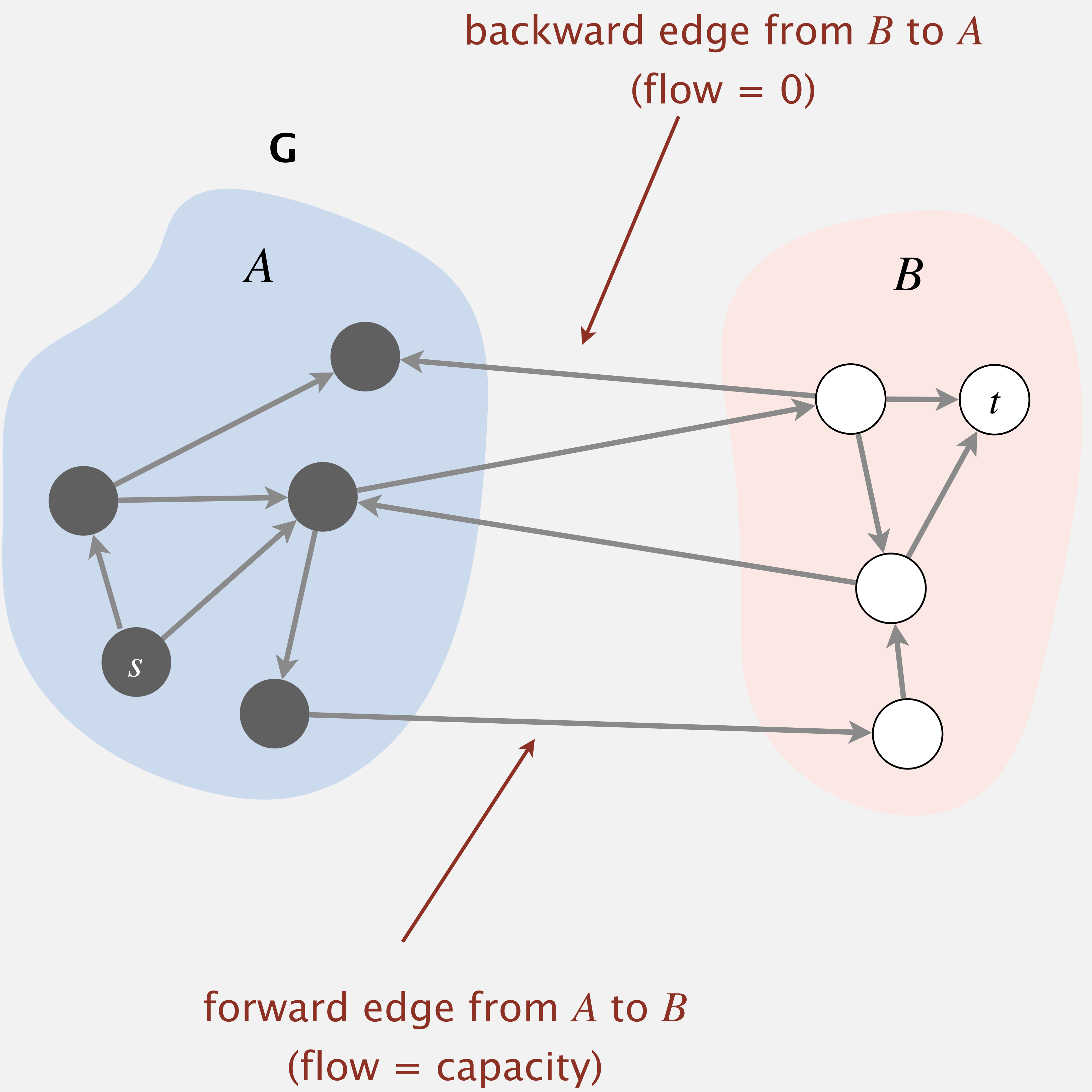
[ii ⇒ iii]

- Let f be a flow with no augmenting paths.
- Let A be set of vertices reachable from s via a path with no full forward or empty backward edges.
- By definition of cut (A, B), s is in A.
- By definition of cut (A, B) and flow f, t is in B.
- Capacity of cut (A, B) = net flow across cut

  = value of flow f. ▪

by construction of cut

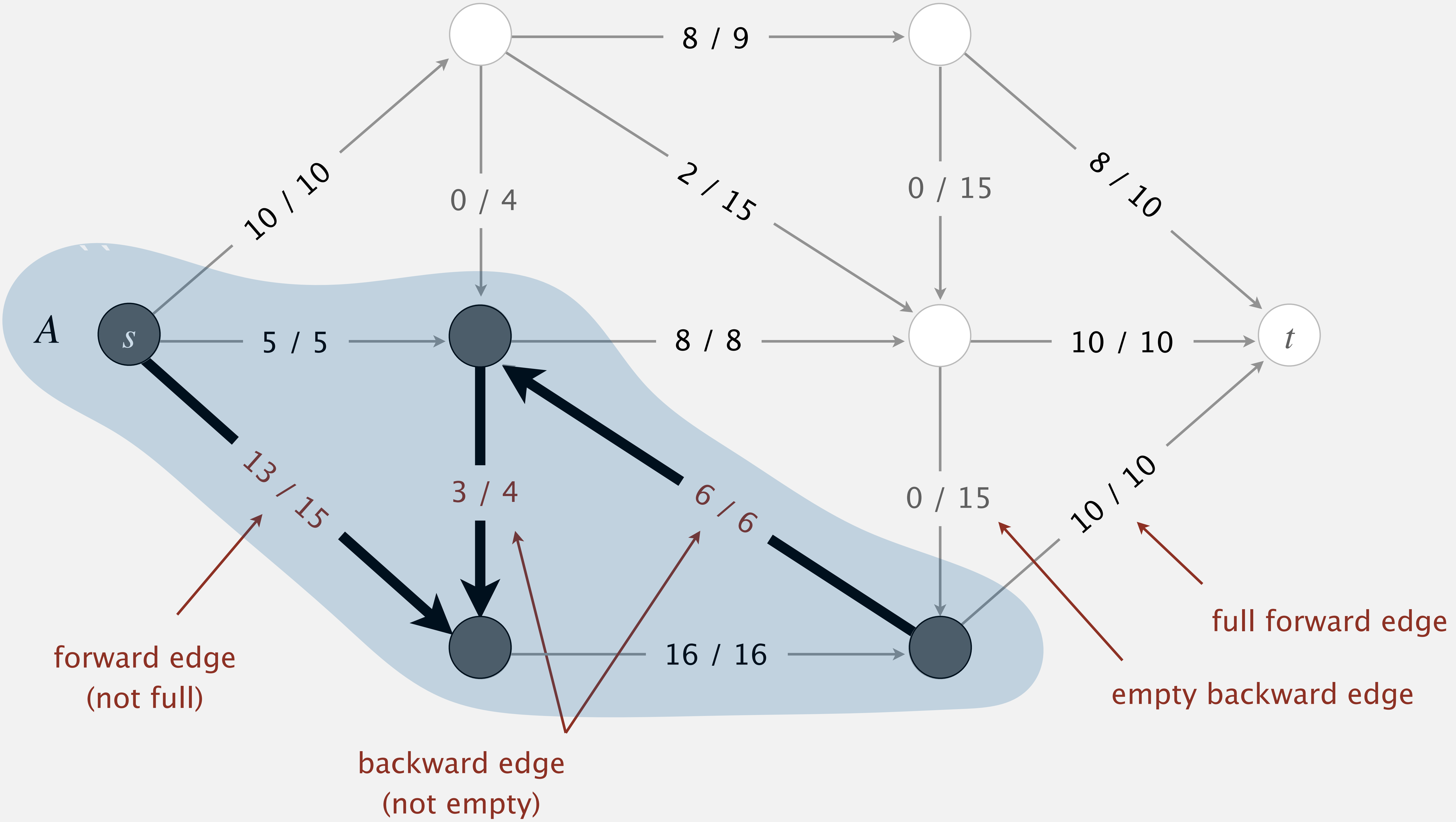
flow-value lemma



Computing a mincut from a maxflow

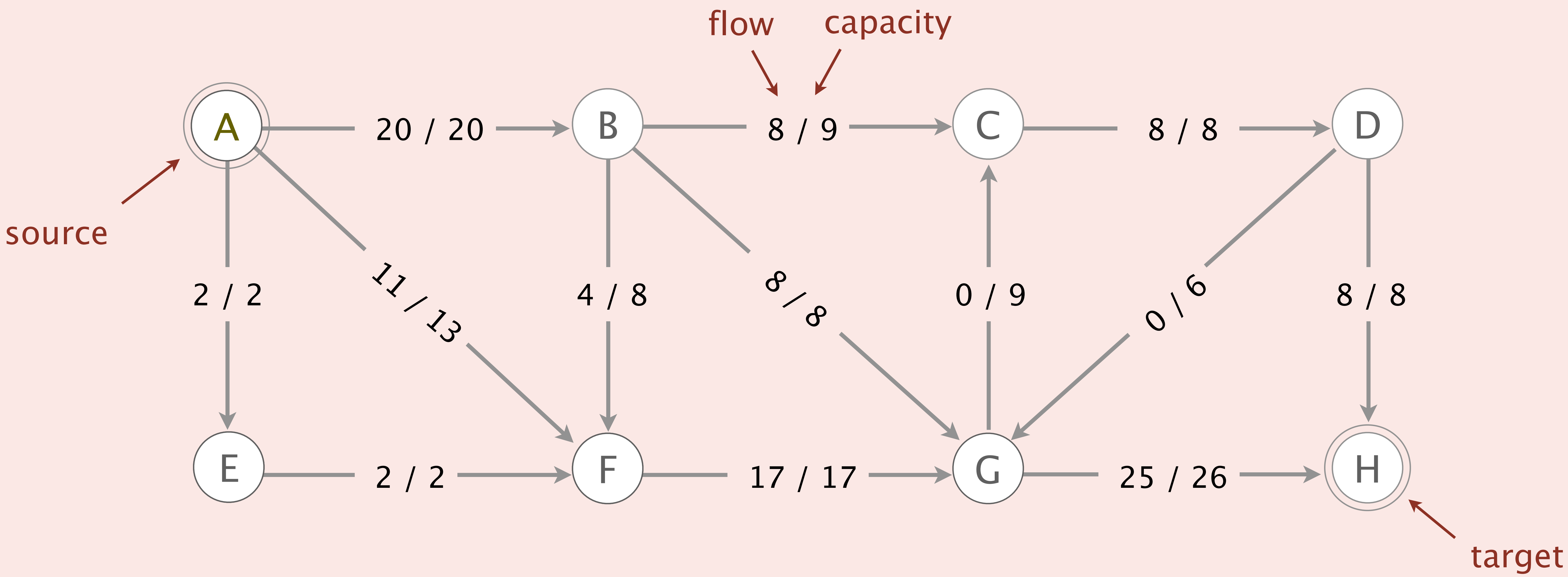
To compute mincut (A, B) from maxflow f :

- By augmenting path theorem, no augmenting paths with respect to f.
- Compute A = set of vertices connected to s by an undirected path with no full forward or empty backward edges.
- Capacity of cut (A, B) = value of flow $f \Rightarrow$ mincut.



Given the following maxflow, which is a mincut?

- **A.** $A = \{ A, F \}$.
- **B.** $A = \{ A, B, C, F \}$.
- **C.** $A = \{ A, B, C, E, F \}$.
- **D.** None of the above.



https://algs4.cs.princeton.edu

6.4 Maximum Flow

Ford–Fulkerson algorithm analysis (with integer capacities)

Important special case. Edge capacities are integers between 1 and U.

flow on each edge is an integer

Invariant. The flow is integral throughout Ford–Fulkerson.

Pf.

- Bottleneck capacity is an integer.
- Flow on an edge increases/decreases by bottleneck capacity. ▪

Proposition. Number of augmentations $\leq$ the value of the maxflow.

Pf. Each augmentation increases the value of the flow by at least one. ▪

critical for some applications (stay tuned)

Integrality theorem. There exists an integral maxflow.

Pf.

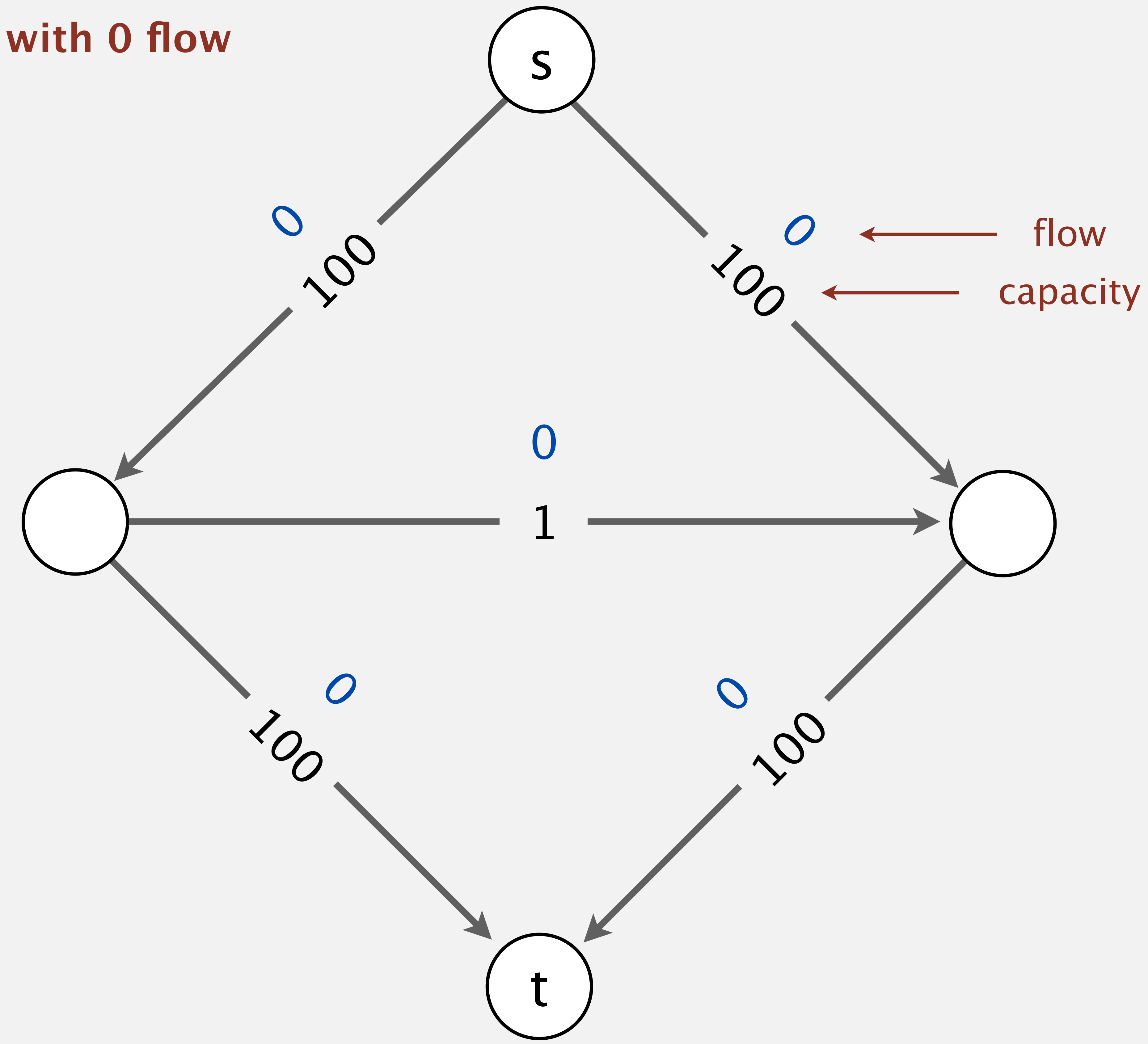
- Proposition + Augmenting path theorem $\Rightarrow$ Ford–Fulkerson terminates with a maxflow.
- Invariant $\Rightarrow$ That maxflow is integral. ▪

Bad case for Ford–Fulkerson

Bad news. Number of augmenting paths can be very large.

even when capacities are integral

initialize with 0 flow



How to choose augmenting paths?

Bad news. Some choices lead to exponential-time algorithms.

Good news. Clever choices lead to polynomial-time algorithms. ← polynomial in input size ($V, E, \log U$)

augmenting path	number of paths	implementation
DFS path	$\leq E\ U$	*stack*
random path	$\leq E\ U$	*randomized queue*
shortest path (fewest edges)	$\leq \frac{1}{2}\ E\ V$	*queue*
fattest path (max bottleneck capacity)	$\leq E \ln(E\ U)$	*priority queue*

flow network with V vertices, E edges, and integer capacities between 1 and U

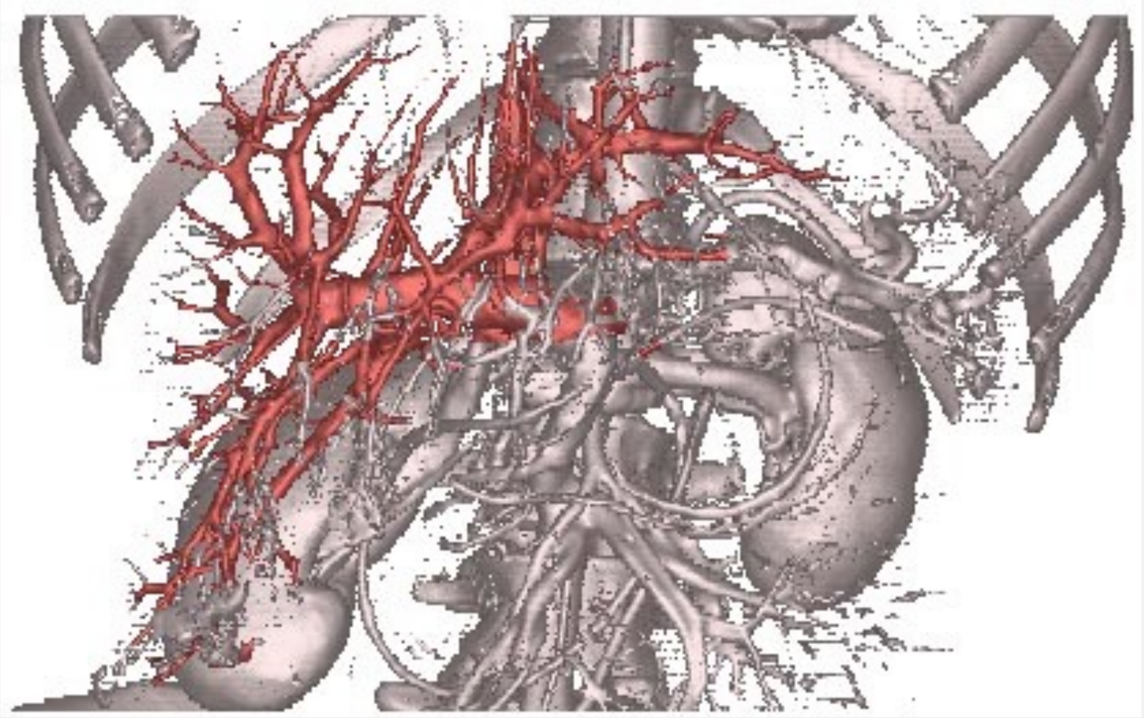
https://algs4.cs.princeton.edu

6.4 Maximum Flow

Maxflow and mincut applications

Maxflow/mincut is a widely applicable problem-solving model.

- Data mining.
- Open-pit mining.
- Bipartite matching.
- Network reliability.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
- Distributed computing.
- Security of statistical data.
- Egalitarian stable matching.
- Multi-camera scene reconstruction.
- Sensor placement for homeland security.
- Many, many, more.



liver and hepatic vascularization segmentation

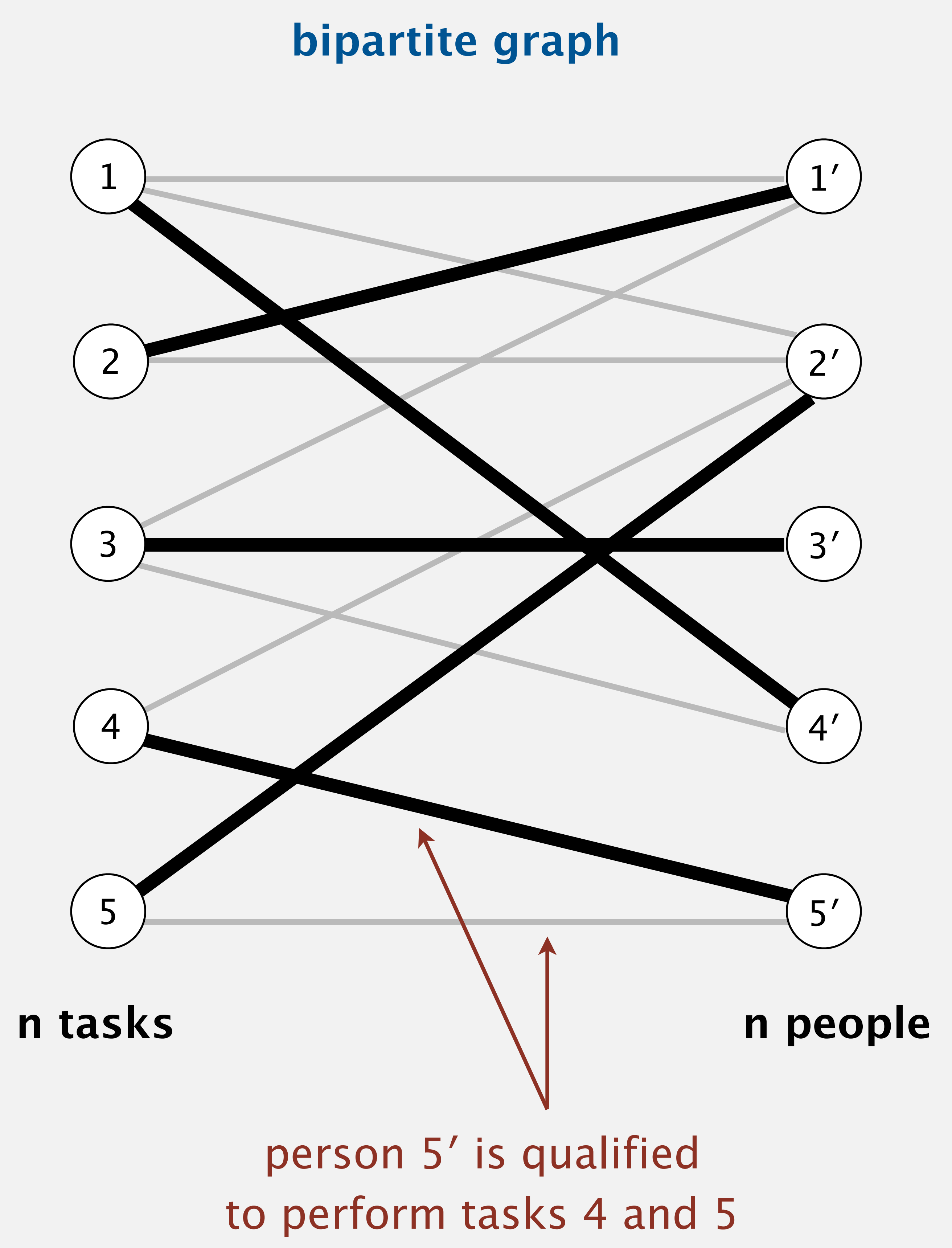
Bipartite matching problem

Problem. Given n people and n tasks, assign the tasks to people so that:

- Every task is assigned to a qualified person.
- Every person is assigned to exactly one task.

Bipartite matching problem

Problem. Given a bipartite graph, find a perfect matching (if one exists).

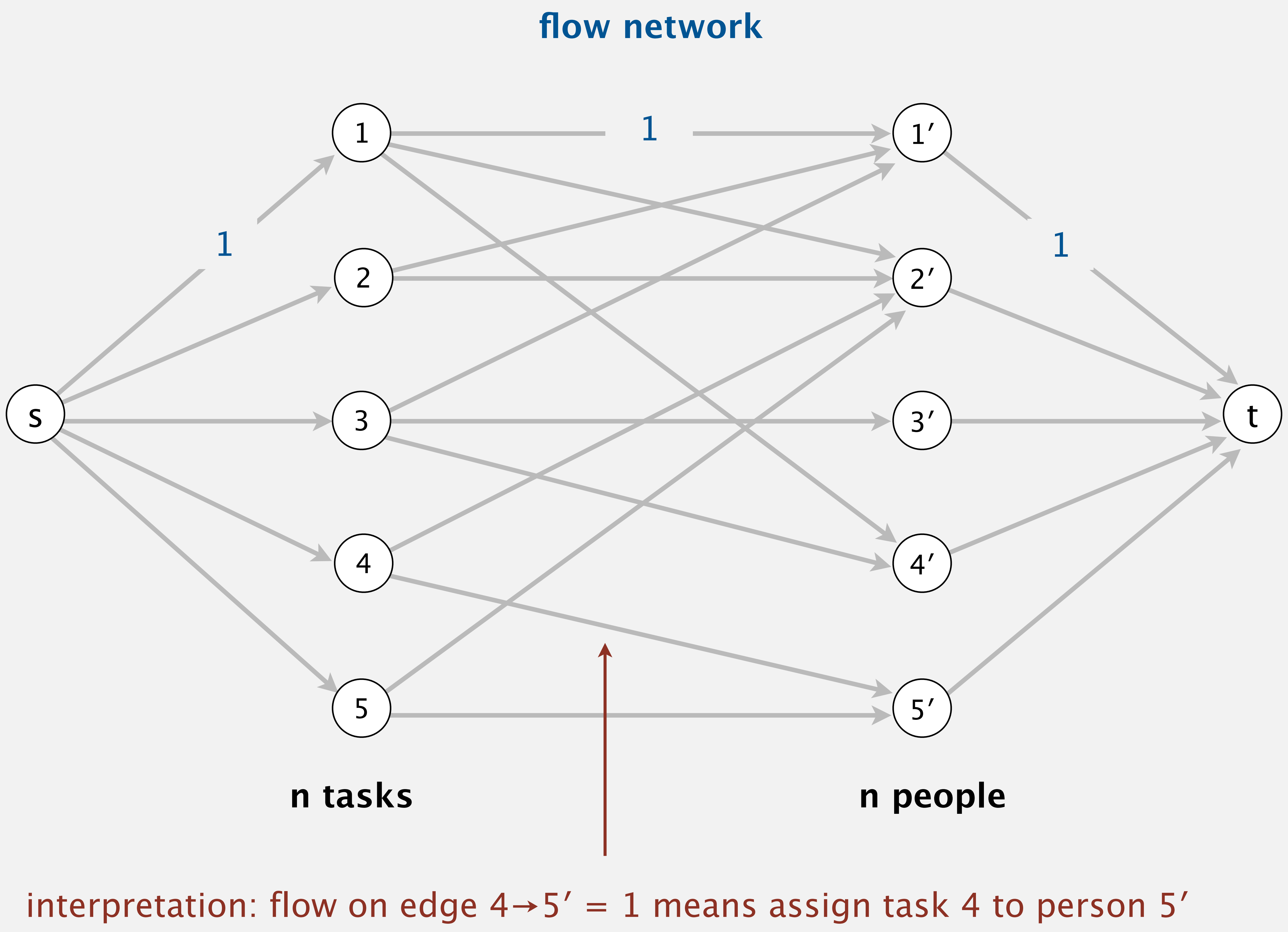


perfect matching

1–4′
2–1′
3–3′
4–5′
5–2′

Maxflow formulation of bipartite matching

- Create source s, target t, one vertex i for each task, and one vertex j' for each person.
- Add edge from s to each task i (of capacity 1).
- Add edge from each person j' to t (of capacity 1).
- Add edge from task i to qualified person j' (of capacity 1 or ∞).

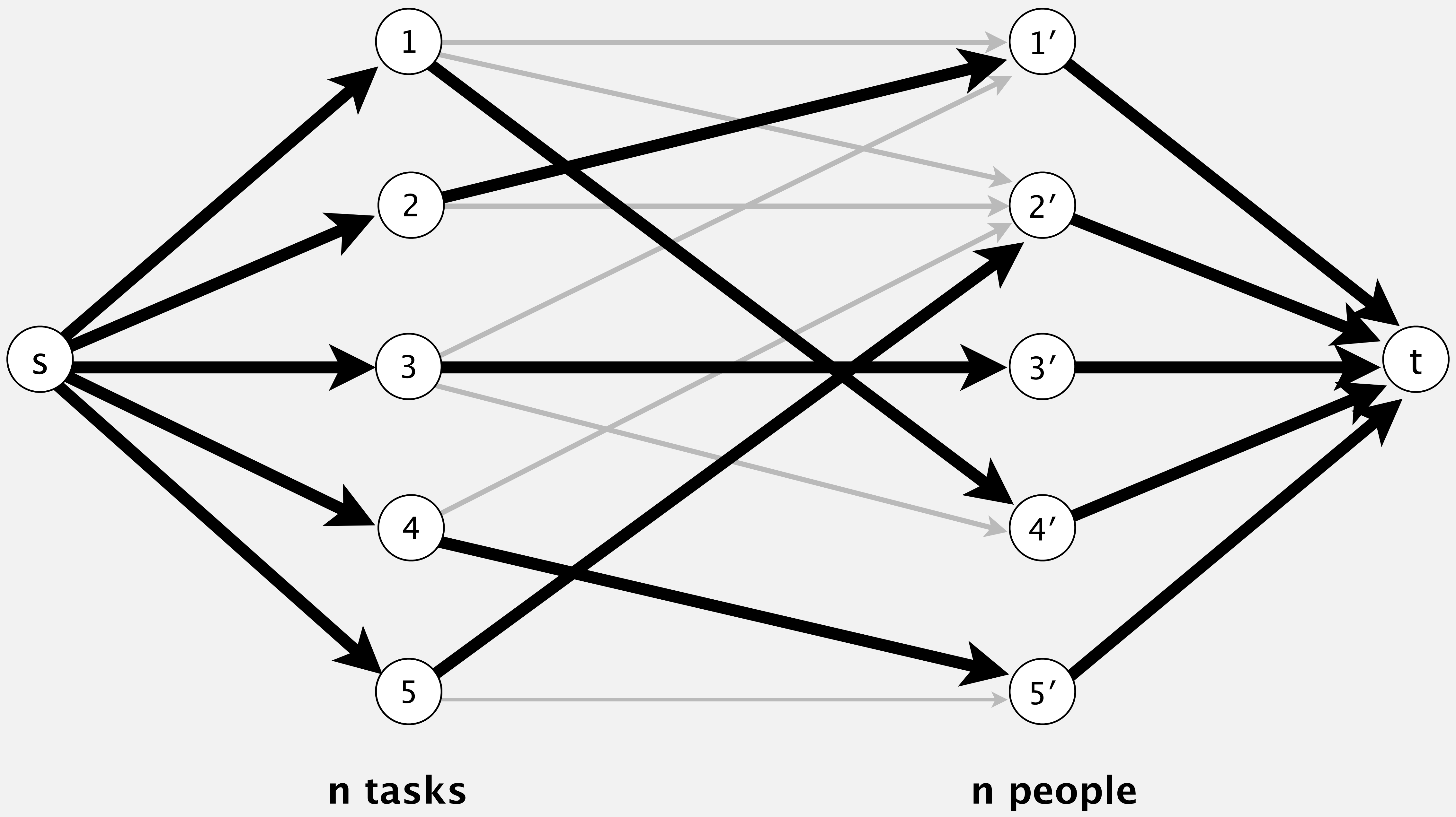


Maxflow formulation of bipartite matching

1-1 correspondence between perfect matchings in bipartite graph and integral flows of value n in flow network.

Integrality theorem + 1-1 correspondence ⇒ Maxflow formulation is correct.

flow network



In the worst case, how many augmenting paths does the Ford–Fulkerson algorithm consider in order to find a perfect matching in a bipartite graph with n vertices per side?

A. $\Theta(n)$

B. $\Theta(n^2)$

C. $\Theta(n^3)$

D. $\Theta(n^4)$

Maximum flow algorithms: theory highlights

year	method	worst case	discovered by
1955	**augmenting paths**	$O(E\ V\ U)$	*Ford–Fulkerson*
1970	**shortest augmenting paths**	$O(E\ V^{\,2})$	*Edmonds–Karp, Dinitz*
1974	**blocking flows**	$O(V^{\,3})$	*Karzanov*
1983	**dynamic trees**	$O(E\ V \log V)$	*Sleator–Tarjan*
1988	**push-relabel**	$O(E\ V \log (V^{\,2} / E))$	*Goldberg–Tarjan*
1998	**binary blocking flows**	$O(E^{3/2} \log (V^{\,2} / E) \log U)$	*Goldberg–Rao*
2013	**compact networks**	$O(E\ V)$	*Orlin*
2014	**interior-point methods**	$\tilde{O}(E\ V^{\,1/2} \log U)$	*Lee–Sidford*
2016	**electrical flows**	$\tilde{O}(E^{10/7}\ U^{1/7})$	*Mądry*
2022	**min ratio cycles**	$O(E^{1+\varepsilon} \log^2 U)$	*CKLPGS*
20xx		???	

max-flow algorithms with E edges, V vertices, and integer capacities between 1 and U

Maximum flow algorithms: practice

Warning. Worst-case order-of-growth is generally not useful for predicting or comparing maxflow algorithm performance in practice.

Often best in practice. Push-relabel method with gap relabeling.

Computer vision. Specialized algorithms for problems with special structure.

On Implementing Push-Relabel Method for the Maximum Flow Problem

Boris V. Cherkassky[1] and Andrew V. Goldberg[2]

[1] Central Institute for Economics and Mathematics, Krasikova St. 32, 117418, Moscow, Russia
cher@cemi.msk.su
[2] Computer Science Department, Stanford University Stanford, CA 94305, USA
goldberg@cs.stanford.edu

Abstract. We study efficient implementations of the push-relabel method for the maximum flow problem. The resulting codes are faster than the previous codes, and much faster on some problem families. The speedup is due to the combination of heuristics used in our implementations. We also exhibit a family of problems for which the running time of all known methods seem to have a roughly quadratic growth rate.

ELSEVIER

European Journal of Operational Research 97 (1997) 509–542

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

Computational investigations of maximum flow algorithms

Ravindra K. Ahuja [a], Murali Kodialam [b], Ajay K. Mishra [c], James B. Orlin [d,*]

[a] *Department of Industrial and Management Engineering, Indian Institute of Technology, Kanpur, 208 016, India*
[b] *AT&T Bell Laboratories, Holmdel, NJ 07733, USA*
[c] *KATZ Graduate School of Business, University of Pittsburgh, Pittsburgh, PA 15260, USA*
[d] *Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*

Received 30 August 1995; accepted 27 June 1996

Summary

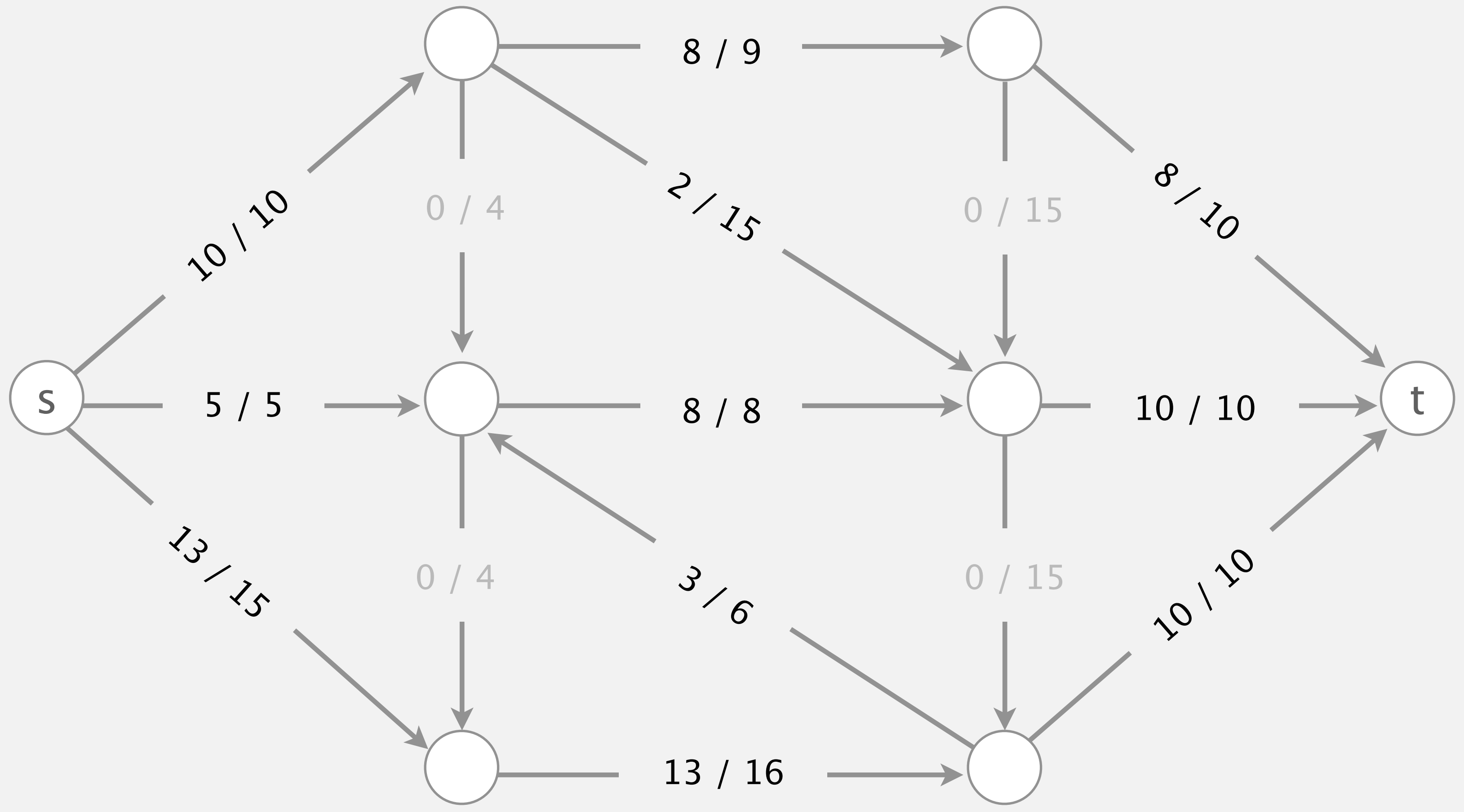
Mincut problem. Find a cut of minimum capacity.

Maxflow problem. Find a flow of maximum value.

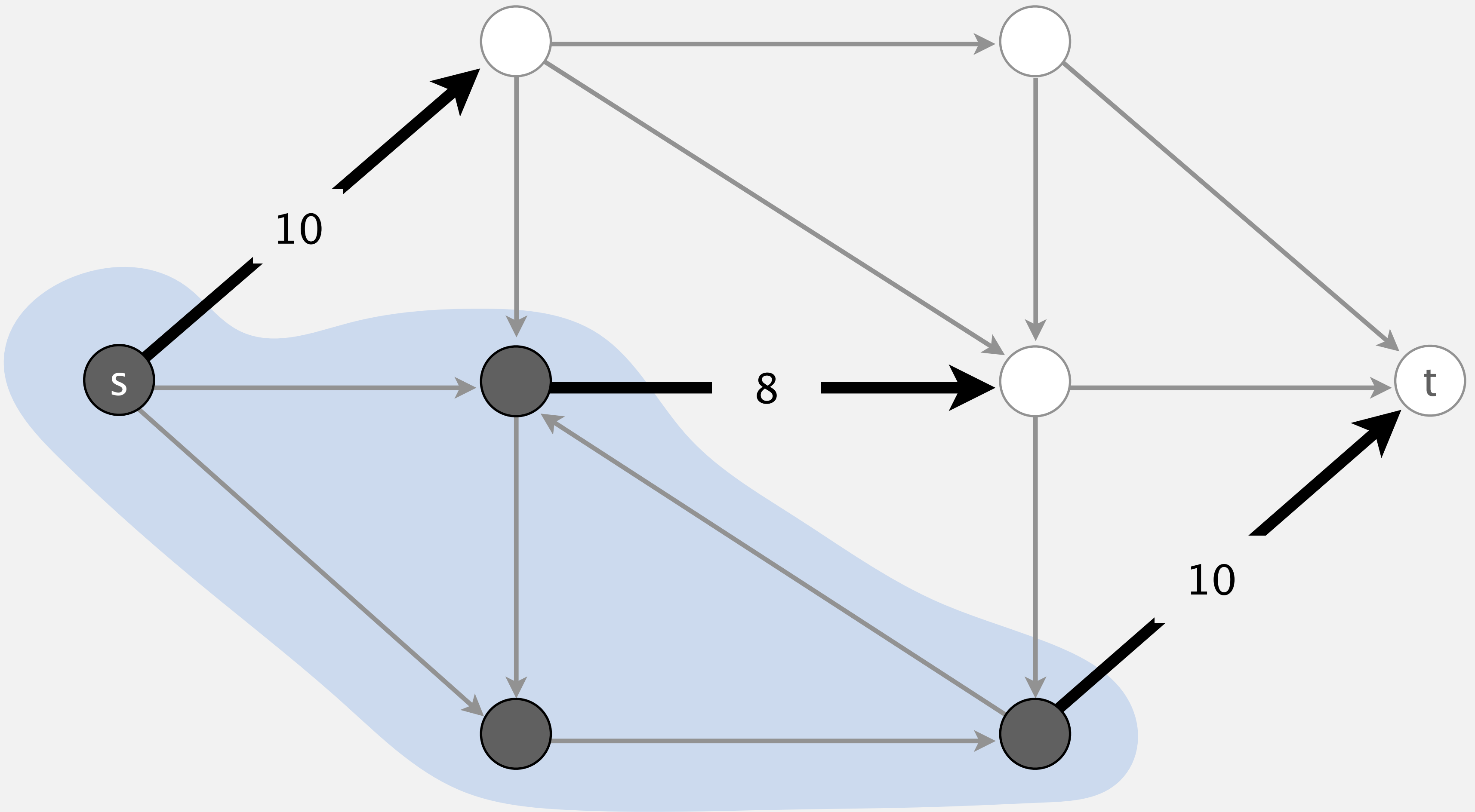
Duality. Value of the maxflow = capacity of mincut.

Proven successful approaches.

- Ford-Fulkerson (various augmenting-path strategies).
- Preflow-push (various versions).



value of flow = 28



capacity of cut = 28

© Copyright 2023 Robert Sedgewick and Kevin Wayne