CS 7800: Advanced Algorithms

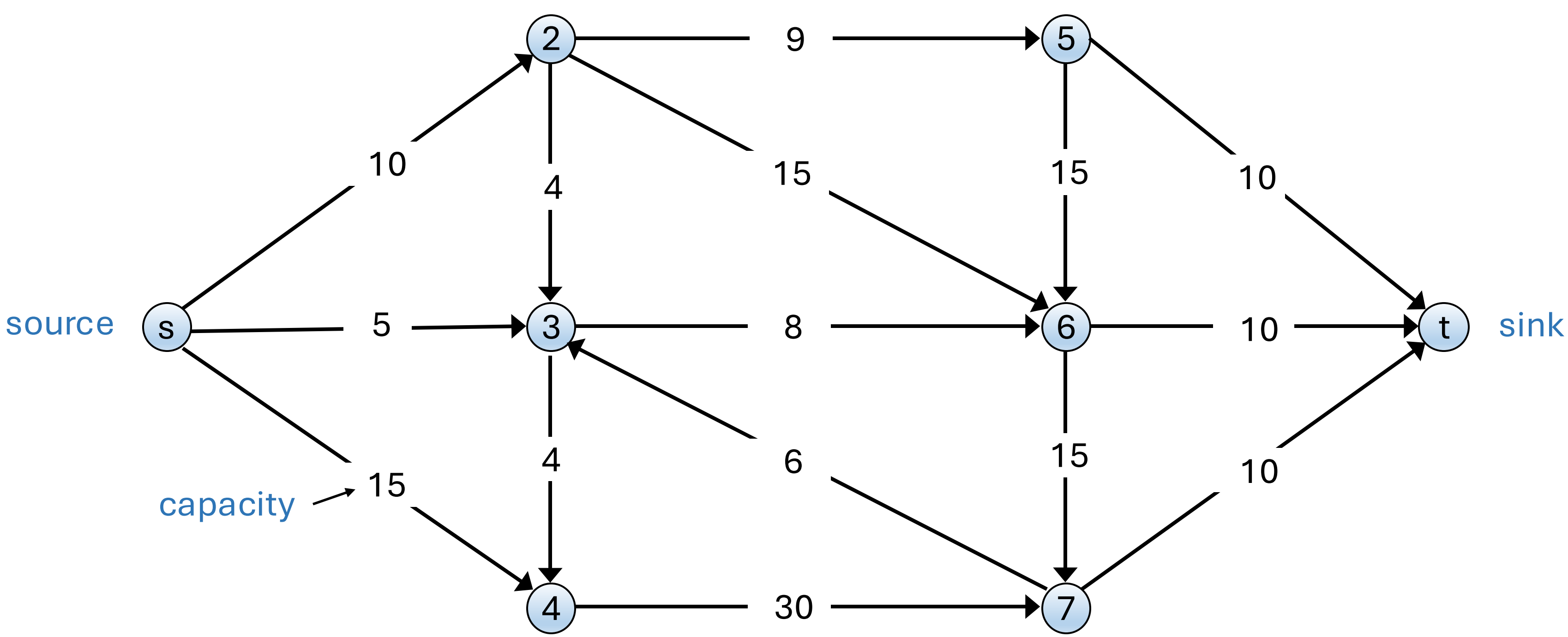
Class 7: Network Flow I

- Ford-Fulkerson
- Duality

Jonathan Ullman
September 26, 2025

Flow Networks

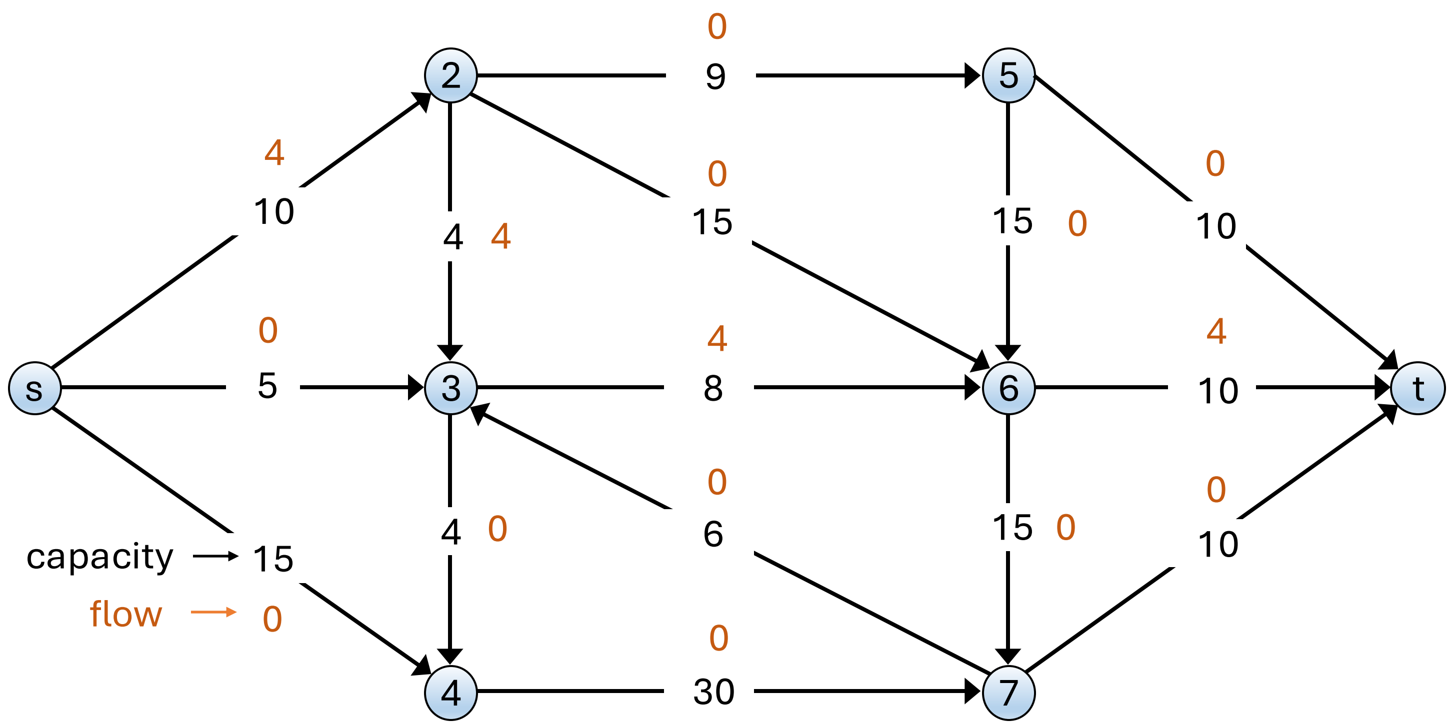
- Directed graph $G = (V, E)$
- Two special nodes: source s and sink t
- Edge capacities $c(e)$
- Assume strongly connected (for simplicity)



Flows

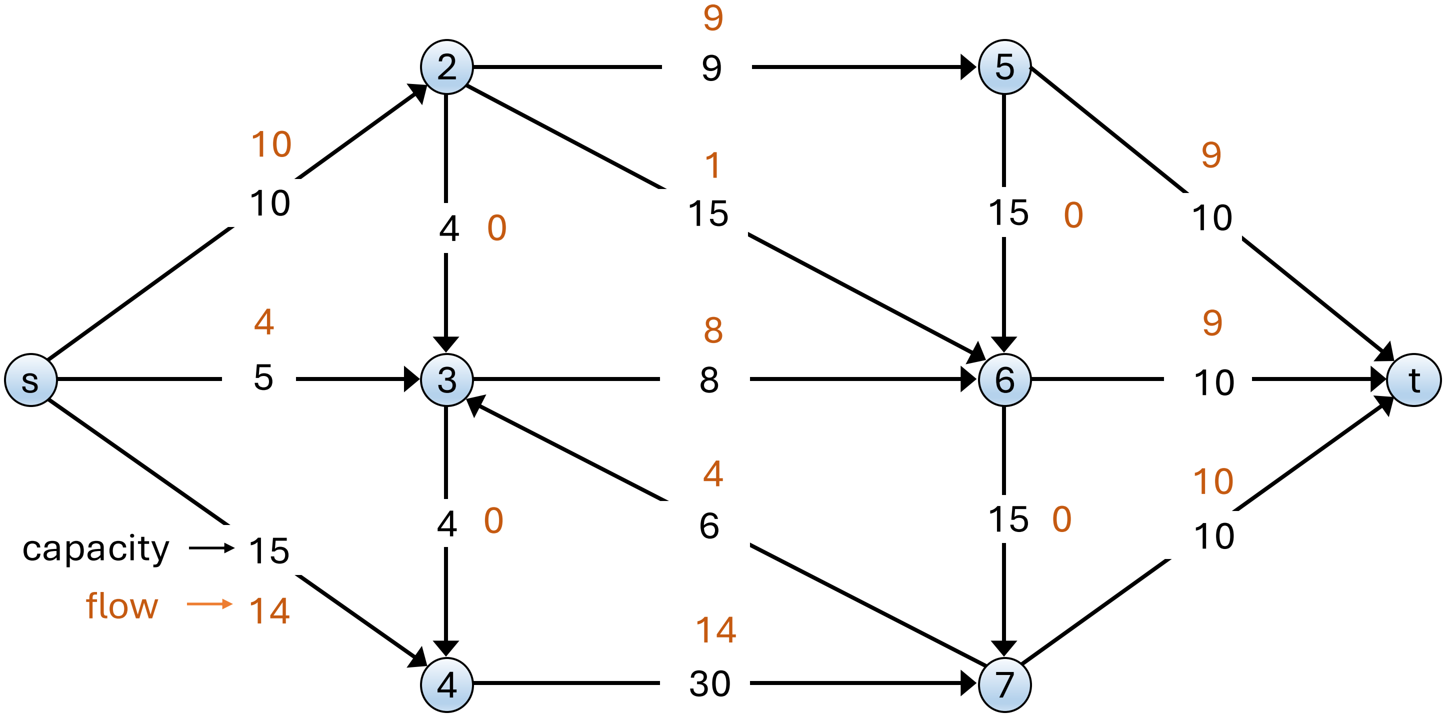
- An s-t flow is a function $f(e)$ such that
  - For every $e \in E$, $0 \leq f(e) \leq c(e)$ (capacity)
  - For every $v \in V \setminus \{s, t\}$, $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

- The value of a flow is $val(f) = \sum_{e \text{ out of } s} f(e)$



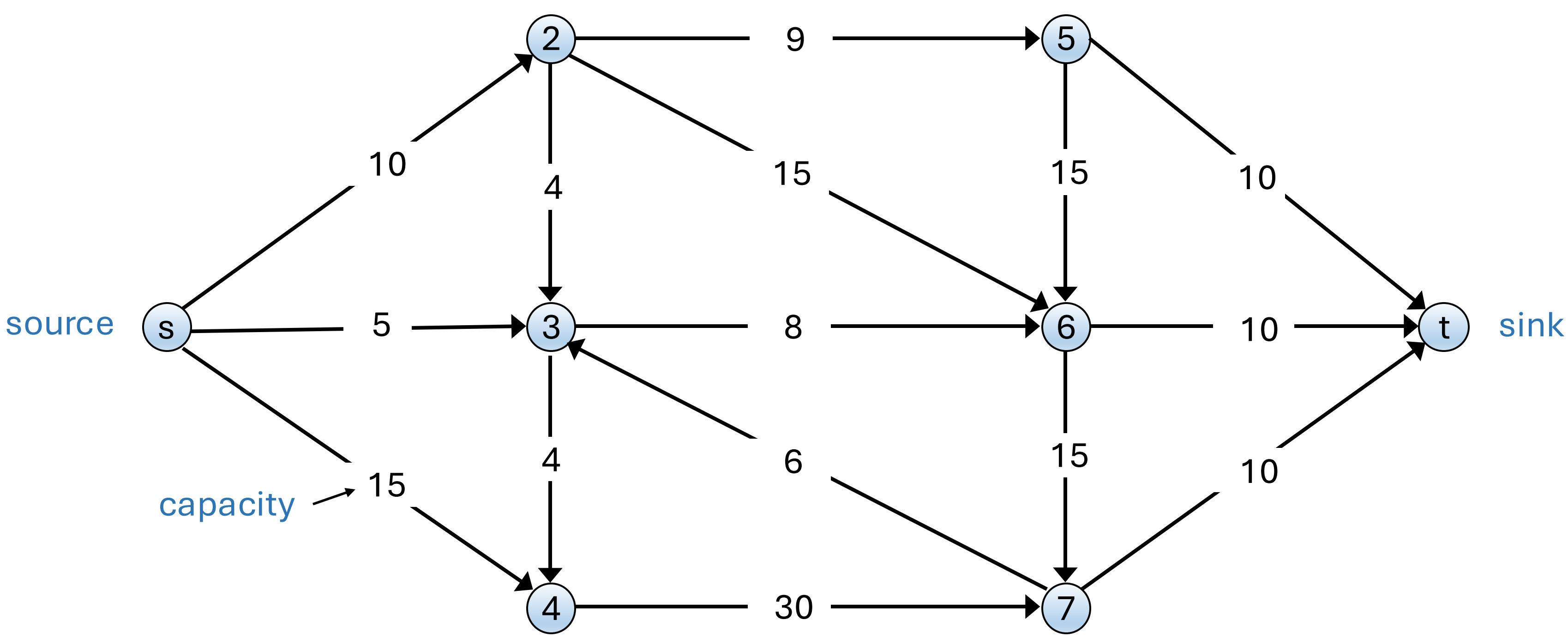
Maximum Flow Problem

- Given G = (V,E,s,t,{c(e)}), find an s-t flow of maximum value
- value(f) =



Cuts

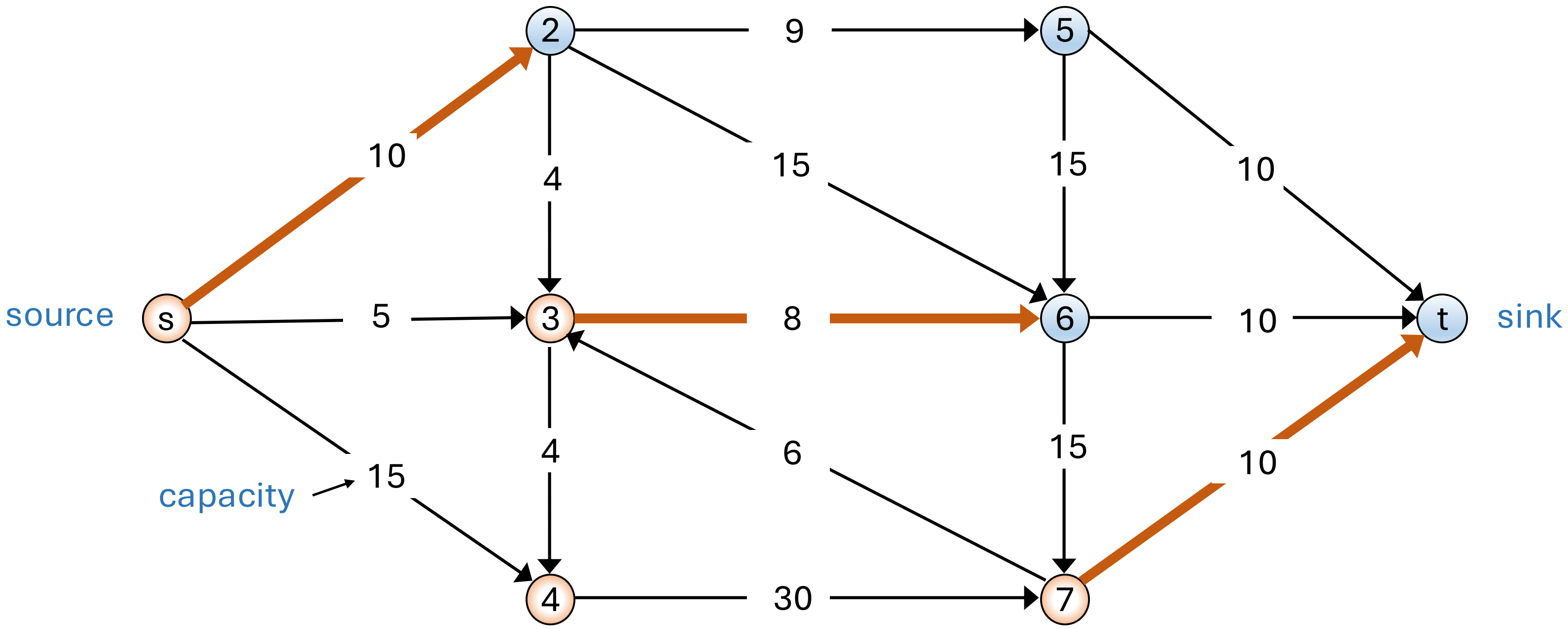
- An s-t cut is a partition (A, B) of V with $s \in A$ and $t \in B$
- The capacity of a cut (A,B) is $cap(A, B) = \sum_{e \text{ out of } A} c(e)$



Minimum Cut problem

- Given G = (V,E,s,t,{c(e)}), find an s-t cut of minimum capacity

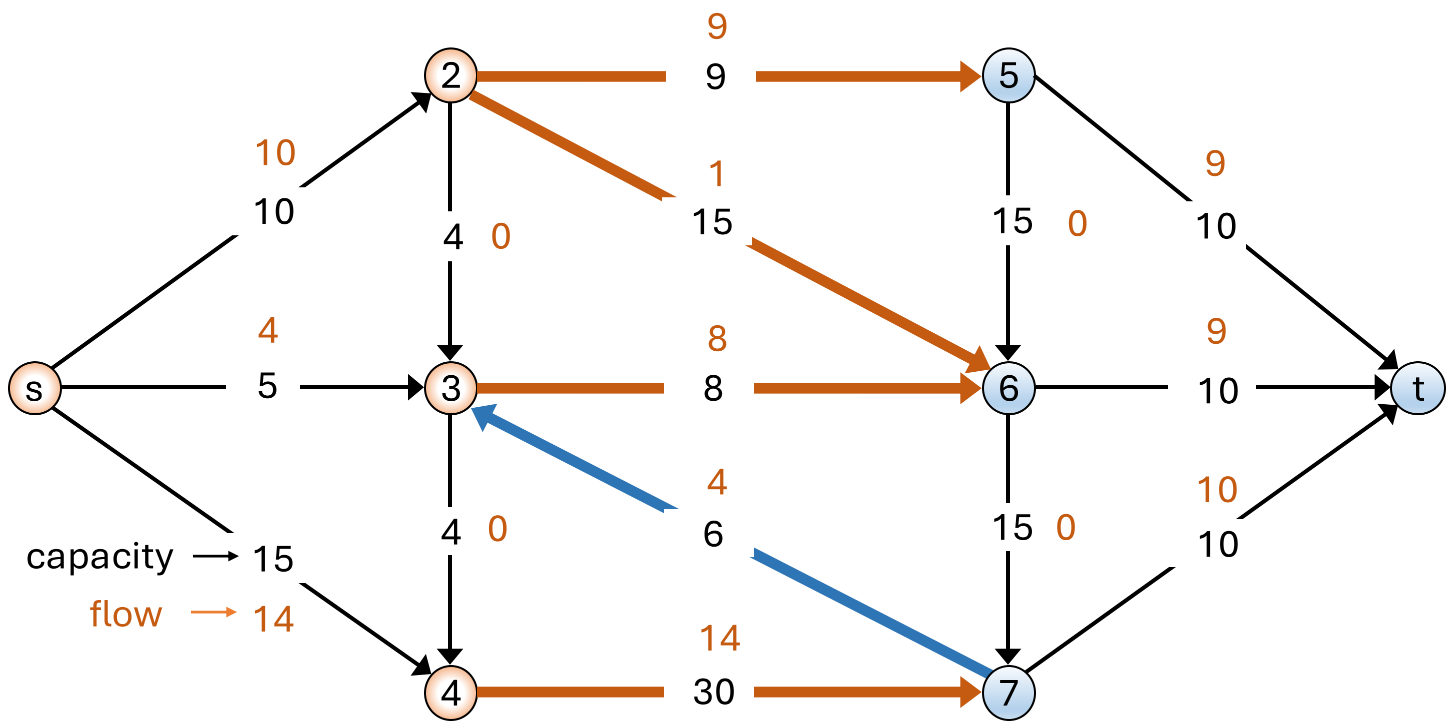
- cap({s,3,4,7}, {2,5,6,t}) =



Flows & Cuts: Closely Related

- **Fact:** If f is *any* s-t flow and (A, B) is any s-t cut, then the net flow across (A, B) is equal to the amount leaving s
- The net flow across any s-t cut is the same!

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



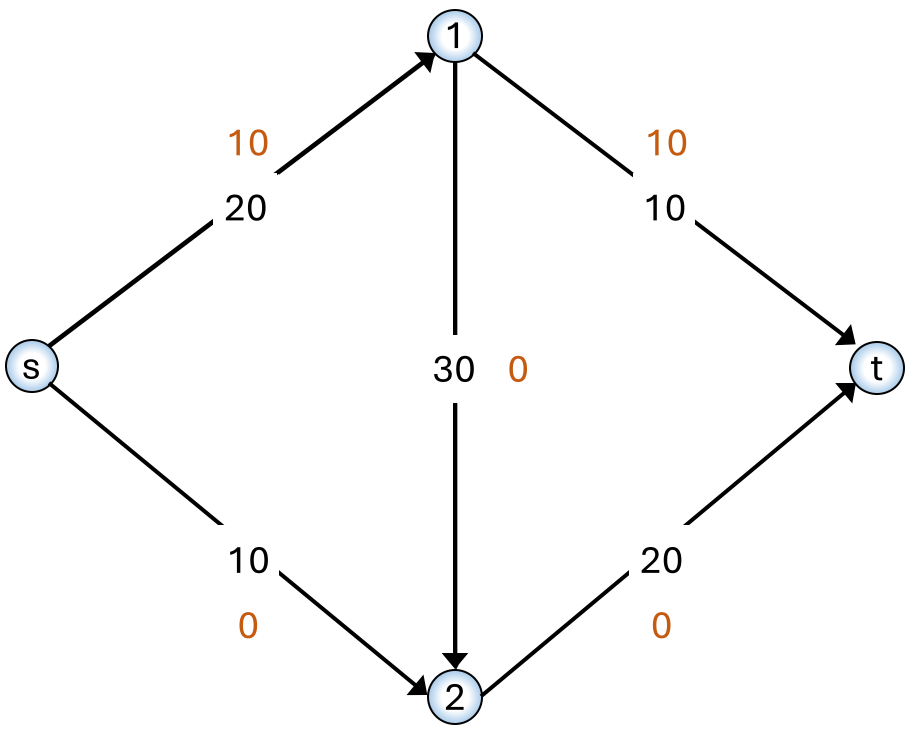
Cuts & Flows

- Let f be any s-t flow and (A, B) any s-t cut,

$$val(f) \leq cap(A, B)$$

Augmenting Paths

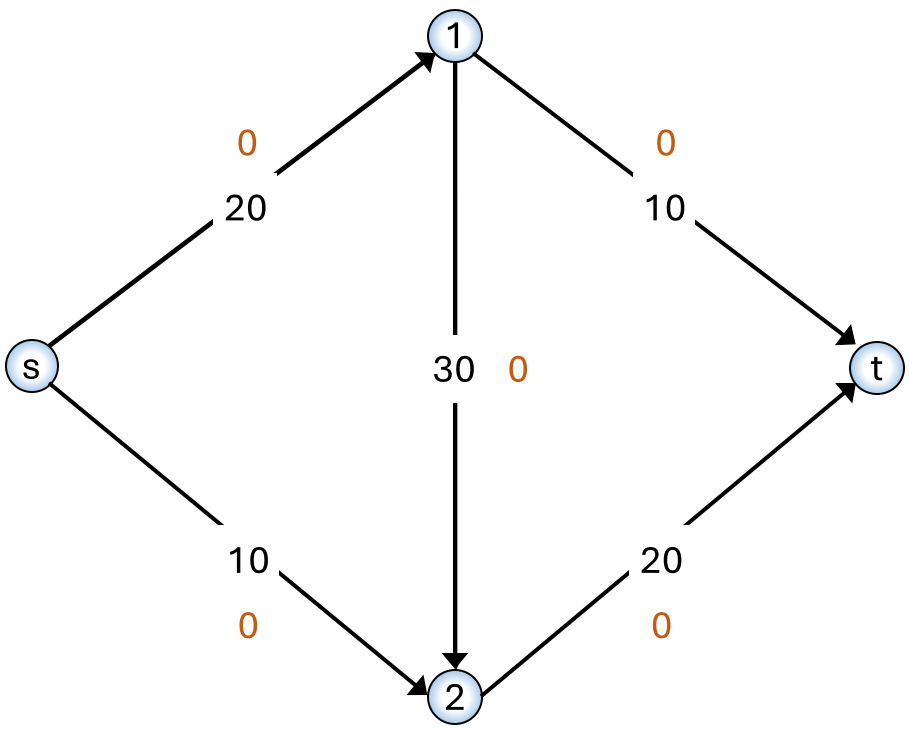
- Given a network $G = (V, E, s, t, \{c(e)\})$ and a flow f, an **augmenting path** P is a simple $s \to t$ path such that $f(e) < c(e)$ for every edge $e \in P$



- Are these augmenting paths?
  - s – 1 – t
  - s – 2 – t
  - s – 1 – 2 - t

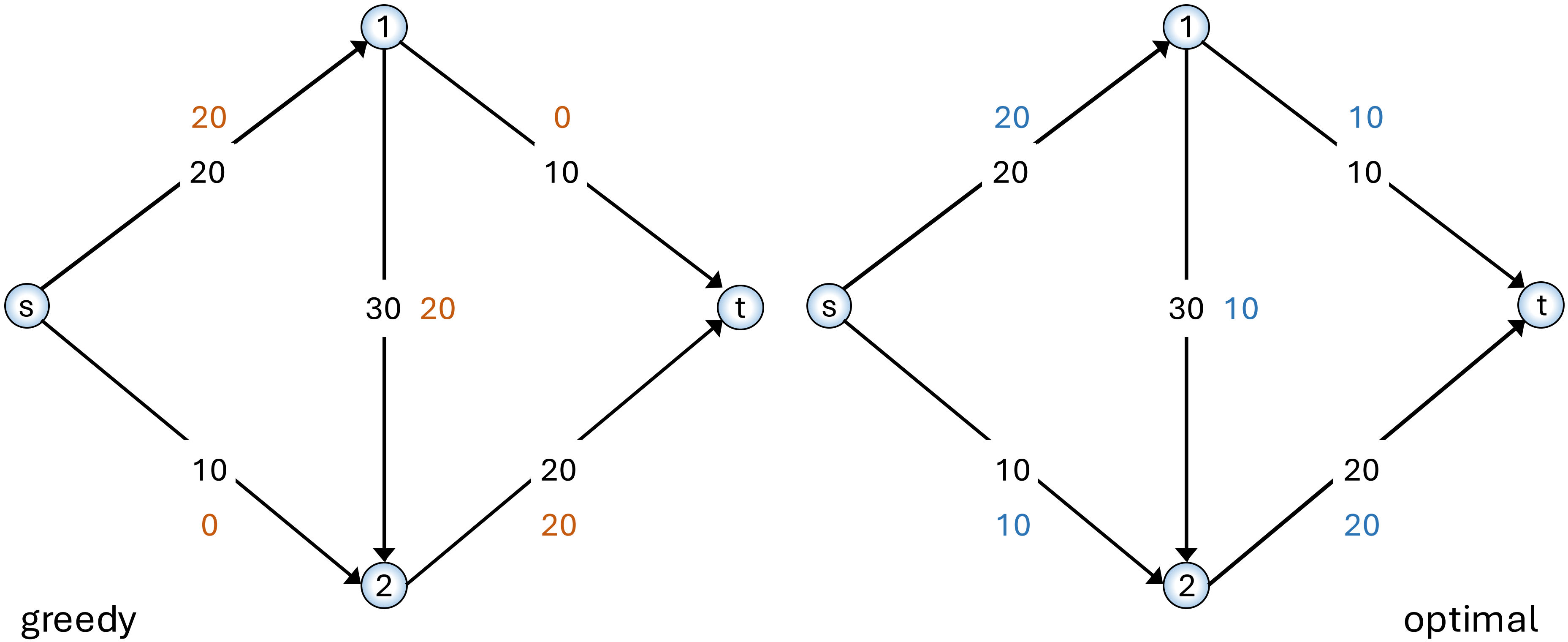
Greedy Max Flow

- Start with $f(e) = 0$ for all edges $e \in E$
- Find an **augmenting path** P & increase flow
- Repeat until you get stuck



Does Greedy Work?

- Greedy gets stuck before finding a max flow
- How can we get from our solution to the max flow?



Residual Graphs

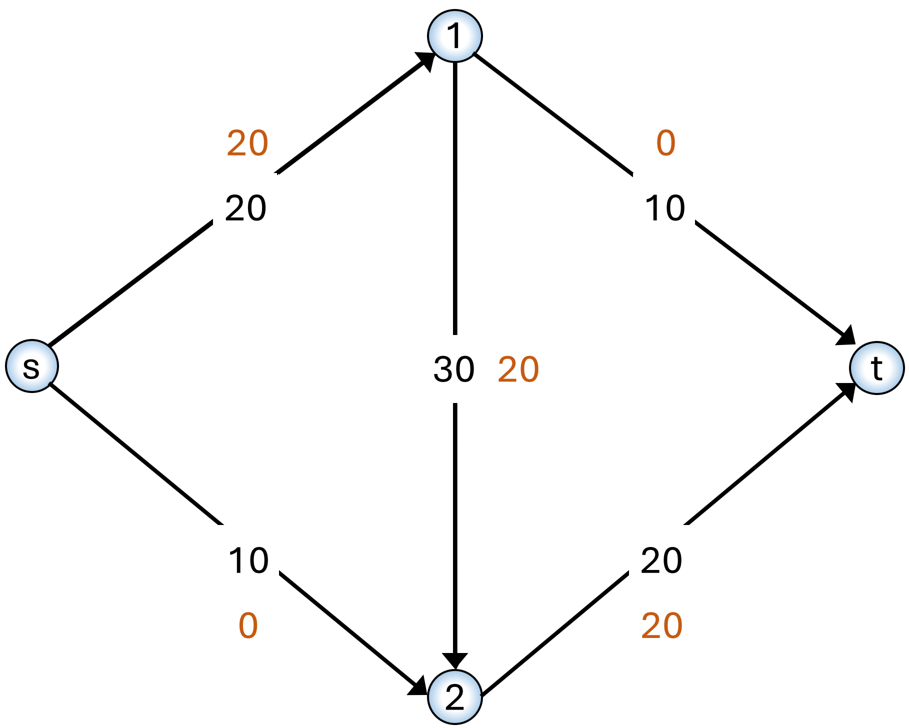
- Original edge: $e = (u, v) \in E$.
  - Flow $f(e)$, capacity $c(e)$
  - Residual capacity: *c(e) – f(e)*

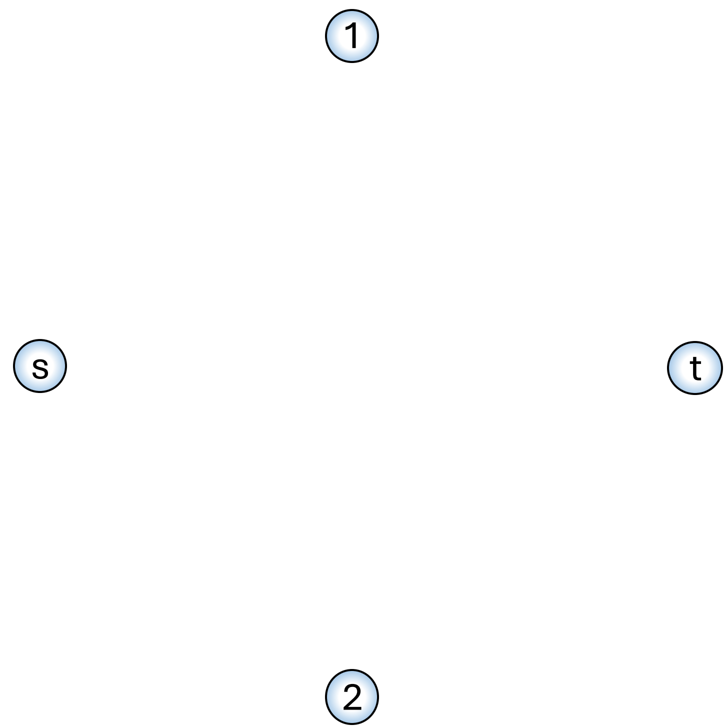
- Residual edge
  - Allows "undoing" flow
  - $e = (u, v)$ and $e^R = (v, u)$.
  - cap(e^R) = *f(e)*

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

Ford-Fulkerson Algorithm

- Start with $f(e) = 0$ for all edges $e \in E$
- Find an **augmenting path** P in the **residual graph**
- Repeat until you get stuck





Augmenting Paths in Residual Graphs

- Let G_f be a **residual graph**
- Let P be an augmenting path in the **residual graph**
- **Fact:** $f' = \text{Augment}(G_f, P)$ is a valid flow

```
Augment(G_f, P)
    b ← the minimum capacity of an edge in P
    for e ∈ P
        if (e is an original edge):
            f(e) ← f(e) + b
        else:
            f(e^R) ← f(e^R) - b
    return f
```

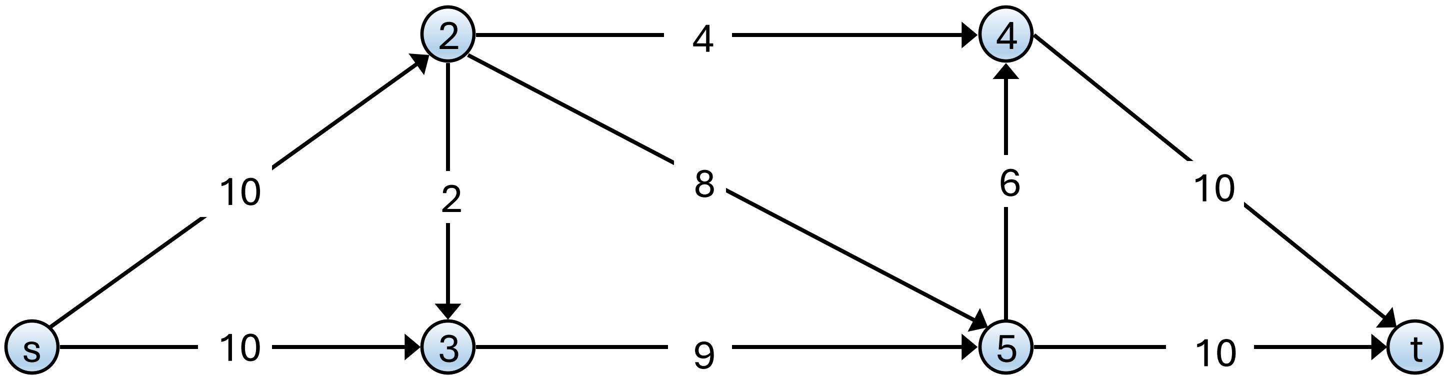
Ford-Fulkerson Algorithm

```
FordFulkerson(G,s,t,{c(e)})
    for e ∈ E: f(e) ← 0
    G_f is the residual graph

    while (there is an s-t path P in G_f)
        f ← Augment(G_f,P)
        update G_f

    return f
```

```
Augment(G_f, P)
    b ← the minimum capacity of an edge in P
    for e ∈ P
        if (e is an original edge): f(e) ← f(e) + b
        else:  f(e^R) ← f( e^R) - b
    return f
```

Ford-Fulkerson Demo

G:



G_f:

What do we want to prove?

Running Time of Ford-Fulkerson

- For **integer capacities**, $\leq val(f^*)$ augmentation steps

- Can perform each augmentation step in $O(m)$ time
  - find augmenting path in $O(m)$
  - augment the flow along path in $O(n)$
  - update the residual graph along the path in $O(n)$

- For integer capacities, FF runs in $O\big(m \cdot val(f^*)\big)$ time
  - $O(mn)$ time if all capacities are $c_e = 1$
  - $O(mnC_{\max})$ time for any integer capacities $\leq C_{\max}$
  - Problematic when capacities are large—more on this later!

Optimality of Ford-Fulkerson

- **Theorem:** f is a maximum s-t flow if and only if there is no augmenting s-t path in G_f

- **Strong MaxFlow-MinCut Duality:** The value of the max s-t flow equals the capacity of the min s-t cut

- We'll prove that the following are equivalent for all f
  1. There exists a cut (A, B) such that $val(f) = cap(A, B)$
  2. Flow f is a maximum flow
  3. There is no augmenting path in G_f

Optimality of Ford-Fulkerson

- **Theorem:** the following are equivalent for all f
  1. There exists a cut (A, B) such that $val(f) = cap(A, B)$
  2. Flow f is a maximum flow
  3. There is no augmenting path in G_f

Optimality of Ford-Fulkerson

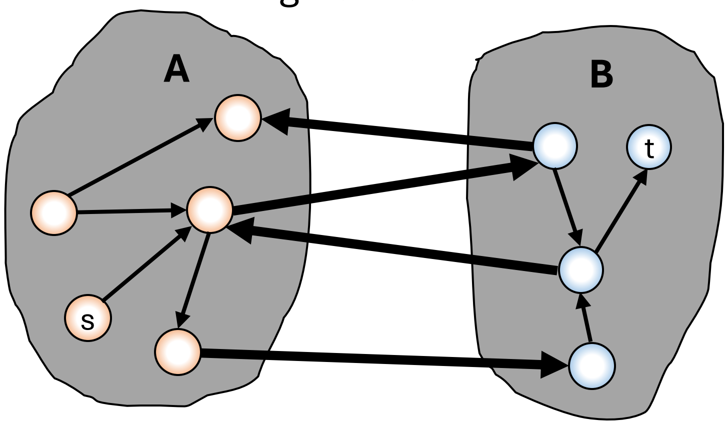
- **(3 → 1)** If there is no augmenting path in G_f, then there is a cut (A, B) such that $val(f) = cap(A, B)$
  - Let A be the set of nodes reachable from s in G_f
  - Let B be all other nodes

Optimality of Ford-Fulkerson

- **(3 → 1)** If there is no augmenting path in G_f, then there is a cut (A, B) such that $val(f) = cap(A, B)$
  - Let A be the set of nodes reachable from s in G_f
  - Let B be all other nodes
  - **Key observation:** no edges in G_f go from A to B

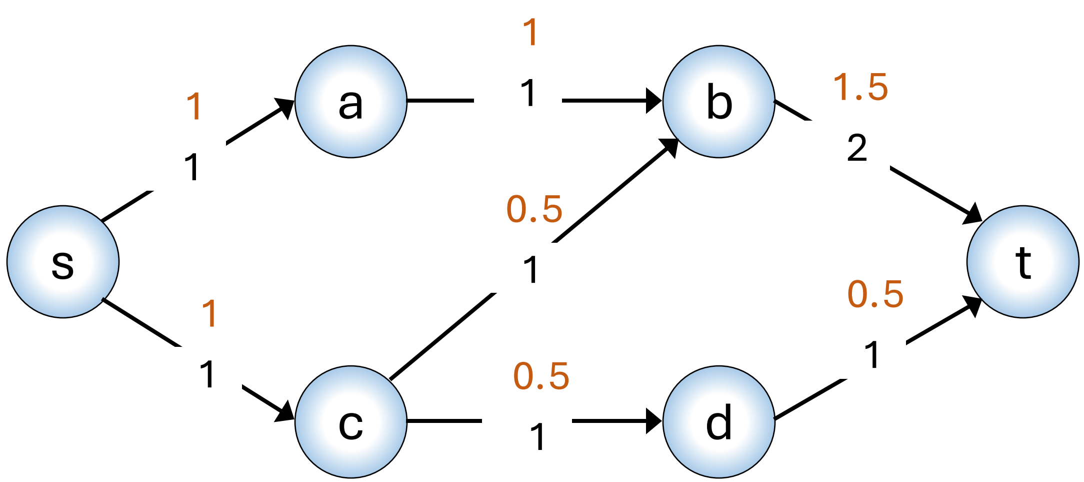
- If e is $A \to B$, then $f(e) = c(e)$
- If e is $B \to A$, then $f(e) = 0$

original network



Ask the Audience

- Is this a maximum flow?



- Is there an **integer maximum flow**?
- Does every graph with **integer capacities** have an **integer maximum flow**?

Summary

- **The Ford-Fulkerson Algorithm solves maximum s-t flow**
  - Running time $O\big(m \cdot val(f^*)\big)$ in networks with integer capacities
- **Strong MaxFlow-MinCut Duality: max flow = min cut**
  - The value of the max s-t flow equals the capacity of the min s-t cut
  - If f^* is a maximum s-t flow, then the set of nodes reachable from s in G_{f^*} gives a minimum cut
  - Given a max-flow, can find a min-cut in time $O(n + m)$
- **Every graph with integer capacities has an integer maximum flow**
  - Ford-Fulkerson will return an integer maximum flow
  - Will be super important later