Ford-Fulkerson Analysis

FORD–FULKERSON(G)

FOREACH edge $e \in E$: $f(e) \leftarrow 0$.

$G_f \leftarrow$ residual network of G with respect to flow f.

WHILE (there exists an s→t path P in G_f)

  $f \leftarrow$ AUGMENT(f, P).

  Update G_f.

RETURN f.

AUGMENT(f, P)

$b \leftarrow$ bottleneck capacity of augmenting path P.

FOREACH edge $e \in P$:

  IF ($e \in E$, *that is, e is forward edge*)

    Increase f(e) in G by b

  ELSE

    Decrease f(e) in G by b

RETURN f.

Lecture Outline

- **Correctness and Value of Flow**:
  - Each iteration of the Ford-Fulkerson algorithm sends a feasible flow through the network
  - With each iteration of the Ford-Fulkerson algorithm the value of the flow increases by $b \leftarrow$ bottleneck capacity of the augmenting path P
- **Optimality**:
  - Ford-Fulkerson algorithm computes the maximum flow f
  - Prove by constructing a s-t cut such that $c(s, t) = v(f)$
- **Running time**:
  - How long does the Ford-Fulkerson algorithm take to compute the max flow?

Correctness & Value of Flow

Augmenting Path & Flow

- **Claim**. Let f be a feasible flow in G and let P be an augmenting path in G_f with bottleneck capacity b. Let $f' \leftarrow$ AUGMENT(f, P), then f' is **a feasible flow** and $v(f') = v(f) + b$.

- **Proof**. Only need to verify constraints on the edges of P (since $f' = f$ for other edges). Let $e = (u, v) \in P$

    - If e is a forward edge:
    $$\begin{aligned} f(e) &\leq f'(e) \\ &\leq f(e) + b \\ &\leq f(e) + (c_e - f(e)) = c_e \end{aligned}$$

    - If e is a backward edge:

    - $$\begin{aligned} f(e) &\geq f'(e) = f(e) - b \\ &\geq f(e) - f(e) = 0 \end{aligned}$$

- Conservation constraint hold on nodes in P (exercise)

Augmenting Path & Flow

- **Claim**. Let f be a feasible flow in G and let P be an augmenting path in G_f with bottleneck capacity b. Let $f' \leftarrow$ AUGMENT(f, P), then f' is a feasible flow and $v(f') = v(f) + b$.
- **Proof**.
  - First edge $e \in P$ must be out of s in G_f
  - P is simple so never visits s again
  - e must be a forward edge (P is a path from s to t)
  - Thus $f(e)$ increases by b, increasing $v(f)$ by b

    ∎

Optimality

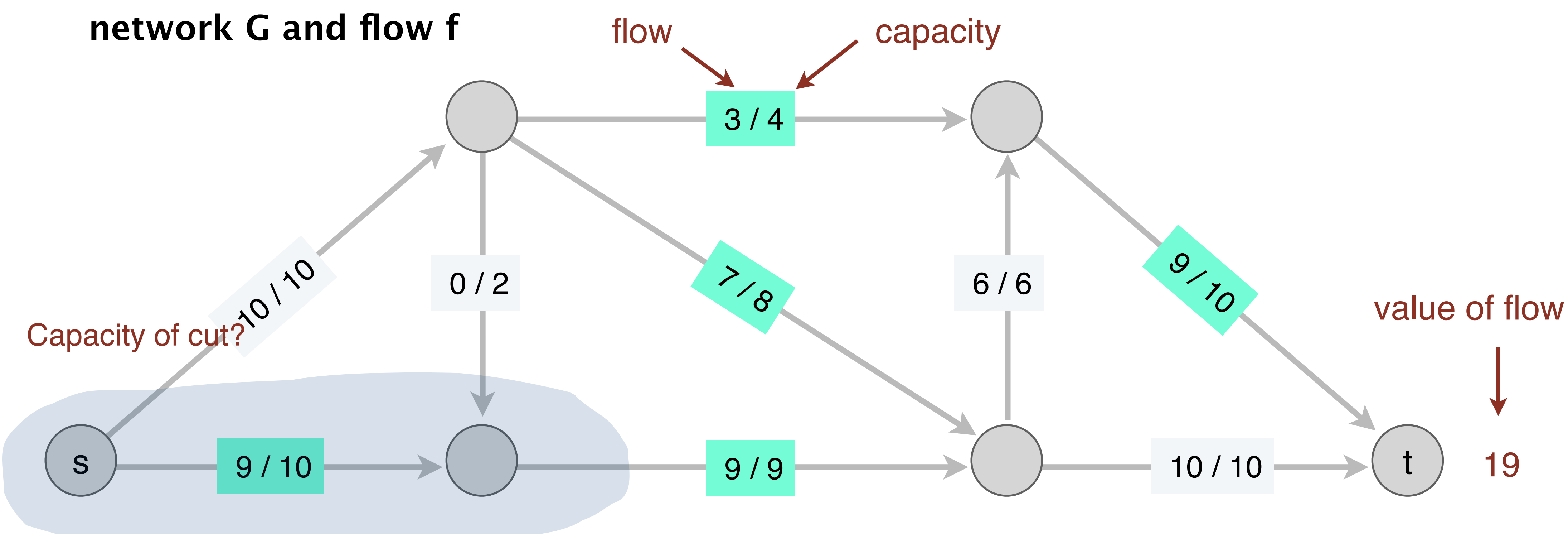
Ford-Fulkerson Optimality

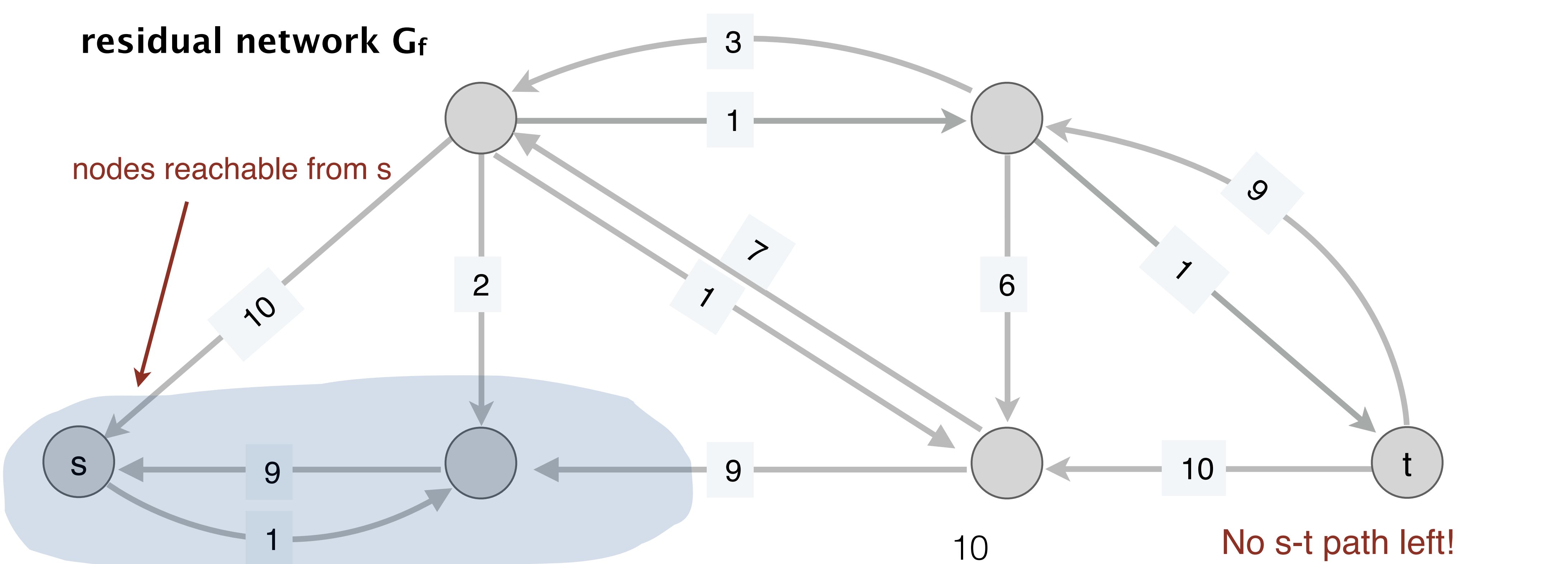
- **Recall**: If f is any feasible s-t flow and (S, T) is any s-t cut then $v(f) \leq c(S, T)$.
- We will show that the Ford-Fulkerson algorithm terminates in a flow that achieves equality, that is,
- Ford-Fulkerson finds a flow f^* and there exists a cut (S^*, T^*) such that
$$v(f^*) = c(S^*, T^*)$$
- Proving this shows that it finds the maximum flow!
- This also **proves the max-flow min-cut theorem**

Ford-Fulkerson Optimality

- **Lemma**. Let f be a s-t flow in G such that there is no augmenting path in the residual graph G_f, then there exists a cut (S^*, T^*) such that $v(f) = c(S^*, T^*)$.
- **Proof**.
- Let $S^* = \{v \mid v$ is reachable from s in $G_f\}$,
  $T^* = V - S^*$
- Is this an s-t cut?
  - $s \in S, t \in T$, $S \cup T = V$ and $S \cap T = \emptyset$
- Consider an edge $e = u \to v$ with $u \in S^*, v \in T^*$, then what can we say about $f(e)$?

Recall: Ford-Fulkerson Example





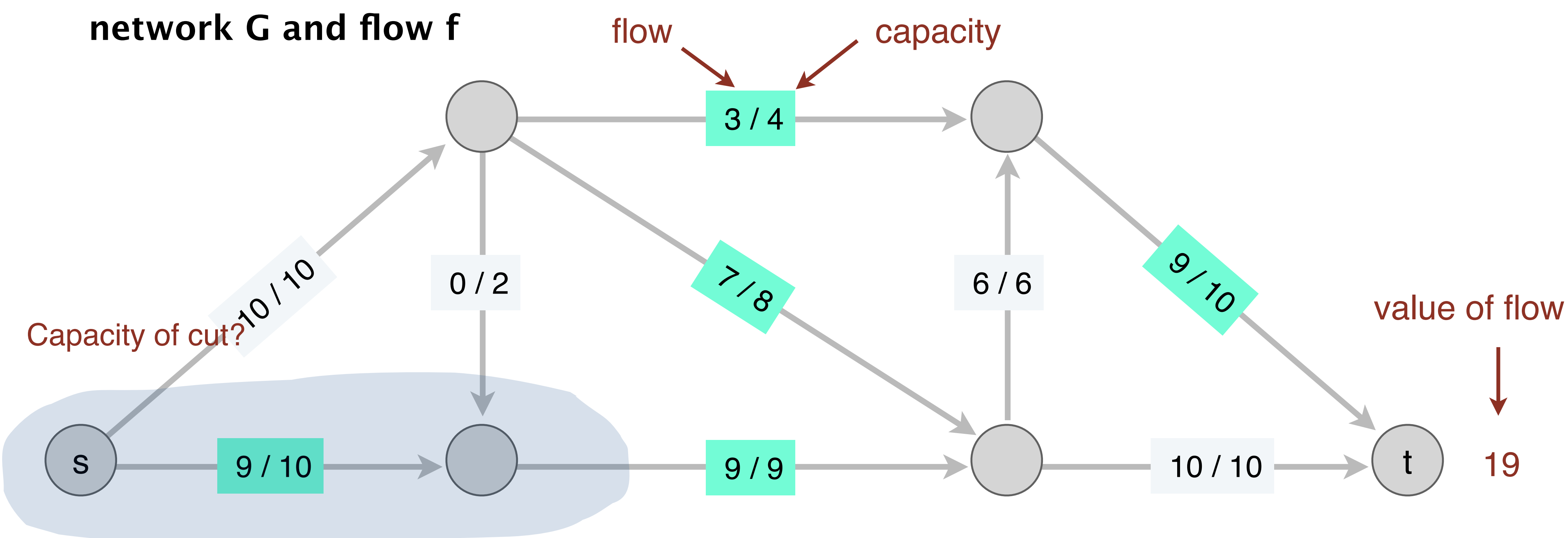
Ford-Fulkerson Optimality

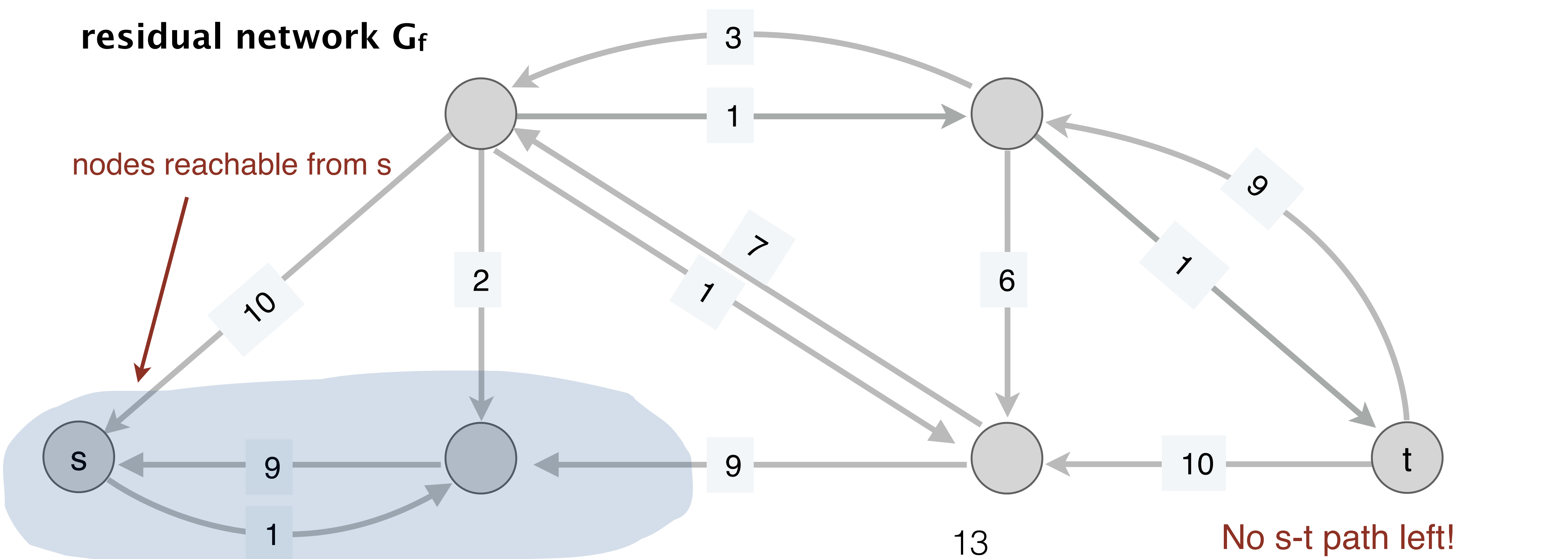
- **Lemma**. Let f be a s-t flow in G such that there is no augmenting path in the residual graph G_f, then there exists a cut (S^*, T^*) such that $v(f) = c(S^*, T^*)$.
- **Proof**.
- Let $S^* = \{v \mid v \text{ is reachable from } s \text{ in } G_f\}$,
  $T^* = V - S^*$
- Is this an s-t cut?
  - $s \in S, t \in T$, $S \cup T = V$ and $S \cap T = \varnothing$
- Consider an edge $e = u \to v$ with $u \in S^*, v \in T^*$, then what can we say about $f(e)$?
  - $f(e) = c(e)$

Ford-Fulkerson Optimality

- **Lemma**. Let f be a s-t flow in G such that there is no augmenting path in the residual graph G_f, then there exists a cut (S^*, T^*) such that $v(f) = c(S^*, T^*)$.
- **Proof**. **(Cont.)**
- Let $S^* = \{v \mid v$ is reachable from s in $G_f\}$,
  $T^* = V - S^*$
- Is this an s-t cut?
  - $s \in S, t \in T$, $S \cup T = V$ and $S \cap T = \emptyset$
- Consider an edge $e = w \to v$ with
  $v \in S^*, w \in T^*$, then what can we say about $f(e)$?

Recall: Ford-Fulkerson Example





Ford-Fulkerson Optimality

- **Lemma**. Let f be a s-t flow in G such that there is no augmenting path in the residual graph G_f, then there exists a cut (S^*, T^*) such that $v(f) = c(S^*, T^*)$.
- **Proof**. **(Cont.)**
- Let $S^* = \{v \mid v$ is reachable from s in $G_f\}$,
  $T^* = V - S^*$
- Is this an s-t cut?
  - $s \in S, t \in T$, $S \cup T = V$ and $S \cap T = \varnothing$
- Consider an edge $e = w \to v$ with
  $v \in S^*, w \in T^*$, then what can we say about $f(e)$?
  - $f(e) = 0$

Ford-Fulkerson Optimality

- **Lemma**. Let f be a s-t flow in G such that there is no augmenting path in the residual graph G_f, then there exists a cut (S^*, T^*) such that $v(f) = c(S^*, T^*)$.
- **Proof**. **(Cont.)**
- Let $S^* = \{v \mid v \text{ is reachable from } s \text{ in } G_f\}$, $T^* = V - S^*$
- Thus, all edges leaving S^* are completely saturated and all edges entering S^* have zero flow
- $v(f) = f_{out}(S^*) - f_{in}(S^*) = f_{out}(S^*) = c(S^*, T^*)$ ∎
- **Corollary**. Ford-Fulkerson returns the maximum flow.

Ford-Fulkerson Algorithm
Running Time

Ford-Fulkerson Performance

FORD–FULKERSON(G)

FOREACH edge $e \in E$: $f(e) \leftarrow 0$.

$G_f \leftarrow$ residual network of G with respect to flow f.

WHILE (there exists an s↝t path P in G_f)

  $f \leftarrow$ AUGMENT(f, P).

  Update G_f.

RETURN f.

- Does the algorithm terminate?
- Can we bound the number of iterations it does?
- Running time?

Ford-Fulkerson Running Time

- Recall we proved that with each call to AUGMENT, we increase **value of flow** by $b = \text{bottleneck}(G_f, P)$

- **Assumption**. Suppose all capacities $c(e)$ are integers.

- **Integrality invariant.** Throughout Ford–Fulkerson, every edge flow $f(e)$ and corresponding residual capacity is an integer. Thus $b \geq 1$.

- Let $C = \max_u c(s \to u)$ be the maximum capacity among edges leaving the source s.

- It must be that $v(f) \leq (n-1)C = O(nC)$

- Since, $v(f)$ increases by $b \geq 1$ in each iteration, it follows that FF algorithm terminates in at most $v(f) = O(nC)$ iterations.

Ford-Fulkerson Running Time

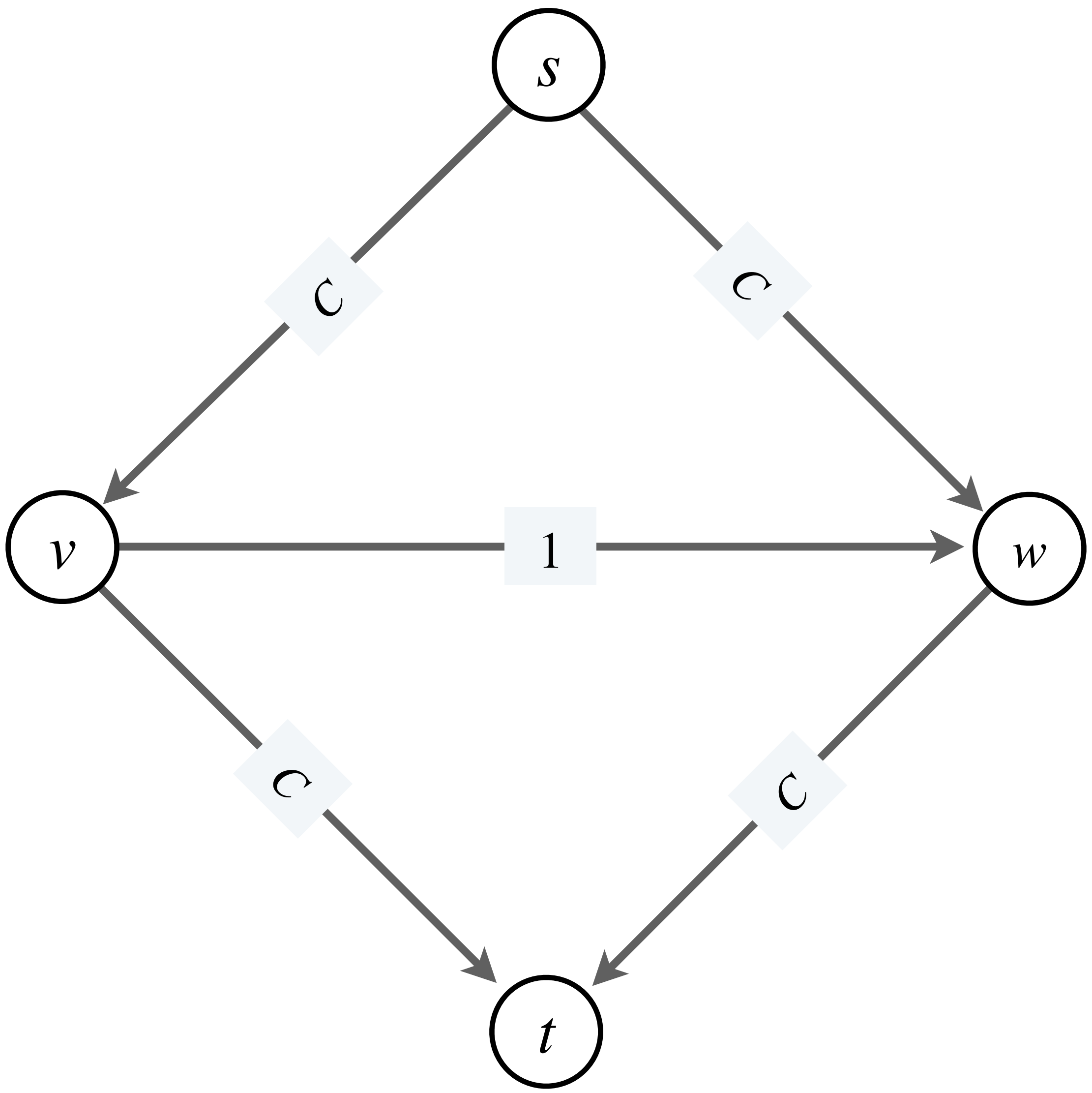
- **Claim.** Ford-Fulkerson can be implemented to run in time $O(nmC)$, where $m = |E| \geq n - 1$ and $C = \max_u c(s \to u)$.
- **Proof**. We know algorithm terminates in at most C iterations. Each iteration takes $O(m)$ time:
  - We need to find an augmenting path in G_f
  - G_f has at most $2m$ edges, using BFS/DFS takes $O(m + n) = O(m)$ time
  - Augmenting flow in P takes $O(n)$ time
  - Given new flow, we can build new residual graph in $O(m)$ time ∎

[Digging Deeper] Polynomial time?

- Does the Ford-Fulkerson algorithm run in time polynomial in the input size?
- Running time is $O(nmC)$, where $C = \max_u c(s \rightarrow u)$, suppose it is even larger, that is, $C = \max_e c(e)$
- What is the input size?
- Let’s take an example

[Digging Deeper] Polynomial time?

- **Question**. Does the Ford-Fulkerson algorithm run in polynomial-time in the size of the input? ← ~ m, n, and $\log C$
- **Answer**. No. if max capacity is C, the algorithm can take $\geq C$ iterations. Consider the following example.



- $s \to v \to w \to t$
- $s \to w \to v \to t$
- $s \to v \to w \to t$
- $s \to w \to v \to t$
- …
- $s \to v \to w \to t$
- $s \to w \to v \to t$

← each augmenting path sends only 1 unit of flow (# augmenting paths = $2C$)

[Digger Deeper] Pseudo-Polynomial

- Input graph has n nodes and $m = O(n^2)$ edges, each with capacity c_e
- $C = \max_{e \in E} c(e)$, then $c(e)$ takes $O(\log C)$ bits to represent
- Input size: $O(n \log n + m \log n + m \log C)$ bits
- Let $t = \log n$, $b = \log C$
- Input size: $O(nv + m(v + b))$
- Running time: $O(nm2^b)$, exponential in the size of C
- Such algorithms are called **pseudo-polynomial**
    - If the running time is polynomial in the **magnitude** but **not size** of an input parameter.

Summary

- Given a flow network with integer capacities, Ford-Fulkerson computes the **max flow** in $O(mnC)$ time
- A **constructive proof** of the max-flow min-cut theorem
- It is a pseudo-polynomial algorithm
    - Can take exponential time wrt to size of C
    - Bad performance in the worst case can be blamed on poor augmenting path choices
- **Next. (Flow Applications)** Solving other optimization problems by reduction them to a network flow problem

Network Flow [Optional]: Beyond Ford Fulkerson

Edmond and Karp's Algorithms

- Ford and Fulkerson's algorithm does not specify which path in the residual graph to augment
- Poor worst-case behavior of the algorithm can be blamed on bad choices on augmenting path
- **Better choice of augmenting paths.** In 1970s, Jack Edmonds and Richard Karp published two natural rules for choosing augmenting paths
  - Fattest augmenting paths first
  - Shortest (in terms of edges) augmenting paths first (Dinitz independently discovered & analyzed this rule)

Fattest Augmenting Paths First

- Ford Fulkerson is essentially a greedy algorithm way of augmenting paths:
  - Choose the augmenting path with largest bottleneck capacity
- Largest bottleneck path can be computed in $O(m \log n)$ time in a directed graph
  - Similar to Dijkstra's analysis
- How many iterations if we use this rule?
  - Won't prove this: takes $O(m \log C)$ iterations
- Overall running time is $O(m^2 \log n \log C)$ (polynomial time!)

Shortest Augmenting Paths First

- Choose the augmenting path with the smallest # of edges
- Can be found using BFS on G_f in $O(m+n) = O(m)$ time
- Surprisingly, this resulting a polynomial-time algorithm independent of the actual edge capacities !
- Analysis looks at “level” of vertices in the BFS tree of G_f rooted at s —levels only grow over time
- Analyzes # of times an edge $u \to v$ disappears from G_f
- Takes $O(mn)$ iterations overall
- Thus overall running time is $O(m^2 n)$

Progress on Network Flows

1951	$O(m\, n^2\, C)$	Dantzig
1955	$O(m\, n\, C)$	Ford–Fulkerson
1970	$O(m\, n^2)$	Edmonds–Karp, Dinitz
1974	$O(n^3)$	Karzanov
1983	$O(m\, n \log n)$	Sleator–Tarjan
1985	$O(m\, n \log C)$	Gabow
1988	$O(m\, n \log (n^2 / m))$	Goldberg–Tarjan
1998	$O(m^{3/2} \log (n^2 / m) \log C)$	Goldberg–Rao
2013	$O(m\, n)$	Orlin
2014	$\tilde{O}(m\, n^{1/2} \log C)$	Lee–Sidford
2016	$\tilde{O}(m^{10/7}\, C^{1/7})$	Mądry

For unit capacity networks

Summary

- Given a flow network with integer capacities, the **maximum flow and minimum cut** can be computed in $O(mn)$ time.
- **Next.** Network flow applications!