https://algs4.cs.princeton.edu

6.4 MAXIMUM FLOW

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

https://algs4.cs.princeton.edu

6.4 MAXIMUM FLOW

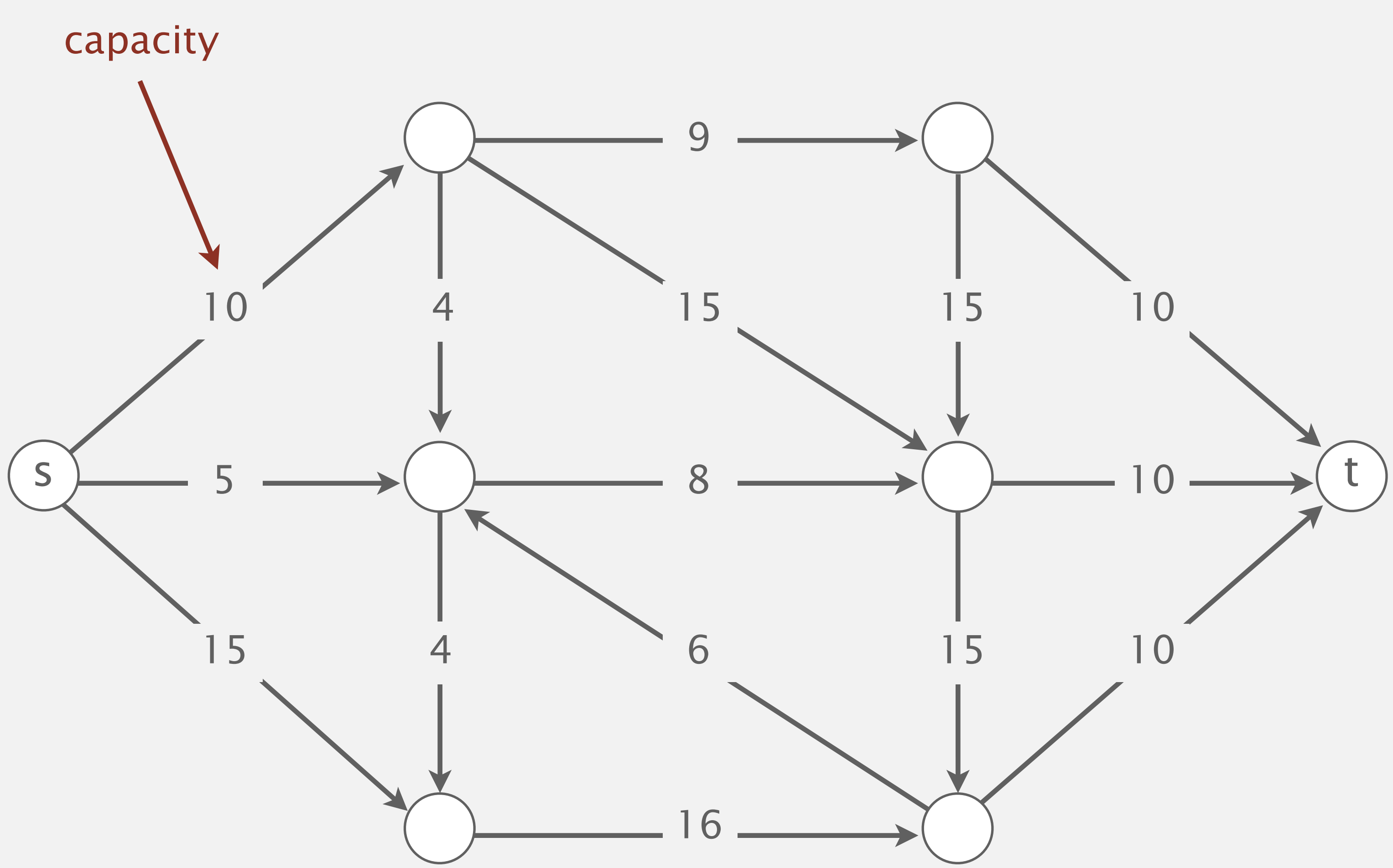
- *introduction*
- *Ford–Fulkerson algorithm*
- *maxflow–mincut theorem*
- *analysis of running time*
- *Java implementation*
- *applications*

Network flow

Input. An edge-weighted digraph, source vertex s, and target vertex t.

each edge has a
positive capacity

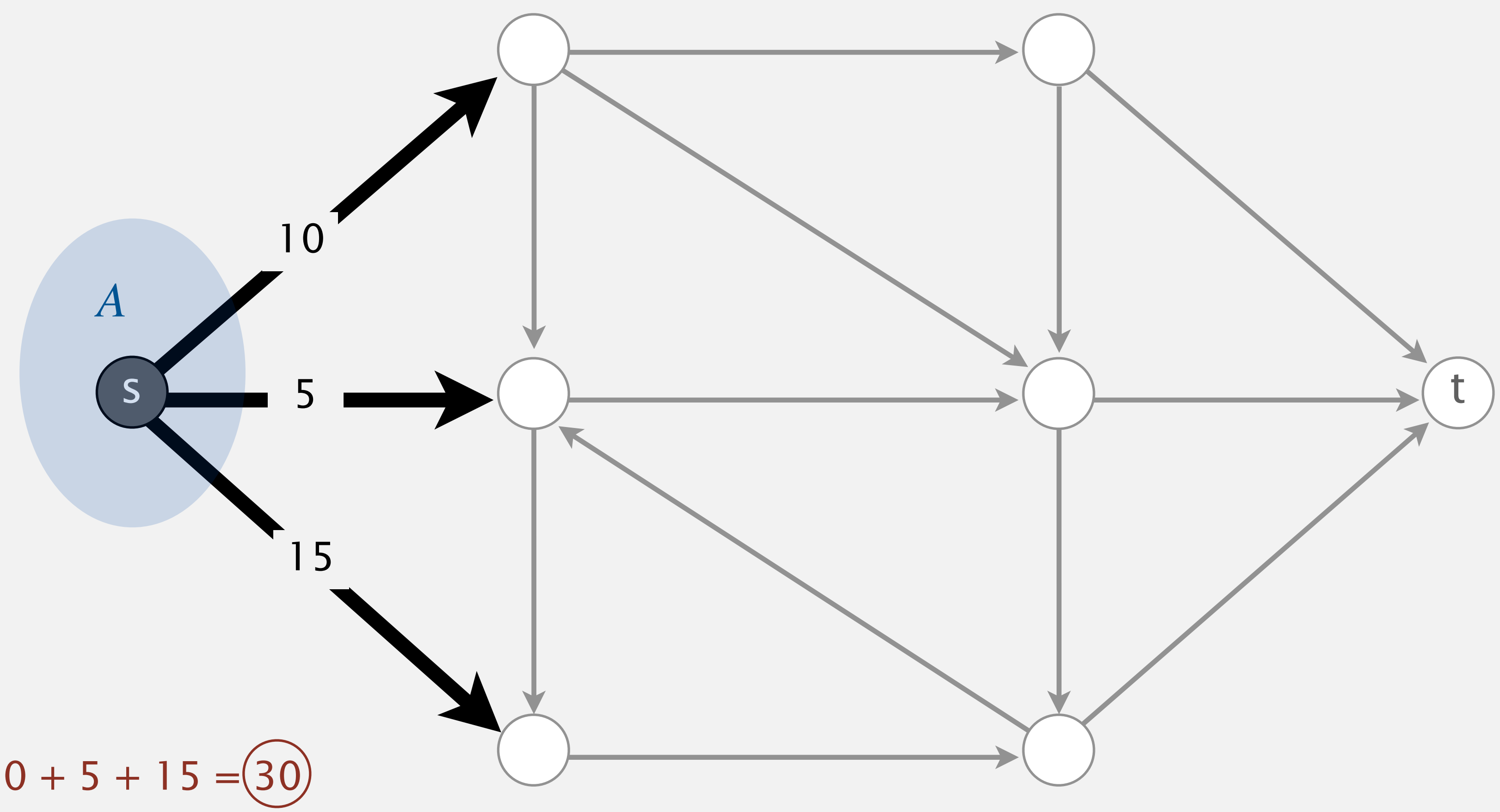
Intuition. Water flows from a source to a sink through a network of pipes. Each pipe has a flow capacity.



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.

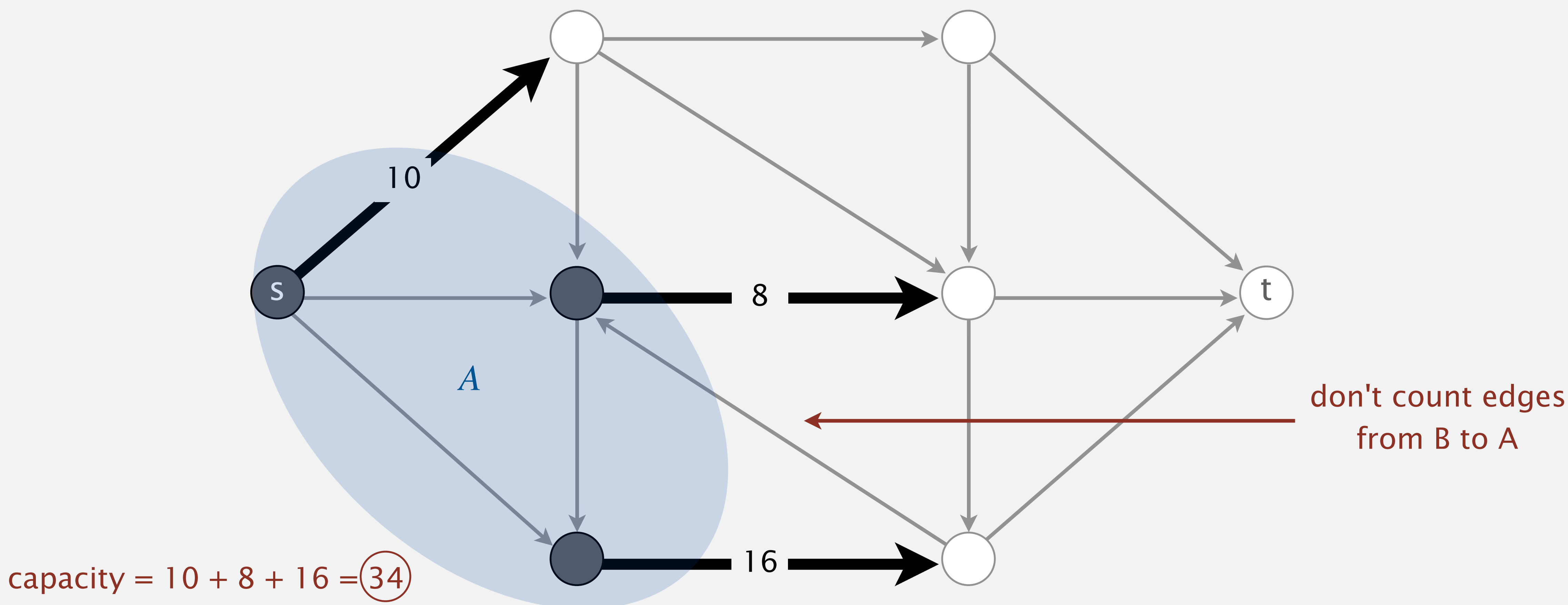


capacity = 10 + 5 + 15 = 30

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.

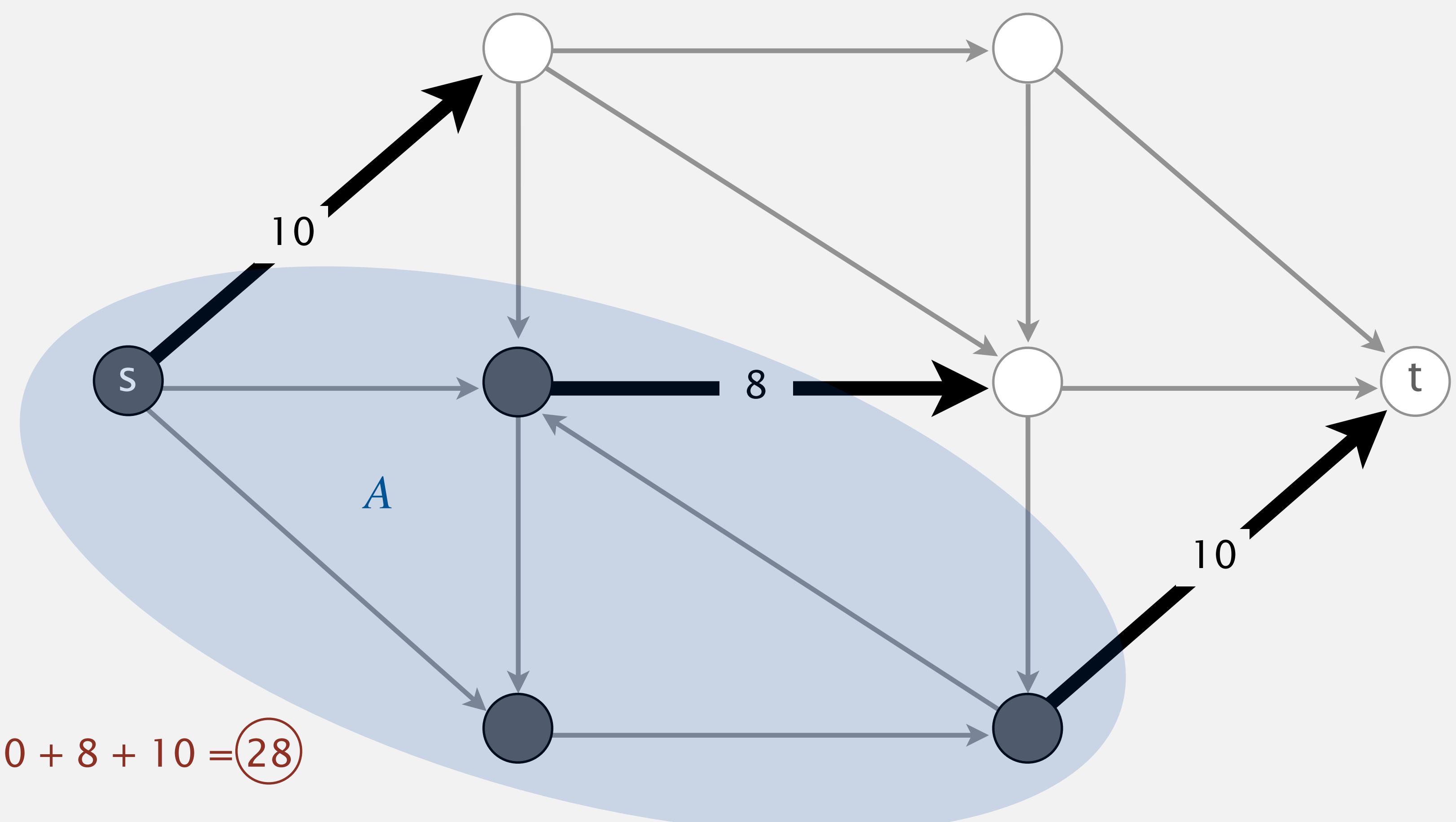


Mincut problem

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

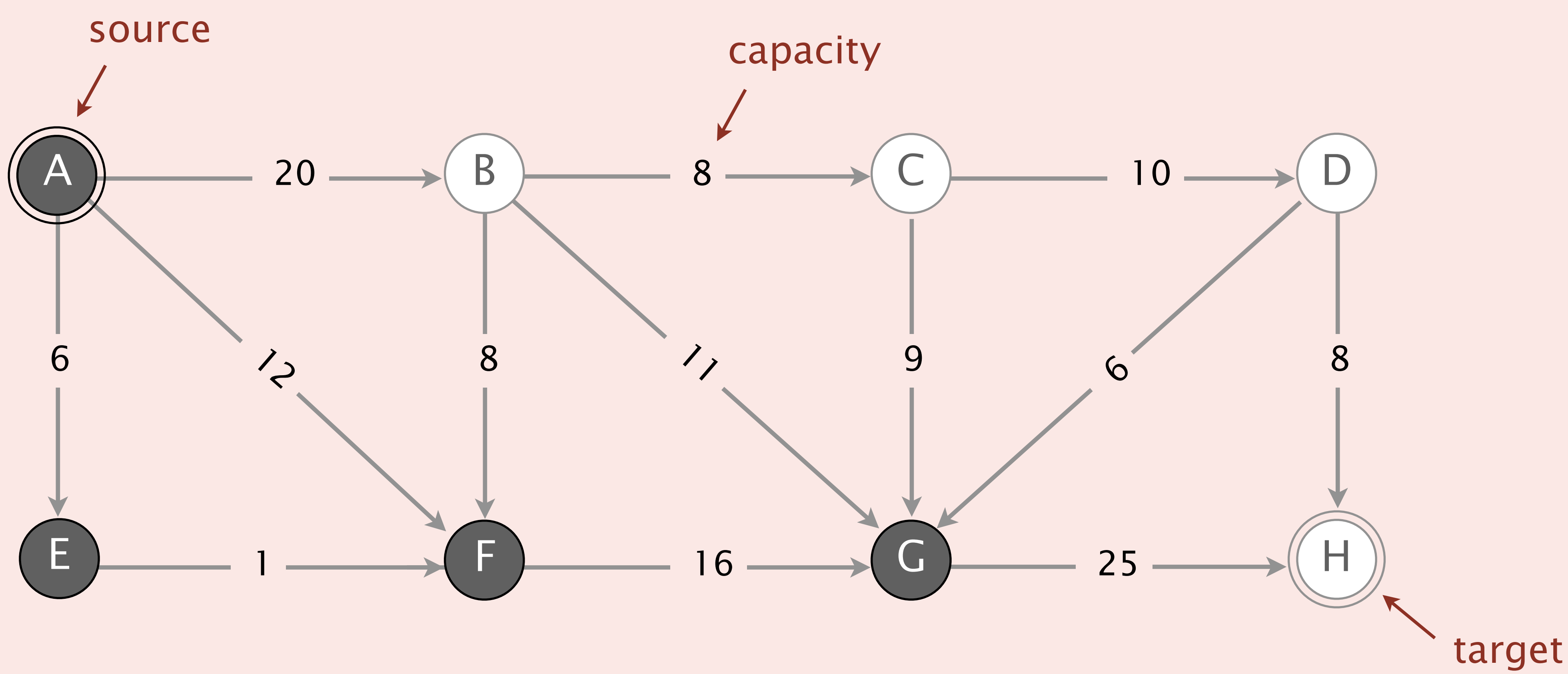
What is the capacity of the st-cut $\{A, E, F, G\}$?

A. 11 (20 + 25 − 8 − 11 − 9 − 6)

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

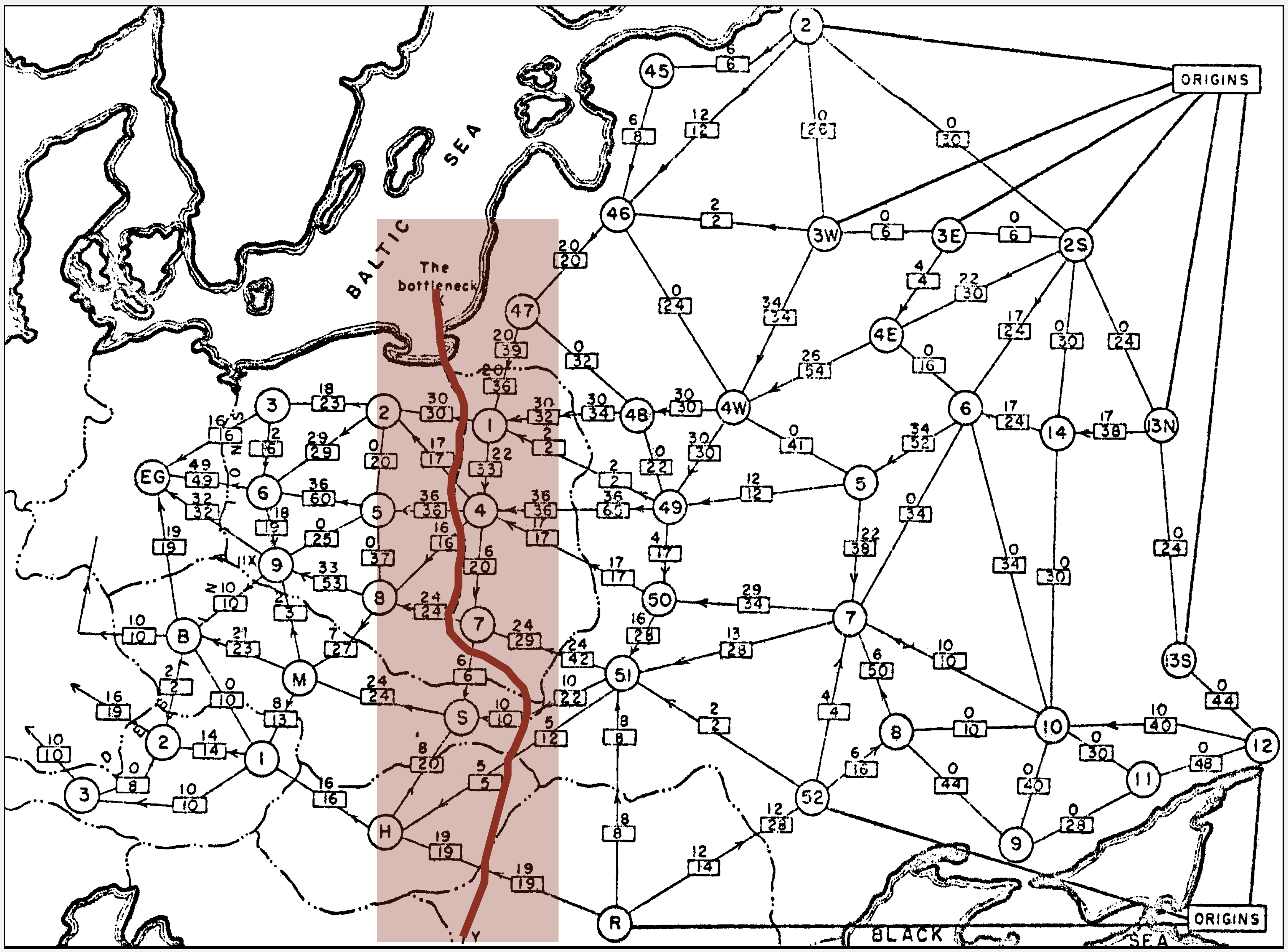
C. 45 (20 + 25)

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



Mincut application

U.S. goal. Cut supplies (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

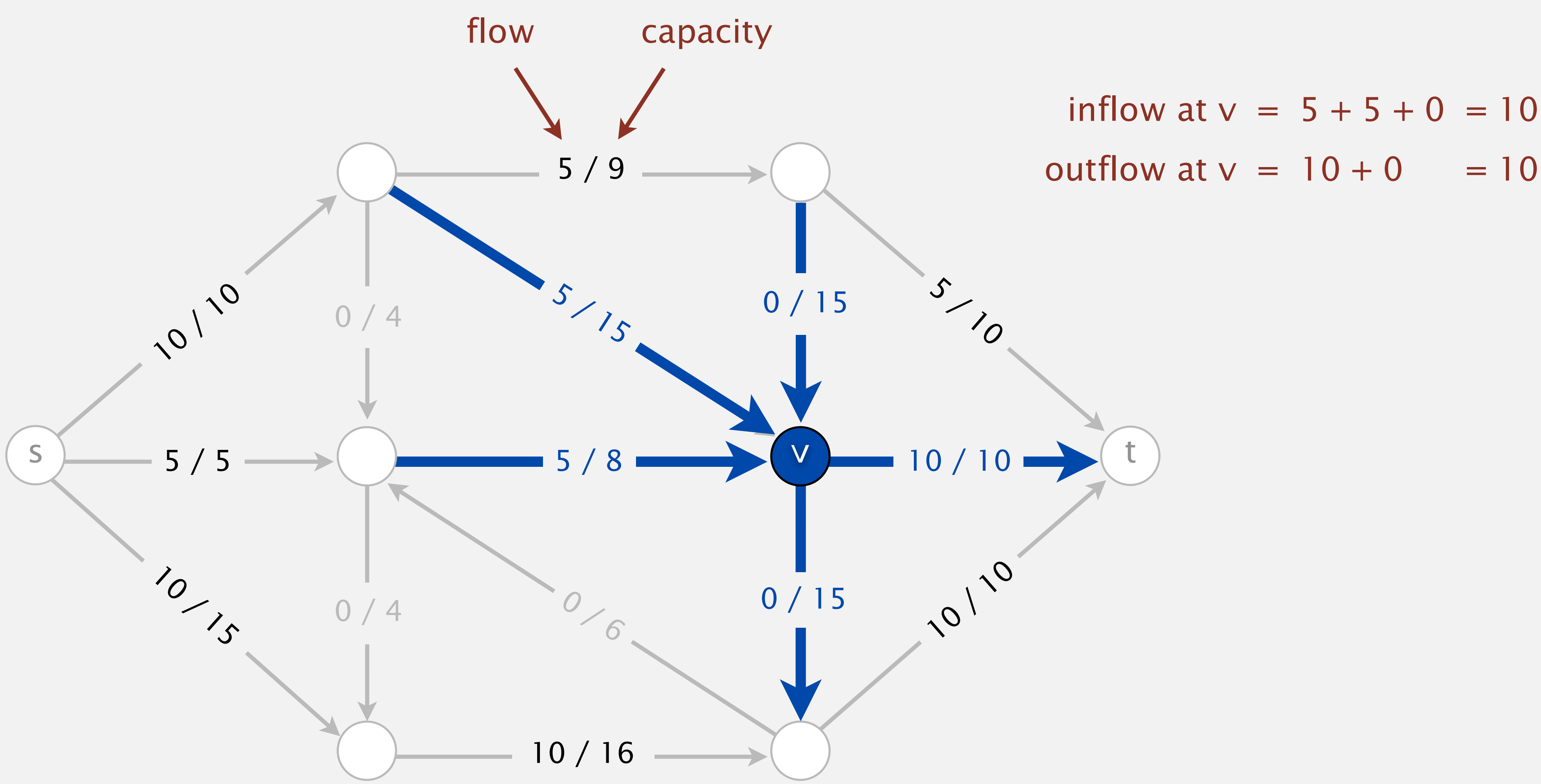
Efficient Maximum Flow Algorithms by Andrew Goldberg and Bob Tarjan

http://vimeo.com/100774435

Maxflow problem

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

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



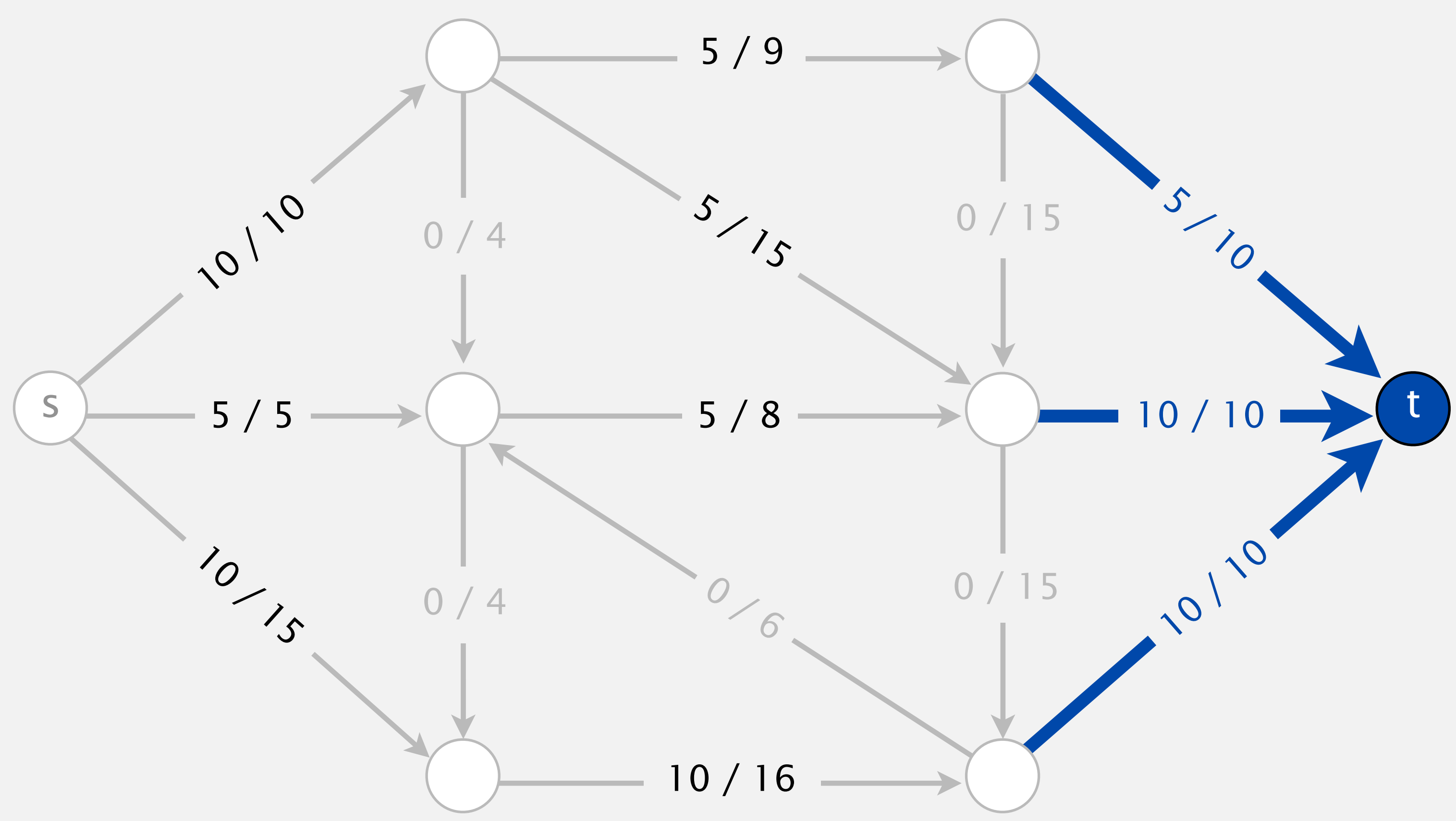
Maxflow problem

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

- Capacity constraint: $0 \le$ edge's flow $\le$ edge's capacity.
- Local equilibrium: inflow = outflow at every vertex (except s and t).

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

we assume no edges point to s or from t



value = 5 + 10 + 10 = 25

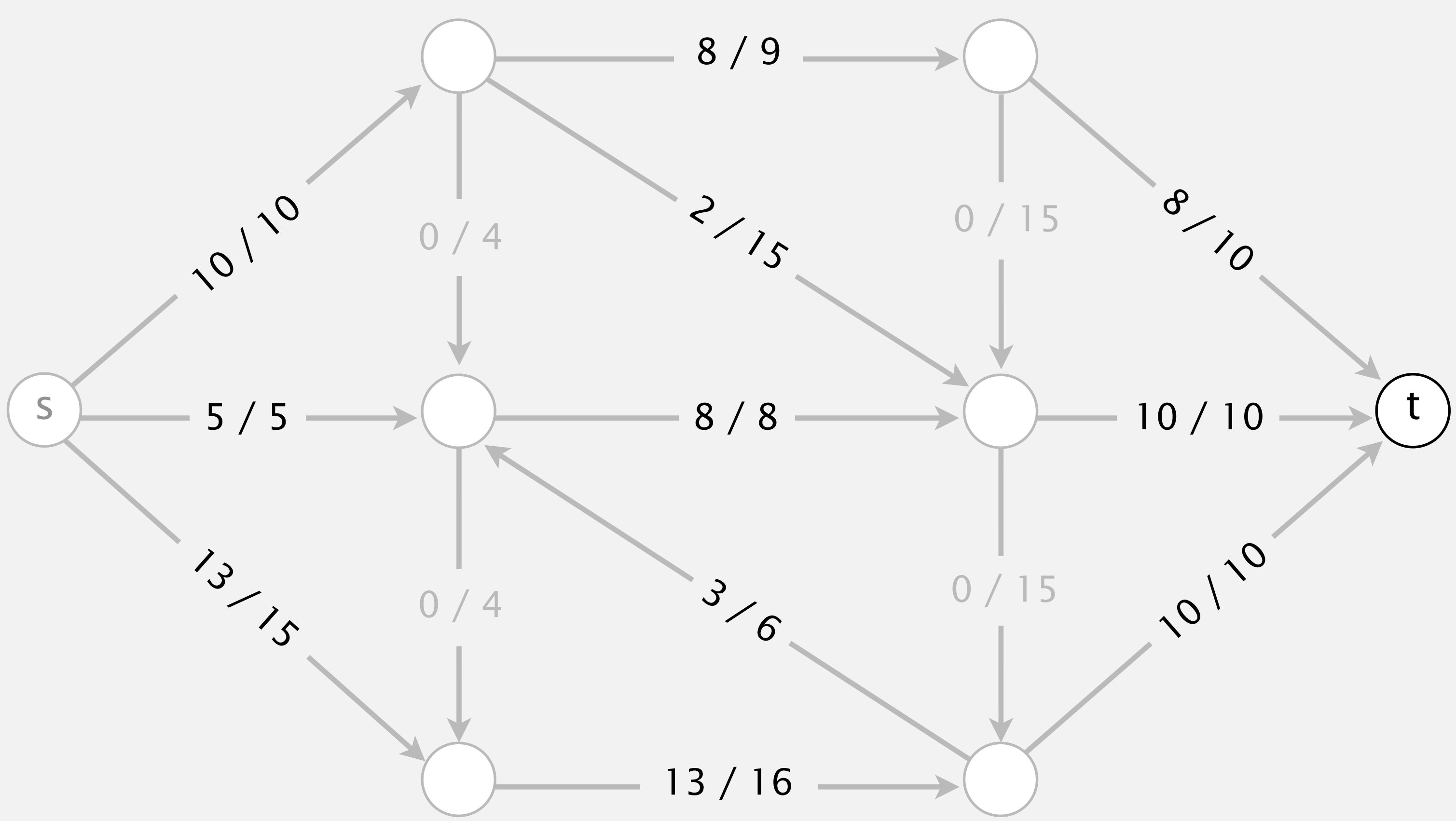
Maxflow problem

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

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

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

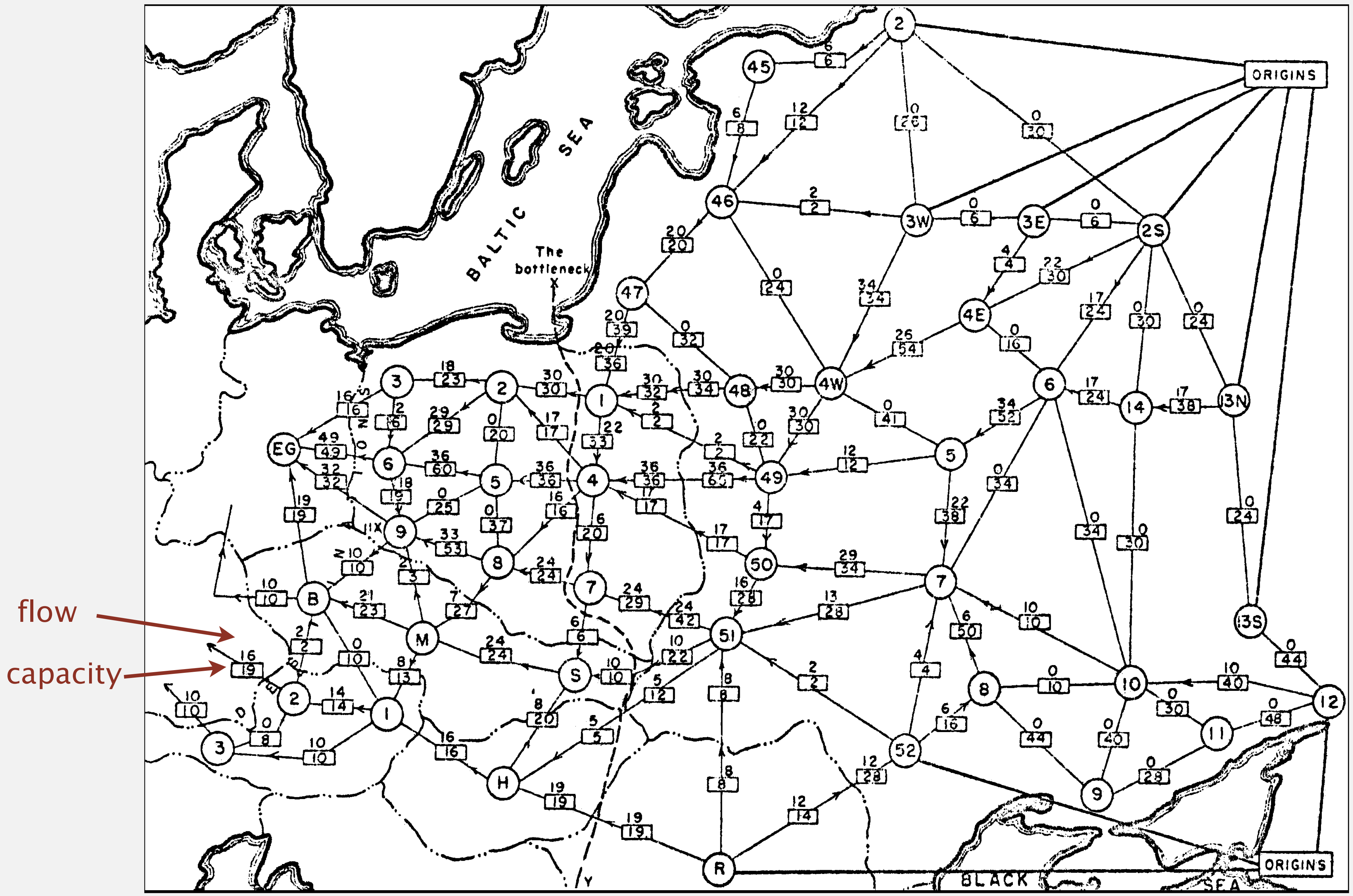
Maximum st-flow (maxflow) problem. Find a flow of maximum value.



maximum flow value
= 8 + 10 + 10 = 28

Maxflow application

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



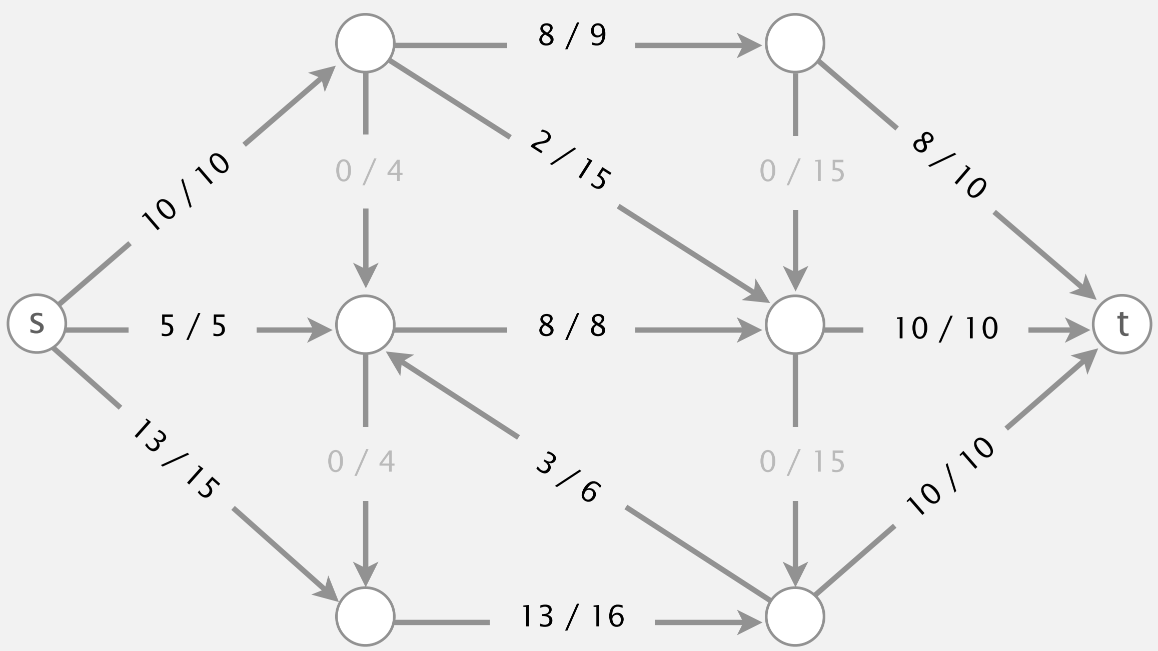
rail network connecting Soviet Union with Eastern European countries
(map declassified by Pentagon in 1999)

Summary

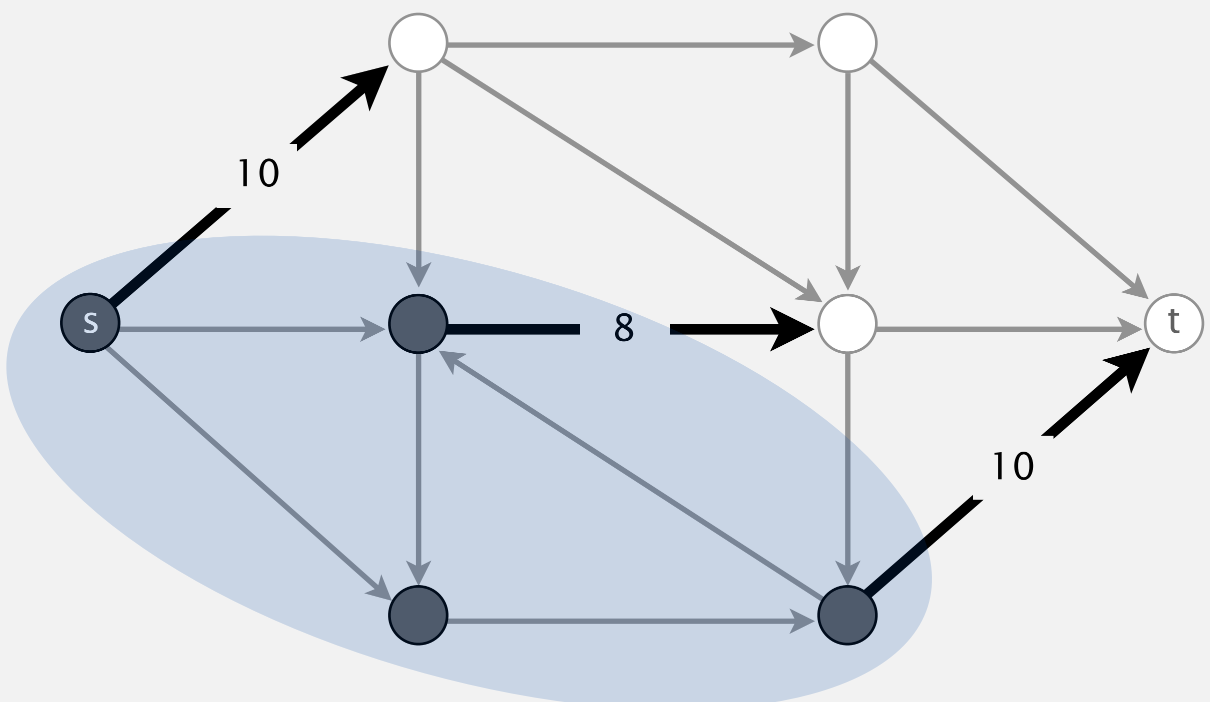
Input. An edge-weighted digraph, source vertex s, and target vertex t.

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!

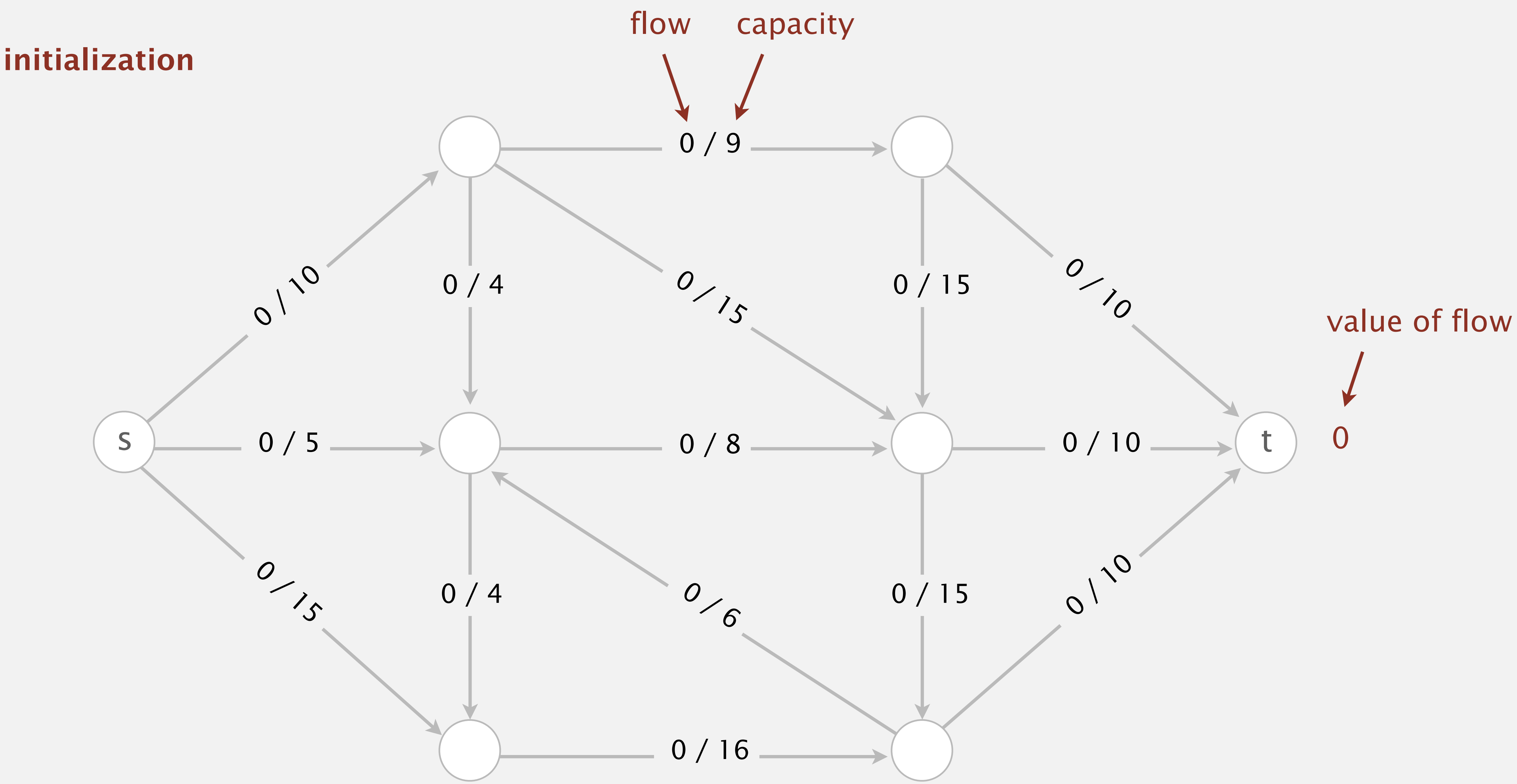
https://algs4.cs.princeton.edu

6.4 MAXIMUM FLOW

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

Ford–Fulkerson algorithm

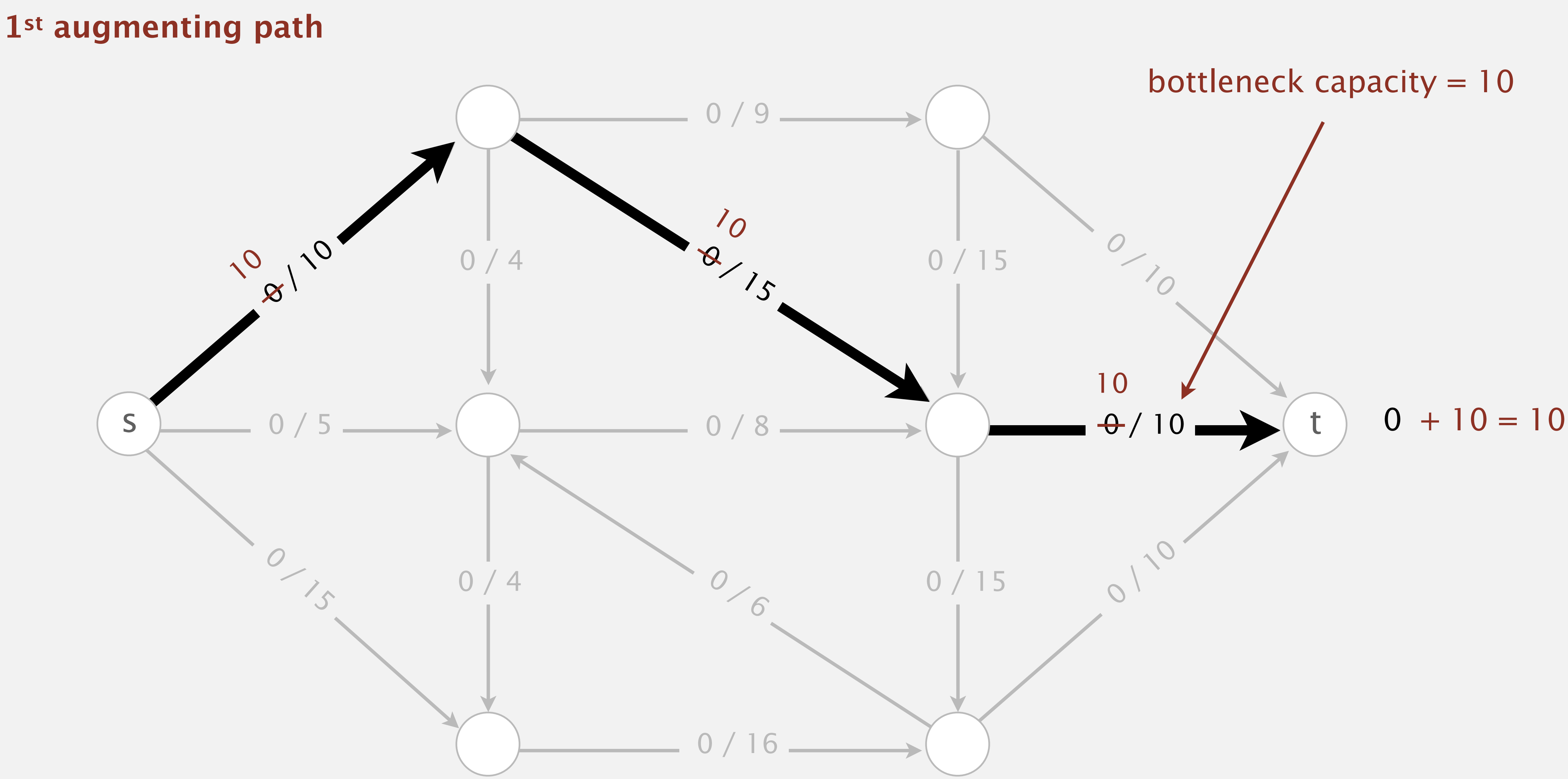
Initialization. Start with 0 flow.



Idea: increase flow along augmenting paths

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).

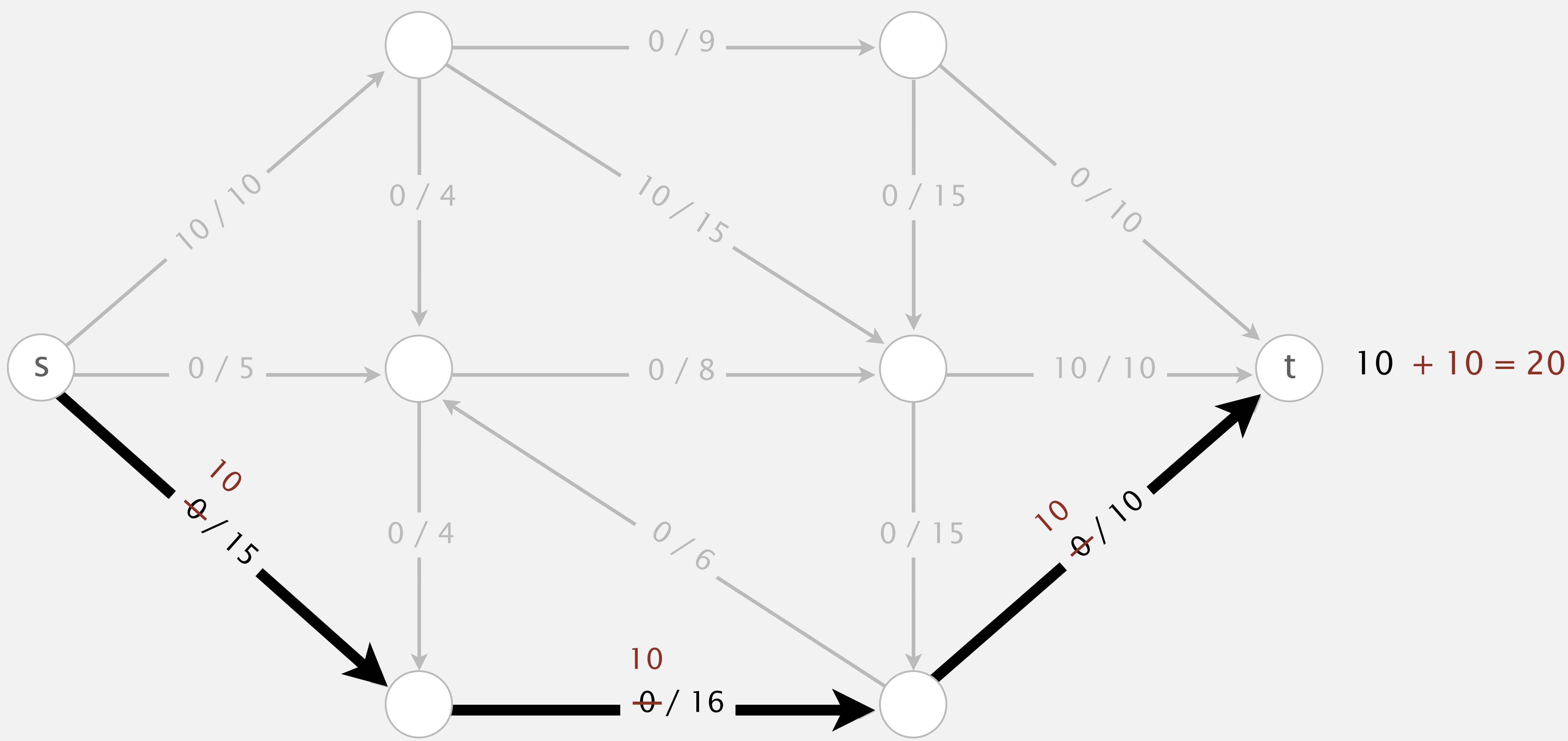


Idea: increase flow along augmenting paths

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).

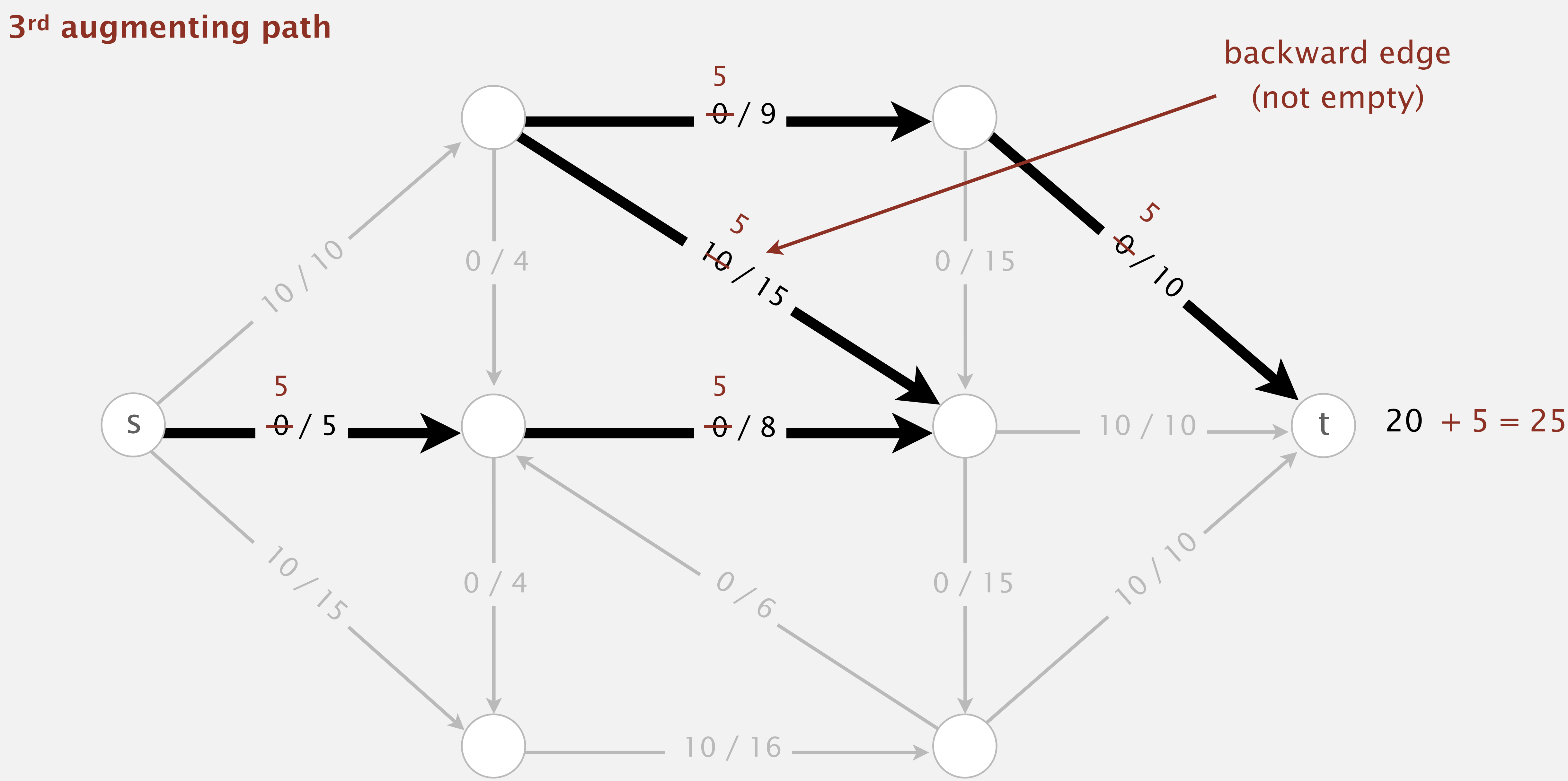
2nd augmenting path



Idea: increase flow along augmenting paths

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).

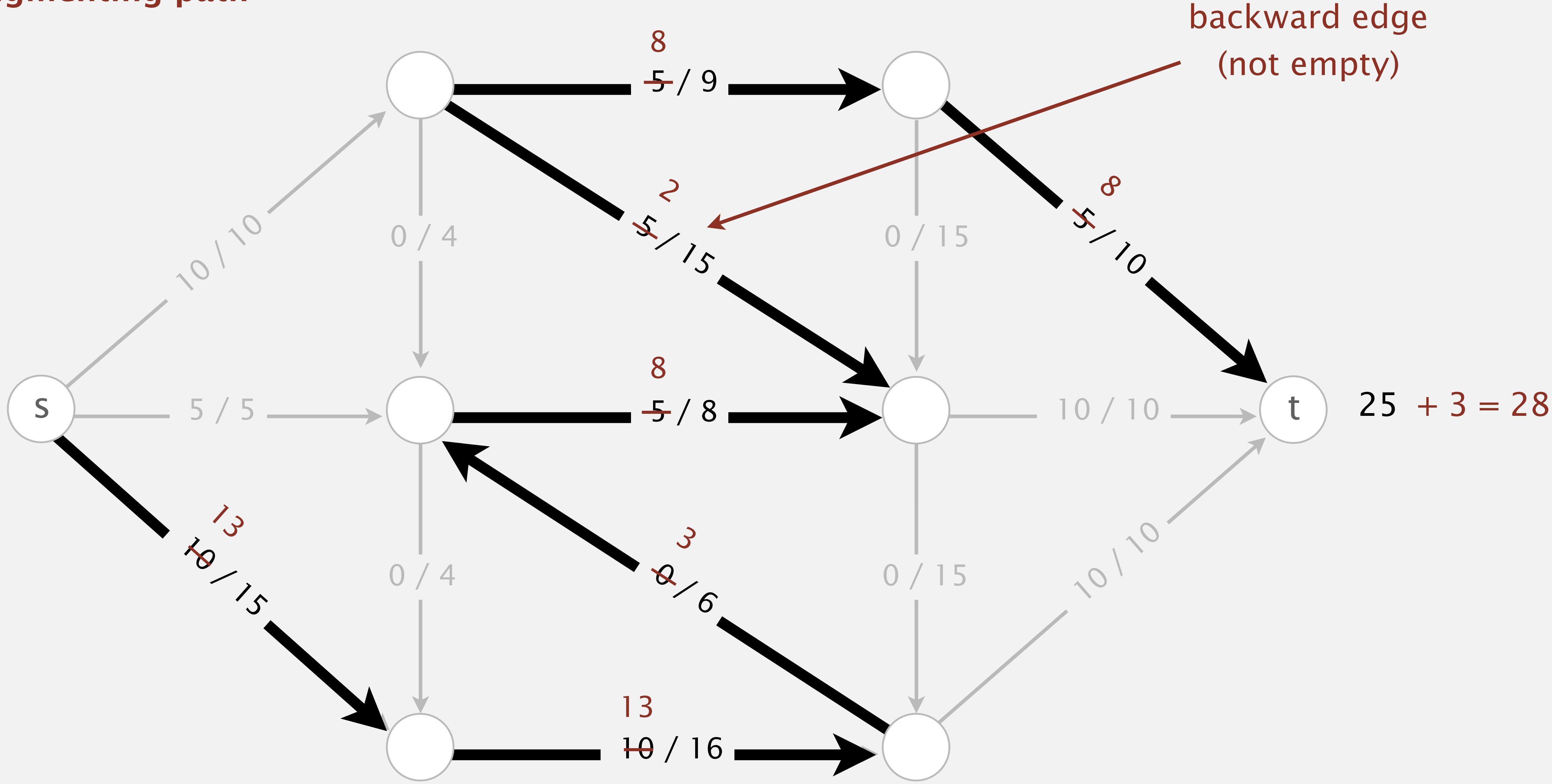


Idea: increase flow along augmenting paths

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).

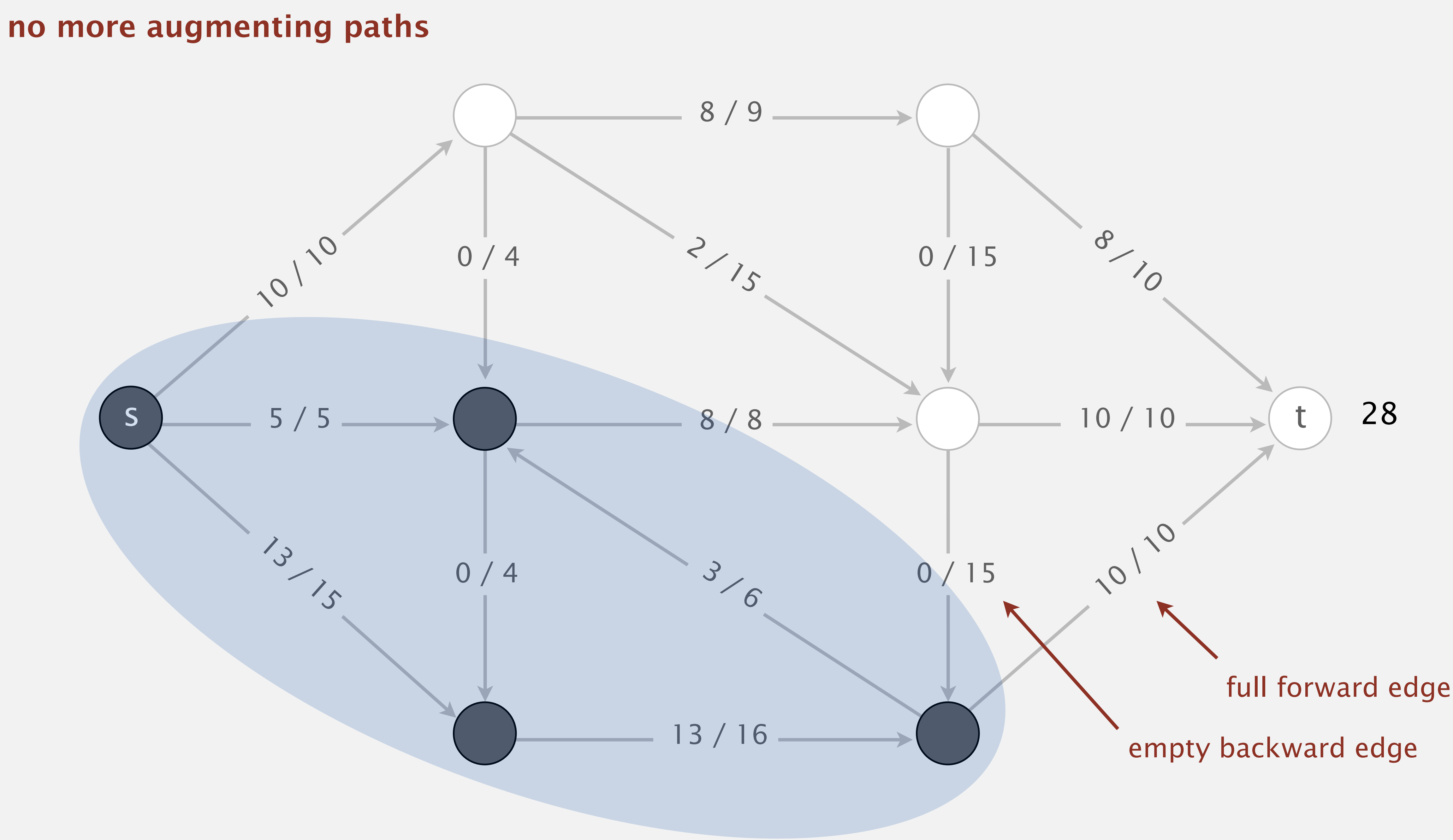
4th augmenting path



Idea: increase flow along augmenting paths

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

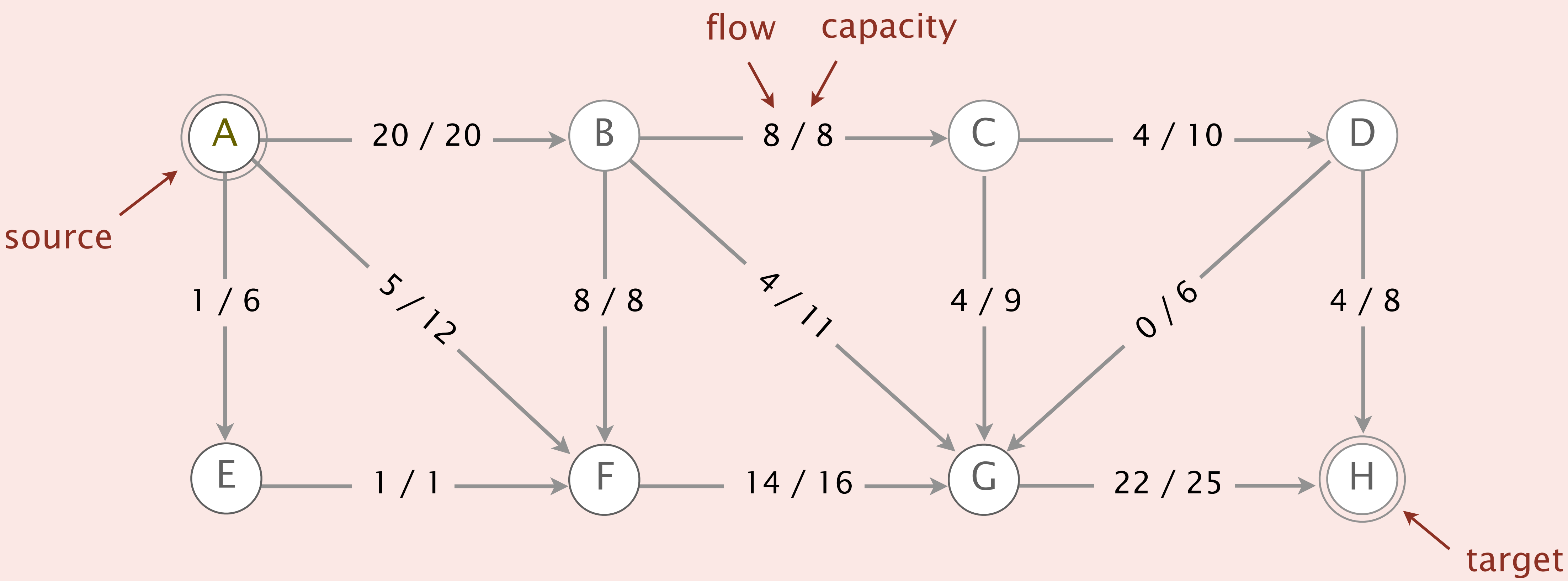
- Full forward edge.
- Empty backward edge.



Maxflow: quiz 2

Which is an augmenting path?

- **A.** $A \rightarrow E \rightarrow F \rightarrow G \rightarrow D \rightarrow H$
- **B.** $A \rightarrow F \rightarrow B \rightarrow G \rightarrow C \rightarrow D \rightarrow H$
- **C.** Both A and B.
- **D.** Neither A nor B.



Ford–Fulkerson algorithm

Ford–Fulkerson algorithm

Start with 0 flow.

While there exists an augmenting path:

- **find an augmenting path**
- **compute bottleneck capacity**
- **update flow on that path 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?

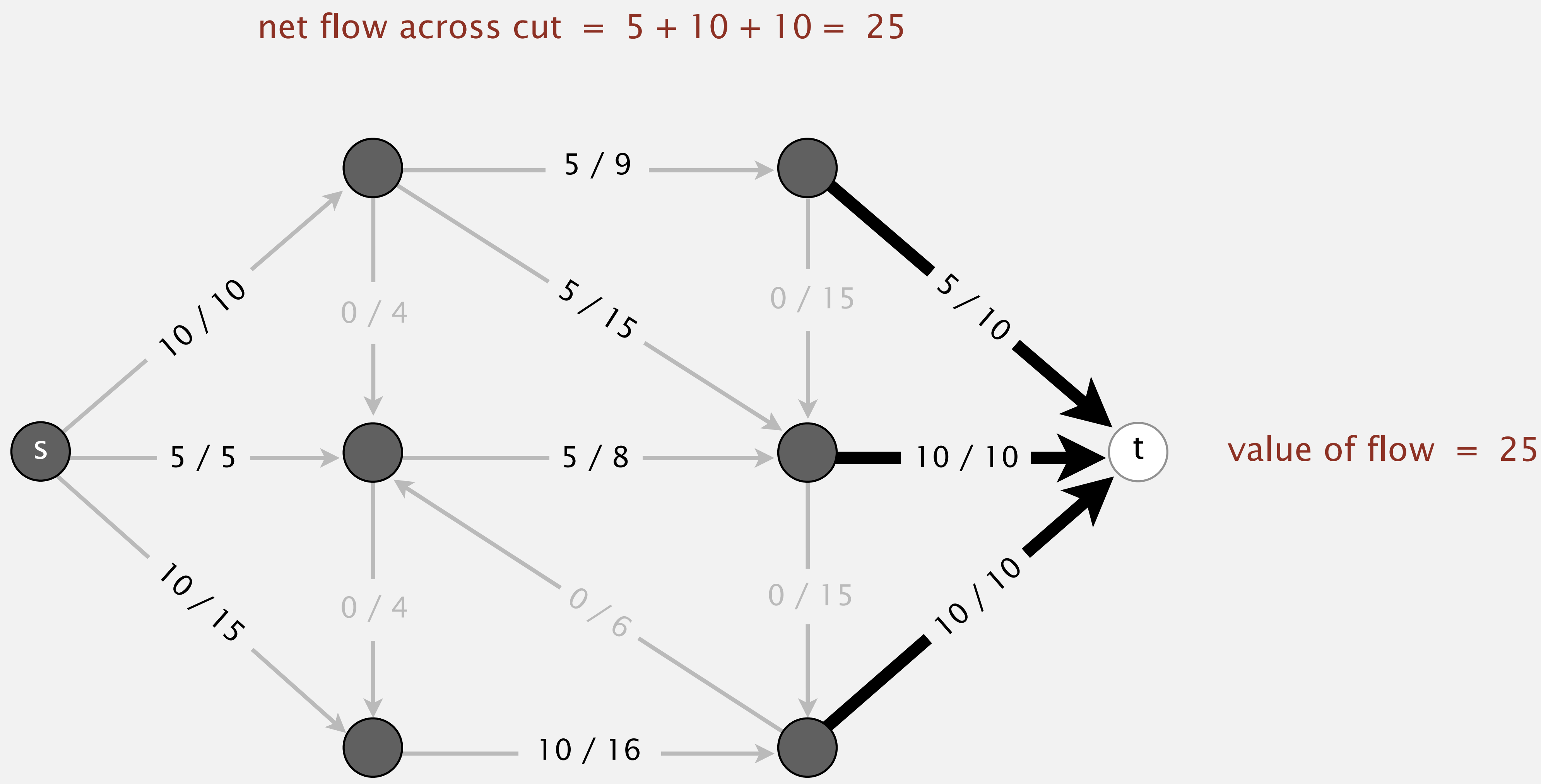
https://algs4.cs.princeton.edu

6.4 Maximum Flow

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

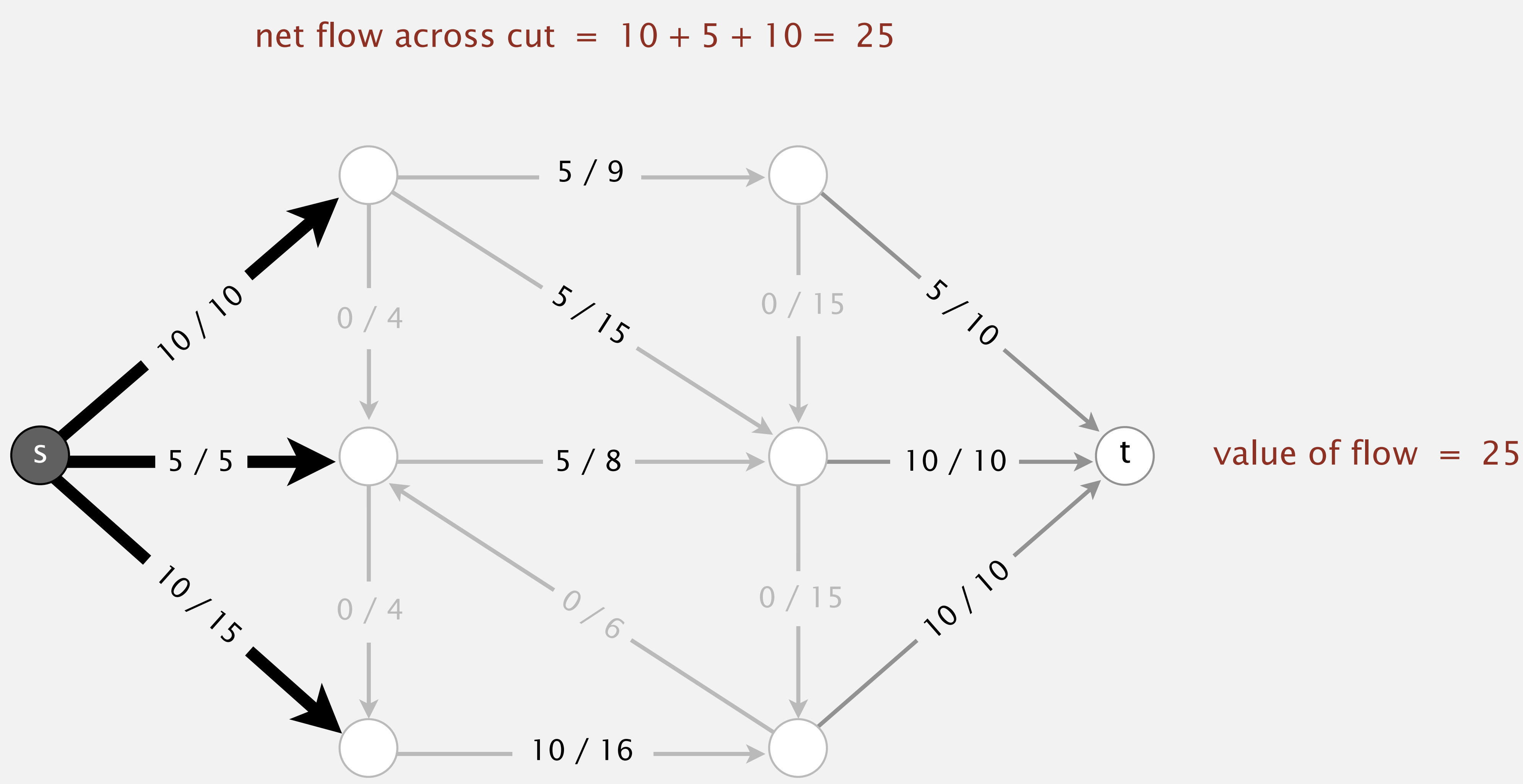
Relationship between flows and cuts

Def. 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

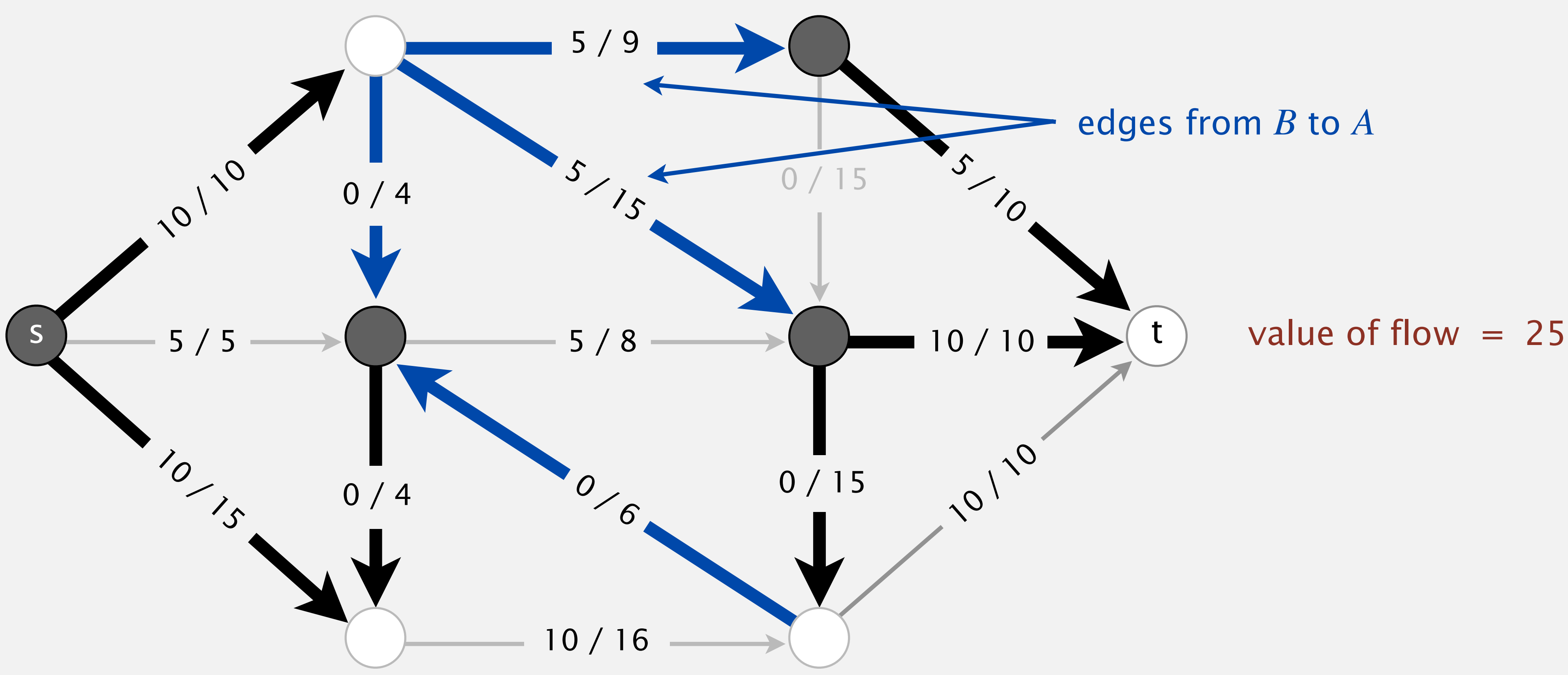
Def. The net flow across a cut (A, B) is the sum of the flows on its edges from A to B minus the sum of the flows on its edges from B to A.



Relationship between flows and cuts

Def. 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 + 5 + 10 + 0 + 0) – (5 + 5 + 0 + 0) = 25



Maxflow: quiz 3

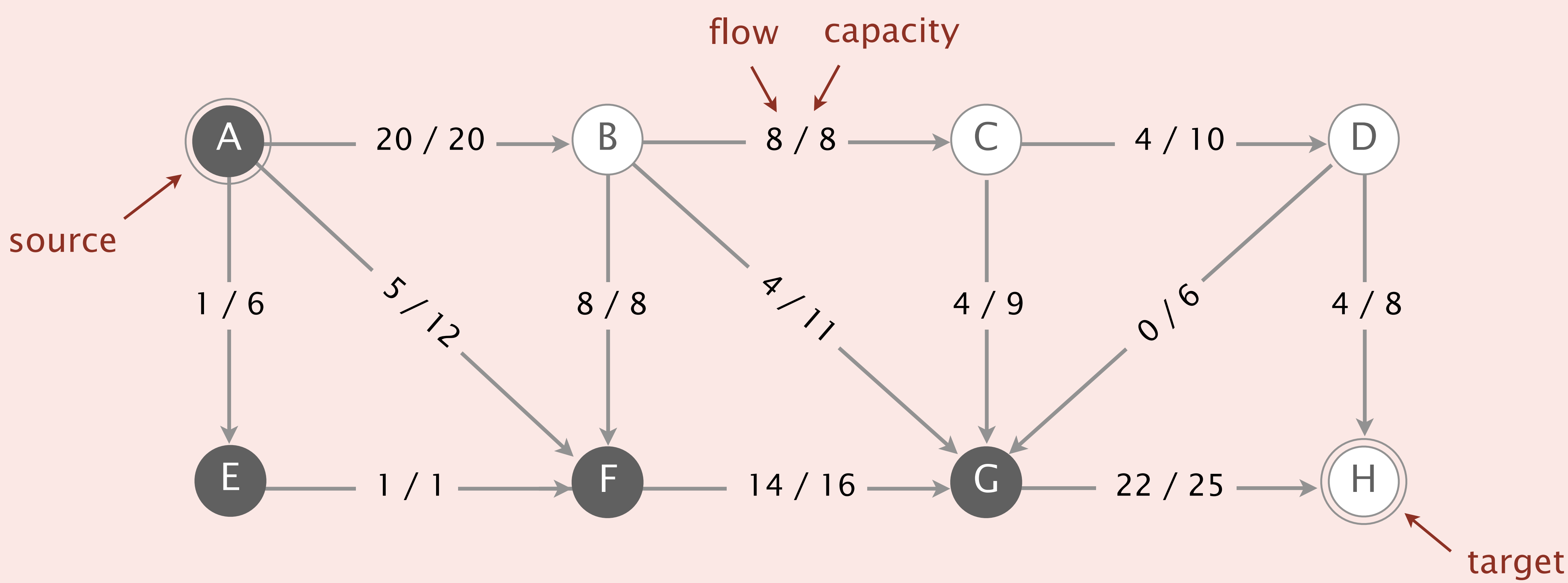
Which is the net flow across the st-cut $\{A, E, F, G\}$?

A. $11 \ (20 + 25 - 8 - 11 - 9 - 6)$

B. $26 \ (20 + 22 - 8 - 4 - 4)$

C. $42 \ (20 + 22)$

D. $45 \ (20 + 25)$



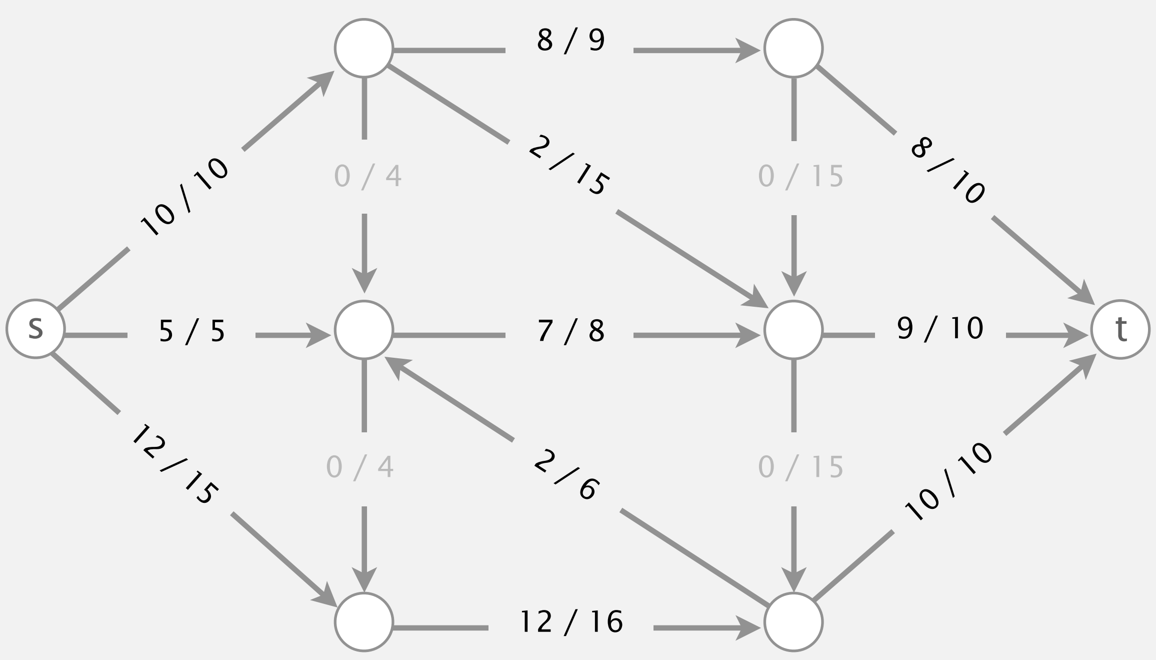
Relationship between flows and cuts

Property 1. The net flow across any cut is the same as the outflow from the source and the inflow to the target.

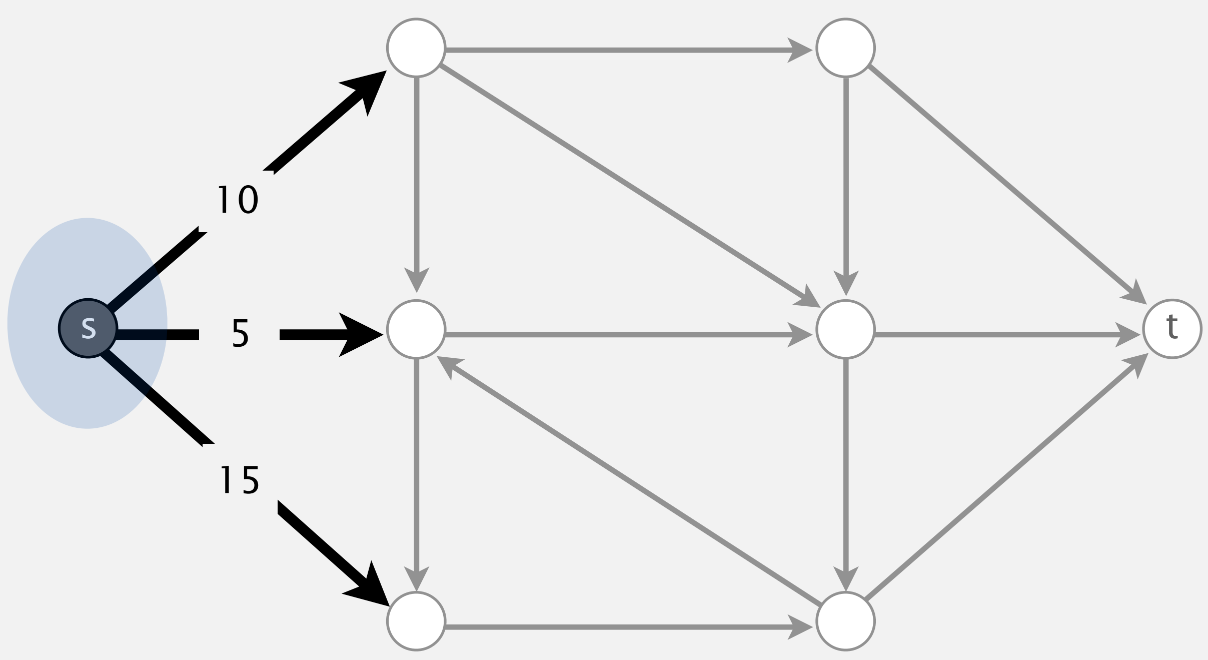
Intuition. Conservation of flow.

Property 2. Value of any flow ≤ capacity of any cut.

Intuition. Flow is bounded by capacity.



value of flow = 27

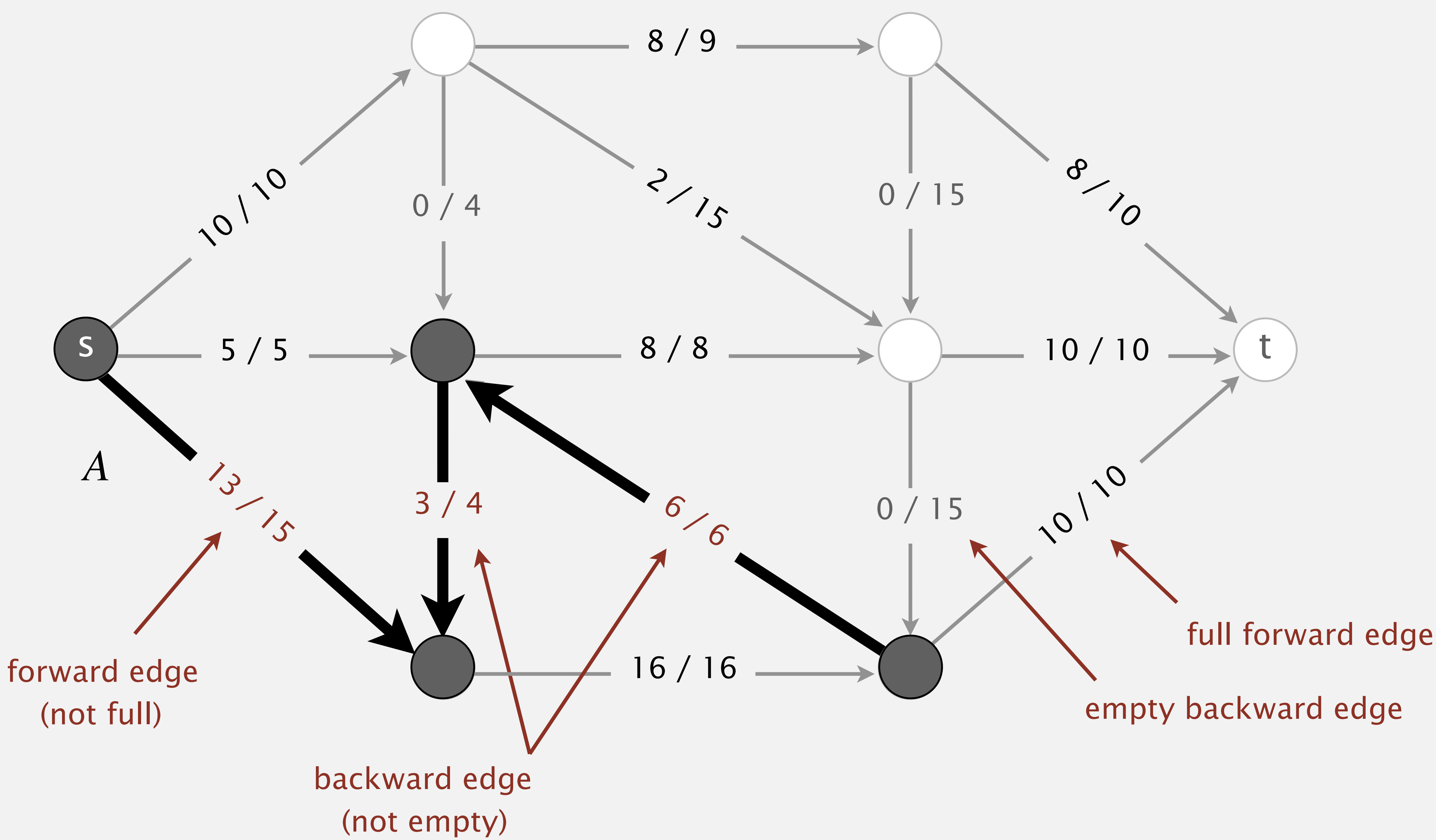


capacity of cut = 30

Exercise: computing a cut from a maxflow

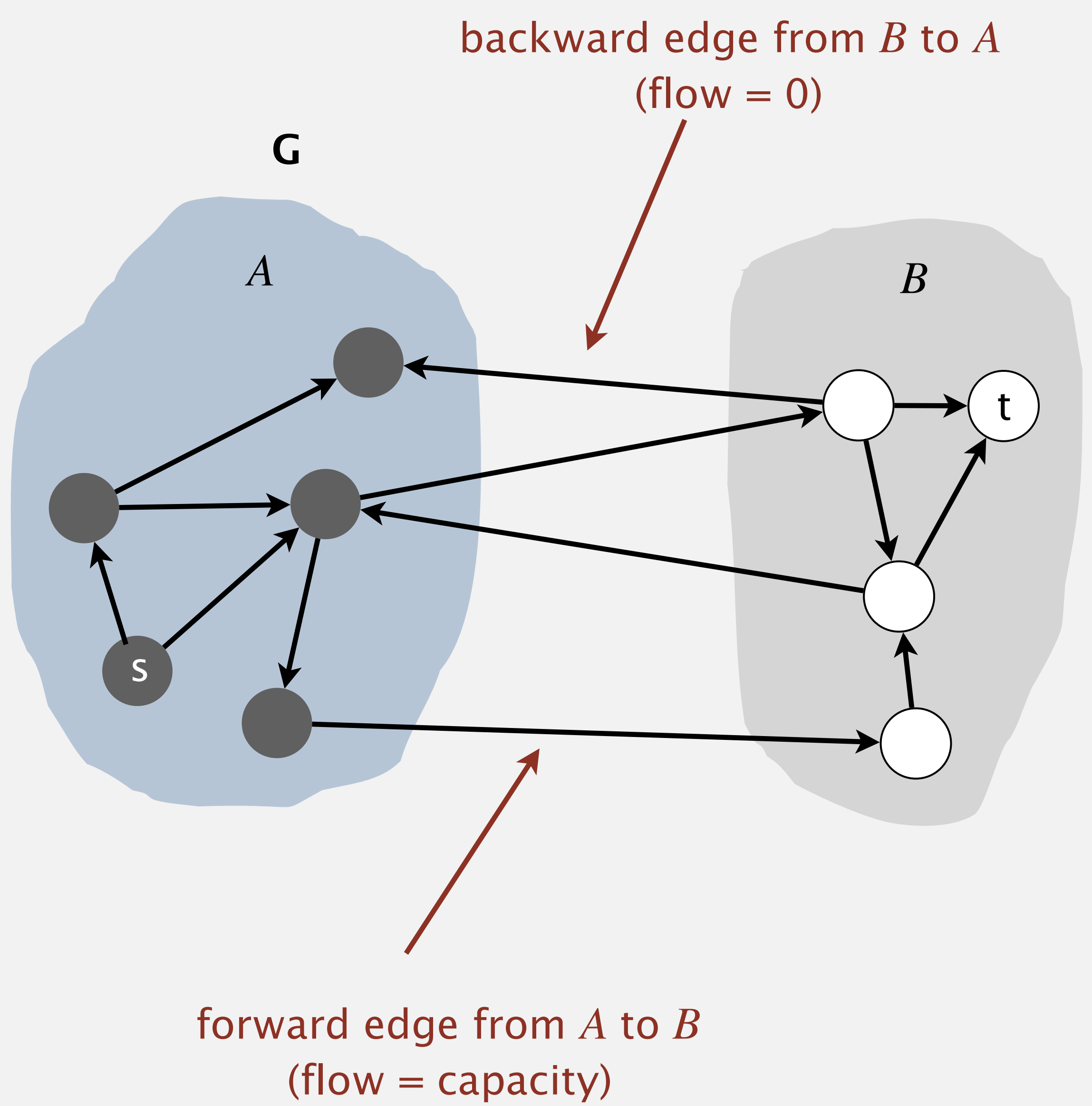
Try to find augmenting paths. Go as far as you can; stop when you can go no further

- Note: 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; B = all other vertices.
- What are the properties of the edges crossing the cut (A, B)?
- What is the capacity of the cut?



Computing a mincut from a maxflow

- Note: 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.
- Since value of any flow ≤ capacity of any cut, this must be a mincut!



By construction of cut:

Net flow across cut = capacity of cut

Maxflow-mincut theorem

Augmenting path theorem. A flow f is a maxflow iff no augmenting paths.

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

Alternative formulation. For any flow f, these three conditions are equivalent:

i. f is a maxflow.

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

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

Proof.

[i $\Rightarrow$ ii] If condition ii is false, then there is an augmenting path, so we can improve f by sending flow across that path [which contradicts condition i].

[ii $\Rightarrow$ iii] There is an algorithm to construct such a cut from a flow [prev slide]

[iii $\Rightarrow$ i] Since value of any flow $\leq$ capacity of any cut.

Maxflow: quiz 4

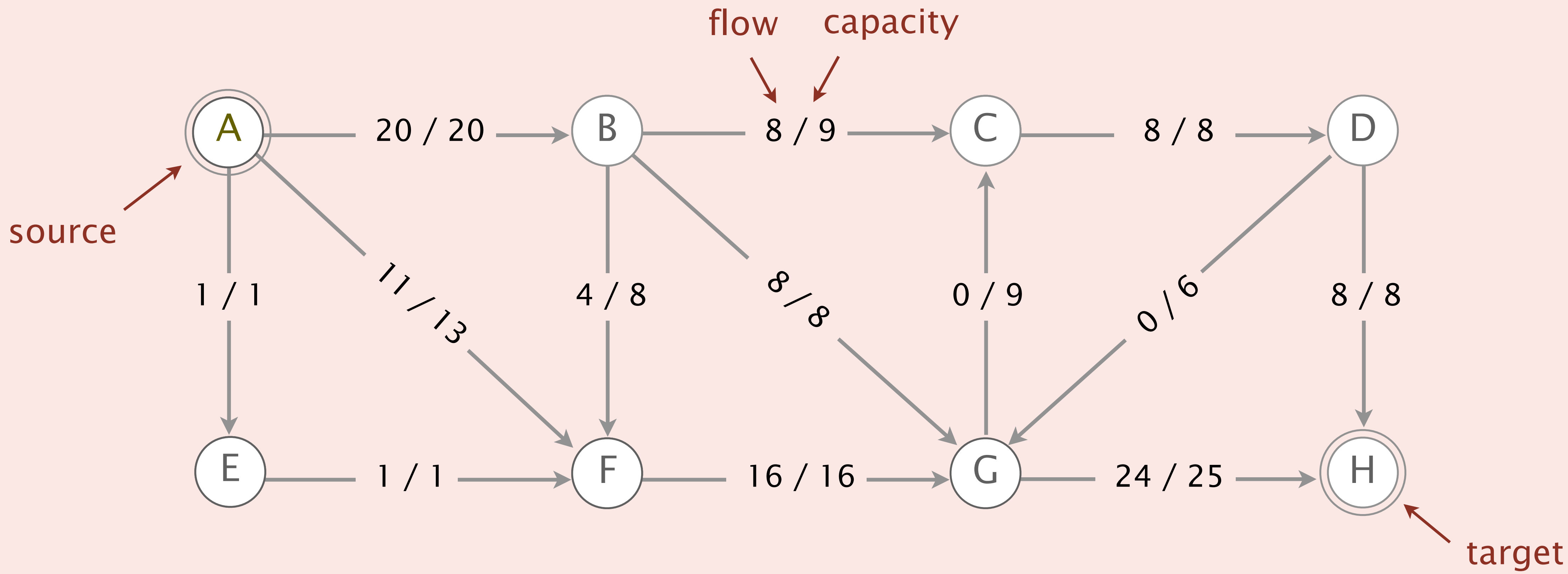
Given the following maxflow, which is a mincut?

A. $S = \{ A \}$.

B. $S = \{ A, B, C, E, F \}$.

C. Both A and B.

D. Neither A nor B.



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 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. [by induction]

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

Proposition. Number of augmentations ≤ the value of the maxflow.

Pf. Each augmentation increases the value by at least 1. ▪

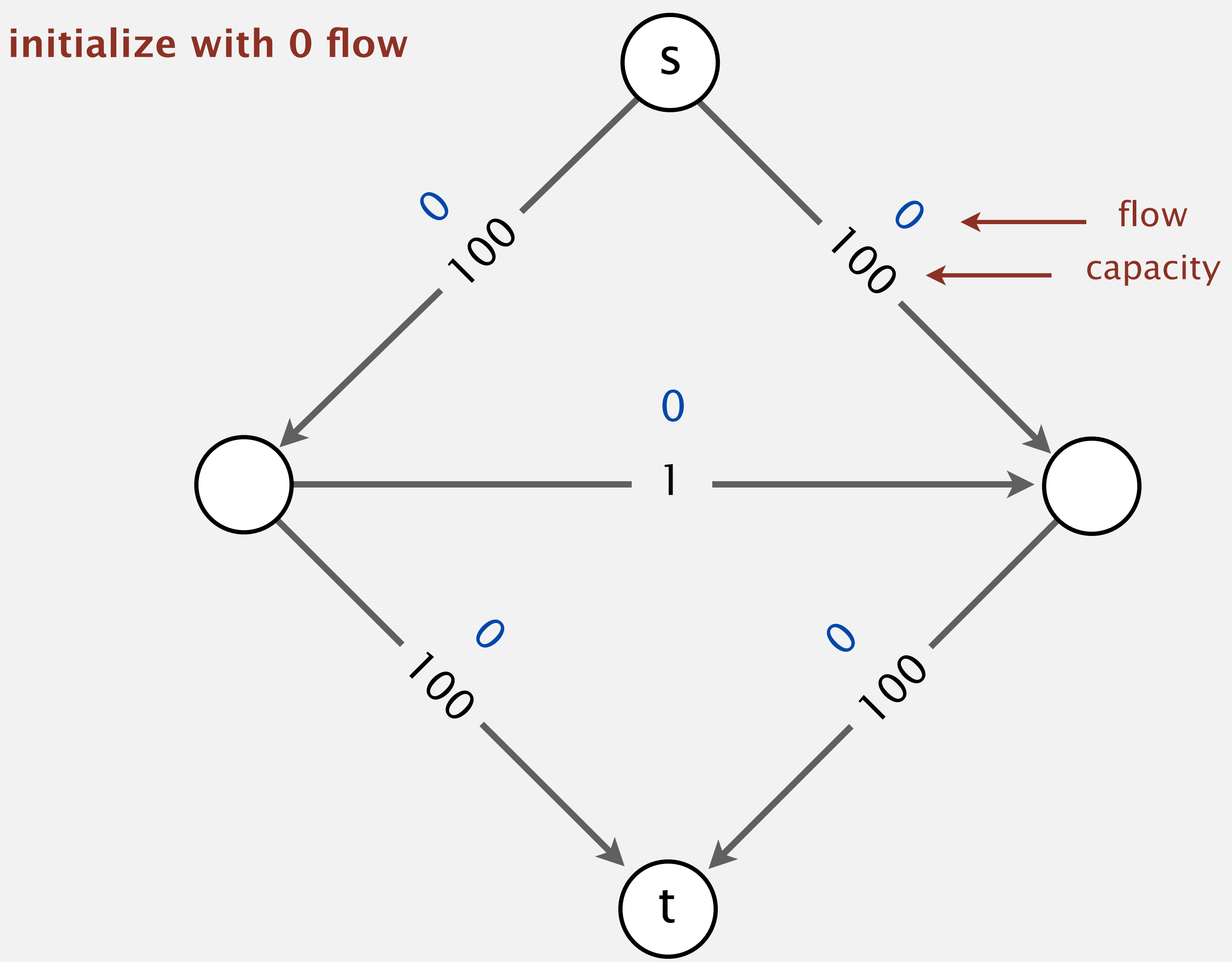
Integrality theorem. There exists an integral maxflow.

Pf.

- Proposition + Augmenting path theorem $\Rightarrow$ FF terminates with maxflow.
- Proposition + Invariant $\Rightarrow$ FF terminates with an integral flow. ▪

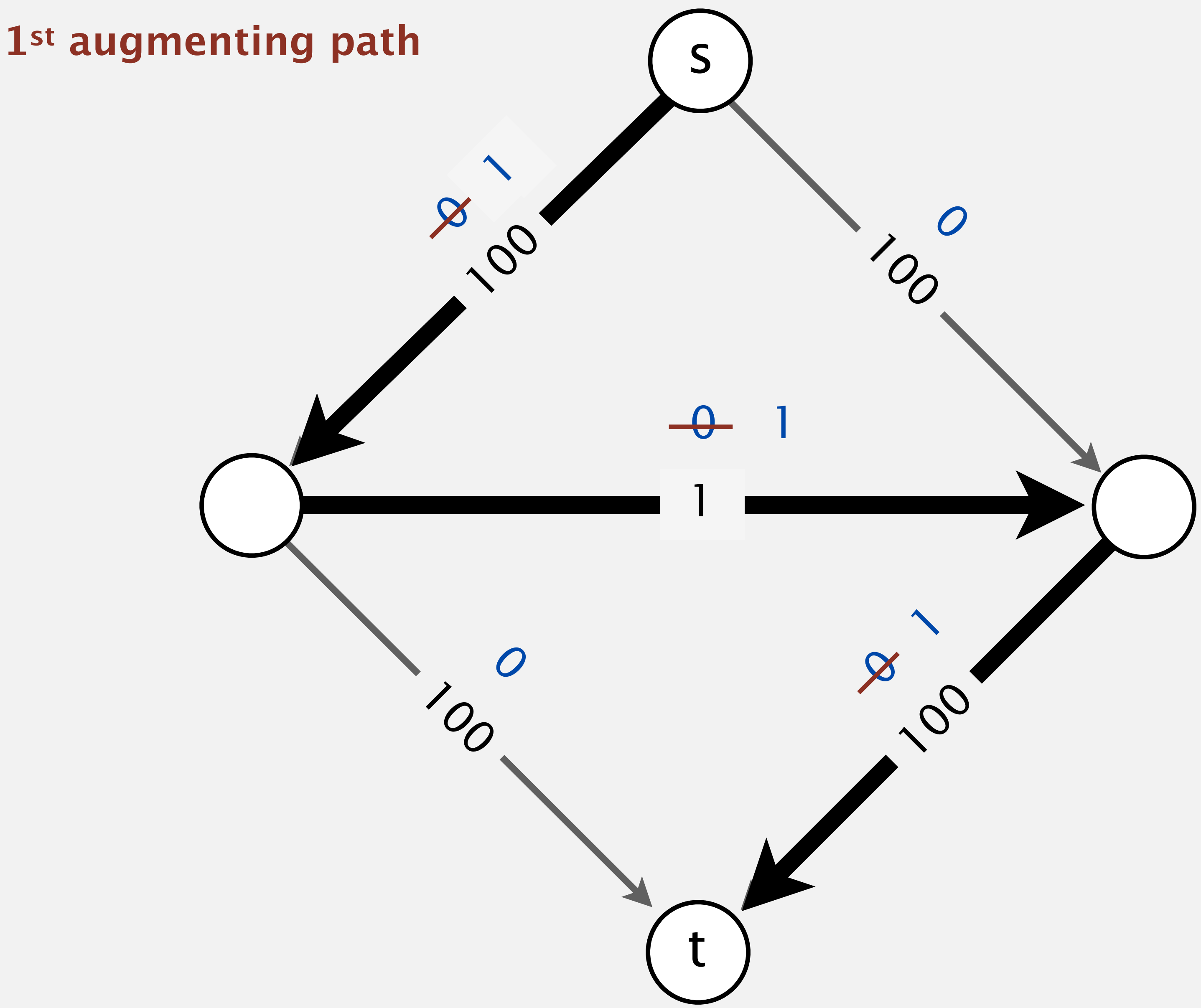
Bad case for Ford–Fulkerson

Bad news. Even when edge capacities are integers, number of augmenting paths could be very large.



Bad case for Ford–Fulkerson

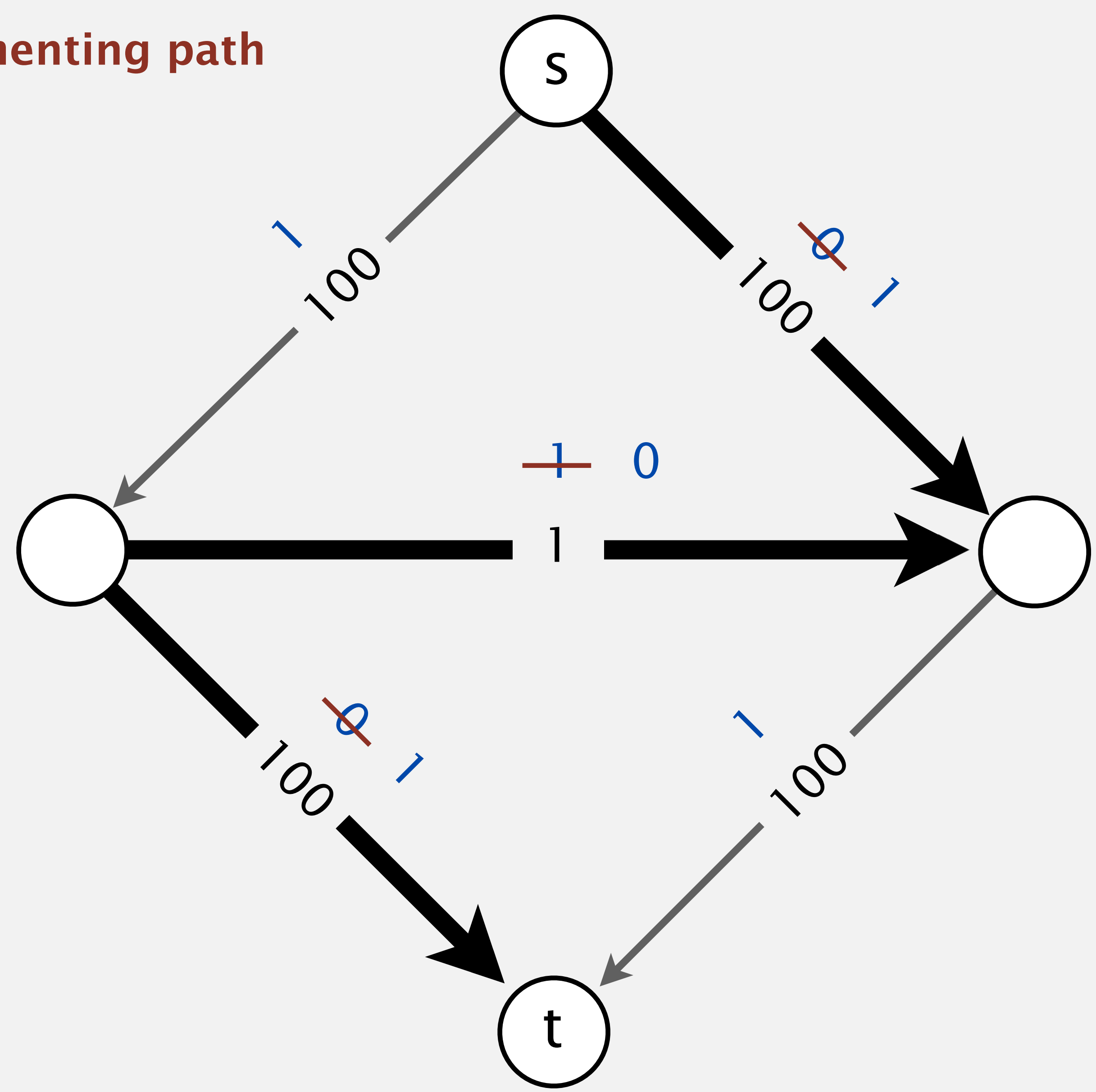
Bad news. Even when edge capacities are integers, number of augmenting paths could be very large.



Bad case for Ford–Fulkerson

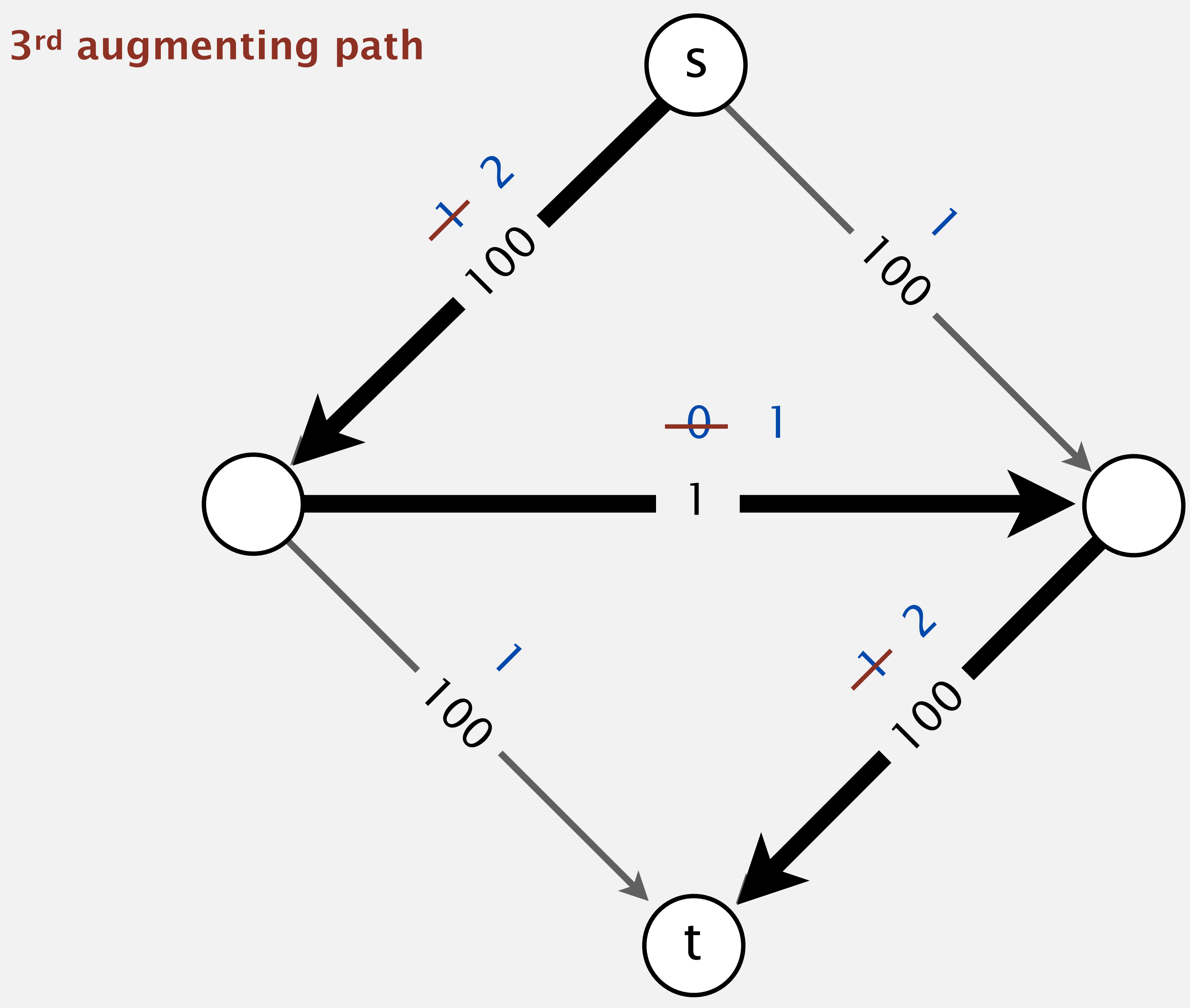
Bad news. Even when edge capacities are integers, number of augmenting paths could be very large.

2nd augmenting path



Bad case for Ford–Fulkerson

Bad news. Even when edge capacities are integers, number of augmenting paths could be very large.

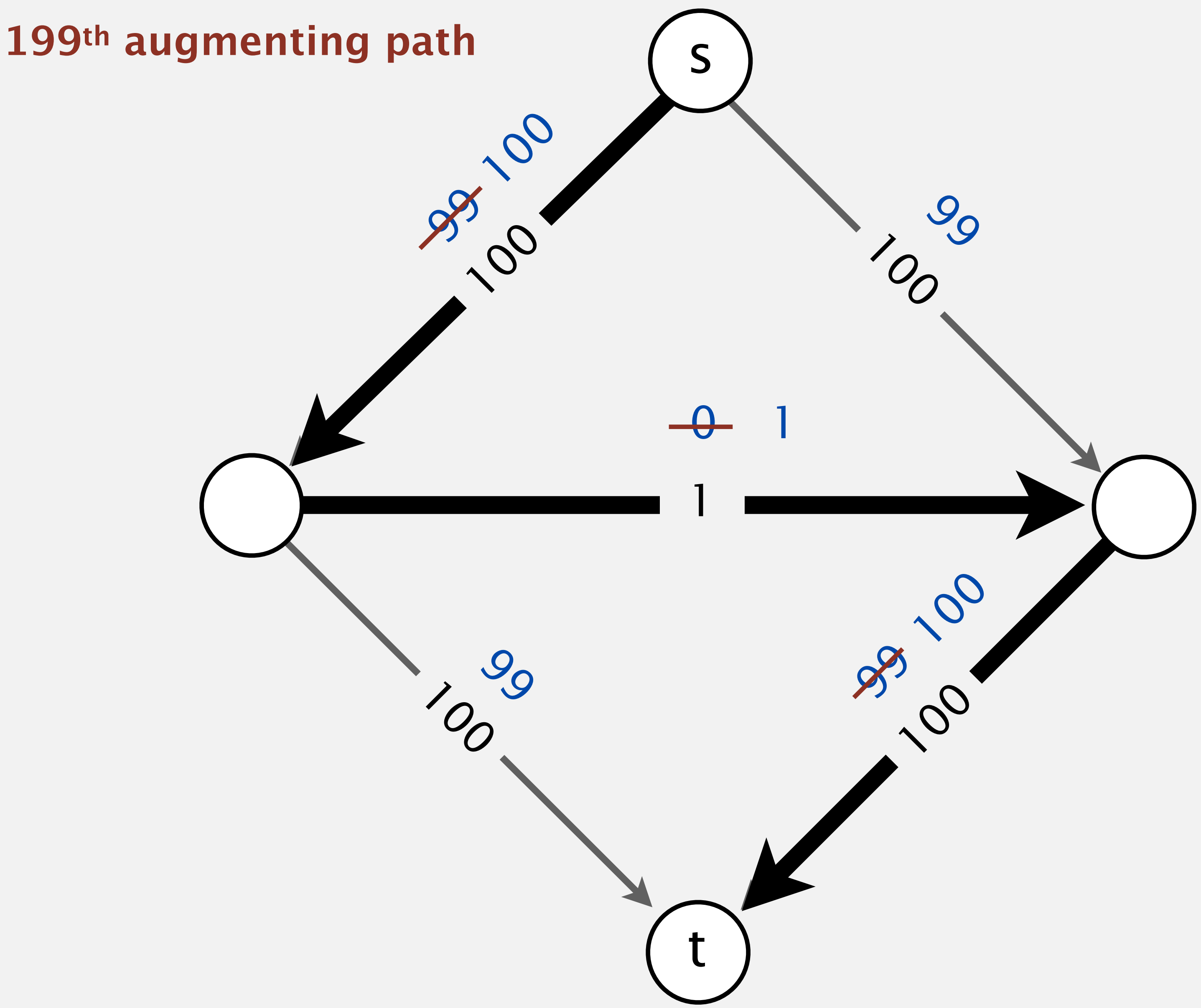


Bad case for Ford–Fulkerson

Bad news. Even when edge capacities are integers, number of augmenting paths could be very large.

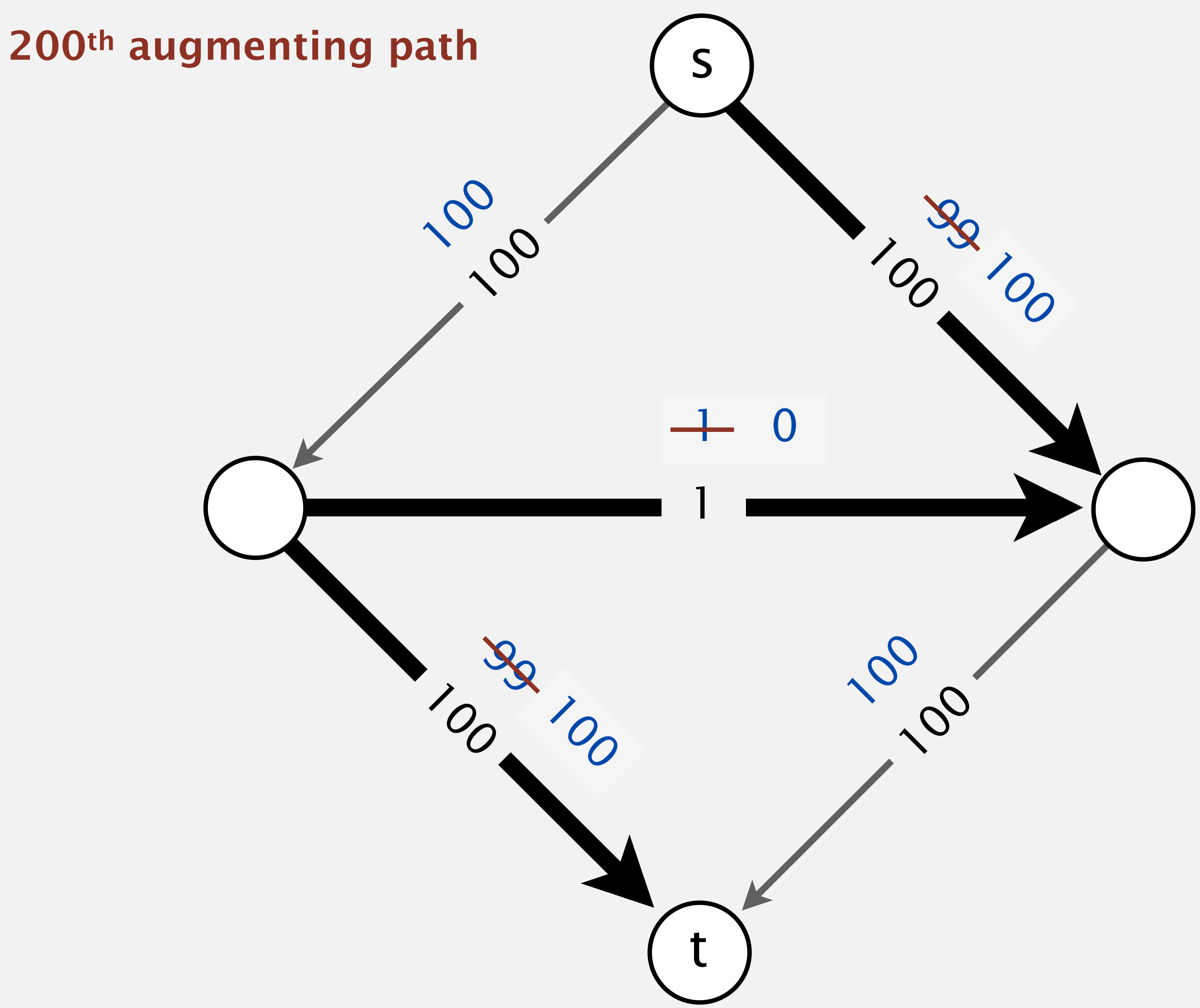
Bad case for Ford–Fulkerson

Bad news. Even when edge capacities are integers, number of augmenting paths could be very large.



Bad case for Ford–Fulkerson

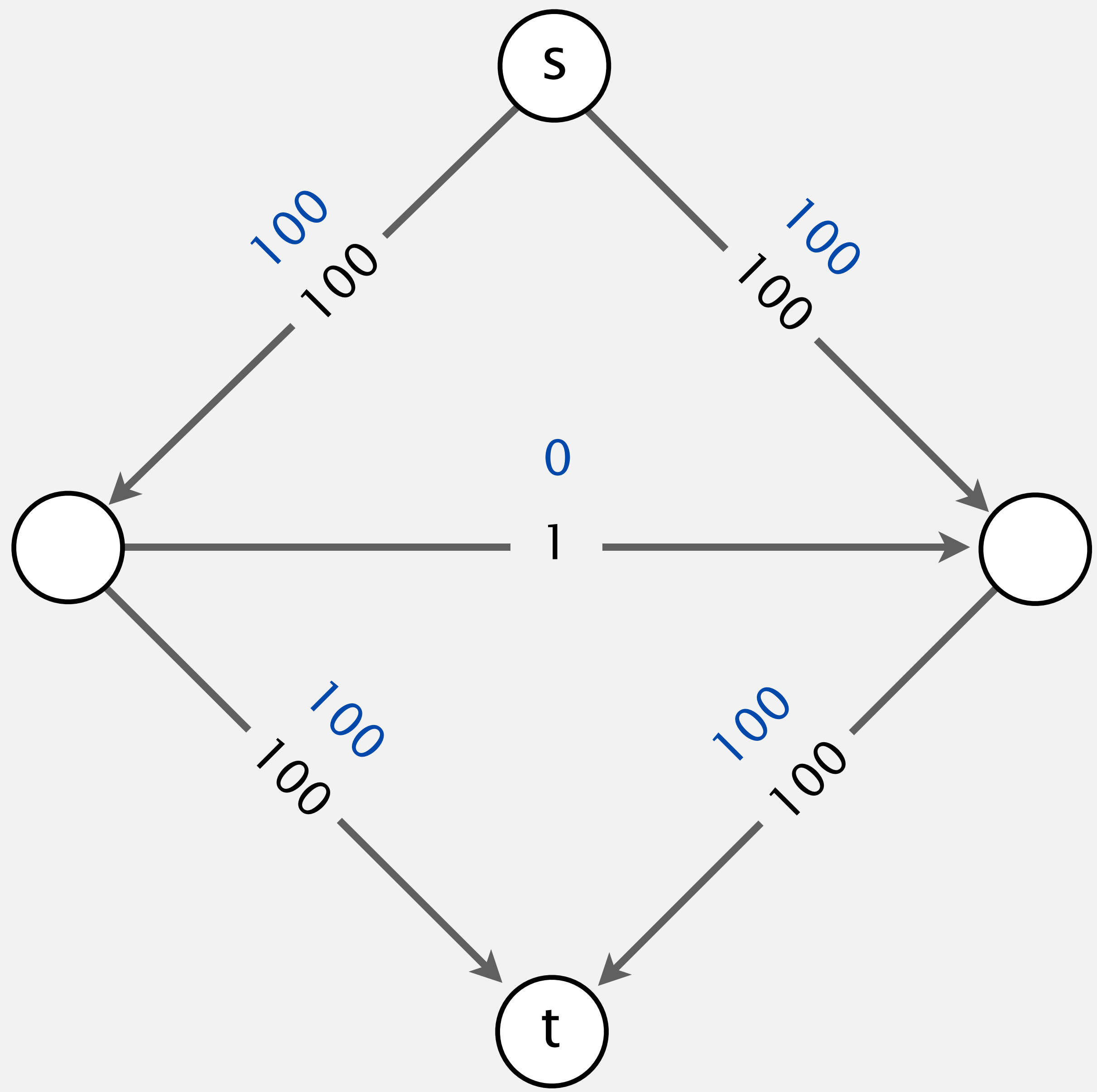
Bad news. Even when edge capacities are integers, number of augmenting paths could be very large.



Bad case for Ford–Fulkerson

Bad news. Even when edge capacities are integers, number of augmenting paths could be very large.

exponential in input size



How to choose augmenting paths?

Good news. Clever choices lead to efficient algorithms.

augmenting path	number of paths	implementation
shortest path (fewest edges)	$\leq \frac{1}{2} E V$	queue (BFS)
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

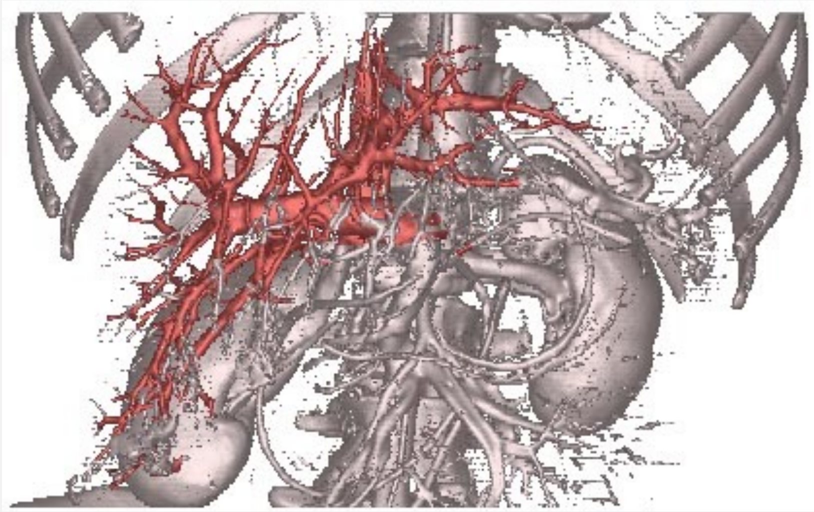
6.4 Maximum Flow

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

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

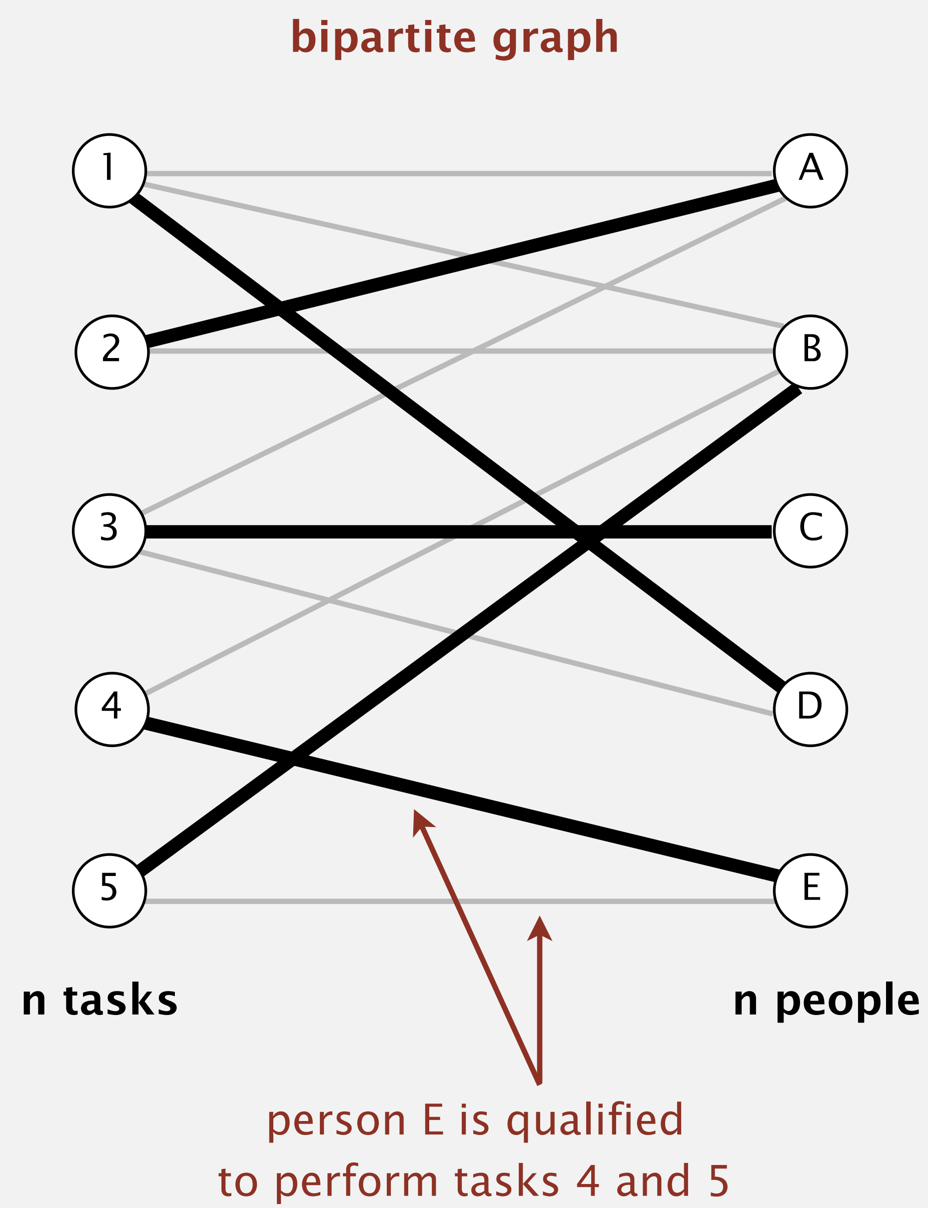
Bipartite matching problem

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).

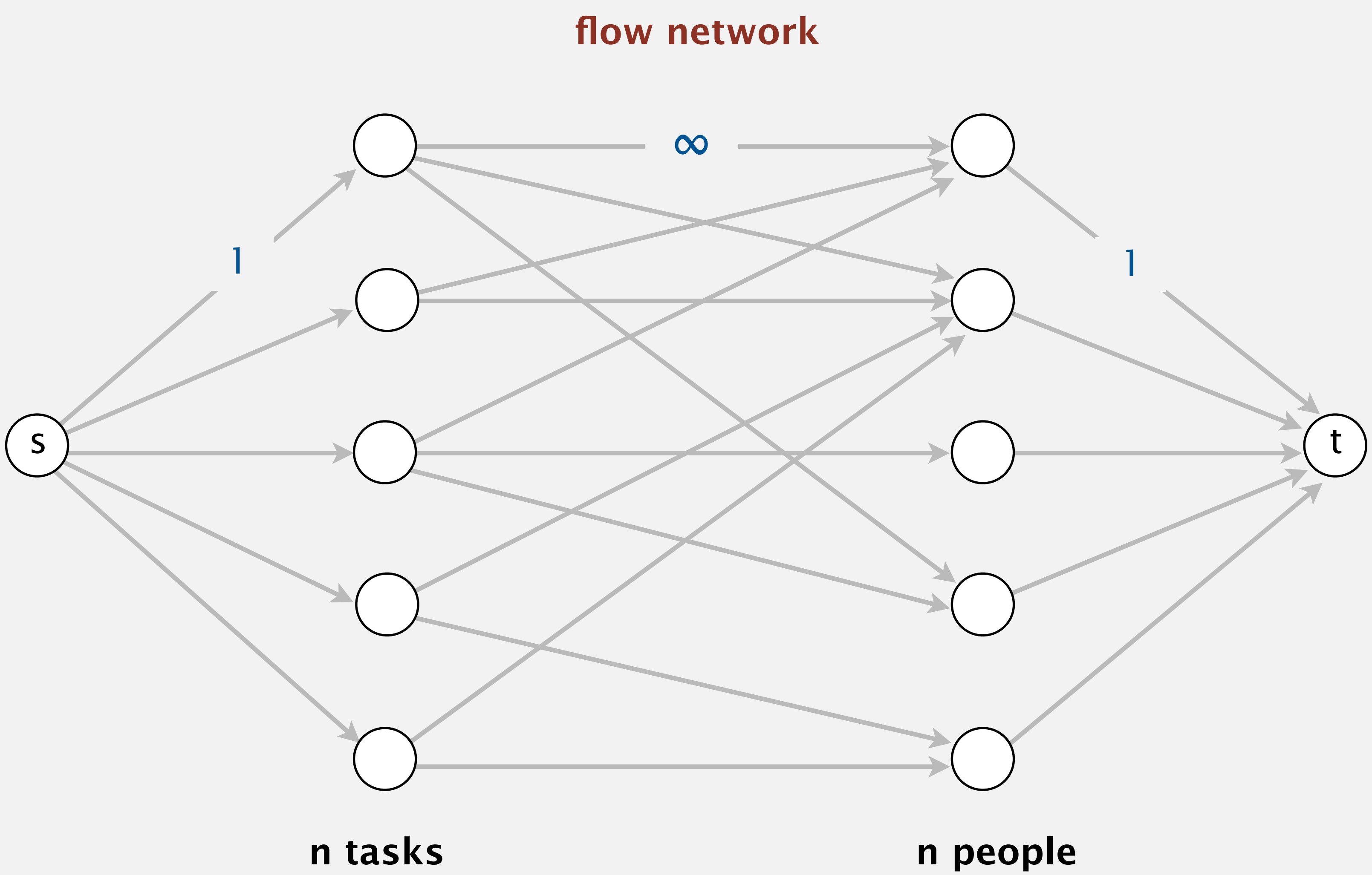


perfect matching

1–D
2–A
3–C
4–E
5–B

Maxflow formulation of bipartite matching

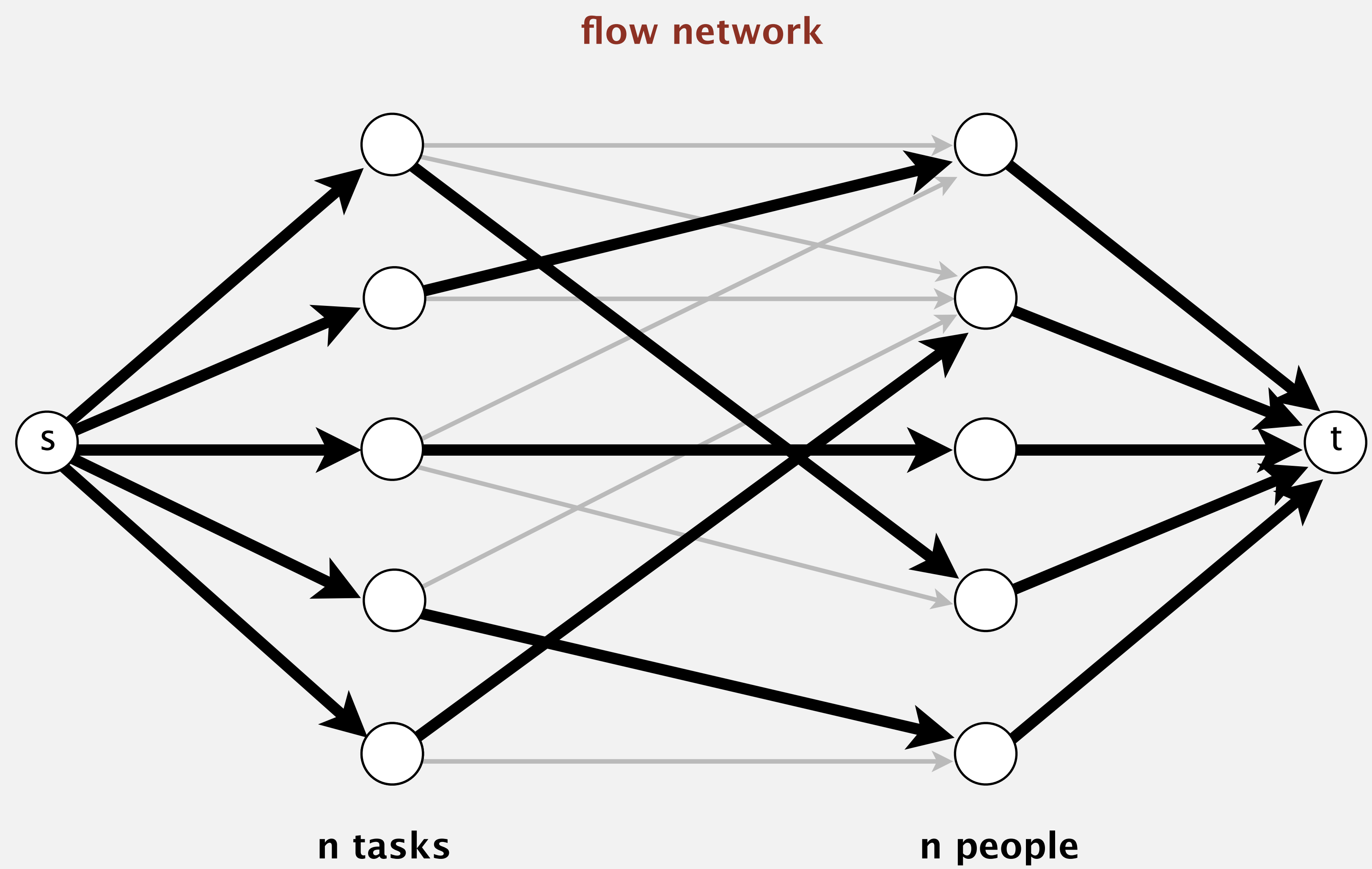
- Create s, t, one vertex for each task, and one vertex for each person.
- Add edge from s to each task (of capacity 1).
- Add edge from each person to t (of capacity 1).
- Add edge from task to qualified person (of infinite capacity).



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.



How many augmentations does the Ford–Fulkerson algorithms make to find a perfect matching in a bipartite graph with n vertices per side?

- **A.** n
- **B.** n^2
- **C.** n^3
- **D.** n^4

Maximum flow algorithms: theory

(Yet another) holy grail for theoretical computer scientists.

year	method	worst case	discovered by
1951	**simplex**	$E^3 U$	Dantzig
1955	**augmenting path**	$E^2 U$	Ford–Fulkerson
1970	**shortest augmenting path**	E^3	Dinitz, Edmonds–Karp
1970	**fattest augmenting path**	$E^2 \log E \log(E U)$	Dinitz, Edmonds–Karp
1977	**blocking flow**	$E^{5/2}$	Cherkasky
1978	**blocking flow**	$E^{7/3}$	Galil
1983	**dynamic trees**	$E^2 \log E$	Sleator–Tarjan
1985	**capacity scaling**	$E^2 \log U$	Gabow
1997	**length function**	$E^{3/2} \log E \log U$	Goldberg–Rao
2012	**compact network**	$E^2 / \log E$	Orlin
?	**?**	E	?

maxflow algorithms for sparse networks with E edges, 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.

Best in practice. Push–relabel method with gap relabeling: $E^{3/2}$.

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 an *st*-cut of minimum capacity.

Maxflow problem. Find an *st*-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).

Open research challenges.

- Practice: solve real-world maxflow/mincut problems in linear time.
- Theory: prove it for worst-case inputs.
- Still much to be learned!