

# CS3000: Algorithms & Data

## Paul Hand

### Lecture 21:

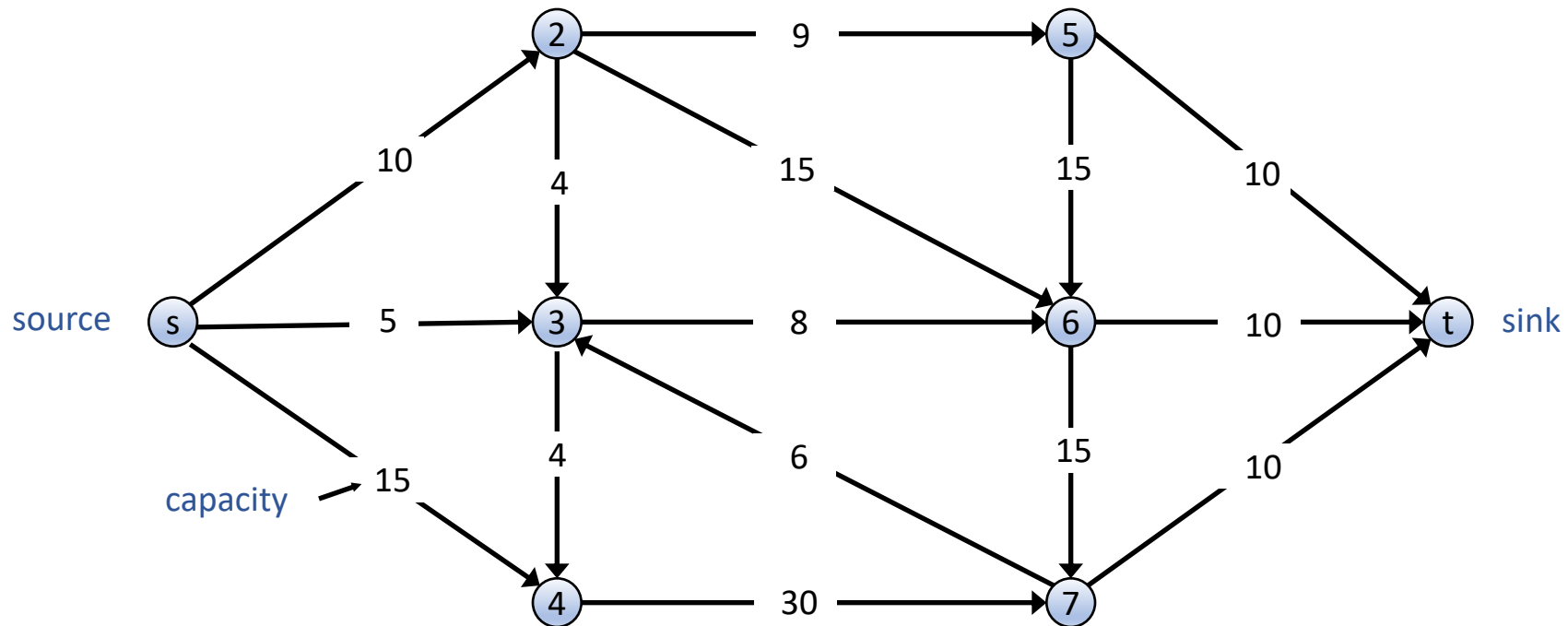
- Network Flow: flows, cuts, duality
- Ford-Fulkerson

Apr 10, 2019

# Flow Networks

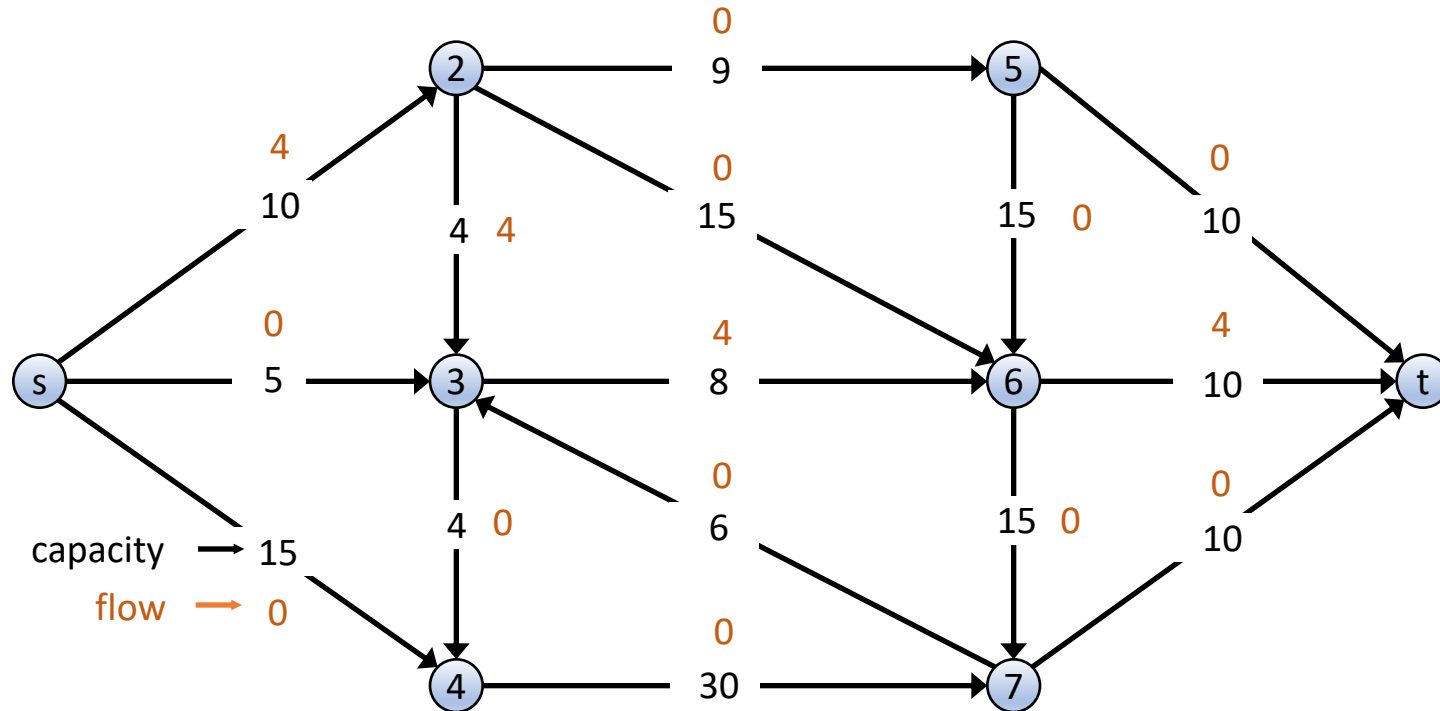
# Flow Networks

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



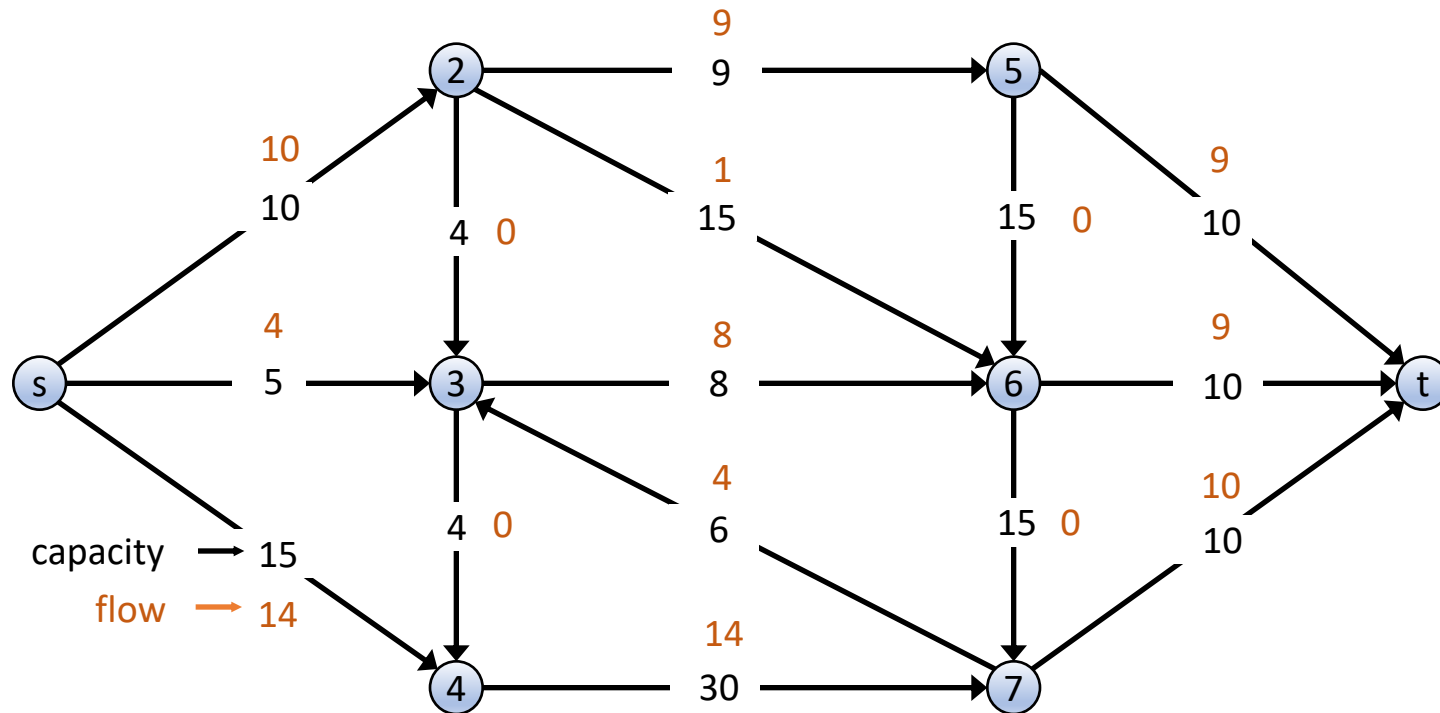
# Flows

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



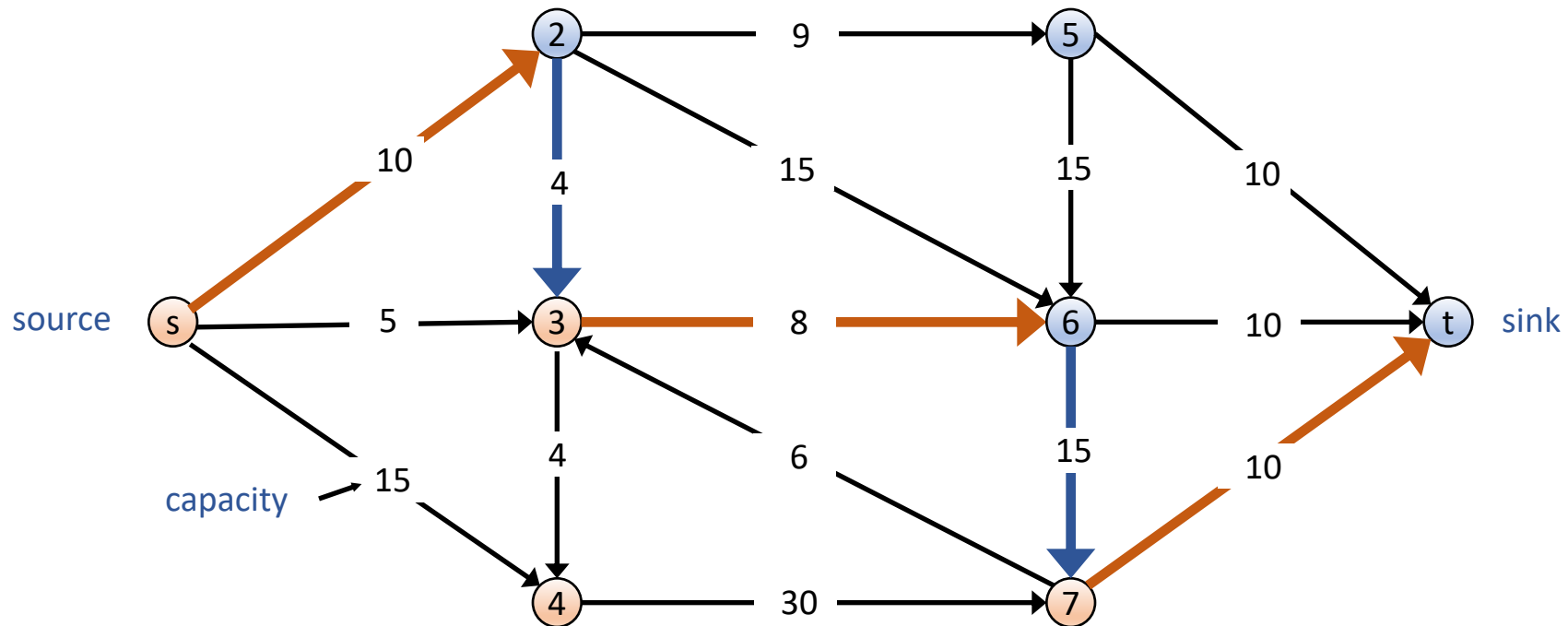
# Maximum Flow Problem

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



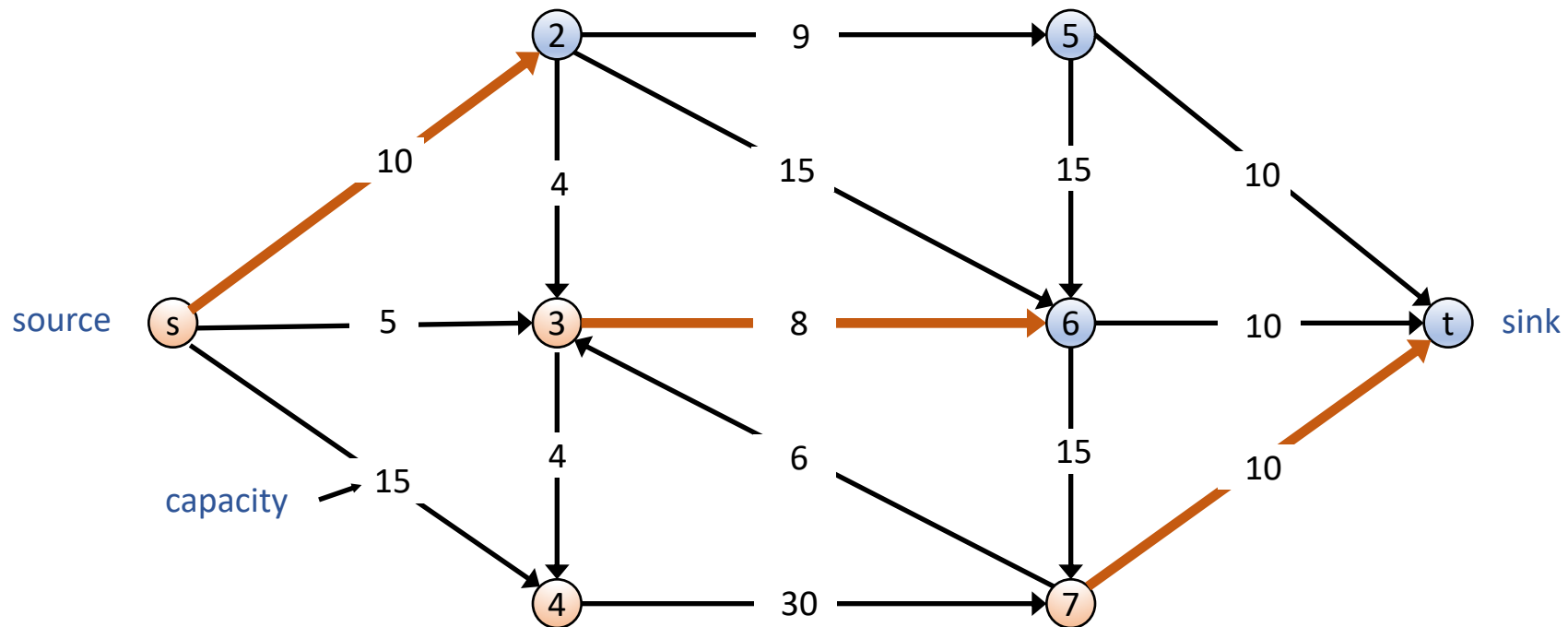
# Cuts

- 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

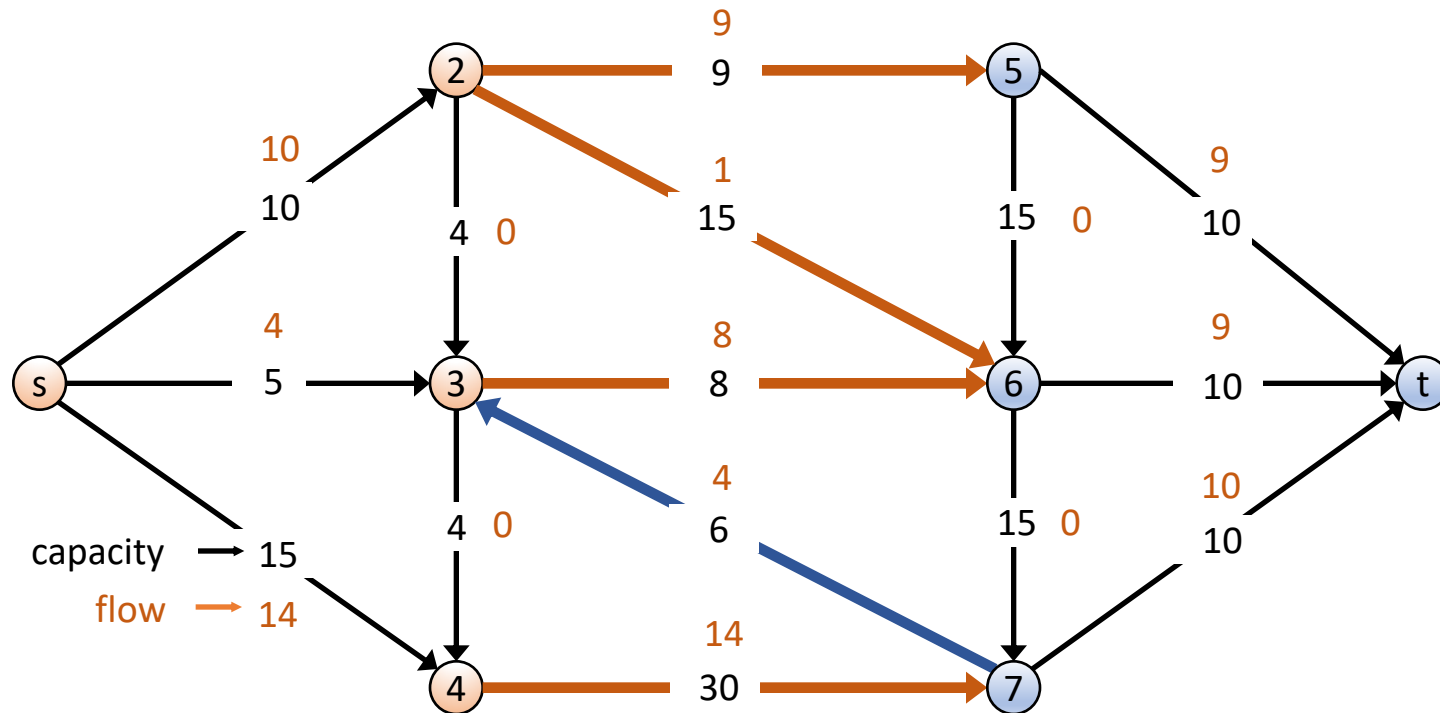
- Given  $G = (V, E, s, t, \{c(e)\})$ , find an s-t cut of minimum capacity



# 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

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





# Max Flow Min Cut Duality

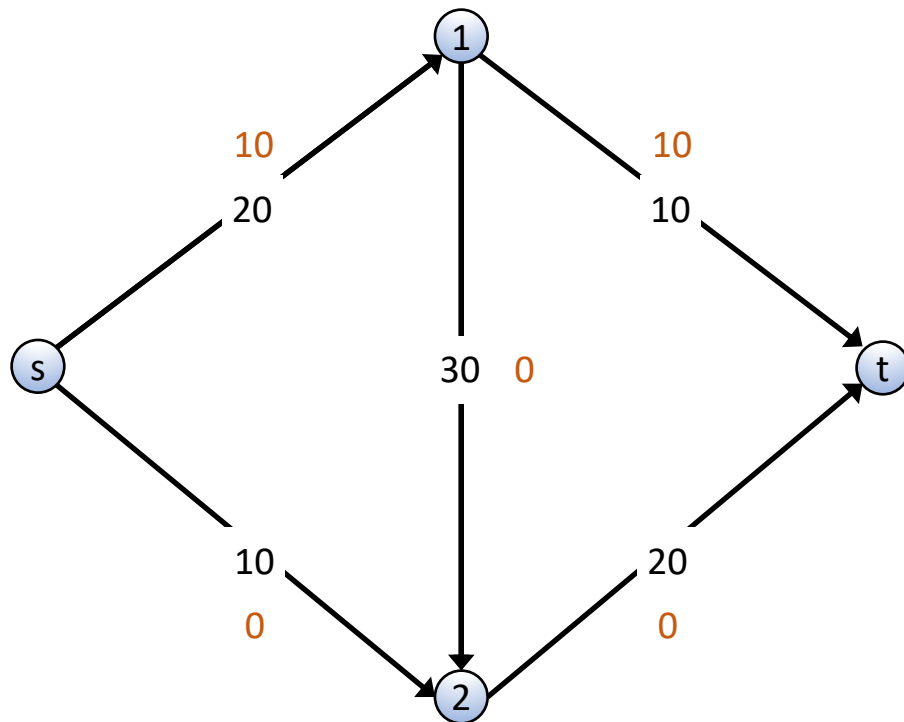
- **Weak Duality:** Let  $f$  be any s-t flow and  $(A, B)$  any s-t cut,

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

- **Proof:**

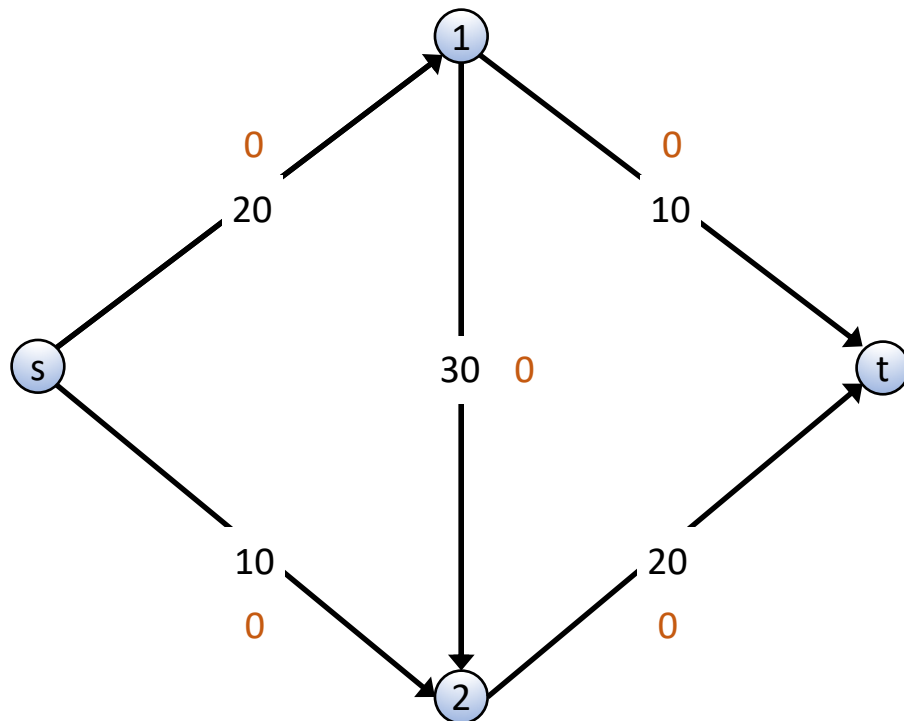
# Augmenting Paths

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



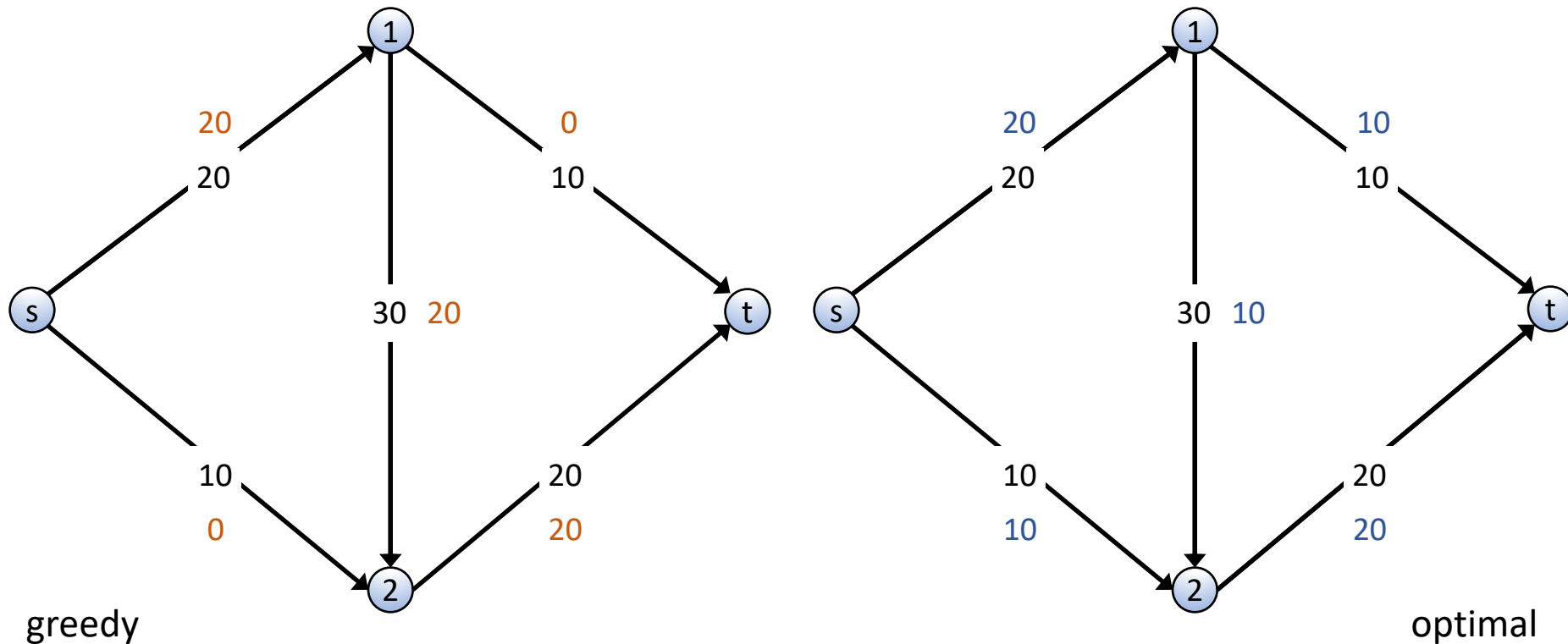
# Greedy Max Flow

- Start with  $f(e) = 0$  for all edges  $e \in E$
- Find an **augmenting path**  $P$ , max it out
- 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 edge
  - Allows “undoing” flow
  - $e = (u, v)$  and  $e^R = (v, u)$ .
  - Residual capacity
- Residual graph  $G_f = (V, E_f)$ 
  - Edges with positive residual capacity.
  - $E_f = \{e : f(e) < c(e)\} \cup \{e^R : f(e) > 0\}$ .

# 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( $G_f, P$ )
   $b \leftarrow$  the minimum capacity of an edge in  $P$  of  $G_f$ 
  for  $e \in P$ 
    if  $e \in E$ :     $f(e) \leftarrow f(e) + b$ 
    else:          $f(e) \leftarrow f(e) - b$ 
  return  $f$ 
```

# Ford-Fulkerson Algorithm

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

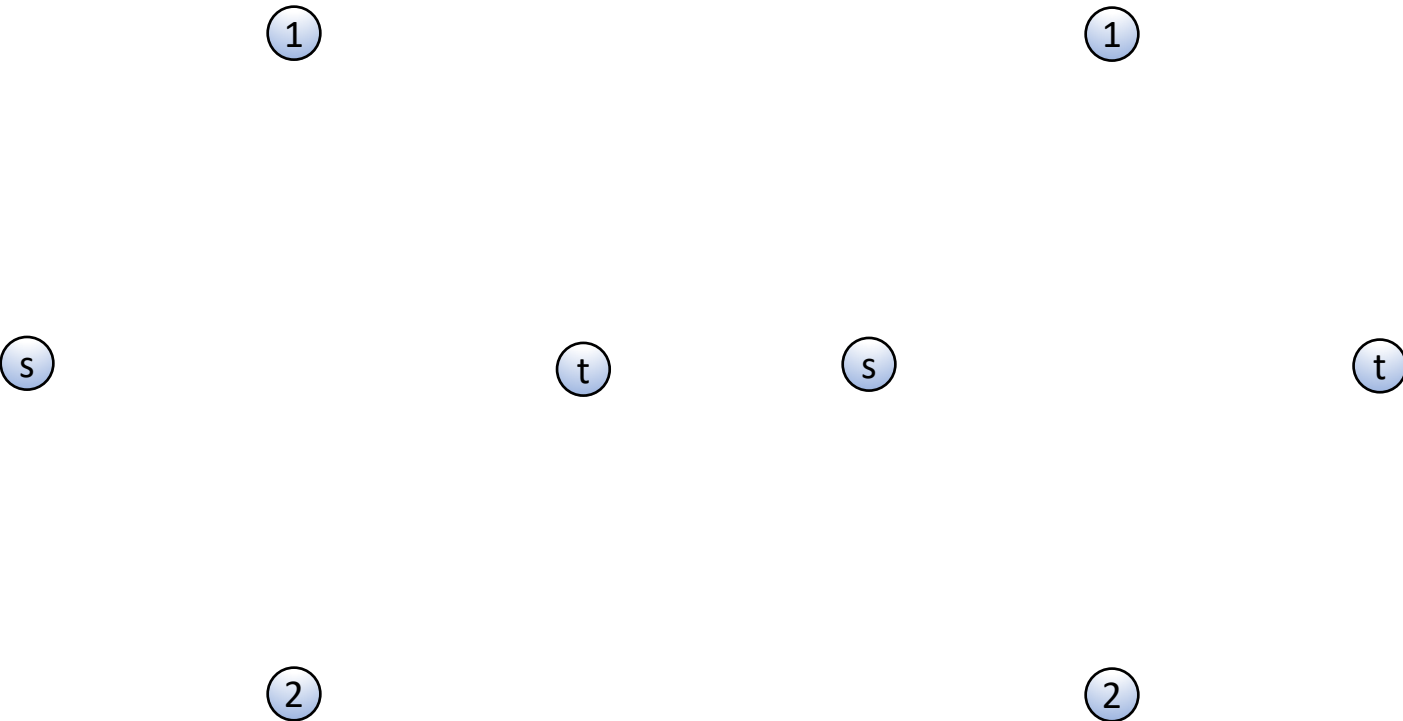
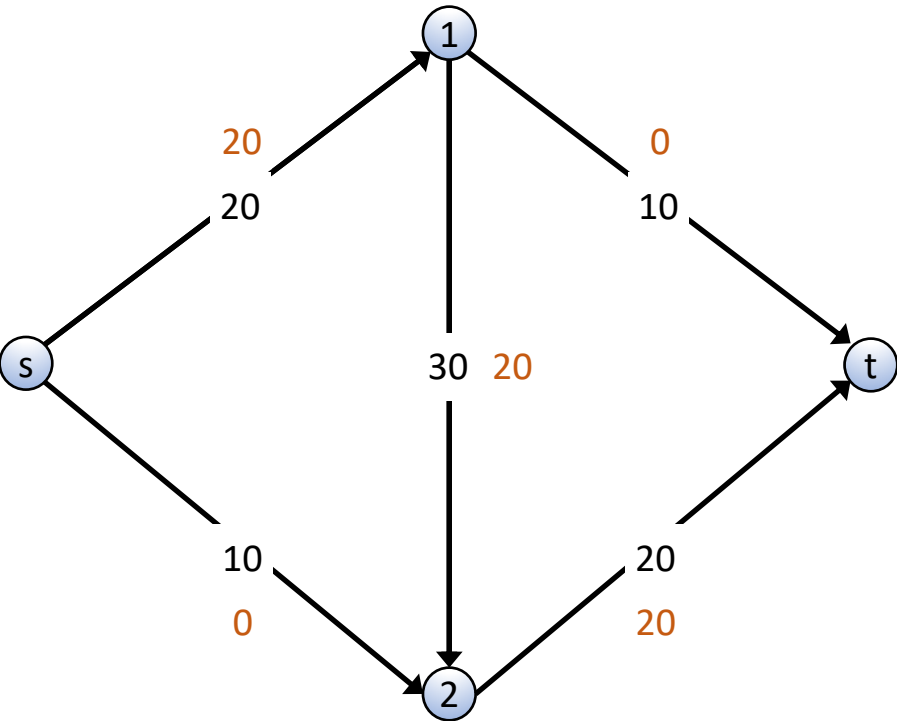
  while (there is an s-t path P in Gf)
    f ← Augment(Gf, P)
    update Gf

  return f
```

```
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 Algorithm

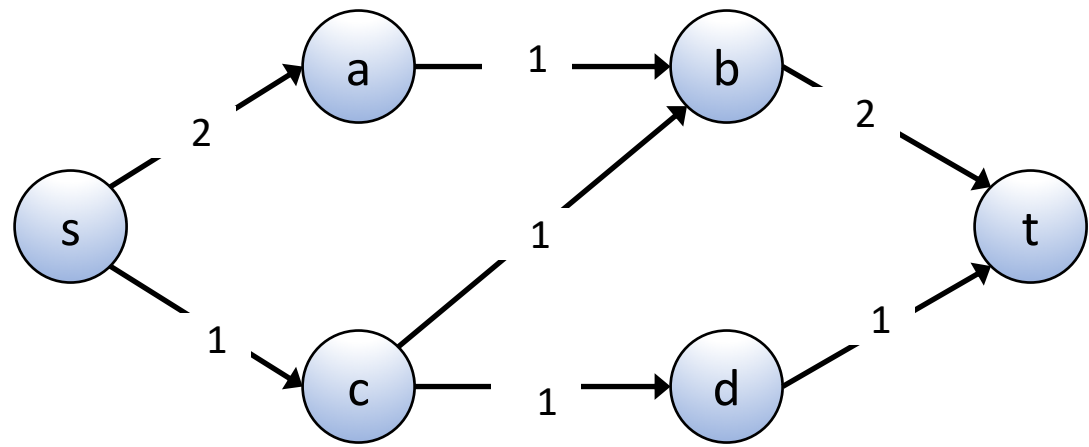
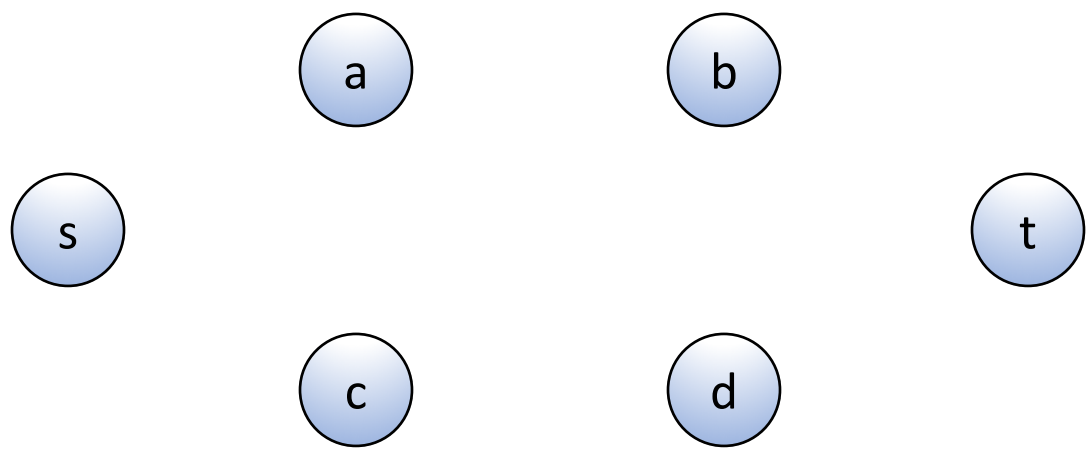
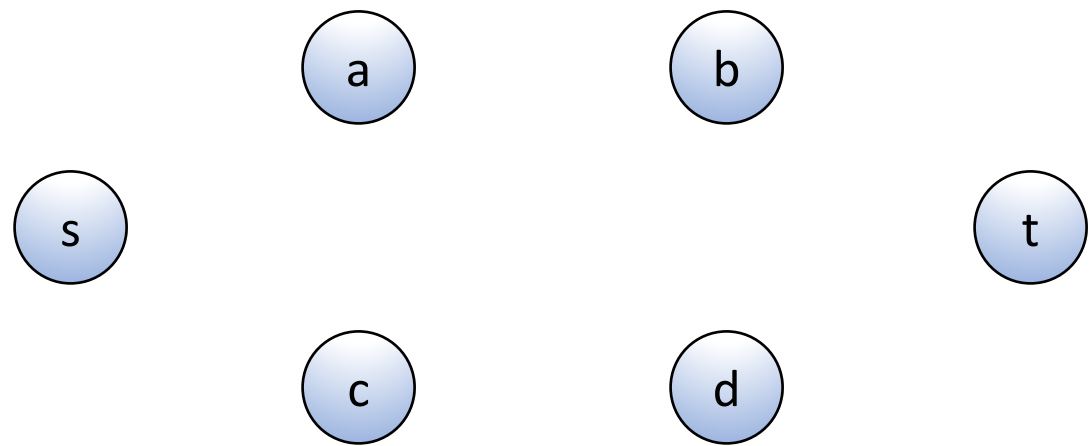
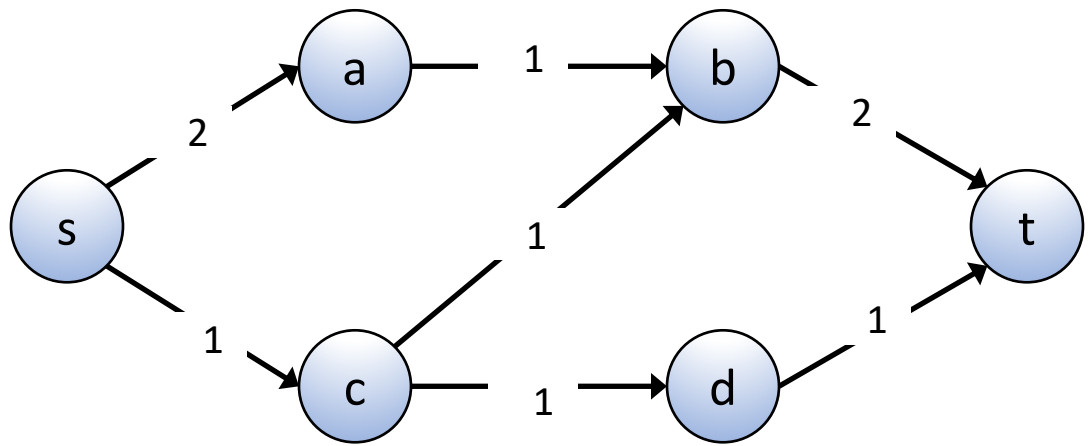
- Start with  $f(e) = 0$  for all edges  $e \in E$
- Find an **augmenting path**  $P$  in the **residual graph**
- Max it out
- Repeat until you get stuck





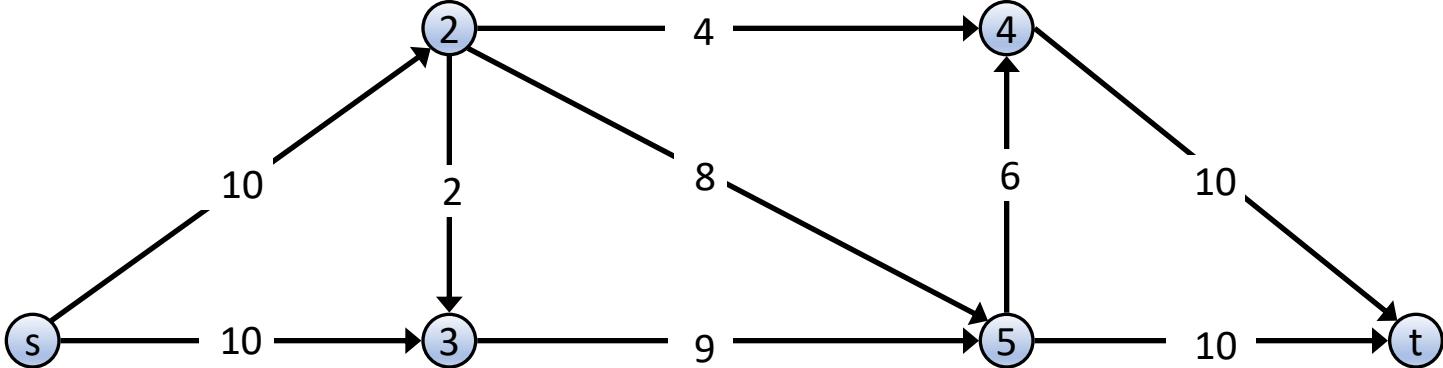
# Ford-Fulkerson Demo

- Run Ford-Fulkerson on the following network



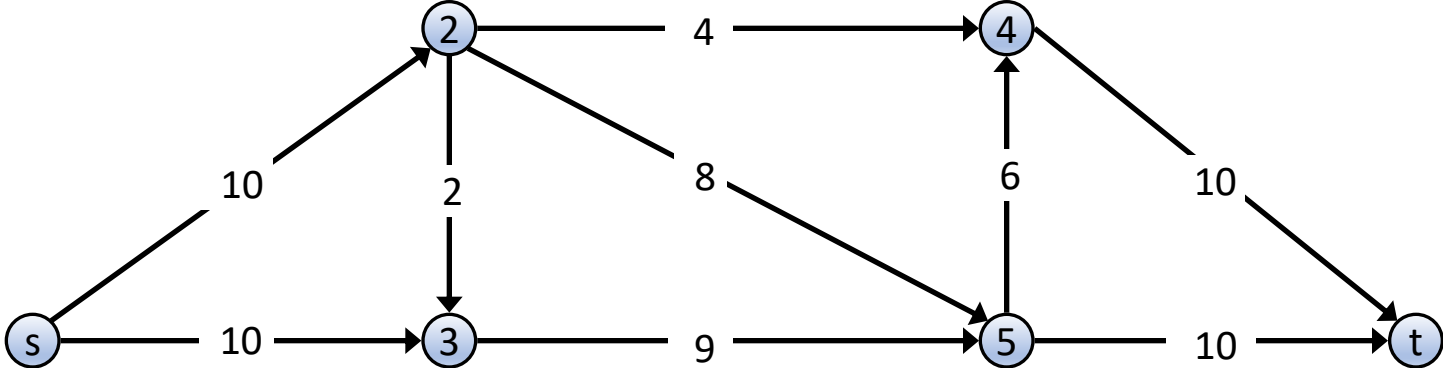
# Ford-Fulkerson Demo

$G$ :



# Ford-Fulkerson Demo

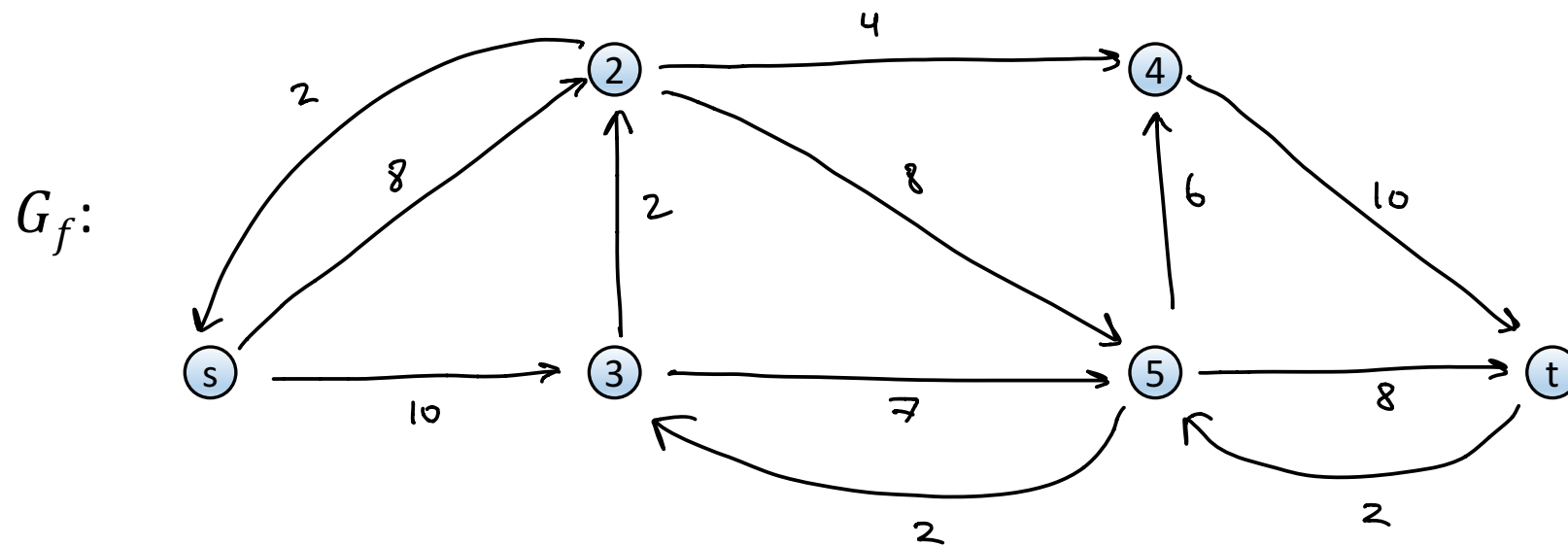
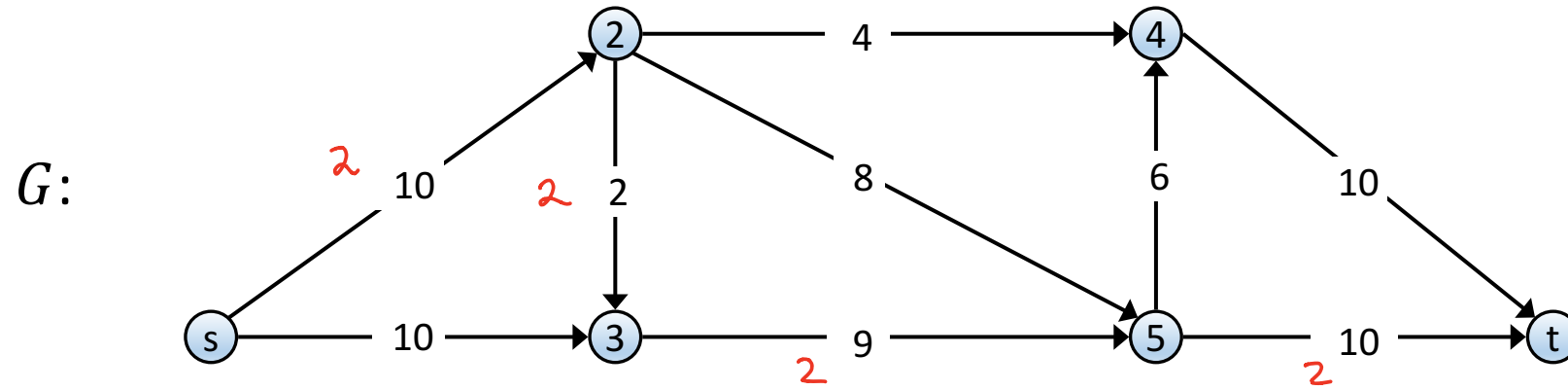
$G$ :



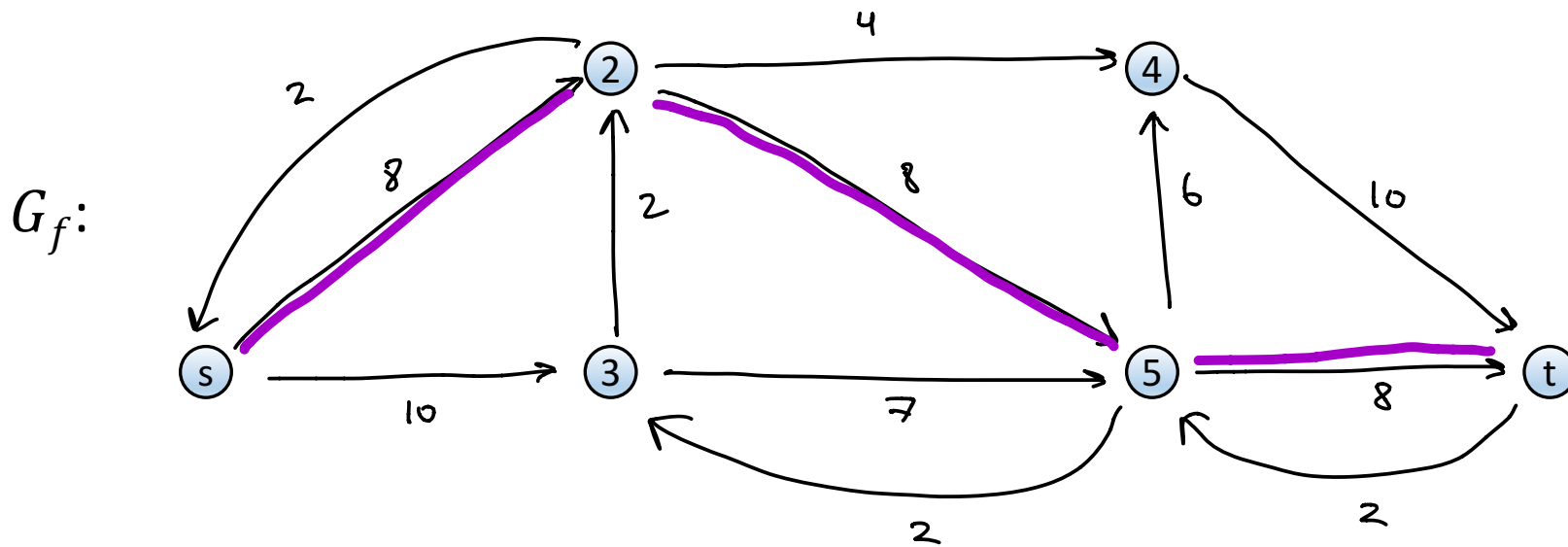
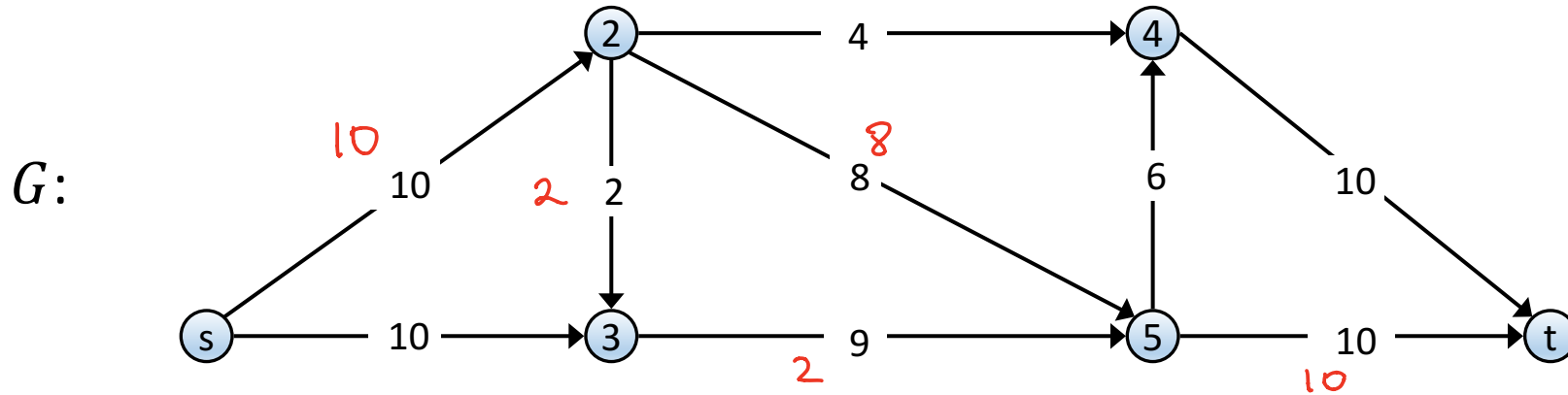
$G_f$ :



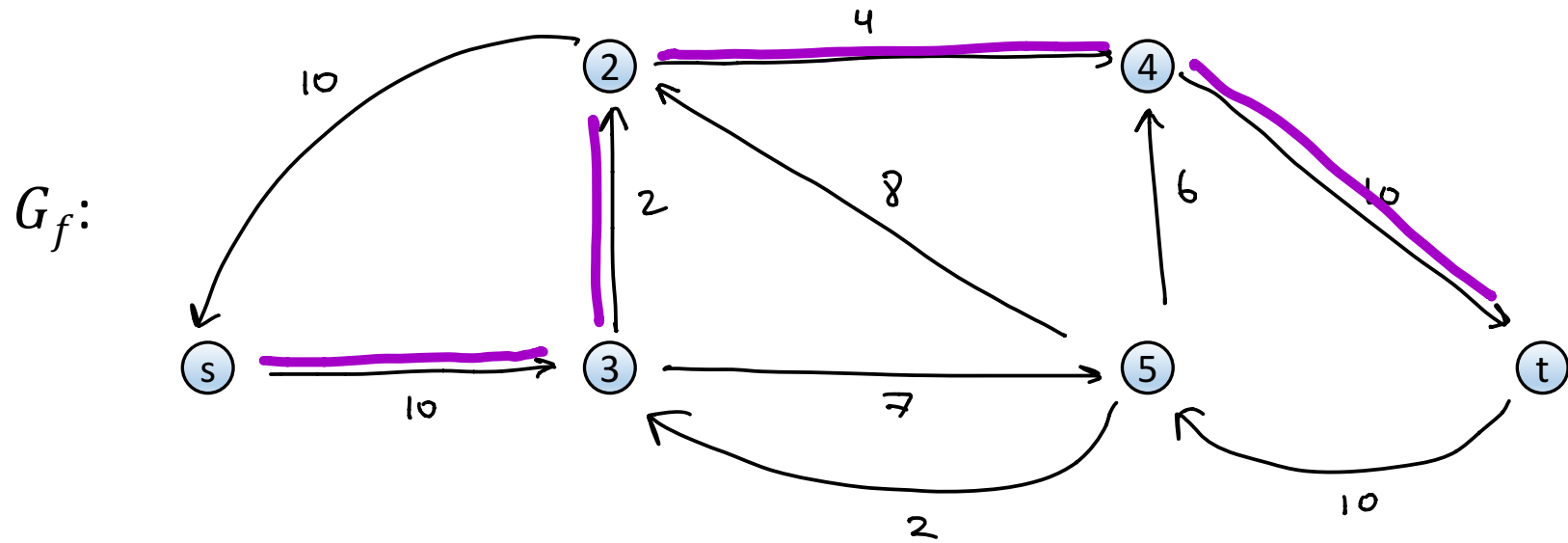
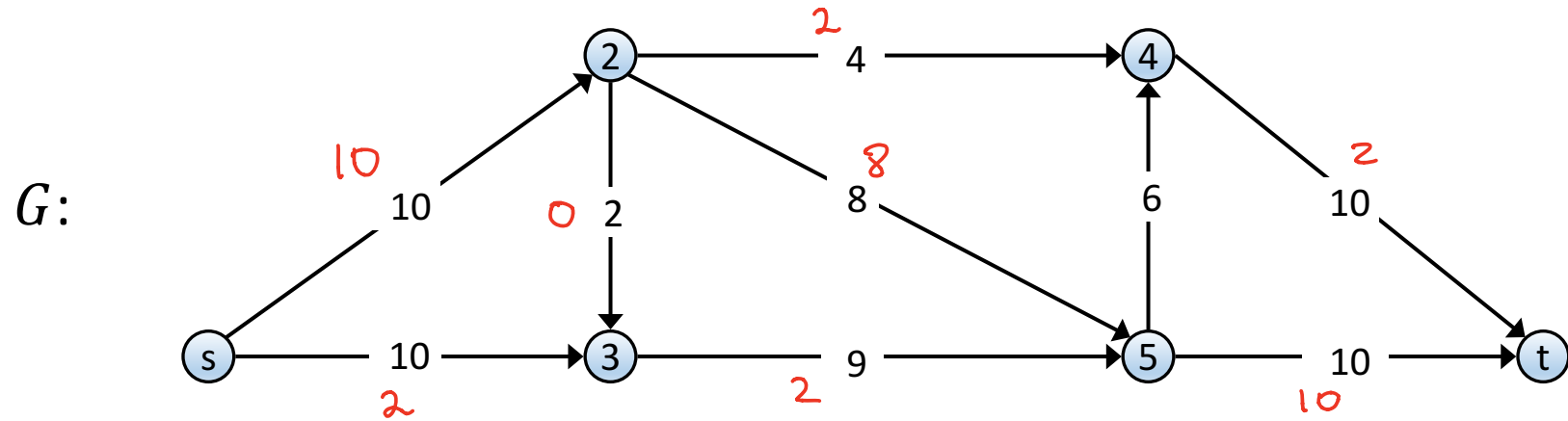
# Ford-Fulkerson Demo



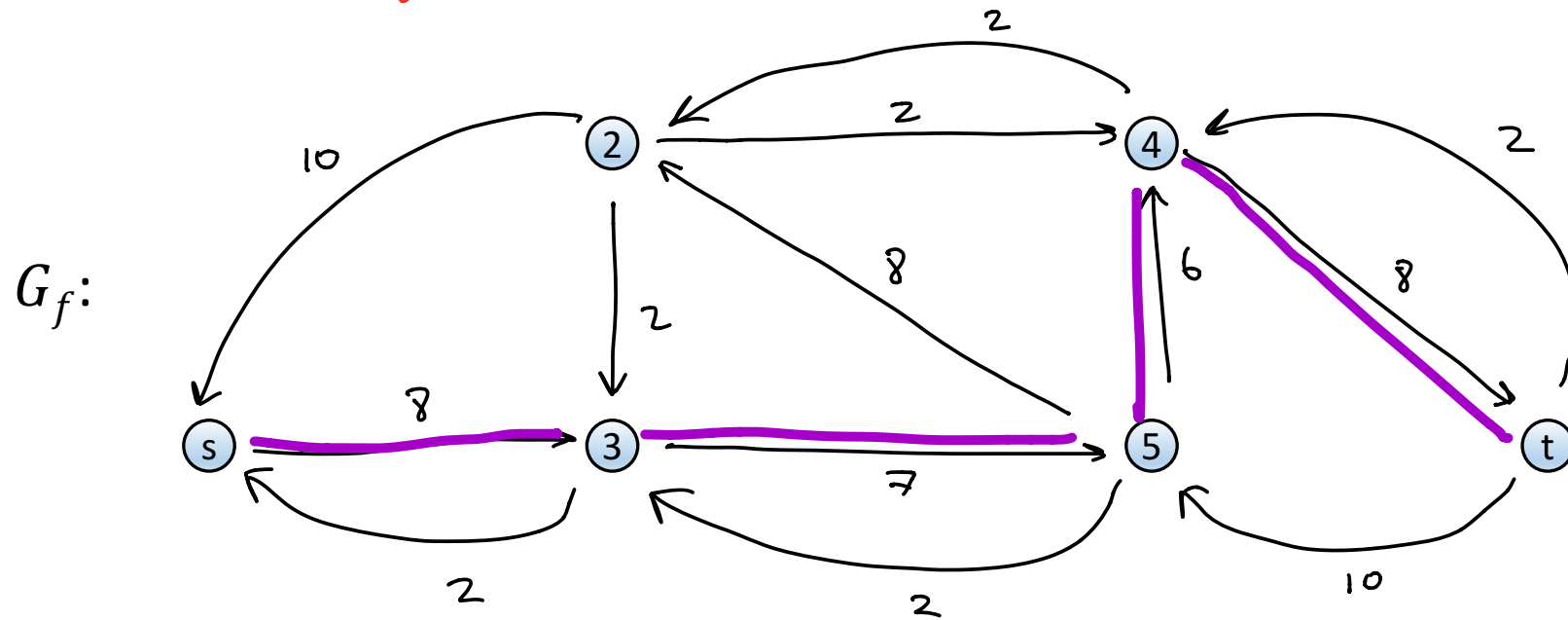
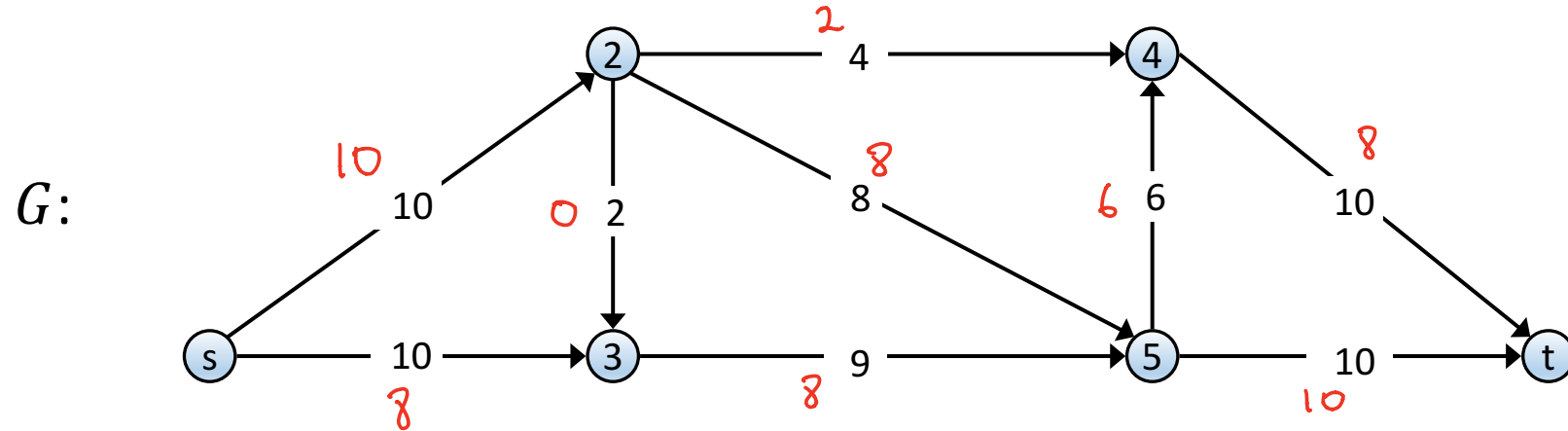
# Ford-Fulkerson Demo



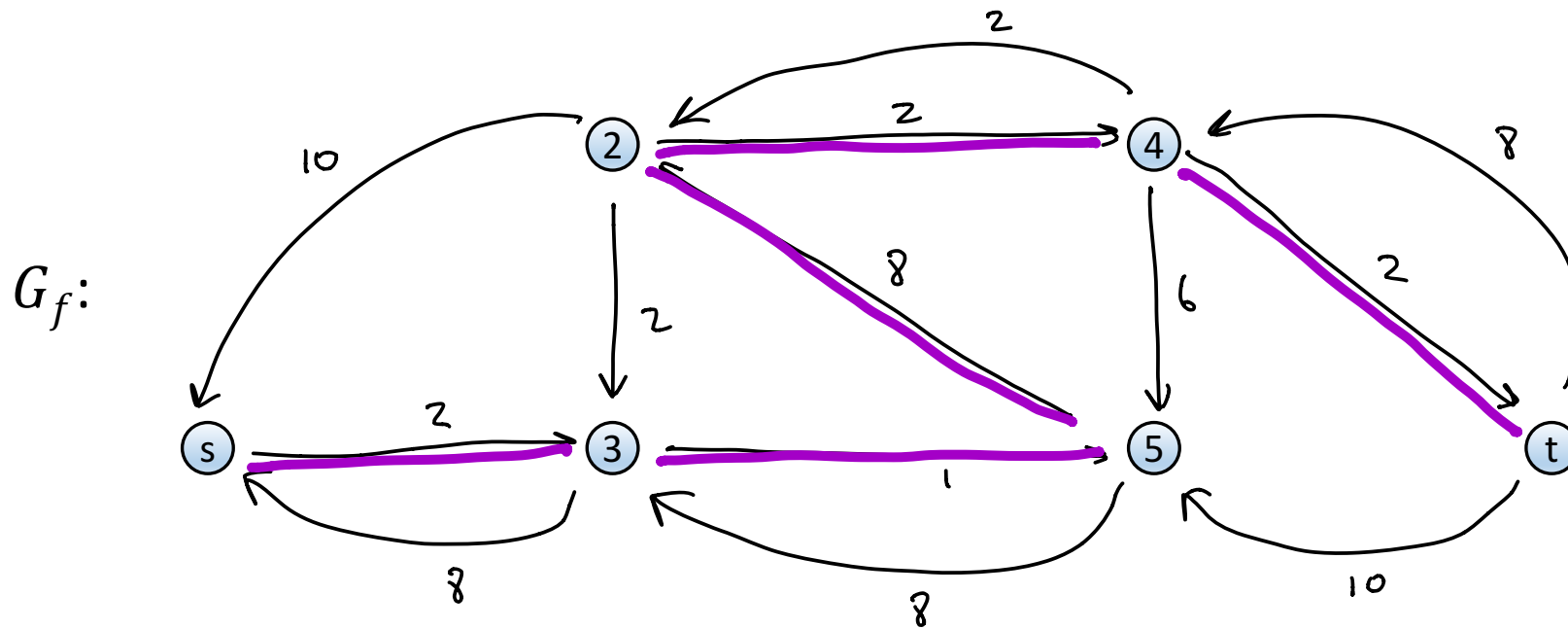
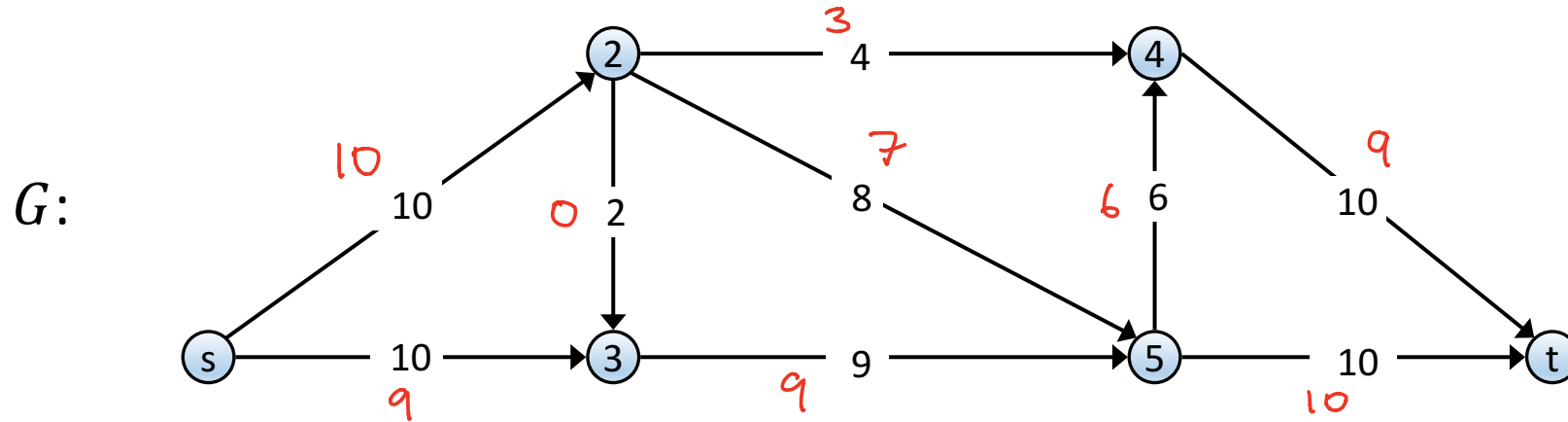
# Ford-Fulkerson Demo



# Ford-Fulkerson Demo



# Ford-Fulkerson Demo





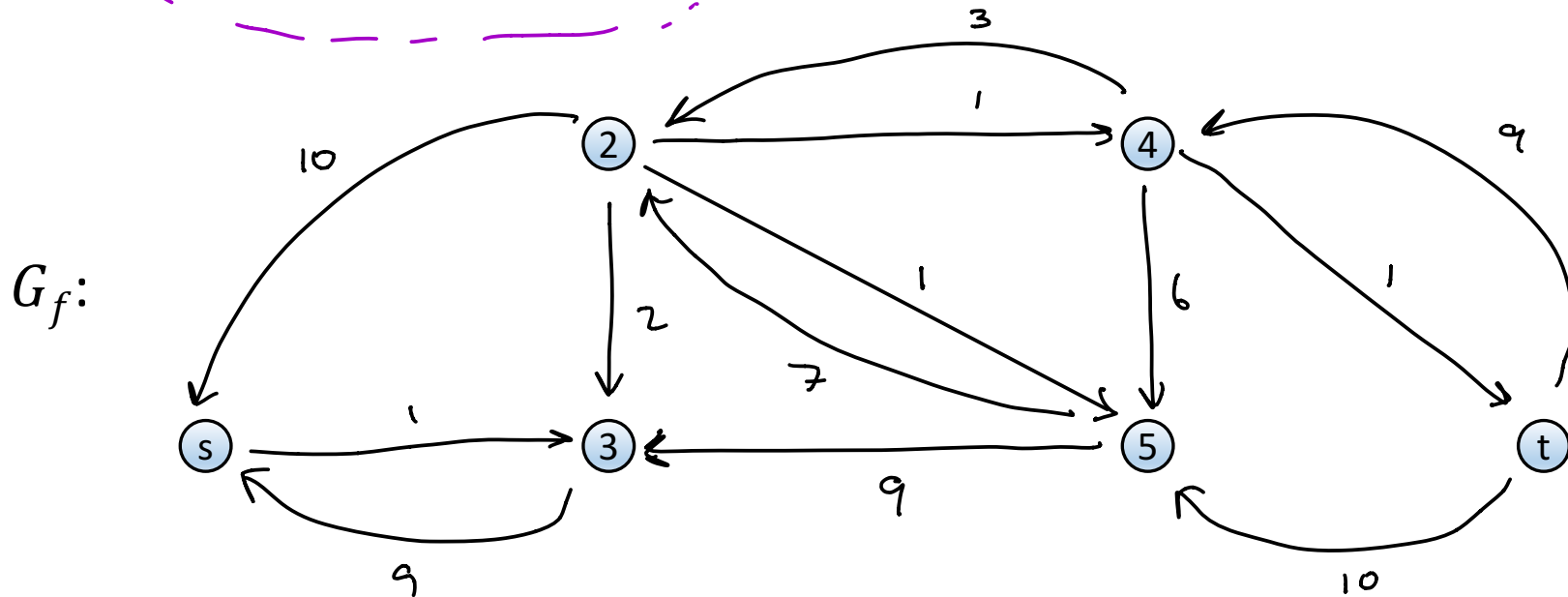
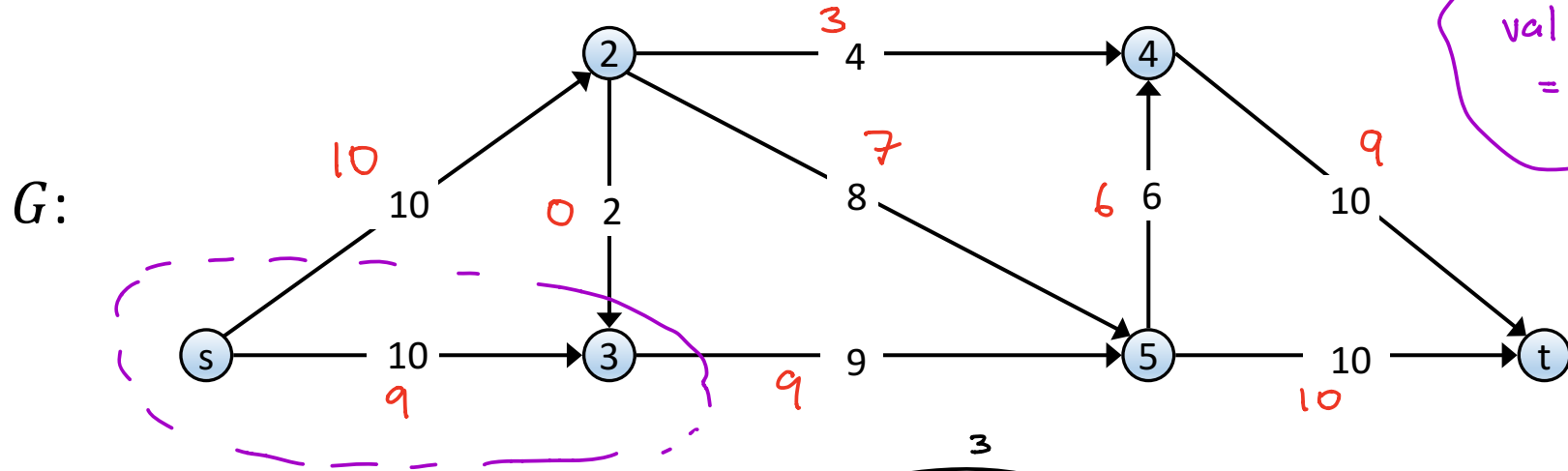
# Ford-Fulkerson Demo

$$A = \{s, 3\}$$

$$B = \{2, 4, 5, t\}$$

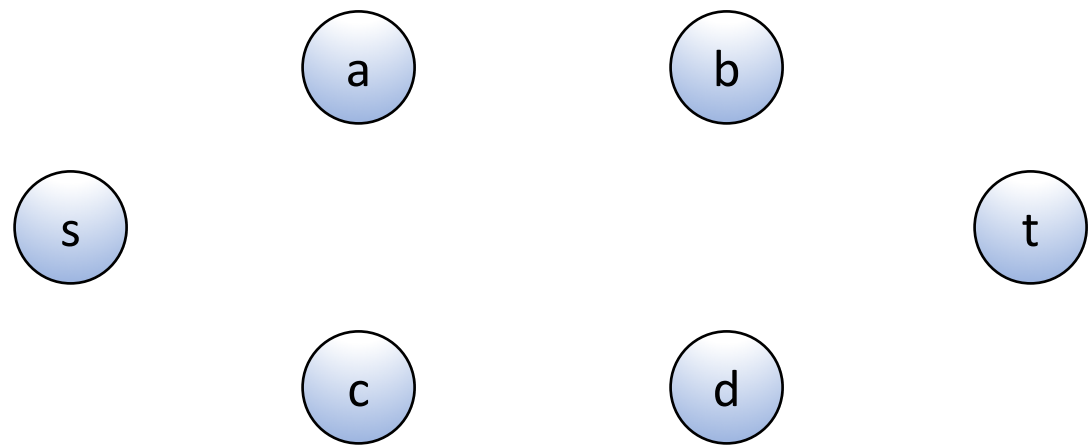
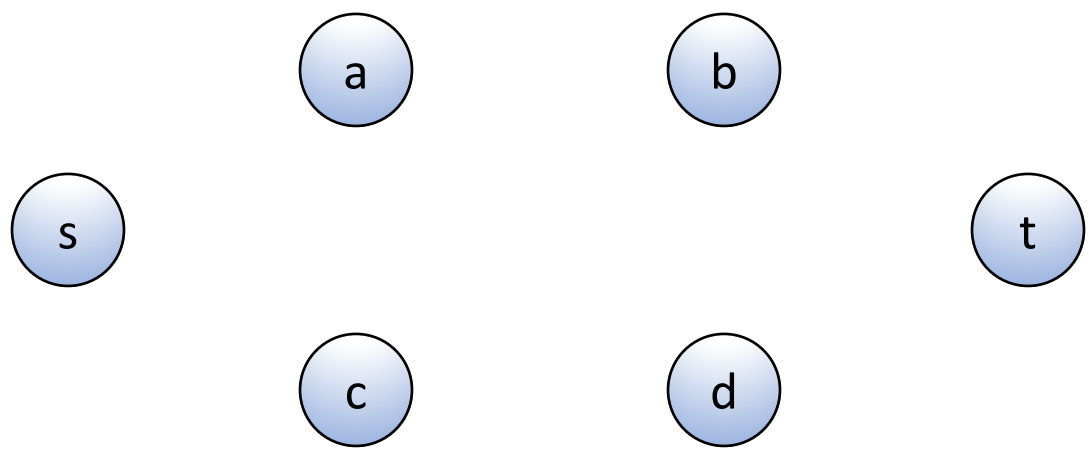
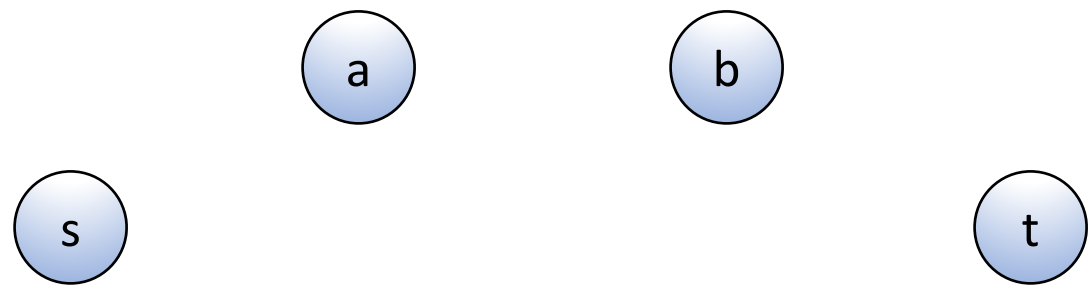
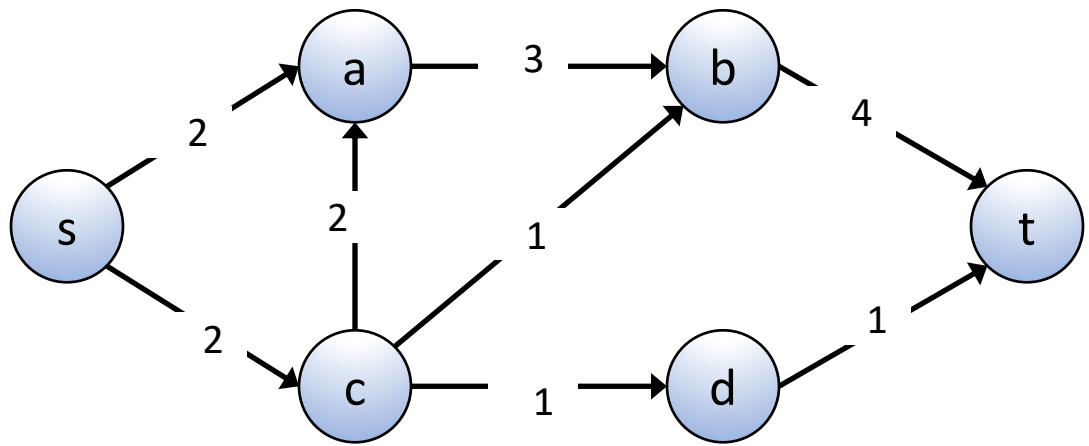
$$\text{cap}(A, B) = 19$$

$$\text{val}(f) = 19$$



# Ford-Fulkerson Demo

- Run Ford-Fulkerson on the following network



# What do we want to prove?

- FF Terminates
- FF finds a maximum s-t flow
- There is always a cut  $(A,B)$  such that  $\text{val}(f) = \text{cap}(A,B)$

# Ford-Fulkerson Algorithm – Run Time

```
FordFulkerson(G, s, t, {c})  
  for e ∈ E: f(e) ← 0  
  Gf is the residual graph  
  
  while (there is an s-t path P in Gf)  
    f ← Augment(Gf, P)  
    update Gf  
  
  return f
```

```
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
```

# Running Time of Ford-Fulkerson

- For **integer capacities**,  $\leq \text{val}(f^*)$  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  $O(m \cdot \text{val}(f^*))$  time
  - $O(mn)$  time if all capacities are  $c_e = 1$
  - $O(mnC_{\max})$  time for any integer capacities
  - Problematic when capacities are large

# 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  $f$ 
  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  $G_f$

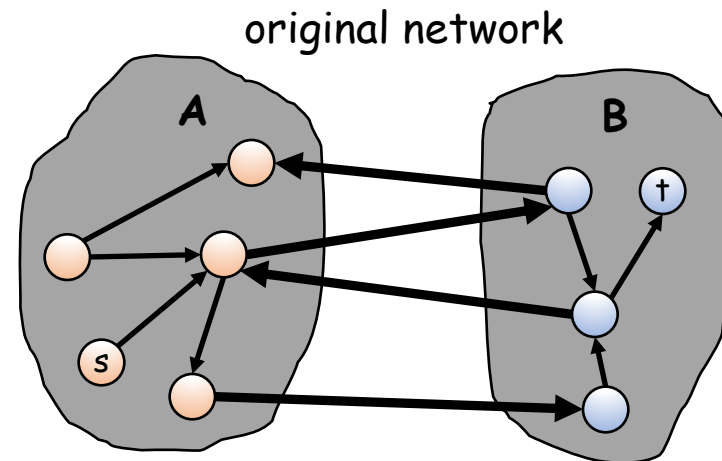
# Optimality of Ford-Fulkerson

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

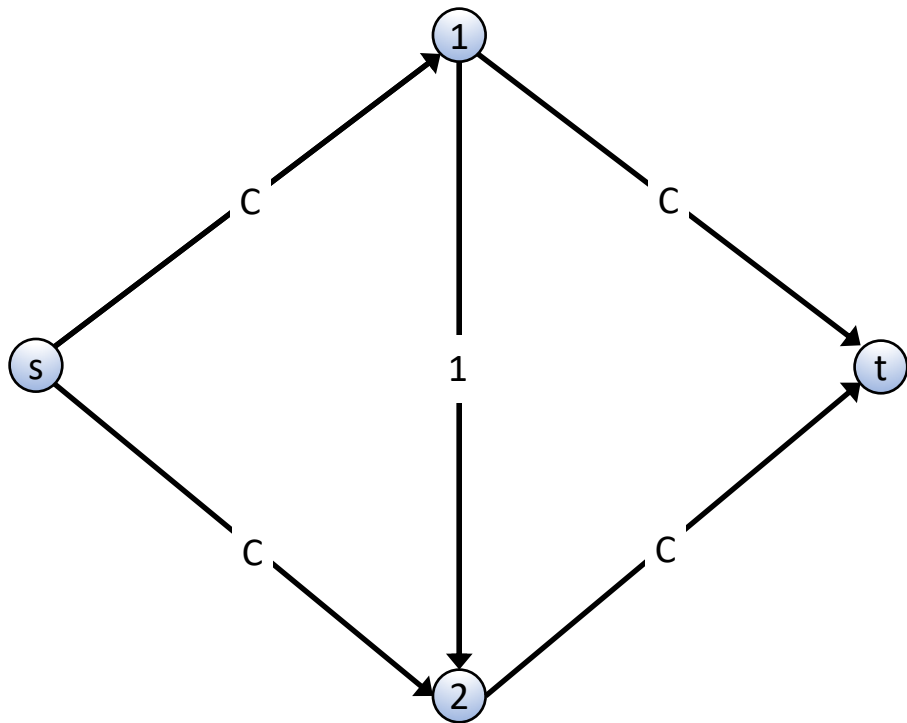


# Summary

- **The Ford-Fulkerson Algorithm solves maximum s-t flow**
  - Running time  $O(m \cdot val(f^*))$  in networks with integer capacities
  - Space  $O(n + m)$
- **MaxFlow-MinCut Duality:** 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

# Ford-Fulkerson Algorithm is slow if augmenting paths are not chosen well

- 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



# Choosing Good Augmenting Paths

- If augmenting paths are chosen arbitrarily:
  - If FF terminates, it outputs a maximum flow
  - Might not terminate, or might require many augmentations
- Augmenting paths can be chosen cleverly
  - Maximum-capacity augmenting path (“fattest augmenting path”)
  - Shortest augmenting paths (“shortest augmenting path”)