Algorithm Design and Analysis

CSE 565

LECTURES 16

Dynamic Programming

- Shortest Paths: Bellman-Ford
- Detecting negative cycles

Network Flow

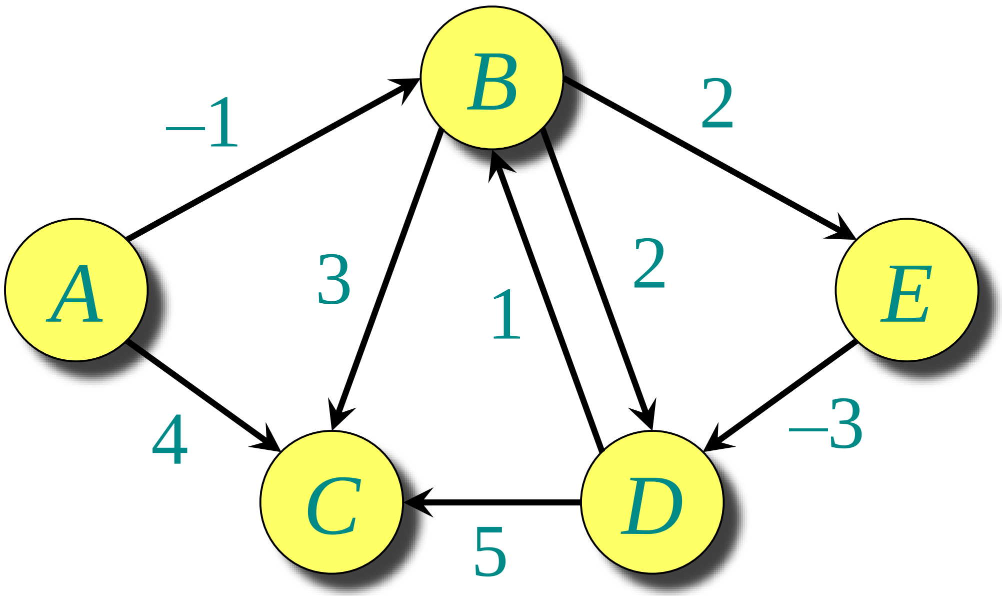
- Duality of Max Flow **and** Min Cut

Sofya Raskhodnikova

Belman-Ford: Efficient Implementation

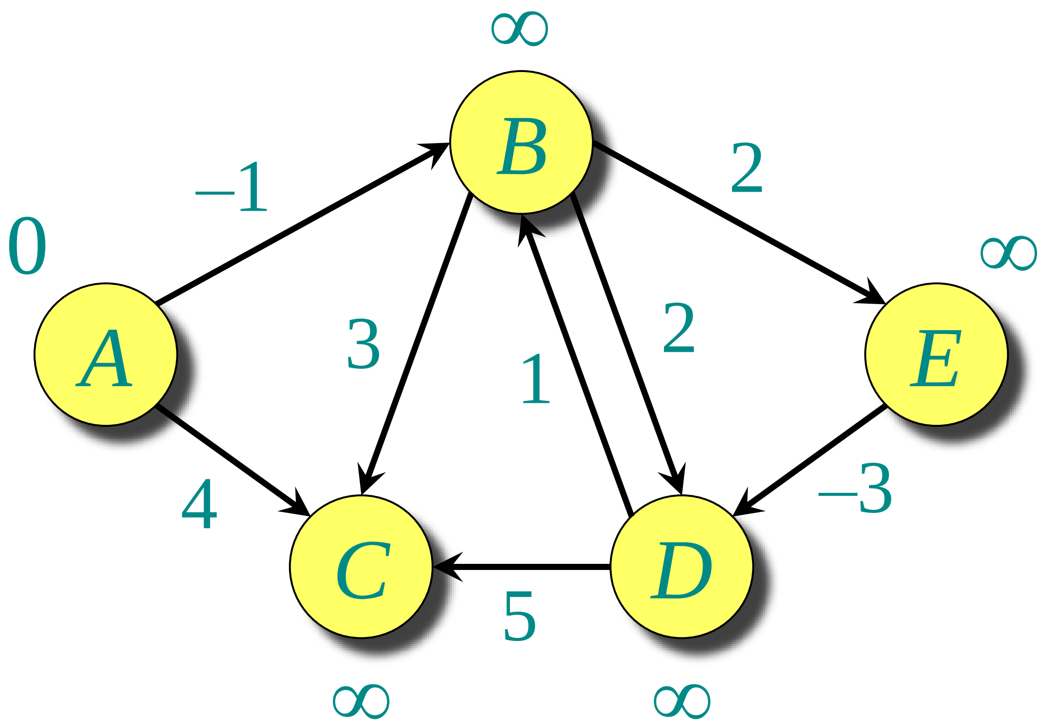
```
Bellman-Ford-Shortest-Path(G, s, t) {
   foreach node v ∈ V {
      M[v] ← ∞
      successor[v] ← ϕ
   }

   M[t] = 0
   for i = 1 to n-1 {
      foreach node w ∈ V {
      if (M[w] has been updated in previous iteration) {
         foreach node v such that (v, w) ∈ E {
            if (M[v] > M[w] + c_vw) {
               M[v] ← M[w] + c_vw
               successor[v] ← w
            }
         }
      }
      If no M[w] value changed in iteration i, stop.
   }
}
```

Example of Bellman-Ford



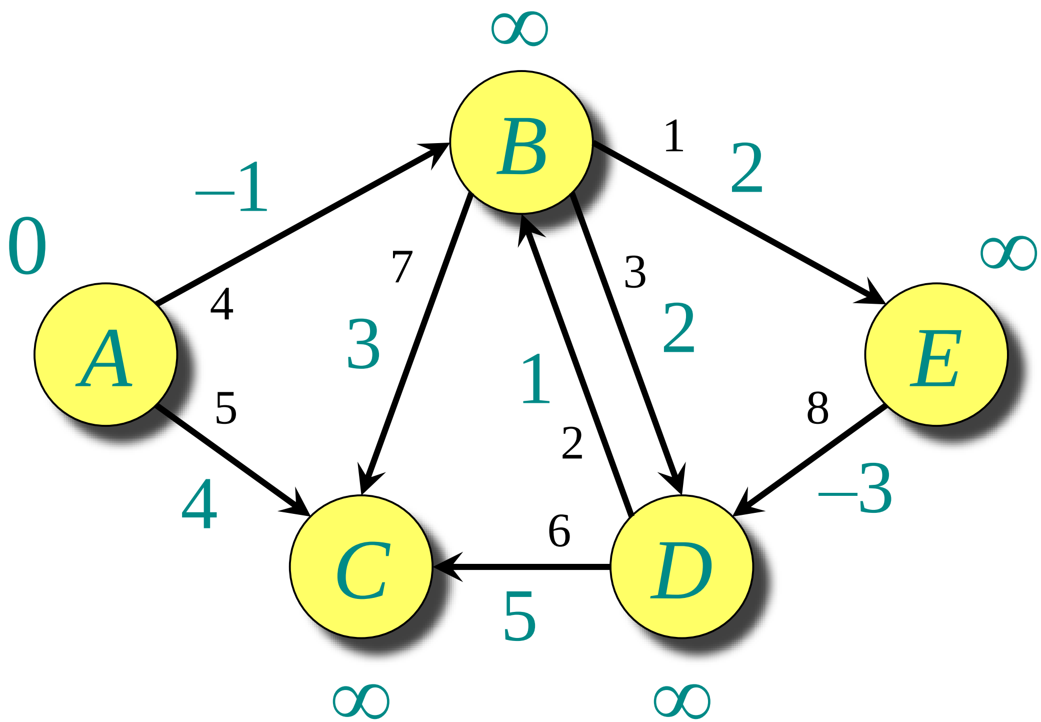
The demonstration is for a sligtly different version of the algorithm (see CLRS) that computes distances from the sourse node rather than distances to the destination node.

Example of Bellman-Ford



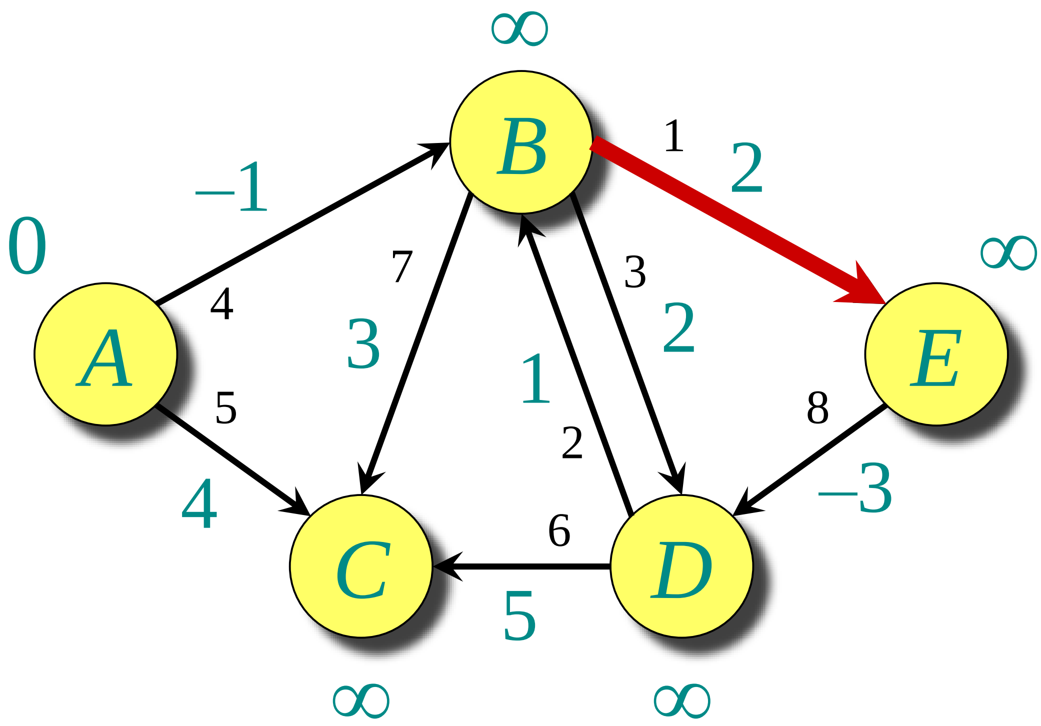
Initialization.

Example of Bellman-Ford

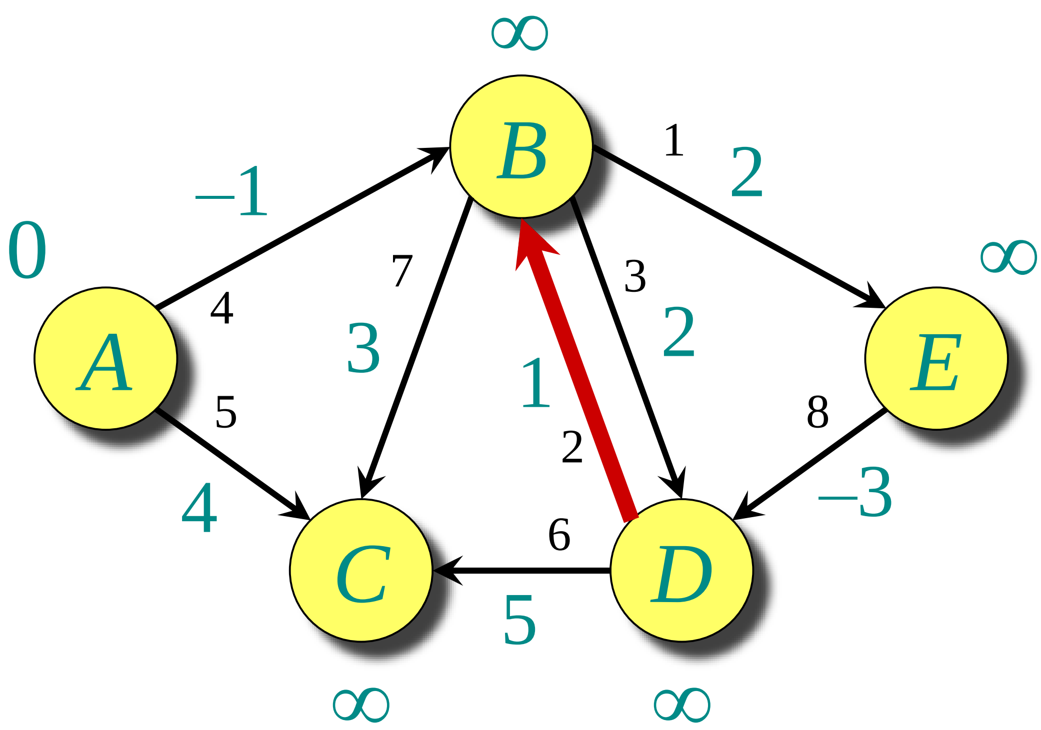


Order of edge relaxation.

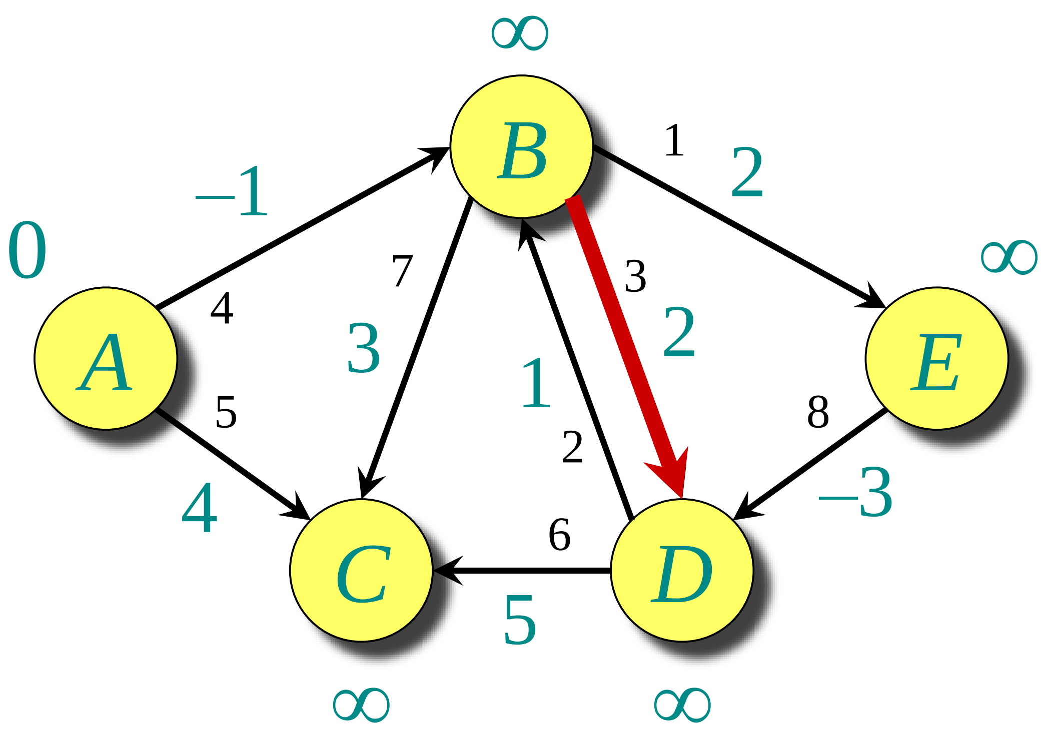
Example of Bellman-Ford



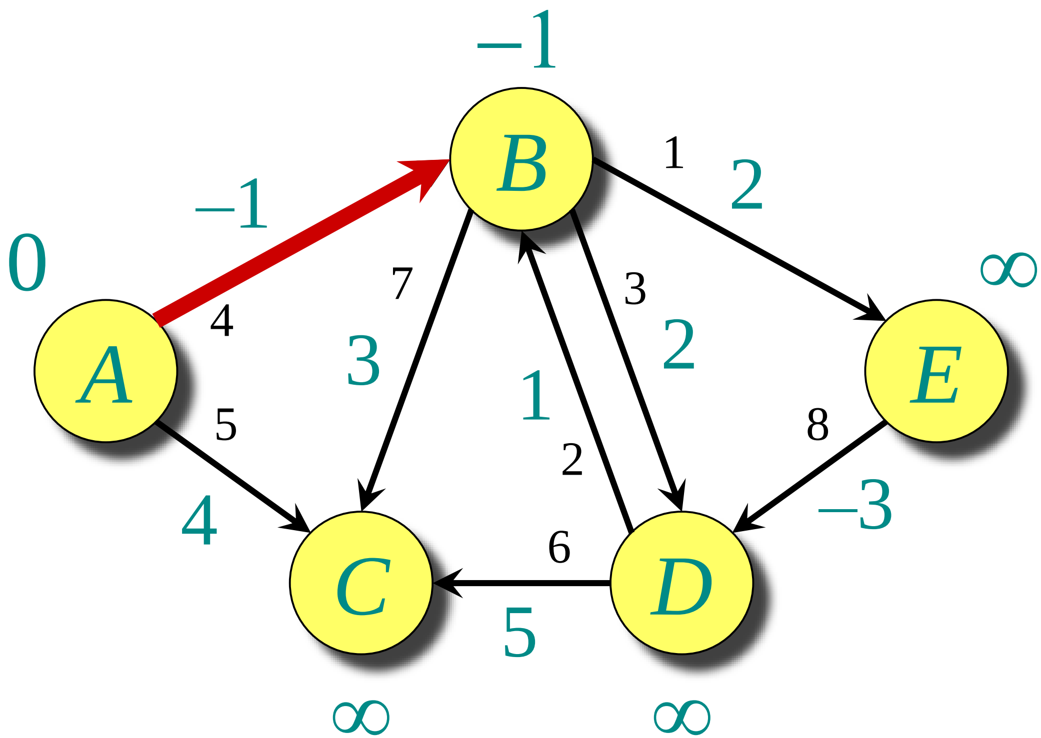
Example of Bellman-Ford



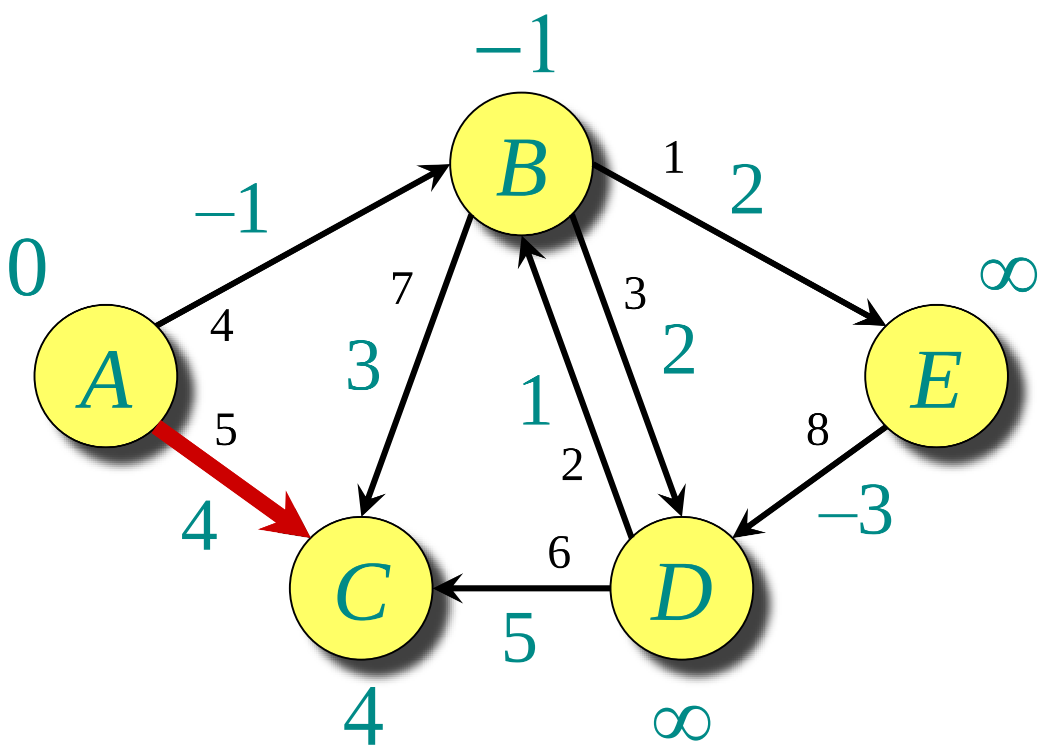
Example of Bellman-Ford



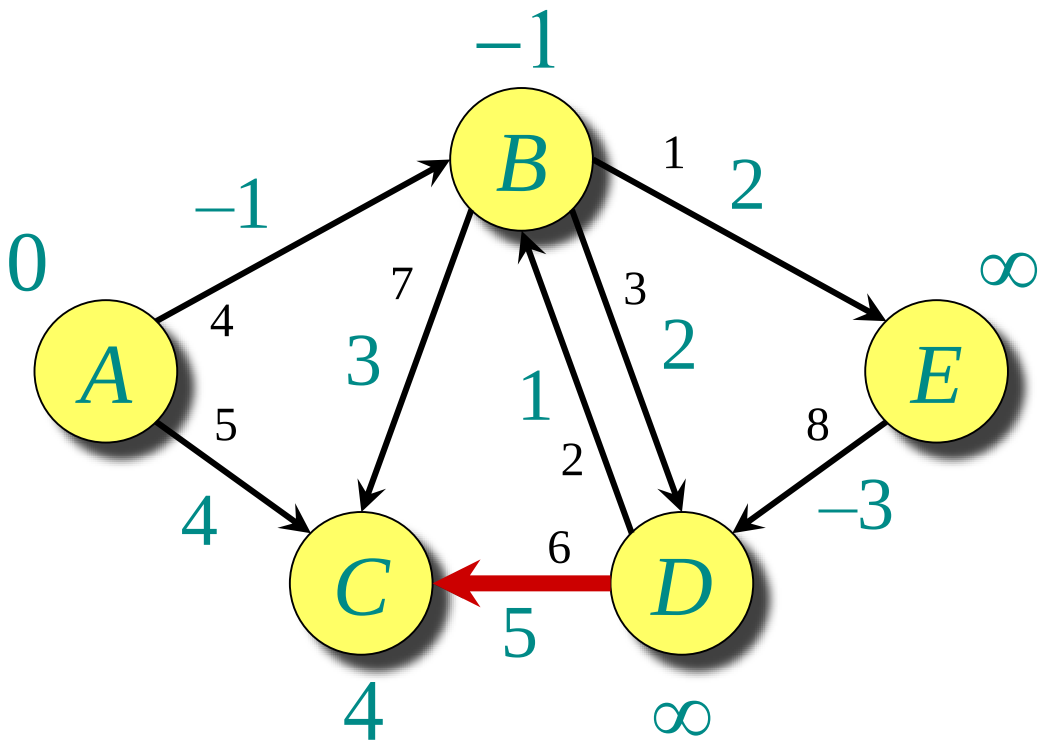
Example of Bellman-Ford



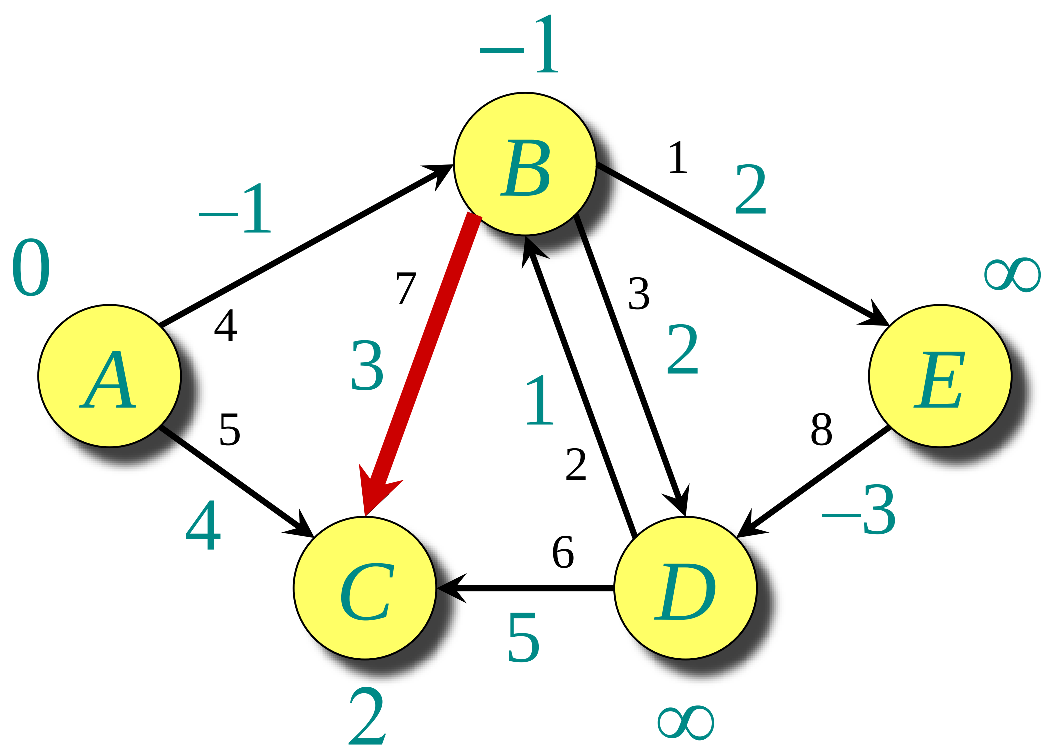
Example of Bellman-Ford



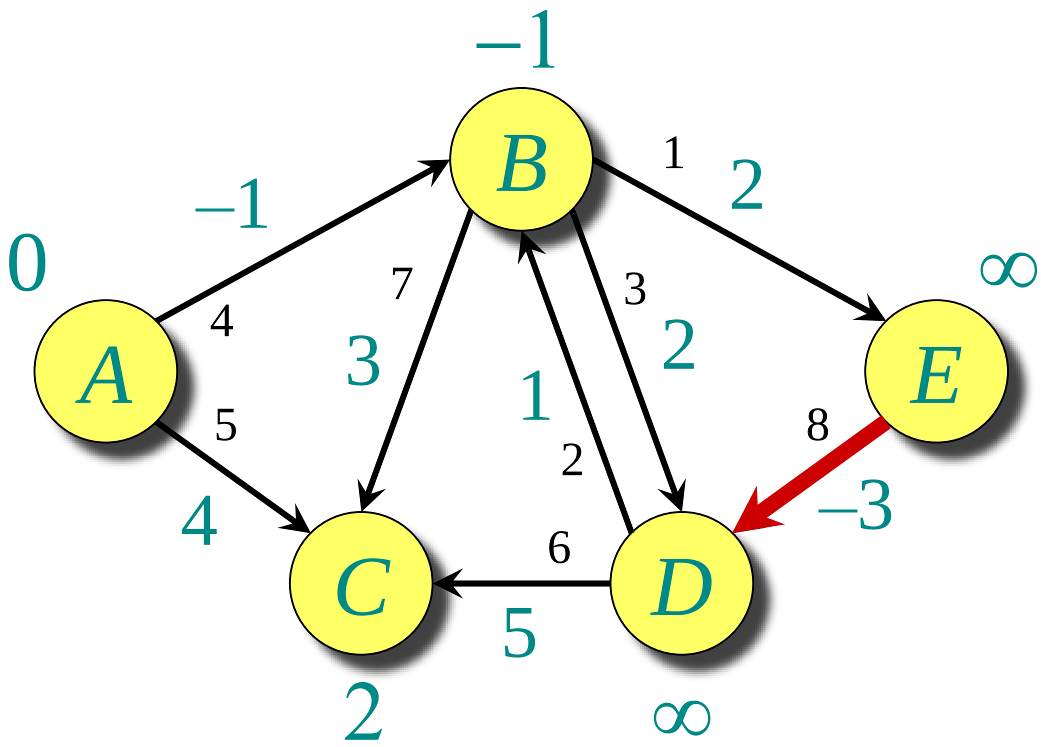
Example of Bellman-Ford



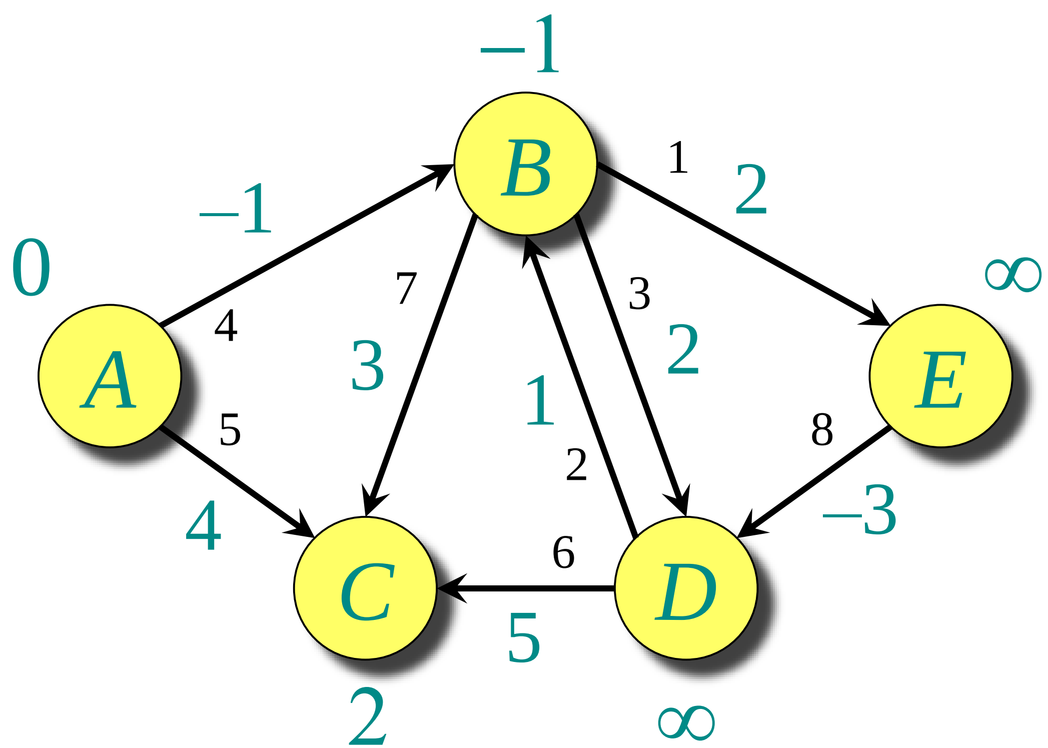
Example of Bellman-Ford



Example of Bellman-Ford

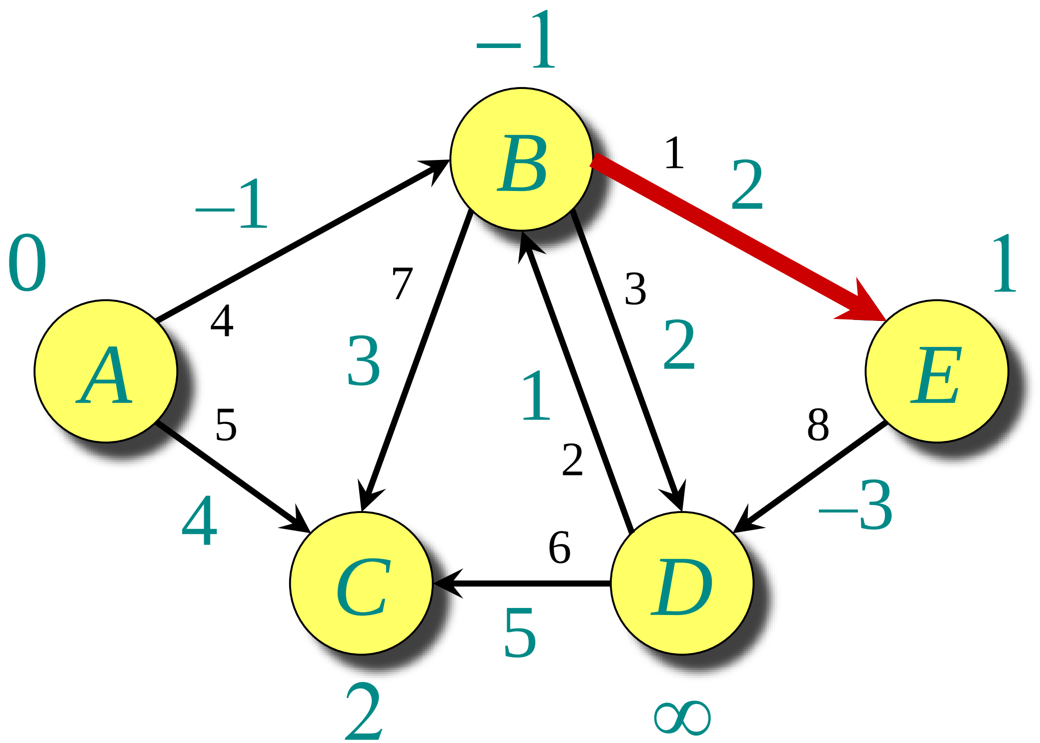


Example of Bellman-Ford

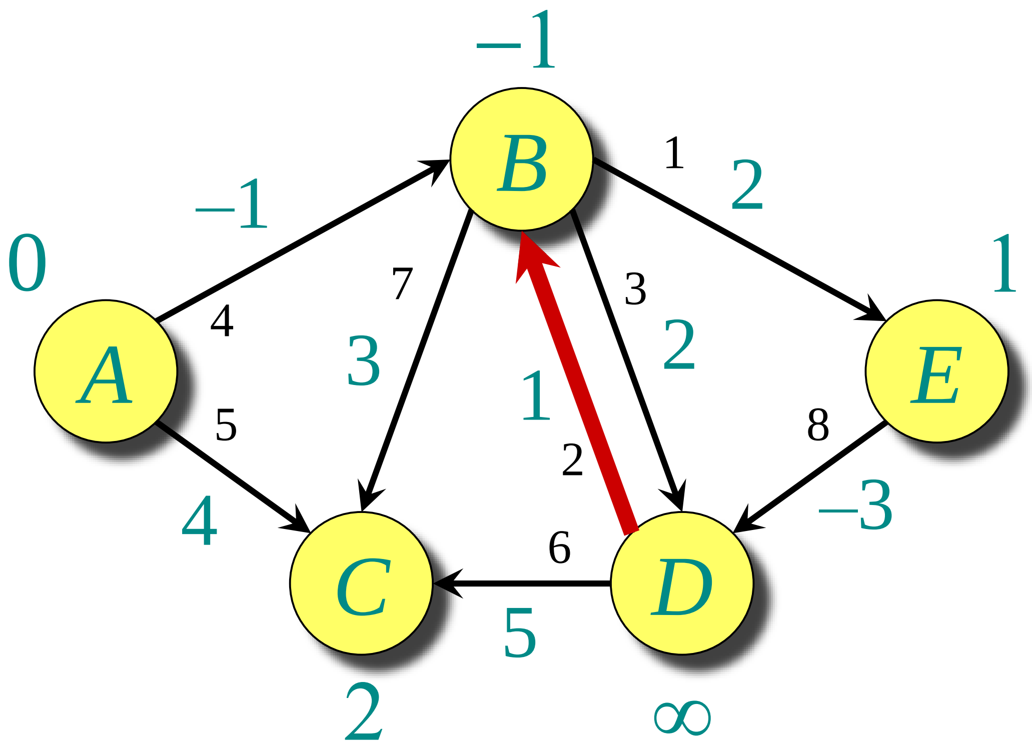


End of pass 1.

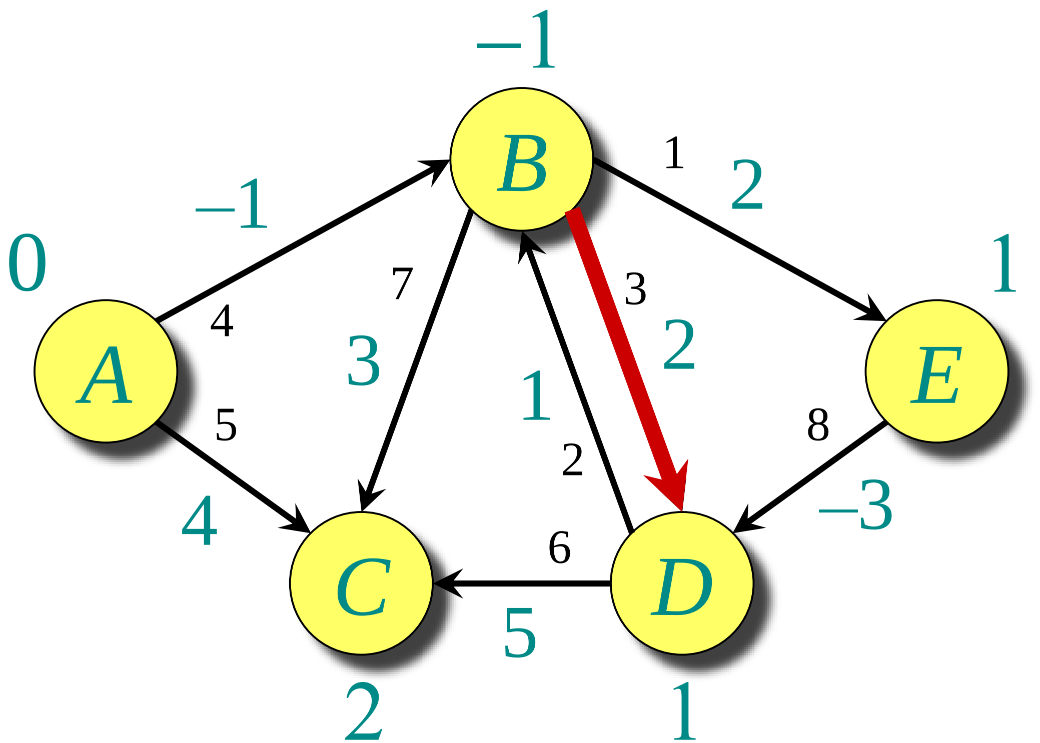
Example of Bellman-Ford



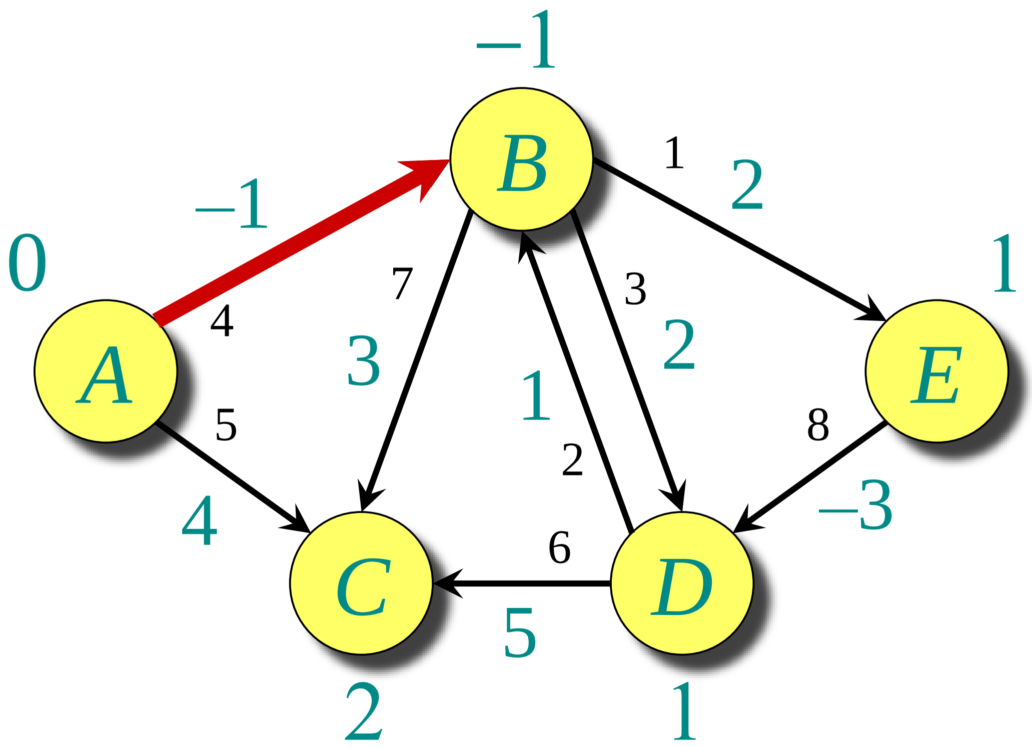
Example of Bellman-Ford



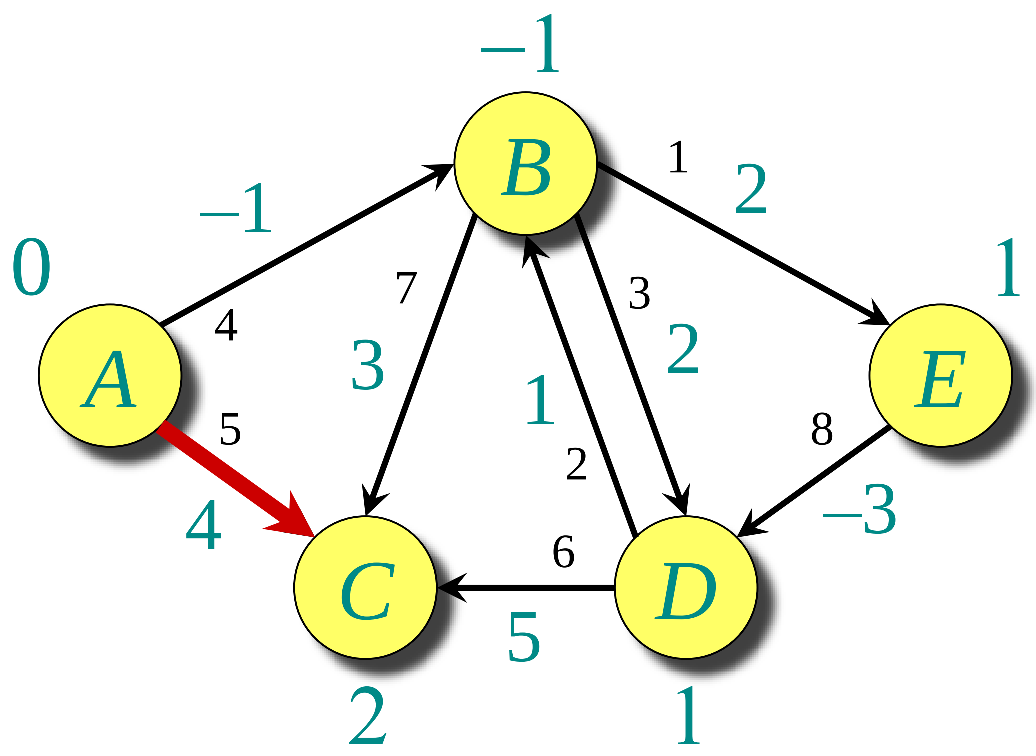
Example of Bellman-Ford



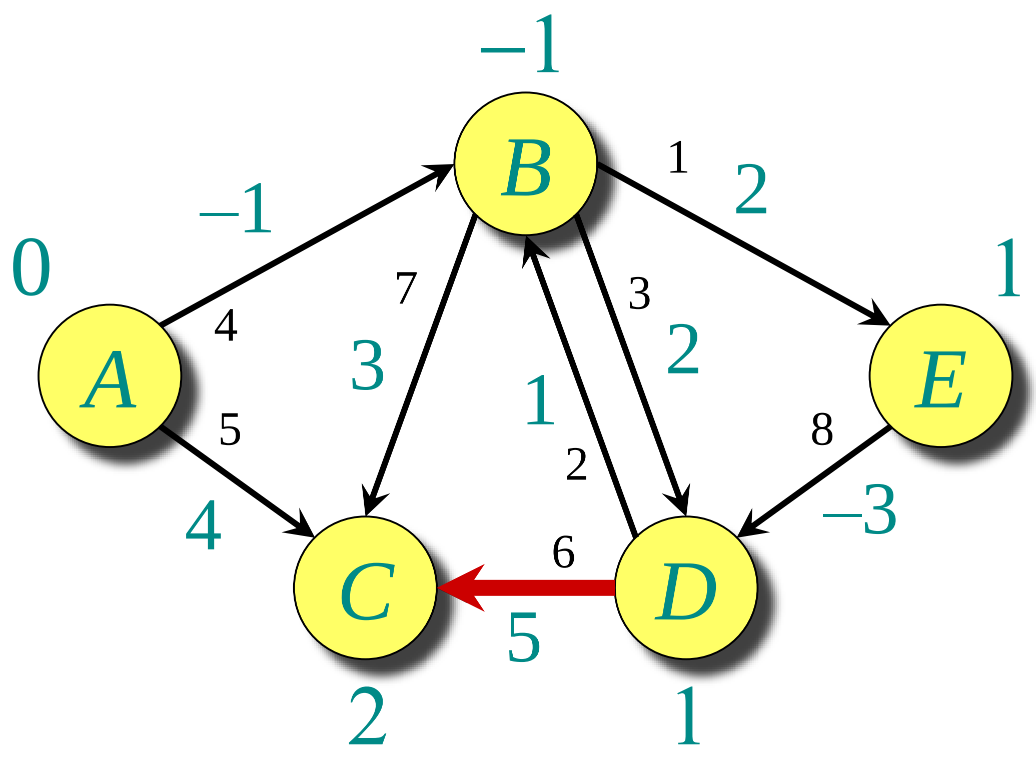
Example of Bellman-Ford



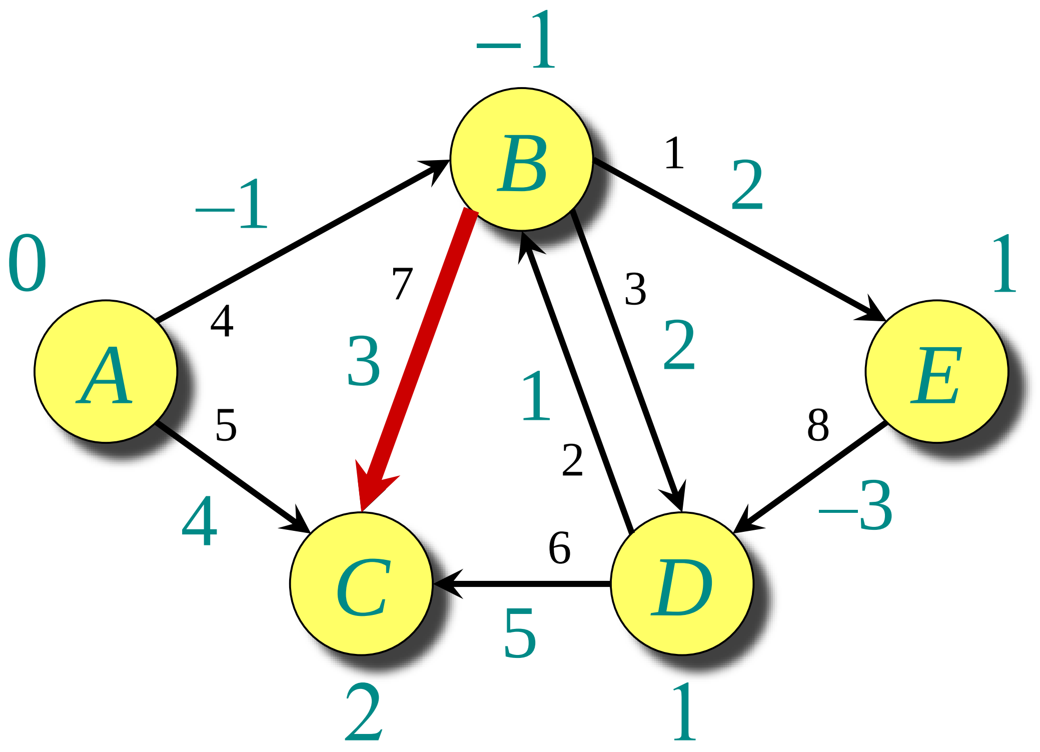
Example of Bellman-Ford



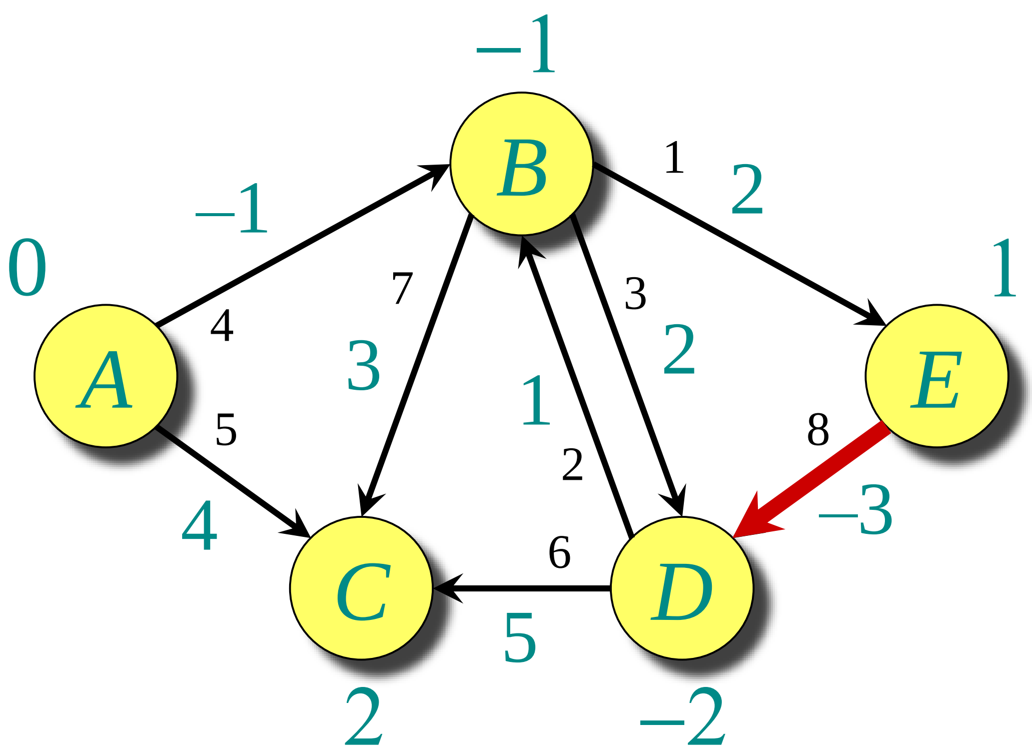
Example of Bellman-Ford



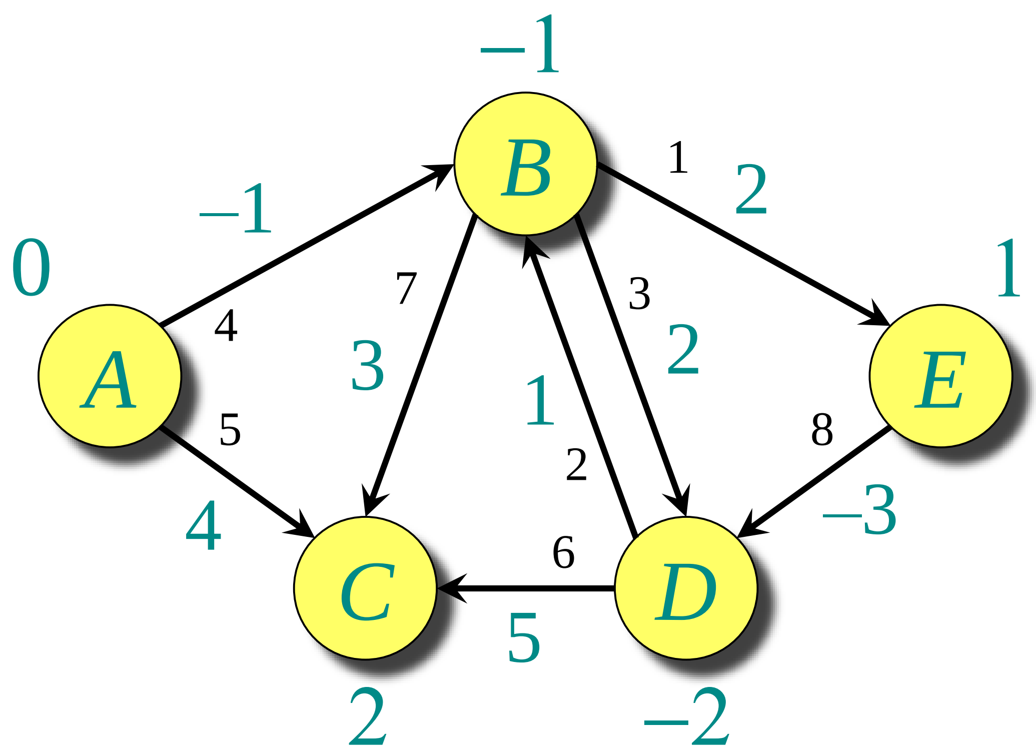
Example of Bellman-Ford



Example of Bellman-Ford



Example of Bellman-Ford



End of pass 2 (and 3 and 4).

Distance Vector Protocol

Distance Vector Protocol

Communication network.

- Nodes ≈ routers.
- Edges ≈ direct communication link.
- Cost of edge ≈ delay on link. ← naturally nonnegative, but Bellman-Ford used anyway!

Dijkstra's algorithm. Requires global information of network.

Bellman-Ford. Uses only local knowledge of neighboring nodes.

Synchronization. We don't expect routers to run in lockstep. The order in which each `foreach` loop executes is not important. Moreover, algorithm still converges even if updates are asynchronous.

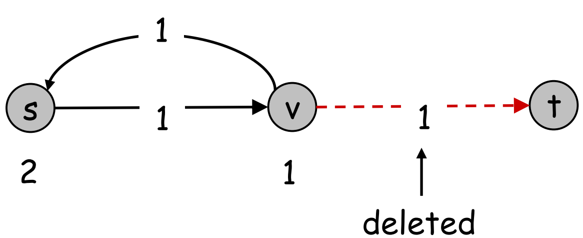
Distance Vector Protocol

Distance vector protocol.

- Each router maintains a vector of shortest path lengths to every other node (distances) and the first hop on each path (directions).
- Algorithm: each router performs n separate computations, one for each potential destination node.
- "Routing by rumor."

Ex. RIP, Xerox XNS RIP, Novell's IPX RIP, Cisco's IGRP, DEC's DNA Phase IV, AppleTalk's RTMP.

Caveat. Edge costs may change during algorithm (or fail completely).



"counting to infinity"

Path Vector Protocols

Link state routing.

not just the distance and first hop

- Each router also stores the entire path.
- Based on Dijkstra's algorithm.
- Avoids "counting-to-infinity" problem and related difficulties.
- Requires significantly more storage.

Ex. Border Gateway Protocol (BGP), Open Shortest Path First (OSPF).

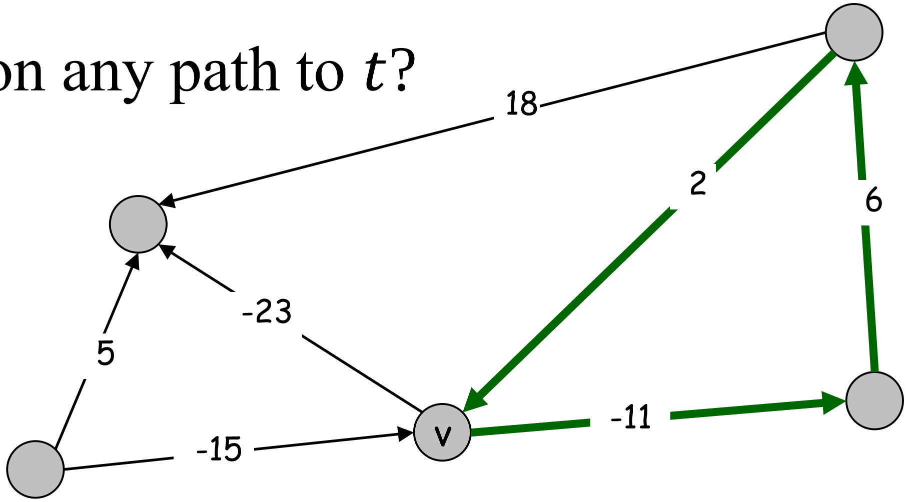
Detecting negative cycles in a graph

Detecting Negative Cycles

Bellman-Ford is guaranteed to work if there are no negative-cost cycles.

How can we tell if a negative-cost cycle exists?

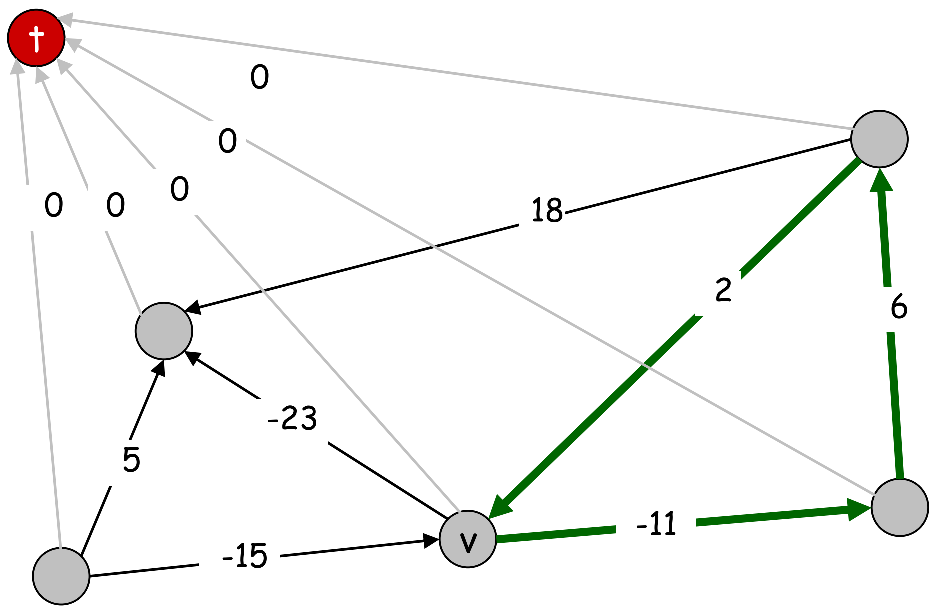
- We could pick a destination vertex t and check whether cost estimates in Bellman-Ford converge
- What is wrong with it?
  - What if the cycle isn't on any path to t?



Detecting Negative Cycles

Theorem. Can detect a negative cost cycle in O(mn) time.

- Add new node t and connect all nodes to t with 0-cost edge.
- Check if OPT(n, v) = OPT$(n-1, v)$ for all nodes v.
  - if yes, then no negative cycles
  - if no, then extract cycle from shortest path from v to t



Detecting Negative Cycles

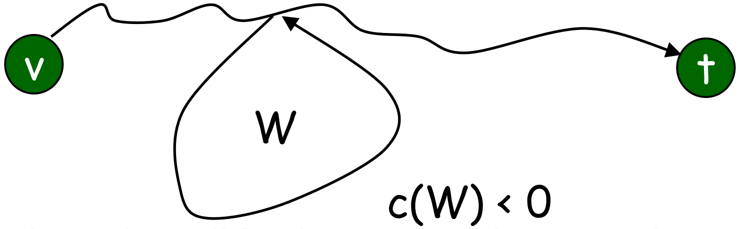
Lemma. If OPT(n, v) = OPT$(n-1, v)$ for all v, then no negative cycles are connected to t.

Proof. If OPT(n, v)=OPT$(n-1, v)$ for all v, then distance estimates won't change again even with many executions of the for loop. So there are no negative cost cycles on any path from v to t, for all v.

Lemma. If OPT(n, v) < OPT$(n-1, v)$ for some node v, then some path from v to t contains a cycle W of negative cost.

Proof. (by contradiction)

- Since OPT(n, v) < OPT$(n-1, v)$, current path P from v to t has n edges.
- By pigeonhole principle, P must contain a directed cycle W.
- Deleting W yields a v-t path with < n edges $\Rightarrow$ W has negative cost.

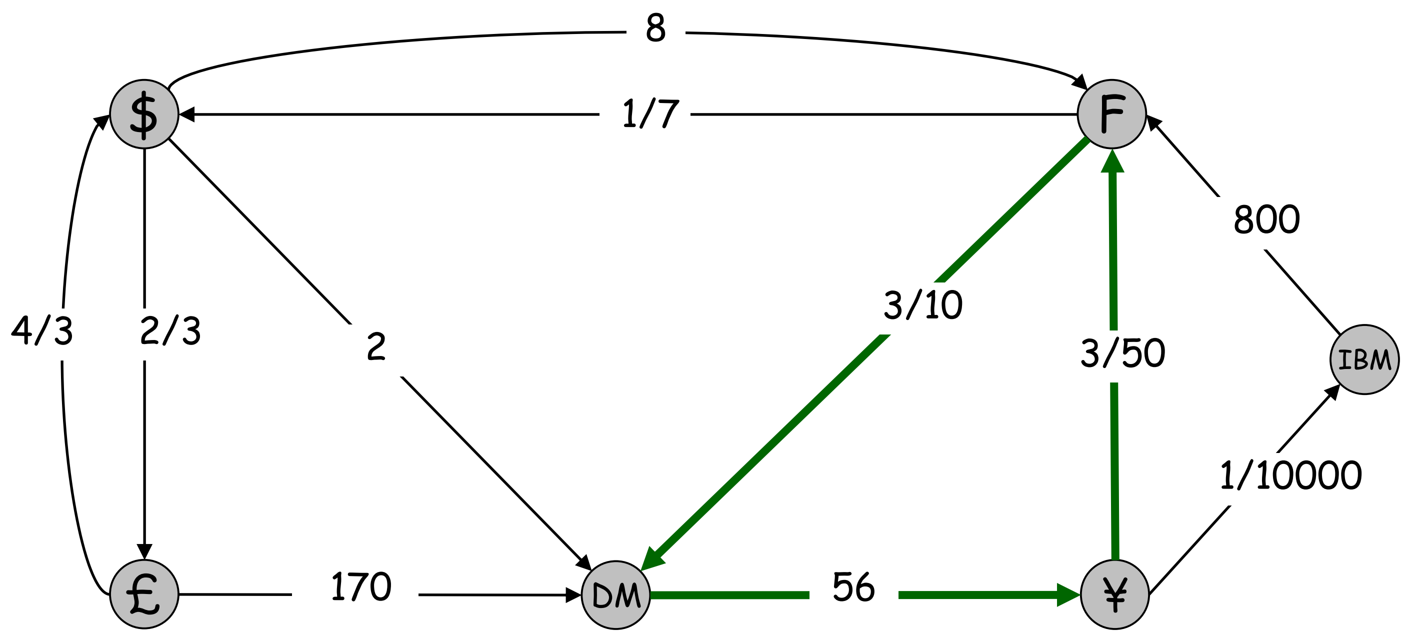


Detecting Negative Cycles: Application

Currency conversion. Given n currencies and exchange rates between pairs of currencies, is there an arbitrage opportunity?

Remark. Fastest algorithm very valuable!

Question. What should we use as edge costs?



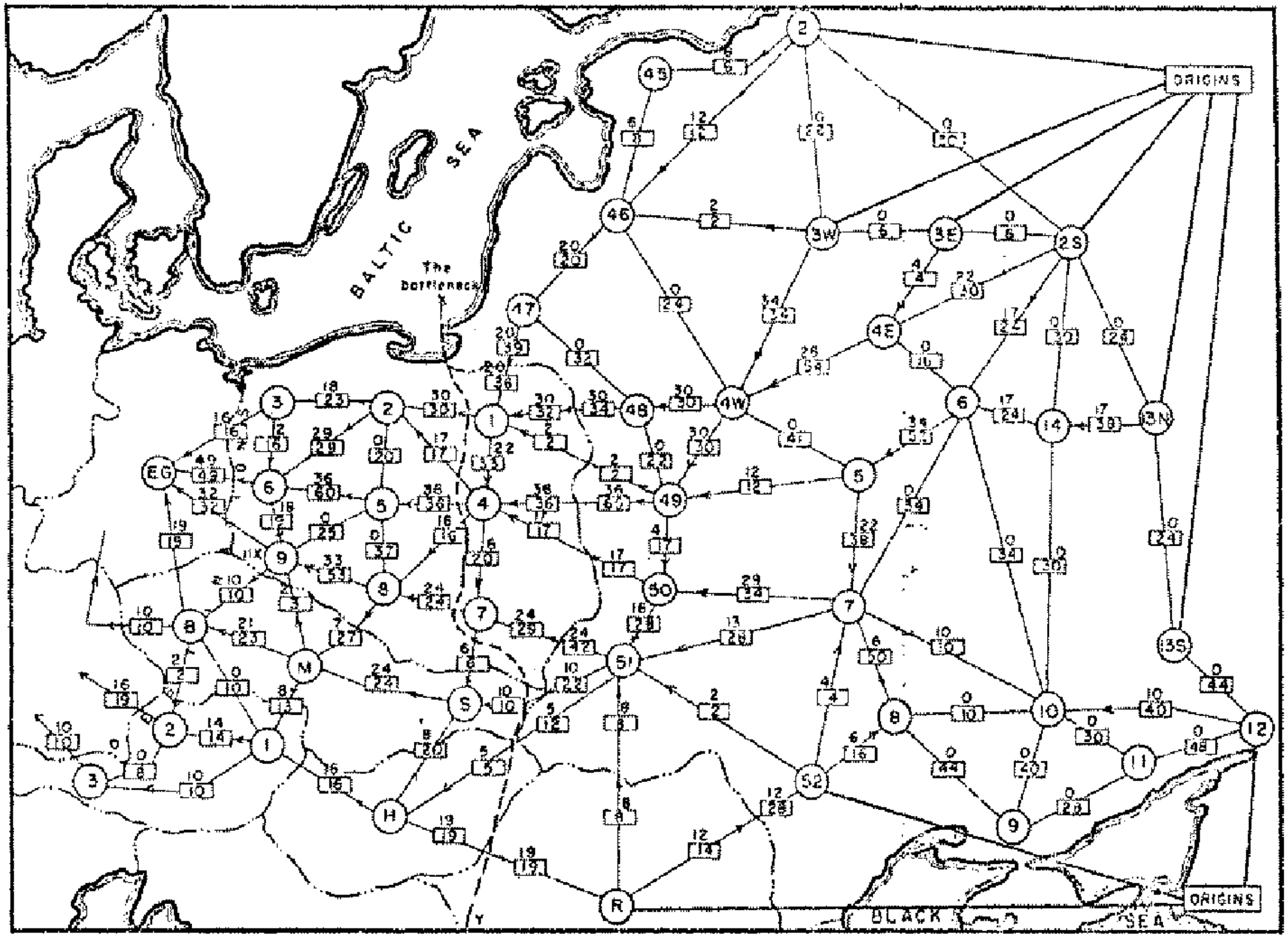
Detecting Negative Cycles: Summary

Bellman-Ford. $O(mn)$ time, $O(m + n)$ space.

- Run Bellman-Ford for n iterations (instead of n-1).
- Upon termination, Bellman-Ford successor variables trace a negative cycle if one exists.
- See p. 304 for improved version and early termination rule.

Network Flow and Linear Programming

Soviet Rail Network, 1955



Reference: *On the history of the transportation and maximum flow problems.*
Alexander Schrijver in Math Programming, 91: 3, 2002.

Maximum Flow and Minimum Cut

Max flow and min cut.

- Two very rich algorithmic problems.
- Cornerstone problems in combinatorial optimization.
- Beautiful mathematical duality.

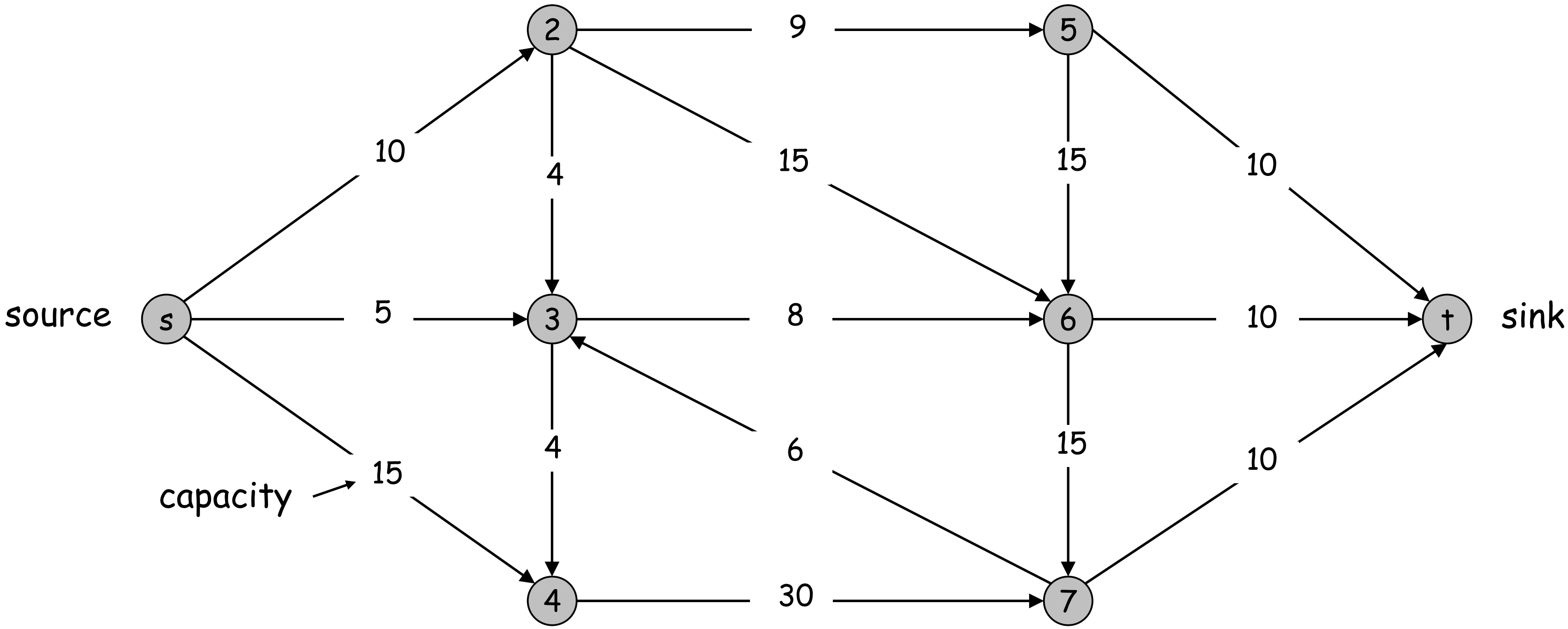
Nontrivial applications / reductions.

- Data mining.
- Open-pit mining.
- Project selection.
- Airline scheduling.
- Bipartite matching.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
- Network reliability.
- Distributed computing.
- Egalitarian stable matching.
- Security of statistical data.
- Network intrusion detection.
- Multi-camera scene reconstruction.
- Many many more . . .

Minimum Cut Problem

Flow network.

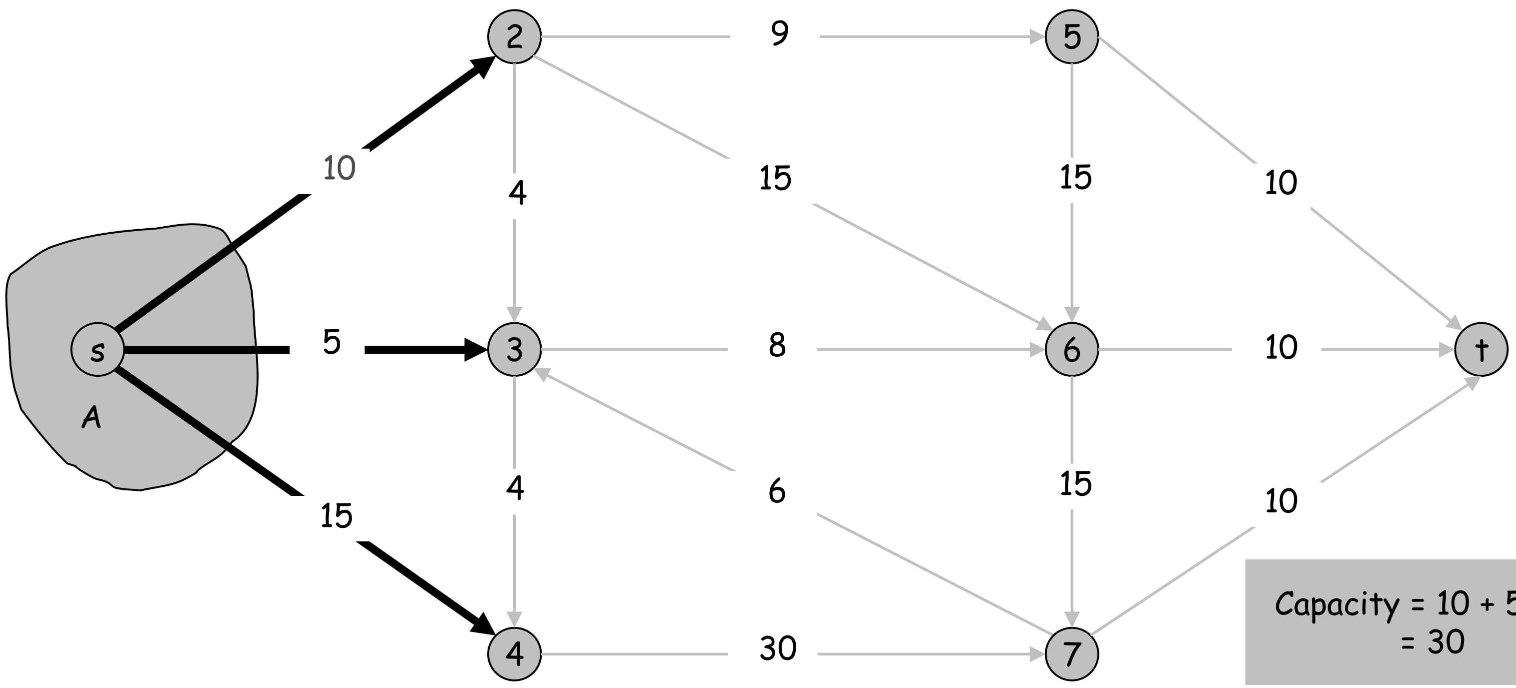
- Abstraction for material flowing through the edges.
- G = (V, E) = directed graph, no parallel edges.
- Two distinguished nodes: s = source, t = sink.
- c(e) = capacity of edge e.



Cuts

Def. An s-t cut is a partition (A, B) of V with $s \in A$ and $t \in B$.

Def. The capacity of a cut (A, B) is: $$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$



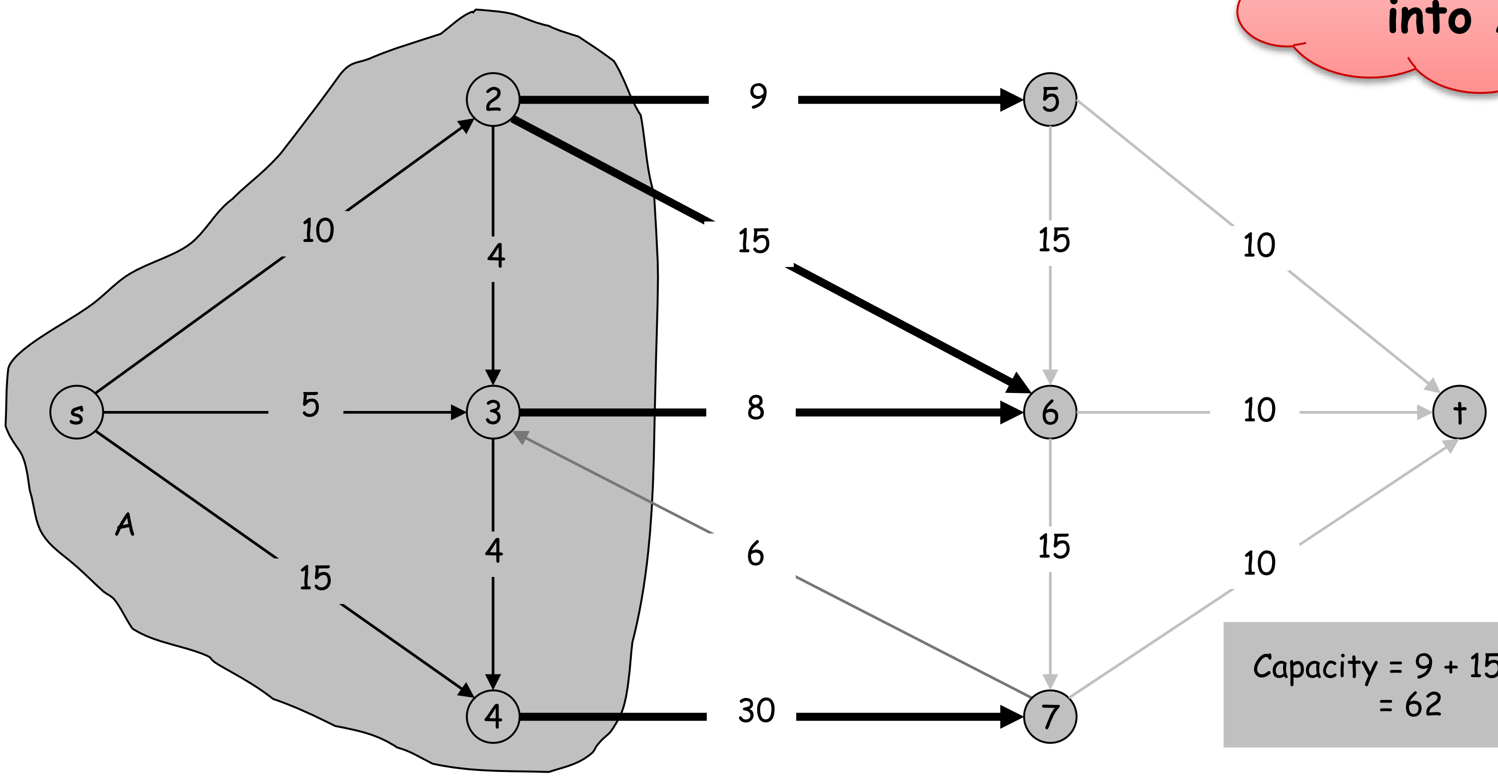
Capacity = 10 + 5 + 15
= 30

Cuts

Def. An s-t cut is a partition (A, B) of V with s $\in$ A and t $\in$ B.

Def. The capacity of a cut (A, B) is: $$cap(A, B) \ = \sum_{e \text{ out of } A} c(e)$$

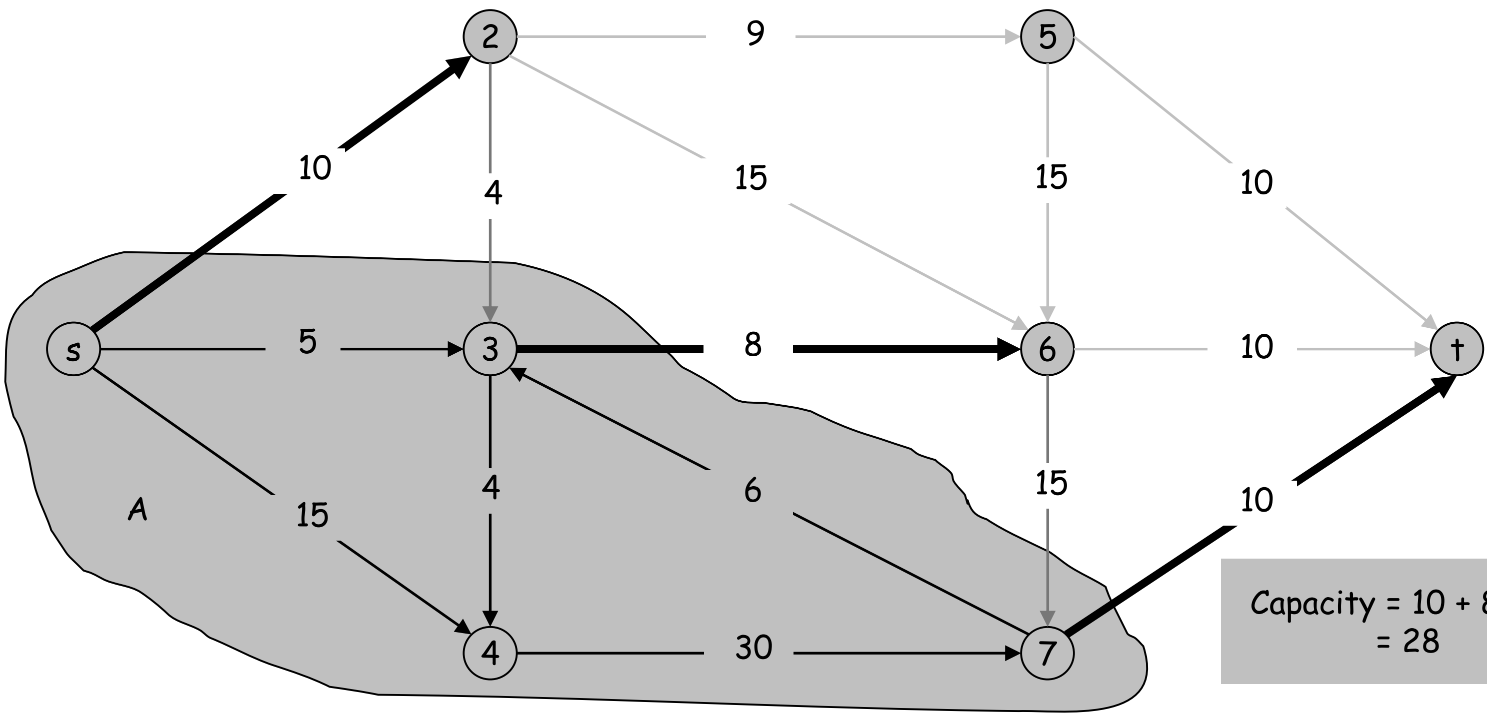
we don't count edges **into A**



Capacity = 9 + 15 + 8 + 30
= 62

Minimum Cut Problem

Min s-t cut problem. Find an s-t cut of minimum capacity.



Capacity = 10 + 8 + 10
= 28

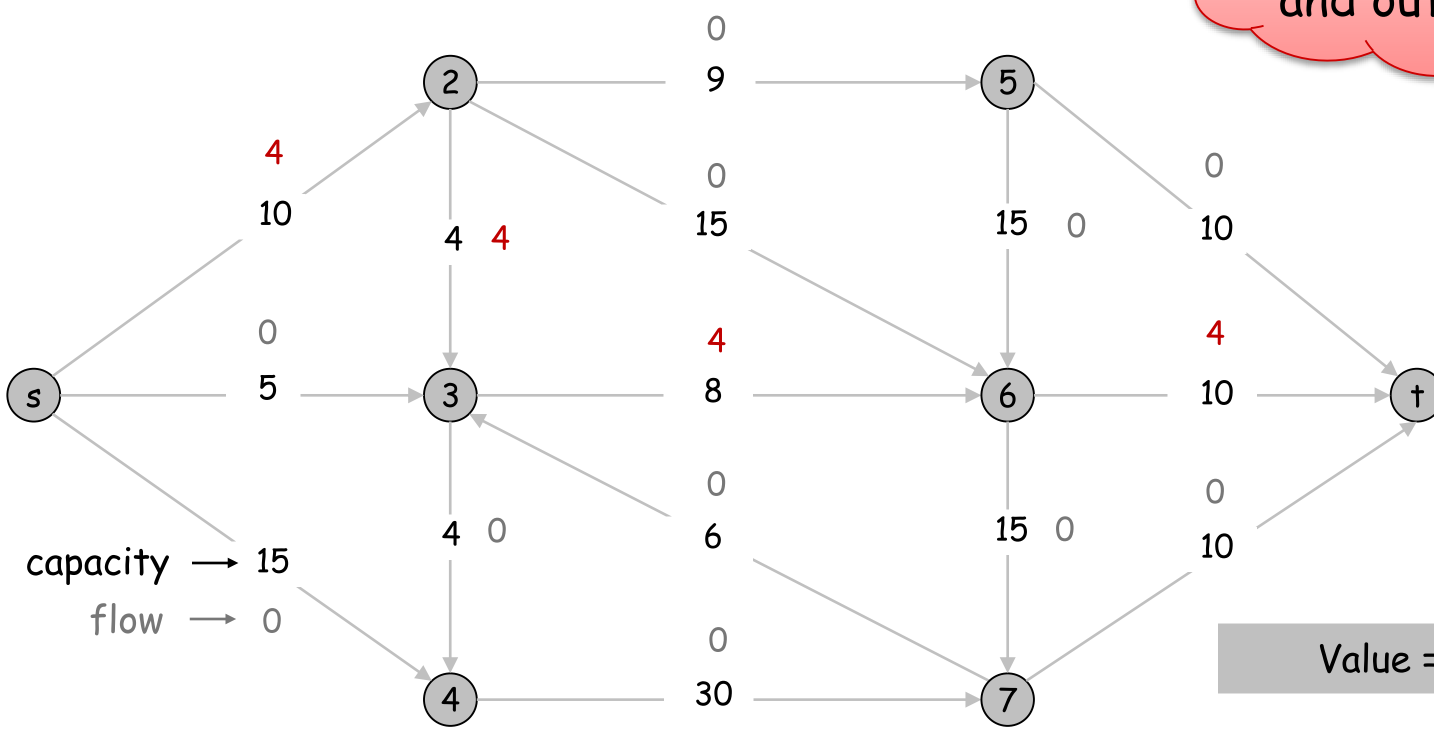
Flows

Def. An s-t flow is a function that satisfies:

- For each $e \in E$: $0 \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

Water flowing in and out of v

Def. The value of a flow f is: $v(f) = \sum_{e \text{ out of } s} f(e)$.



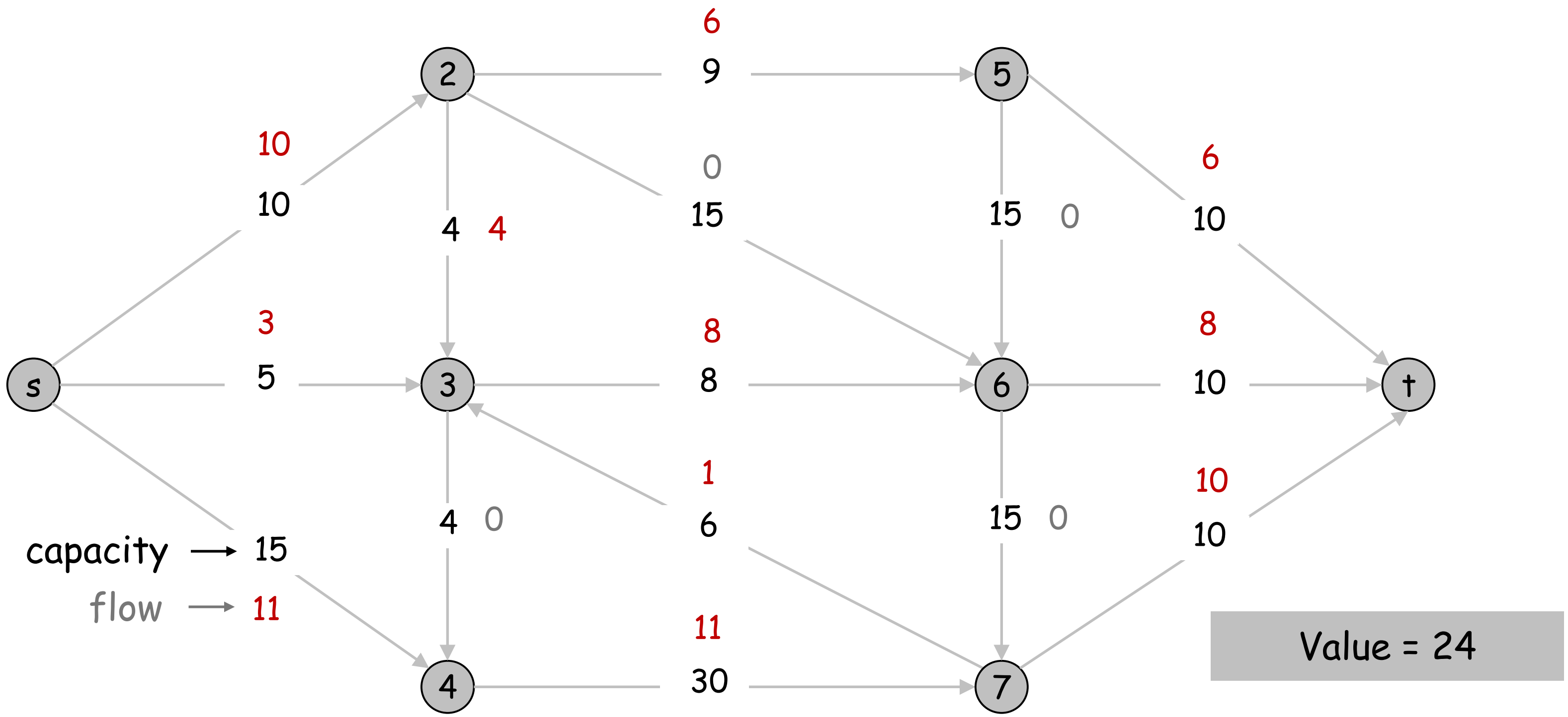
Value = 4

Flows

Def. An s-t flow is a function that satisfies:

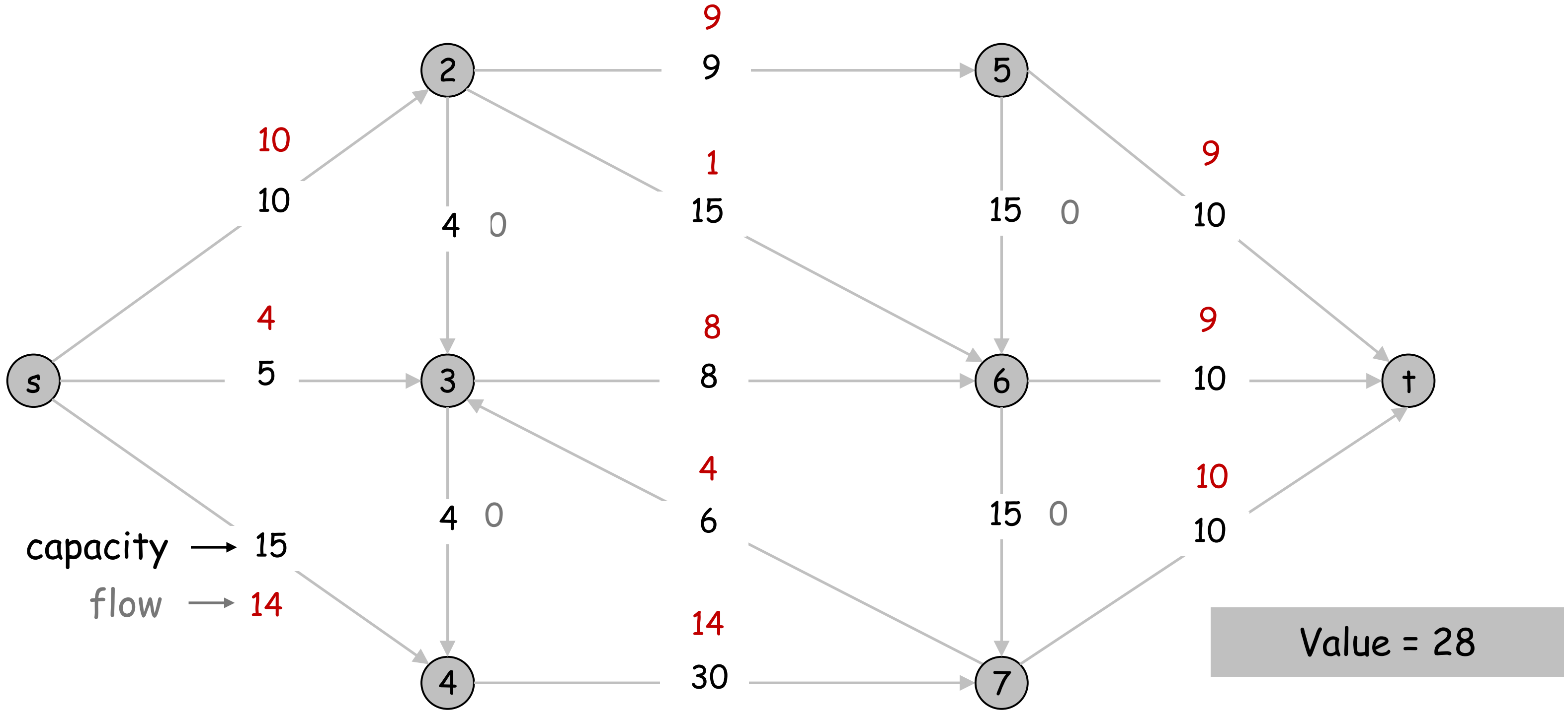
- For each $e \in E$: $0 \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

Def. The value of a flow f is: $v(f) = \sum_{e \text{ out of } s} f(e)$.



Maximum Flow Problem

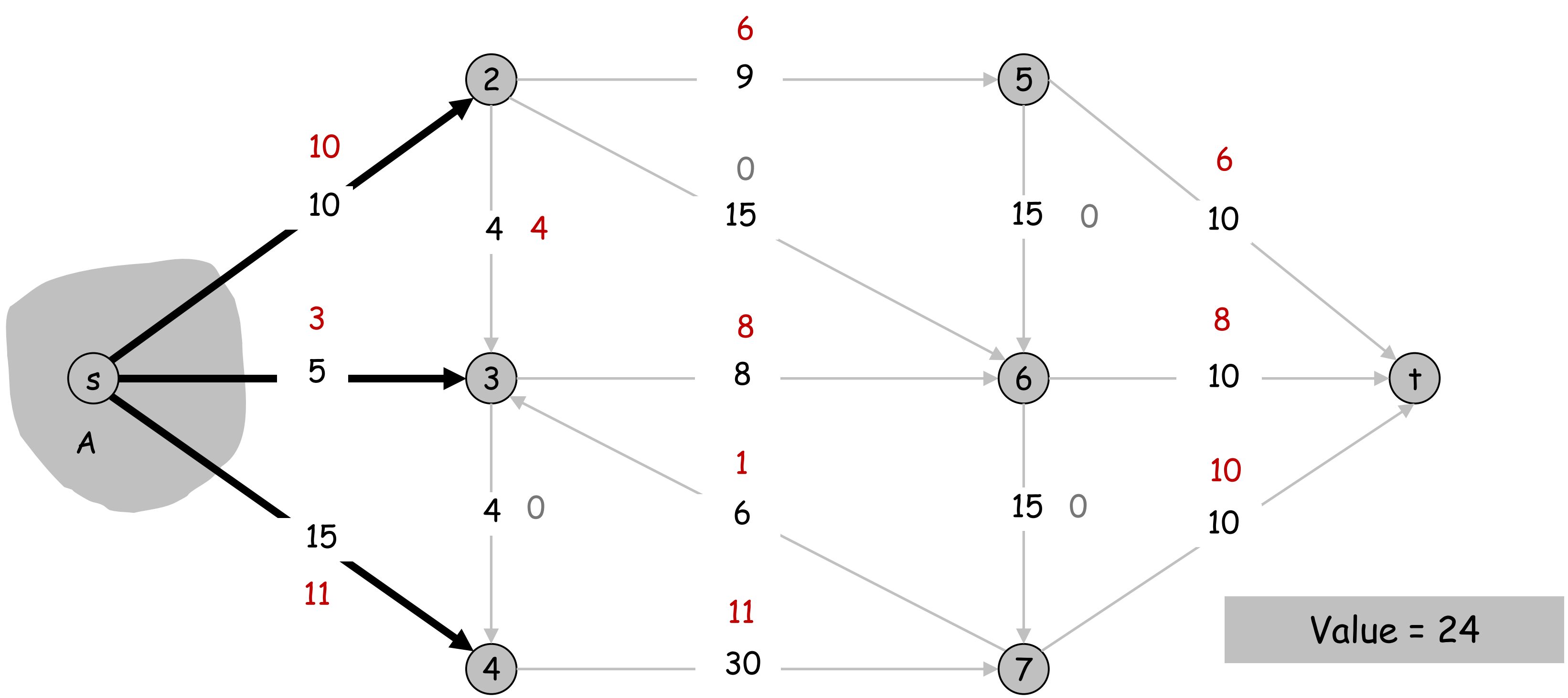
Max flow problem. Find s-t flow of maximum value.



Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then the net flow sent across the cut is equal to the amount leaving s.

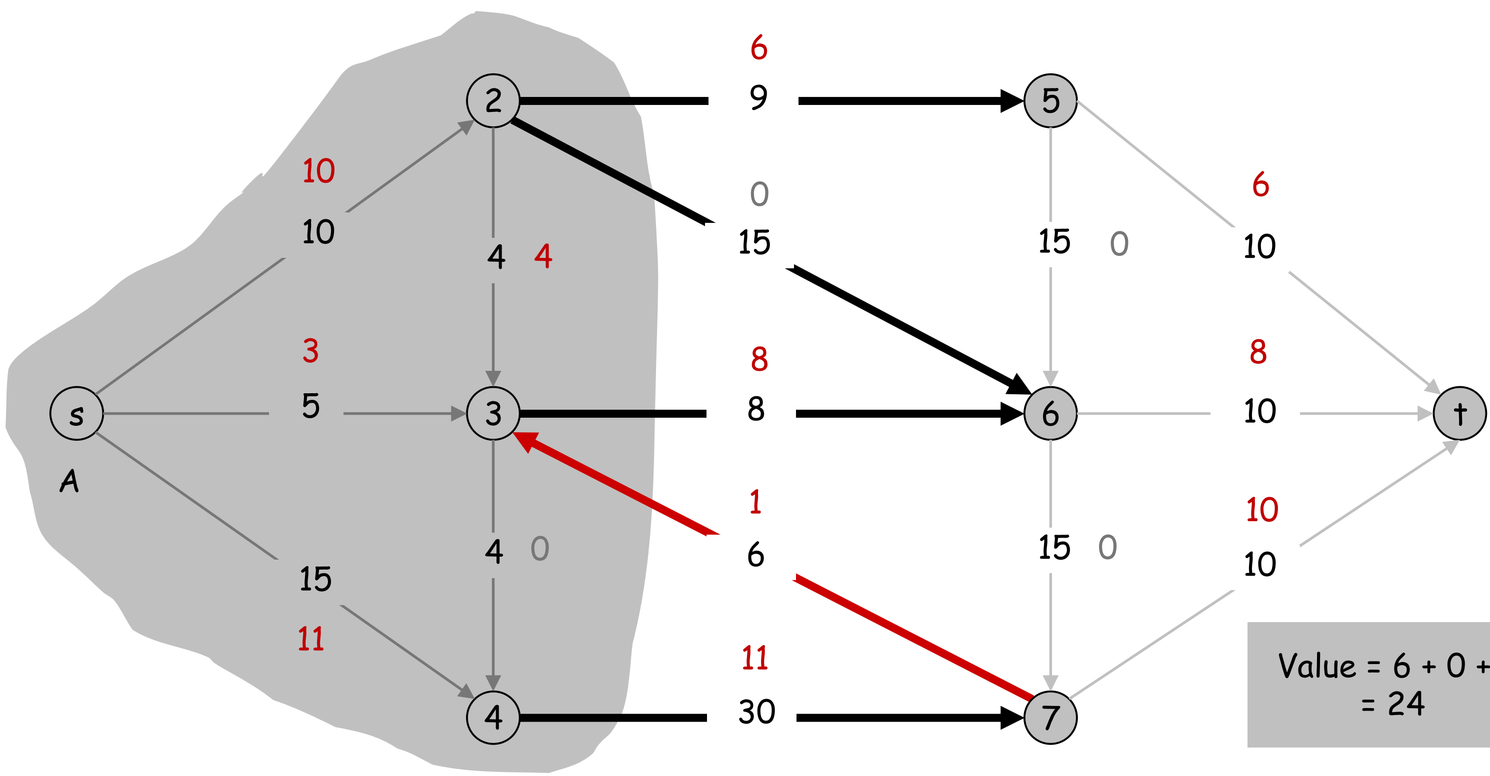
$$\sum_{e \text{ out of } A} f(e) \; - \sum_{e \text{ in to A}} f(e) \; = \; v(f)$$



Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then the net flow sent across the cut is equal to the amount leaving s.

$$\sum_{e \text{ out of } A} f(e) \; - \; \sum_{e \text{ in to A}} f(e) \; = \; v(f)$$

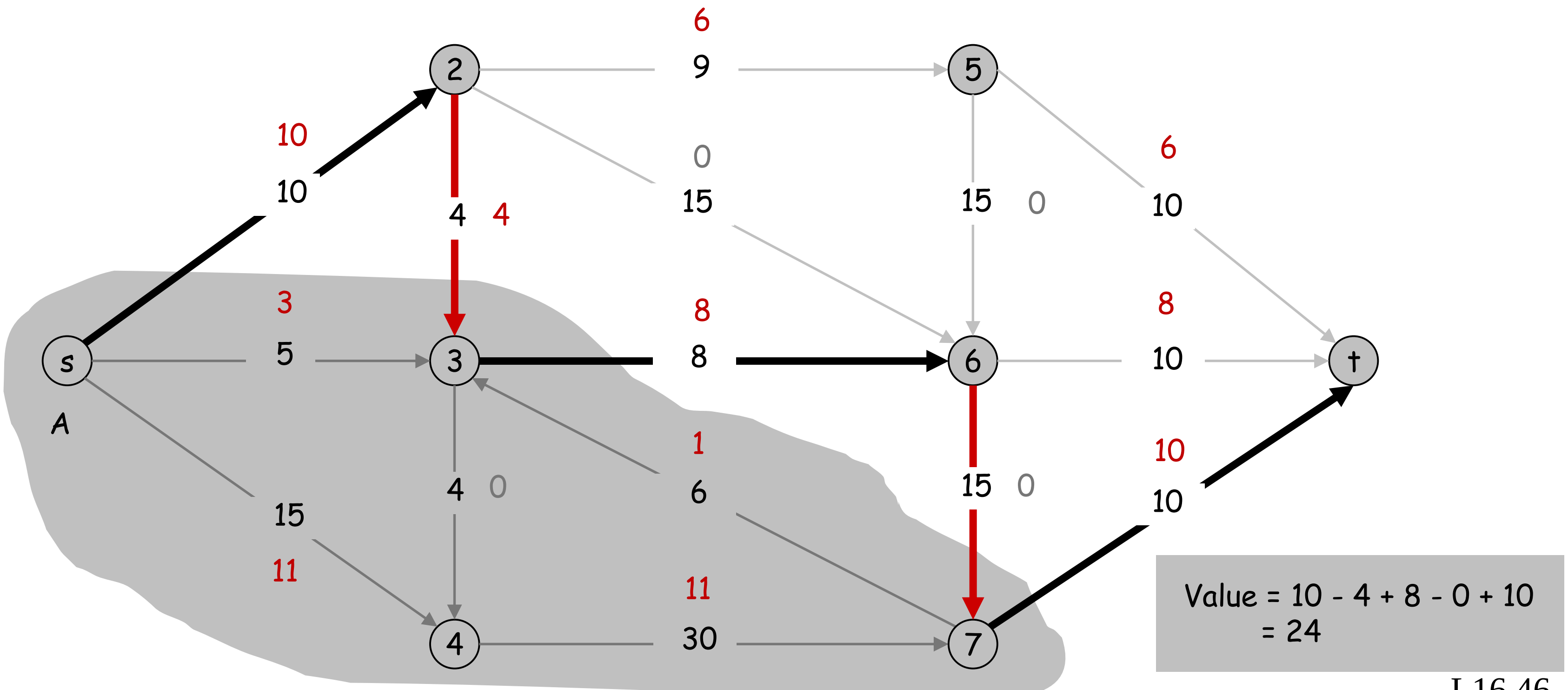


Value = 6 + 0 + 8 - 1 + 11
= 24

Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then the net flow sent across the cut is equal to the amount leaving s.

$$\sum_{e \text{ out of } A} f(e) \; - \; \sum_{e \text{ in to A}} f(e) \; = \; v(f)$$



Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f).$$

Proof.

$$v(f) = \sum_{e \text{ out of } s} f(e)$$

by flow conservation, all terms except v = s are 0 $\longrightarrow$

$$= \sum_{v \in A} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to v}} f(e) \right)$$

$$= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to A}} f(e).$$

Question

Two problems

- Min Cut
- Max Flow

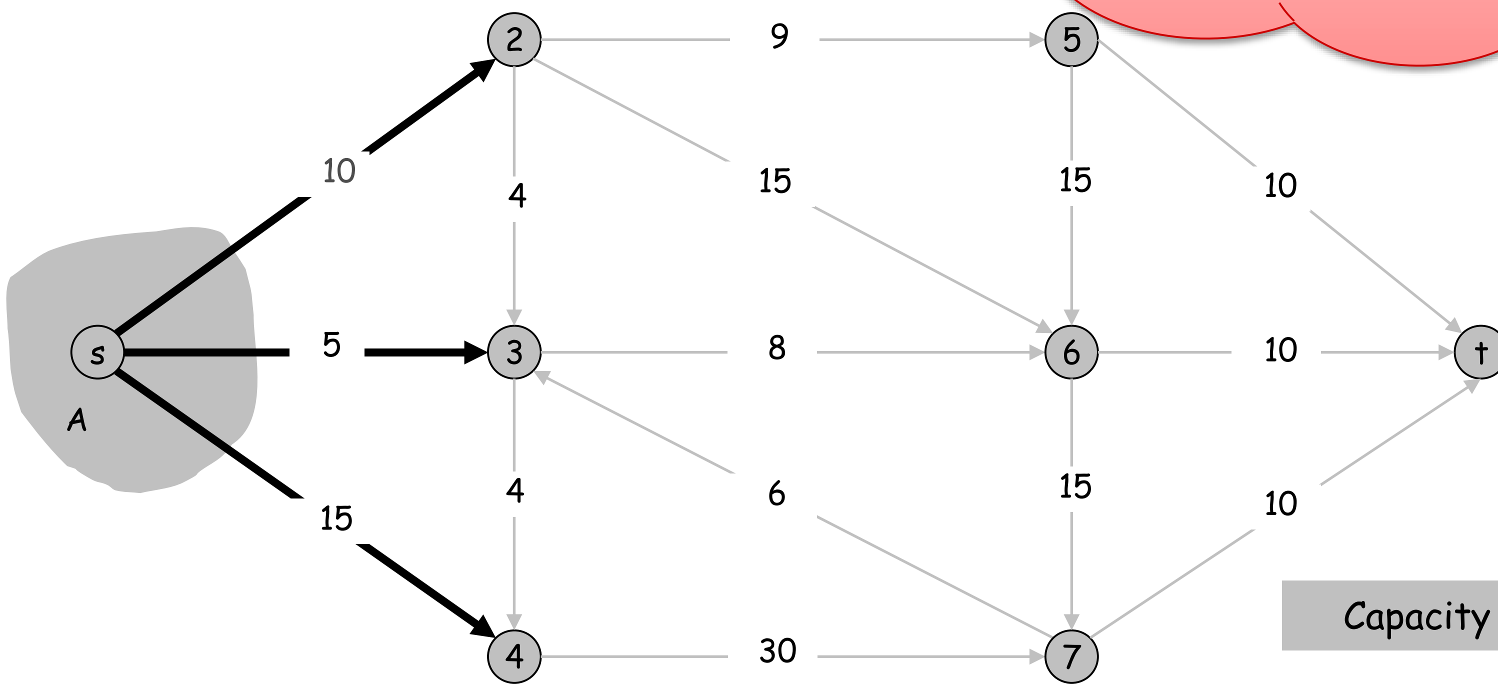
- How do they relate?

Flows and Cuts

Weak duality. Let f be any flow, and let (A, B) be any s-t cut. Then the value of the flow is at most the capacity of the cut.

Big Optimization Idea #1:
Look for **structural constraints**, e.g.
max flow ≤ min-cut

Cut capacity = 30 $\Rightarrow$ Flow value ≤ 30



Capacity = 30

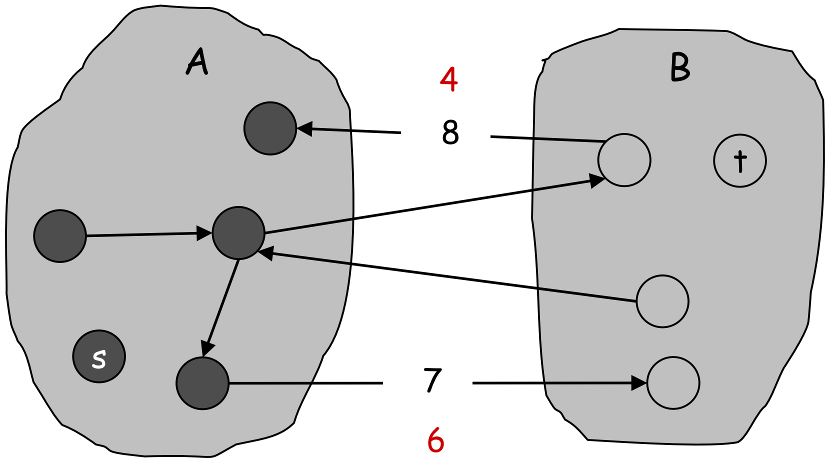
Flows and Cuts

Weak duality. Let f be any flow. Then, for any s-t cut (A, B),

$v(f) \le cap(A, B)$.

Proof.

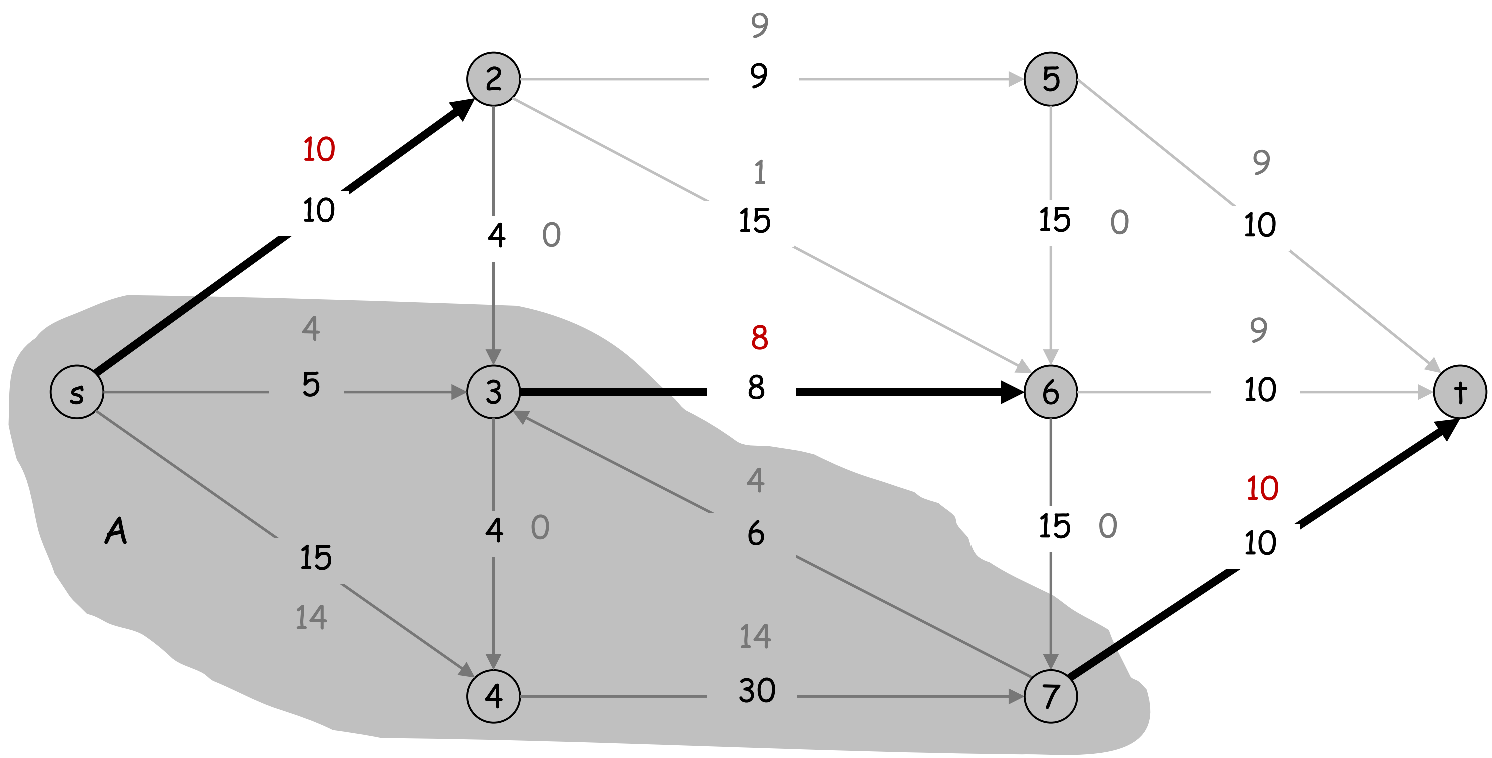
$$
\begin{aligned}
v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\
&\le \sum_{e \text{ out of } A} f(e) \\
&\le \sum_{e \text{ out of } A} c(e) \\
&= \operatorname{cap}(A, B) \quad \blacksquare
\end{aligned}
$$



Certificate of Optimality

Corollary. Let f be any flow, and let (A, B) be any cut.
If v(f) = cap(A, B), then f is a max flow and (A, B) is a min cut.

Value of flow = 28
Cut capacity = 28 $\Rightarrow$ Flow value $\leq$ 28



Review Questions

True/False

Let G be an arbitrary flow network, with a source s, and sink t, and a positive integer capacity c_e on every edge e.

1) If f is a maximum s-t flow in G, then f saturates every edge out of s with flow (i.e., for all edges e out of s, we have $f(e) = c_e$).

2) Let (A,B) be a minimum s-t cut with respect to these capacities . If we add 1 to every capacity, then (A,B) is still a minimum s-t cut with respect to the new capacities.