Chapter 7

Network Flow

Algorithm Design

JON KLEINBERG · ÉVA TARDOS

Soviet Rail Network, 1955
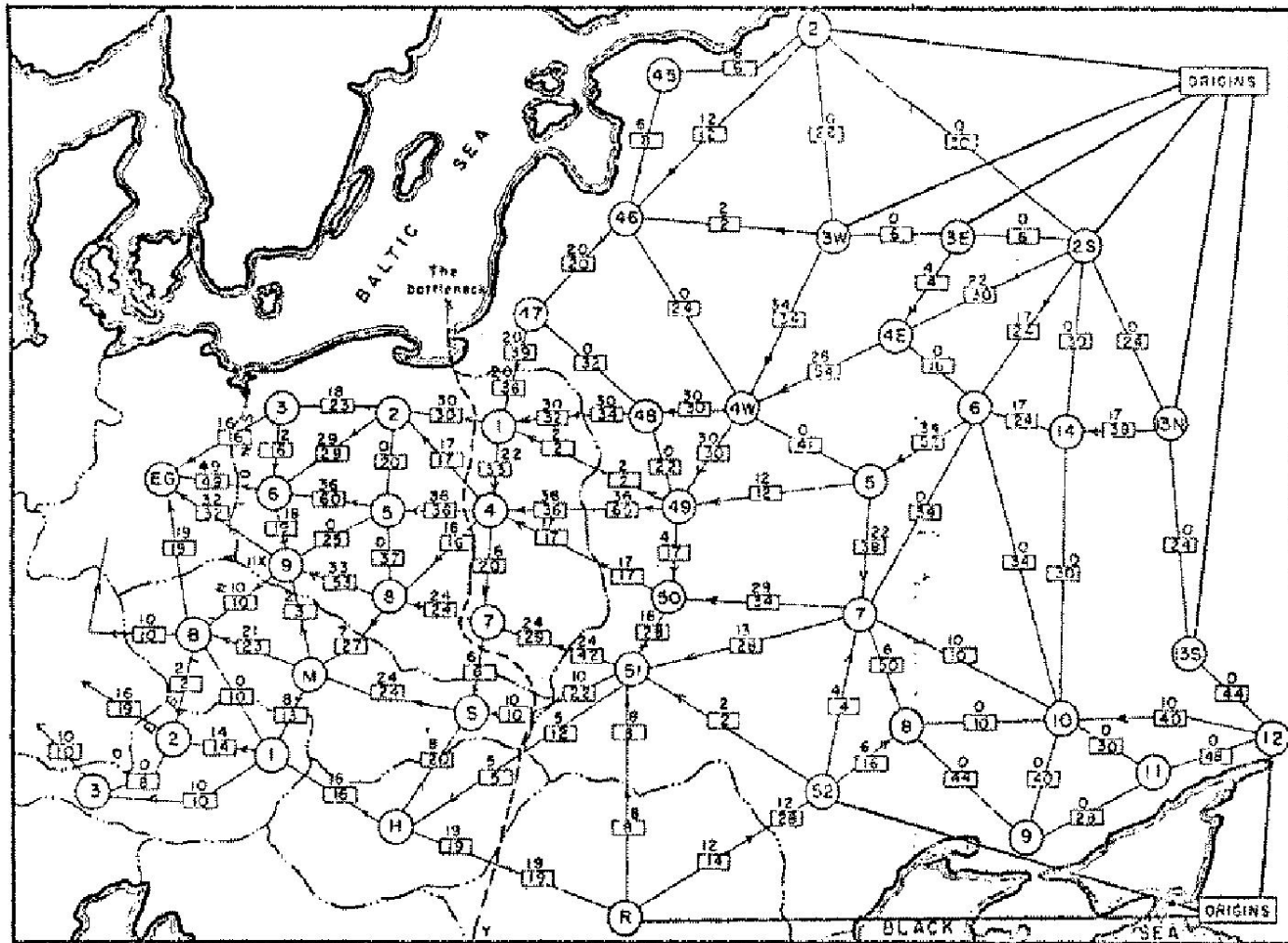
Reference: *On the history of the transportation and maximum flow problems.*
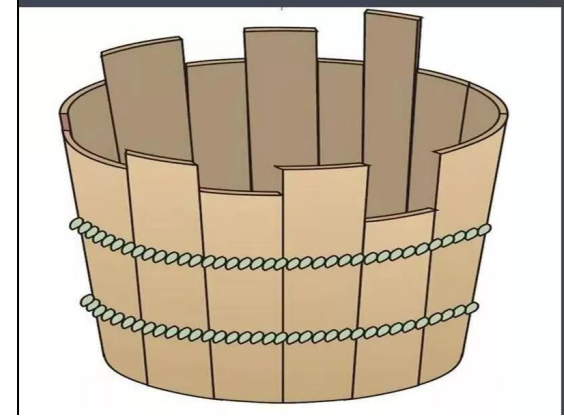Alexander Schrijver in Math Programming, 91: 3, 2002.

Maximum Flow and Minimum Cut

Max flow and min cut.

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

the barrel effect
the capacity of a barrel is determined not by the longest wooden bars, but by the shortest

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.
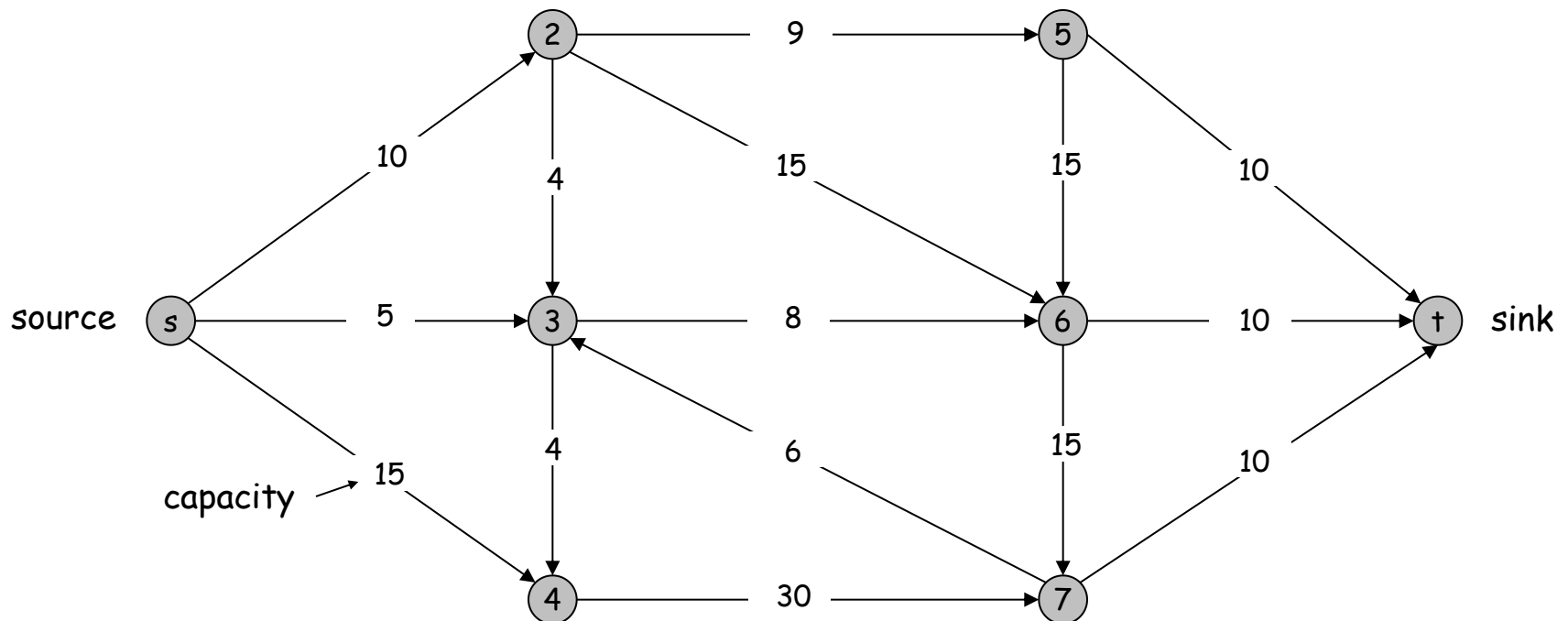- Multi-camera scene reconstruction.
- Many many more …

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.

source

sink

capacity

2   9   5

10   15   15   10

4

s   5   3   8   6   10   t

4   6   15   10

15

4   30   7

Cuts

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

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

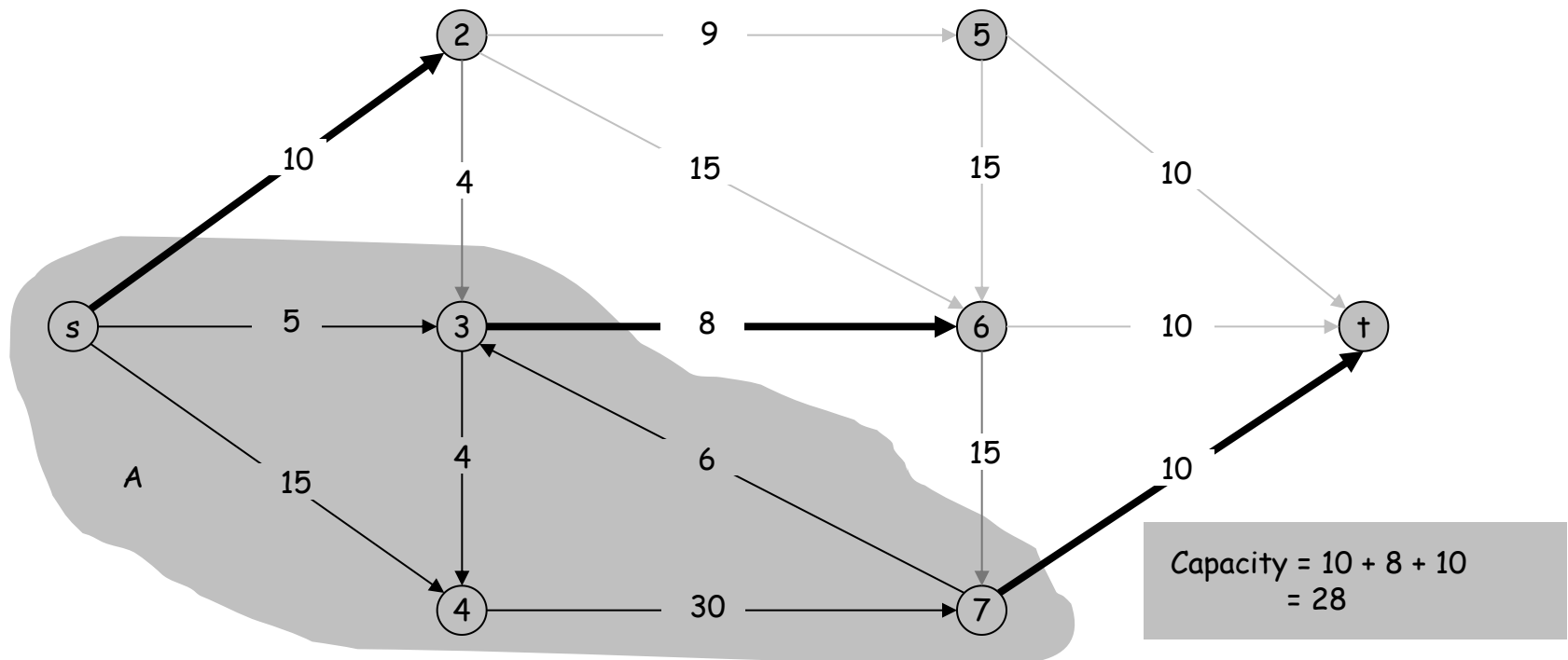Capacity = 9 + 15 + 8 + 30
= 62
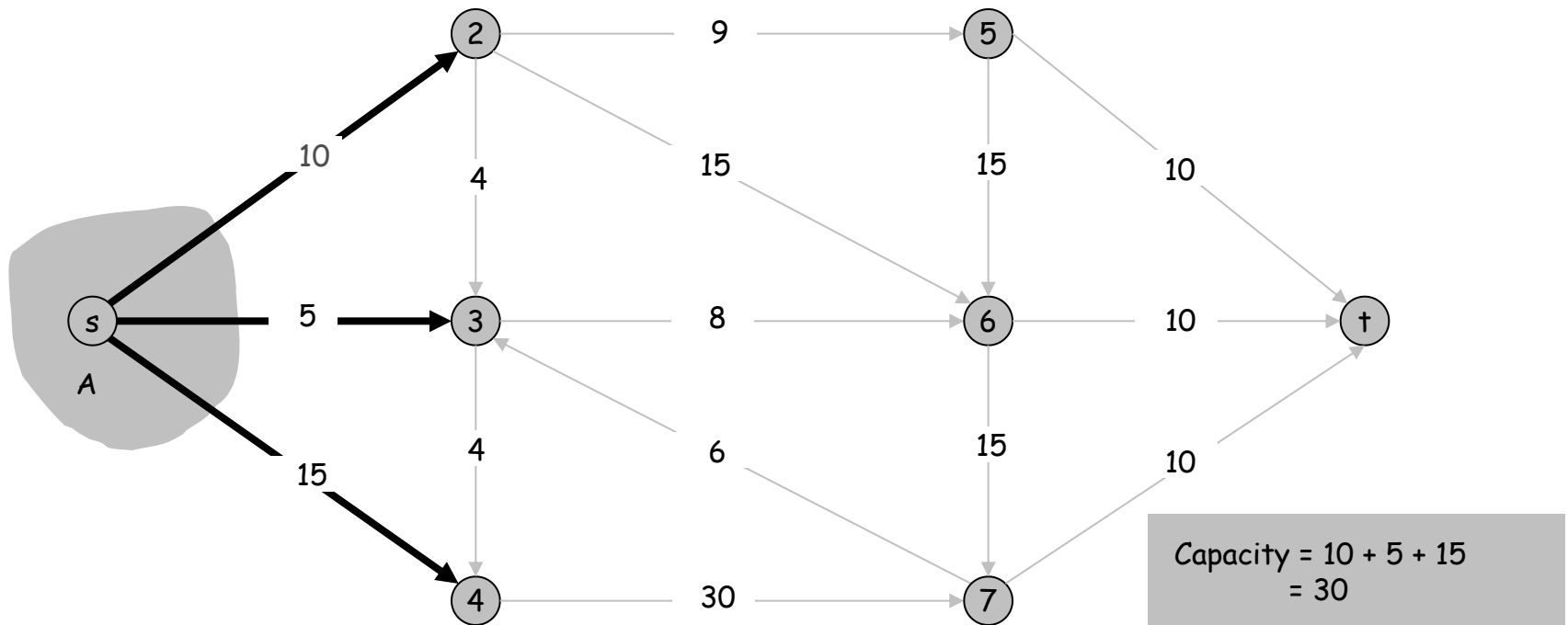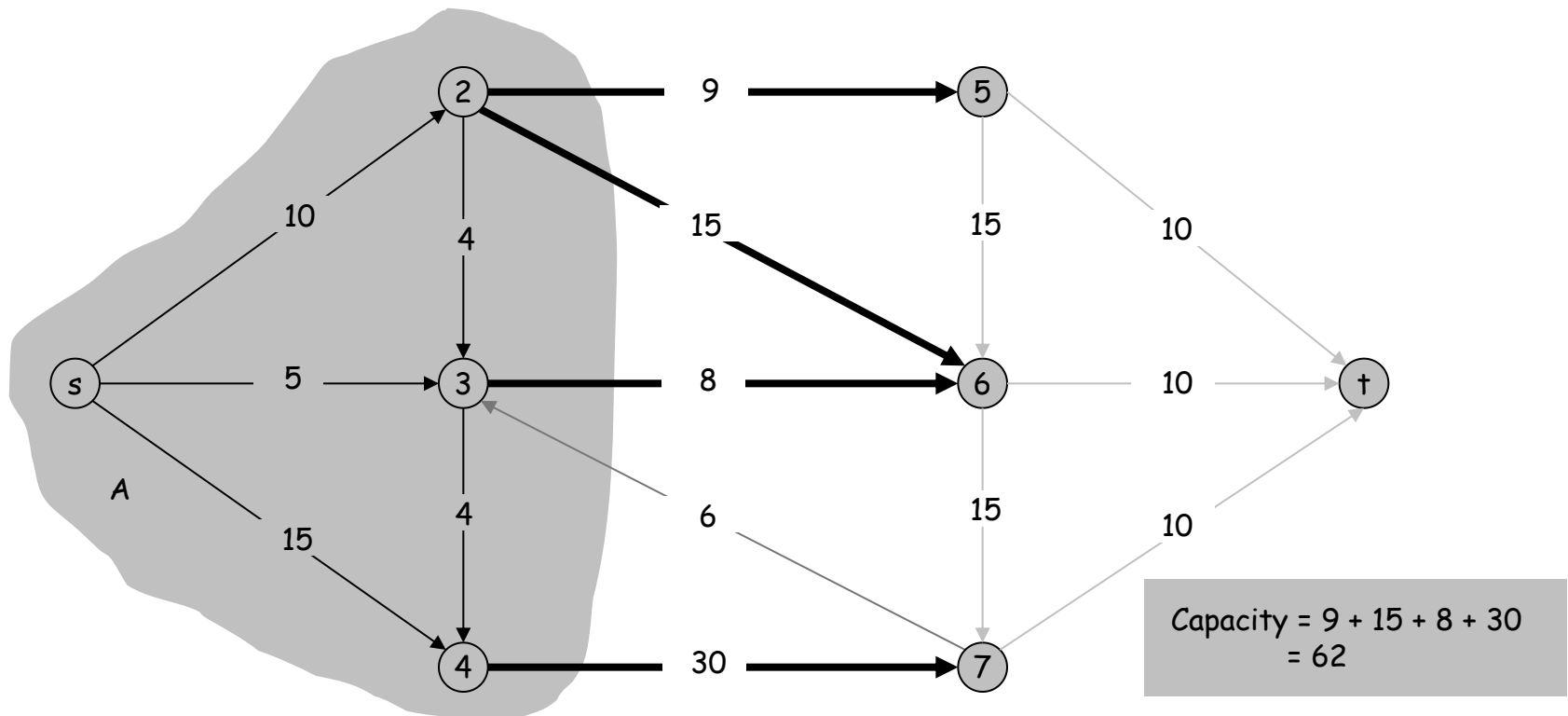
Minimum Cut Problem

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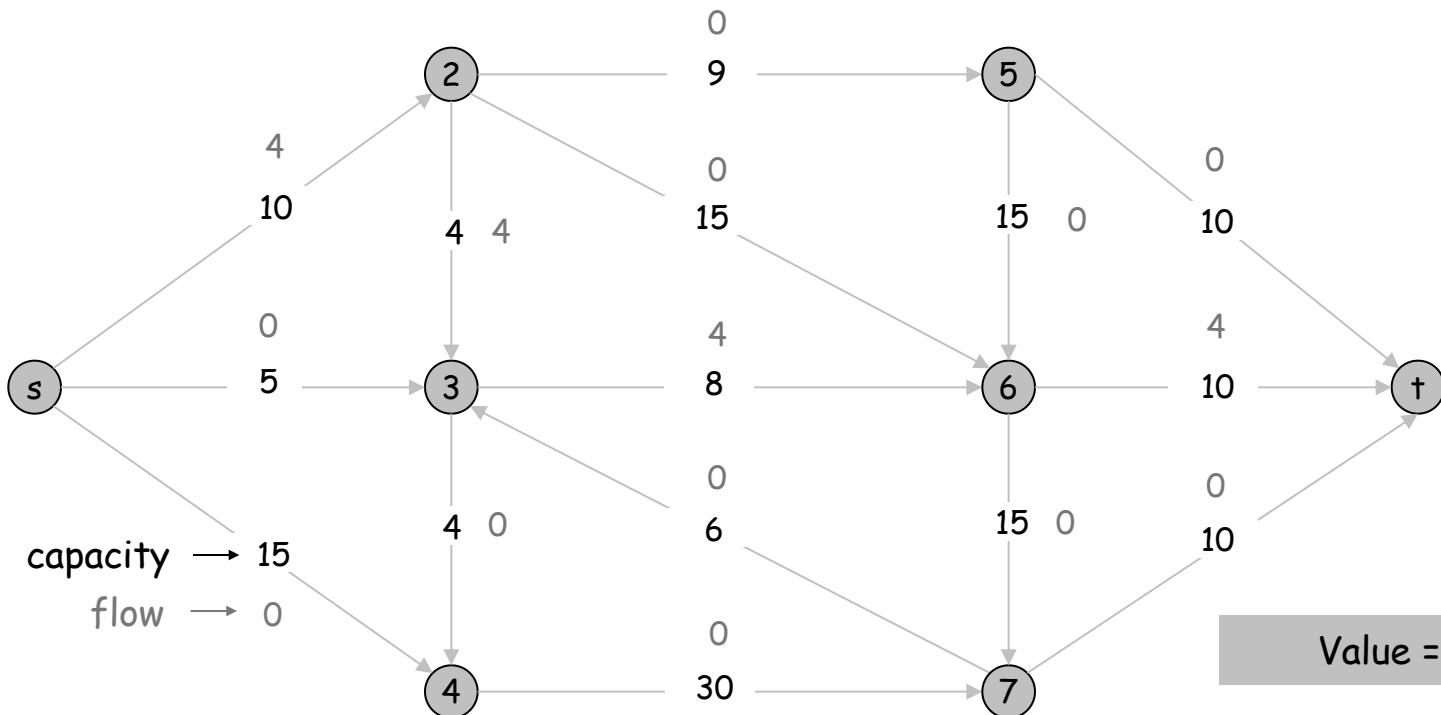

Capacity = 10 + 8 + 10
= 28

Flows

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

- For each $e \in E$: $\qquad 0 \le f(e) \le c(e)$ [capacity]
- For each $v \in V - \{s, t\}$: $\quad \sum\limits_{e \text{ in to } v} f(e) \; = \; \sum\limits_{e \text{ out of } v} f(e)$ [conservation]

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

capacity $\longrightarrow$ 15
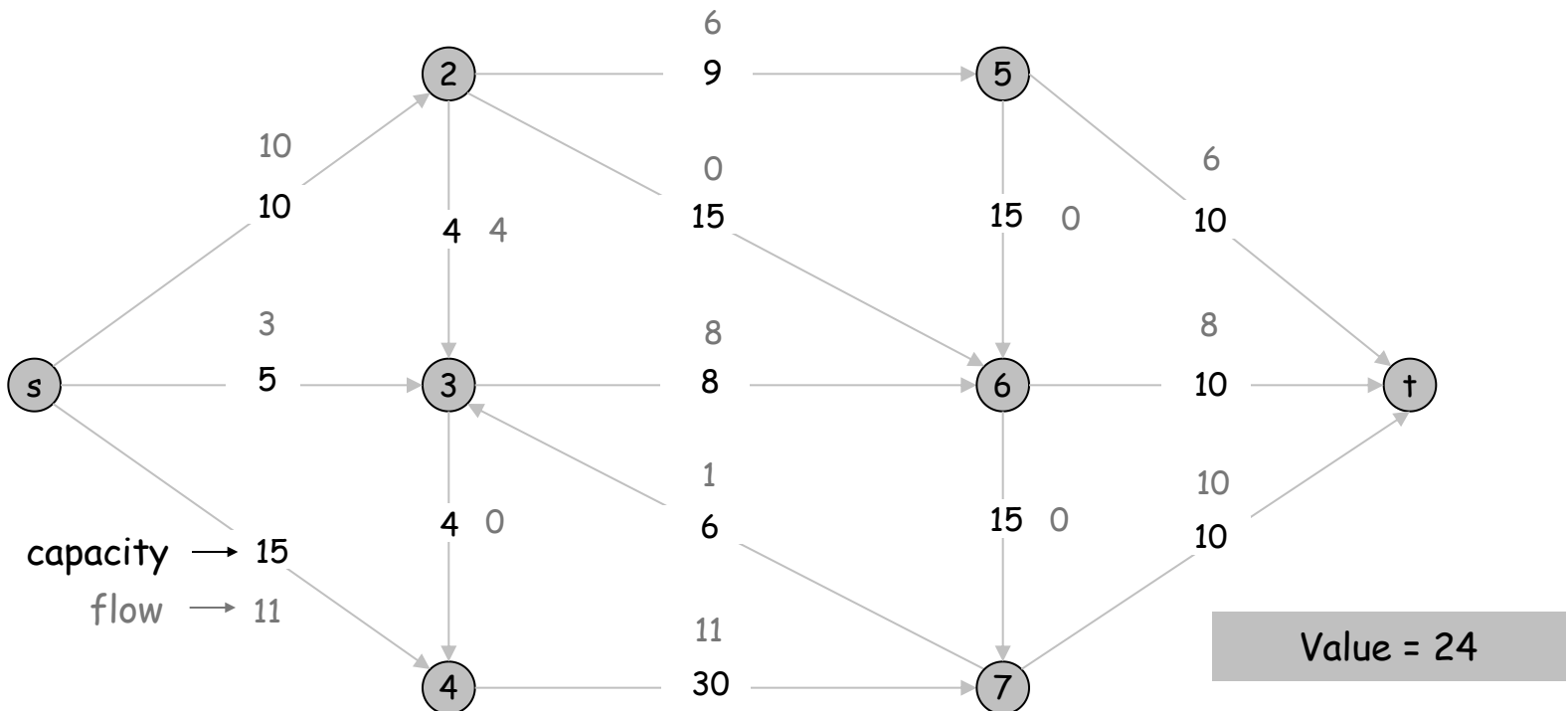flow $\longrightarrow$ 0

Value = 4

Flows

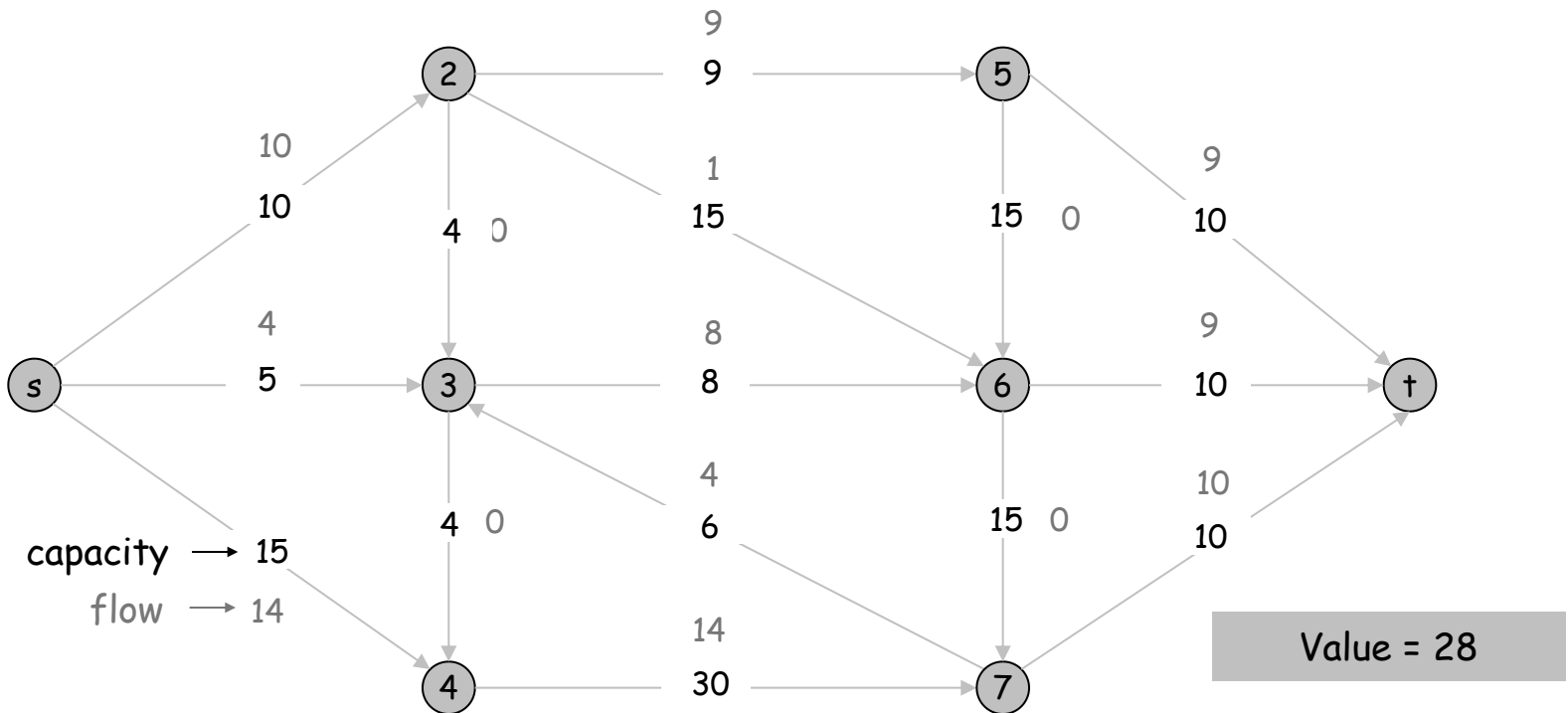Def. An s-t flow is a function that satisfies:
- For each $e \in E$: $0 \le f(e) \le c(e)$ [capacity]
- For each $v \in V - \{s, t\}$: $\sum\limits_{e \text{ in to } v} f(e) = \sum\limits_{e \text{ out of } v} f(e)$ [conservation]

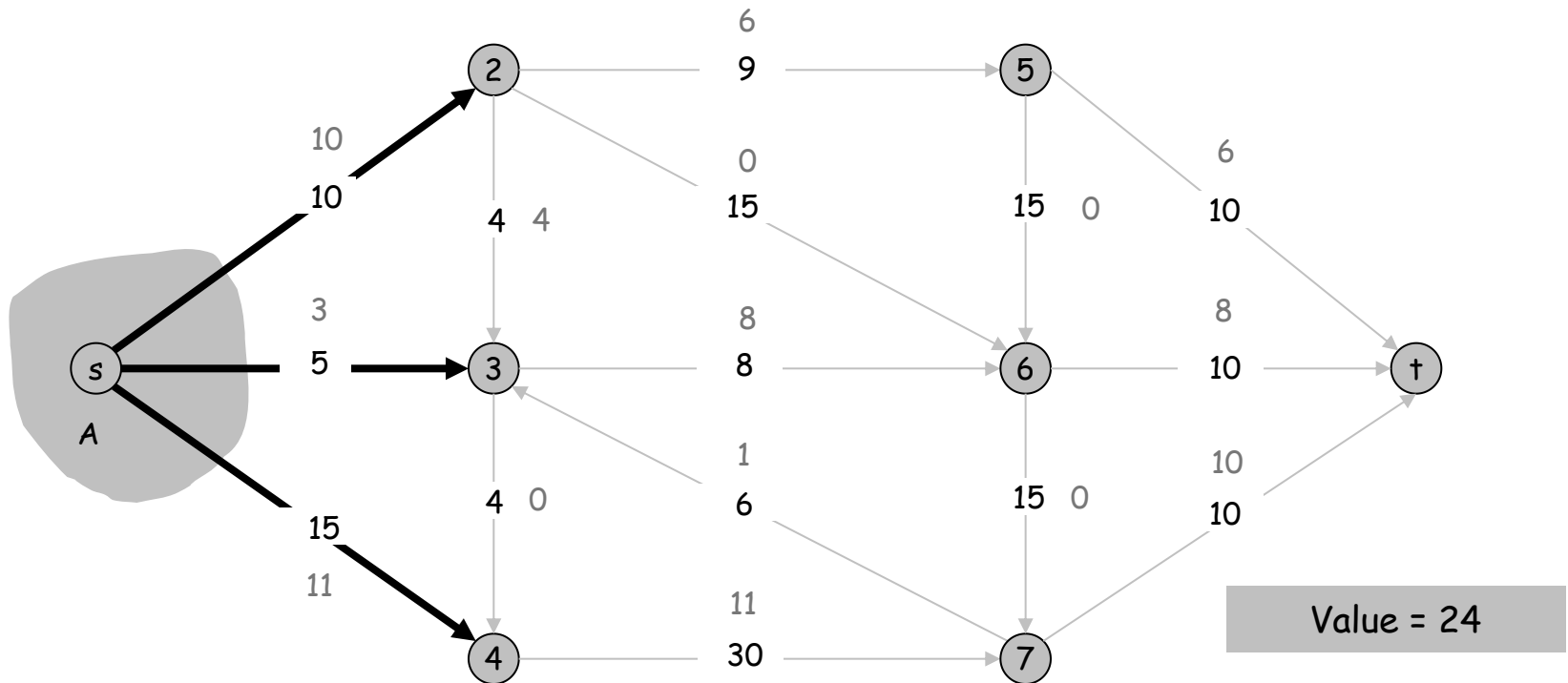Def. The value of a flow f is: $v(f) = \sum\limits_{e \text{ out of } s} f(e)$.

capacity → 15
flow → 11

Value = 24

Maximum Flow Problem

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

capacity → 15
flow → 14

Value = 28

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 = 24

Flows and Cuts

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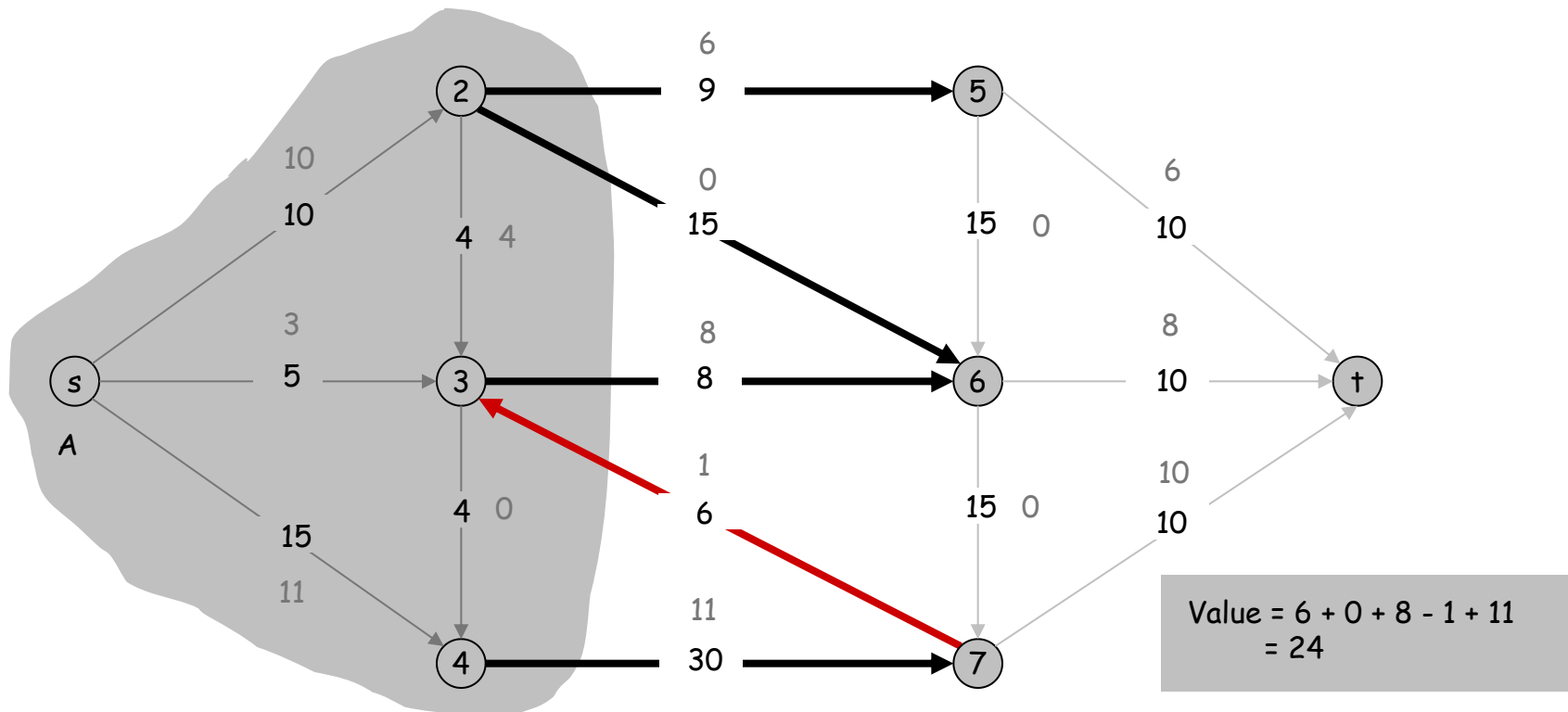Then, the net flow sent across the cut is equal to the amount leaving s.

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

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.
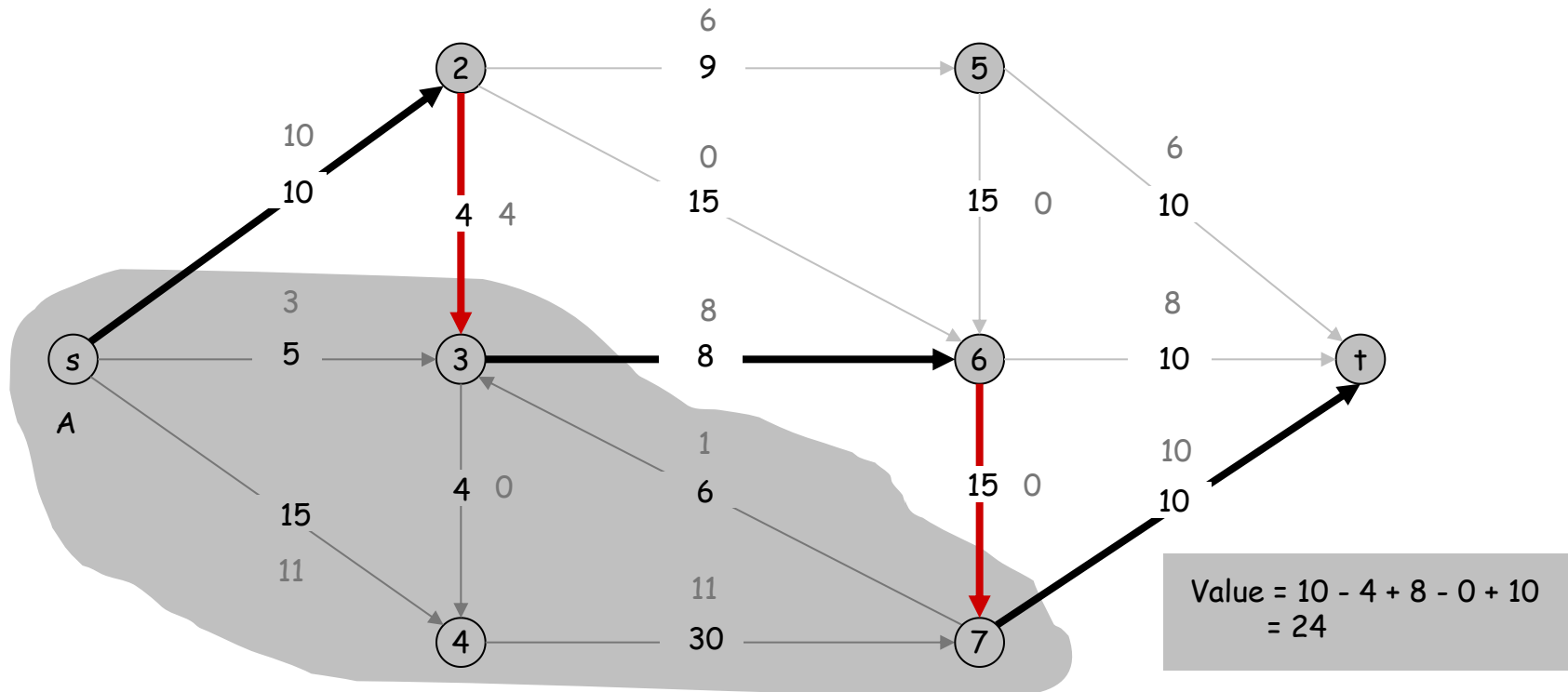Then, the net flow sent across the cut is equal to the amount leaving s.

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

Value = 10 - 4 + 8 - 0 + 10
= 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.

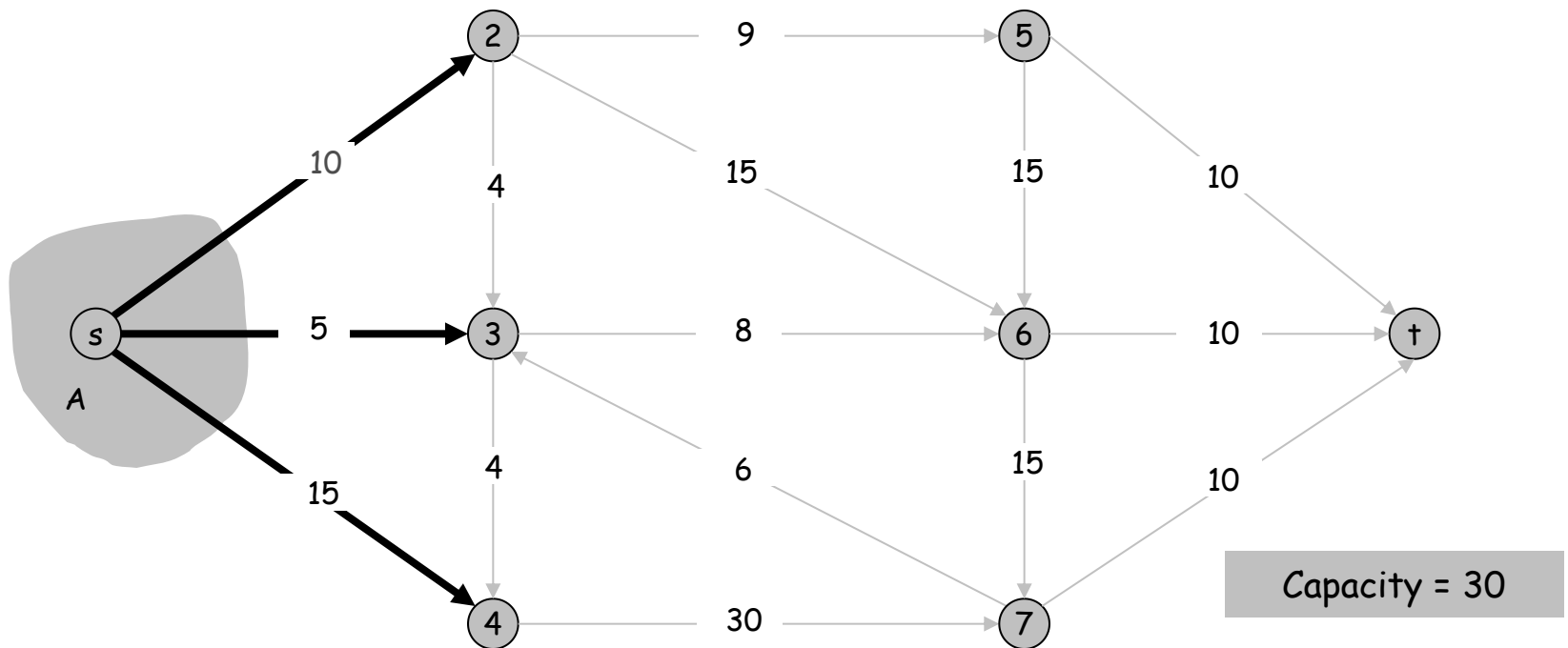$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f).$$

Pf.

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

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

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

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

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 ⇒ Flow value ≤ 30

Capacity = 30
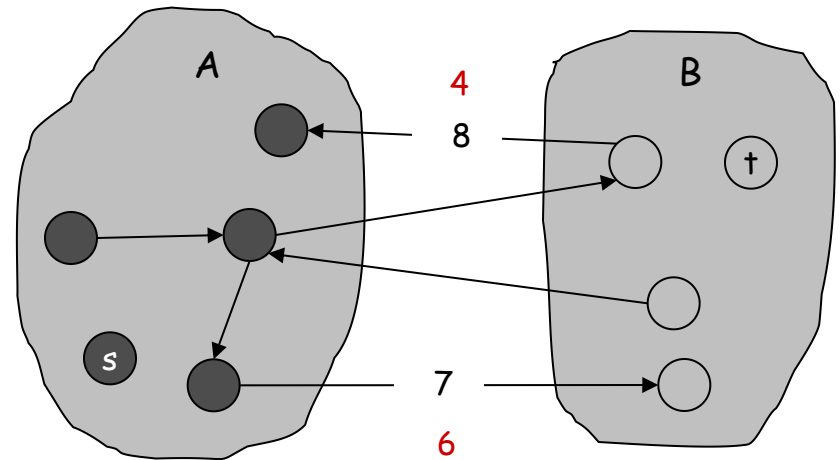
Flows and Cuts

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

Pf.

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

$$\leq \sum_{e \text{ out of } A} f(e)$$

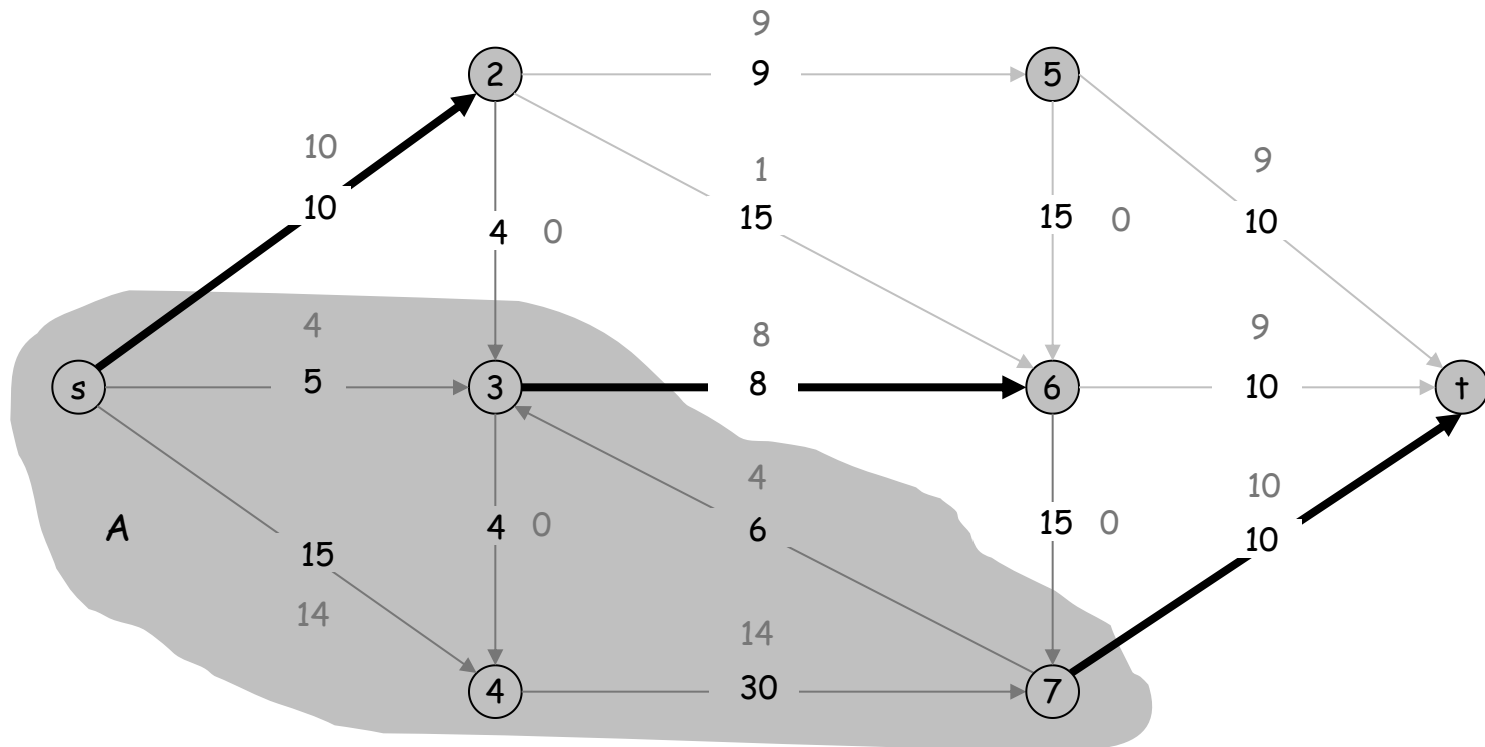$$\leq \sum_{e \text{ out of } A} c(e)$$

$$= cap(A, B) \qquad \blacksquare$$

Certificate of Optimality

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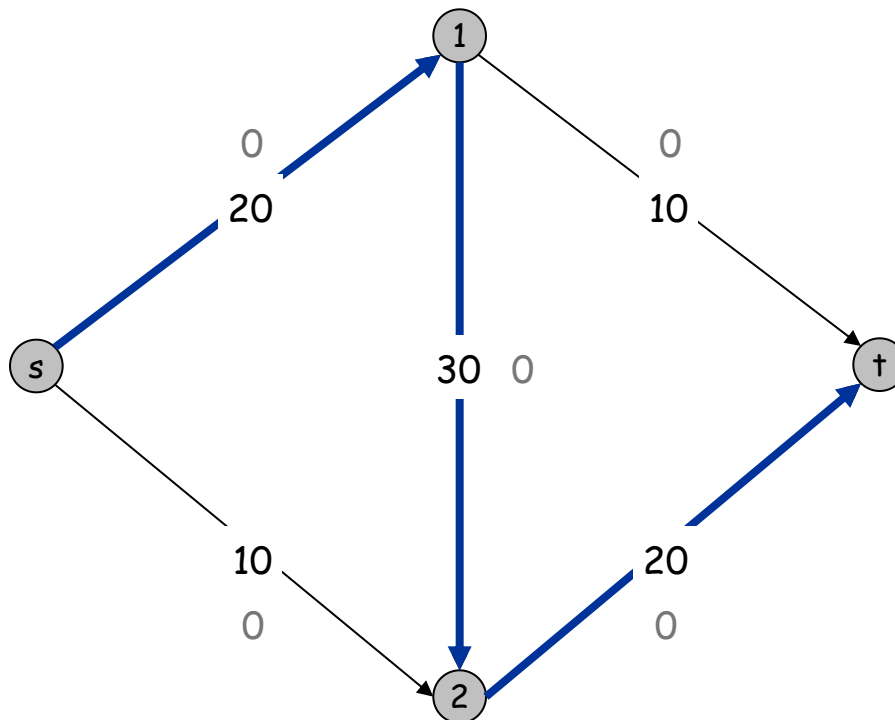If v(f) = cap(A, B), then f is a max flow and (A, B) is a min cut.

Value of flow = 28
Cut capacity = 28 ⇒ Flow value ≤ 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

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.

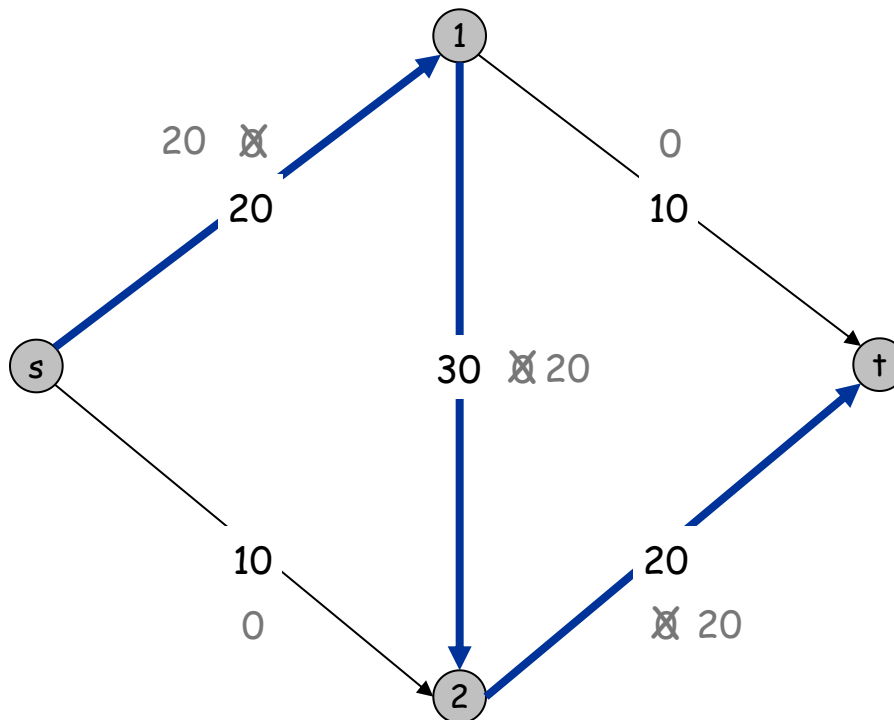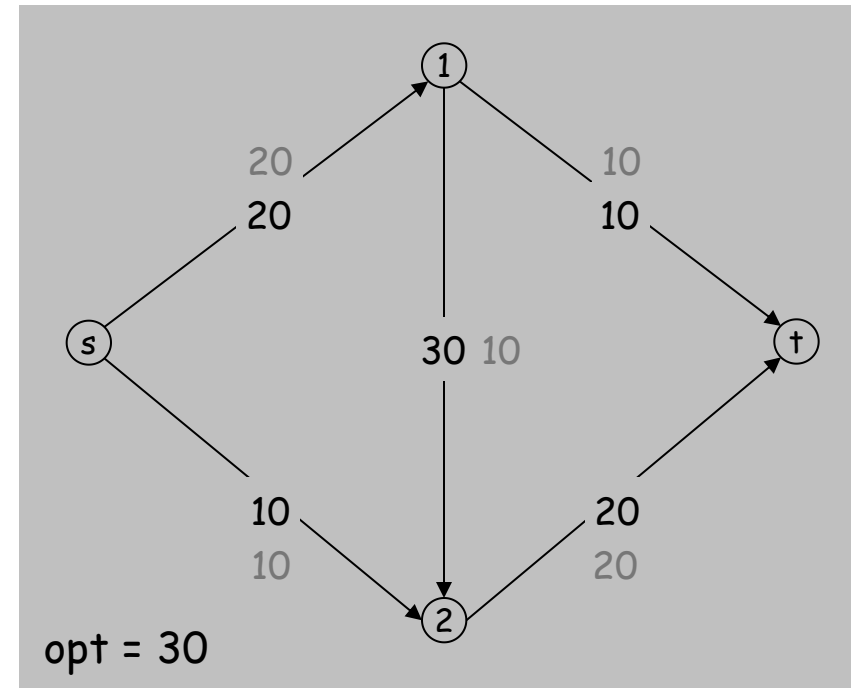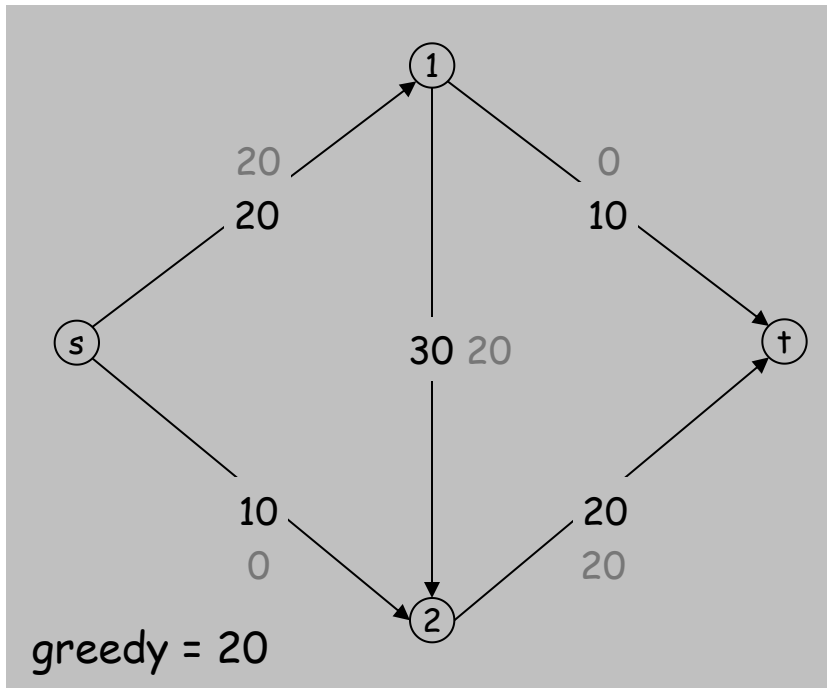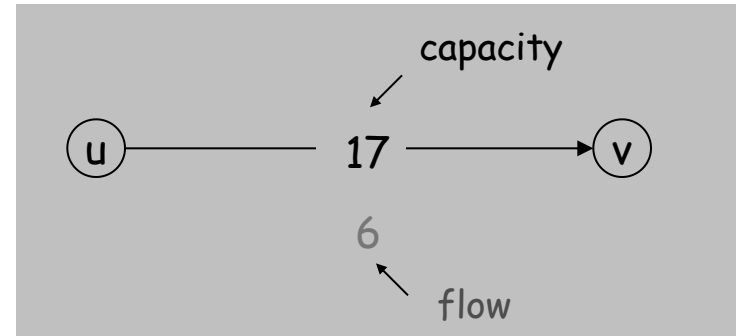locally optimality ⇏ global optimality

greedy = 20

opt = 30

Residual Graph

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

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

capacity
u — 17 → v
6
flow

Residual edge.

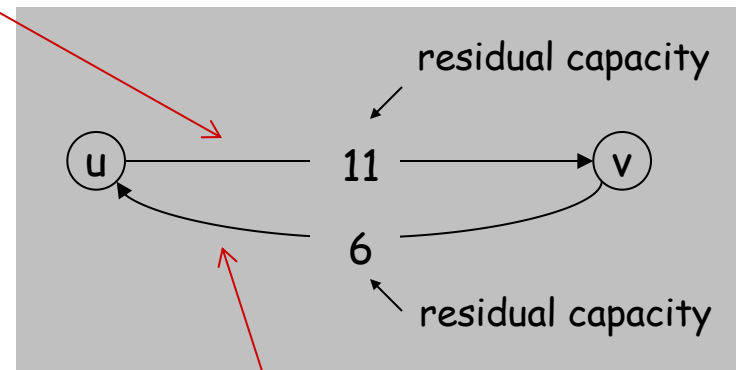- "Undo" flow sent.
- $e = (u, v)$ and $e^R = (v, u)$.
- Residual capacity:

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

Forward edge

residual capacity
u — 11 → v
6
residual capacity

Backward edge

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

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

Augmenting path

Def. An augmenting path is a simple s->t path in the residual graph G_f

Def. The bottleneck capacity of an augmenting path P is the minimum residual capacity of any edge in P.

Key property. Let f be a flow and let P be an augmenting path in G_f, then after calling f' ← Augment(f,c,P), the resulting f' is flow and

$v(f') = v(f) + bottleneck(G_f, P)$

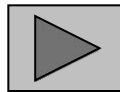# Augmenting Path Algorithm

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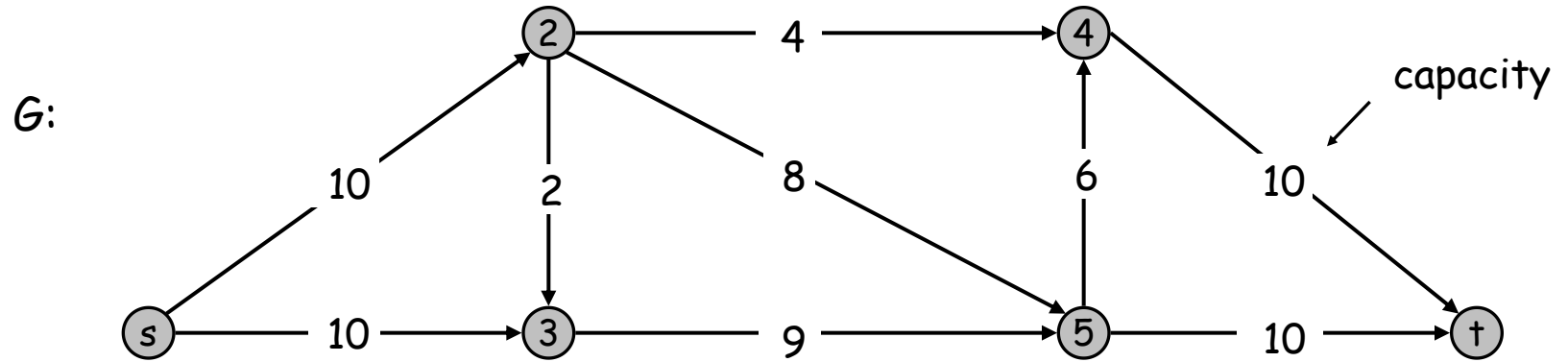
forward edge

reverse edge

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

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

Ford-Fulkerson Algorithm

G:

capacity

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. (Slide 17)

(ii) $\Rightarrow$ (iii) We show contrapositive.
  - Let f be a max flow. If there exists an augmenting path, then we can improve f by sending flow along path.
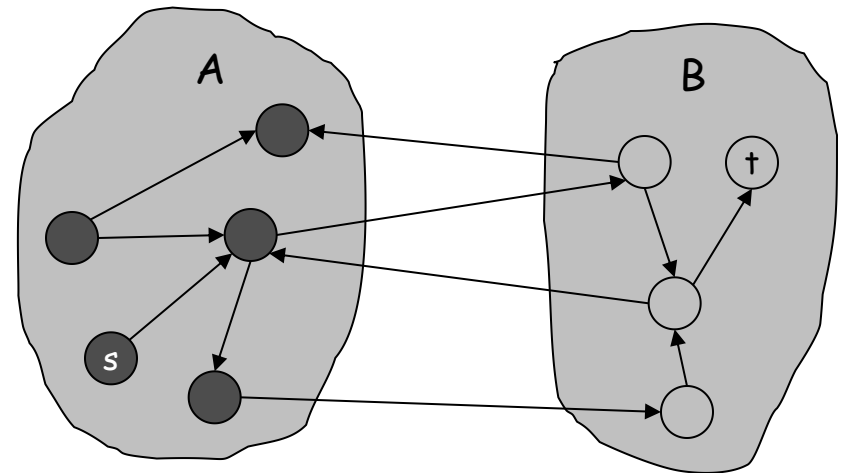
(iii) $\Rightarrow$ (i)

- 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, $t \notin A$.

f(e) = 0, if not, there will be a backward edge in G_f,
Violate no augmenting paths in G_f

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

$$=\; \sum_{e \text{ out of } A} c(e)$$

$$=\; cap(A, B) \quad \blacksquare$$

If not, there will be a forward edge in G_f,
Violate no augmenting paths in G_f

original network

Running Time

Assumption. All capacities are integers between 1 and $C = \sum_{e \text{ out of } s} c(e)$.

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 $v(f^*) \leq C$ iterations.

Pf. Each augmentation increase value by at least 1.   ∎

> To find an s-t path in G_f , say by BFS, O(m+n) with $m \geq n/2$,
> Procedure augment(f,P) takes O(n), as the path has at most n-1 edges

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.   ∎