

# CSE 421

## Ford-Fulkerson Algorithm

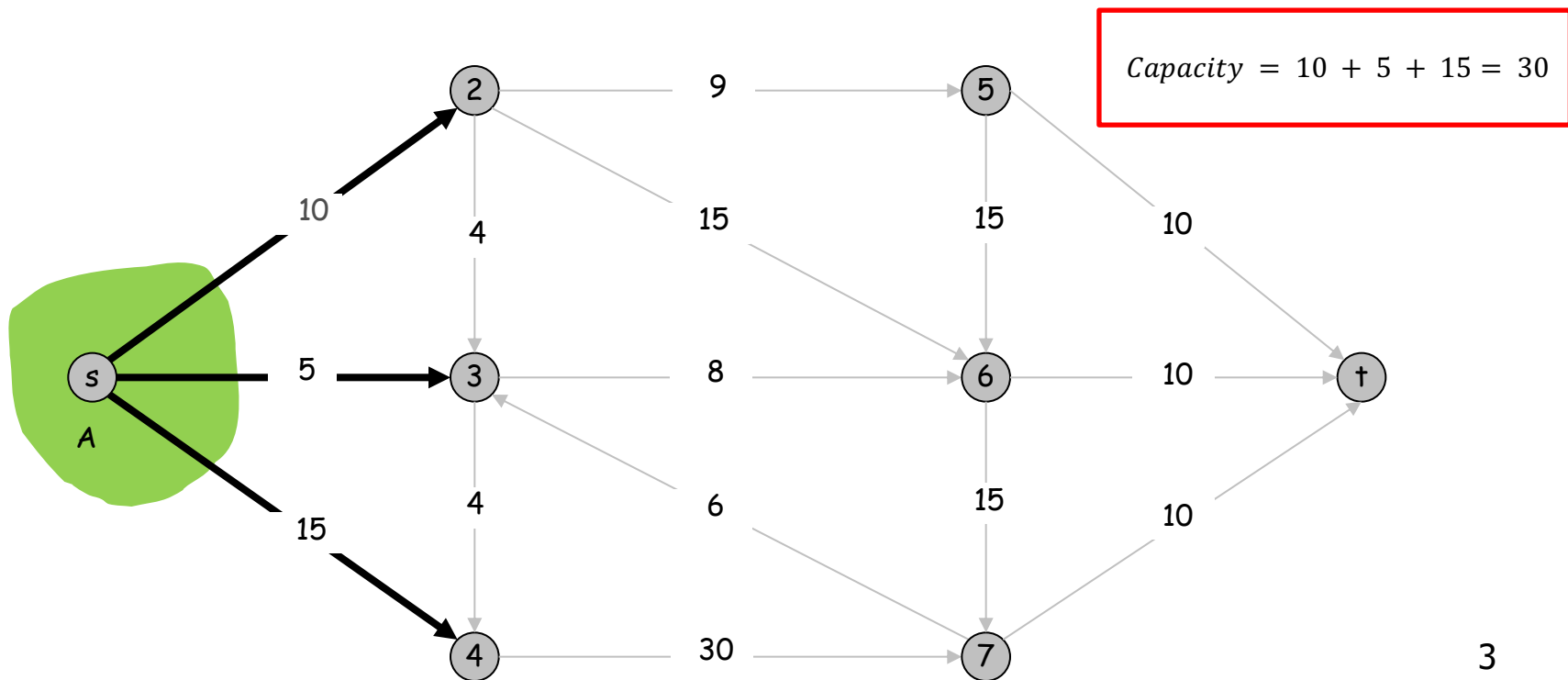
Shayan Oveis Gharan

# Network Flows

# s-t cuts

Def. An **s-t cut** is a partition  $(A, B)$  of  $V$  with  $s \in A$  and  $t \in B$ .

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

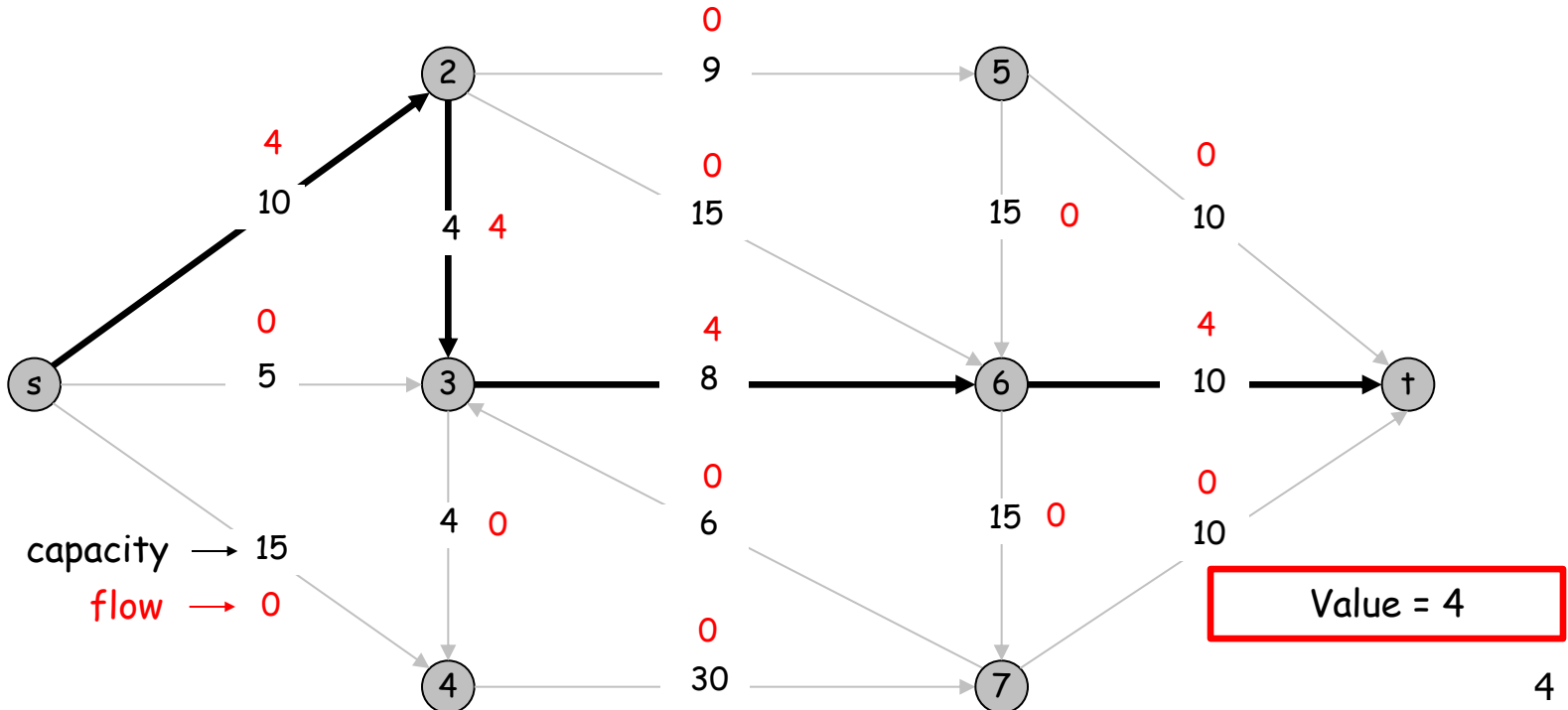


# s-t Flows

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

- For each  $e \in E$ :  $0 \leq f(e) \leq c(e)$  (capacity)
- For each  $v \in V - \{s, t\}$ :  $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$  (conservation)

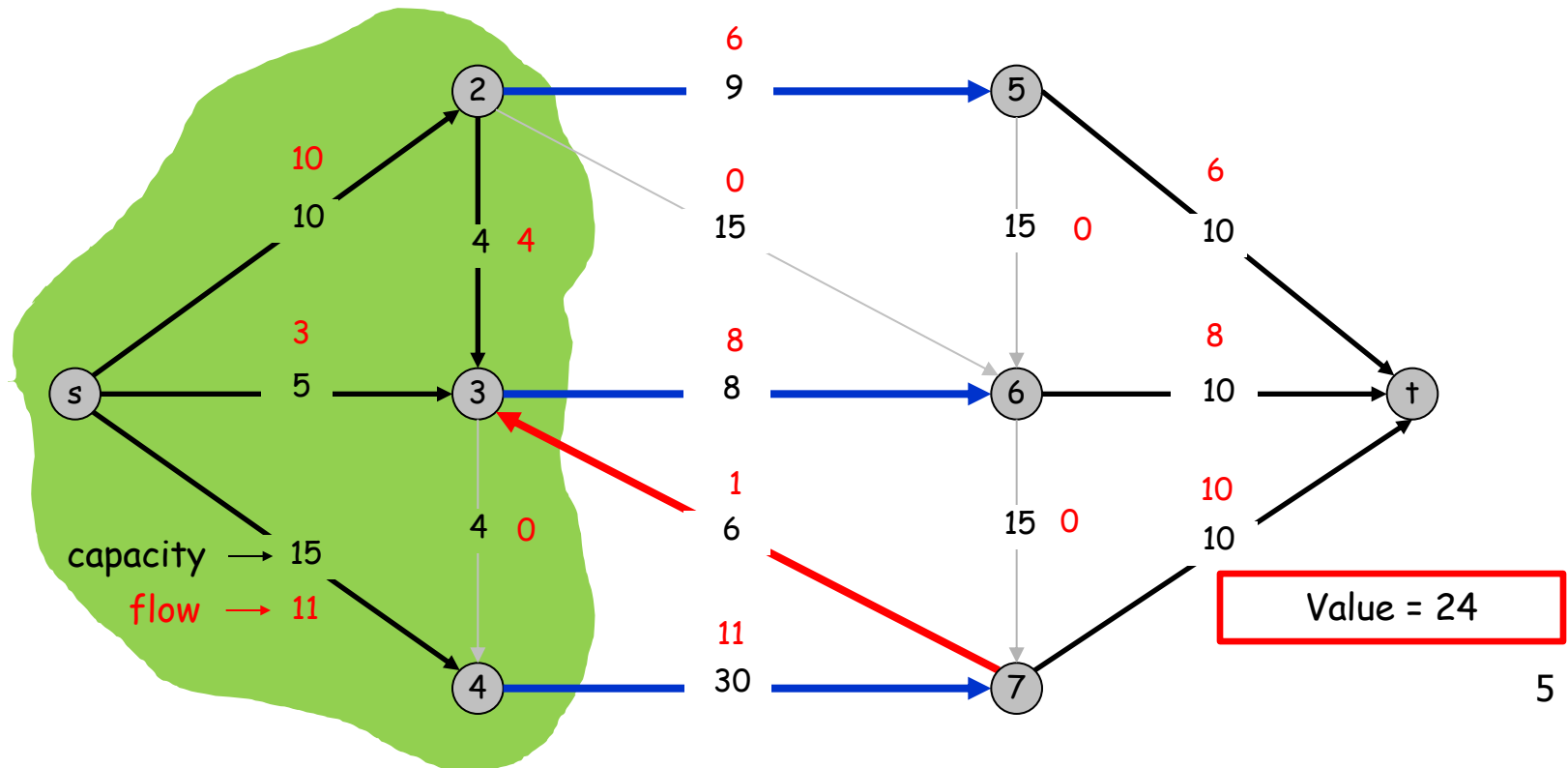
Def. The **value** of a flow  $f$  is:  $v(f) = \sum_{e \text{ out of } s} f(e)$



# 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  $s$ .

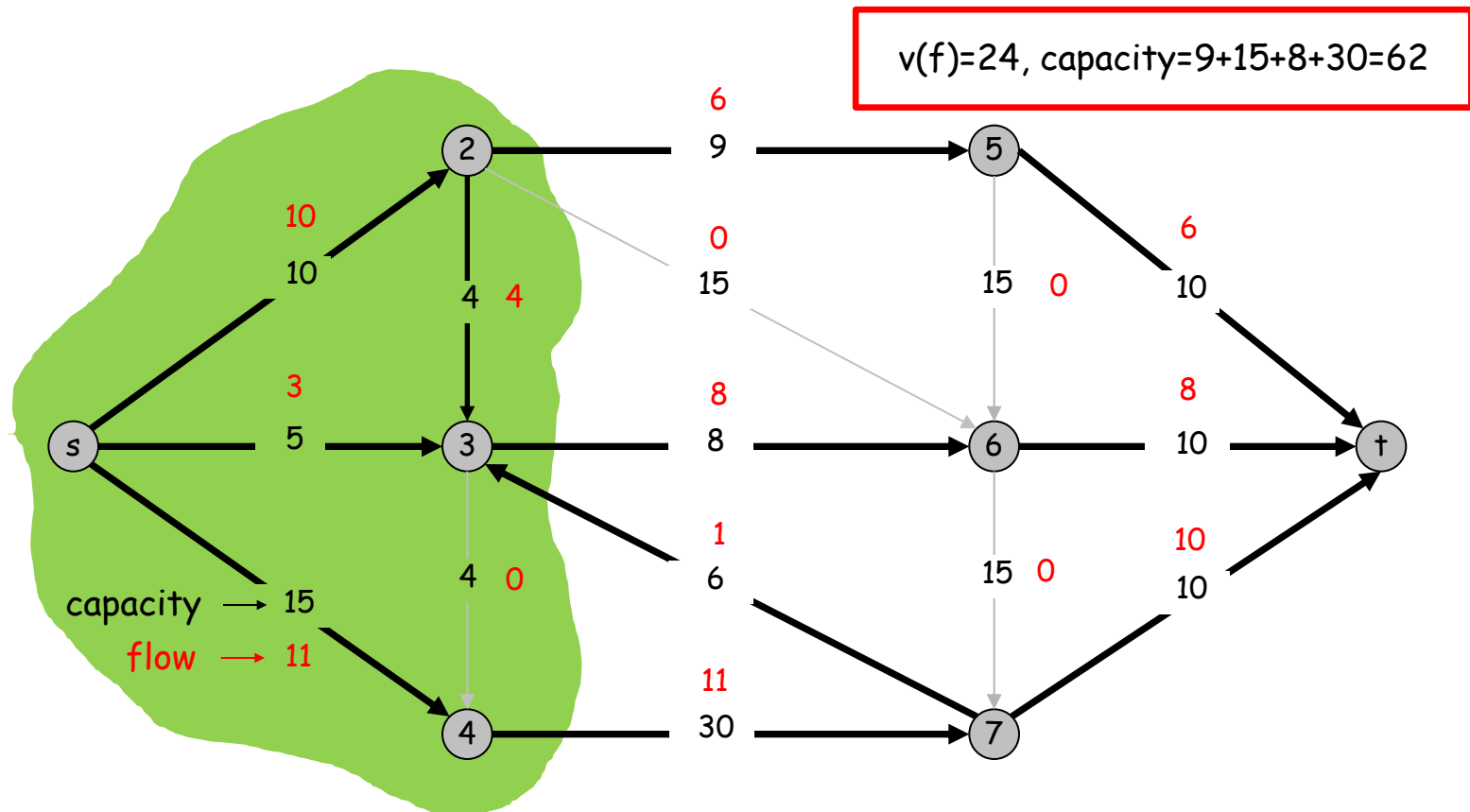
$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$



# Weak Duality of Flows and Cuts

**Weak Duality lemma.** 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.

$$v(f) \leq \text{cap}(A, B)$$

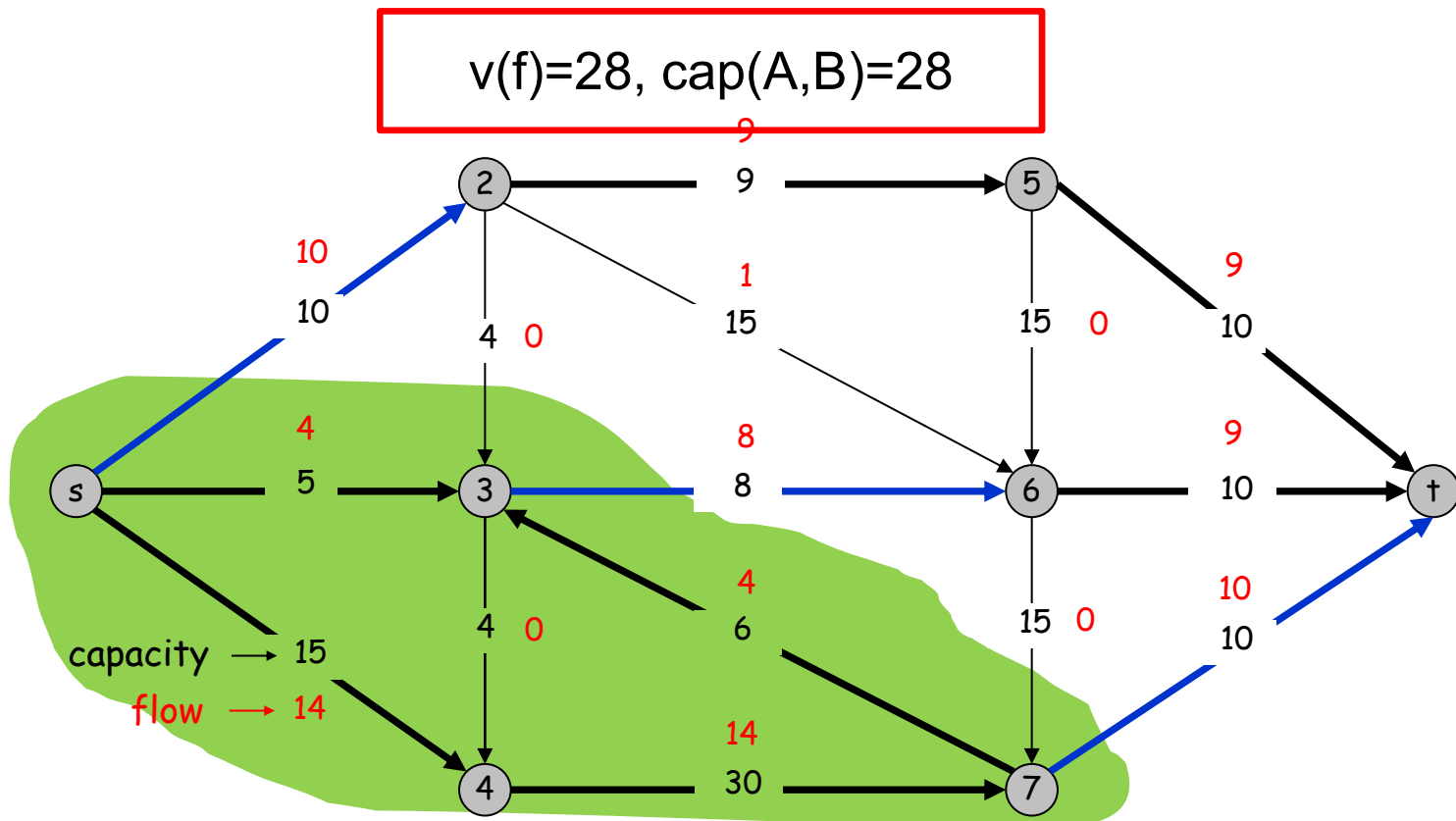


# Certificate of Optimality

**Corollary:** Suppose there is a s-t cut (A,B) such that

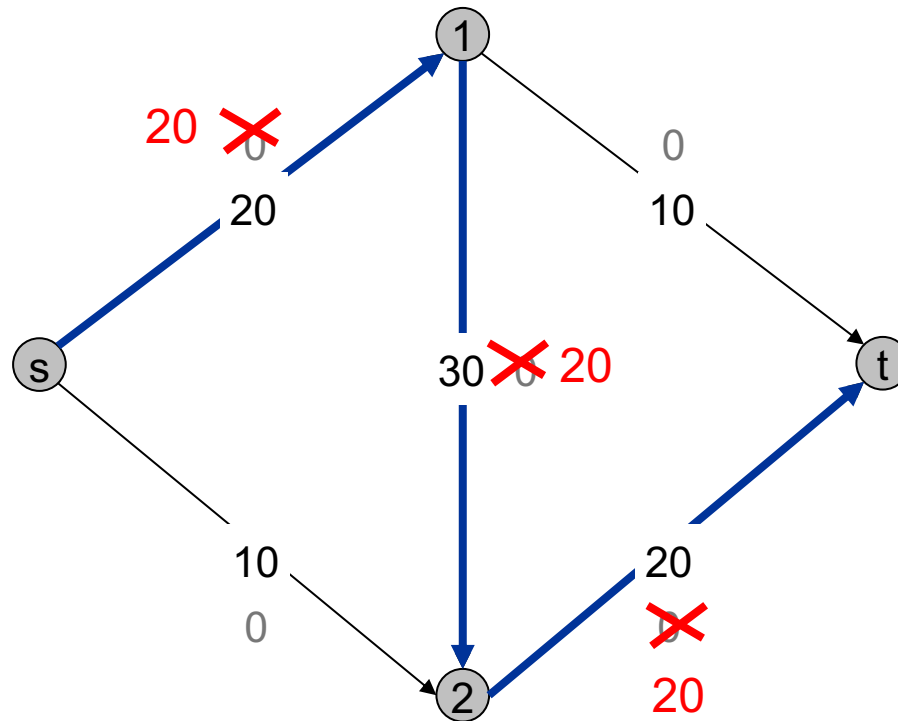
$$v(f) = \text{cap}(A, B)$$

Then,  $f$  is a maximum flow and (A,B) is a minimum cut.



# A Greedy Algorithm for Max Flow

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

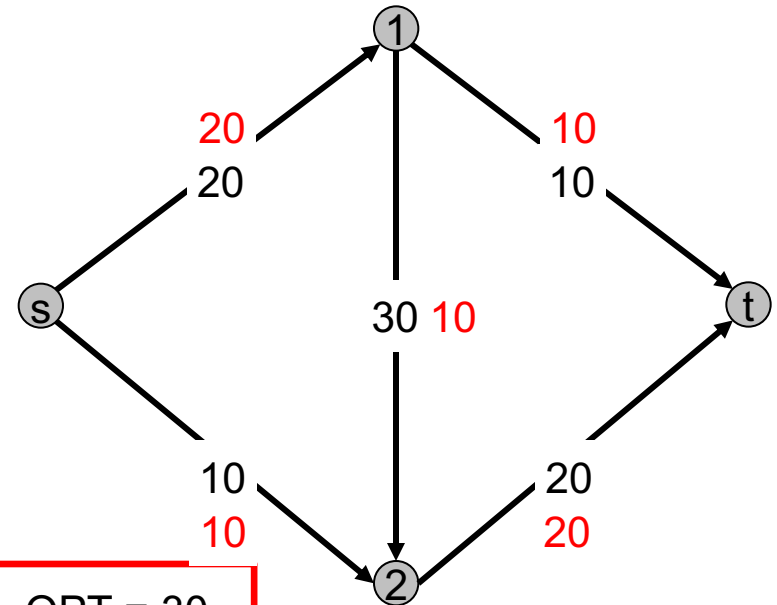
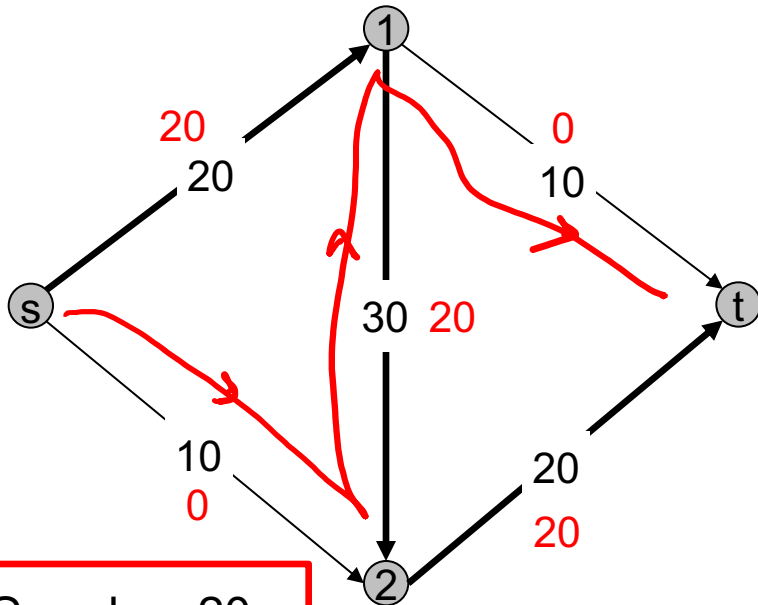




# A Greedy Algorithm for Max Flow

- Start with  $f(e) = 0$  for all edge  $e \in E$ .
- Find an  $s$ - $t$  path  $P$  where each edge has  $f(e) < c(e)$ .
- Augment flow along path  $P$ .
- Repeat until you get **stuck**.

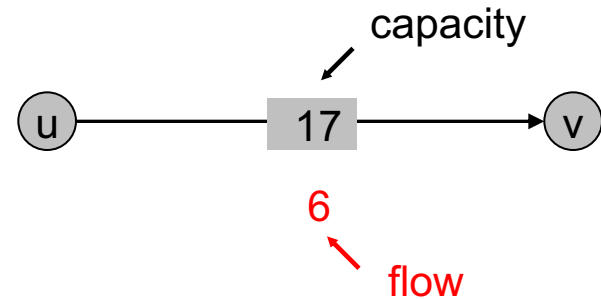
Local Optimum  $\neq$  Global Optimum



# Residual Graph

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

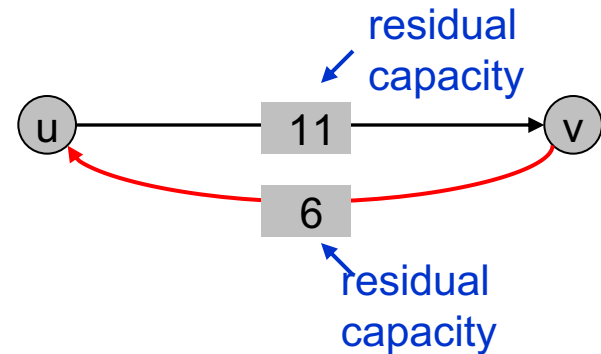
- Flow  $f(e)$ , capacity  $c(e)$ .



Residual edge.

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

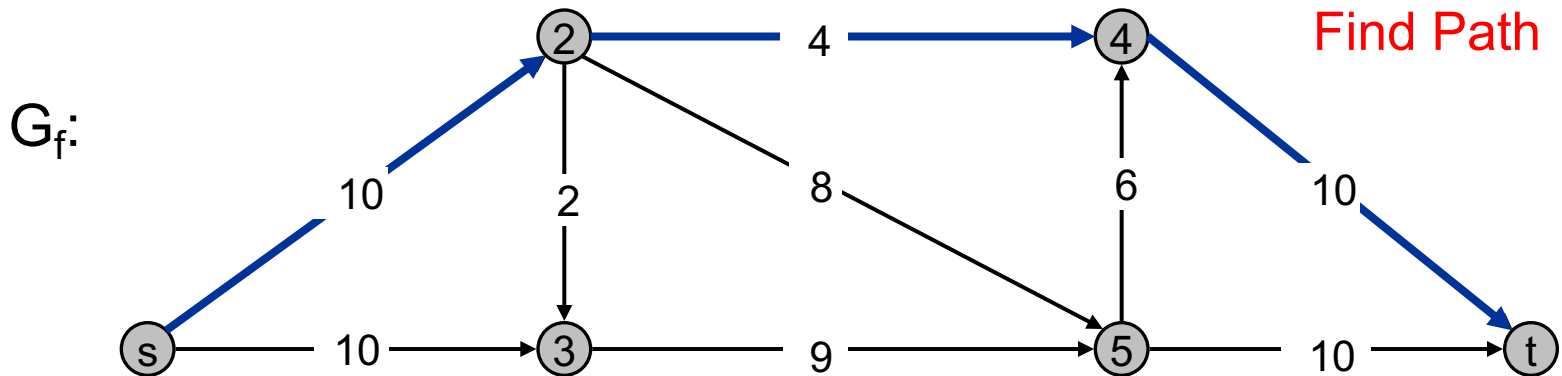
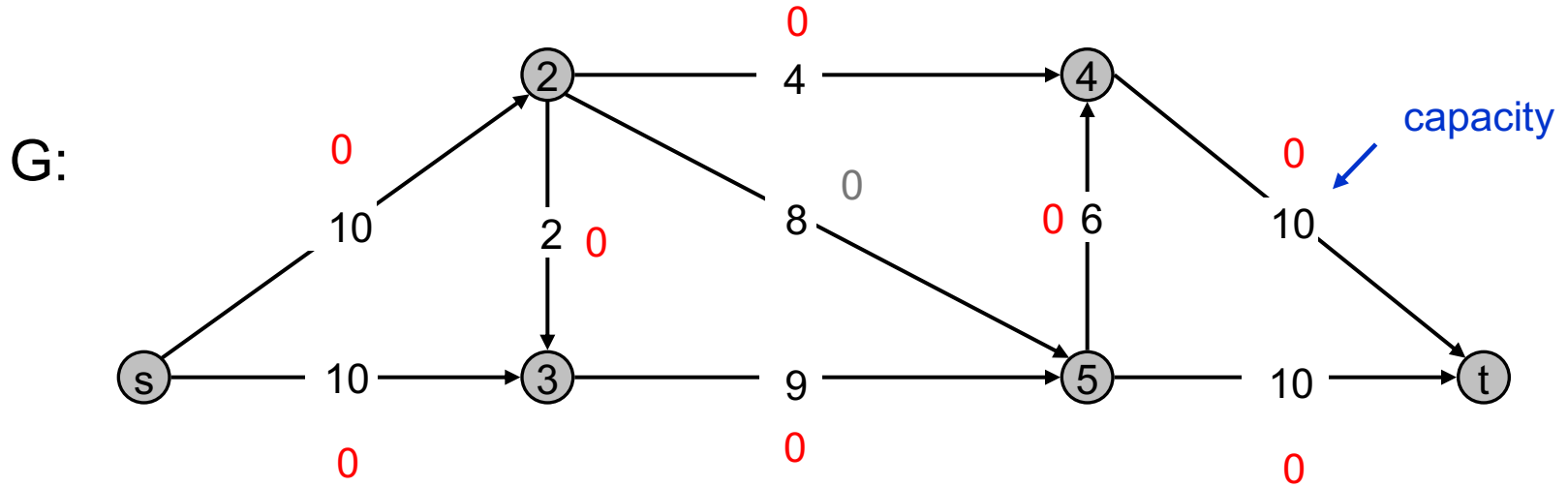
$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^R \in E \end{cases}$$



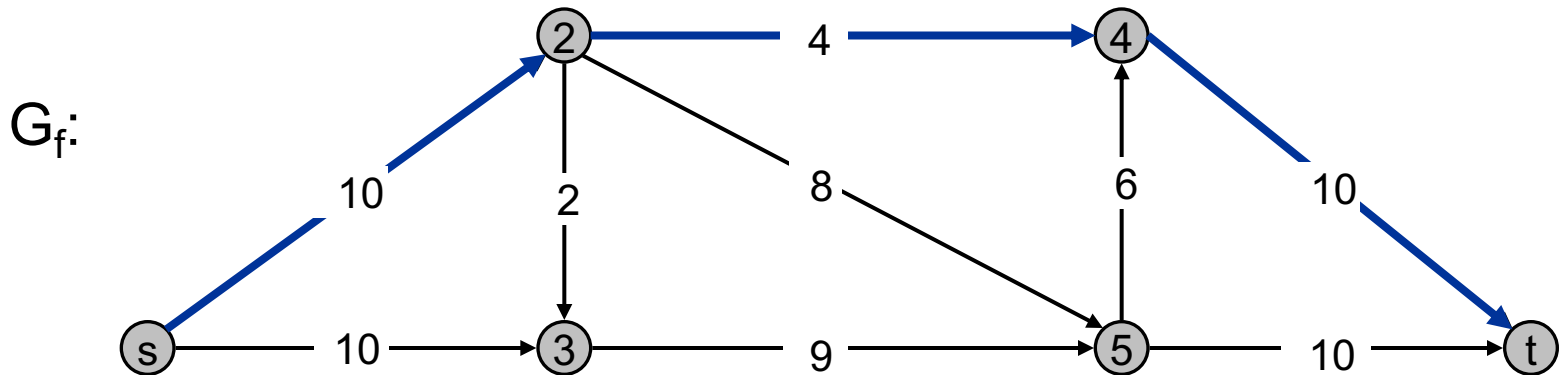
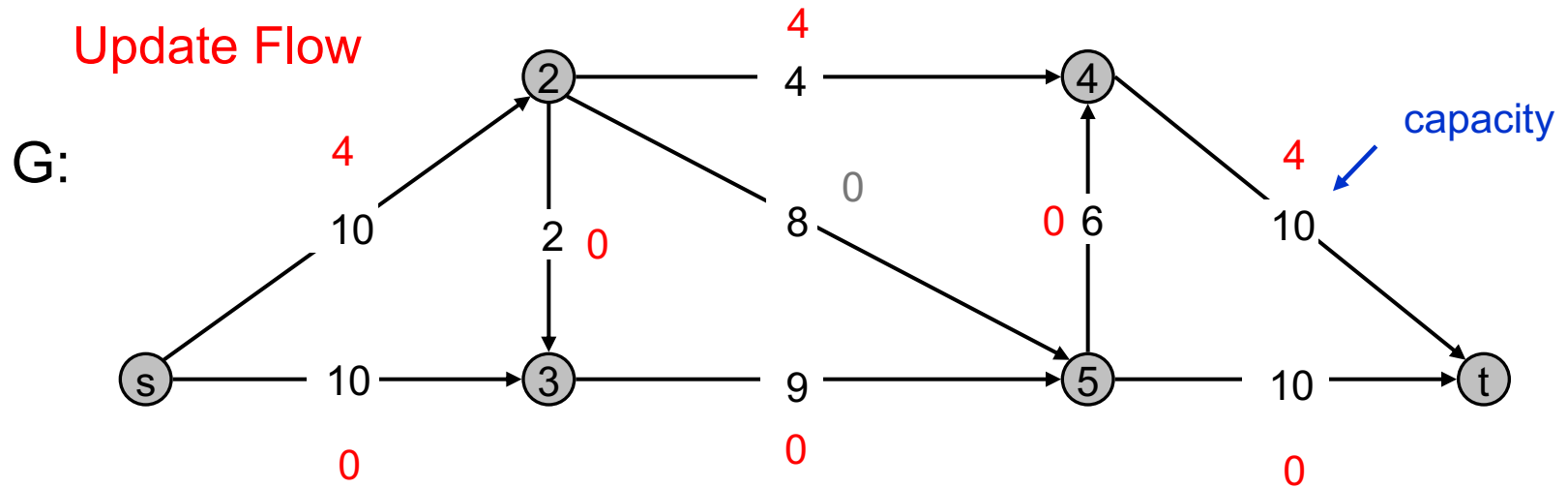
Residual graph:  $G_f = (V, E_f)$ .

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

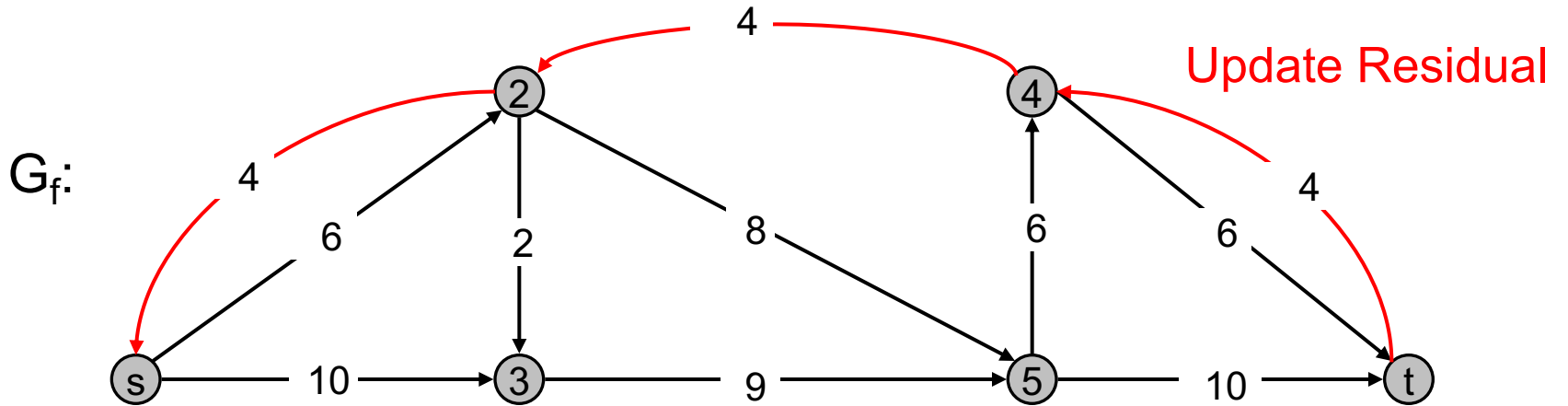
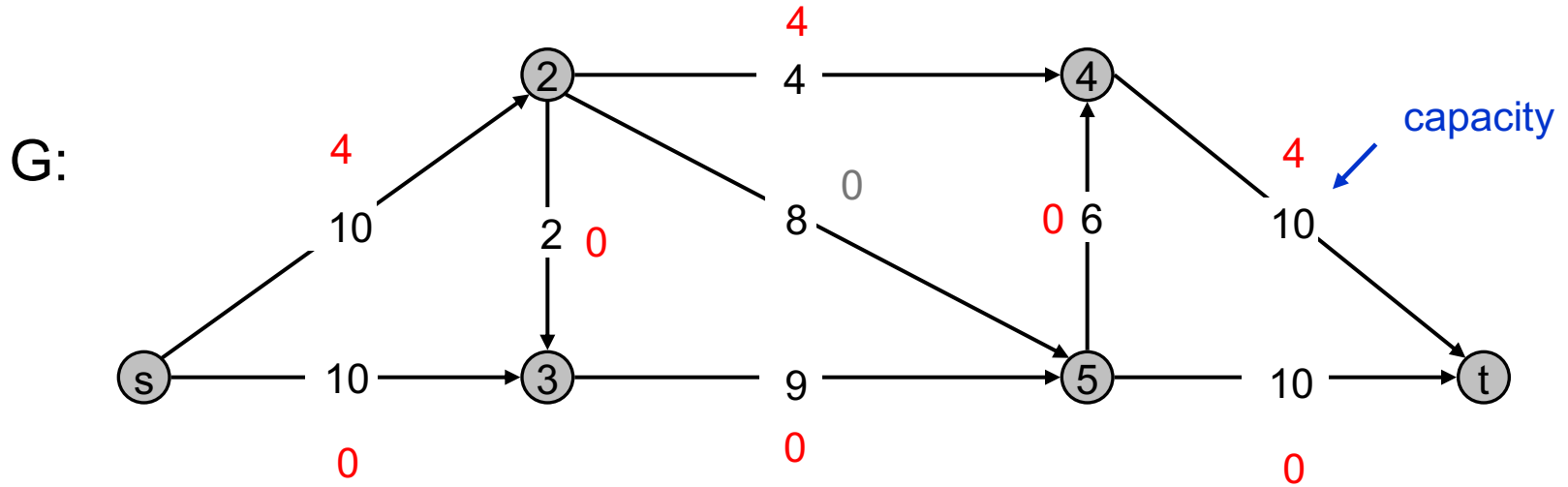
# Ford-Fulkerson Alg: Greedy on $G_f$



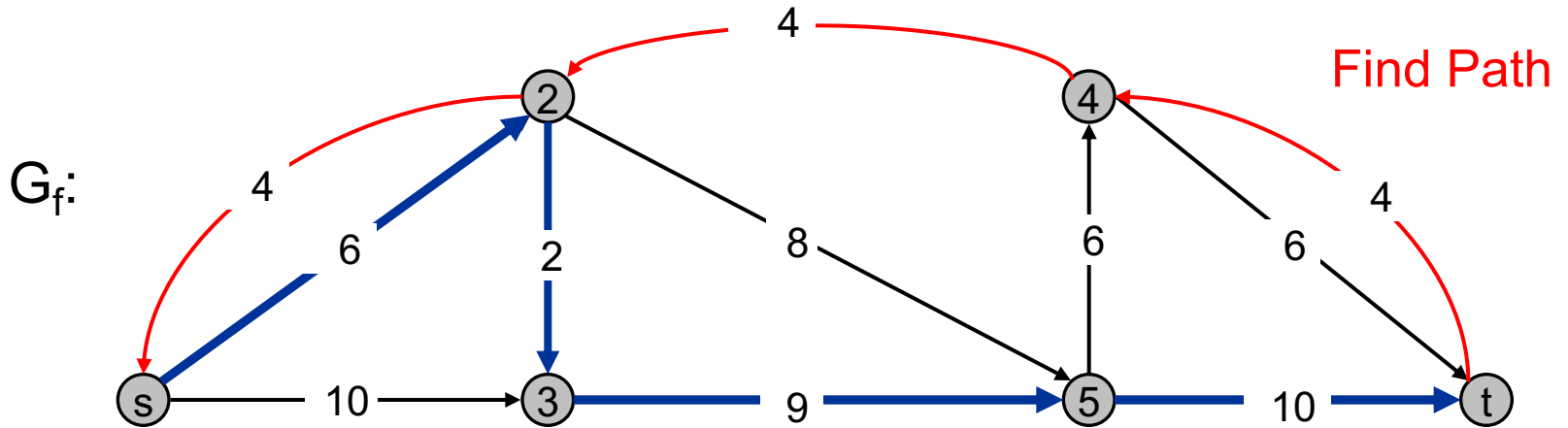
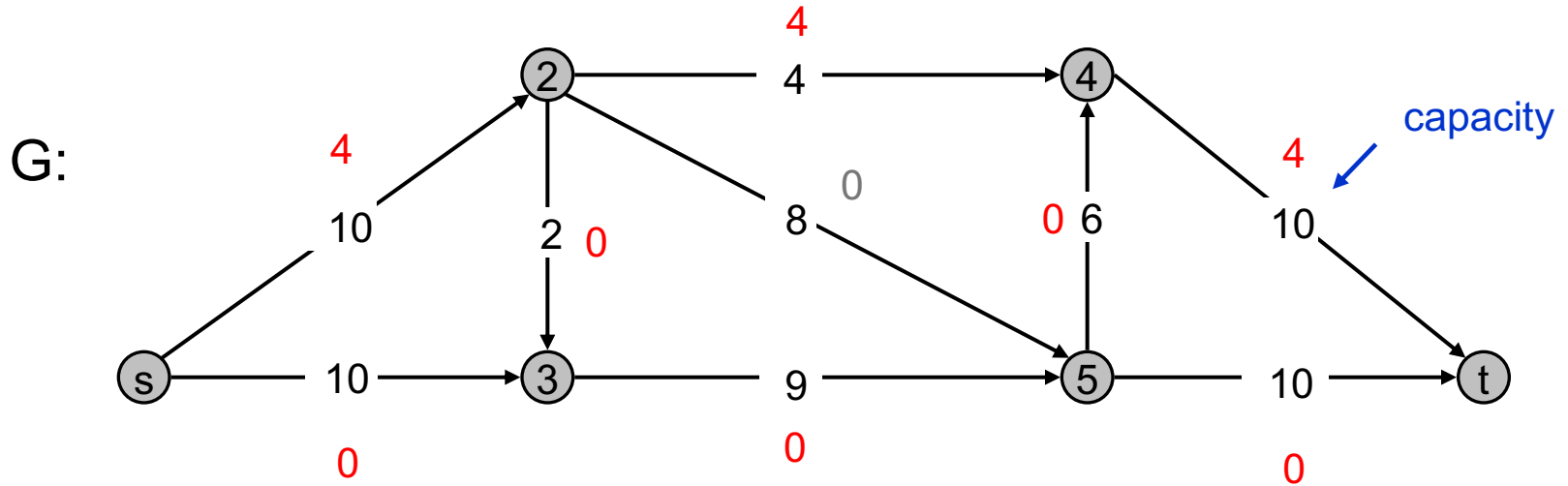
# Ford-Fulkerson Alg: Greedy on $G_f$



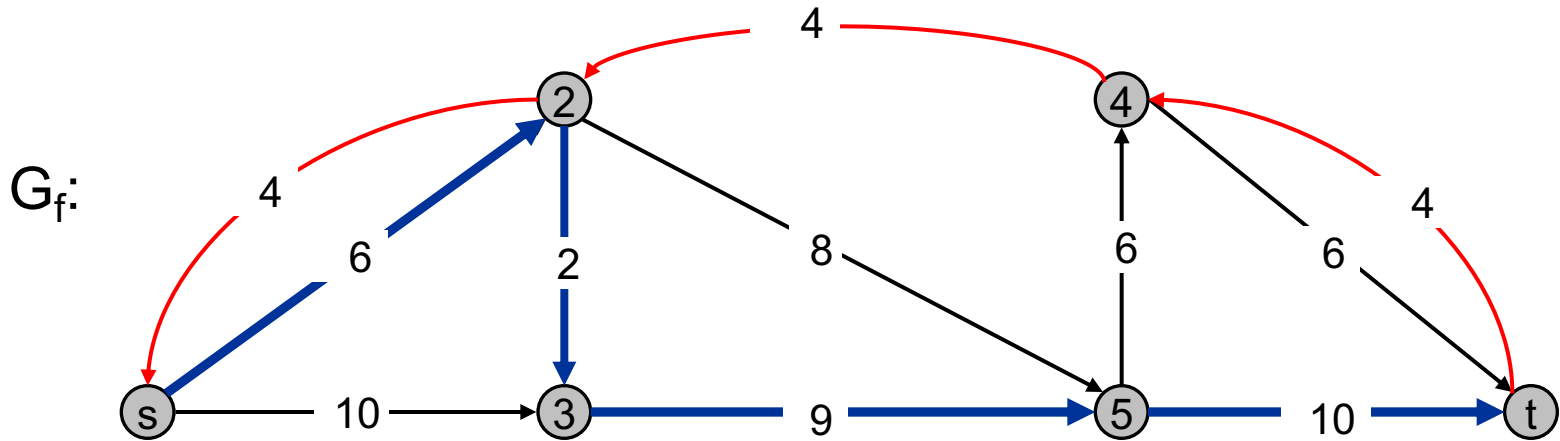
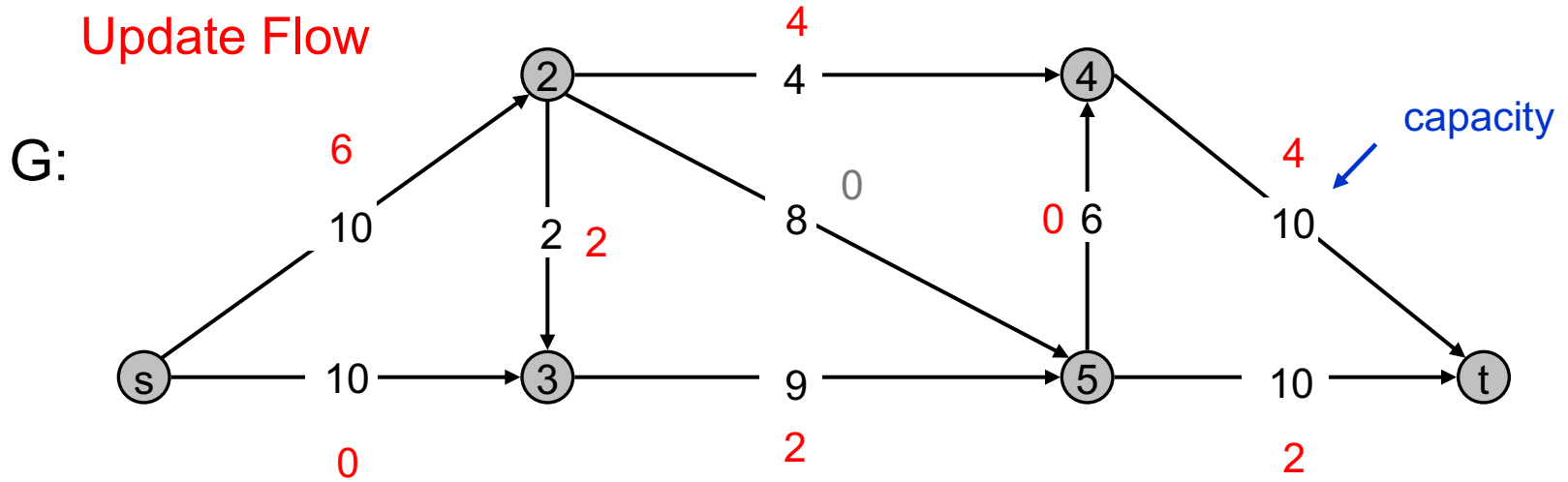
# Ford-Fulkerson Alg: Greedy on $G_f$



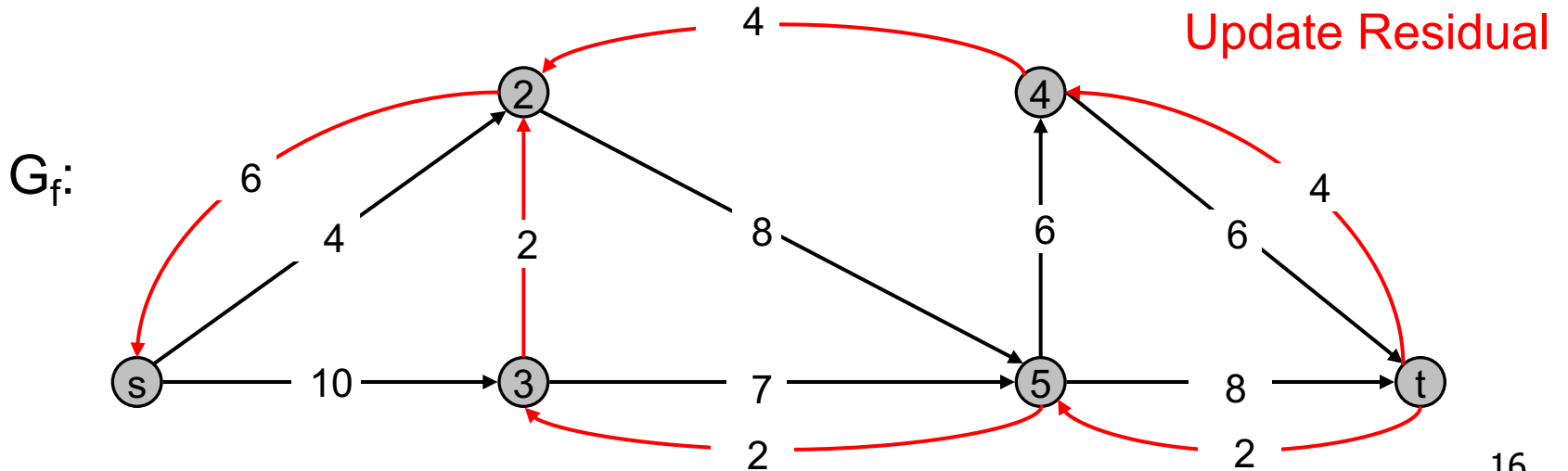
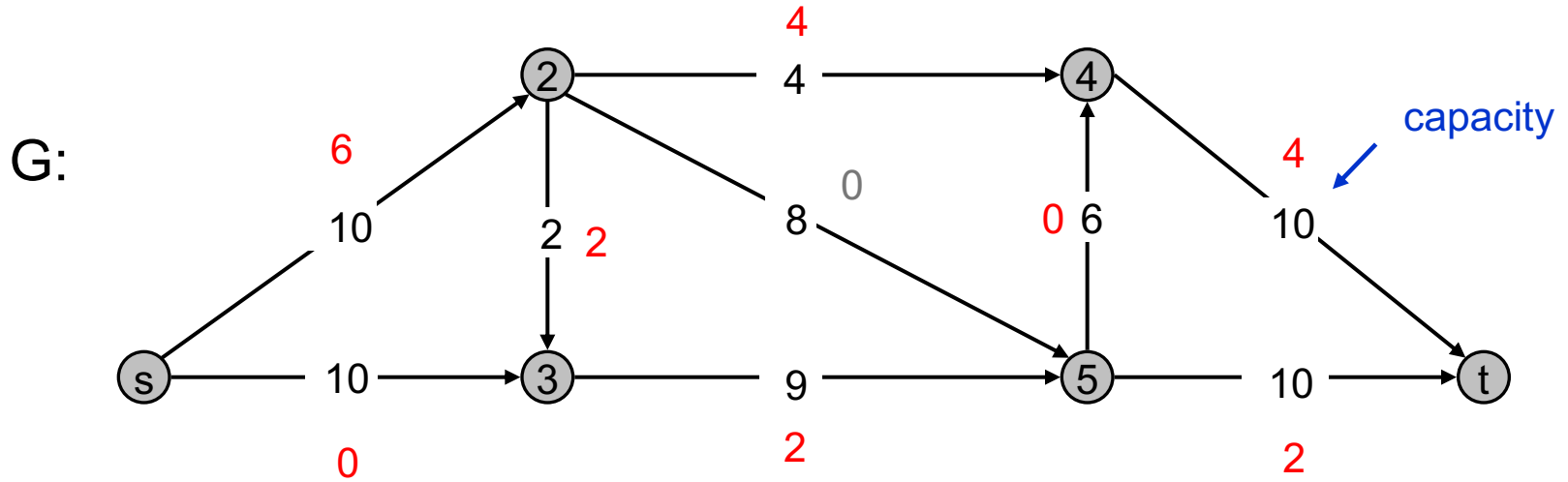
# Ford-Fulkerson Alg: Greedy on $G_f$



# Ford-Fulkerson Alg: Greedy on $G_f$

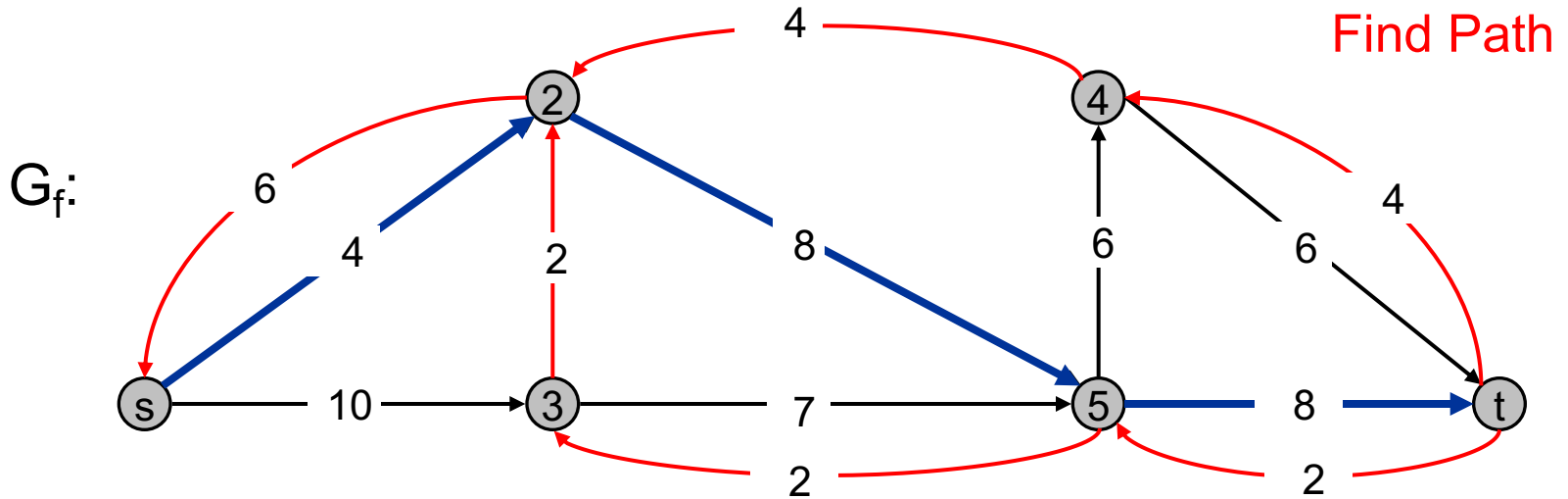
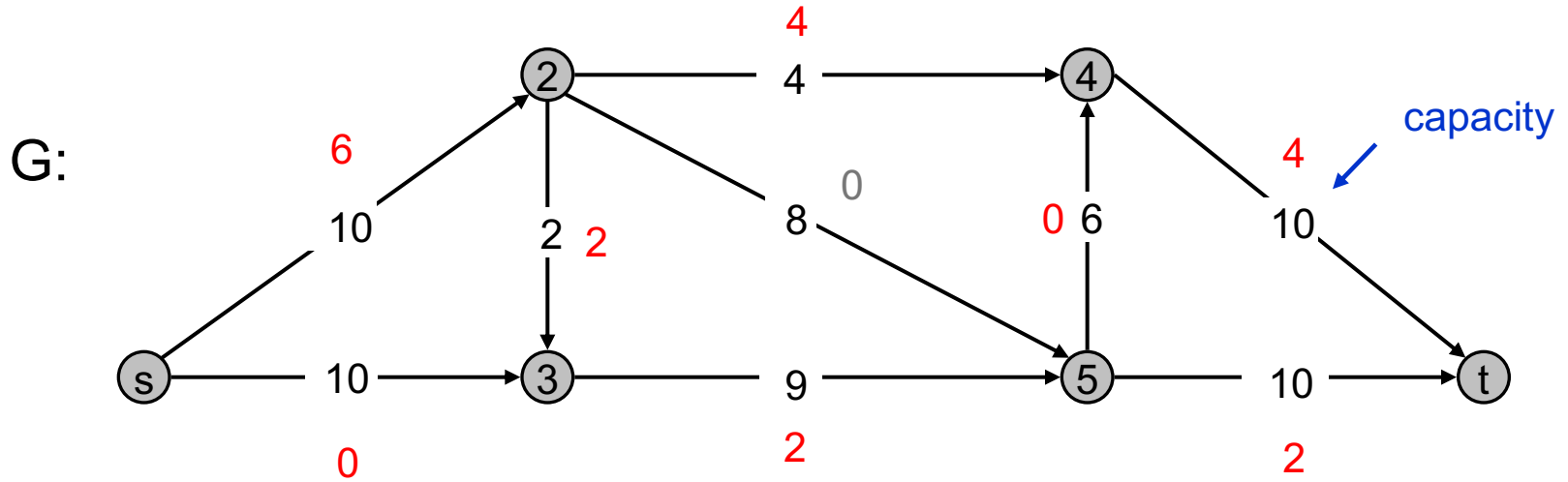


# Ford-Fulkerson Alg: Greedy on $G_f$



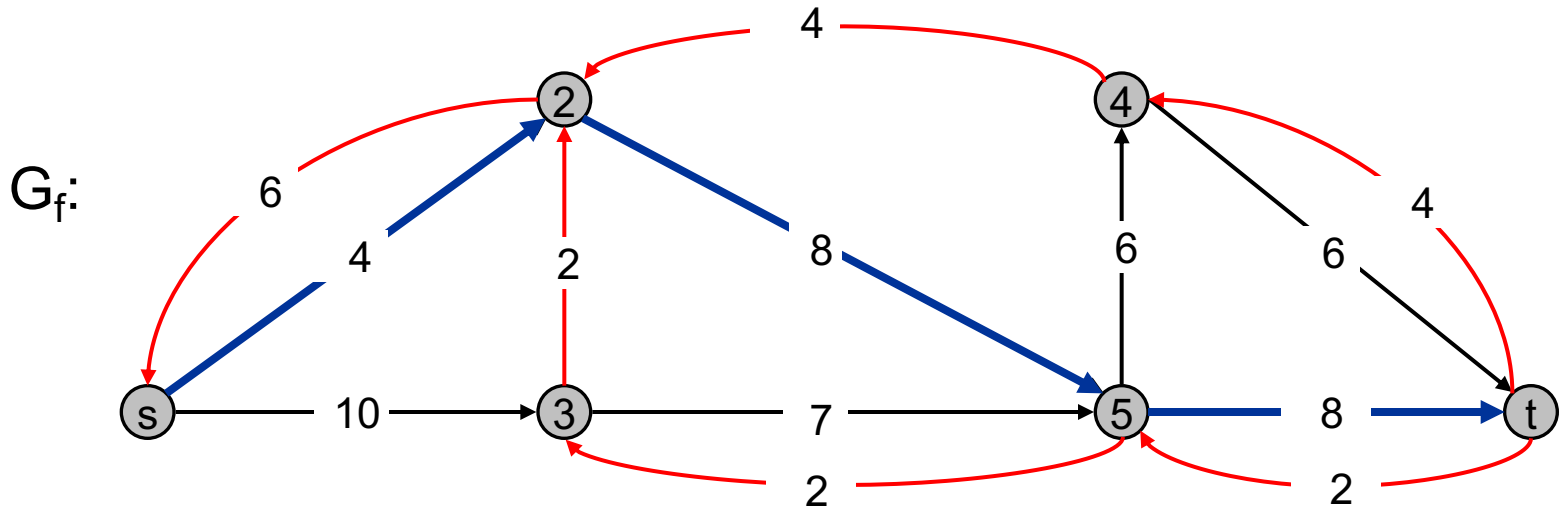
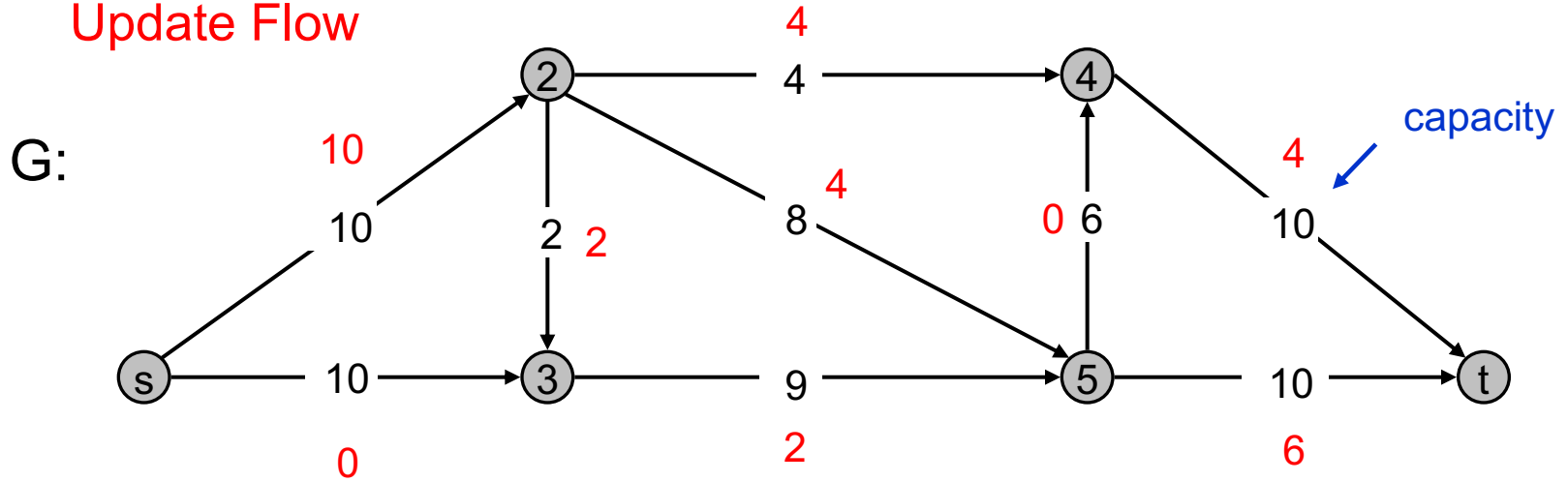


# Ford-Fulkerson Alg: Greedy on $G_f$

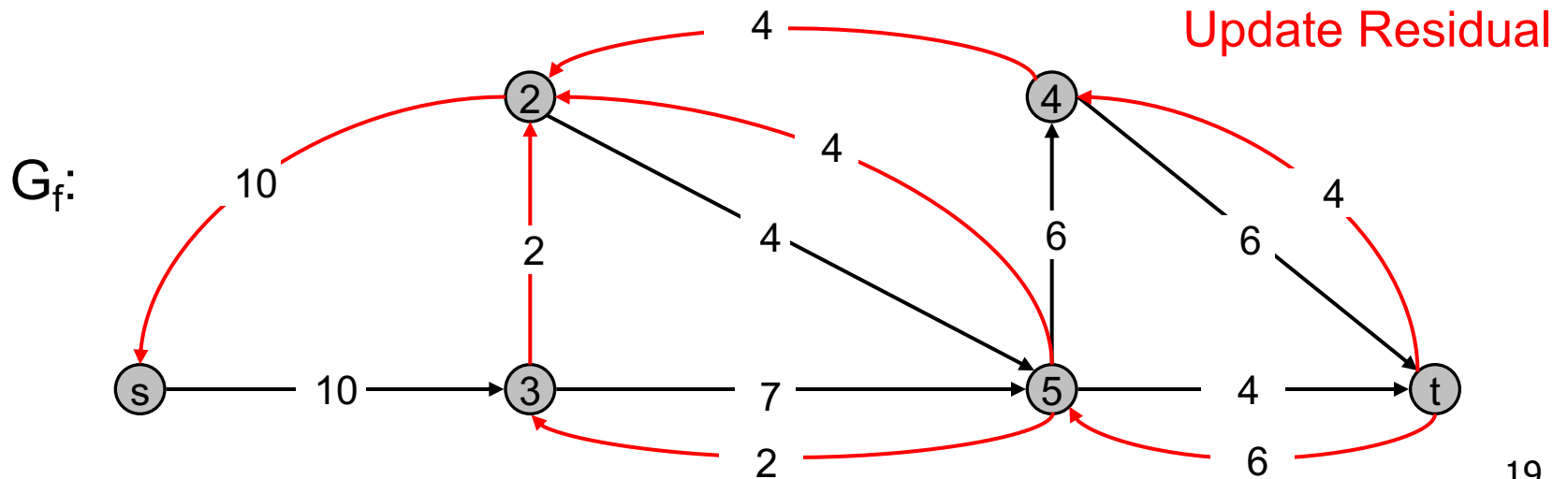
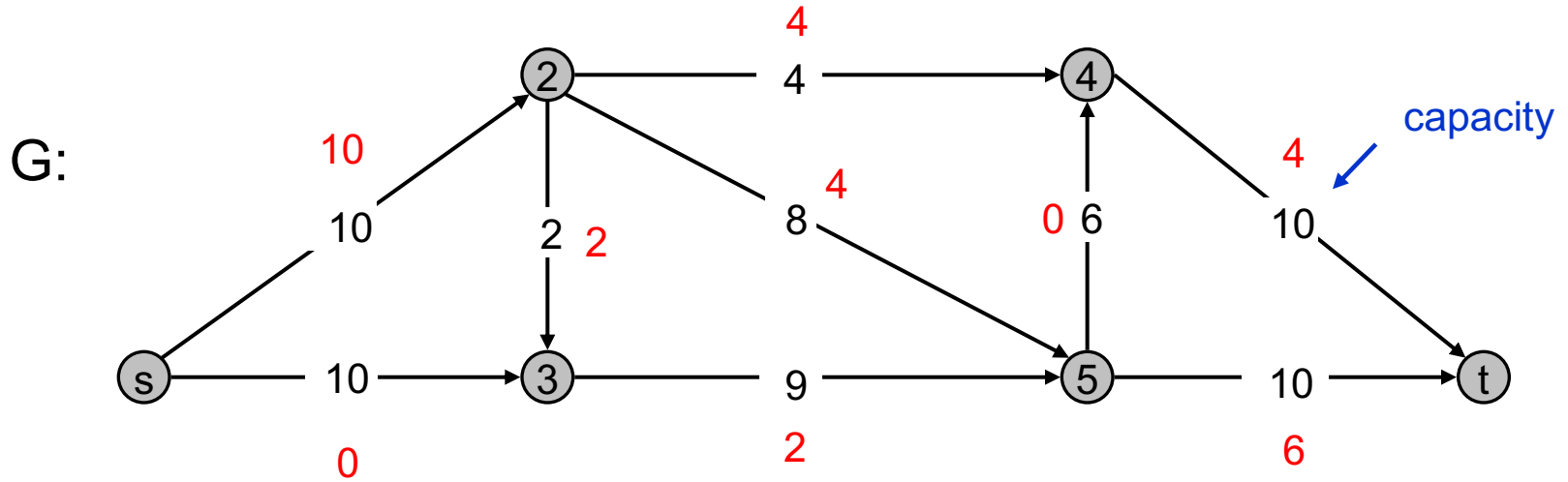


# Ford-Fulkerson Alg: Greedy on $G_f$

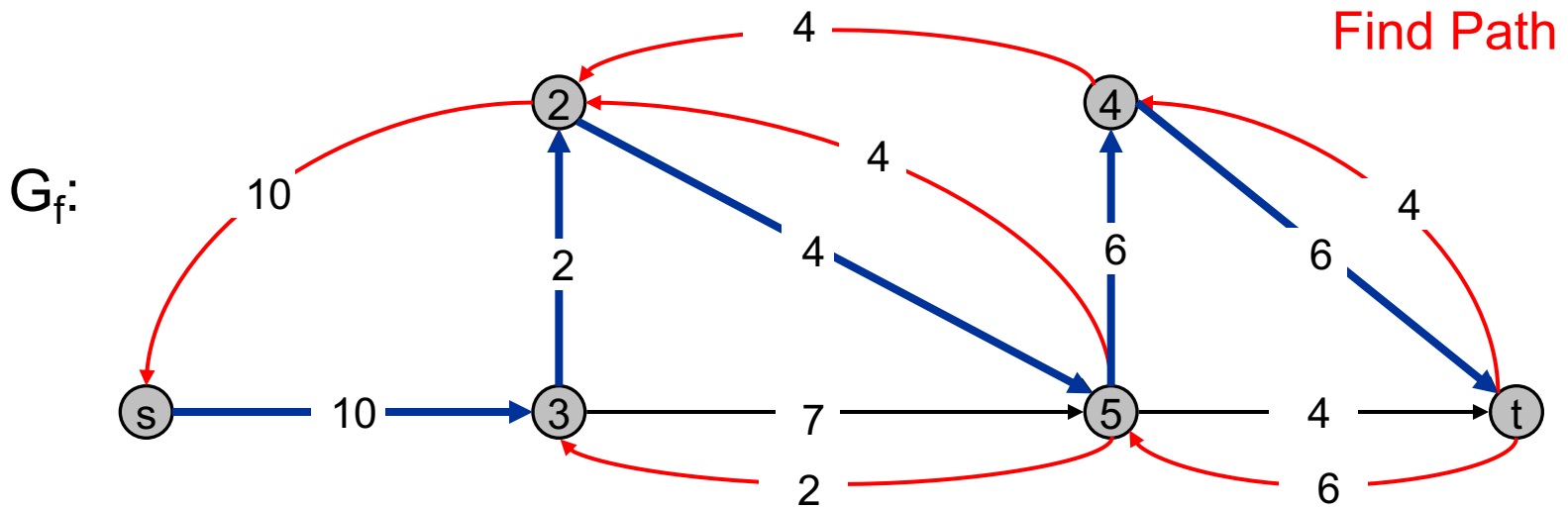
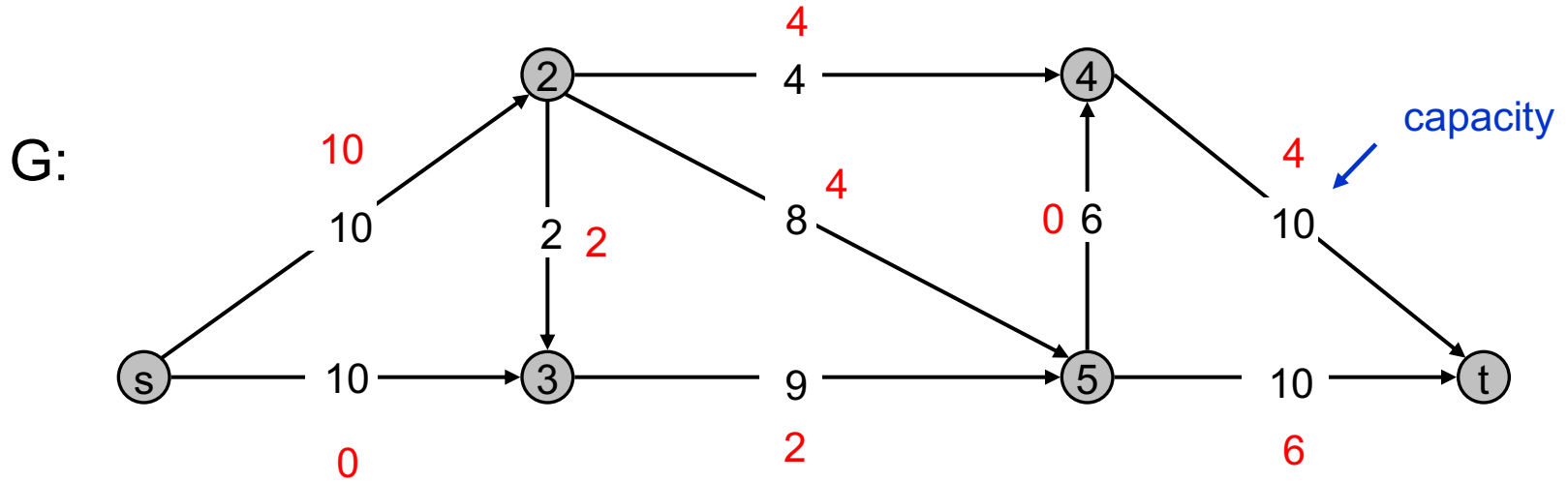
Update Flow



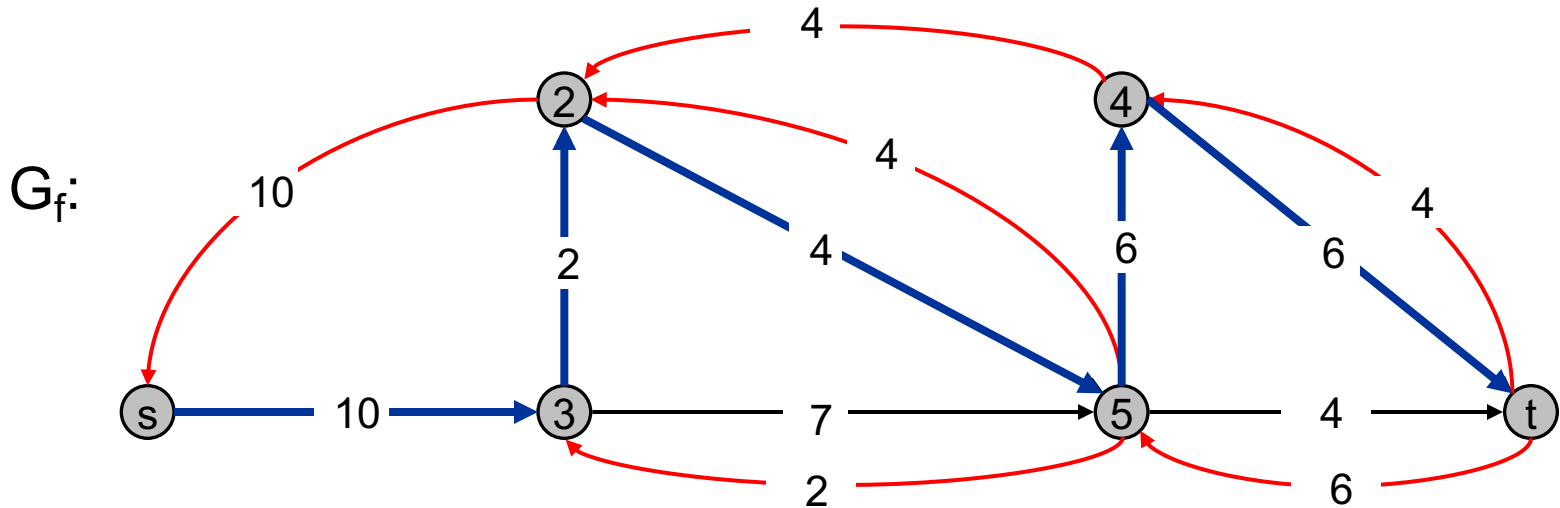
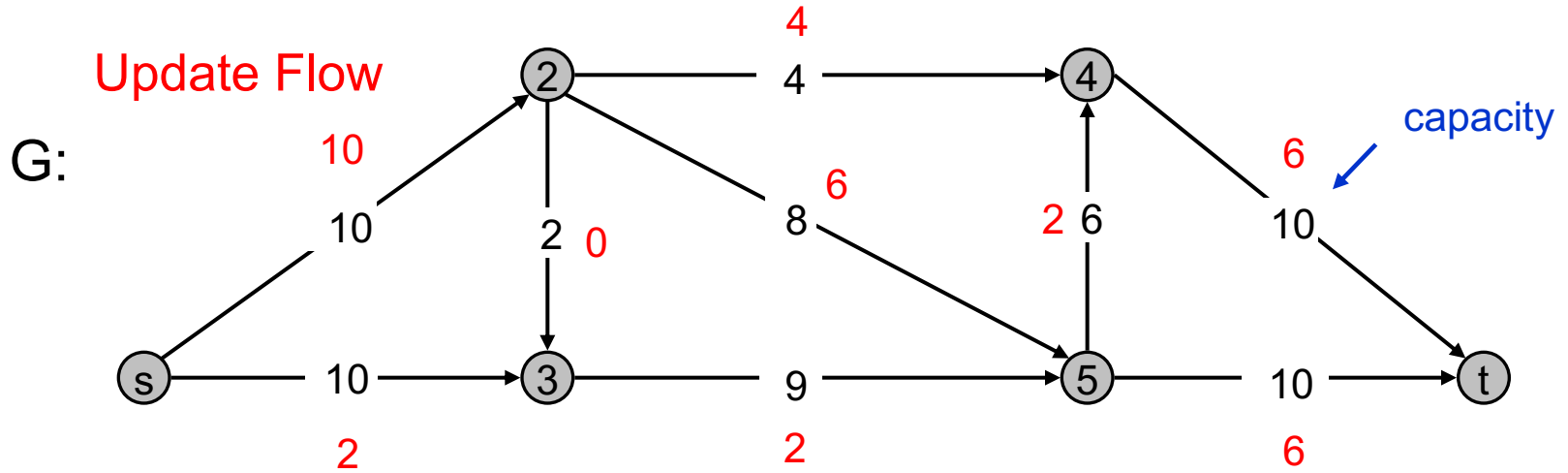
# Ford-Fulkerson Alg: Greedy on $G_f$



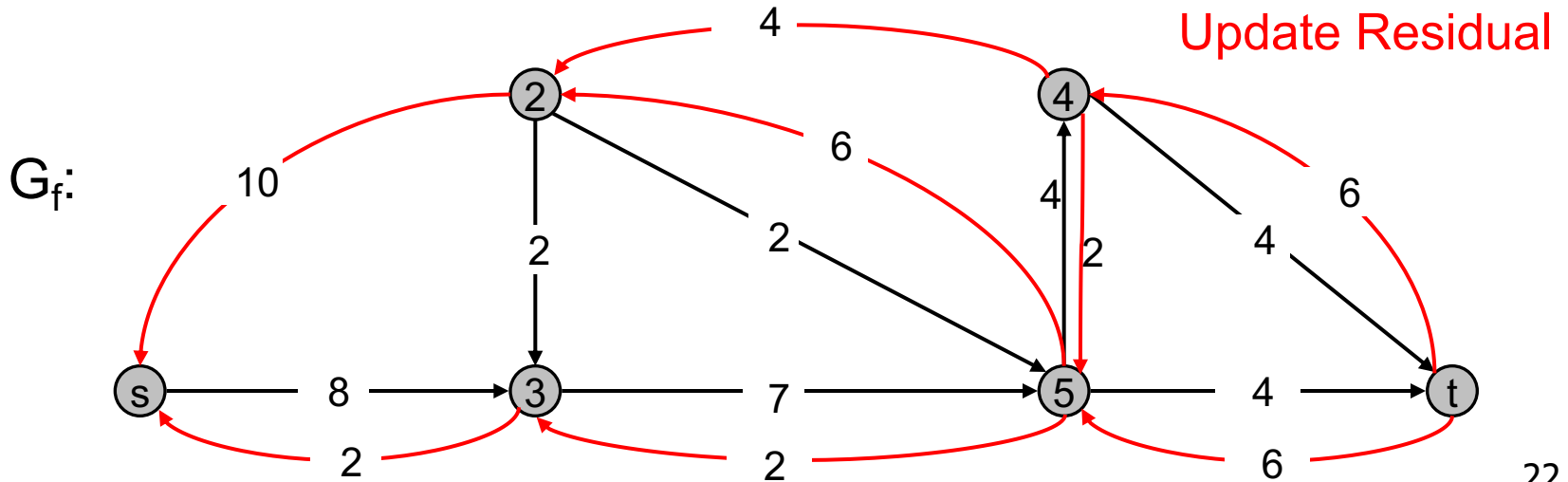
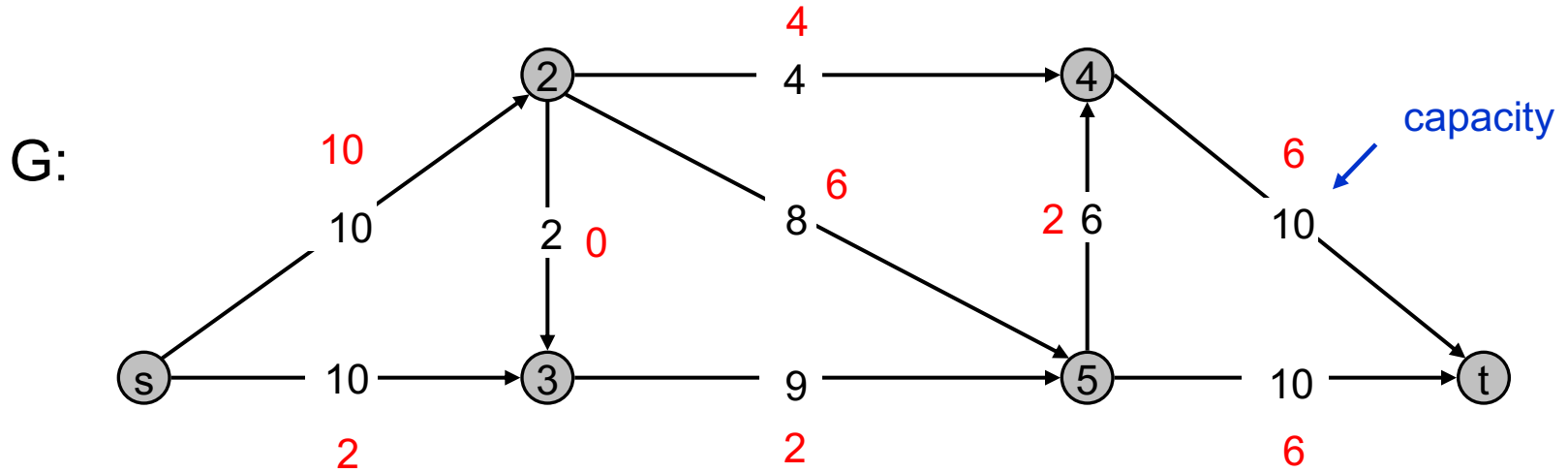
# Ford-Fulkerson Alg: Greedy on $G_f$



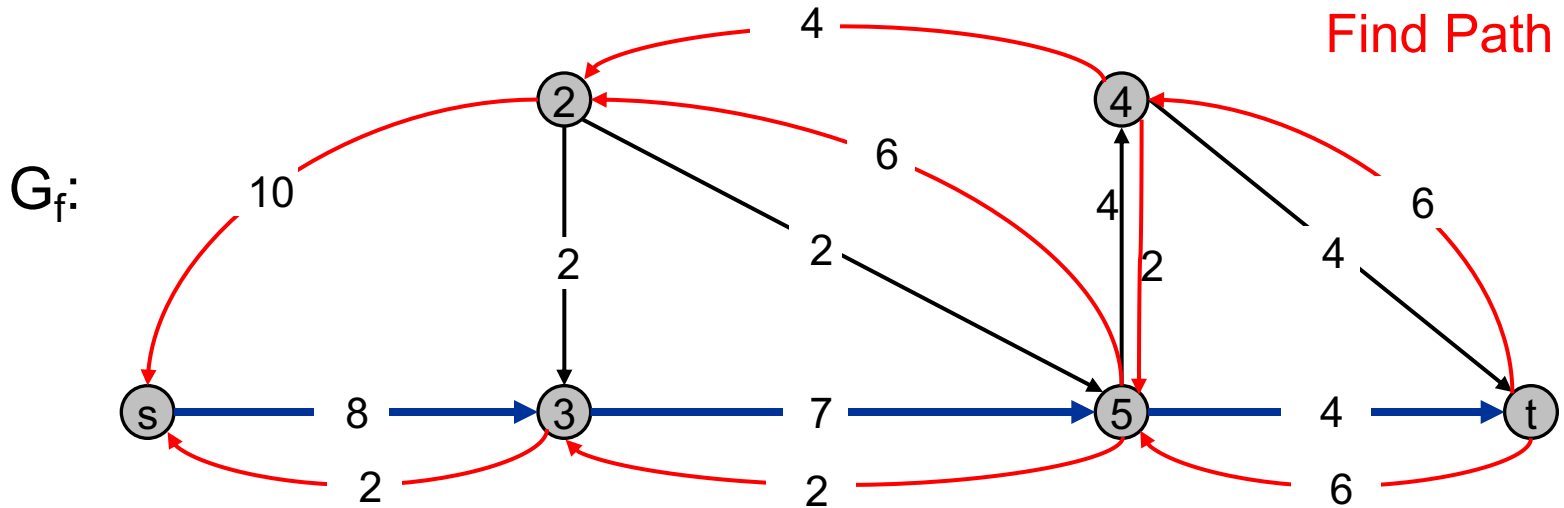
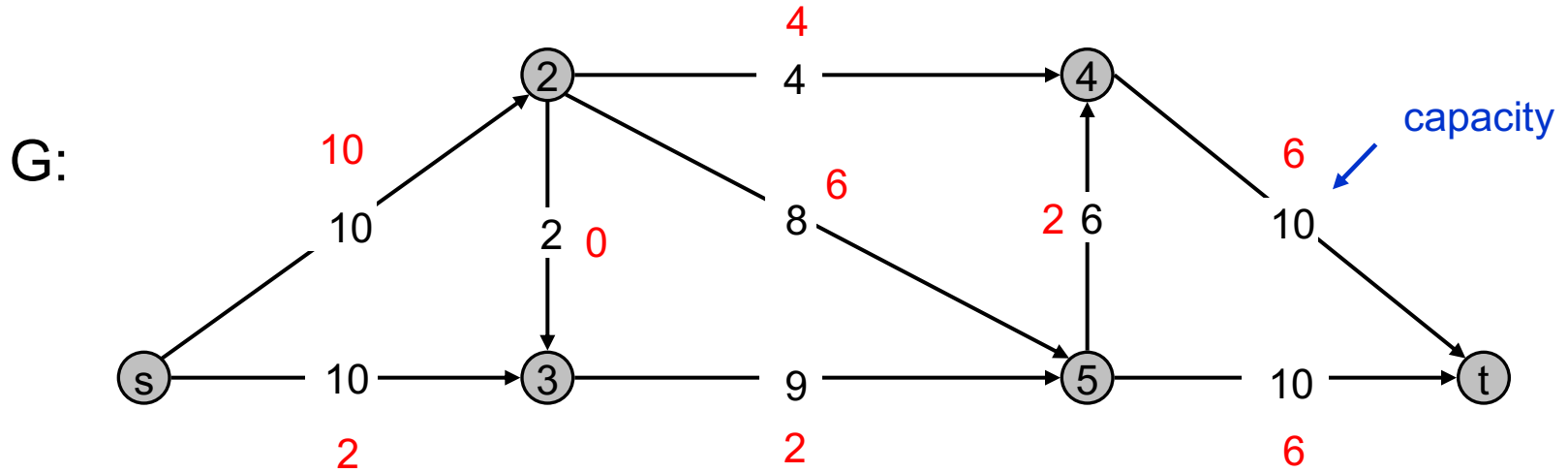
# Ford-Fulkerson Alg: Greedy on $G_f$



# Ford-Fulkerson Alg: Greedy on $G_f$

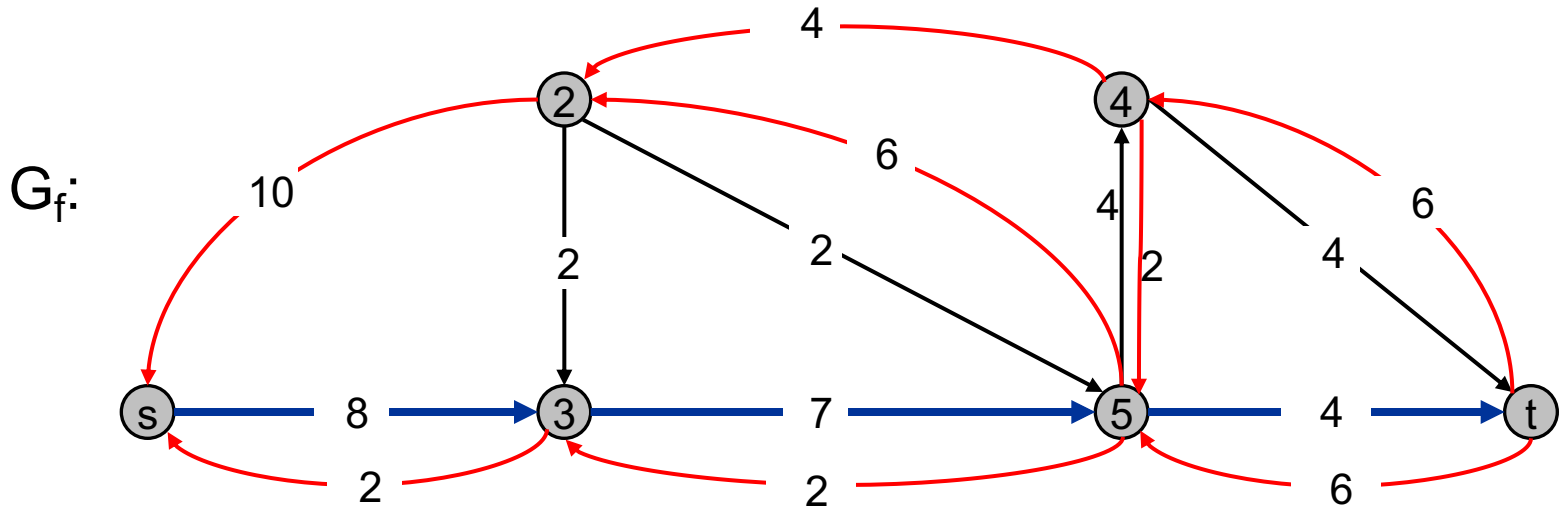
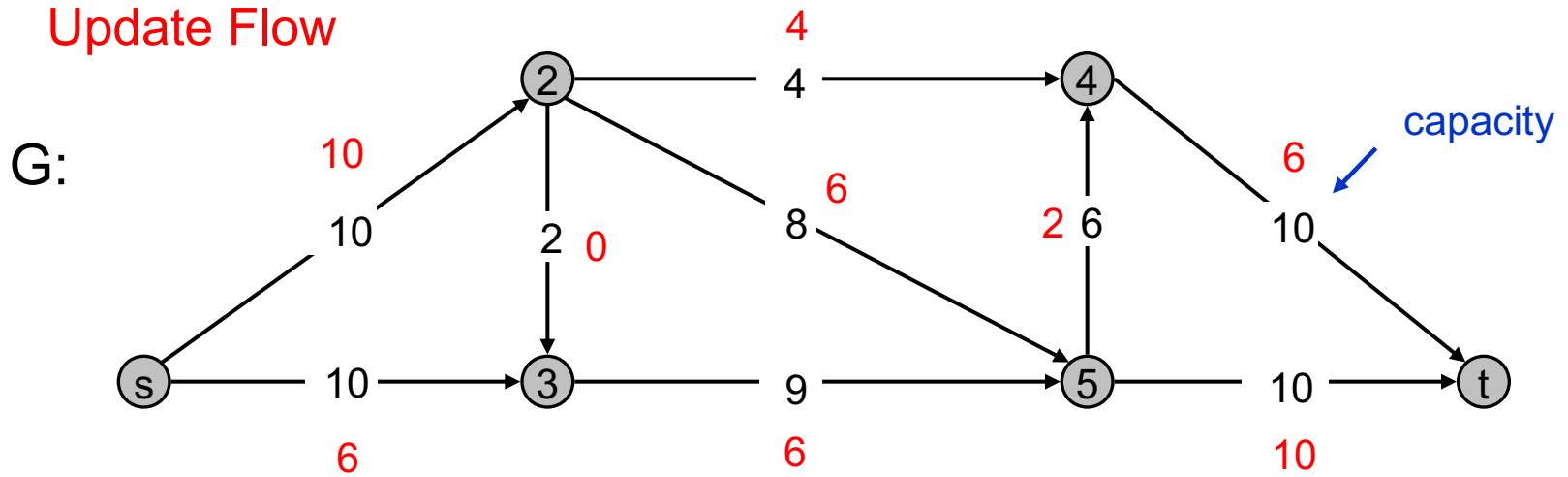


# Ford-Fulkerson Alg: Greedy on $G_f$



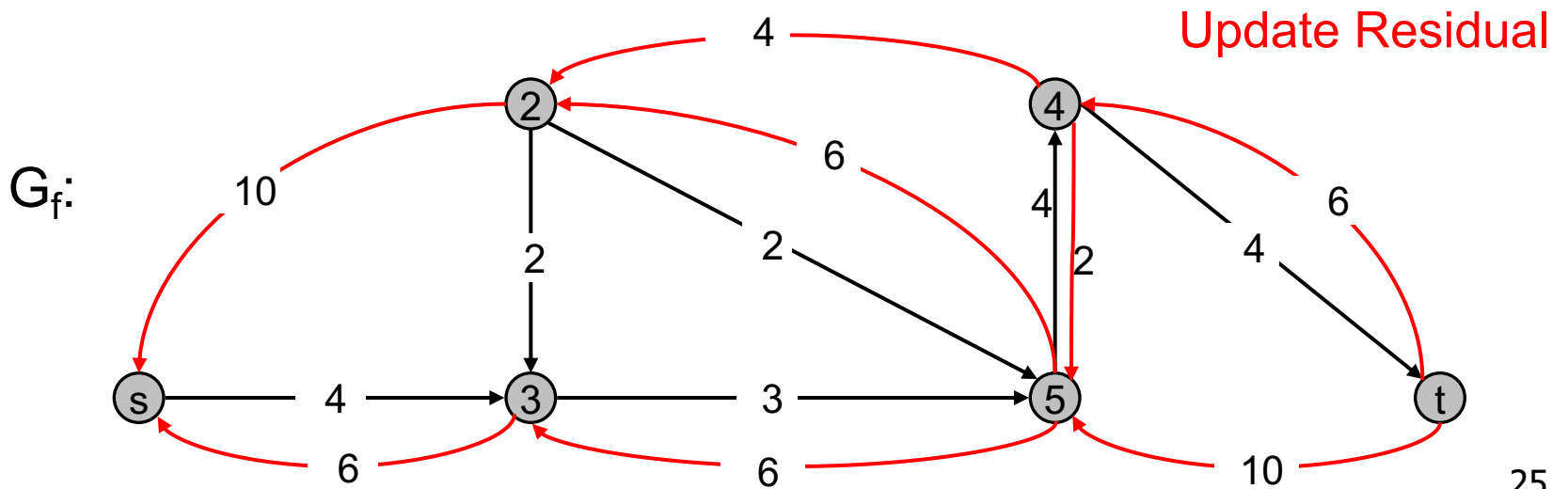
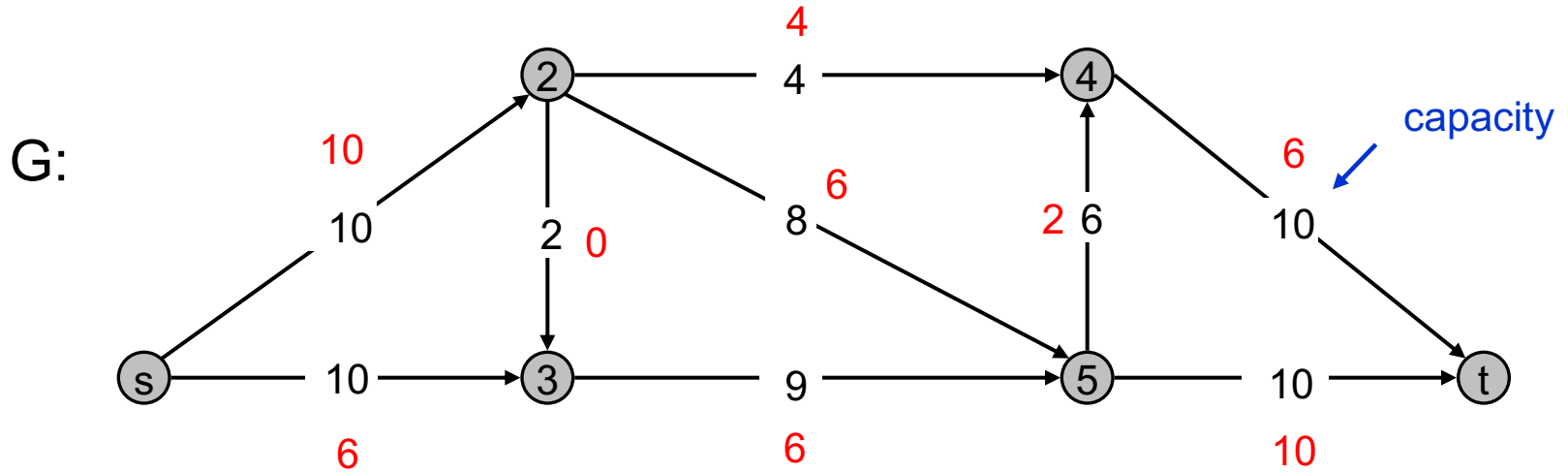
# Ford-Fulkerson Alg: Greedy on $G_f$

Update Flow

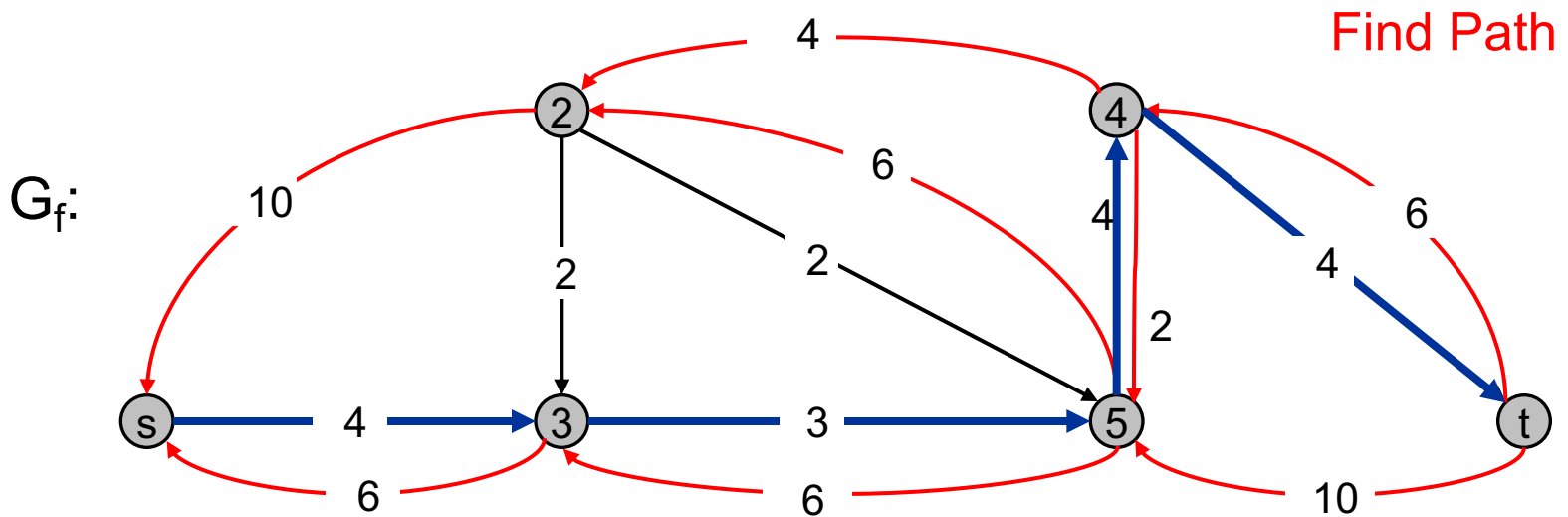
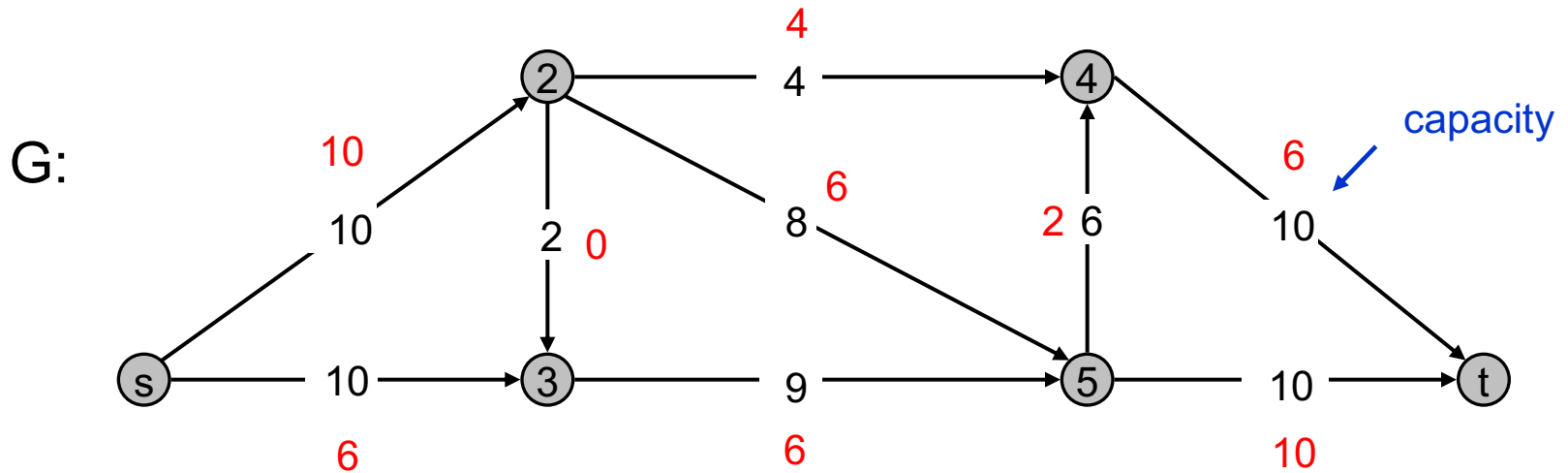




# Ford-Fulkerson Alg: Greedy on $G_f$

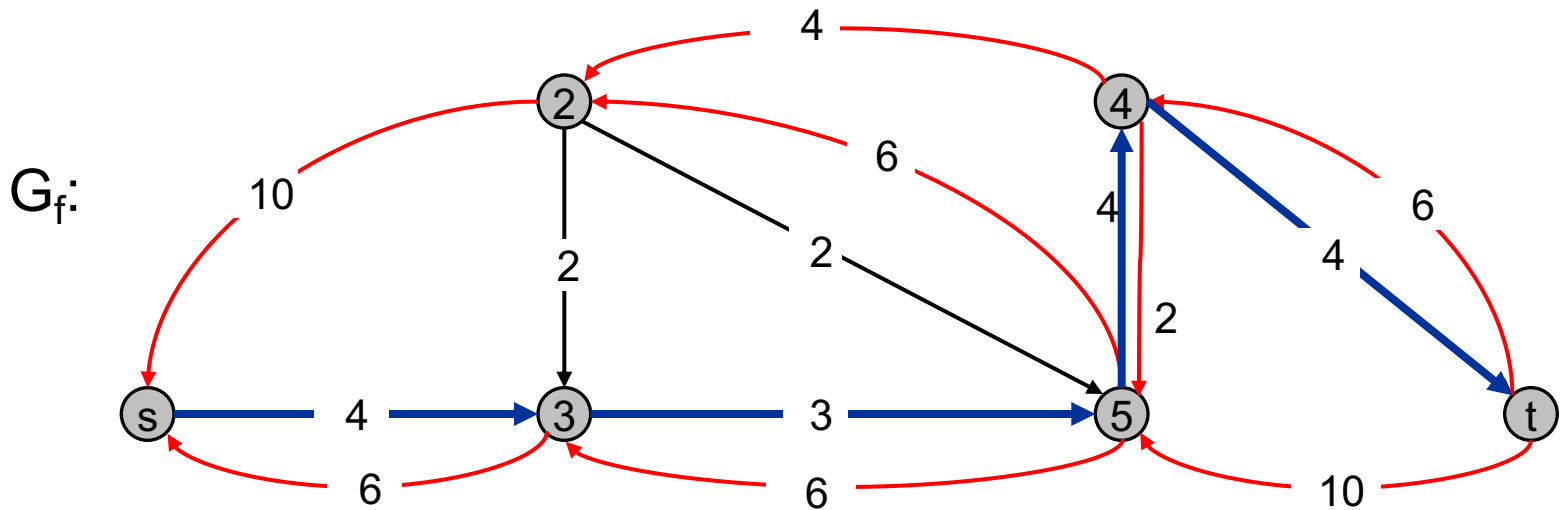
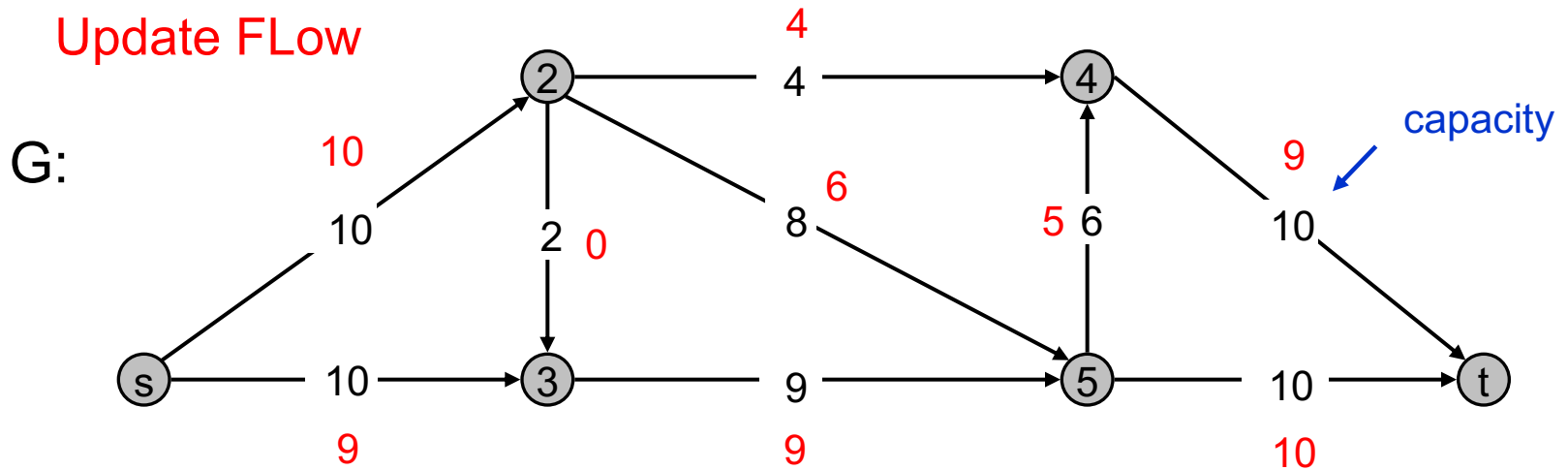


# Ford-Fulkerson Alg: Greedy on $G_f$

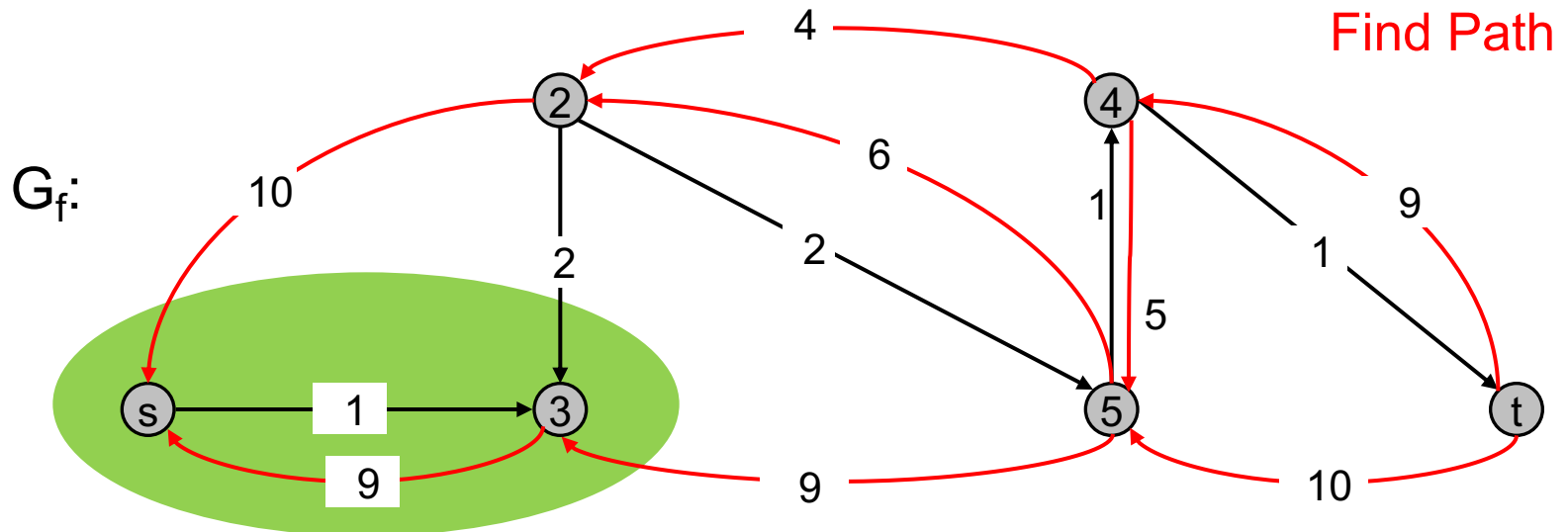
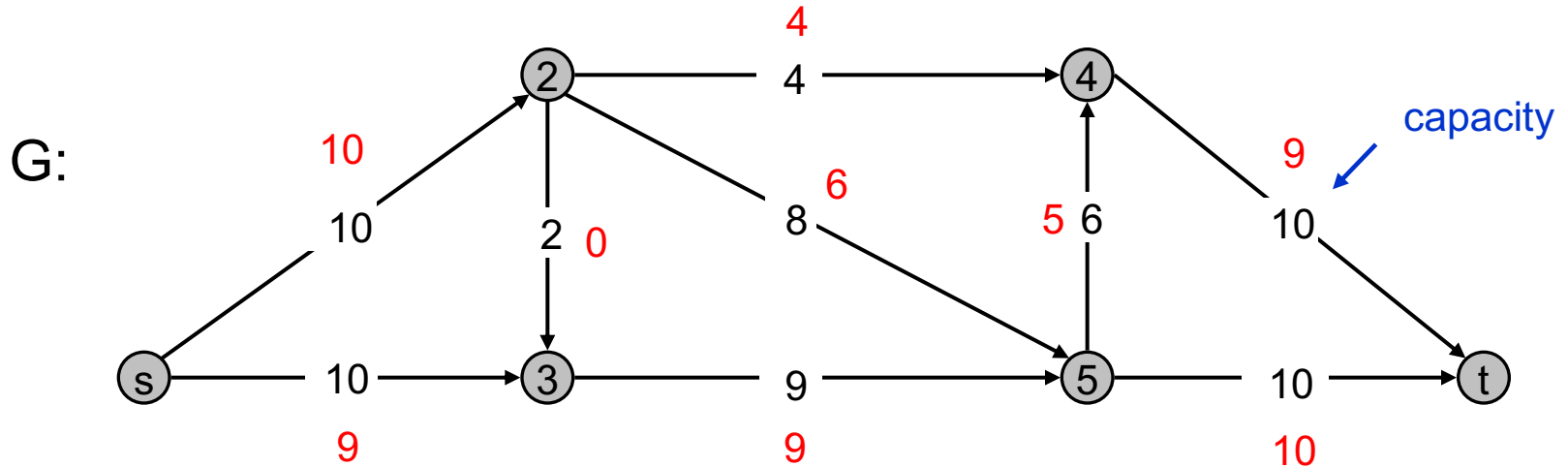


# Ford-Fulkerson Alg: Greedy on $G_f$

Update FLOW



# Ford-Fulkerson Alg: Greedy on $G_f$



# Augmenting Path Algorithm

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


```
Ford-Fulkerson(G, s, t, c) {  
  foreach e ∈ E f(e) ← 0. Gf is residual graph  
  while (there exists augmenting path P) {  
    f ← Augment(f, c, P)  
  }  
  return f  
}
```

# 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.** [Ford-Fulkerson 1956] The value of the max  $s$ - $t$  flow is equal to the value of the min  $s$ - $t$  cut.

**Proof strategy.** We prove both simultaneously by showing the TFAE:

- 
- (i) There exists a cut  $(A, B)$  such that  $v(f) = \text{cap}(A, B)$ .
  - (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.

(ii)  $\Rightarrow$  (iii) We show contrapositive.

Let  $f$  be a flow. If there exists an augmenting path, then we can improve  $f$  by sending flow along that path.

# Pf of Max Flow Min Cut Theorem

(iii)  $\Rightarrow$  (i)

No augmenting path for  $f \Rightarrow$  there is a cut  $(A,B)$ :  $v(f)=\text{cap}(A,B)$

- 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  $f$ ,  $t \notin A$ .

$$\begin{aligned}v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\ &= \sum_{e \text{ out of } A} c(e) \\ &= \text{cap}(A, B)\end{aligned}$$

# Running Time

**Assumption.** All capacities are integers between 1 and  $C$ .

**Invariant.** Every flow value  $f(e)$  and every residual capacities  $c_f(e)$  remains an **integer** throughout the algorithm.

**Theorem.** The algorithm terminates in at most  $v(f^*) \leq nC$  iterations, if  $f^*$  is optimal flow.

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

**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.



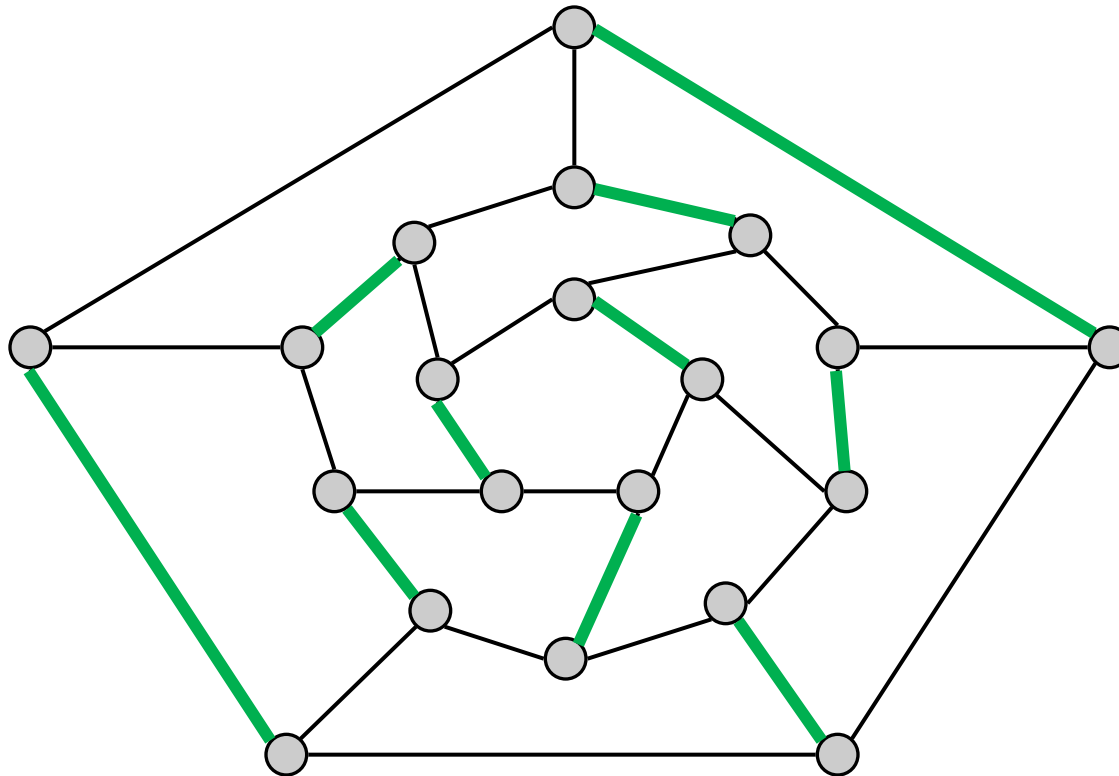
# Applications of Max Flow: Bipartite Matching

# Maximum Matching Problem

Given an undirected graph  $G = (V, E)$ .

A set  $M \subseteq E$  is a **matching** if each node appears in at most one edge in  $M$ .

**Goal:** find a matching with largest cardinality.

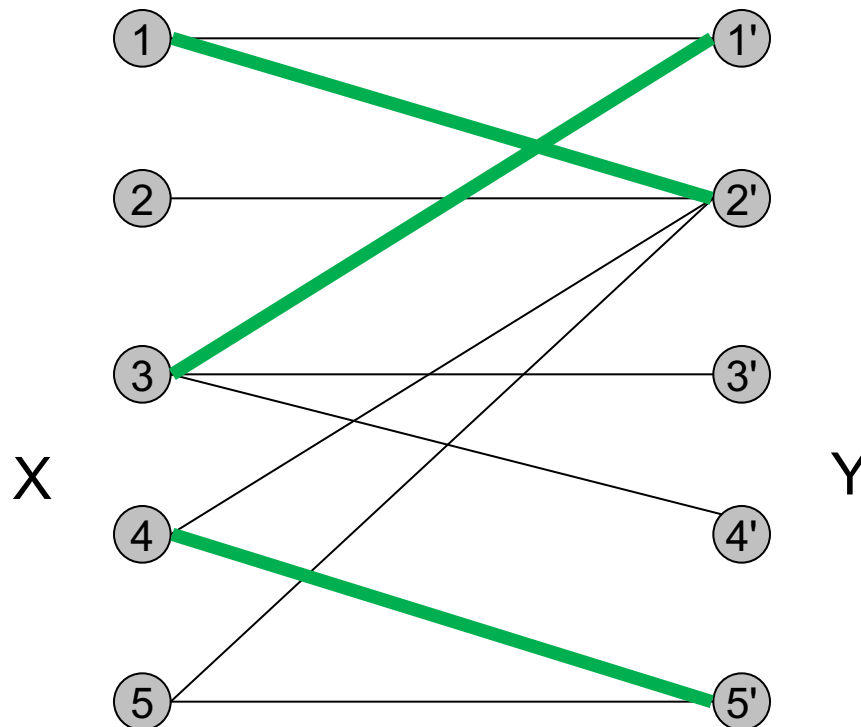


# Bipartite Matching Problem

Given an undirected bipartite graph  $G = (X \cup Y, E)$

A set  $M \subseteq E$  is a **matching** if each node appears in at most one edge in  $M$ .

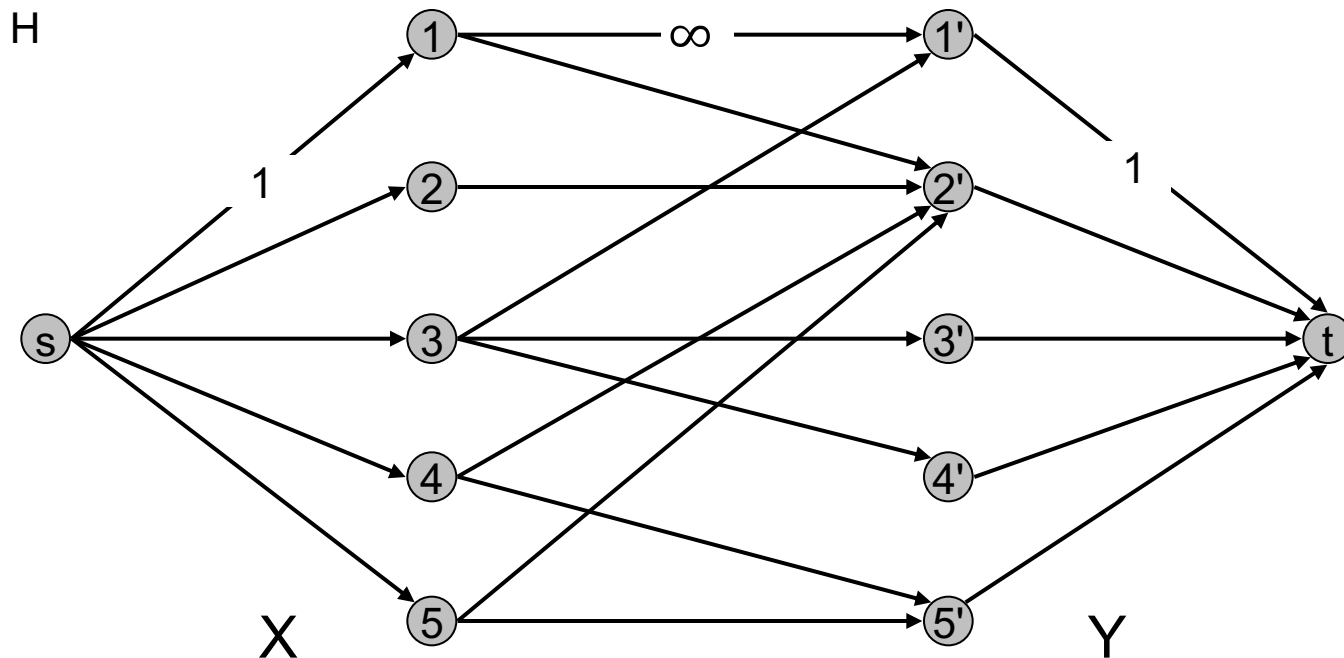
**Goal:** find a matching with largest cardinality.



# Bipartite Matching using Max Flow

Create digraph H as follows:

- Orient all edges from X to Y, and assign infinite (or unit) capacity.
- Add source s, and **unit** capacity edges from s to each node in L.
- Add sink t, and **unit** capacity edges from each node in R to t.



# Bipartite Matching: Proof of Correctness

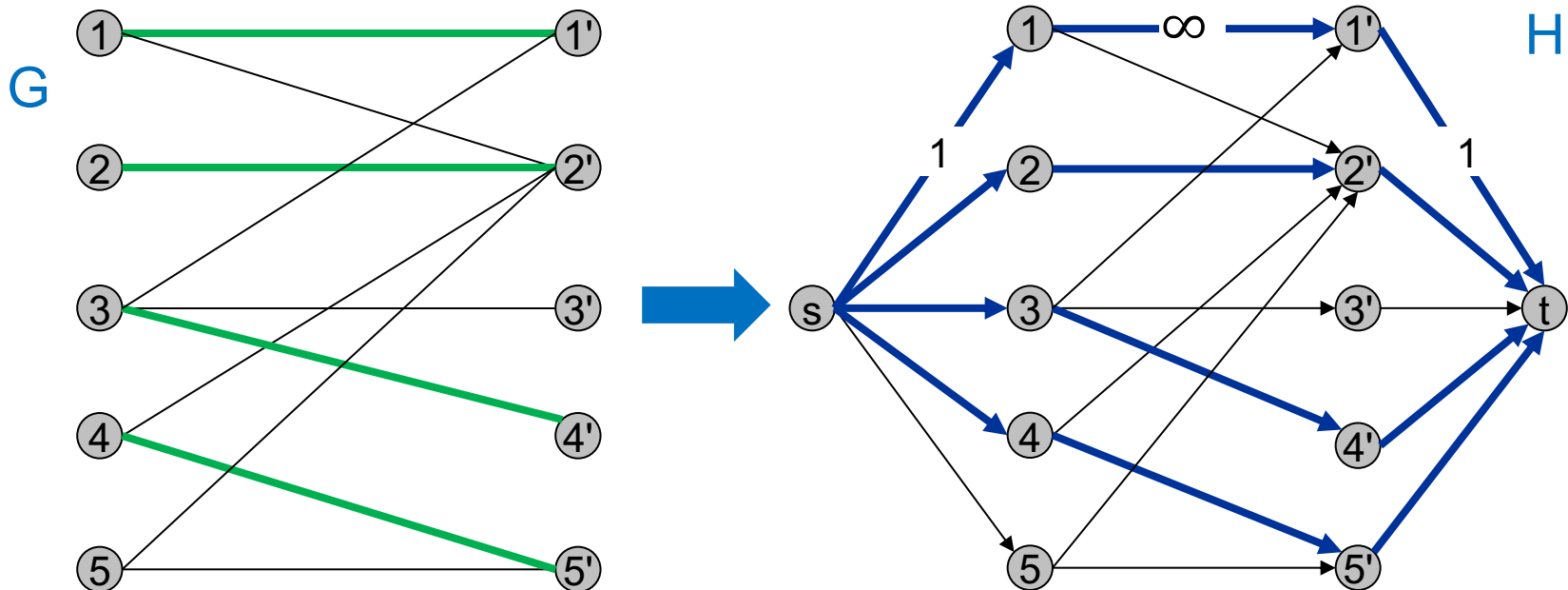
**Thm.** Max cardinality matching in  $G$  = value of max flow in  $H$ .

**Pf.**  $\leq$

Given max matching  $M$  of cardinality  $k$ .

Consider flow  $f$  that sends 1 unit along each of  $k$  edges of  $M$ .

$f$  is a flow, and has cardinality  $k$ . ■



# Bipartite Matching: Proof of Correctness

**Thm.** Max cardinality matching in  $G$  = value of max flow in  $H$ .

**Pf. (of  $\geq$ )** Let  $f$  be a max flow in  $H$  of value  $k$ .

Integrality theorem  $\Rightarrow$   $k$  is integral and we can assume  $f$  is 0-1.

Consider  $M$  = set of edges from  $X$  to  $Y$  with  $f(e) = 1$ .

- each node in  $X$  and  $Y$  participates in at most one edge in  $M$
- $|M| = k$ : consider s-t cut  $(s \cup X, t \cup Y)$

