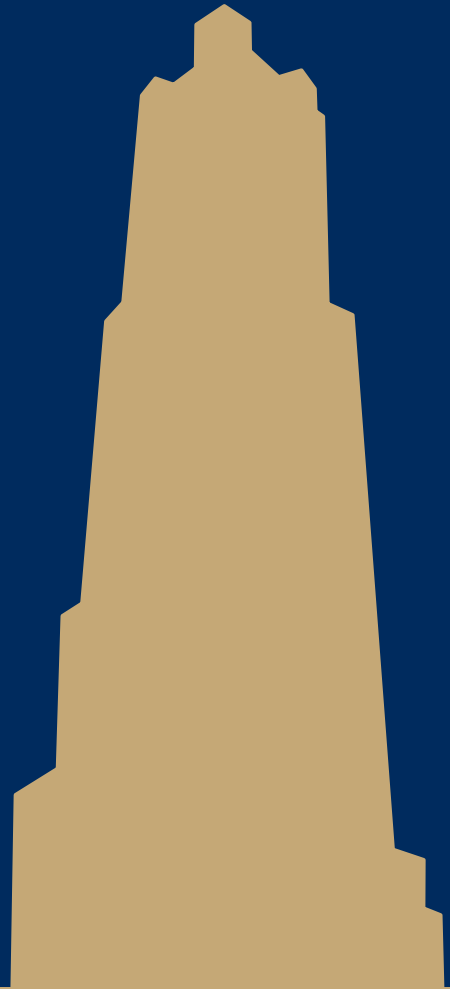


# CS/COE 1501

[www.cs.pitt.edu/~lipschultz/cs1501/](http://www.cs.pitt.edu/~lipschultz/cs1501/)

## Network Flow



# Defining network flow

- Consider a directed, weighted graph  $G(V, E)$ 
  - Weights are applied to edges to state their *capacity*
    - $c(v, w)$  is the capacity of an edge from vertex  $v$  to vertex  $w$
    - if there is no edge from  $v$  to  $w$ ,  $c(v, w) = 0$
- Consider two nodes, a *source*  $s$  and a *sink*  $t$ 
  - Let's determine the maximum flow that can run from  $s$  to  $t$  in the graph  $G$

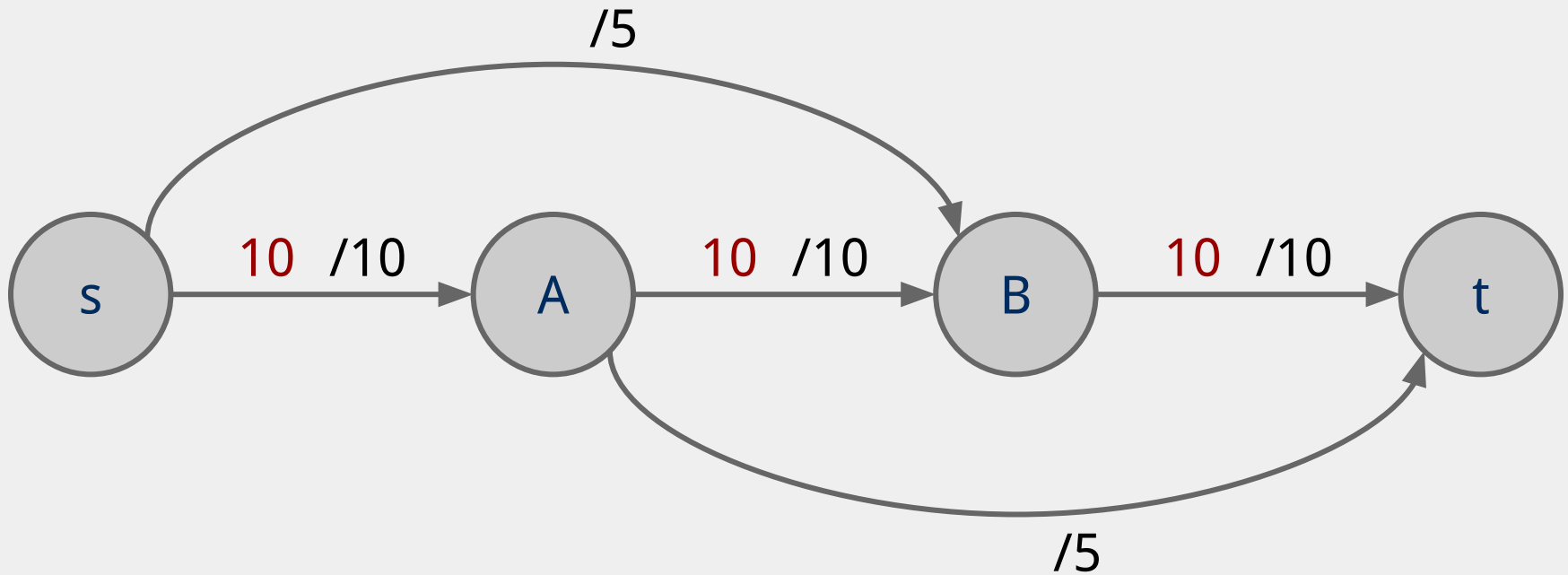
# Flow

- Let the  $f(v, w)$  be the amount of flow carried along an edge from vertex  $v$  to vertex  $w$
- Some rules on the flow running through an edge:
  - $\forall (v, w) \in E \quad f(v, w) \leq c(v, w)$
  - $\forall (v, w) \in E \quad f(v, w) = -f(w, v)$
  - $\forall v \in (V - \{s, t\}) \quad \sum_{w \in V} f(v, w) = 0$

# Ford Fulkerson

- Let all edges in  $G$  have a flow of 0
- While there is path  $p$  from  $s$  to  $t$  in  $G$  such that all edges in  $p$  have some residual capacity:
  - Find the edge with the minimum residual capacity in  $p$ 
    - We'll call this residual capacity *new\_flow*
  - Increment the flow on all edges in  $p$  by *new\_flow*

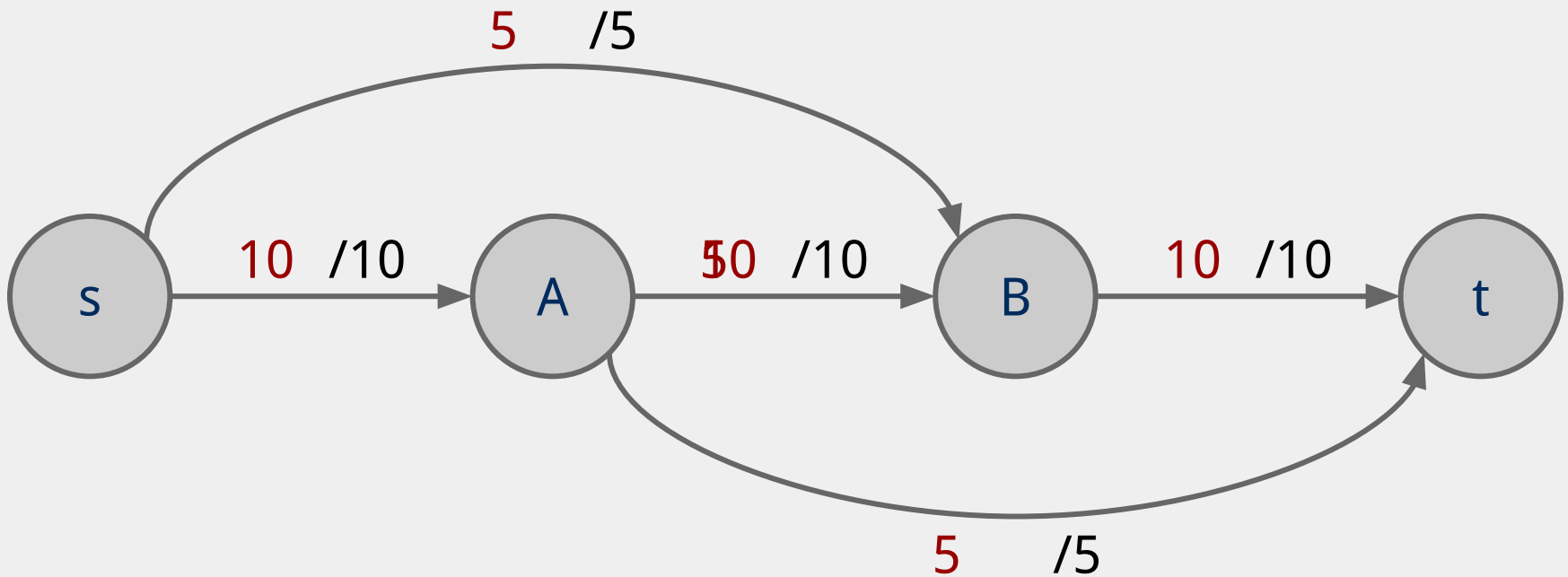
# Ford Fulkerson example



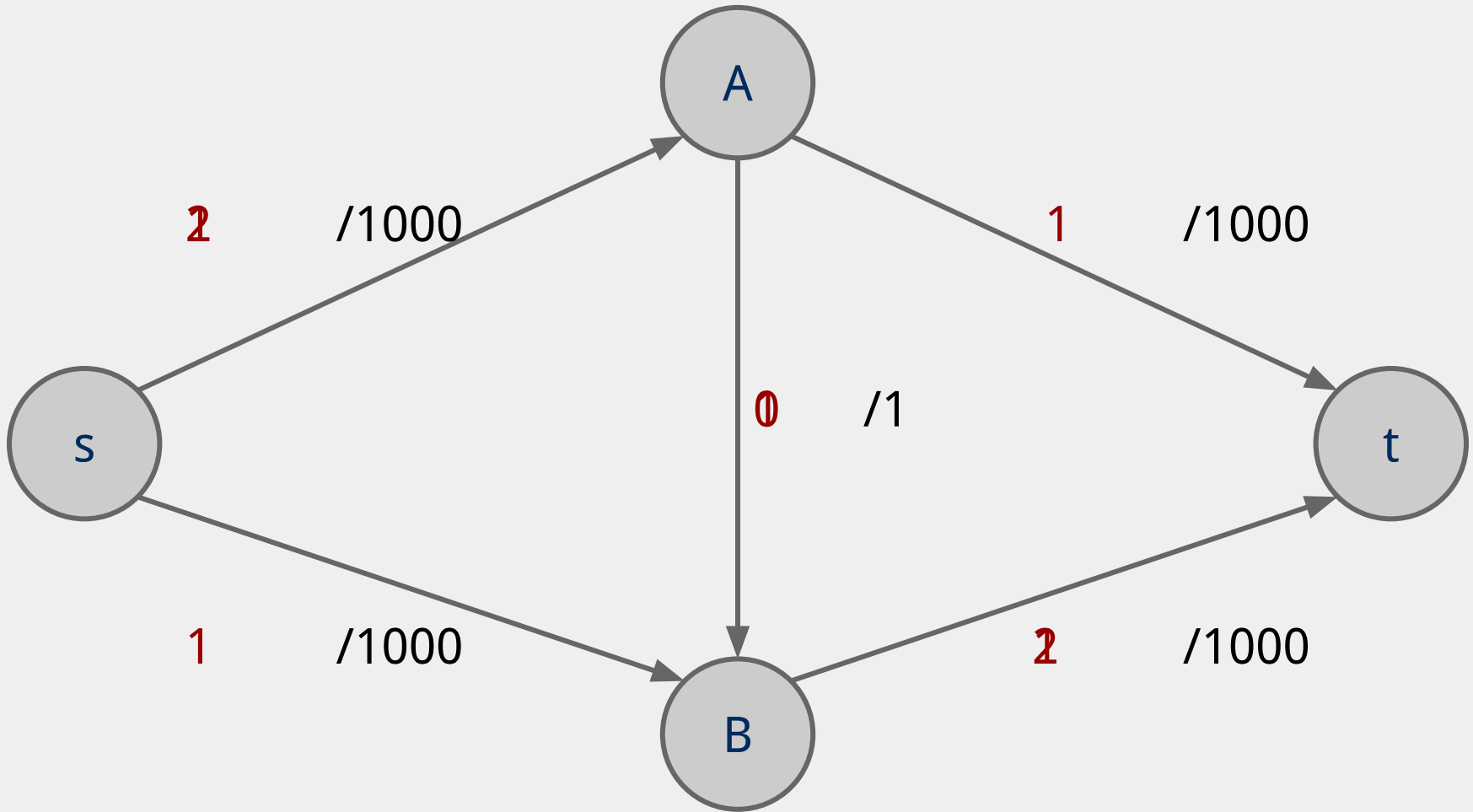
# Expanding on residual capacity

- For an edge  $(v, w)$  in  $E$ :
  - Edge has residual capacity of  $c(v, w) - f(v, w)$
  - To find the max flow, however, we also need to consider re-routing flow we had previously allocated
    - This means, for each edge  $(v, w)$  in  $E$ , we also need to consider the residual capacity of *backwards edge*  $(w, v)$ 
      - Which is  $f(v, w)$

# Ford Fulkerson example



# Another example

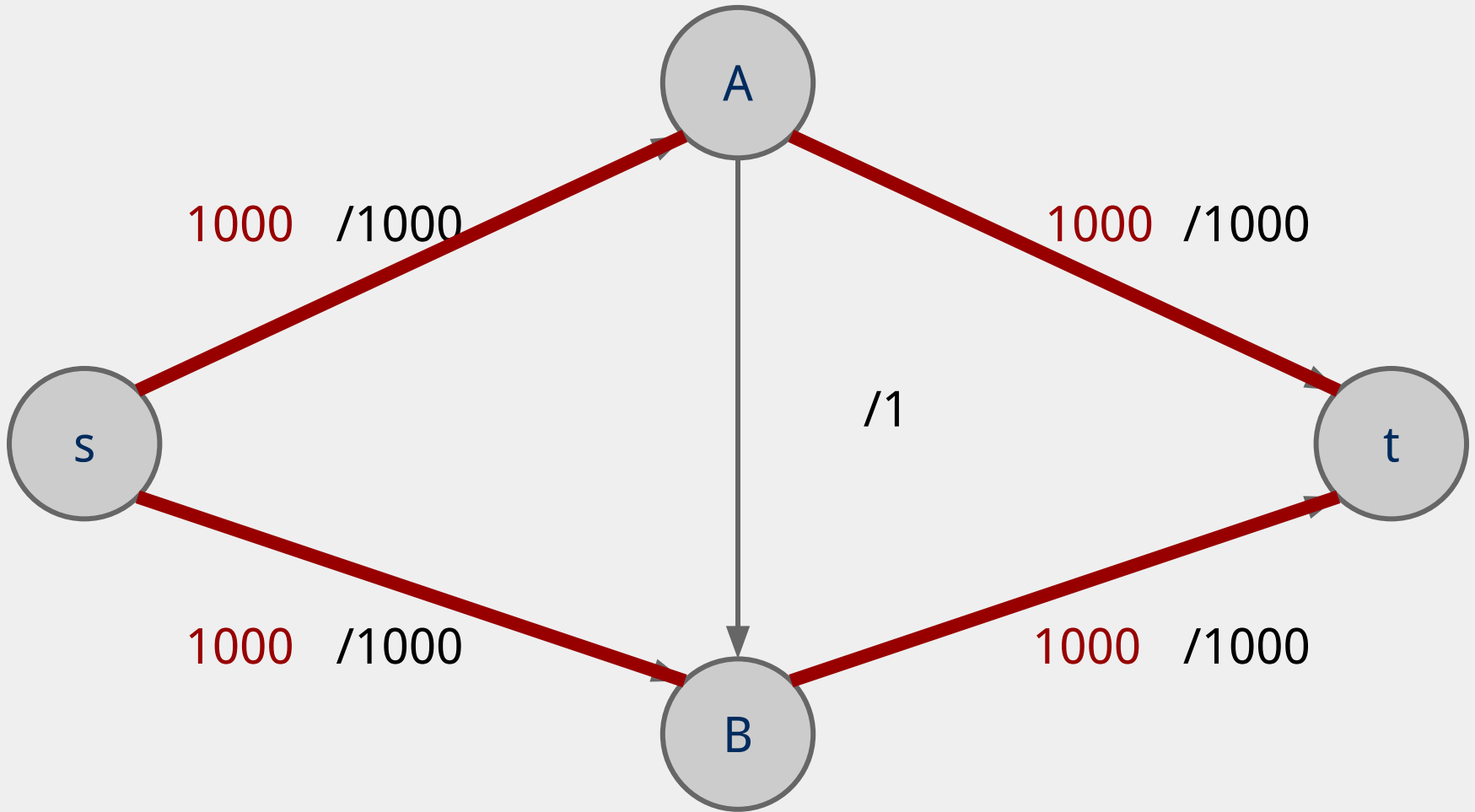




# Edmonds Karp

- How the augmenting path is chosen affects the performance of the search for max flow
- Edmonds and Karp proposed a shortest path heuristic for Ford Fulkerson
  - Use BFS to find augmenting paths
- See revised example

# Another example



# But our flow graph is weighted...

- Edmonds-Karp only uses BFS
  - Used to find spanning trees and shortest paths for unweighted graphs
  - Why do we not use some measure of priority to find augmenting paths?

# Implementation concerns

- Representing the graph:
  - Similar to a directed graph
  - Can store an adjacency list of directed edges
    - Actually, more than simply directed edges
      - Flow edges

# Flow edge implementation

- For each edge, we need to store:
  - Start point, the *from* vertex
  - End point, the *to* vertex
  - Capacity
  - Flow
  - Residual capacities
    - For forwards and backwards edges

# FlowEdge.java

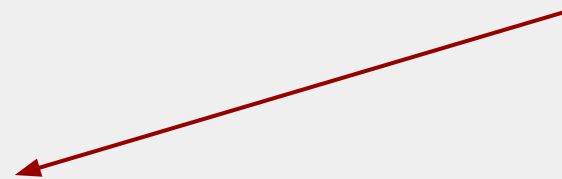
```
public class FlowEdge {
    private final int v;           // from
    private final int w;           // to
    private final double capacity; // capacity
    private double flow;           // flow
    ...
    public double residualCapacityTo(int vertex) {
        if (vertex == v) return flow;
        else if (vertex == w) return capacity - flow;
        else throw new
            IllegalArgumentException("Illegal endpoint");
    }
    ...
}
```

# BFS search for an augmenting path (pseudocode)

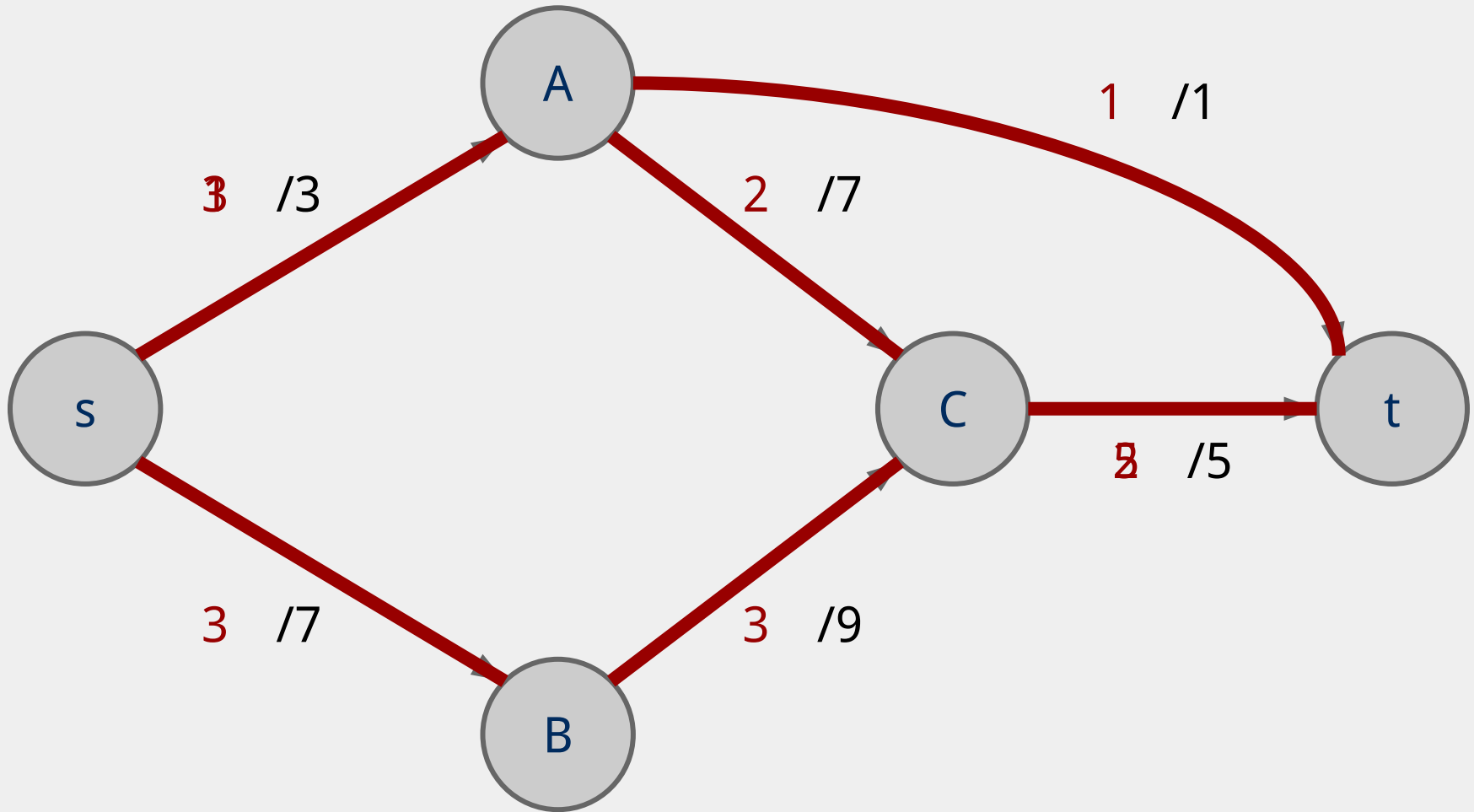
```
edgeTo = [|V|]
marked = [|V|]
Queue q
q.enqueue(s)
marked[s] = true
while !q.isEmpty():
    v = q.dequeue()
    for each (v, w) in AdjList[v]:
        if residualCapacity(v, w) > 0:
            if !marked[w]:
                edgeTo[w] = e;
                marked[w] = true;
                q.enqueue(w);
```

Each FlowEdge object is stored  
in the adjacency list twice:

Once for its forward edge  
Once for its backwards edge



# An example to review





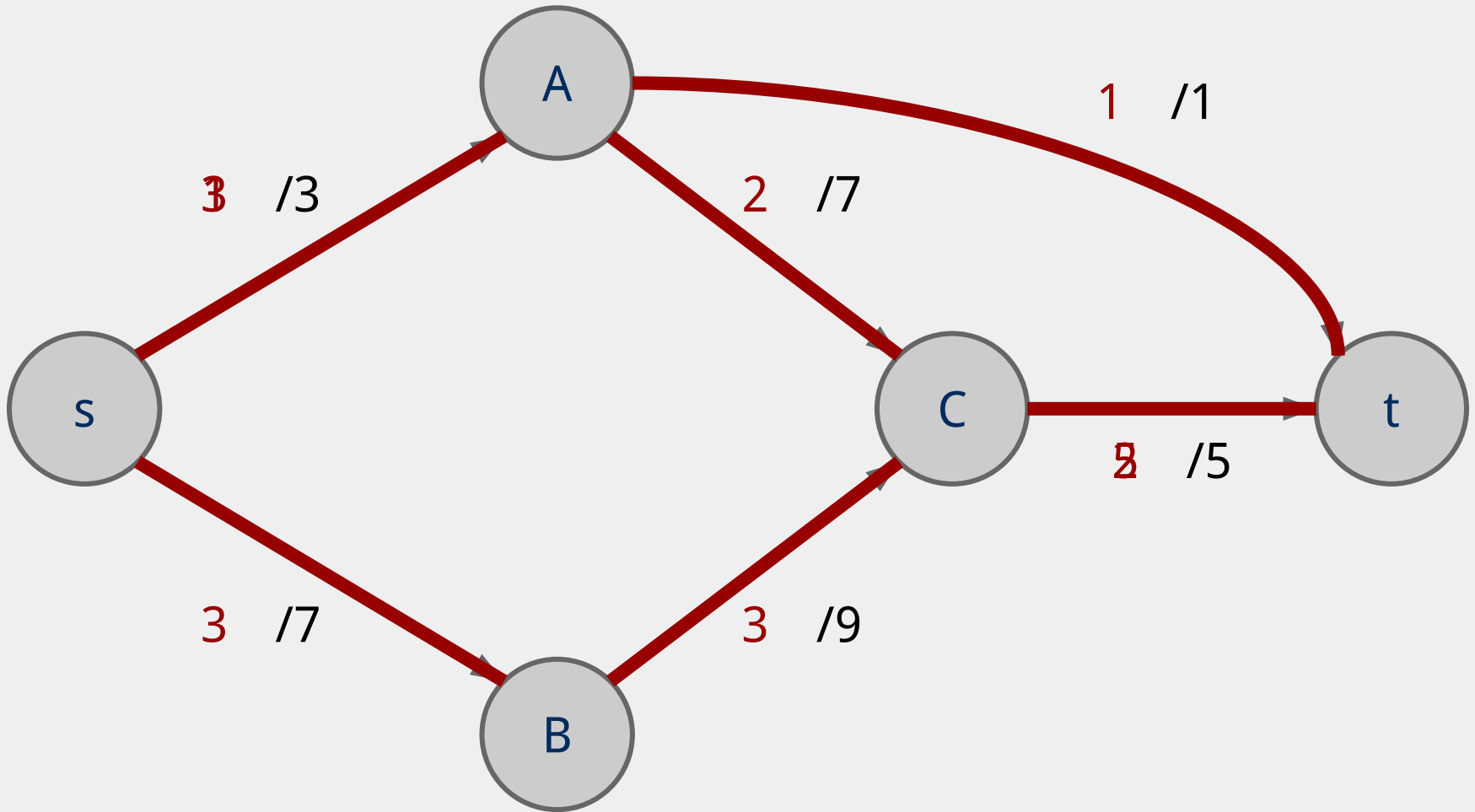
# Let's look for bottlenecks

- An st-cut on  $G$  is a set of edges in  $G$  that, if removed, will partition the vertices of  $G$  into two disjoint sets
  - One contains  $s$
  - One contains  $t$
- May be many st-cuts for a given graph
- Let's focus on finding the minimum st-cut
  - The st-cut with the smallest capacity
  - May not be unique

# How do we find the min cut?

- We could examine residual graphs
  - Specifically, try and allocate flow in the graph until we get to a residual graph with no existing augmenting paths
    - The set of saturated edges makes up the minimum st-cut

# Min cut example



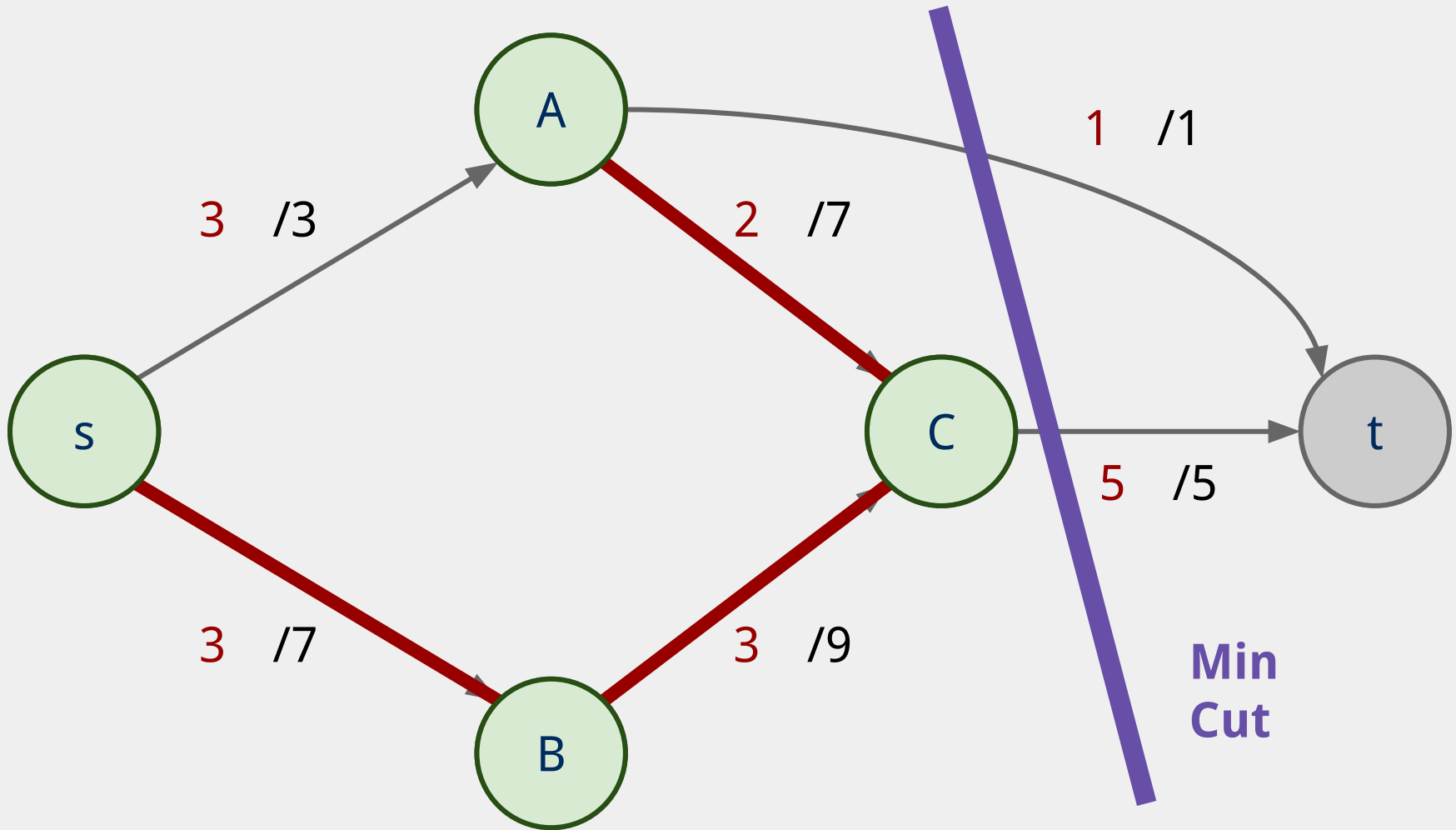
# Max flow == min cut

- A special case of duality
  - i.e., you can look at an optimization problem from two angles
    - In this case to find the maximum flow or minimum cut
  - In general, dual problems do not have to have equal solutions
    - The differences in solutions to the two ways of looking at the problem is referred to as the *duality gap*
      - If the duality gap = 0, strong duality holds
        - Max flow/min cut uphold strong duality
      - If the duality gap > 0, weak duality holds

# Determining a minimum st-cut

- First, run Ford Fulkerson to produce a residual graph with no further augmenting paths
- The last attempt to find an augmenting path will visit every node reachable from  $s$ 
  - Edges with only one endpoint in this set comprise a minimum st-cut

# Determining the min cut



# Max flow / min cut on unweighted graphs

- Is it possible?
- How would we measure the Max flow / min cut?
- What would an algorithm to solve this problem look like?

# Determining the min cut

