- Office hours on for today
- HW6 + 7 both due Mar 20th

CS3000: Algorithms & Data
Jonathan Ullman

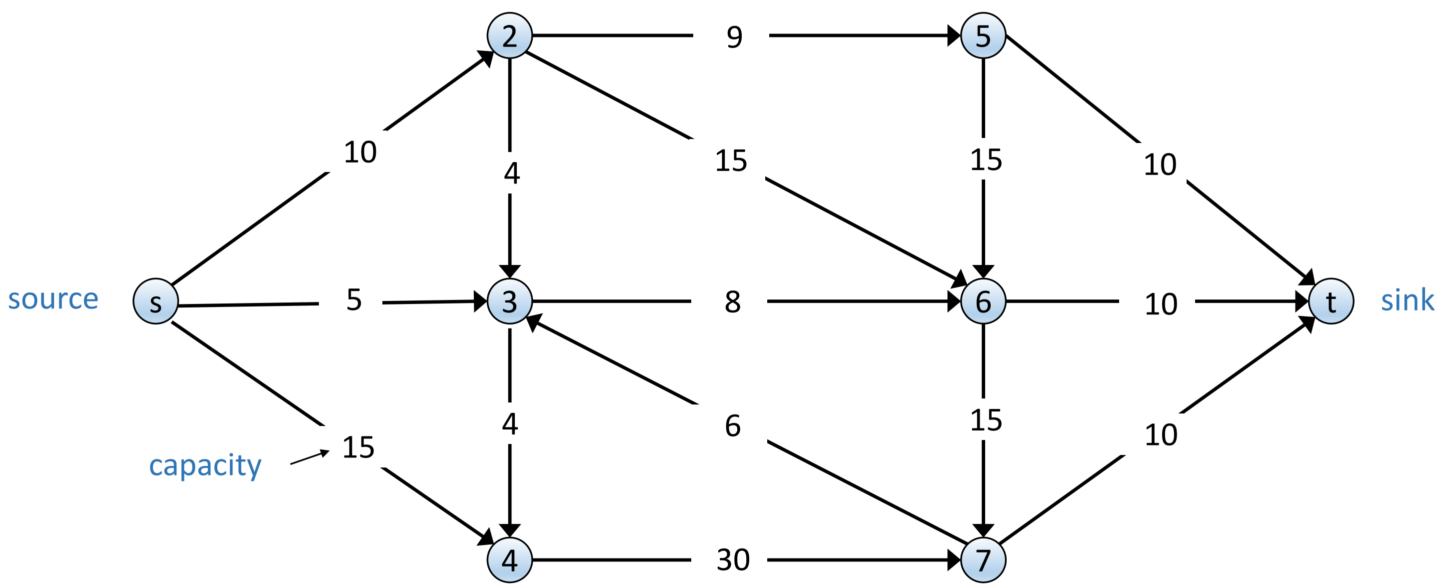
Lecture 14:
- Network Flow: flows, cuts, duality
- Ford-Fulkerson

Mar 11, 2020

Flow Networks

Flow Networks

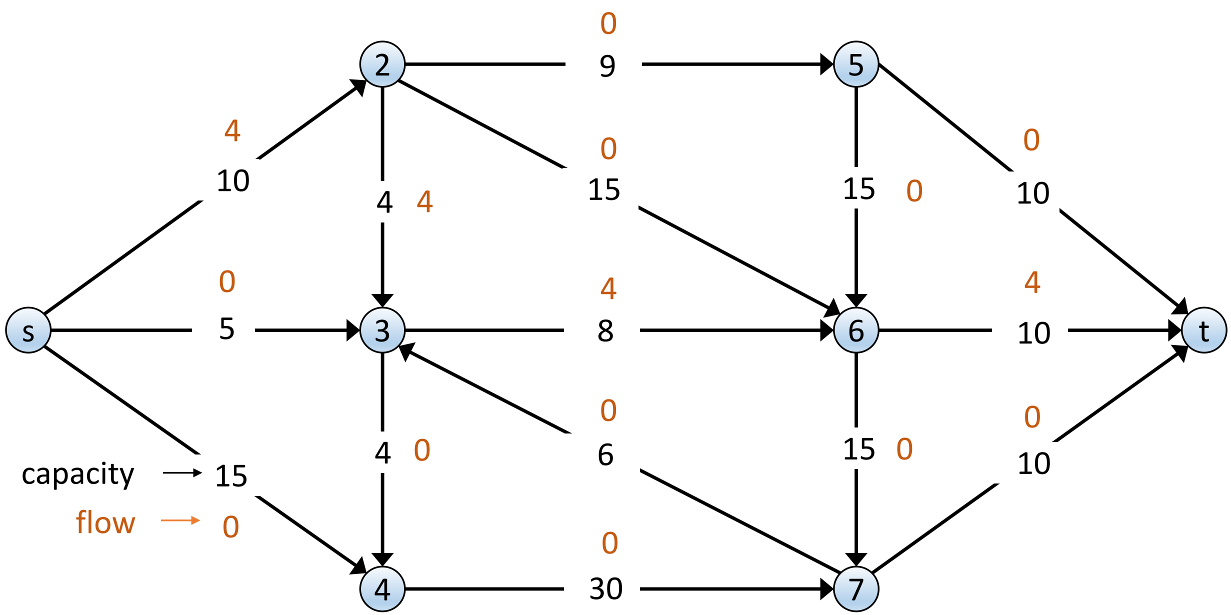
- Directed graph $G = (V, E)$
- Two special nodes: source s and sink t
- Edge capacities $c(e)$



Flows

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

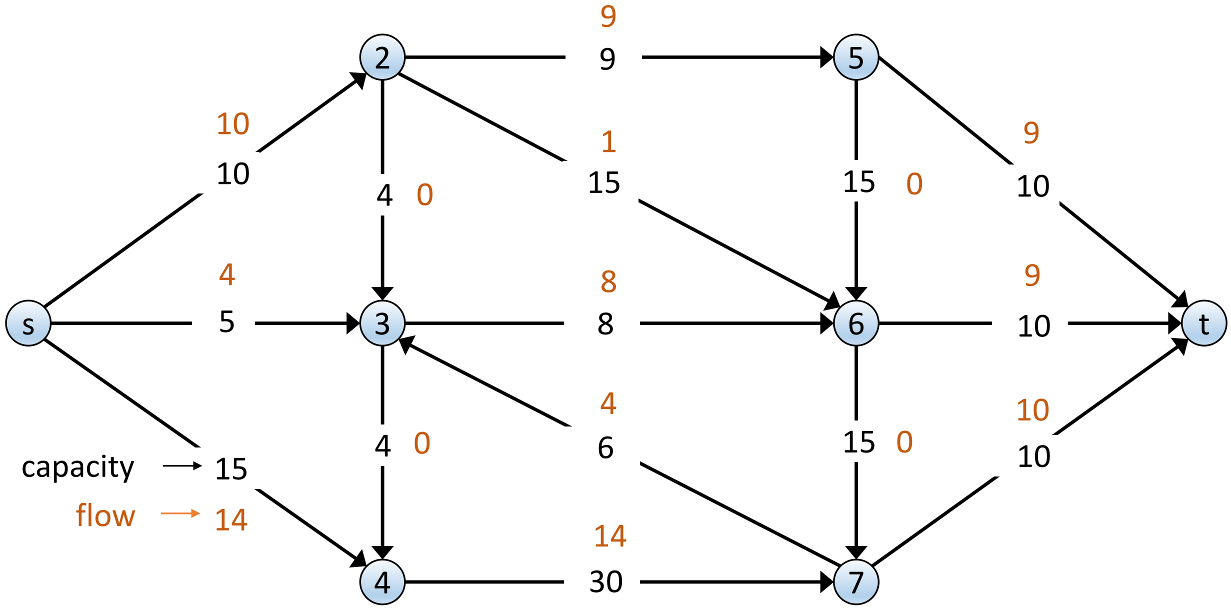
- The value of a flow is $val(f) = \sum_{e \text{ out of } s} f(e)$



Maximum Flow Problem

- Given $G = (V, E, s, t, \{c(e)\})$, find an s-t flow of max. value

$val(f) = 10 + 4 + 14 = 28$



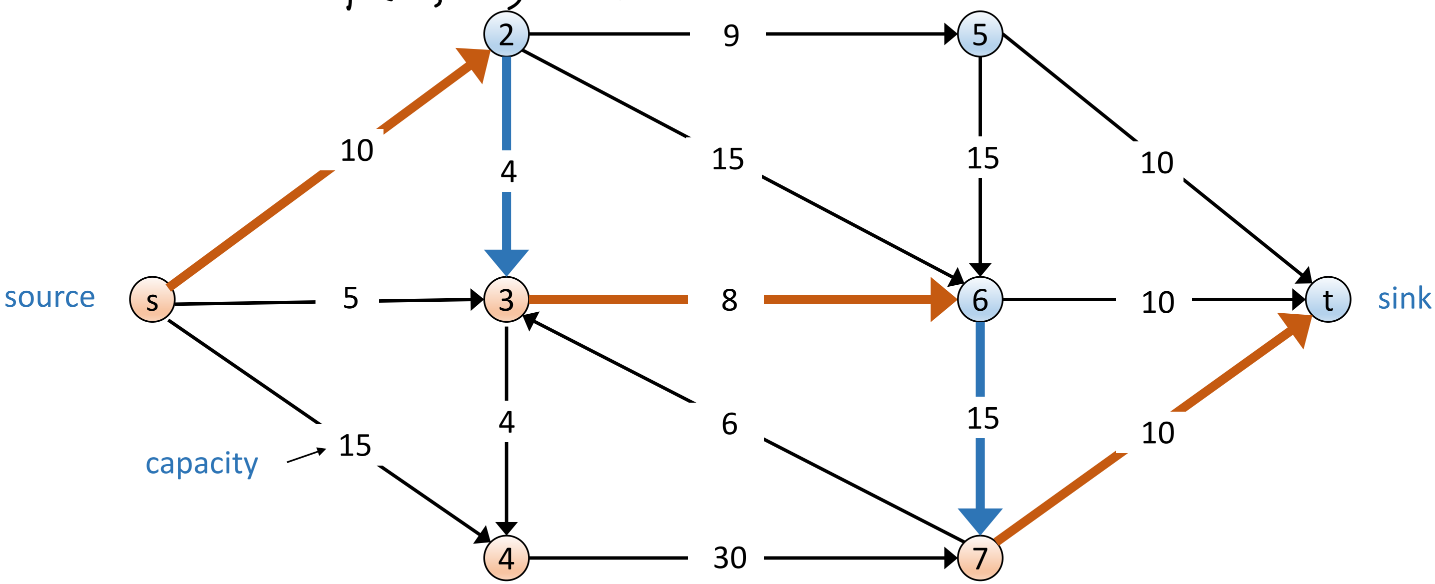
Cuts

- 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 $cap(A, B) = \sum_{e \text{ out of } A} c(e)$

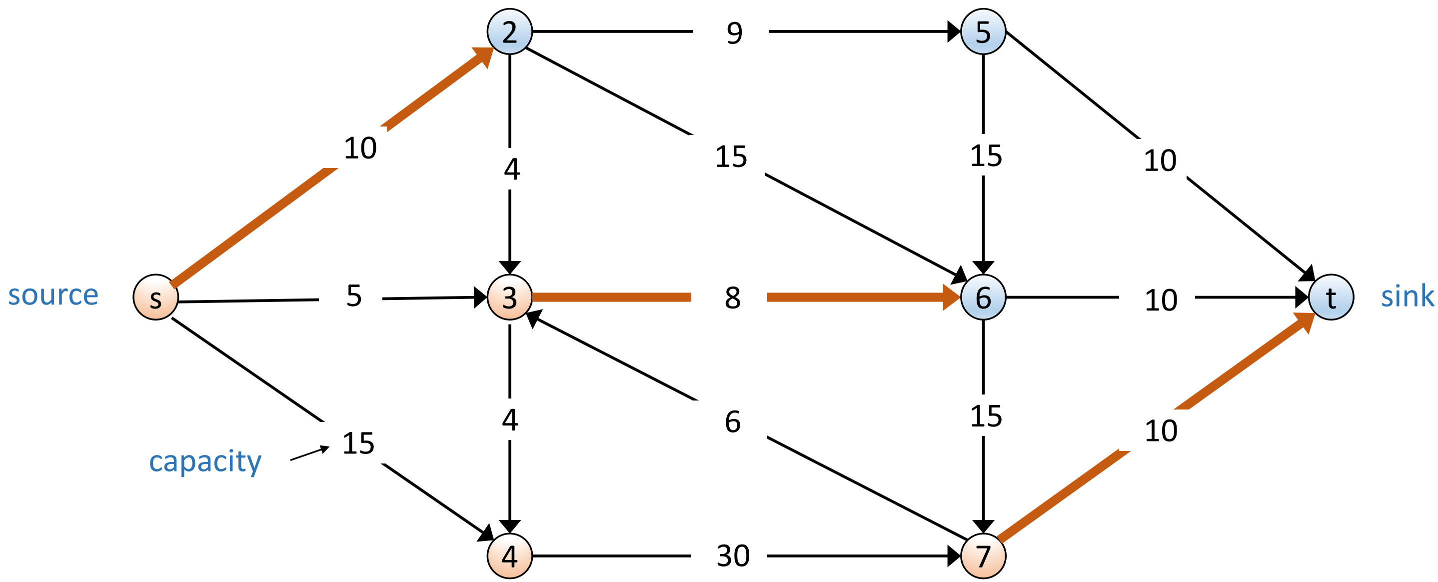
$A = \{s, 3, 4, 7\}$ $B = \{2, 5, 6, t\}$

$cap(A, B) = 10 + 8 + 10 = 28$



Minimum Cut problem

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



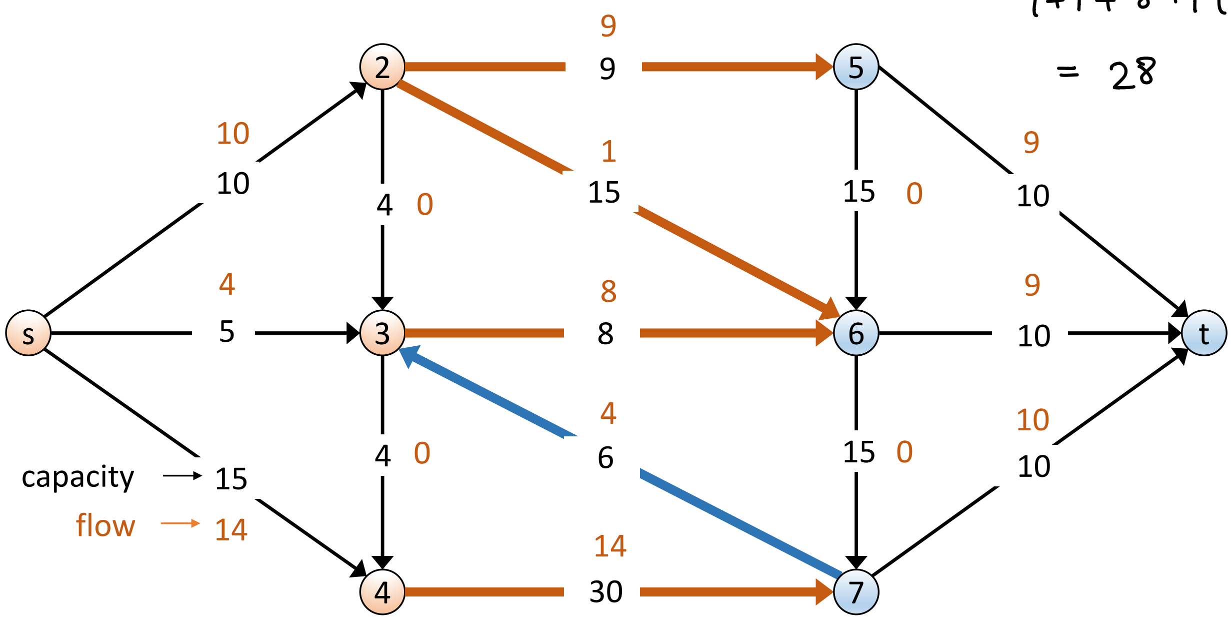
Flows vs. Cuts

- **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

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

$9+1+8+14-4$

$= 28$



Weak MaxFlow-MinCut Duality

- For any s-t flow f and any s-t cut (A, B) $val(f) \le cap(A, B)$

$$val(f) = \sum_{e \text{ from } A \to B} f(e) - \sum_{e \text{ from } B \to A} f(e)$$

$$\le \sum_{e \text{ from } A \to B} f(e) \qquad \text{(non-negativity)}$$

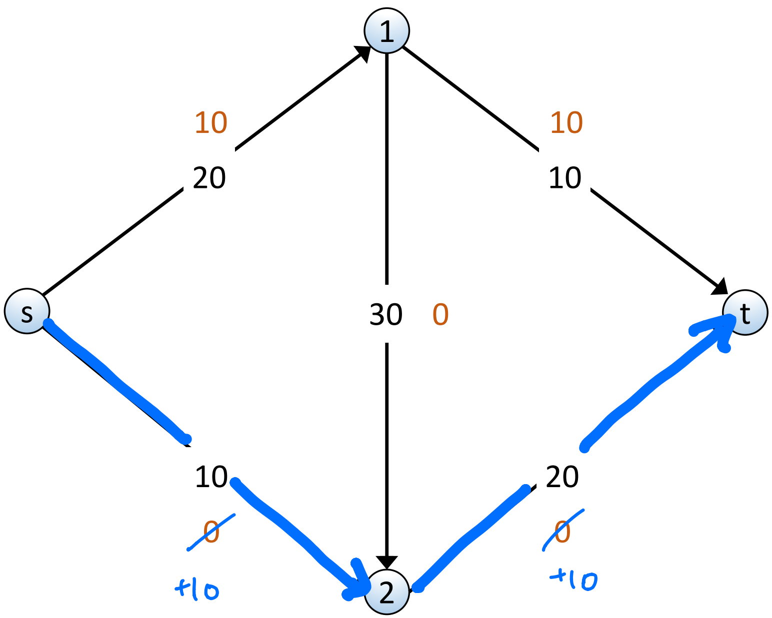
$$\le \sum_{e \text{ from } A \to B} c(e) = cap(A, B) \qquad \text{(capacity)}$$

- If f is a flow, (A, B) is a cut, and $val(f) = cap(A, B)$, then f is a max flow and (A, B) is a min cut

Augmenting Paths

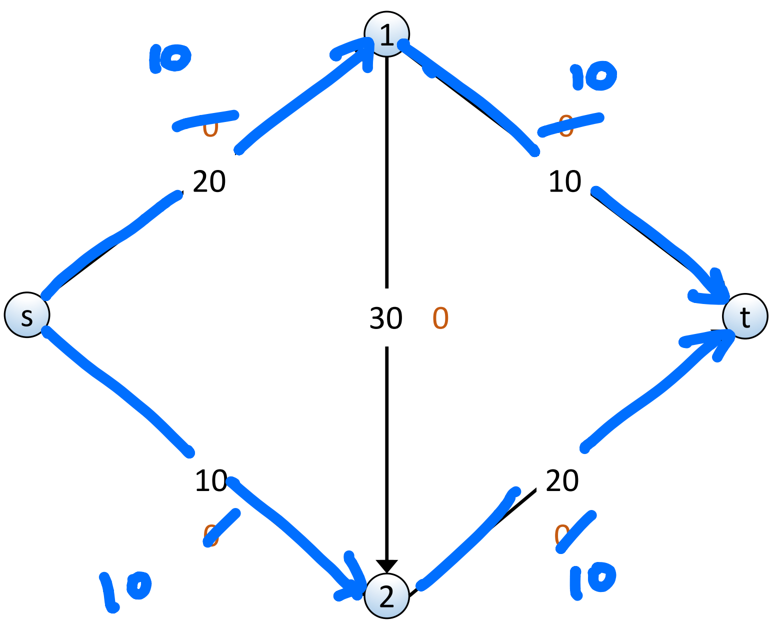
- Given a network $G = (V, E, s, t, \{c(e)\})$ and a flow f, an **augmenting path** P is an $s \to t$ path such that $f(e) < c(e)$ for every edge $e \in P$

Adding flow uniformly on an augmenting path gives a new valid s-t flow.



Greedy Max Flow

- Start with $f(e) = 0$ for all edges $e \in E$
- Find an **augmenting path** P (Add as much as you can on the path.)
- Repeat until you get stuck

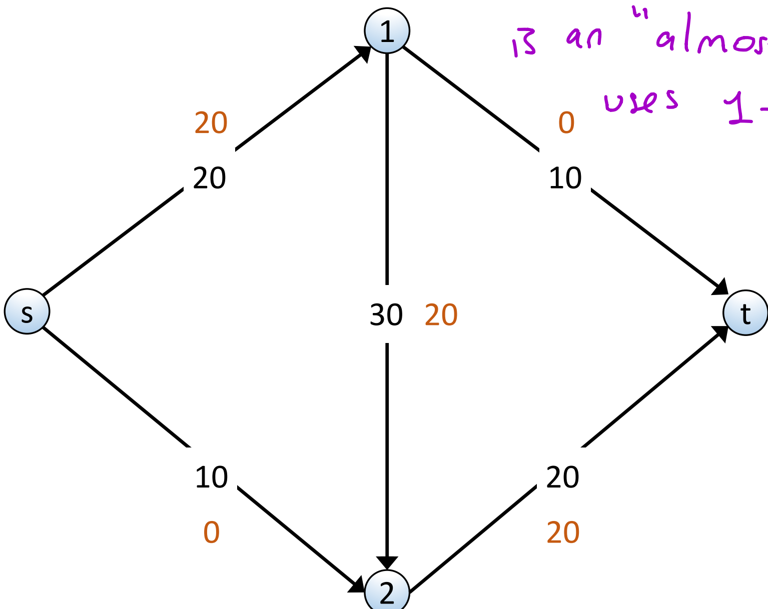


Does Greedy Work?

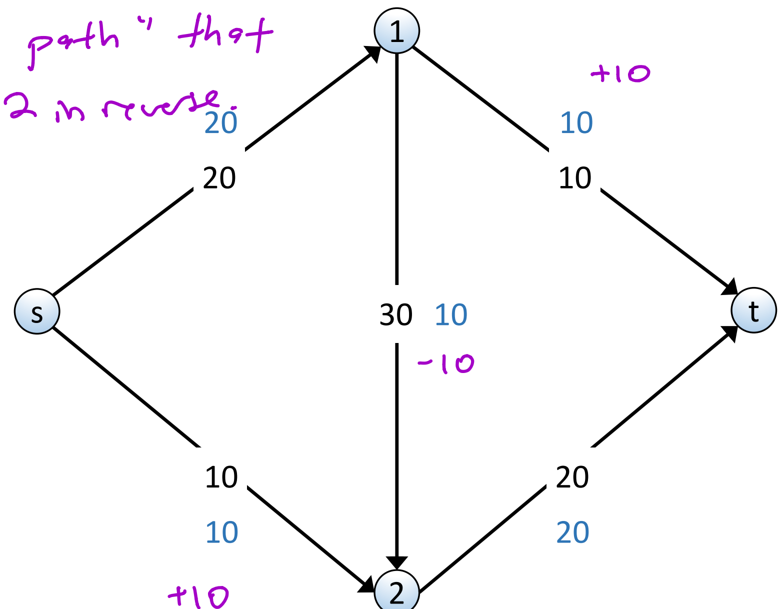
- Greedy gets stuck before finding a max flow
- How can we get from our solution to the max flow?

= diff btw two flows

The difference btw the two flows is an "almost path" that uses $1 \to 2$ in reverse.

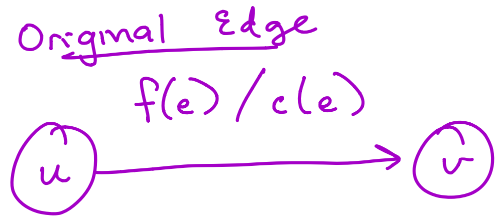


greedy

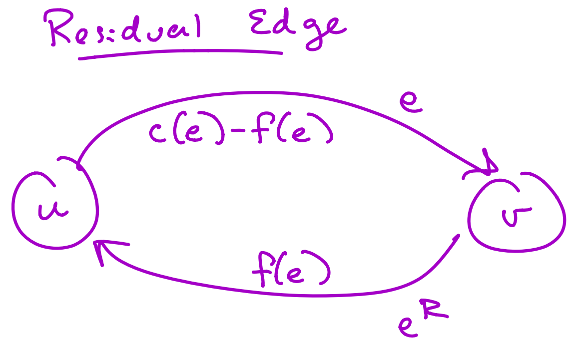


optimal

Residual Graphs



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



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

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

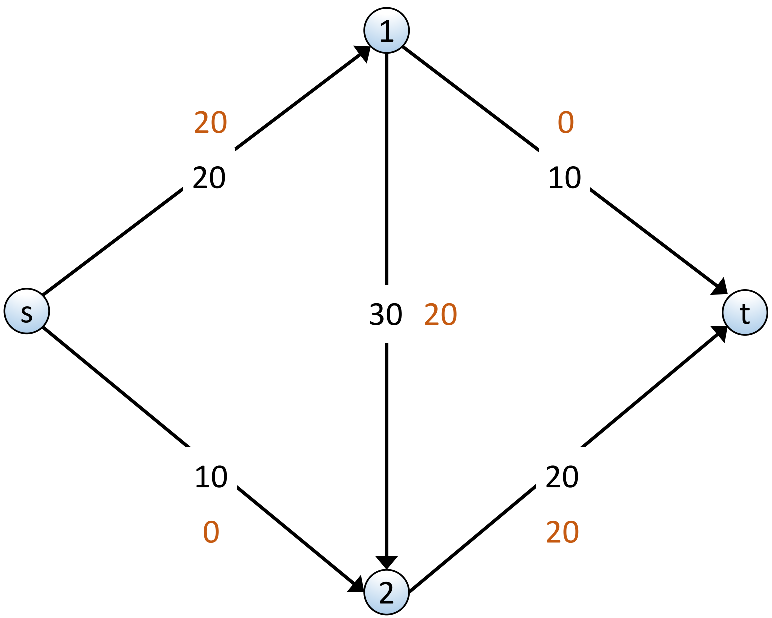
Augmenting Paths in Residual Graphs

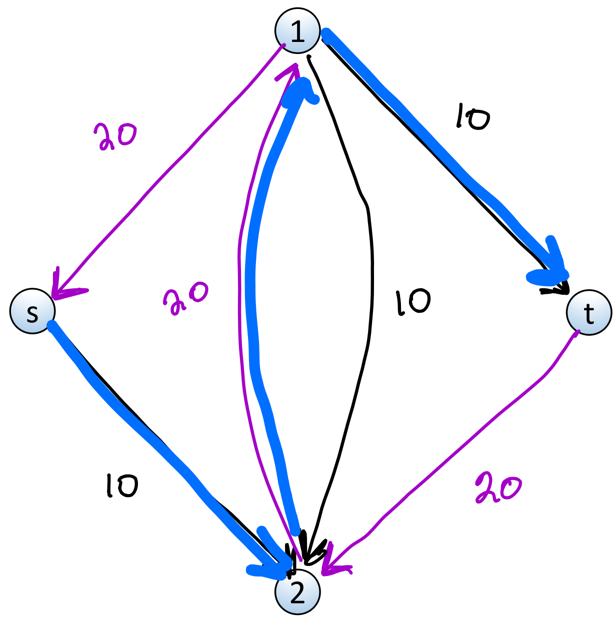
- Let G_f be a **residual graph**
- Let P be an augmenting path in the **residual graph**
- **Fact:** $f' = \text{Augment}(G_f, P)$ is a valid flow

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

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





Ford-Fulkerson Algorithm

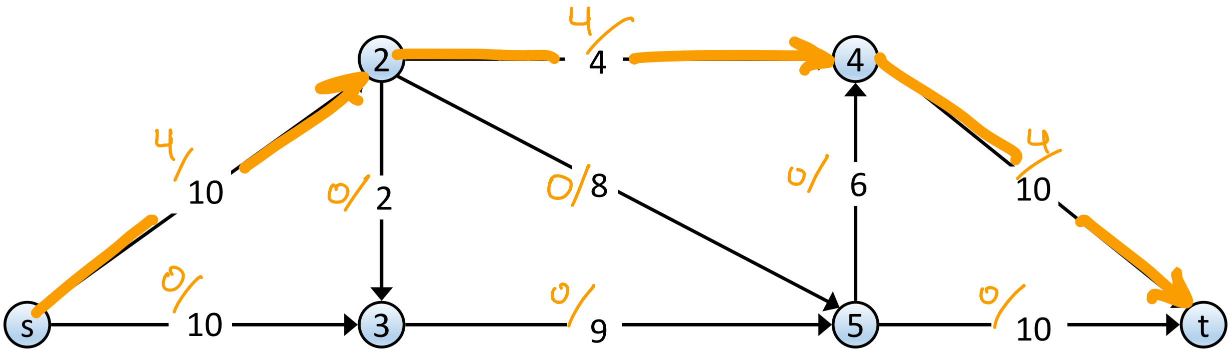
```
FordFulkerson(G,s,t,{c(e)})
    for e ∈ E: f(e) ← 0
    Gf is the residual graph
      "G
    while (there is an s-t path P in Gf)
        f ← Augment(Gf,P) // O(m+n) to augment
        update Gf  // O(m+n) to update the residual graph

    return f
```

Finding a path takes O(m+n) by BFS

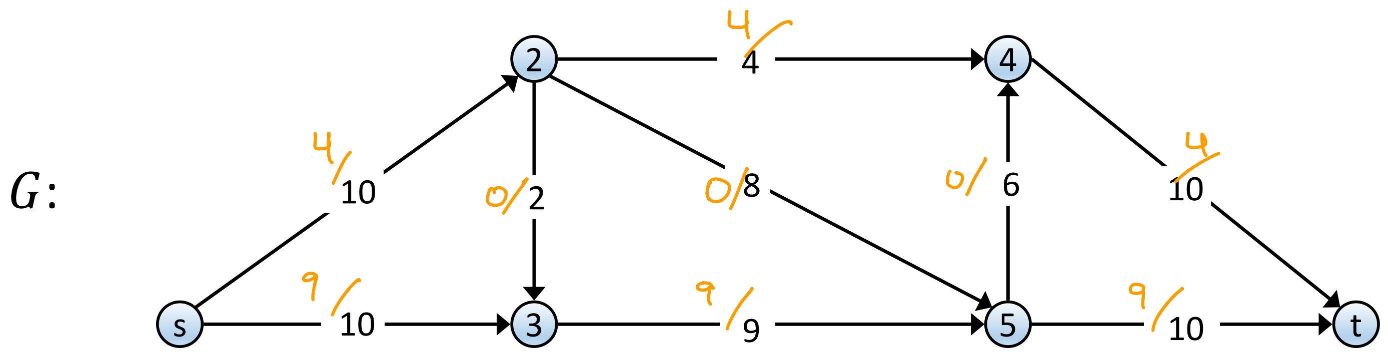
```
Augment(Gf, P)
    b ← the minimum capacity of an edge in P
    for e ∈ P
        if e ∈ E:   f(e) ← f(e) + b
        else:       f(e) ← f(e) - b
    return f
```

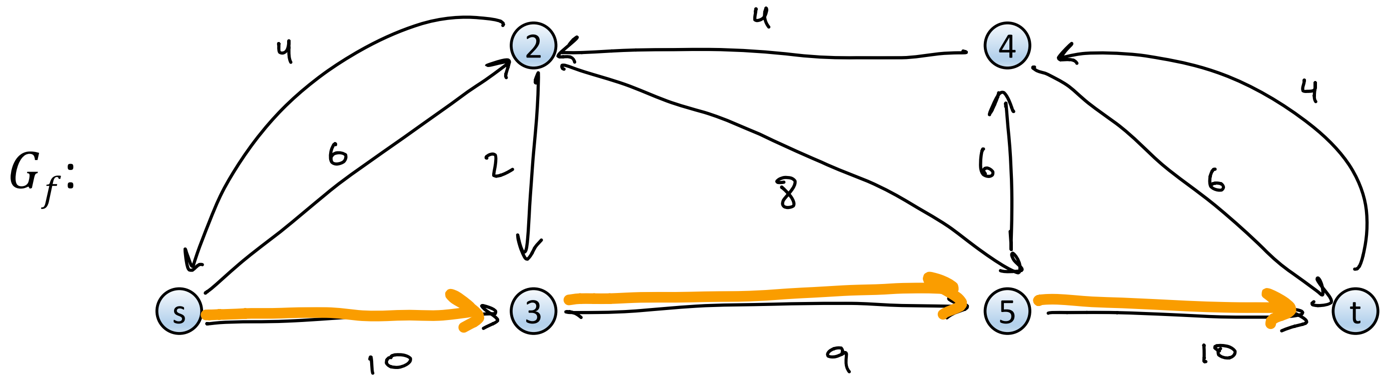
Ford-Fulkerson Demo

G:



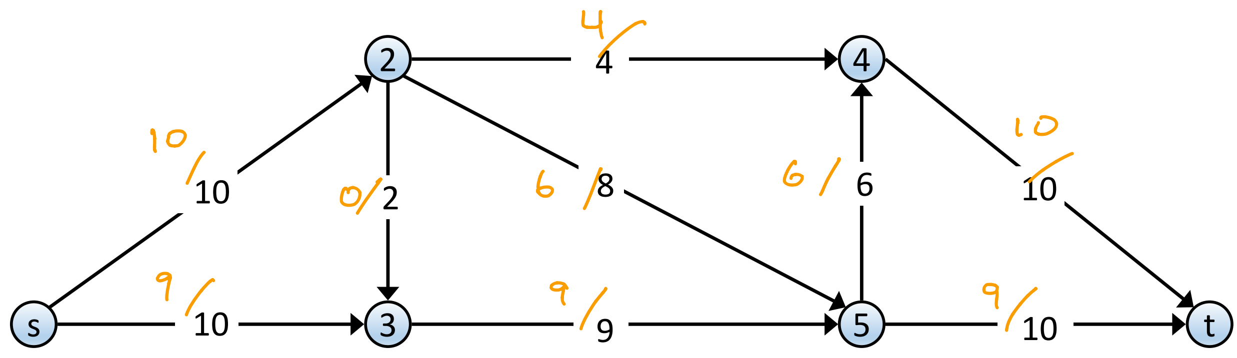
Ford-Fulkerson Demo



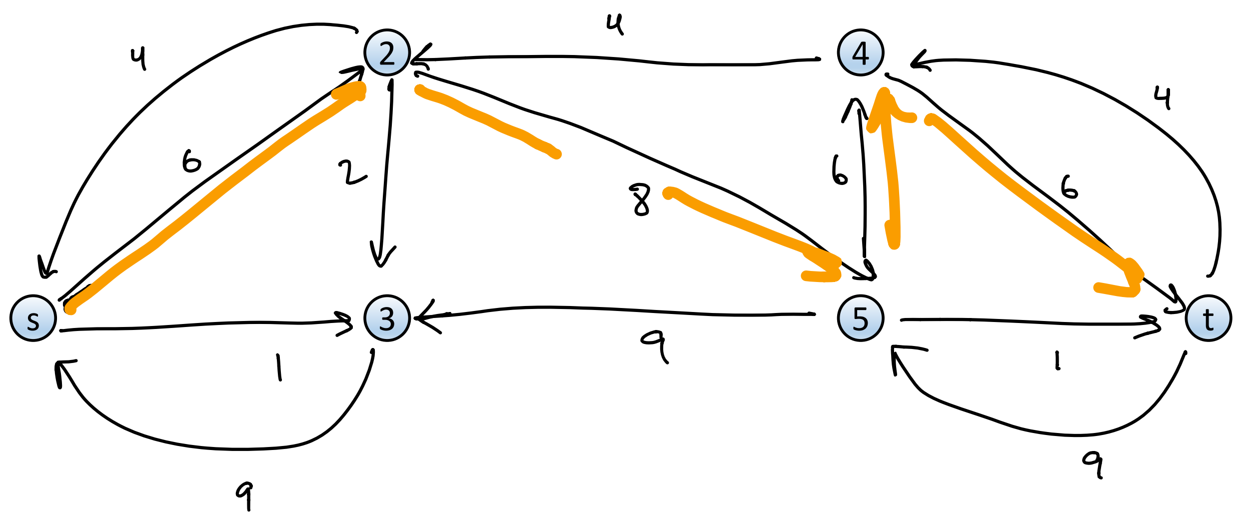


Ford-Fulkerson Demo

G:



G_f:

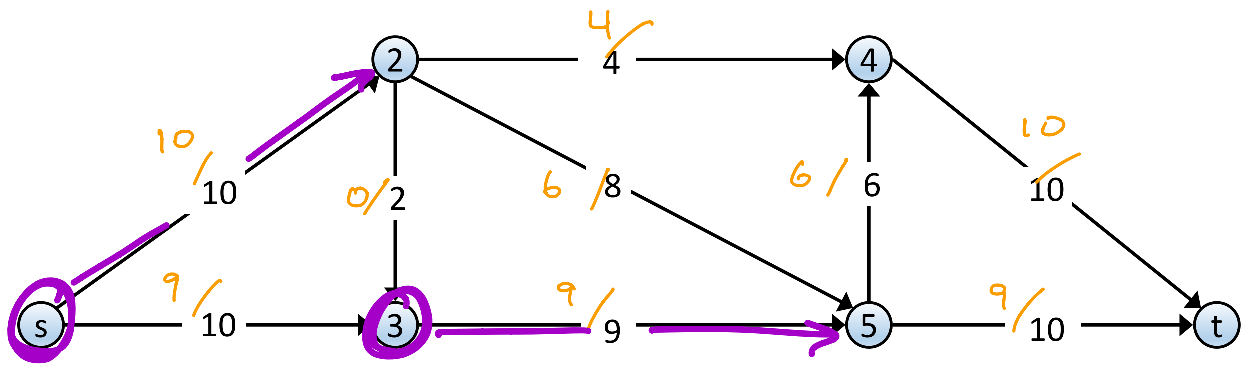


Ford-Fulkerson Demo

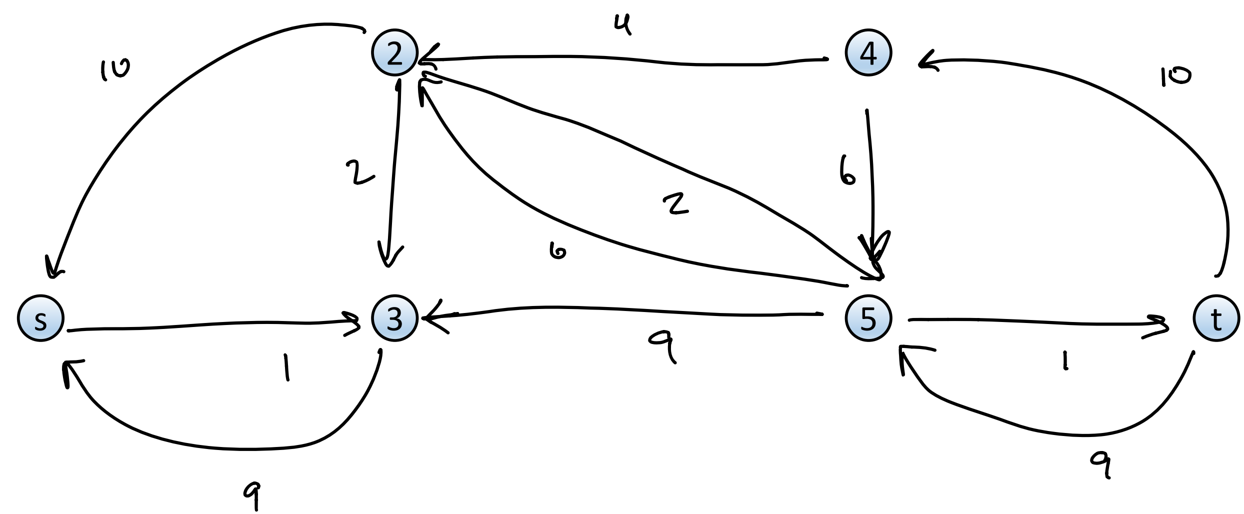
$A = \{s, 3\}$

$cap(A, B) = 19$

G:



G_f:



What do we want to prove?

Running Time of Ford-Fulkerson

Assume for now

- 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!

Correctness 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

weak duality

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

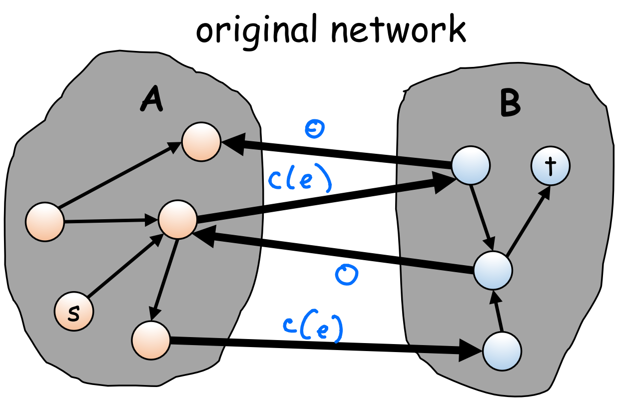
Optimality of Ford-Fulkerson

- **(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
  - Note $s \in A$ $t \in B$ because there is no augmenting path

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 $f(e) = 0$



$$val(f) = \sum_{e:A\to B} f(e) - \sum_{e:B\to A} f(e)$$

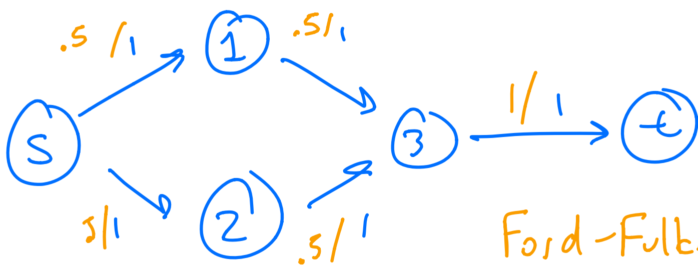
$$= \sum_{e:A\to B} f(e)$$

$$= \sum_{e:A\to B} c(e) = cap(A, B)$$

Summary

- **The Ford-Fulkerson Algorithm solves maximum s-t flow**
  - Running time $O\big(m \cdot val(f^*)\big)$ in networks with integer capacities

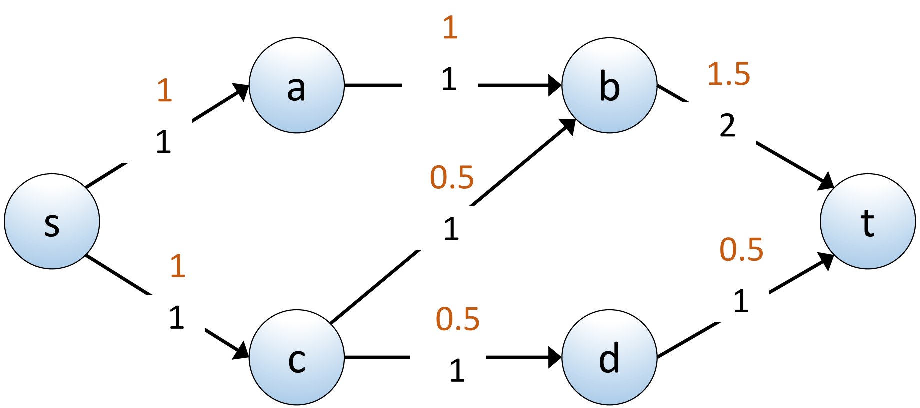
- **Strong MaxFlow-MinCut Duality: max flow = min cut**
  - The value of the maximum s-t flow equals the capacity of the minimum s-t cut
  - If f^* is a maximum s-t flow, then the set of nodes reachable from s in G_{f^*} gives a minimum cut
  - Given a max-flow, can find a min-cut in time $O(n+m)$



Ford-Fulkerson wouldn't find this flow.

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 maximum s-t flow equals the capacity of the minimum 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