CS 580
ALGORITHM DESIGN AND ANALYSIS

Max flow - Min cut

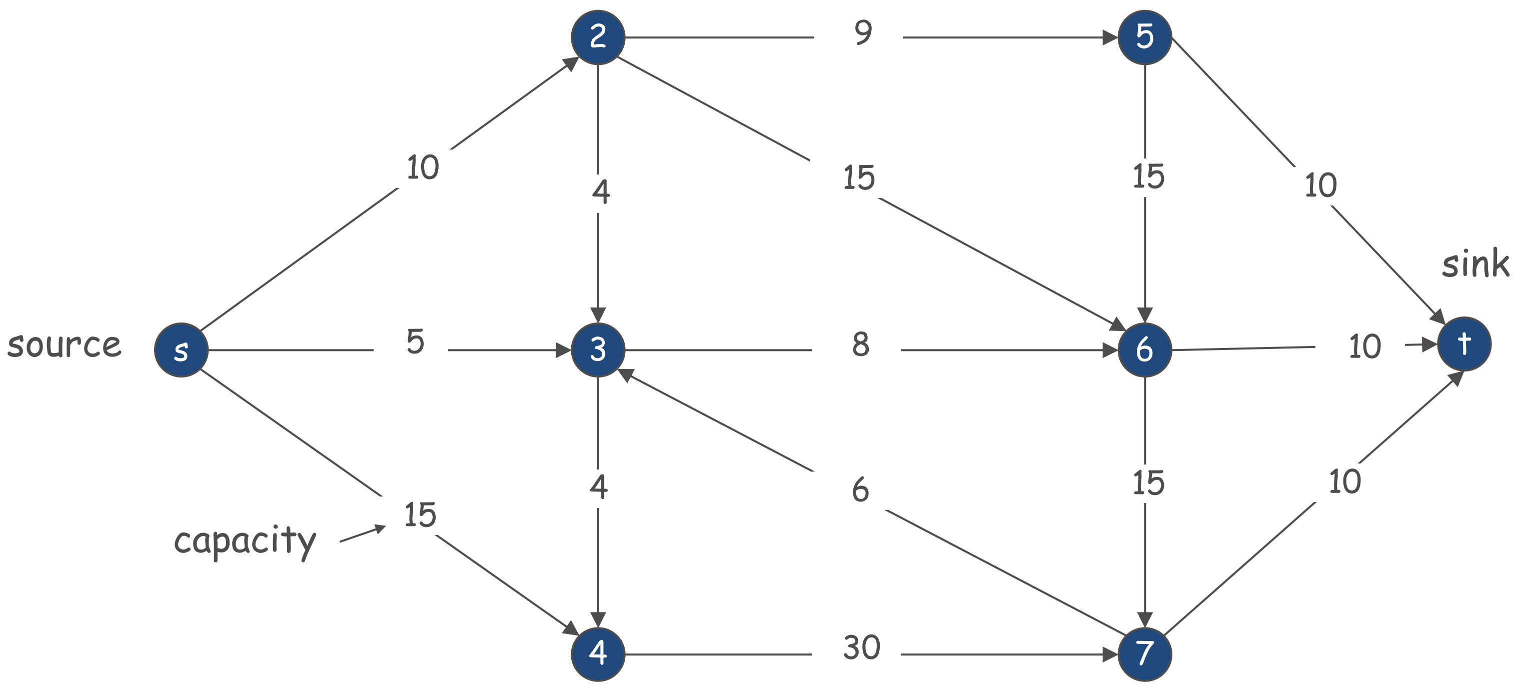
Vassilis Zikas

TODAY

- Maxflow – Mincut!
  - Really important algorithm!
    - Tons of applications (both in theory and practice)
  - Cornerstone of combinatorial optimization

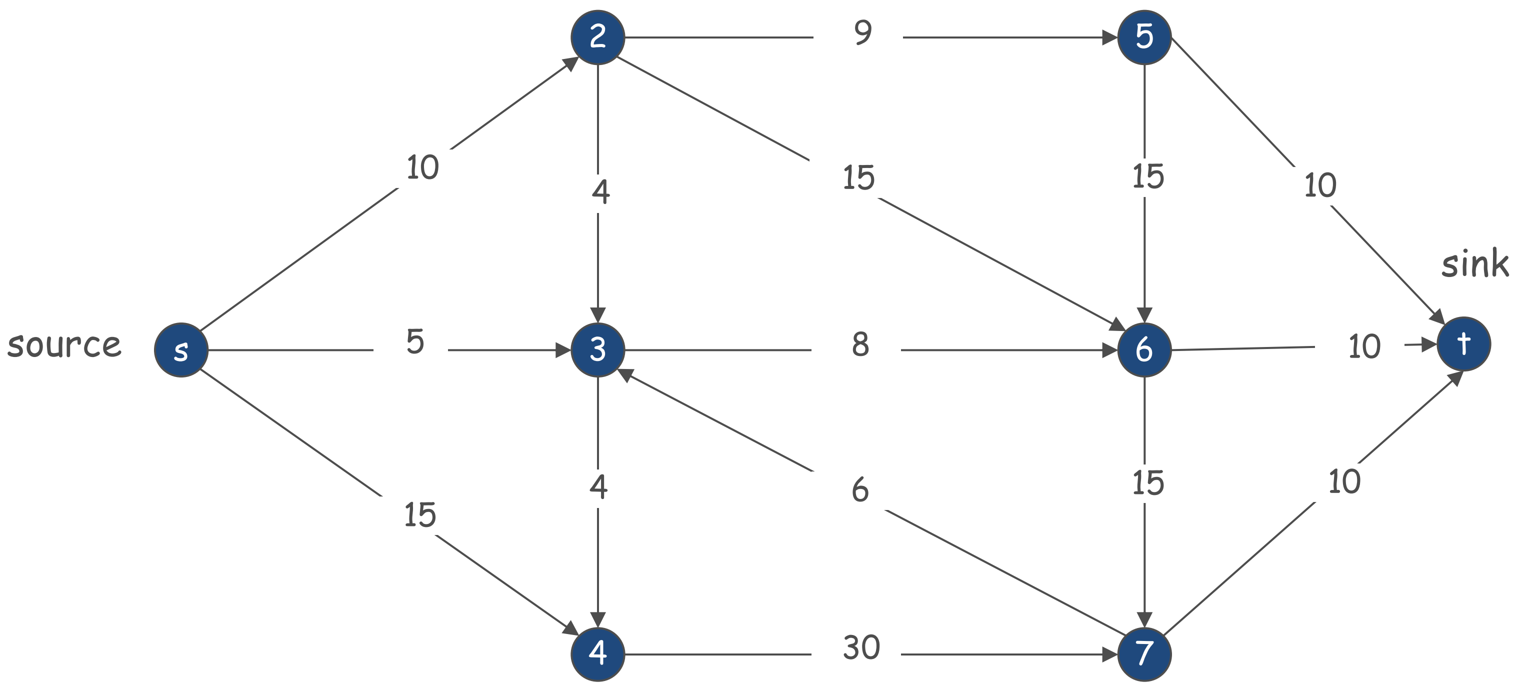
MINIMUM CUT PROBLEM

- Input: 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.



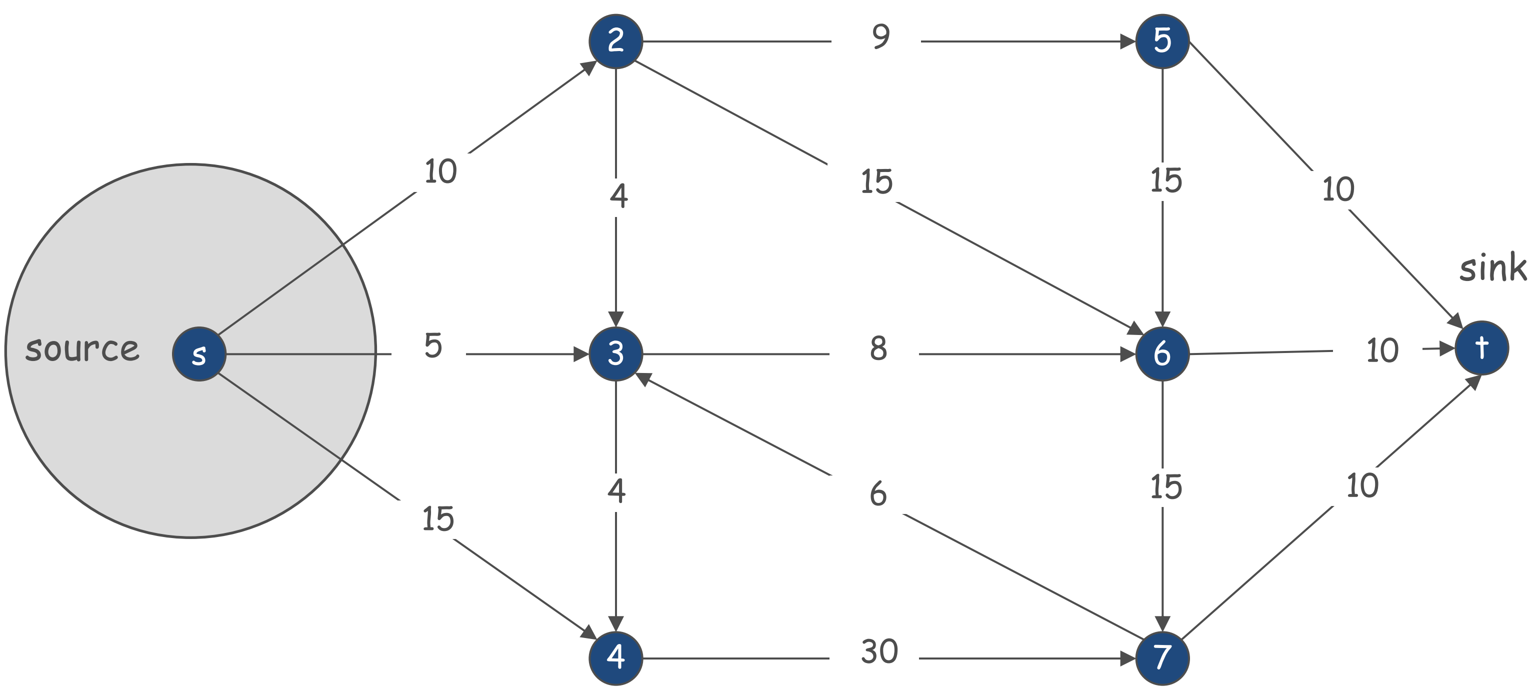
MINIMUM CUT PROBLEM

- An $s - t$ cut is a partition (A, B) of the vertices such that $s \in A$ and $t \in B$
- Capacity of a cut: $cap(A, B) = \sum_{e=(u,v):u\in A, v\in \mathrm{B}} c_e$



MINIMUM CUT PROBLEM

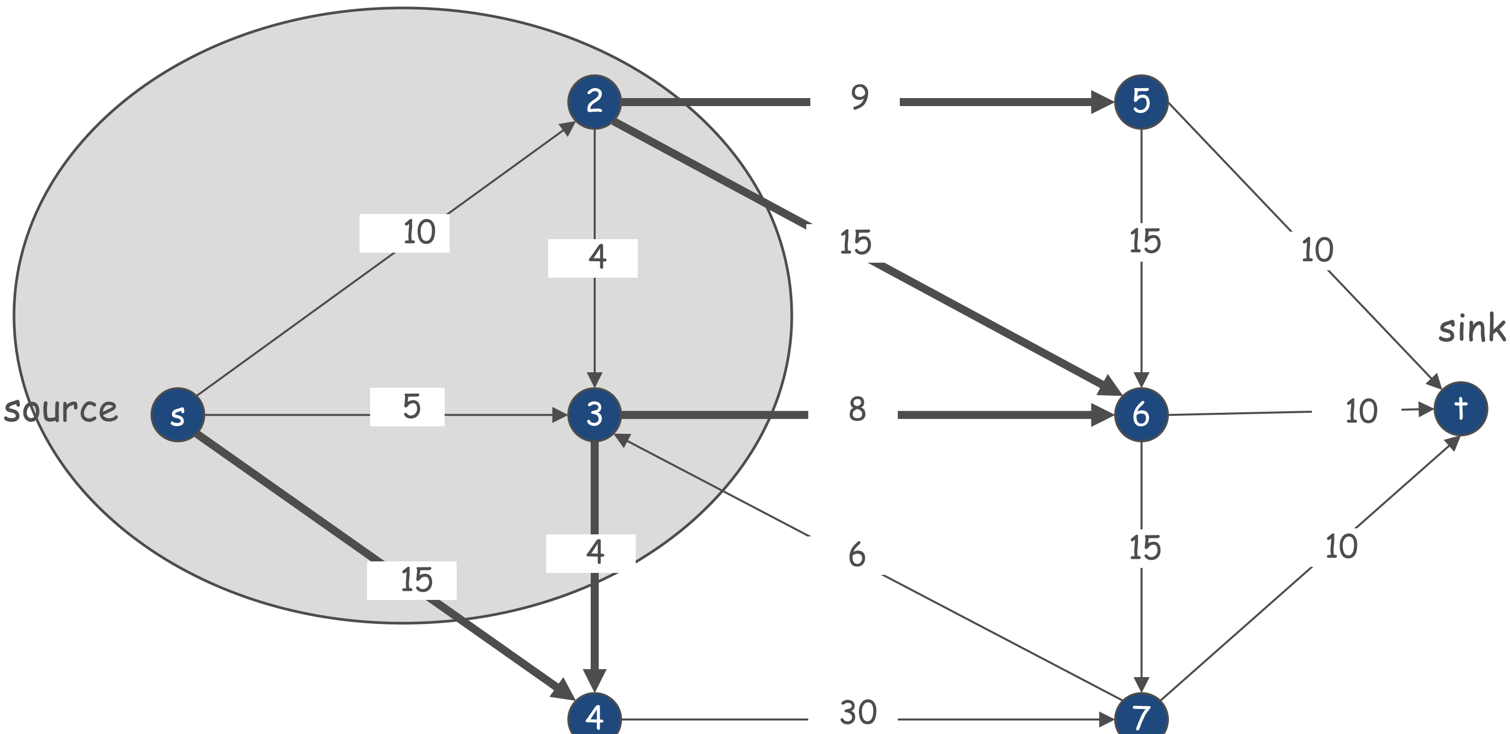
- An $s - t$ cut is a partition (A, B) of the vertices such that $s \in A$ and $t \in B$
- Capacity of a cut: $cap(A, B) = \sum_{e=(u,v):u \in A, v \in \mathrm{B}} c_e$



Capacity = 30

MINIMUM CUT PROBLEM

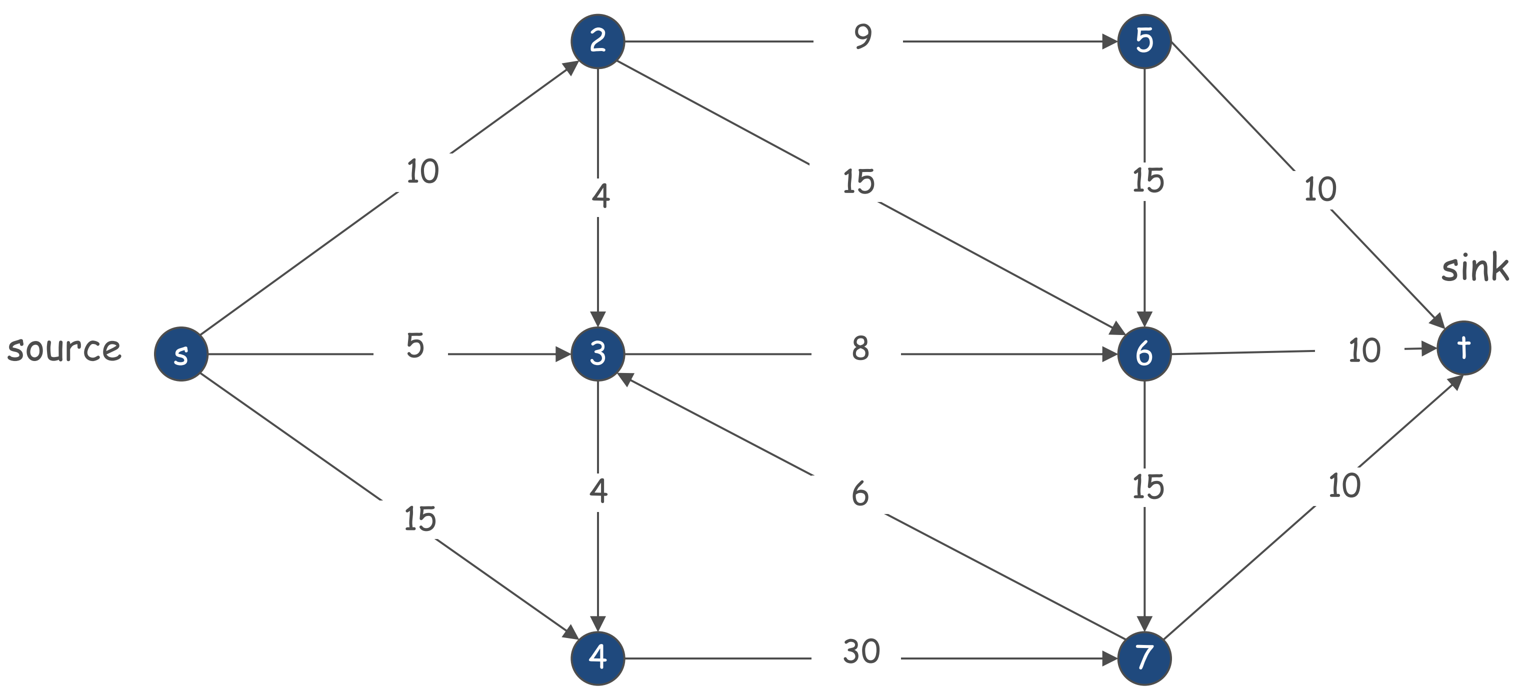
- An $s - t$ cut is a partition (A, B) of the vertices such that $s \in A$ and $t \in B$
- Capacity of a cut: $cap(A, B) = \sum_{e=(u,v):u \in A, v \in \mathrm{B}} c_e$



Capacity = 15+4+8+15+9=51

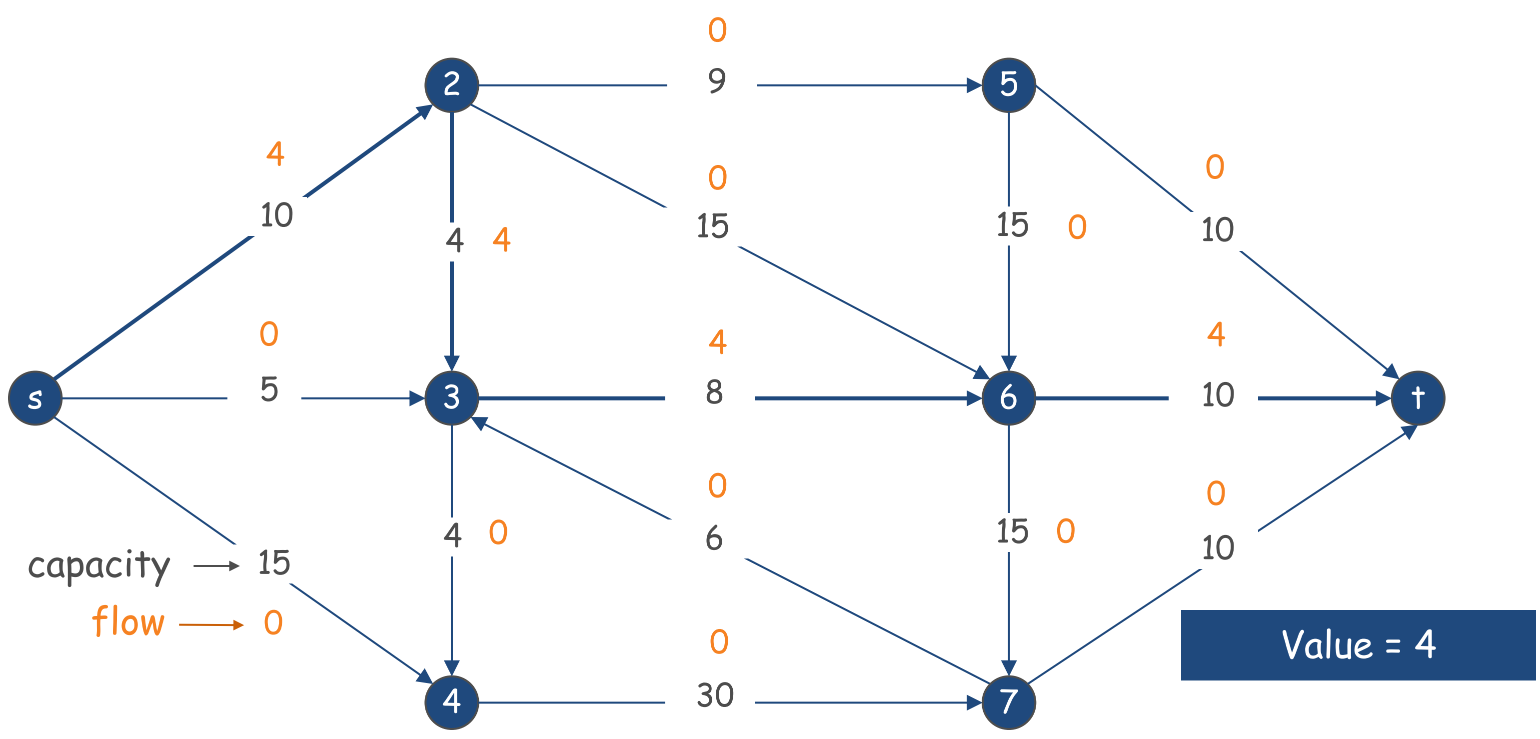
MINIMUM CUT PROBLEM

- Problem: find the $s - t$ cut of minimum capacity



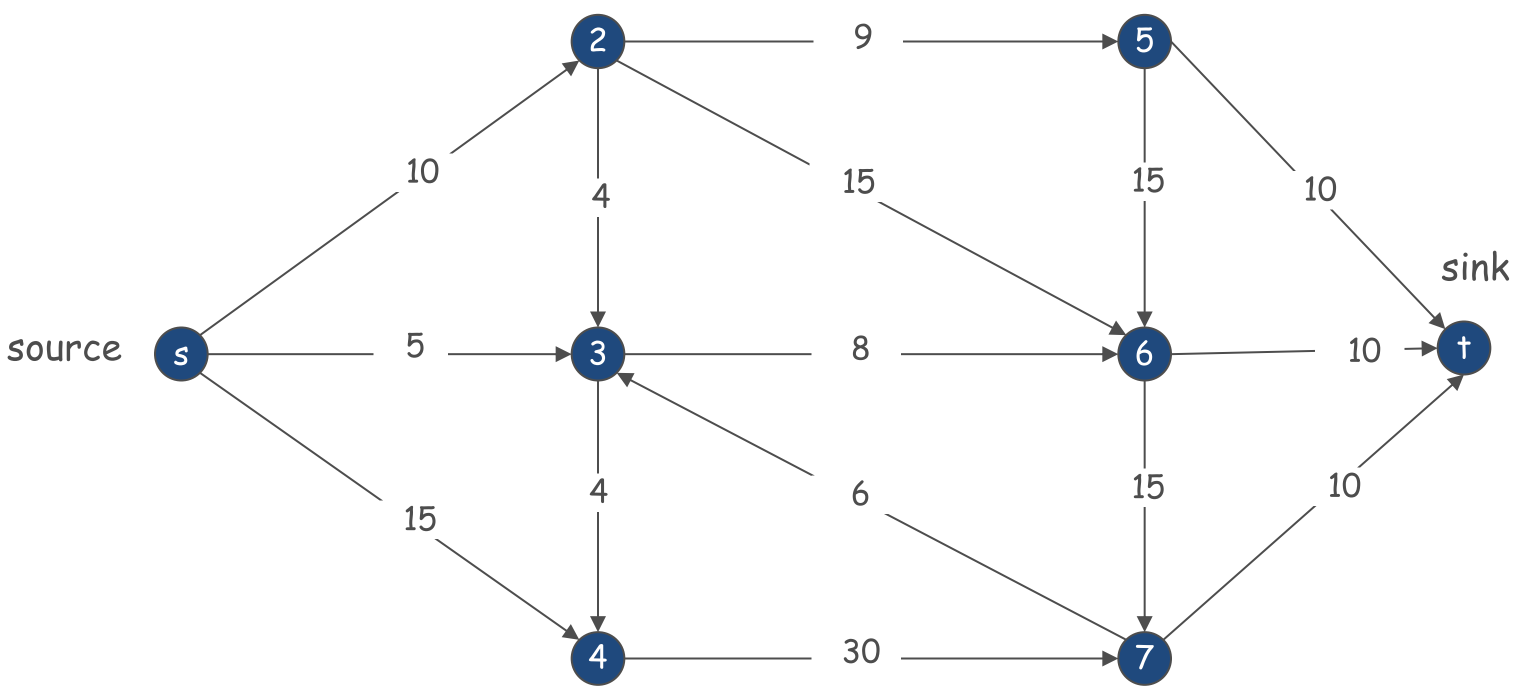
MINIMUM CUT PROBLEM

- Dfn: An $s - t$ flow is a function that satisfies:
  - (capacity) For each $e \in E$: $0 \leq f(e) \leq c(e)$
  - (flow conservation) For each $v \in V \setminus \{s, t\}$: $\sum_{e \text{ in } v} f(e) = \sum_{e \text{ out of } v} f(e)$
- Value of flow: $v(f) = \sum_{e \text{ out of } s} f(e)$



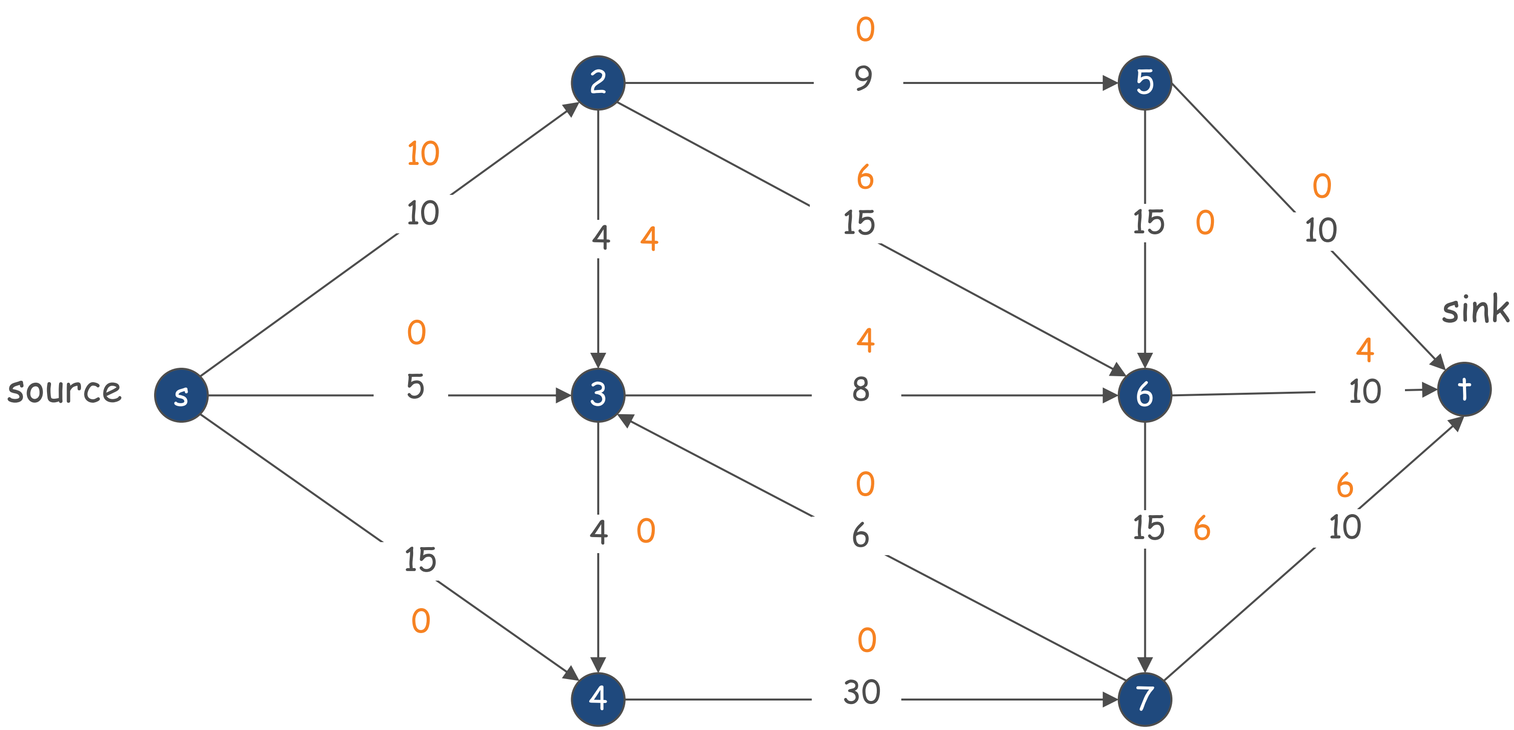
MAX FLOW PROBLEM

- Problem: find the $s - t$ flow of maximum value



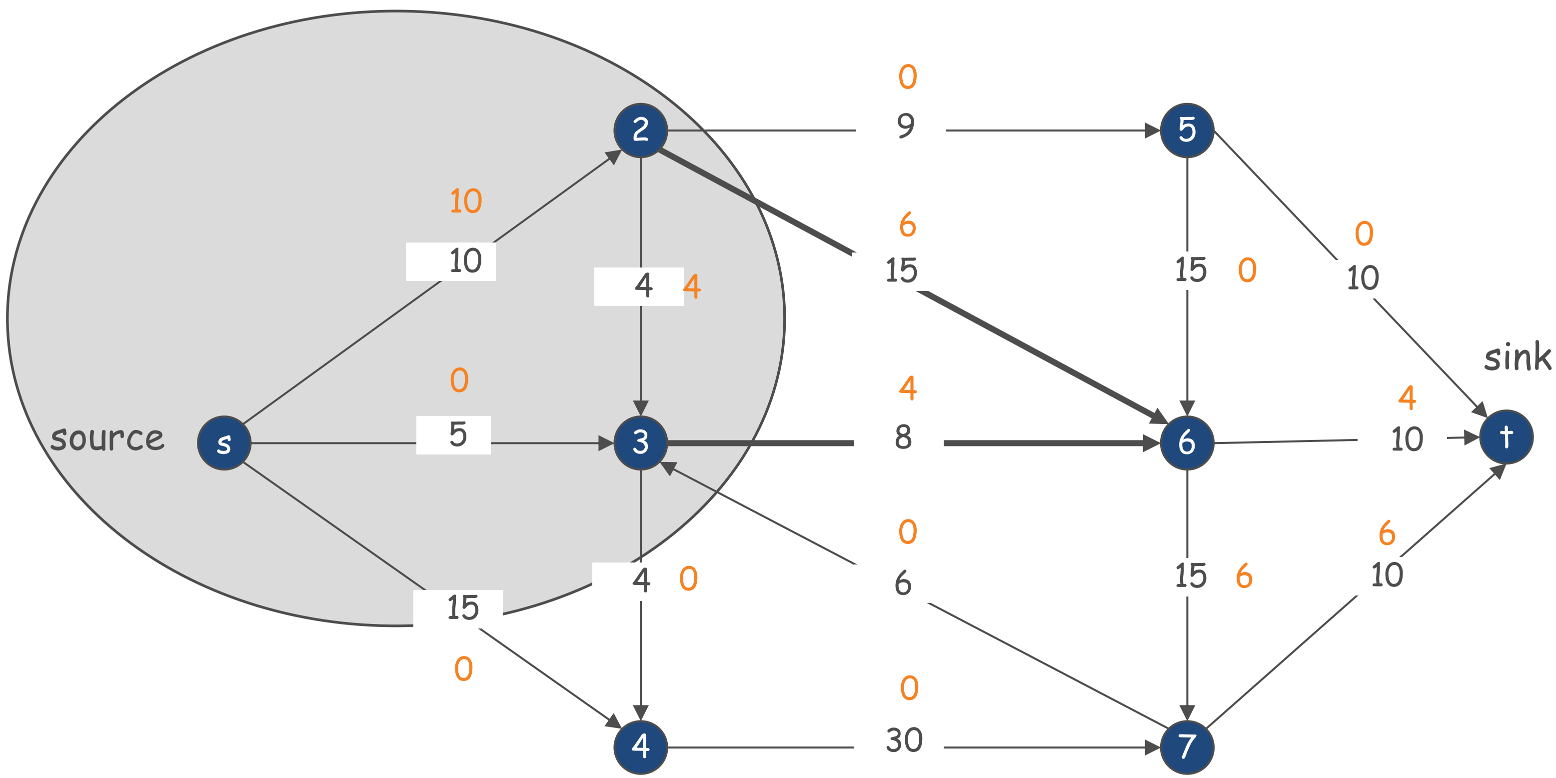
FLOW VALUE

- Lemma: Let f be any valid $s - t$ flow. Let (A, B) be any $s - t$ cut. Then, the net flow across the cut is equal to the amount leaving s



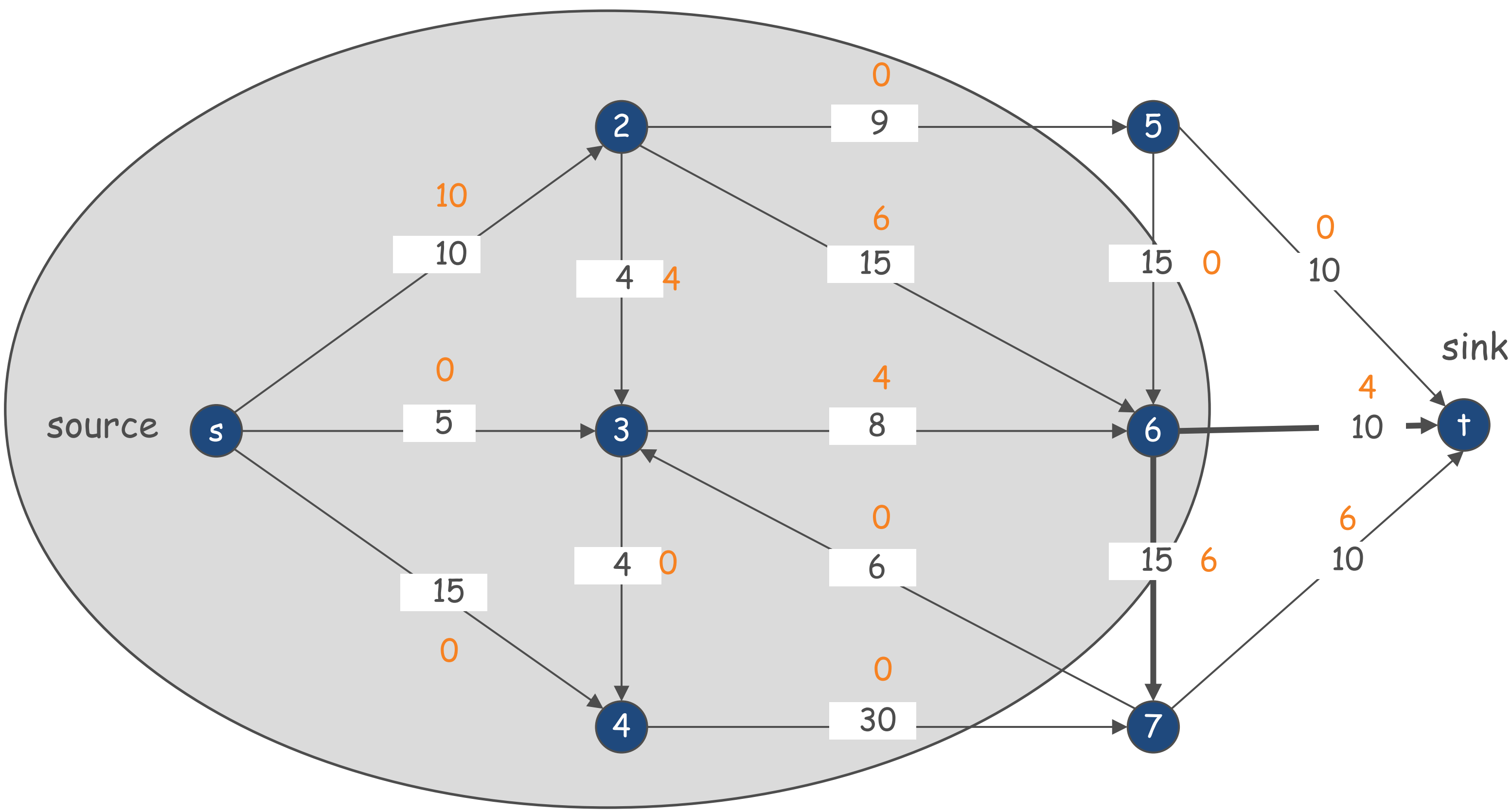
FLOW VALUE

- Lemma: Let f be any valid $s - t$ flow. Let (A, B) be any $s - t$ cut. Then, the net flow across the cut is equal to the amount leaving s



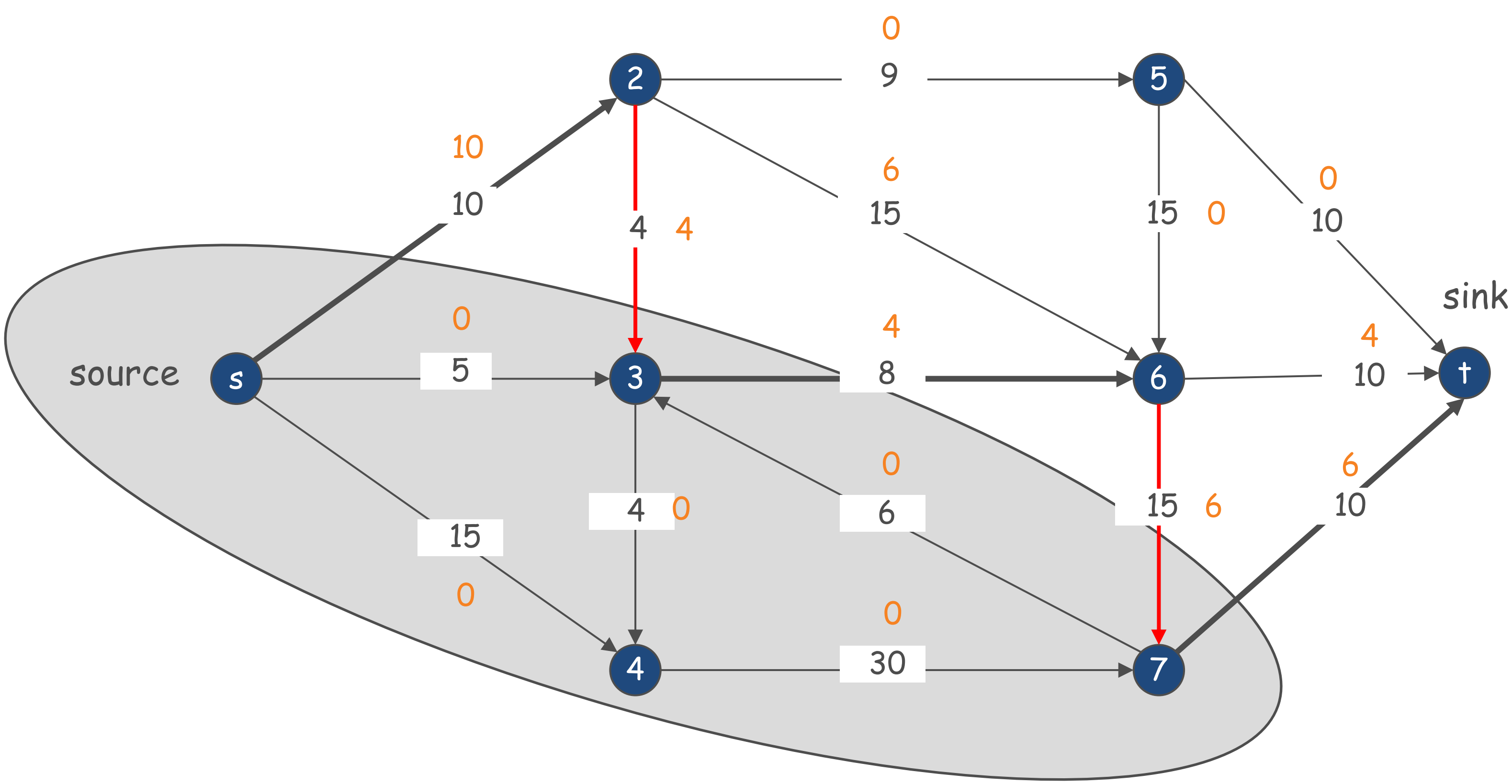
FLOW VALUE

- Lemma: Let f be any valid $s - t$ flow. Let (A, B) be any $s - t$ cut. Then, the net flow across the cut is equal to the amount leaving s



FLOW VALUE

- Lemma: Let f be any valid $s-t$ flow. Let (A, B) be any $s-t$ cut. Then, the net flow across the cut is equal to the amount leaving s



FLOW VALUE

- Lemma: Let f be any valid $s-t$ flow. Let (A,B) be any $s-t$ cut. Then, the net flow across the cut is equal to the amount leaving s
- Proof:
- $v(f) = \sum_{e\ out\ of\ s} f(e) + 0$

FLOW VALUE

- Lemma: Let f be any valid $s-t$ flow. Let (A, B) be any $s-t$ cut. Then, the net flow across the cut is equal to the amount leaving s
- Proof:
- $v(f) = \sum_{e \text{ out of } s} f(e) + 0$
- $0 = \sum_{v \in A \setminus \{s\}} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e)\right)$

Flow conservation

FLOW VALUE

- Lemma: Let f be any valid $s-t$ flow. Let (A,B) be any $s-t$ cut. Then, the net flow across the cut is equal to the amount leaving s
- Proof:
- $v(f) = \sum_{e\ out\ of\ s} f(e) + 0$

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

Flow in to s is zero

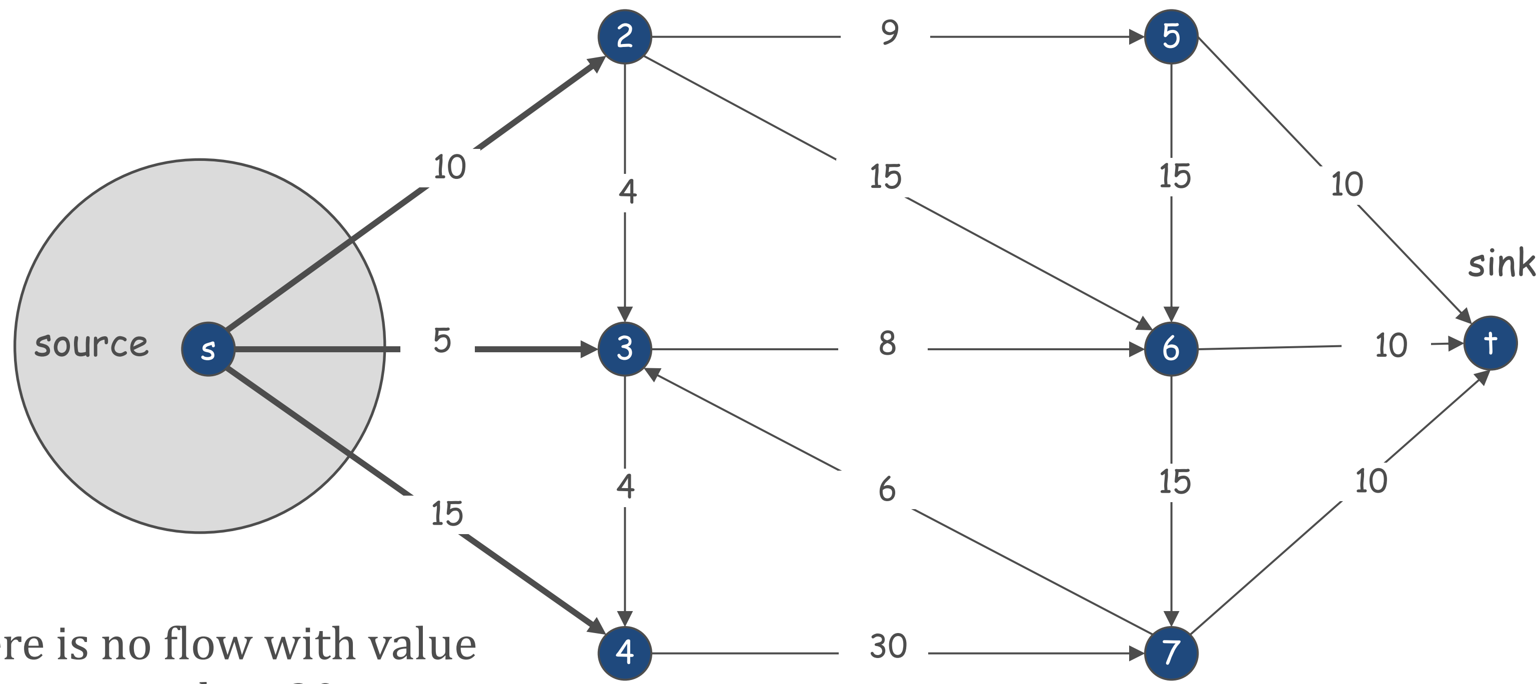
FLOW VALUE

- Lemma: Let f be any valid $s - t$ flow. Let (A, B) be any $s - t$ cut. Then, the net flow across the cut is equal to the amount leaving s
- Proof:

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

FLOWS AND CUTS

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



There is no flow with value more than 30

FLOWS AND CUTS

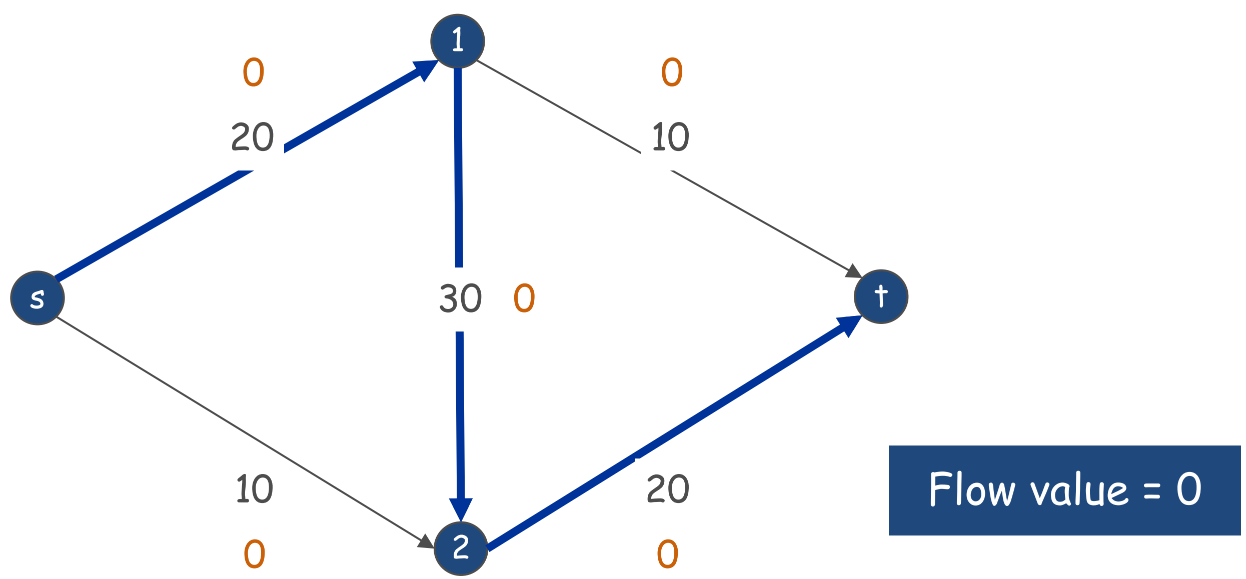
- Proof of weak duality
- $v(f) = \sum_{e\ out\ of\ A} f(e) - \sum_{e\ in\ to\ A} f(e)$
- $\leq \sum_{e\ out\ of\ A} f(e)$
- $\leq \sum_{e\ out\ of\ A} c(e)$
- $= cap(A, B)$

FLOWS AND CUTS

- Weak duality: Let f be any flow and let (A, B) be any $s - t$ cut. The value of f is at most the capacity of the cut
- Corollary: Let f be any flow and let (A, B) be any $s - t$ cut. If $v(f) = cap(A, B)$ then f is a max flow and (A, B) is a min cut!

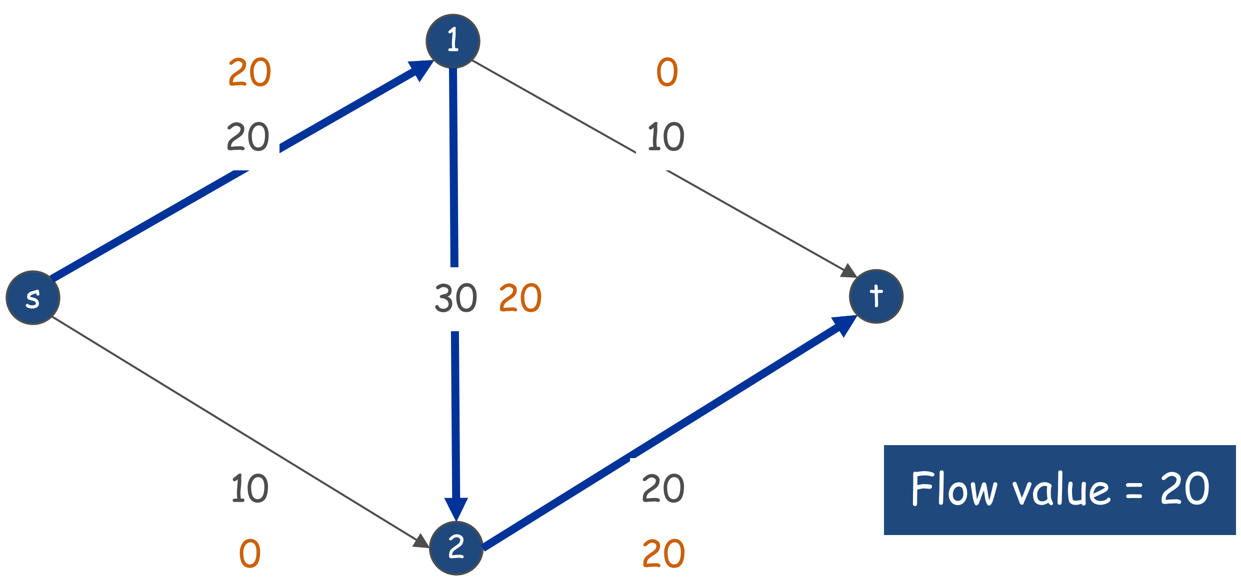
ALGORITHMS

- Natural greedy algorithm
  - Start with an empty flow: $f(e) = 0$ for all $e \in E$
  - Find $s - t$ path where each edge has $f(e) < c(e)$
  - Augment flow along the path
  - Repeat until you get stuck



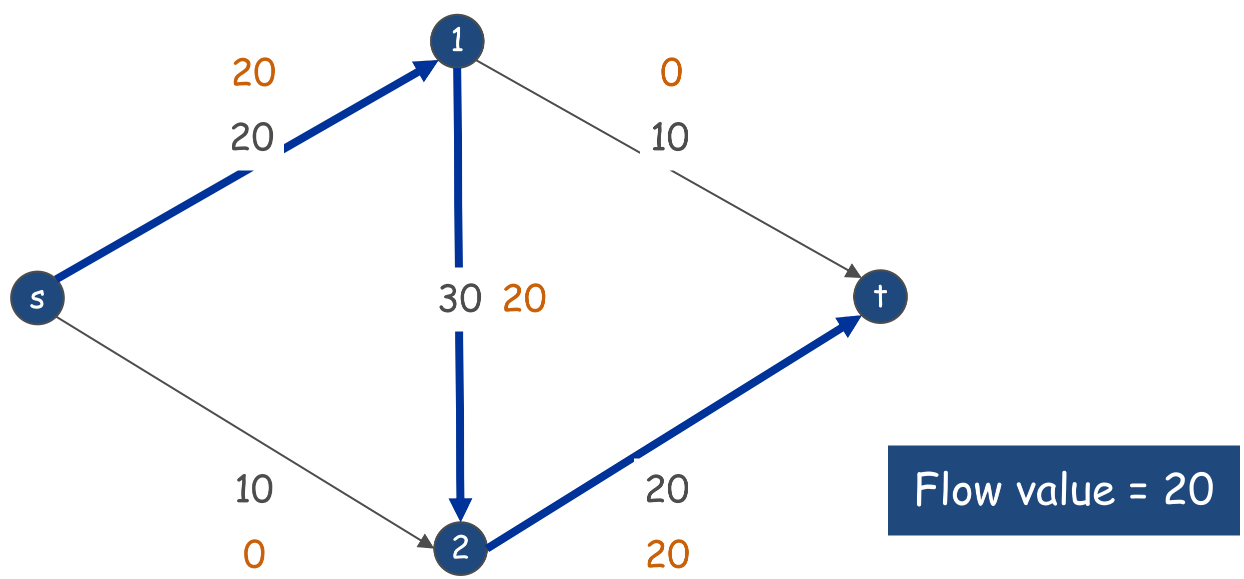
ALGORITHMS

- Natural greedy algorithm
  - Start with an empty flow: $f(e) = 0$ for all $e \in E$
  - Find $s - t$ path where each edge has $f(e) < c(e)$
  - Augment flow along the path
  - Repeat until you get stuck



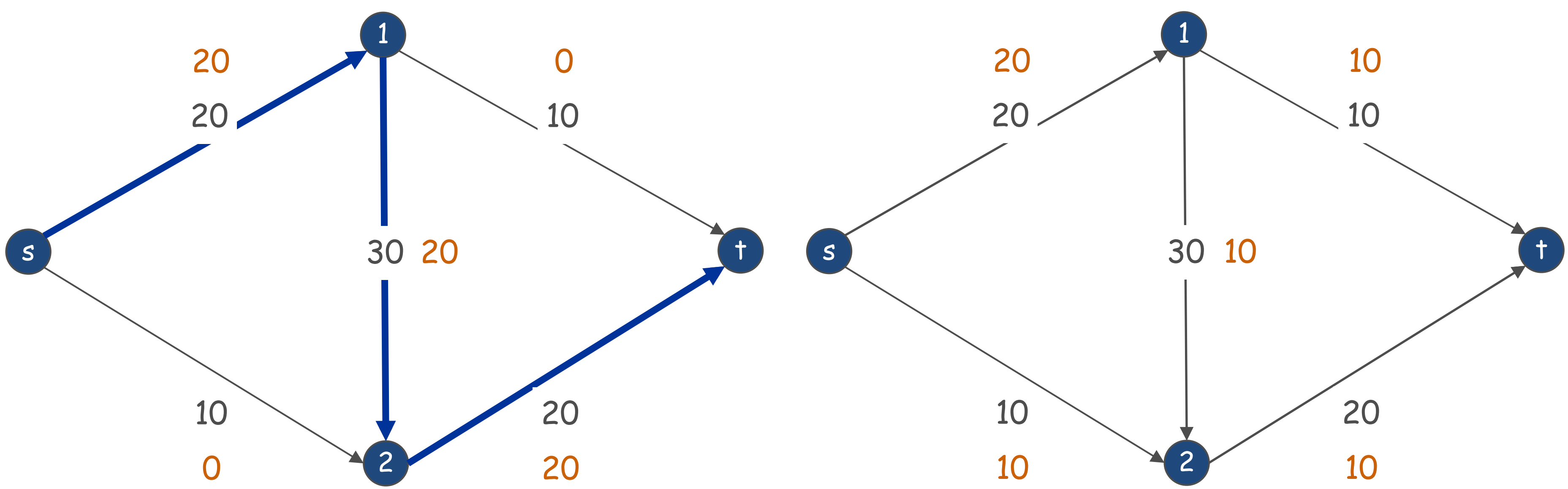
ALGORITHMS

- Natural greedy algorithm
  - Start with an empty flow: $f(e) = 0$ for all $e \in E$
  - Find $s - t$ path where each edge has $f(e) < c(e)$
  - Augment flow along the path
  - Repeat until you get stuck



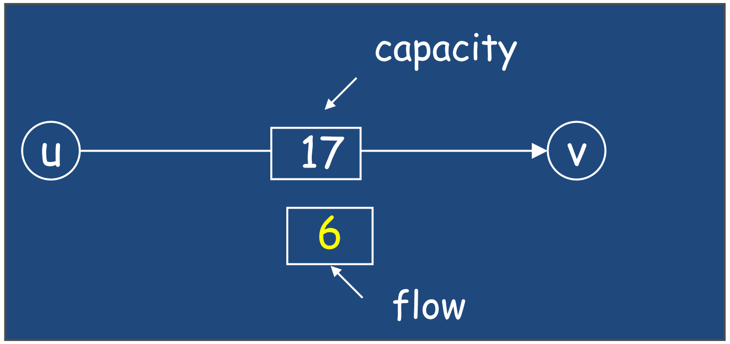
ALGORITHMS

- Local optimality ≠ Global optimality



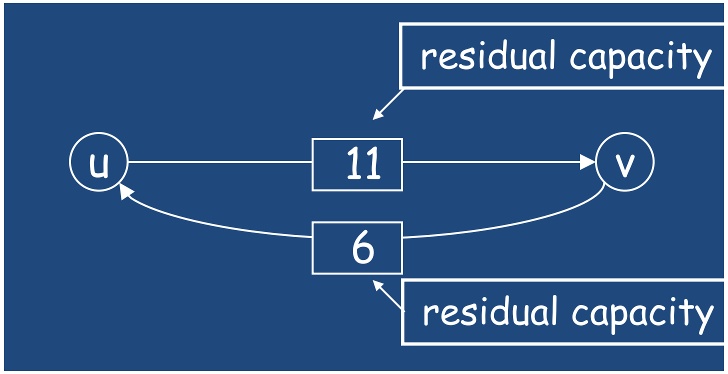
RESIDUAL GRAPH

- 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.
  - $E_f = \{e : f(e) < c(e)\} \cup \{e^R : f(e) > 0\}$.

FORD-FULKERSON DEMO

AUGMENTING PATH ALGORITHM

- Augment($f, c, path\ P$)
  - b =bottleneck of P
  - For each $e \in P$:
    - If $e \in E$: $f(e) = f(e) + b$ (forward edge)
    - Else: $f(e^R) = f(e^R) - b$ (reverse edge)
- Ford-Fulkerson (G, s, t, c)
  - For each $e \in$ E: $f(e) = 0$
  - G_f = residual graph
  - While ∃ residual path P:
    - $f = Augment(f, c, P)$
    - Update G_f

AUGMENTING PATH THEOREM

- Theorem: Flow f is a max-flow iff there are no augmenting paths
- Max-flow min-cut theorem: The value of the max flow is equal to the value of the min cut!
- We will prove both simultaneously by showing that 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

MAX-FLOW MIN-CUT

- $(i) \to (ii)$: Corollary to weak duality
- $(ii) \to (iii)$
  - Contrapositive: Let f be a flow and an augmenting path relative to f. Then, we can improve f by sending more flow across the path (i.e. f is not a max flow)

MAX-FLOW MIN-CUT

- $(iii) \rightarrow (i)$
- I.e. No augmenting path for $f \rightarrow$ there exists a cut (A, B) such that $v(f) = \text{cap}(\text{A}, \text{B})$
- Proof:
  - Let A be the set of vertices reachable from s in the residual graph
  - By definition of A, $s \in A$
  - By definition of f, $t \notin A$
  - $v(f) = \sum_{e \text{ } out \text{ } of \text{ } A} f(e) - \sum_{e \text{ } in \text{ } to \text{ } A} f(e)$
  - The second term must be zero, since there is no edge from A to $V \setminus A$ in the residual graph
  - In the first term, every $f(e)$ is equal to $c(e)$ since there is no edge from A to $V \setminus A$ in the residual graph
  - Therefore, $v(f) = \sum_{e \text{ } out \text{ } of \text{ } A} c(e) = cap(A, B)$

RUNNING TIME

- Assume all capacities are integers between 1 and C
  - Therefore, every flow and residual capacity is an integer
- The algorithm terminates after at most nC iterations
  - Each augmentation increases flow by 1
- Interesting theorem: If all capacities are integers, there exists an integer maximum flow

RUNNING TIME

- Is nC polynomial time?
  - No!
- Use care when selecting augmenting paths
  - Some choices lead to exponential time algorithms
  - Clever choices lead to polynomial time algorithms
- Multiple ideas work
  - Easy one: select path with maximum bottleneck capacity
    - Ok, this is a bit too slow maybe
  - Select path with large bottleneck capacity

CAPACITY SCALING

- Maintain scaling capacity Δ
- Let $G_f(\Delta)$ be the subgraph of the residual graph with edges of capacity at least Δ
  - Find all augmenting paths here
  - If none, update Δ:
    - $\Delta \leftarrow \Delta/2$

SUMMARY

- Max-flow = min-cut
- Ford-Fulkerson algorithm (7.1-7.3 in KT)