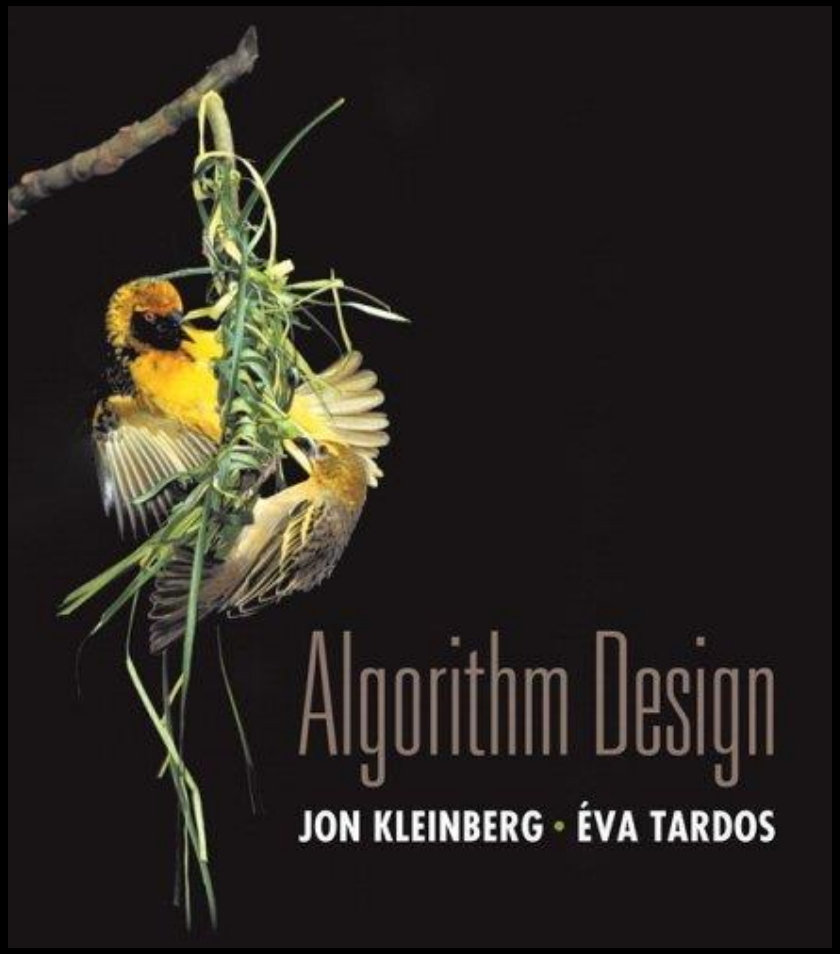
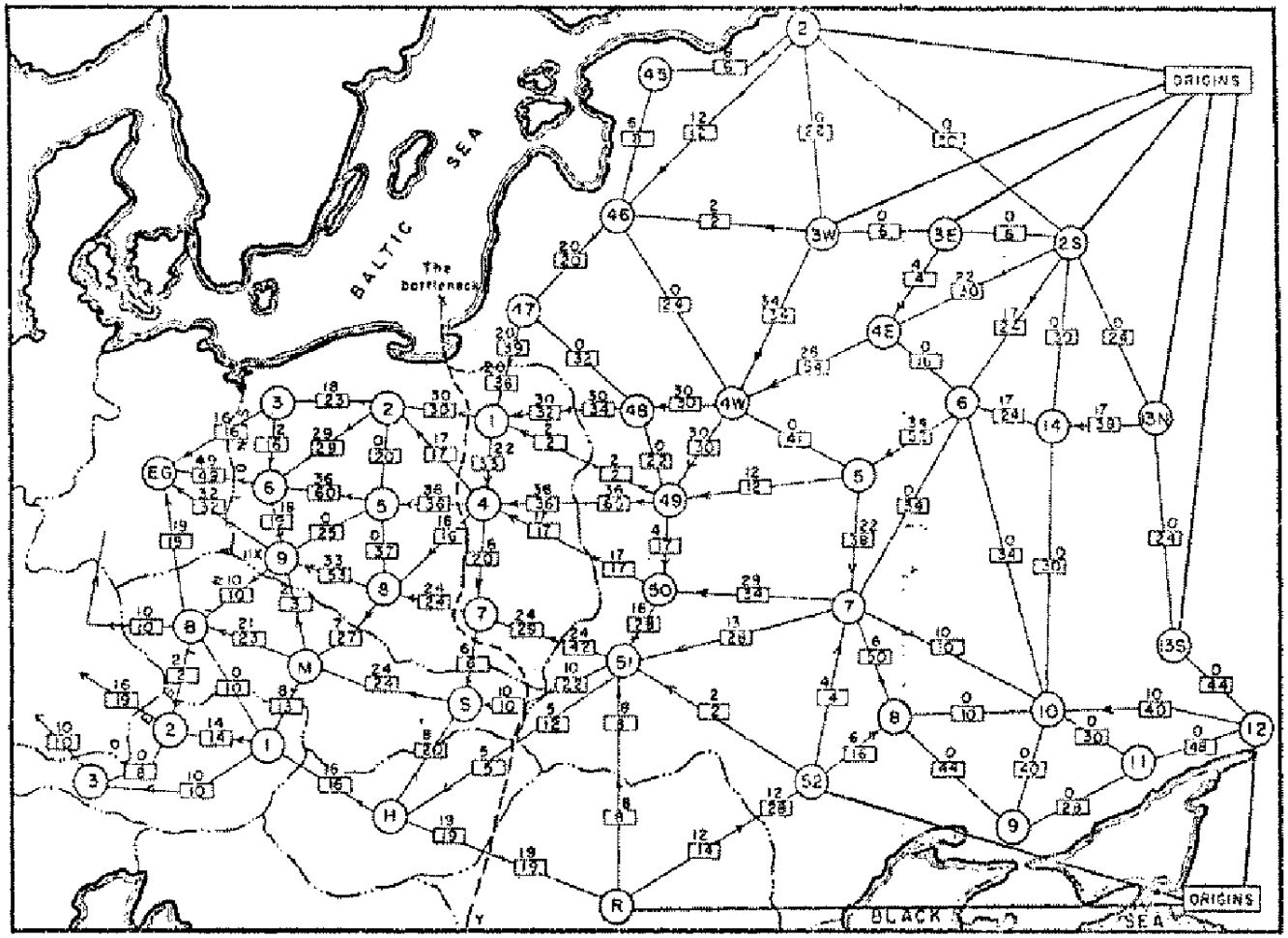


Chapter 7

Network Flow

CS 350: Winter 2018

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.

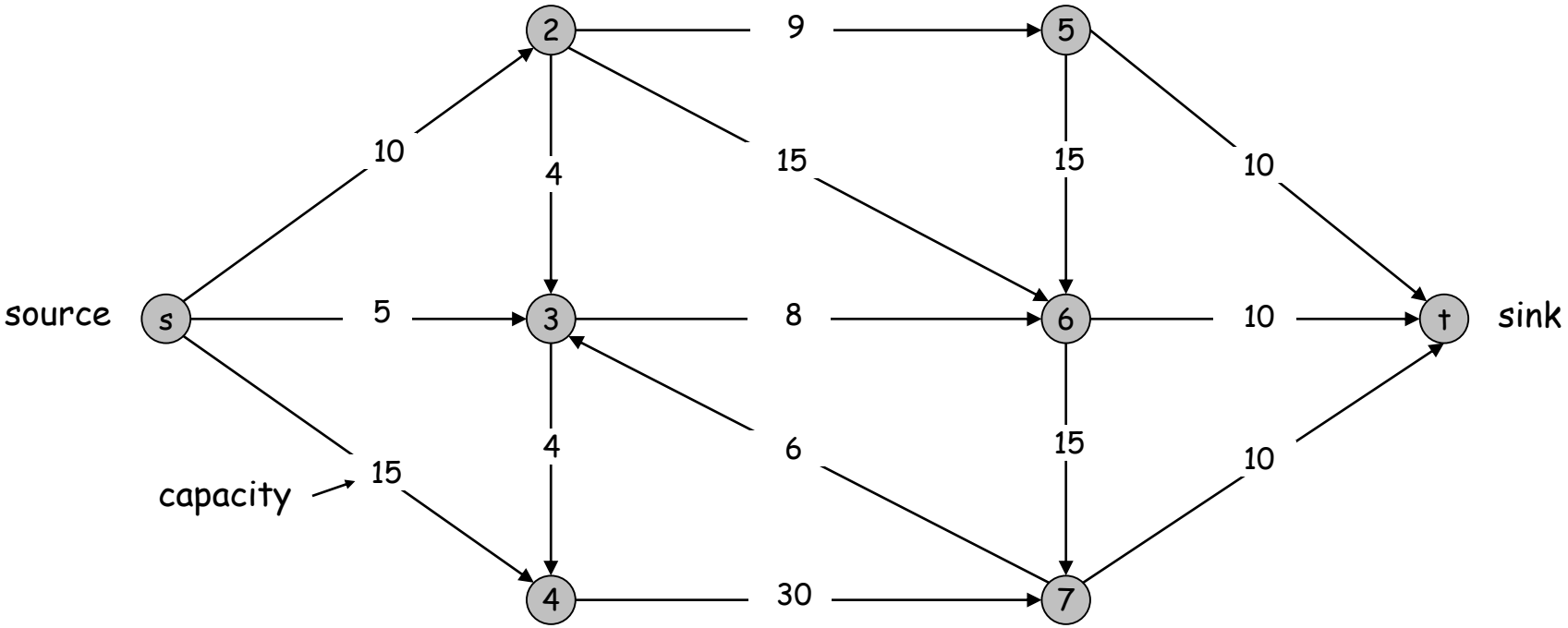
Nontrivial applications / reductions.

- 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.

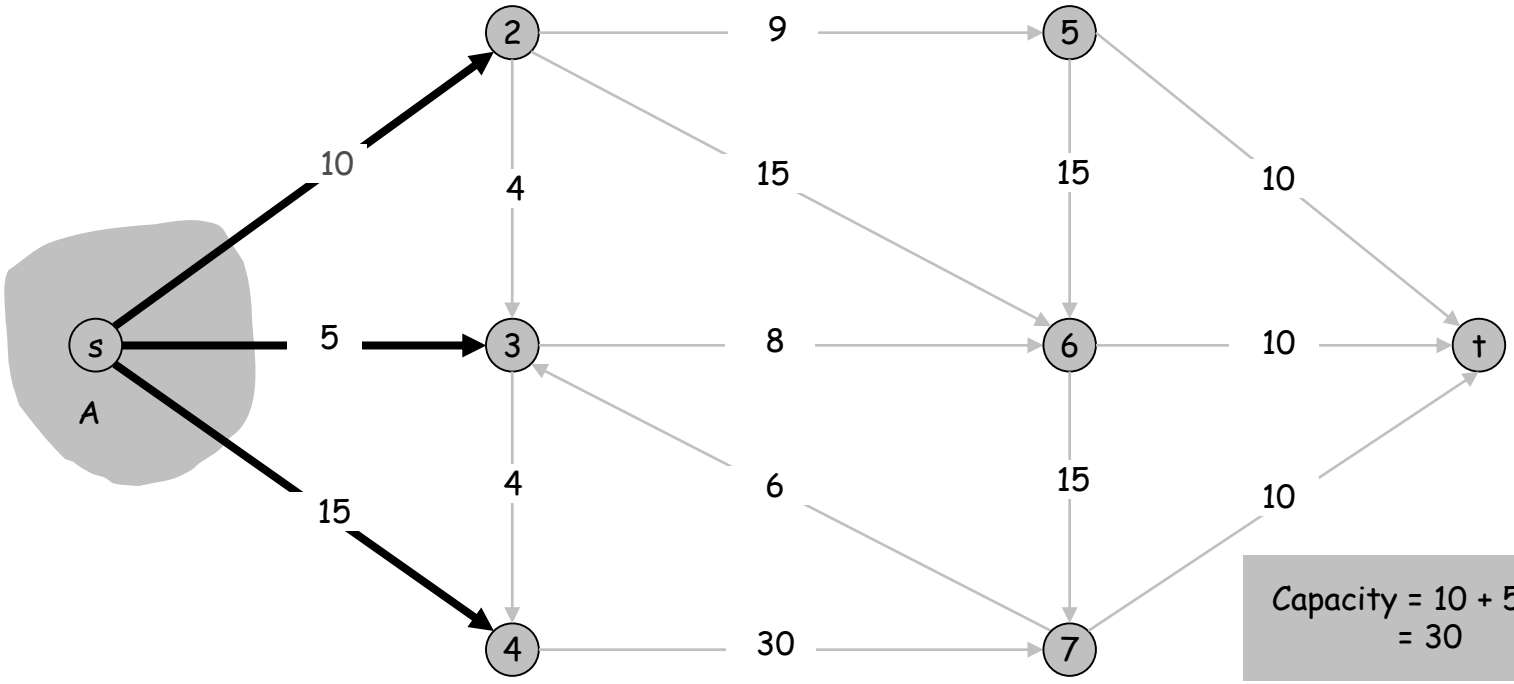
- Abstraction for material flowing through the edges.
- G = (V, E) = directed graph, no parallel edges.
- Two distinguished nodes: s = source, t = sink.
- c(e) = capacity of edge e.



Cuts

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

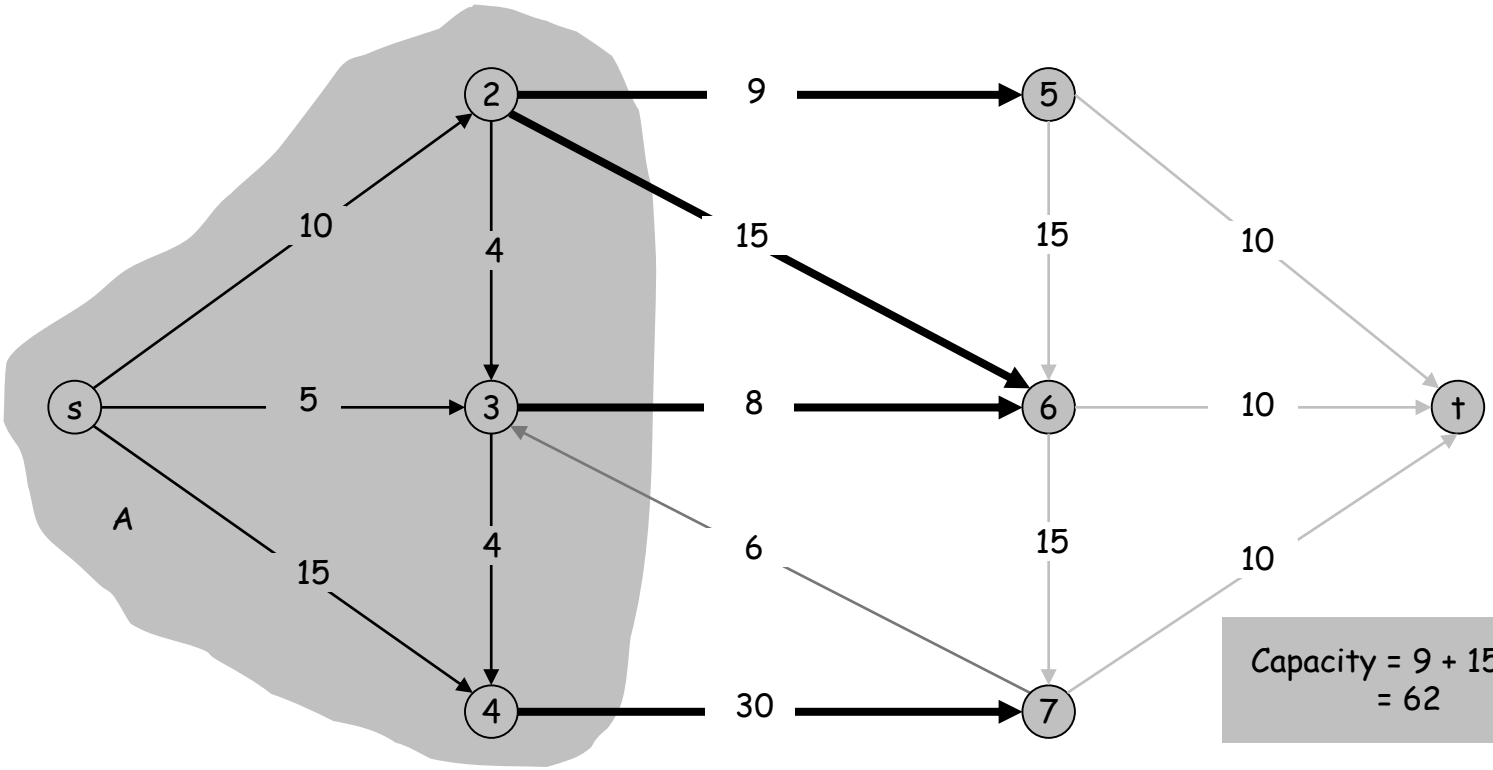
Def. The capacity of a cut (A, B) is: $cap(A, B) = \sum_{e \text{ out of } A} c(e)$



Cuts

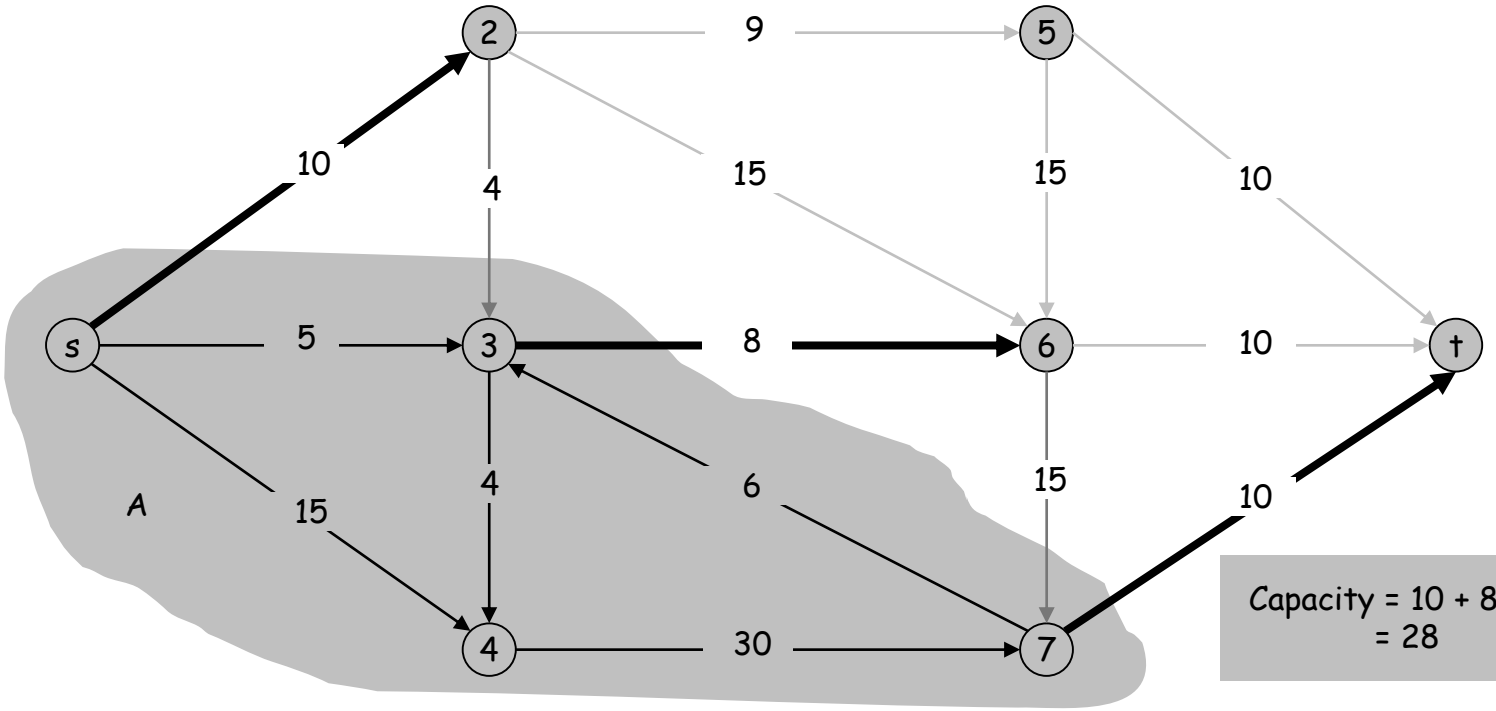
Def. An s-t cut is a partition (A, B) of V with $s \in A$ and $t \in B$.

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



Minimum Cut Problem

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

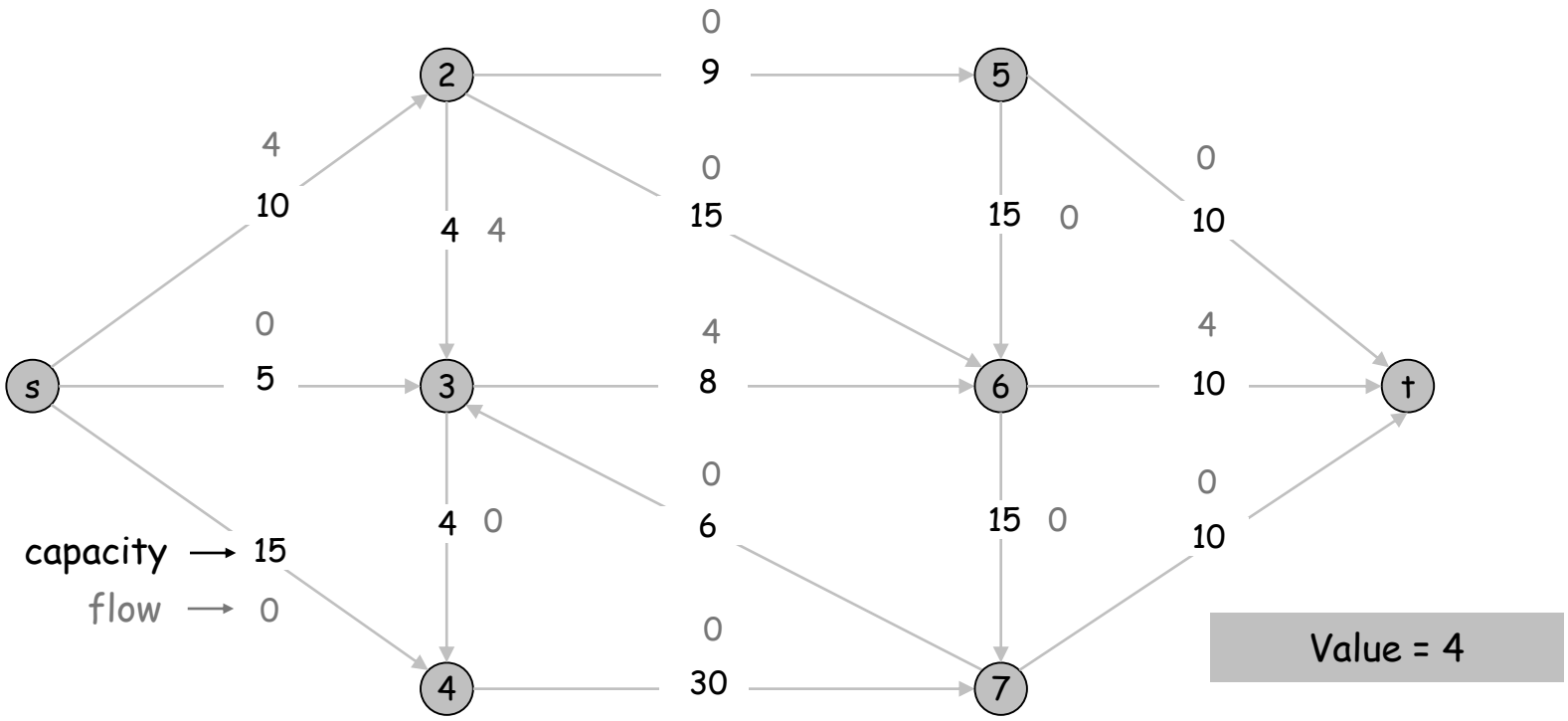


Flows

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

- For each e ∈ E: $0 \le f(e) \le c(e)$ [capacity]
- For each v ∈ 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: $v(f) = \sum_{e \text{ out of } s} f(e)$.

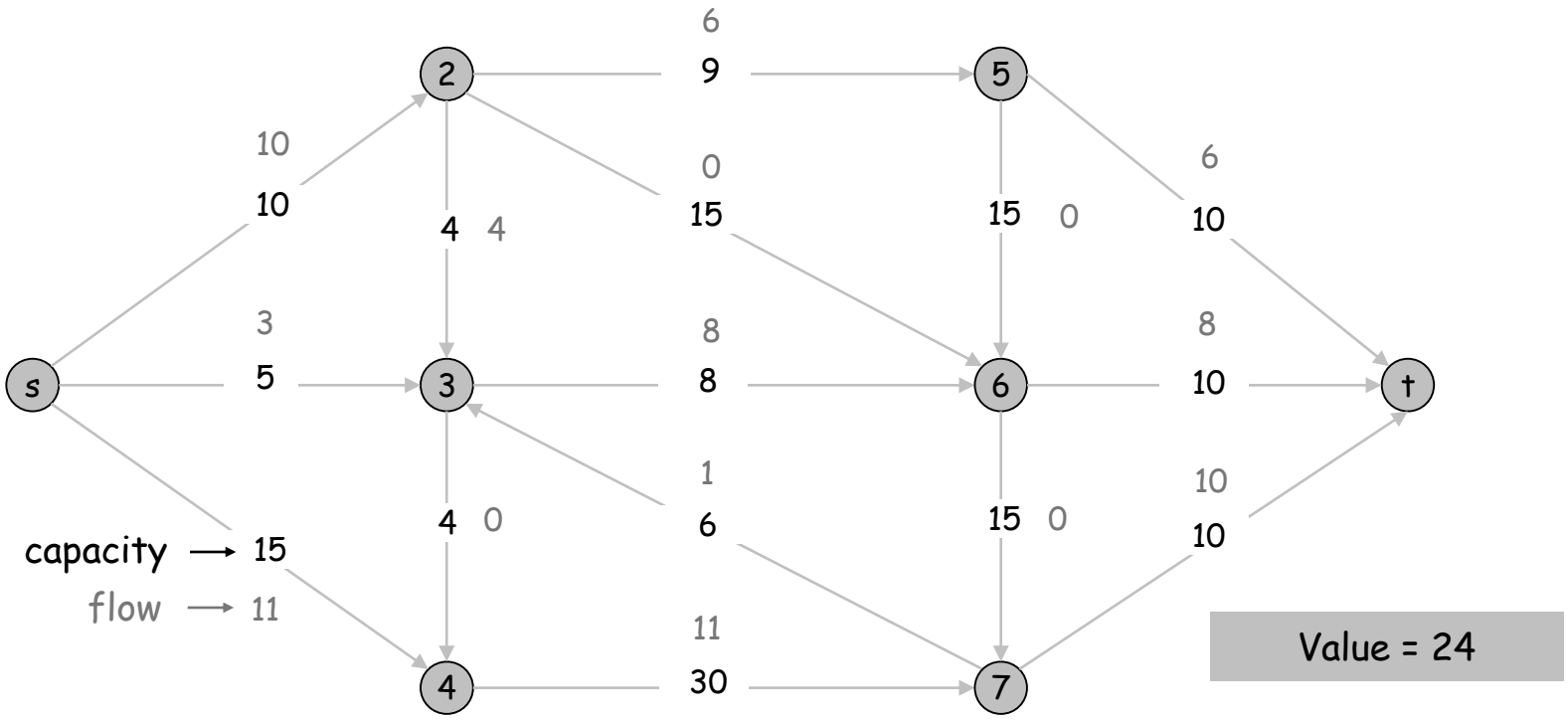


Flows

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

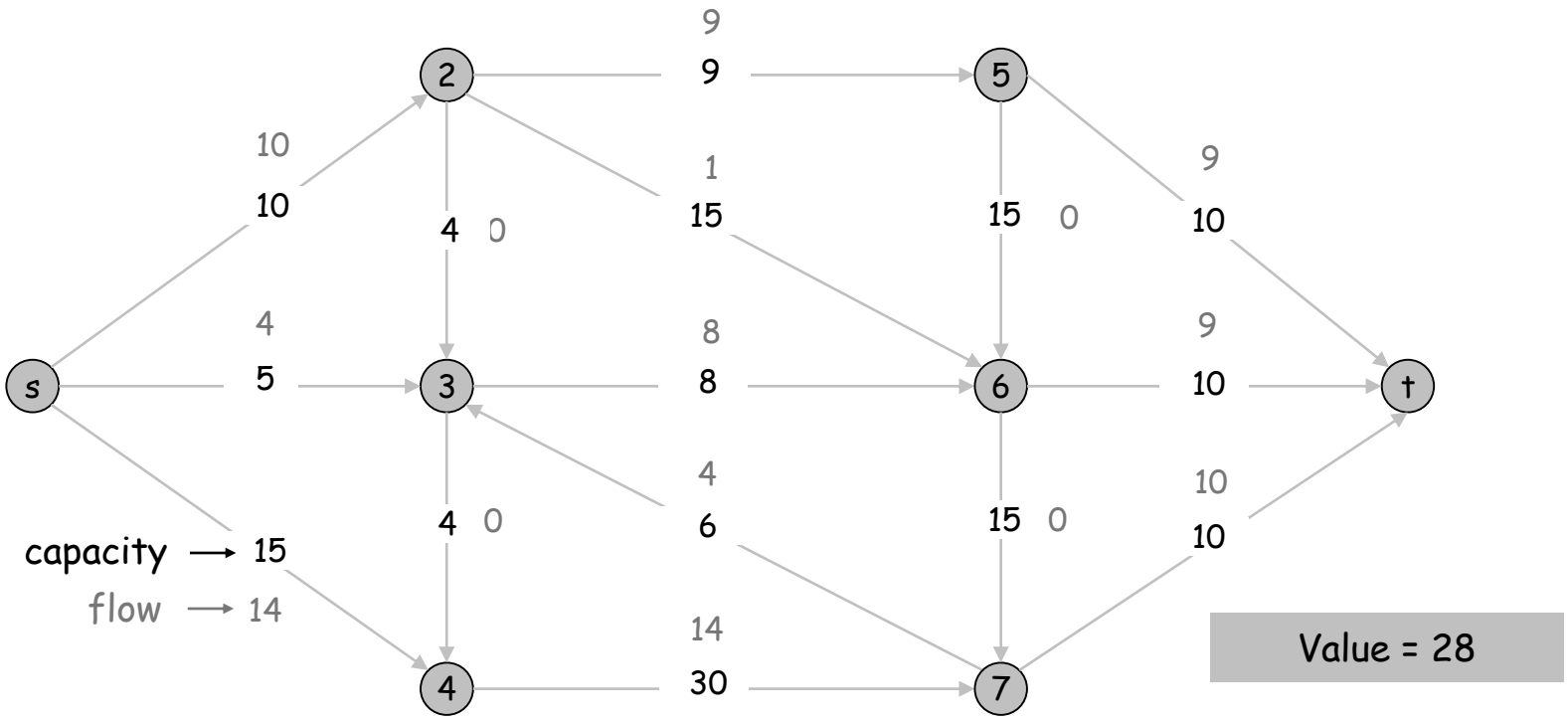
- For each e ∈ E: $0 \le f(e) \le c(e)$ [capacity]
- For each v ∈ V – {s, t}: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [conservation]

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



Maximum Flow Problem

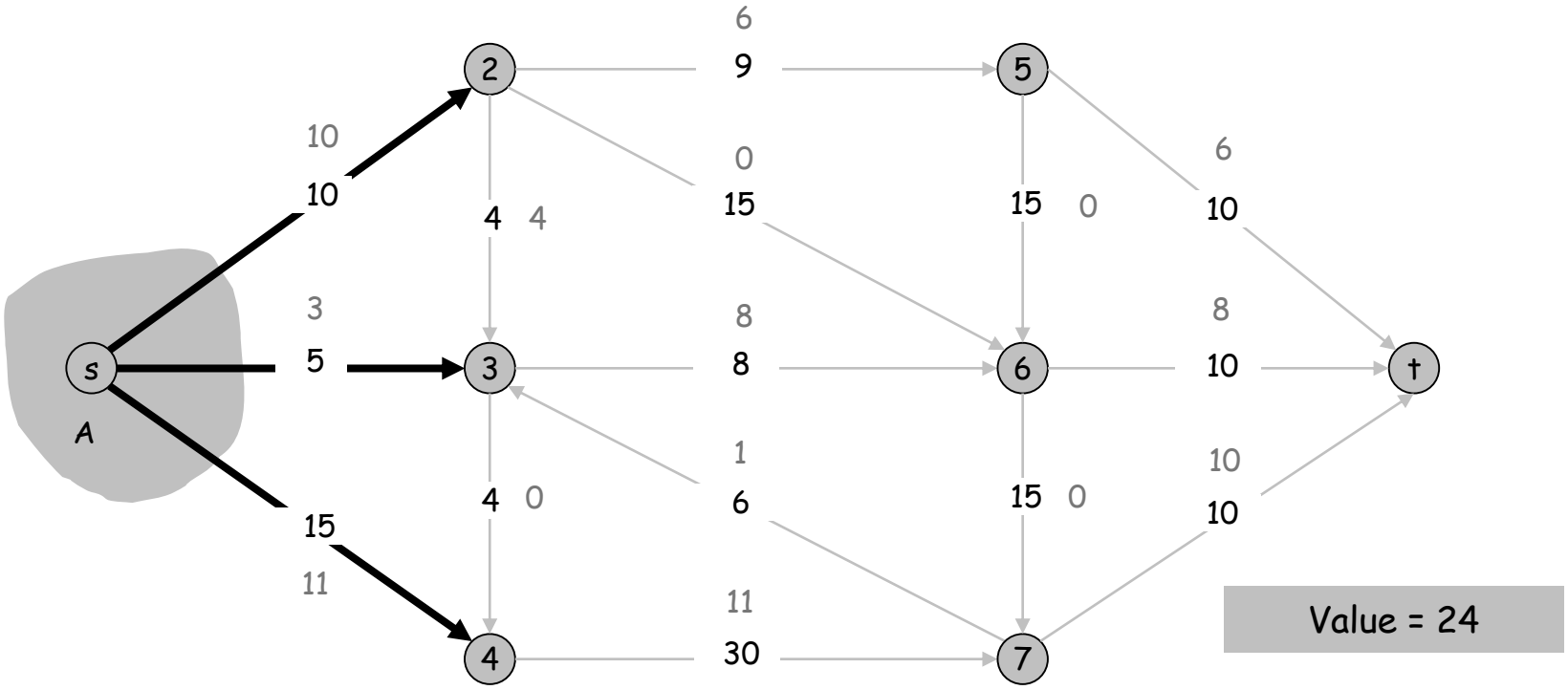
Max flow problem. Find s-t flow of maximum value.



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.

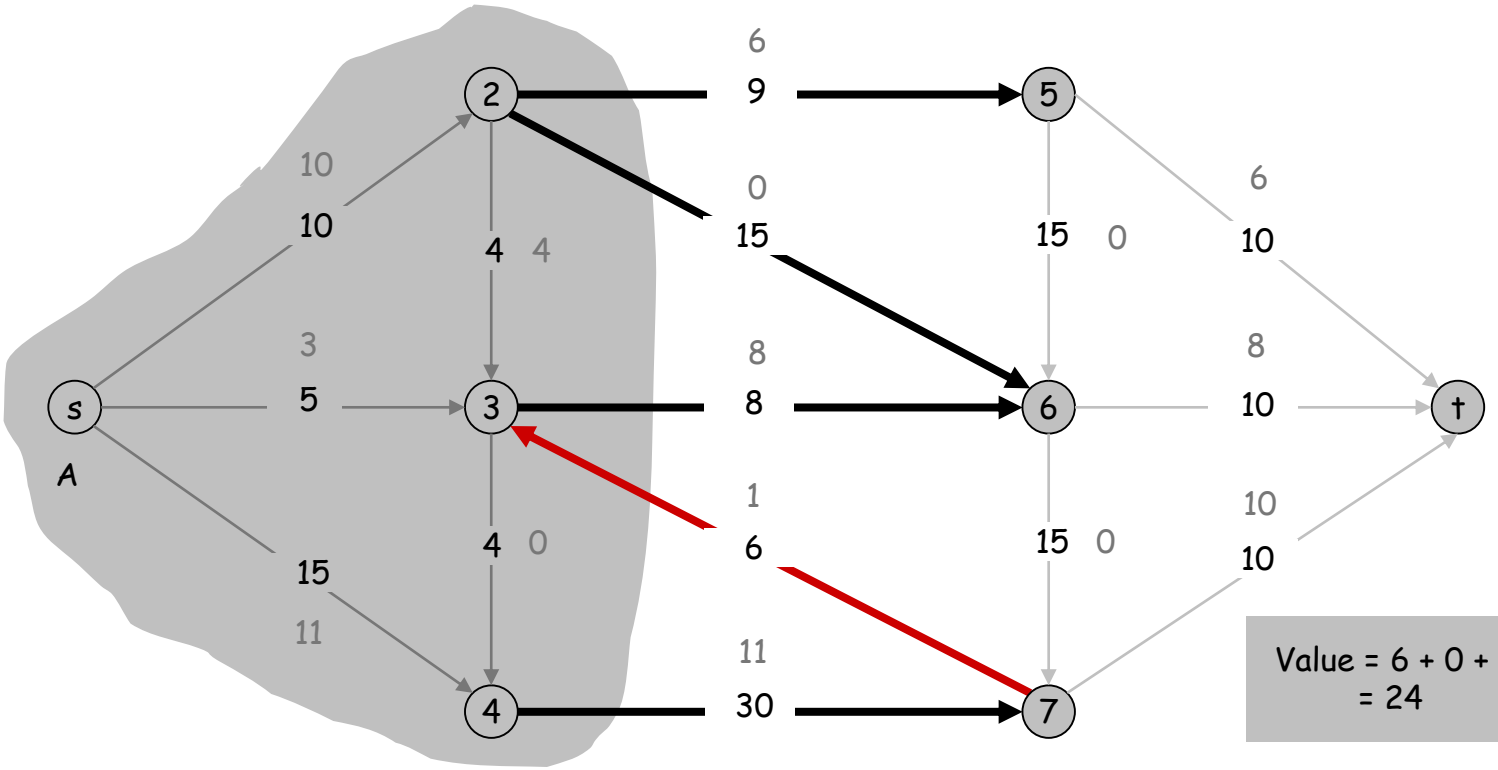
$$\sum_{e \text{ out of } A} f(e) \; - \sum_{e \text{ in to A}} f(e) \; = \; v(f)$$



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.

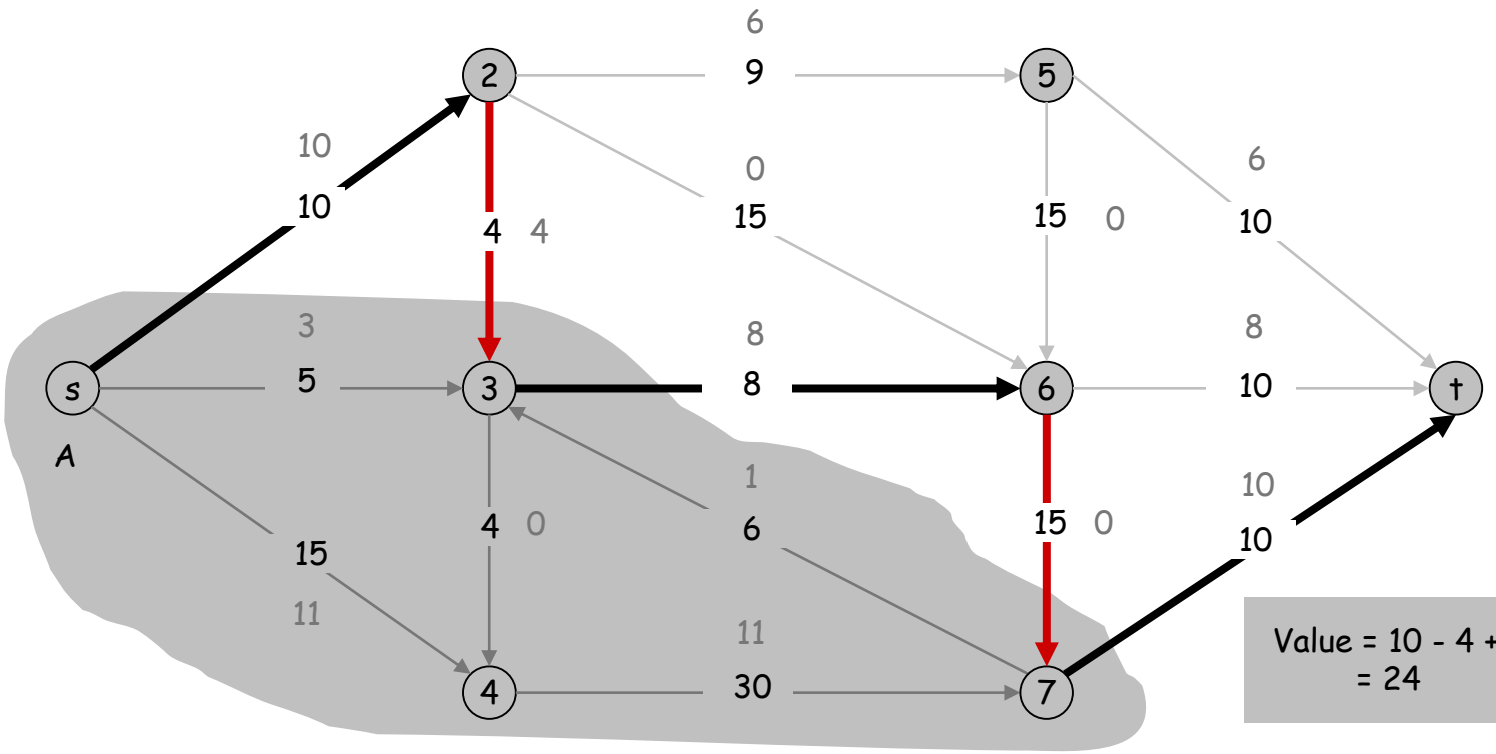
$$\sum_{e \text{ out of } A} f(e) \; - \sum_{e \text{ in to A}} f(e) \; = \; v(f)$$



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

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

Pf.

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

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

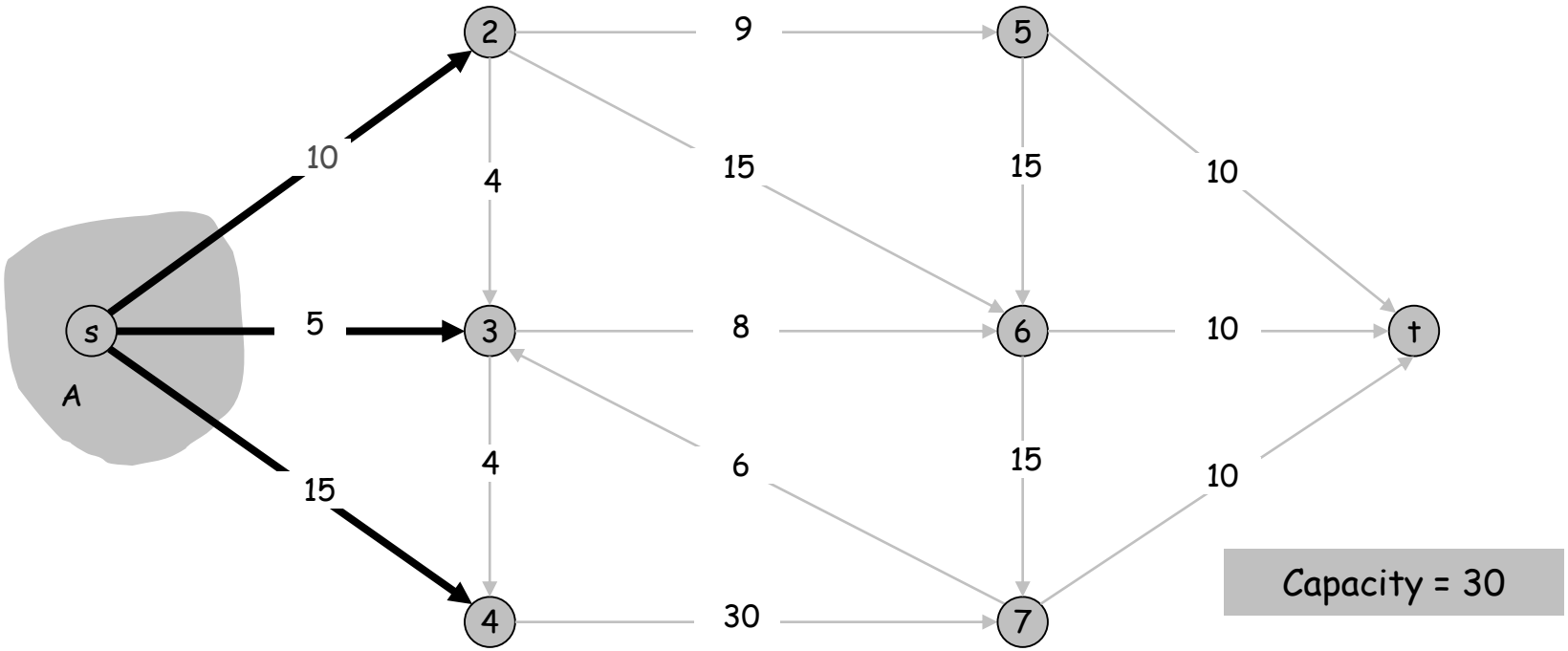
$$= \sum_{v \in A} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to v}} f(e) \right)$$

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

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.

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

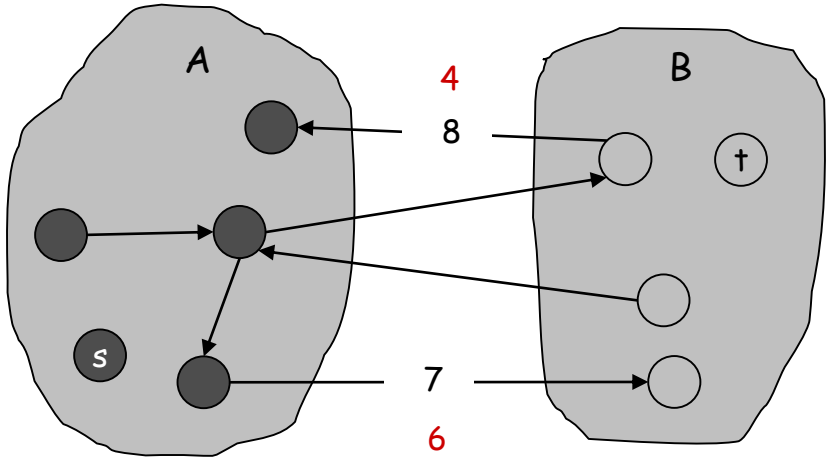


Flows and Cuts

Weak duality. Let f be any flow. Then, for any s-t cut (A, B) we have $v(f) \leq \text{cap}(A, B)$.

Pf.

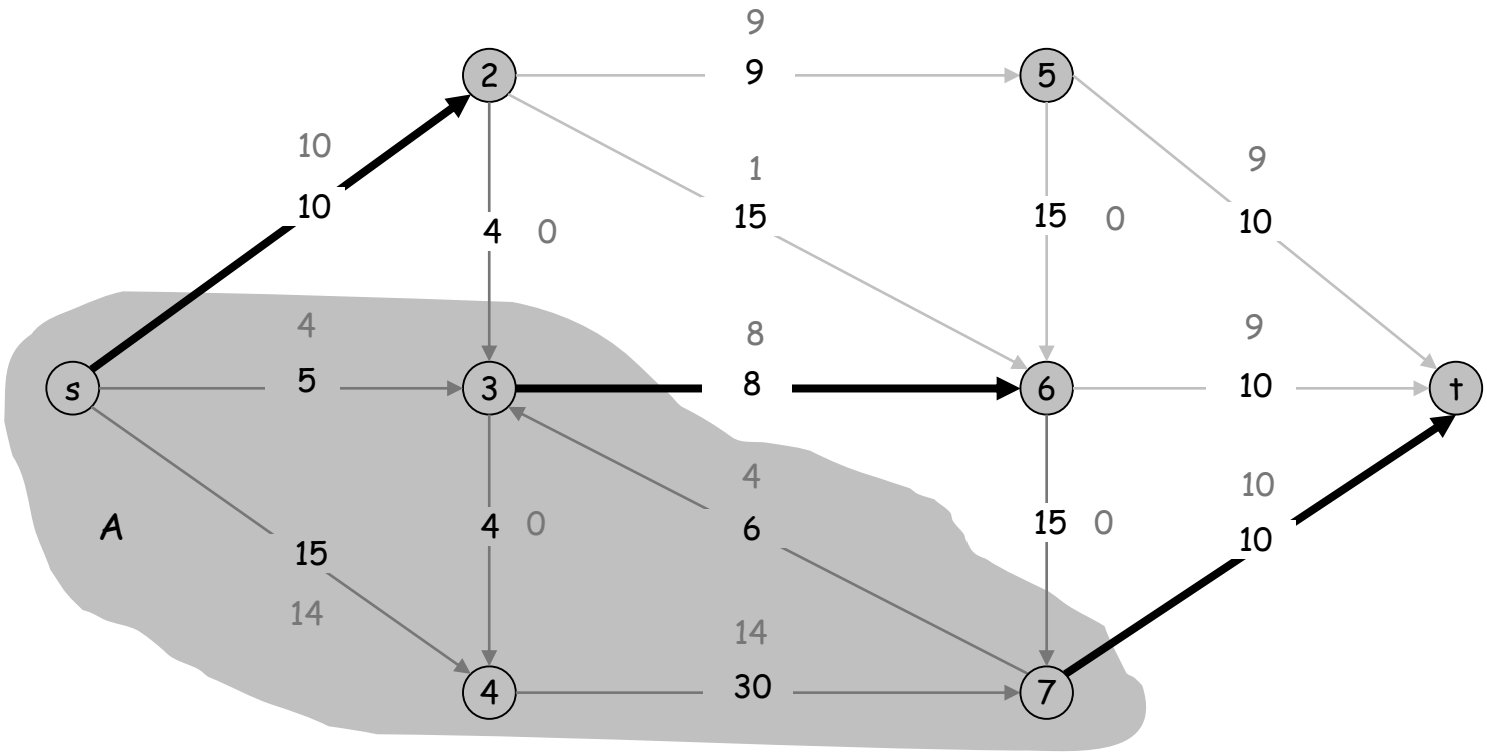
$$
\begin{aligned}
v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\
&\leq \sum_{e \text{ out of } A} f(e) \\
&\leq \sum_{e \text{ out of } A} c(e) \\
&= \text{cap}(A, B) \quad \blacksquare
\end{aligned}
$$



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.

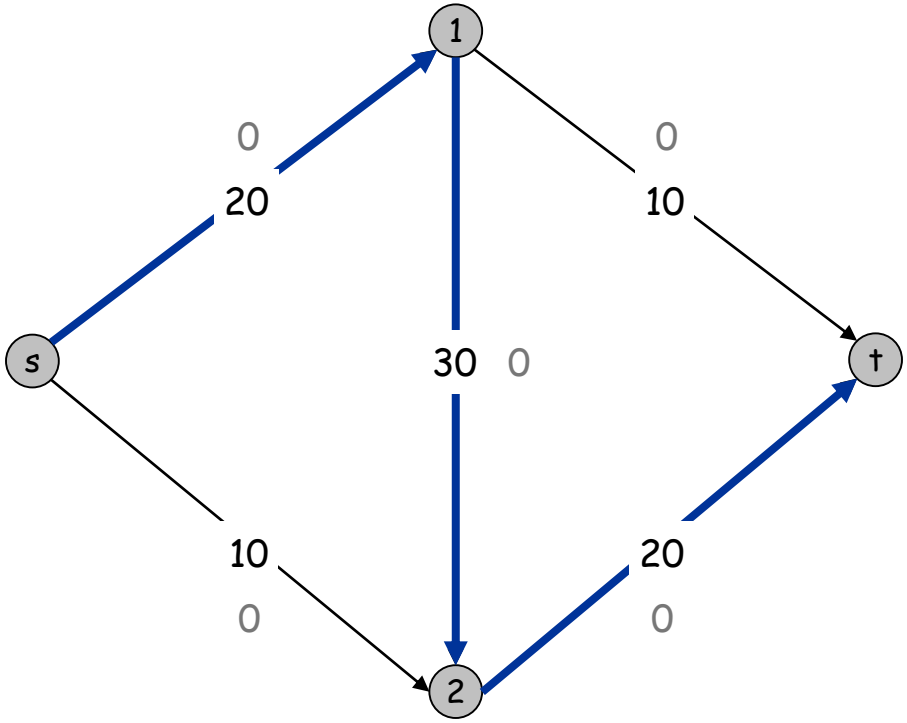
Value of flow = 28
Cut capacity = 28 $\Rightarrow$ Flow value $\leq$ 28



Towards a Max Flow Algorithm

Greedy algorithm.

- Start with f(e) = 0 for all edge e ∈ E.
- Find an s-t path P where each edge has f(e) < c(e).
- Augment flow along path P.
- Repeat until you get stuck.

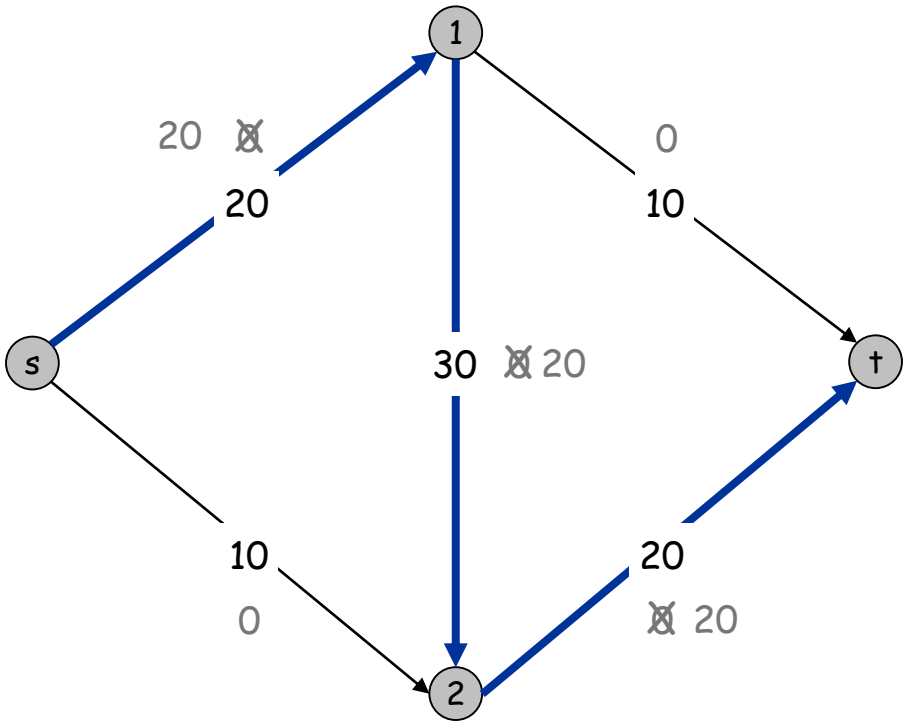


Flow value = 0

Towards a Max Flow Algorithm

Greedy algorithm.

- Start with f(e) = 0 for all edge e ∈ E.
- Find an s-t path P where each edge has f(e) < c(e).
- Augment flow along path P.
- Repeat until you get stuck.



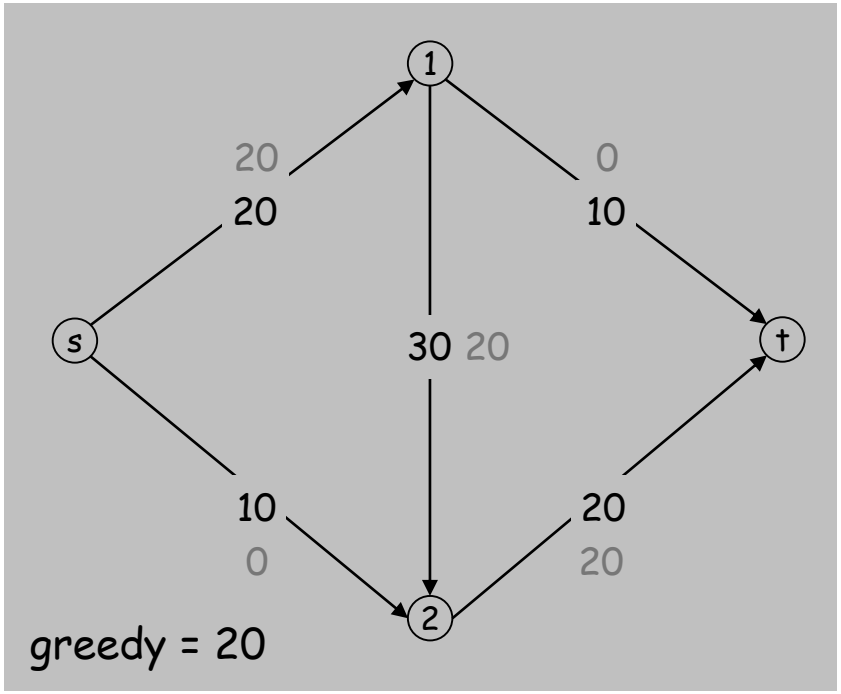
Flow value = 20

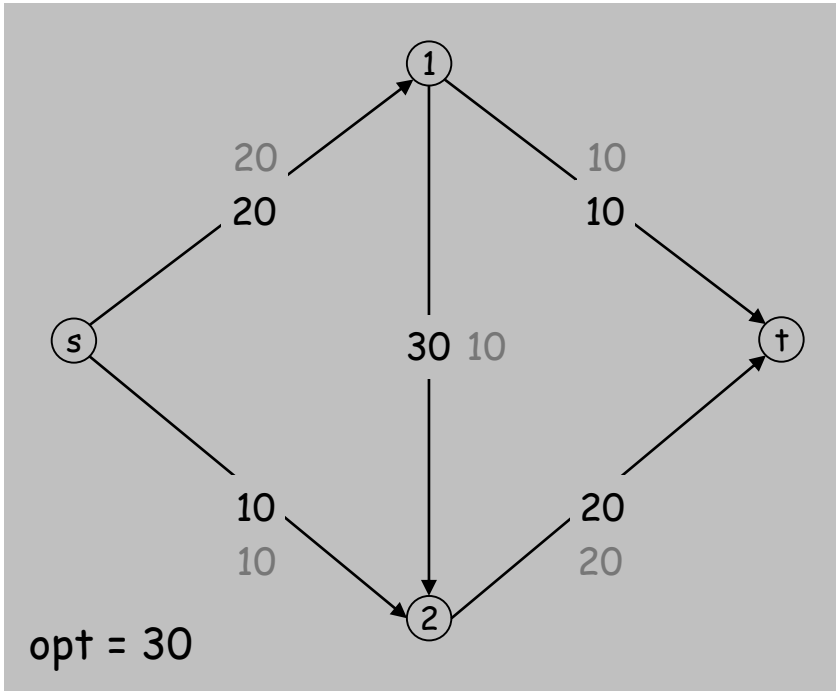
Towards a Max Flow Algorithm

Greedy algorithm.

- Start with f(e) = 0 for all edge e ∈ E.
- Find an s-t path P where each edge has f(e) < c(e).
- Augment flow along path P.
- Repeat until you get stuck.

↖ locally optimality ⇏ global optimality

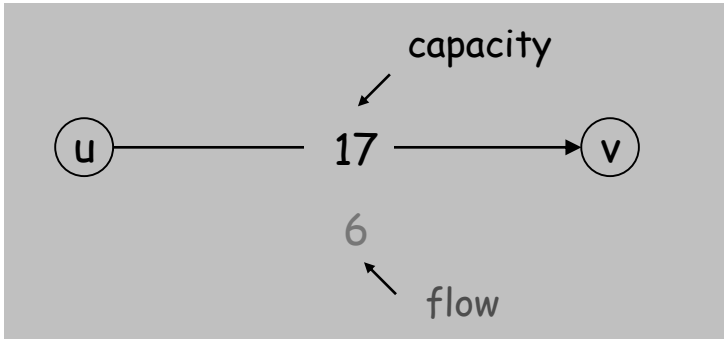




Residual Graph

Original edge: $e = (u, v) \in E$.

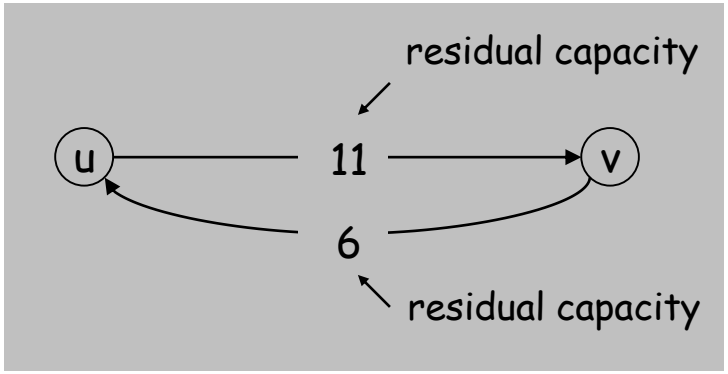
- Flow $f(e)$, capacity $c(e)$.



Residual edge.

- "Undo" flow sent.
- $e = (u, v)$ and $e^R = (v, u)$.
- Residual capacity:

$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^R \in E \end{cases}$$



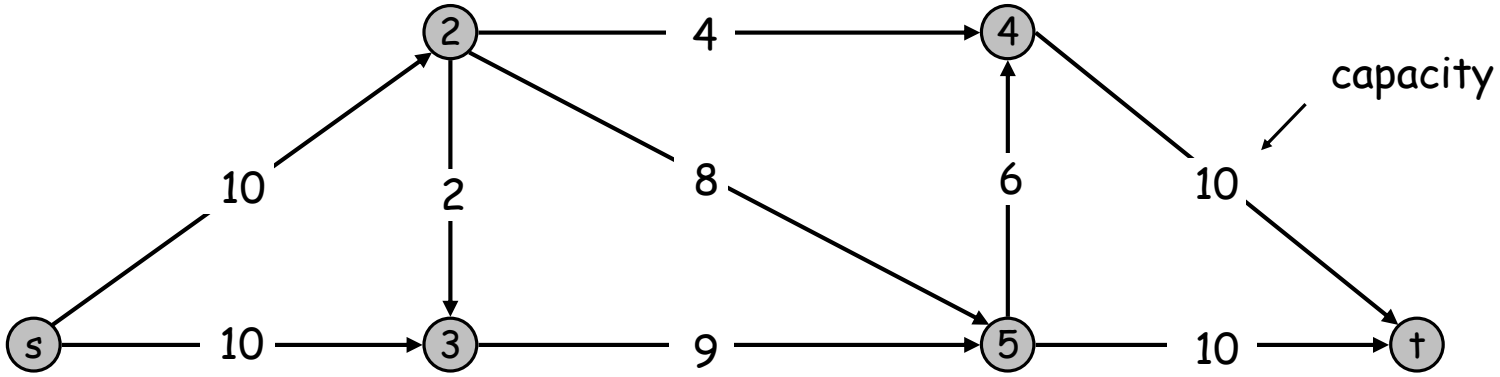
Residual graph: $G_f = (V, E_f)$.

- Residual edges with positive residual capacity.
- $E_f = \{e : f(e) < c(e)\} \cup \{e^R : f(e) > 0\}$.

7. Ford-Fulkerson Demo

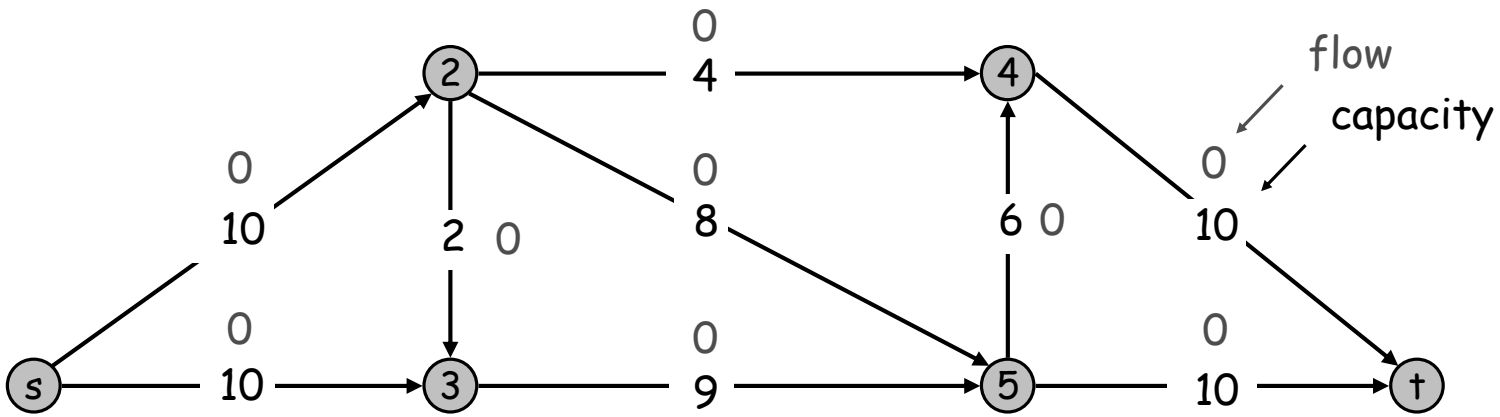
Ford-Fulkerson Algorithm

G:



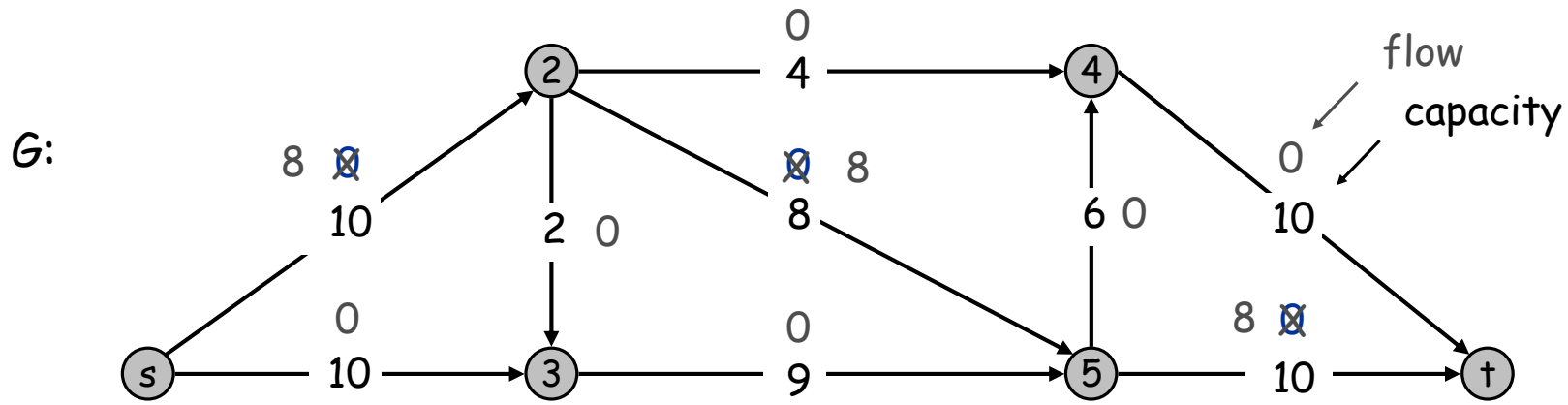
Ford-Fulkerson Algorithm

G:

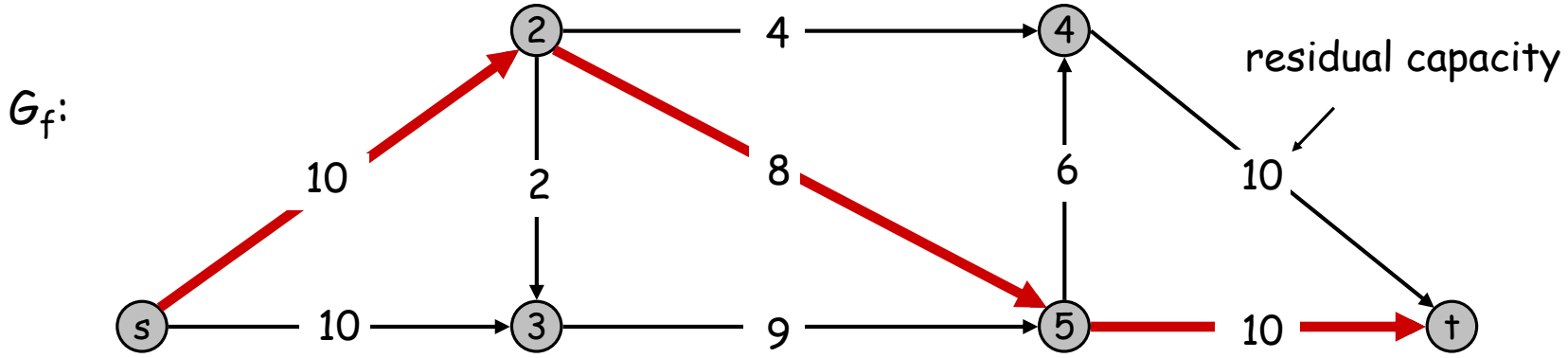


Flow value = 0

Ford-Fulkerson Algorithm

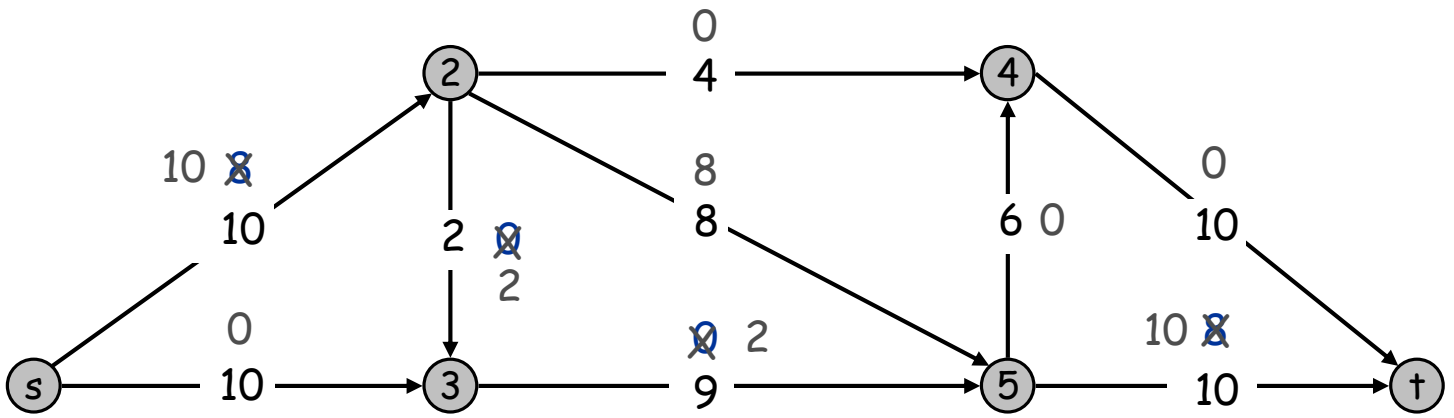


Flow value = 0



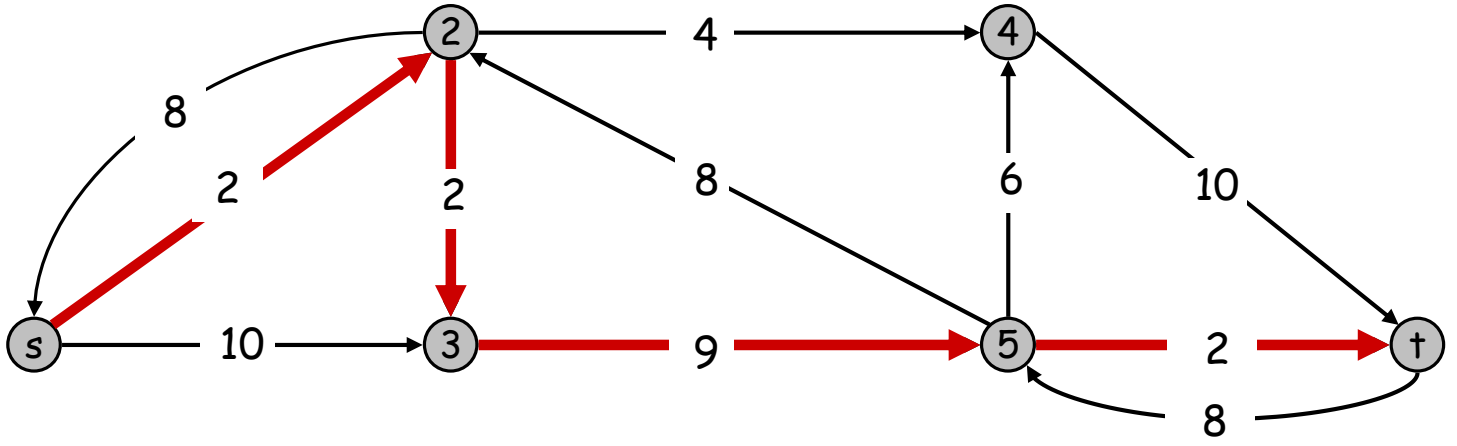
Ford-Fulkerson Algorithm

G:



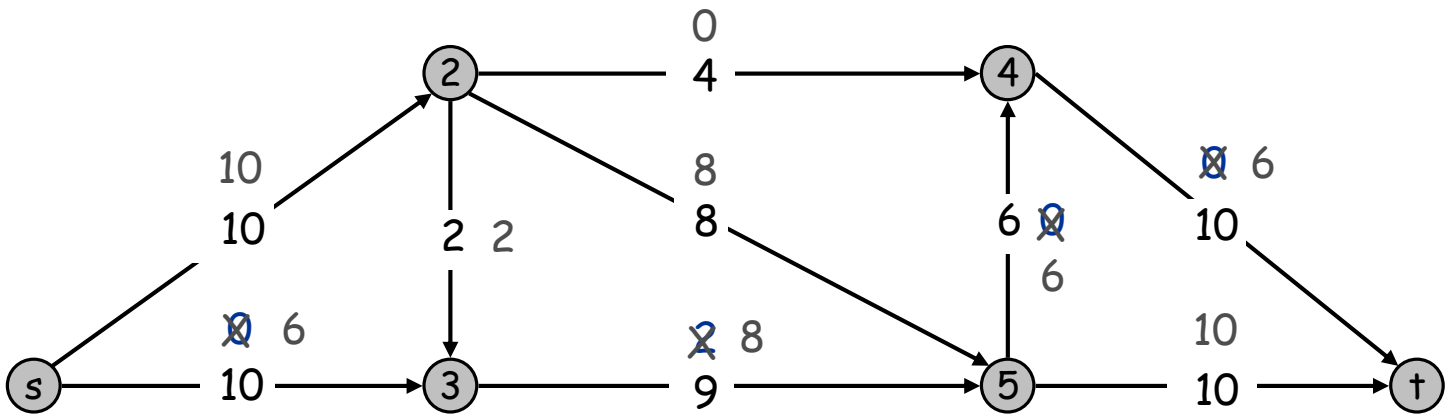
Flow value = 8

G_f:



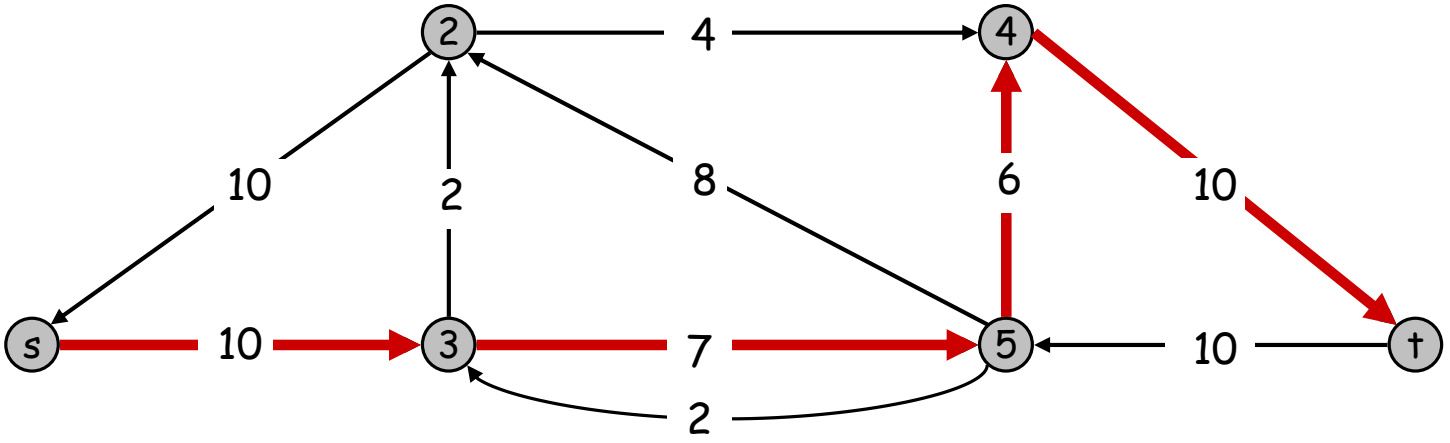
Ford-Fulkerson Algorithm

G:



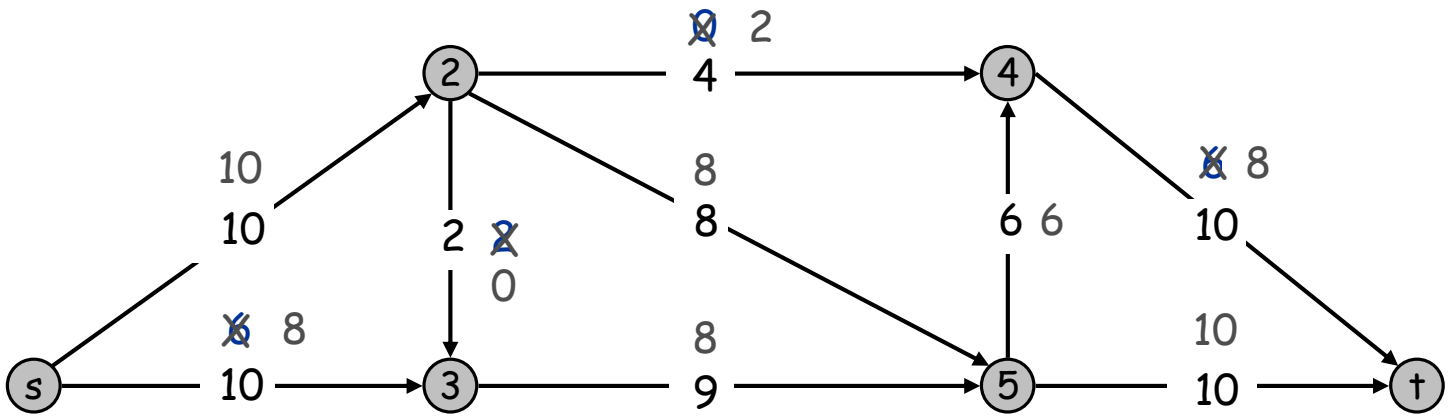
Flow value = 10

G_f:



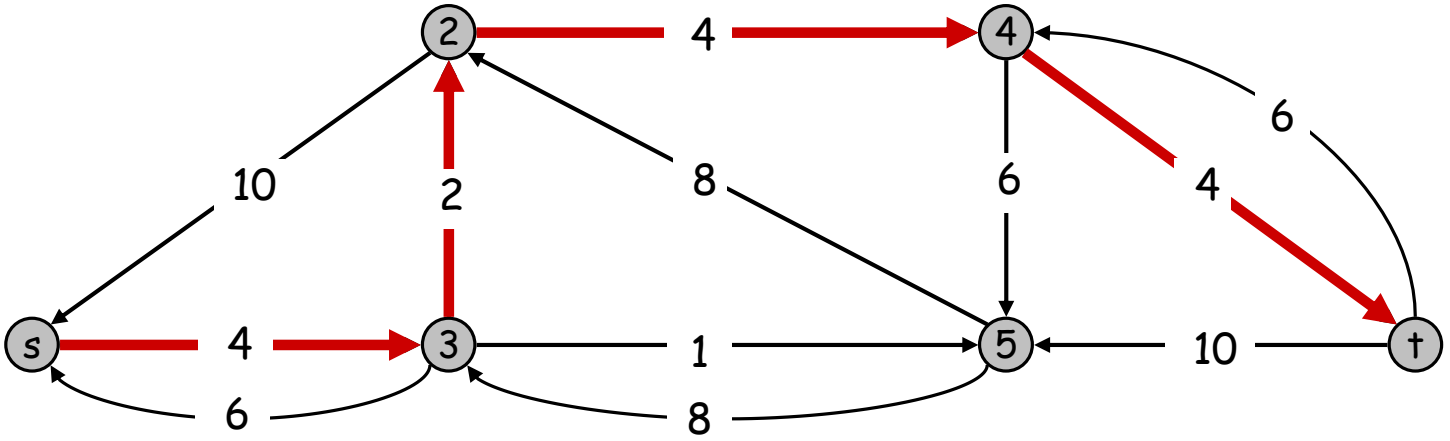
Ford-Fulkerson Algorithm

G:



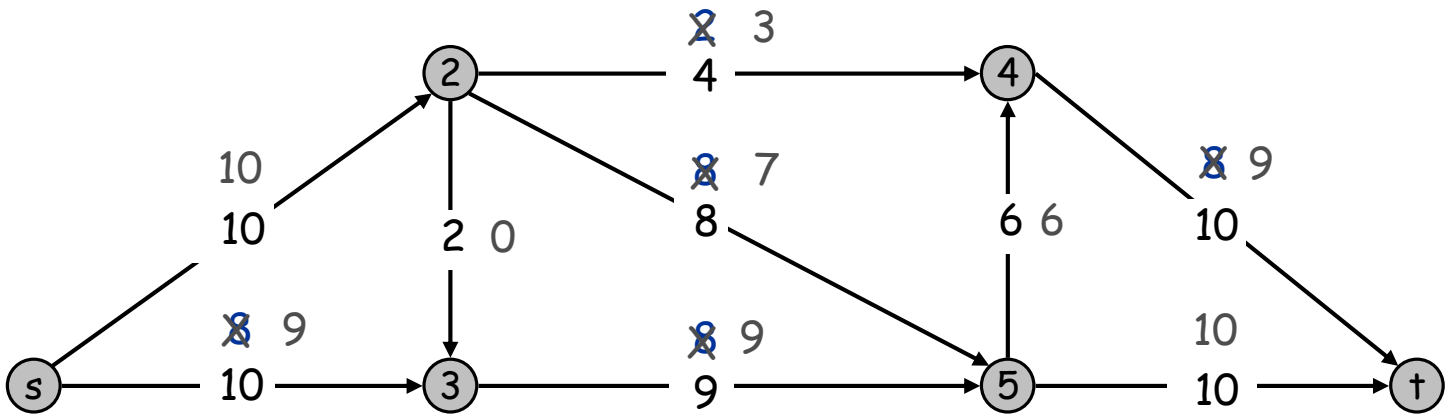
Flow value = 16

G_f:



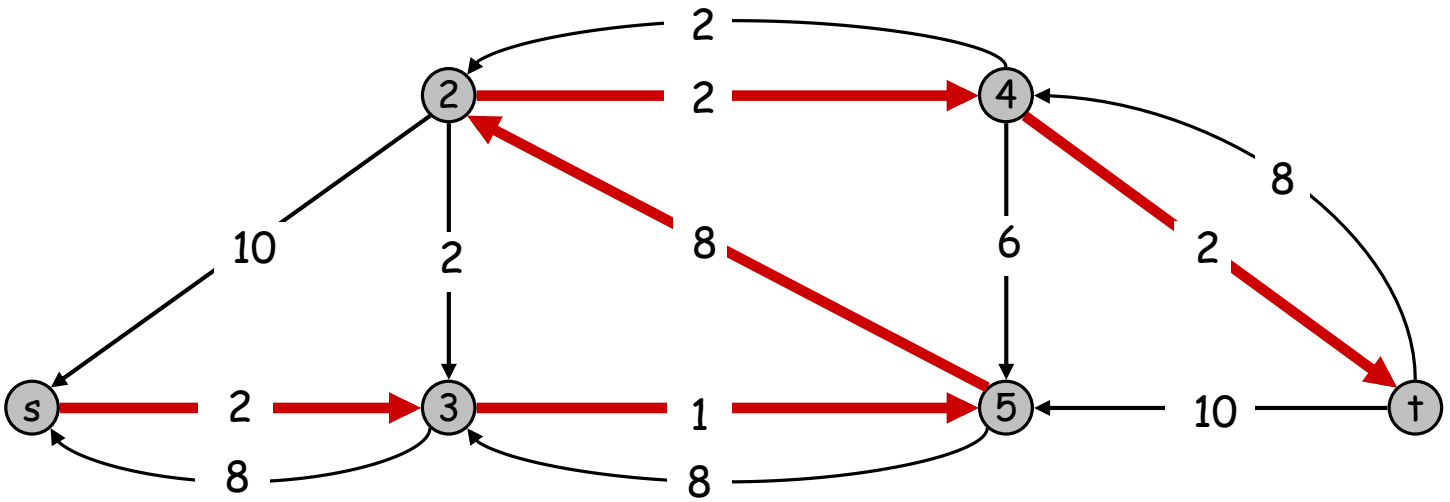
Ford-Fulkerson Algorithm

G:



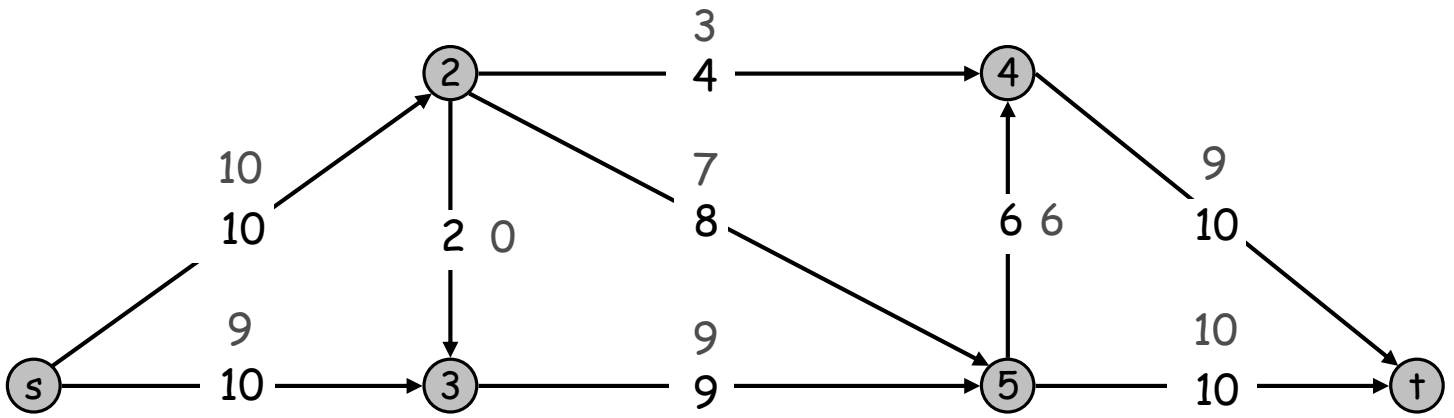
Flow value = 18

G_f:



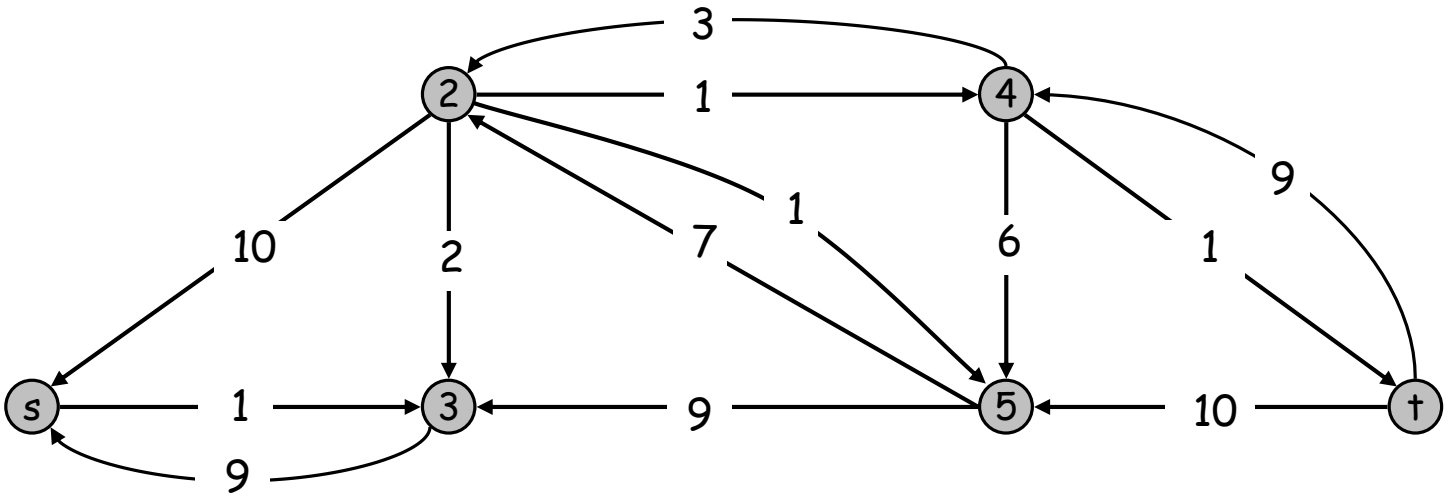
Ford-Fulkerson Algorithm

G:

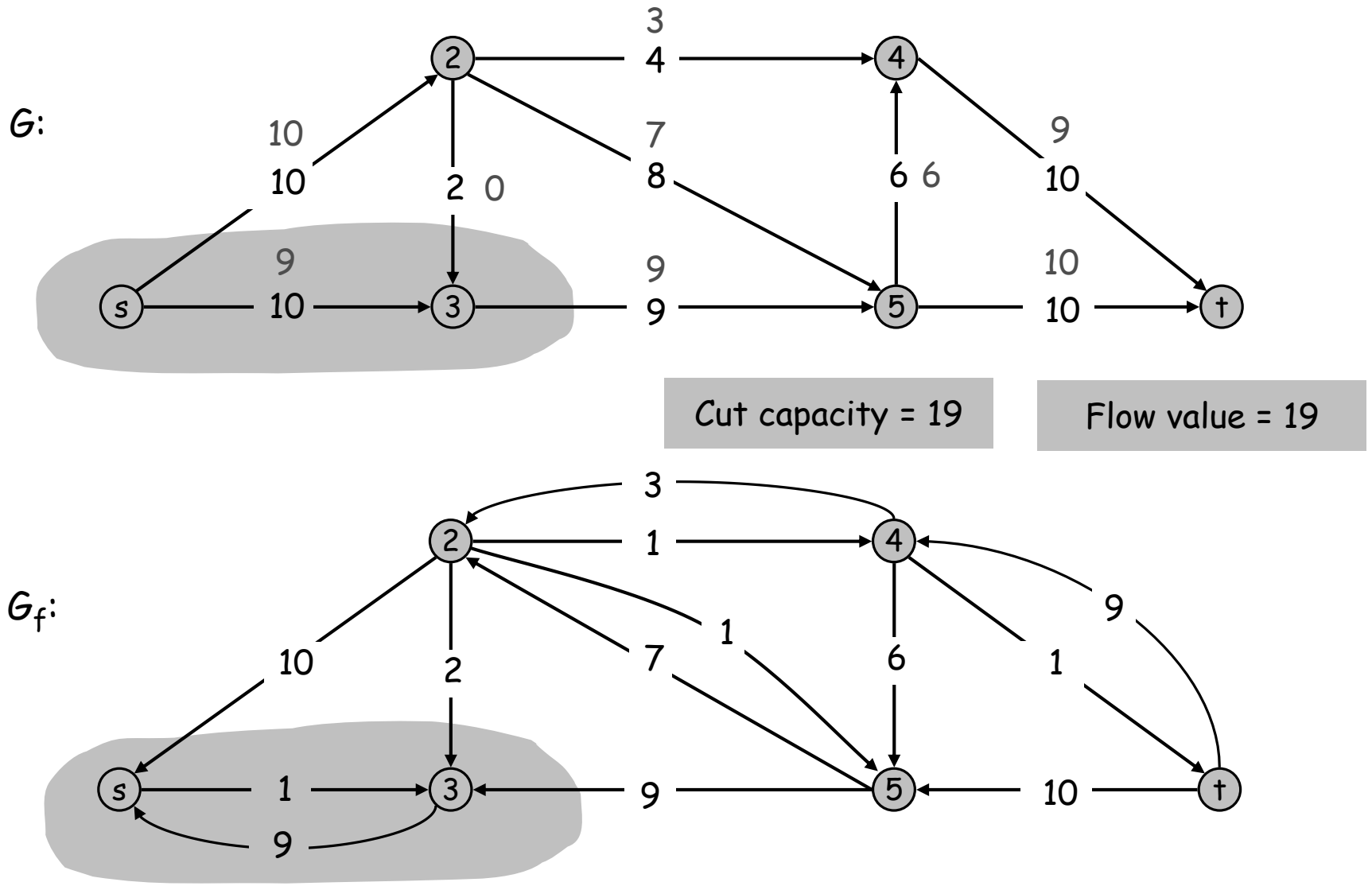


Flow value = 19

G_f:



Ford-Fulkerson Algorithm



Augmenting Path Algorithm

```
Augment(f, c, P) {
   b ← bottleneck(P)
   foreach e ∈ P {
      if (e ∈ E) f(e) ← f(e) + b      forward edge
      else       f(e^R) ← f(e^R) - b  reverse edge
   }
   return f
}
```

```
Ford-Fulkerson(G, s, t, c) {
   foreach e ∈ E  f(e) ← 0
   G_f ← residual graph

   while (there exists augmenting path P) {
      f ← Augment(f, c, P)
      update G_f
   }
   return f
}
```

Max-Flow Min-Cut Theorem

Augmenting path theorem. Flow f is a max flow iff there are no augmenting paths.

Max-flow min-cut theorem. [Elias-Feinstein-Shannon 1956, Ford-Fulkerson 1956] The value of the max flow is equal to the value of the min cut.

Pf. We prove both simultaneously by showing TFAE (the following are equivalent):

(i) There exists a cut (A, B) such that v(f) = cap(A, B).
(ii) Flow f is a max flow.
(iii) There is no augmenting path relative to f.

(i) $\Rightarrow$ (ii) This was the corollary to weak duality lemma.

(ii) $\Rightarrow$ (iii) We show contrapositive.

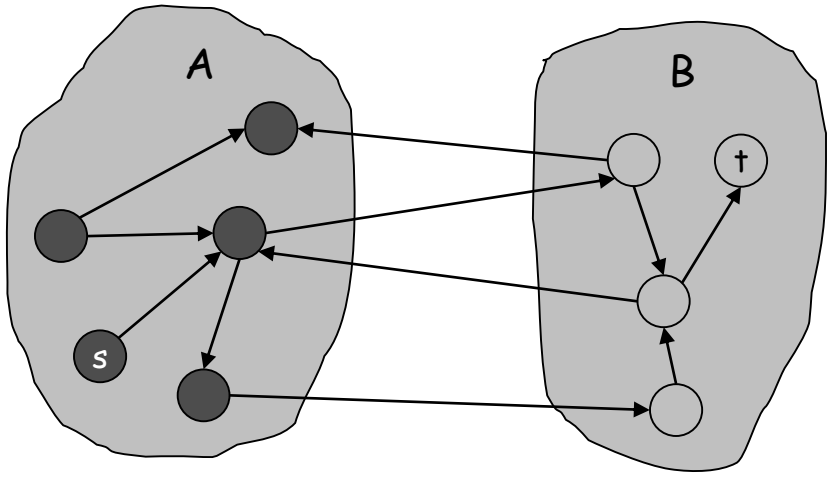
- Let f be a flow. If there exists an augmenting path, then we can improve f by sending flow along path.

Proof of Max-Flow Min-Cut Theorem

(iii) $\Rightarrow$ (i)

- Let f be a flow with no augmenting paths.
- Let A be set of vertices reachable from s in residual graph.
- By definition of A, $s \in A$.
- By definition of f, $t \notin A$.

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



original network

Running Time

Assumption. All capacities are integers between 1 and C.

Invariant. Every flow value $f(e)$ and every residual capacity $c_f(e)$ remains an integer throughout the algorithm.

Theorem. The algorithm terminates in at most $v(f^*) \leq nC$ iterations.
Pf. Each augmentation increase value by at least 1. ▪

Corollary. If $C = 1$, Ford-Fulkerson runs in $O(mn)$ time.

Integrality theorem. If all capacities are integers, then there exists a max flow f for which every flow value $f(e)$ is an integer.
Pf. Since algorithm terminates, theorem follows from invariant. ▪

Kleinberg HW 7.3

The figure shows a flow network on which an s-t flow has been computed. The capacity of each edge appears as a label next to the edge, and the numbers in the boxes give the amount of flow sent on each edge (Edges without boxed numbers have no flow being sent on them).

What is the value of this flow?

Is this a maximum (s,t) flow in this graph?

Kleinberg HW 7.3

The figure shows a flow network on which an s-t flow has been computed. The capacity of each edge appears as a label next to the edge, and the numbers in the boxes give the amount of flow sent on each edge (Edges without boxed numbers have no flow being sent on them).

What is the value of this flow? 10

Is this a maximum (s,t) flow in this graph? No.

Kleinberg HW 7.3

The figure shows a flow network on which an s-t flow has been computed. The capacity of each edge appears as a label next to the edge, and the numbers in the boxes give the amount of flow sent on each edge (Edges without boxed numbers have no flow being sent on them).

(b) Find a minimum s-t cut in the flow network pictured, and also say what its capacity is.

Kleinberg HW 7.3

The figure shows a flow network on which an s-t flow has been computed. The capacity of each edge appears as a label next to the edge, and the numbers in the boxes give the amount of flow sent on each edge (Edges without boxed numbers have no flow being sent on them).

(b) Find a minimum s-t cut in the flow network pictured, and also say what its capacity is.

Min cut is: ({s,a,b,c},{d,t}); the capacity is 11.

Kleinberg HW 7.4

Decide whether you think the following state is true or false. If it is true, give a short explanation. If it is false give a counter-example.

Let G be an arbitrary flow network, with a source s, a sink t, and a positive integers capacity c_e on every edge; and let (A,B) be a minimum s-t cut with respect to these capacities. Now suppose we add 1 to every capacity; then (A,B) is still a minimum s-t cut with respect to these new capacities.

Kleinberg HW 7.4

Decide whether you think the following state is true or false. If it is true, give a short explanation. If it is false give a counter-example.

Let G be an arbitrary flow network, with a source s, a sink t, and a positive integers capacity c_e on every edge; and let (A,B) be a minimum s-t cut with respect to these capacities. Now suppose we add 1 to every capacity; then (A,B) is still a minimum s-t cut with respect to these new capacities.

Answer: False. Consider a graph with nodes s,v,w,t, and edges: (s,v), (v,w), (w,t), capacities of 2 on (s,v) and (w,t) and a capacity of 1 on (v,w). The maximum flow has value 1, and does not saturate the edge out of s.

Kleinberg HW 7.7

Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible base stations. We'll suppose there are n clients, with the position of each client specified by its (x,y) coordinates in the plane. There are also k base stations; the position of each of these is specified by (x,y) coordinates as well.

For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways.

There is a range parameter r: a client can only be connected to a base station that is within distance r. There is also a load parameter L: no more than L clients can be connected to any single base station.

Your goal is to design a polynomial-time algorithm for the following problem. Give the positions of a set of base stations, be connected simultaneously to a base station, subject to the range and load conditions in the previous paragraph.

Kleinberg HW 7.7

There is a range parameter r: a client can only be connected to a base station that is within distance r. There is also a load parameter L: no more than L clients can be connected to any single base station.

Your goal is to design a polynomial-time algorithm for the following problem. Give the positions of a set of base stations, be connected simultaneously to a base station, subject to the range and load conditions in the previous paragraph.

Kleinberg HW 7.7

There is a range parameter r: a client can only be connected to a base station that is within distance r. There is also a load parameter L: no more than L clients can be connected to any single base station.

Your goal is to design a polynomial-time algorithm for the following problem. Give the positions of a set of base stations, be connected simultaneously to a base station, subject to the range and load conditions in the previous paragraph.

Answer: (getting started) Build the flow network – a node v_i for each client, a node w_j for each base and an edge (v_i,w_j) of capacity 1 if client i is in range of base station j. Lastly, expand this graph by adding the source node "s" that connects to each of the client nodes with capacity 1, and we connect each of the base stations to a sink node "t" by an edge of capacity L.

Now show that there is a feasible way to connect all clients to base stations if and only if there is an s-t flow of value n. Running time is time required to solve max-flow on graph with $O(n+k)$ nodes and $O(nk)$ edges