CSC 226

Algorithms and Data Structures: II

Rich Little
rlittle@uvic.ca

Maxflow-Mincut

st-cuts

- Recall: A *cut* in a (directed) graph is a partition of the vertices into two disjoint subsets.
- The *cut edges* of a graph with a cut are the edges that have one endpoint in each subset of the partition.
- An st-*cut*, χ, is a cut that places vertex s in one of its subsets, S, and vertex t in the other, T.

st-cuts (continued)

- *Capacity* of an st-cut, $c(\chi)$, in an st-network is the sum of the capacities of the cut's edges from S to T
  - $c(\chi) = \sum_{e \in S \to T} c(e)$
- *Flow across* an st-cut, $f(\chi)$, in an st-network: the sum of the flows of cut's edges from S to T minus the sum of the flows of cut's edges from T to the S
  - $f(\chi) = f(S \to T) - f(T \to S)$

minimum *st*-cut problem (or *mincut* problem)

- Given an *st*-network, find an *st*-cut such that the capacity of no other cut is smaller.

Properties of feasible *st*-flows in *st*-flow networks

1. For any *st*-flow, f, the flow across each *st*-cut, χ, is equal to the value of the flow, i.e. $|f| = f(\chi)$

2. The outflow from s is equal to the inflow to t

3. No *st*-flow's value can exceed the capacity of any *st*-cut, i.e. $|f| \leq c(\chi)$

4. Let f be an *st*-flow and let χ be an *st*-cut such that $|f| = c(\chi)$. Then f is a maximum flow and χ is a minimum cut.

Maxflow-Mincut Theorem

- Let f be an st-flowvfor graph $G = (V, E)$. Then, the following three conditions are equivalent:

  A. there exists an st-cut whose capacity equals $|f|$

  B. f is a maximum flow

  C. there is no augmenting path with respect to f

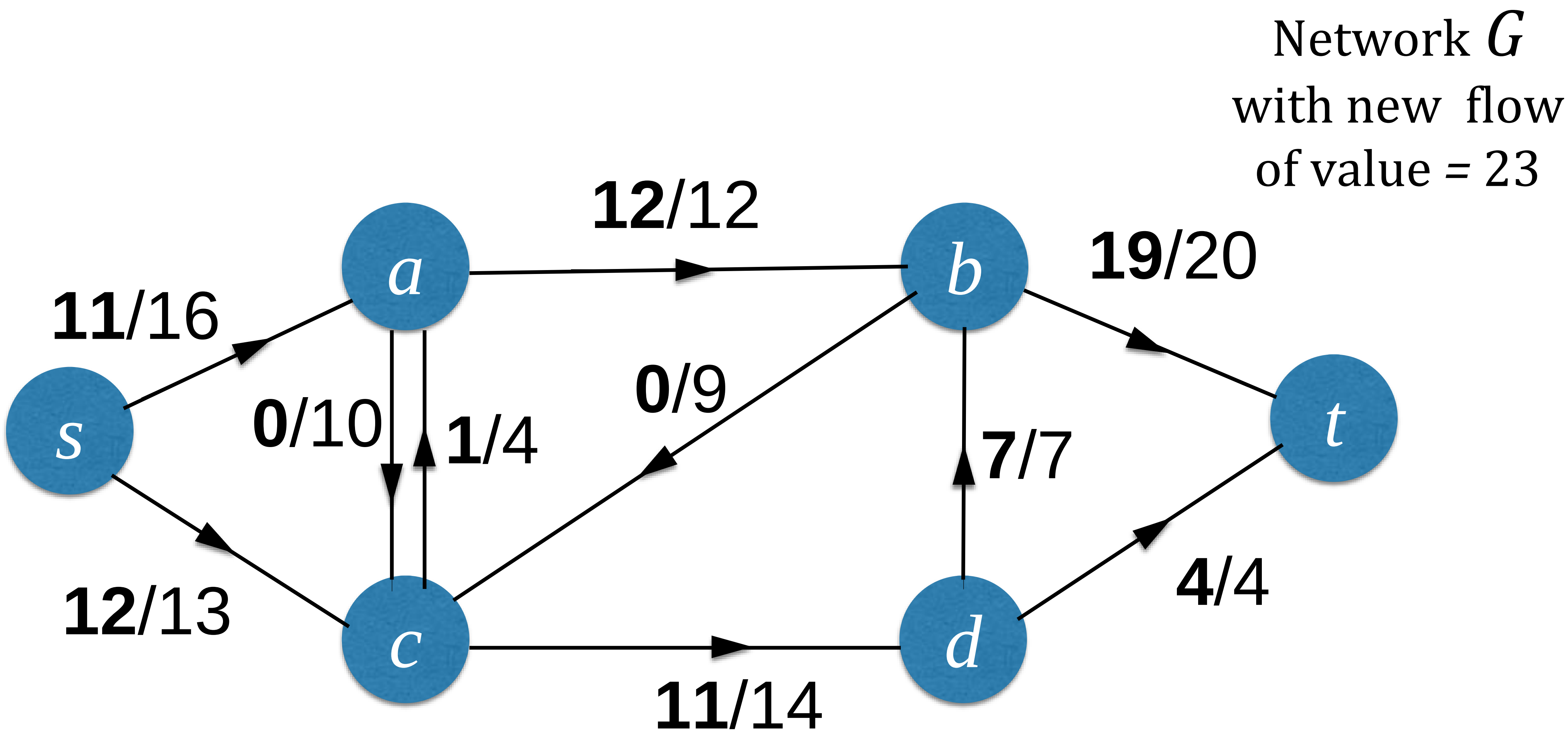
Maxflow-Mincut Proof

- Let f be an st-flow for graph $G = (V, E)$.
  - **<u>A ⇒ B:</u>** Let χ be an *st*-cut such that $c(\chi) = |f|$. We know that for any cut, the flow value is equal to the flow across the cut. So, this implies $f(\chi) = |f|$.
  - We also know that $c(\chi) \geq |f|$ for any cut. This implies that the maximum that can cross cut χ is $c(\chi)$. But $c(\chi) = |f|$ by A so f is the maxflow.
  - **<u>B ⇒ C:</u>** Let f be a maxflow for G. Assume that there is an augmenting path in the residual graph G_f. Increase the flow value by the bottleneck capacity on the augmenting path, getting a new flow value greater than $|f|$ the maxflow value.
  - Contradiction. Thus, there is no augmenting path with respect to f.

Maxflow-Mincut Proof

- Let f be an st-flow for graph $G = (V, E)$.
  - **<u>C ⇒ A:</u>** Assume that there are no augmenting paths with respect to flow f. This means that there is no directed path in the residual graph, G_f, from s to t, (i.e. t is not reachable from s.)
  - Let $S = \{v \in V | v \text{ is reachable from } s \text{ in } G_f\}$ and let $T = V - S$. Note, $s \in S$ and $t \in T$ (why?). Let $\chi = (S, T)$ be an *st*-cut of graph G.
  - By construction of χ, for every edge $(u, v) \in E$, such that $u \in S$ and $v \in T$, we know that $c\big((u, v)\big) - f\big((u, v)\big) = 0$ and $f\big((v, u)\big) = 0$, (since no augmenting paths across this cut).
  - Thus, $|f| = f(\chi) = f(S, T) - f(T, S) = c(S, T) = c(\chi)$. ∎

Claim: the new flow is a maxflow.



What is the cut S,T of minimum capacity?

$S = \{\ ??\ \}$ $T = \{\ ??\ \}$

Check:
Are arcs leaving the S part full?
Are the arcs returning from the T part to S part empty?

Properties of the residual network G_f

- $|E_f| \leq 2|E|$
- The residual network G_f with capacities c_f of st-flow network G is an st-flow network

Definition of Augmenting Path

Given an st-flow f in st-flow network $G = (V, E)$ an augmenting path p is a directed path from s to t in the residual network G_f.

Pseudocode for Algorithm Ford-Fulkerson(G, s, t)

Initialize f as zero-flow

Compute residual network G_f

while there exists a path p from s to t in G_f **do**

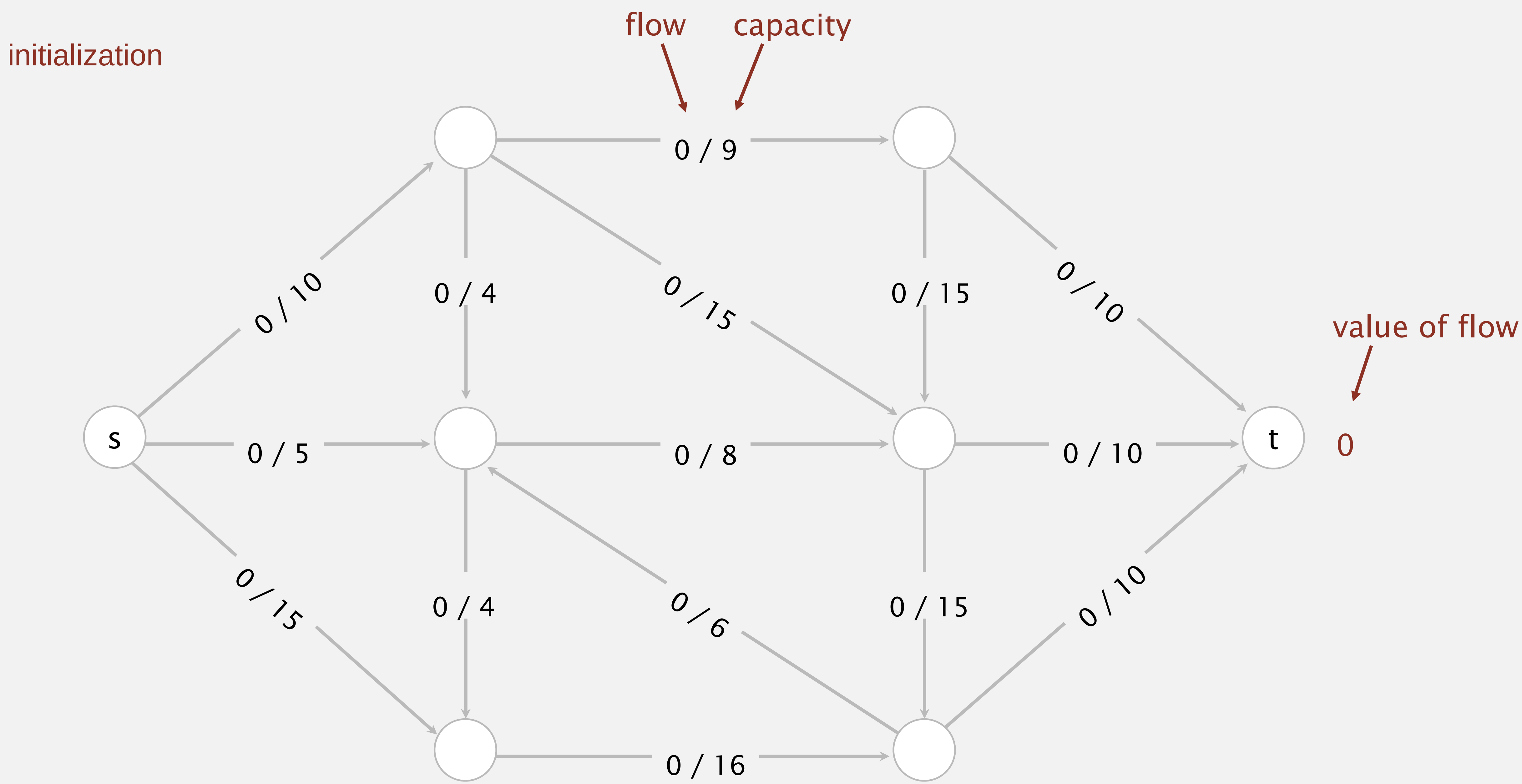
  Augment f using p

  Update G_f

return f

Ford-Fulkerson algorithm for solving MaxFlow

Initialization. Start with 0 flow.

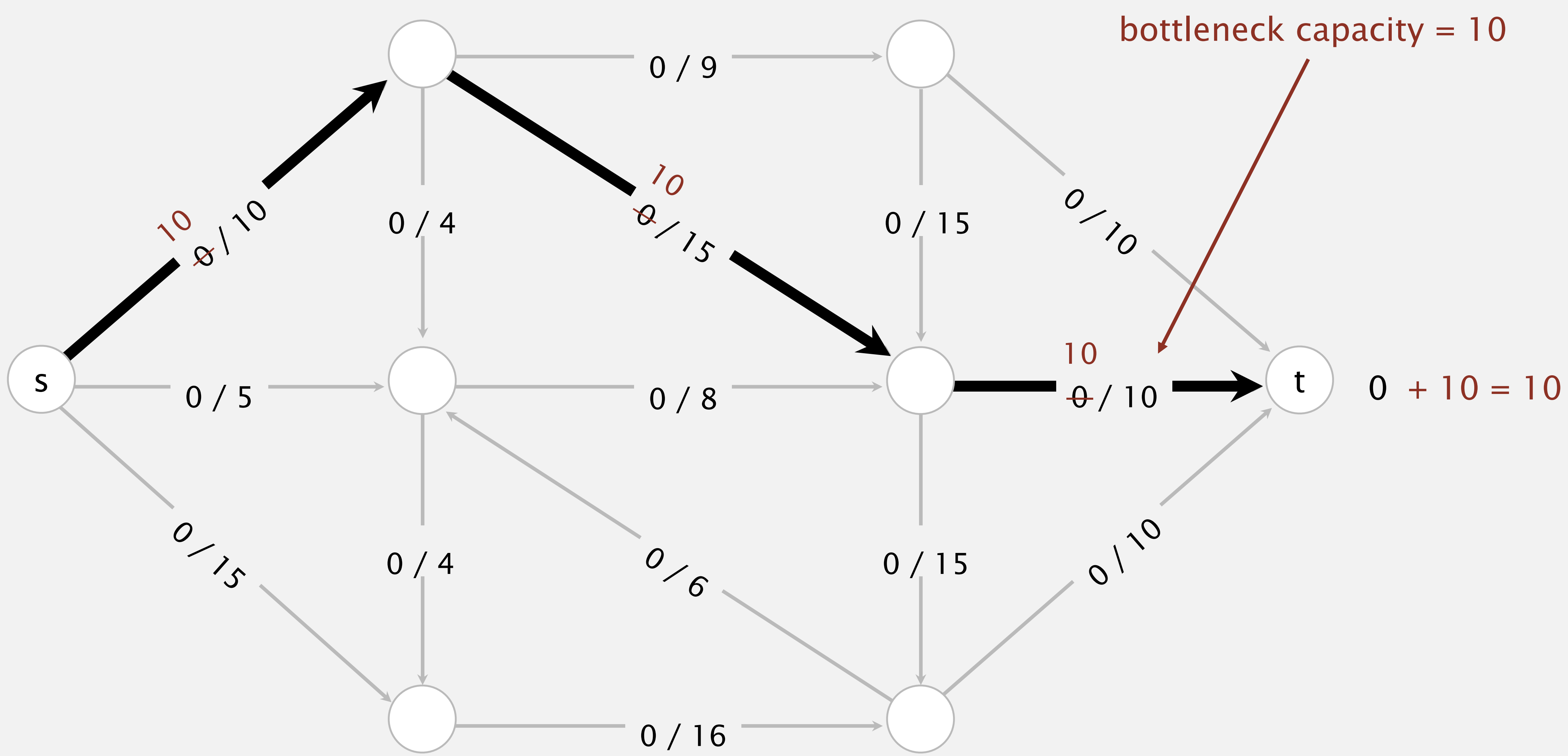


Idea: increase flow along augmenting paths

Definition: Augmenting path -- 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).

1st augmenting path

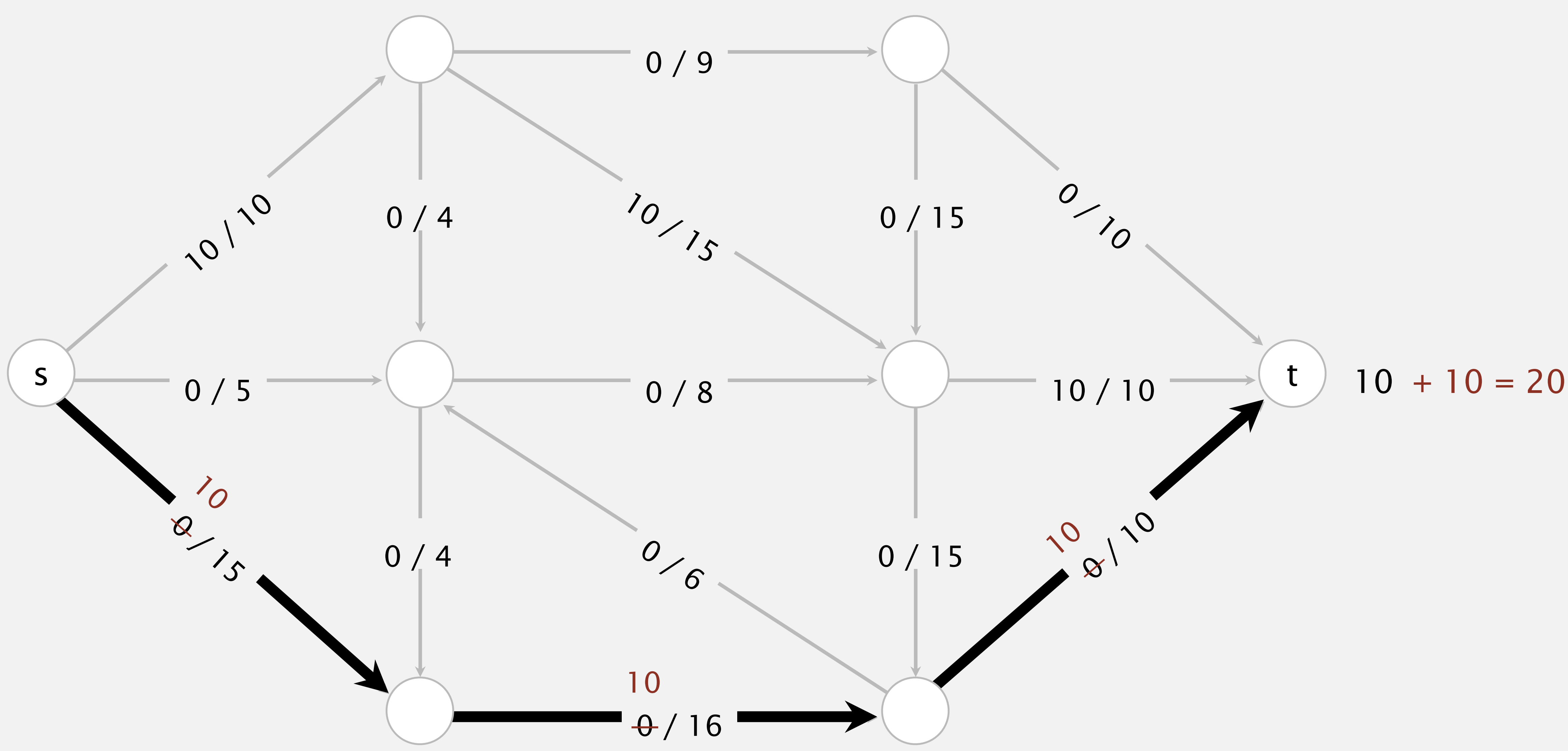


Idea: increase flow along augmenting paths

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).

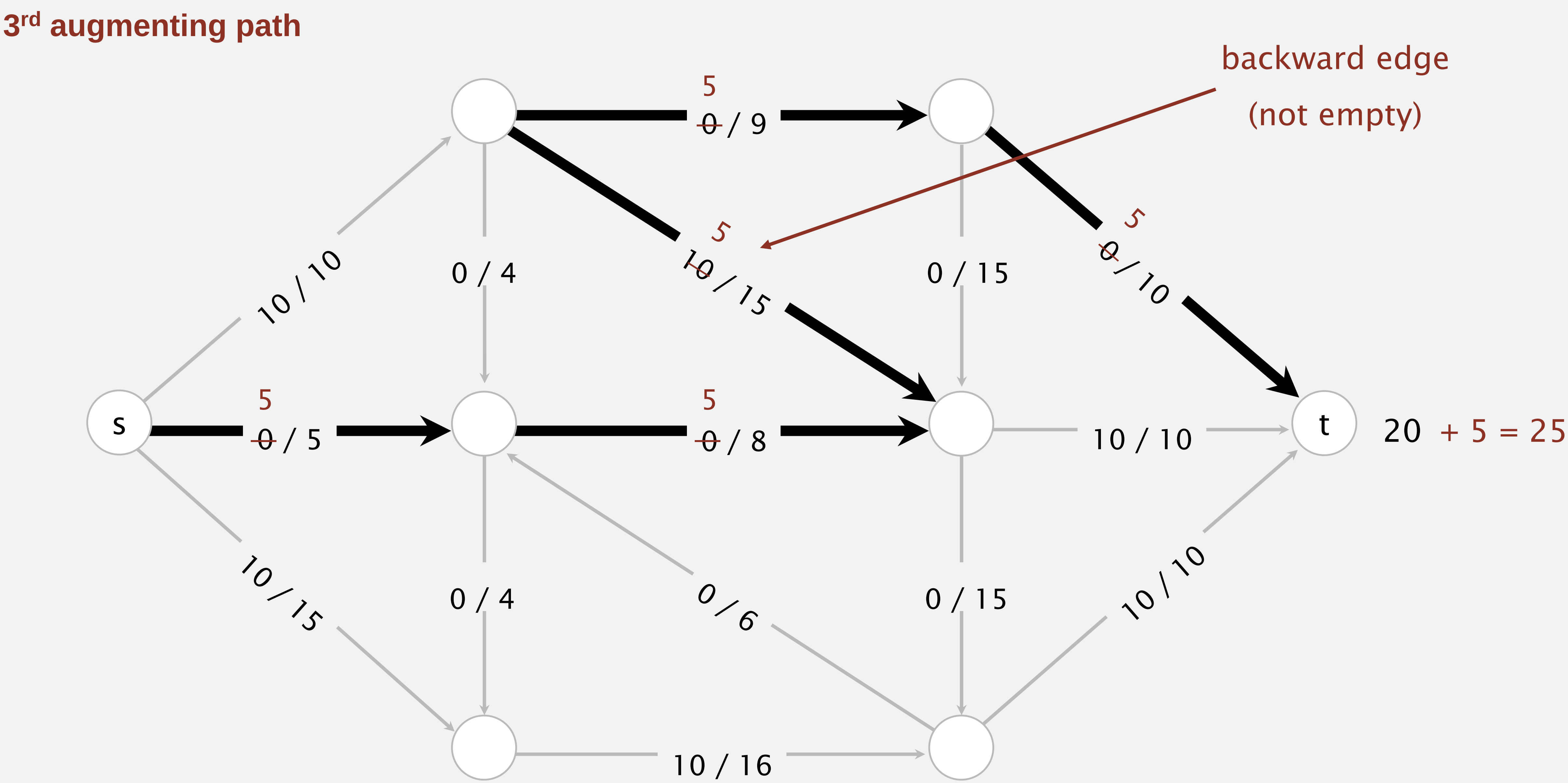
2nd augmenting path



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).

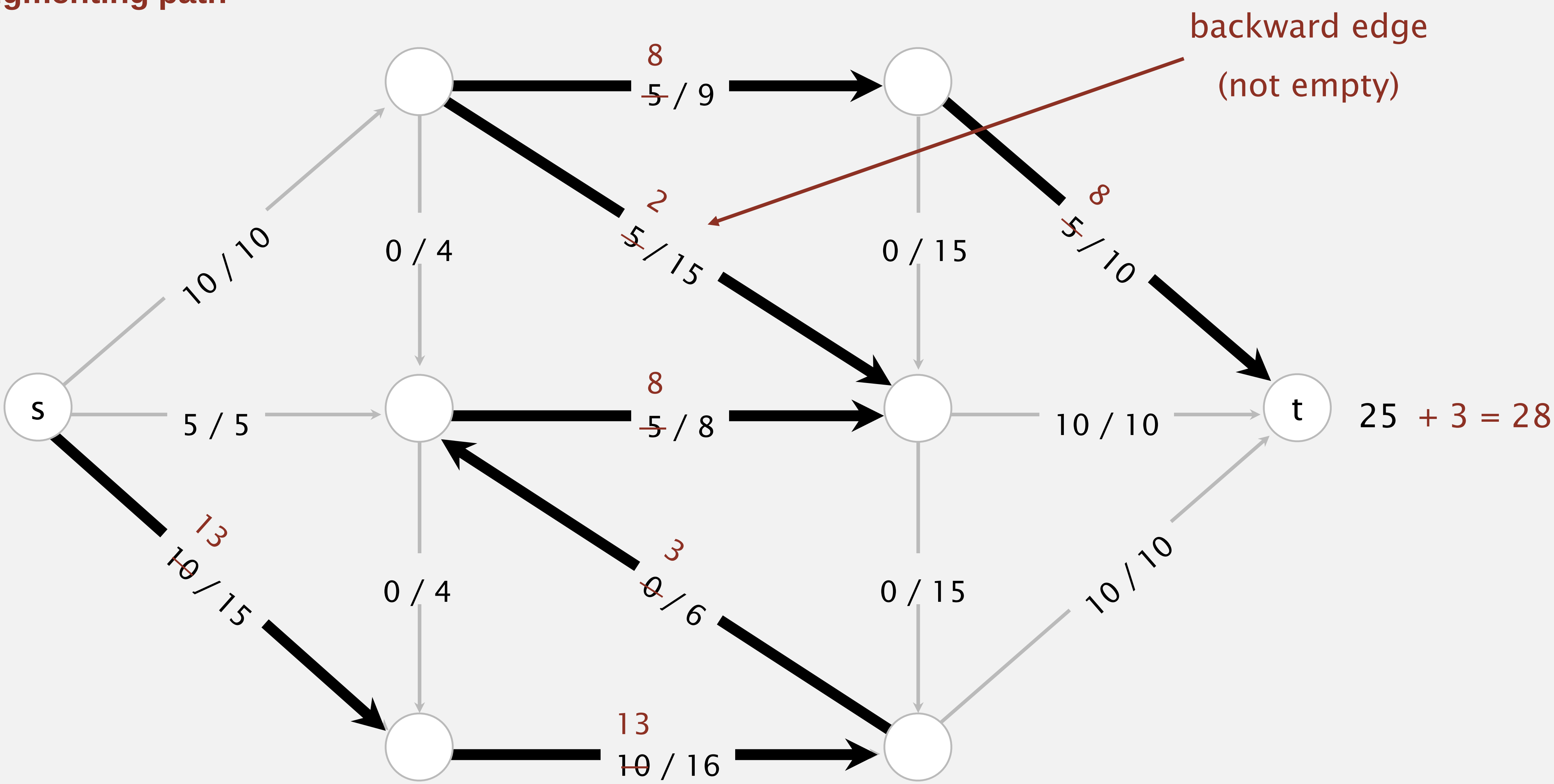


Idea: increase flow along augmenting paths

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).

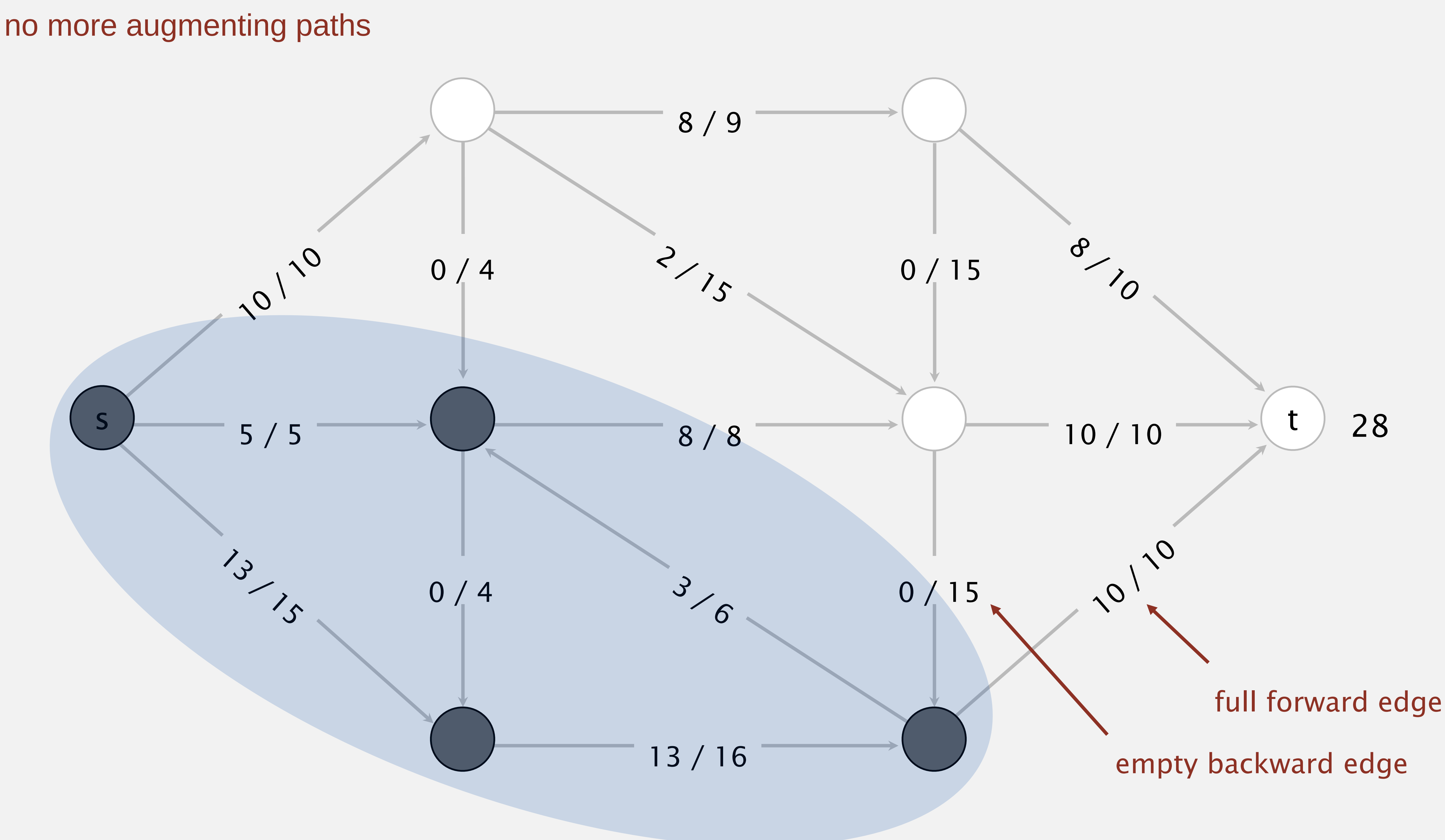
4th augmenting path



Idea: increase flow along augmenting paths

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

- Full forward edge.
- Empty backward edge.



Running time of Ford-Fulkerson

- Building the residual network
- Finding an augmenting path in the residual network
- How many augmenting paths can be found in the worst case, if the capacities are positive integers?
  - value of maximum flow many (no more since the augmenting path has at least residual capacity 1)

EdmondsKarp(G, s, t)

Initialize f as zero-flow and residual network G_f with G
while there exists a path p from s to t in G_f **do**

Let p be a shortest path from s to t in G_f

Augment f using p

Update G_f

return f

Running Time Analysis of Algorithm by Edmonds & Karp

- Overall running time when using BFS (breadth first search) for determining augmenting path: $O(nm^2)$ $(m = |E|, n = |V|)$
  - An edge on an augmenting path in G_f is called *a bottleneck* if its capacity is equal to the path's residual capacity
  - Fact: an edge in G_f can be a bottleneck at most $O(n)$ times
  - The while loop therefore will not run more than $O(nm)$ times
  - The while loop's running time is $O(m)$