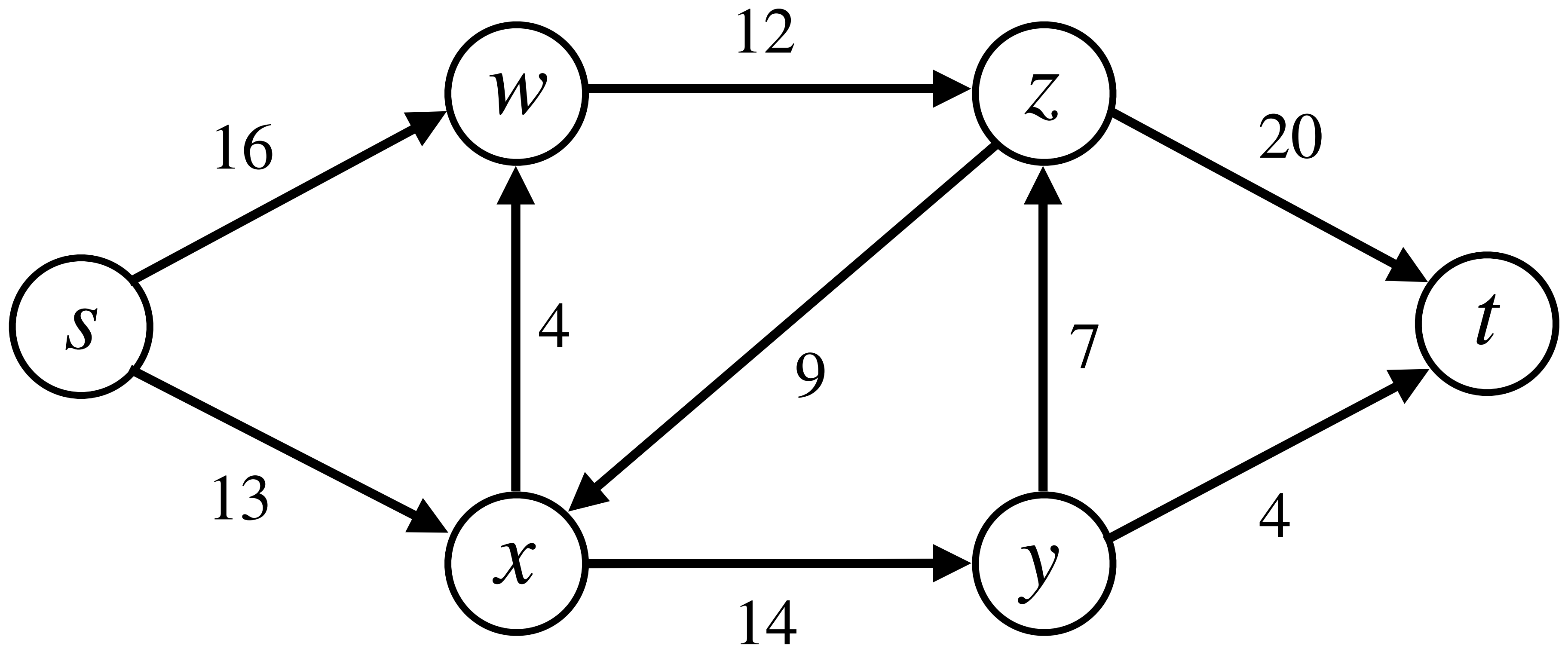
Lecture 17

Flow Networks, Ford-Fulkerson Method

Source: Introduction to Algorithms, CLRS and Kleinberg & Tardos

Flow Networks

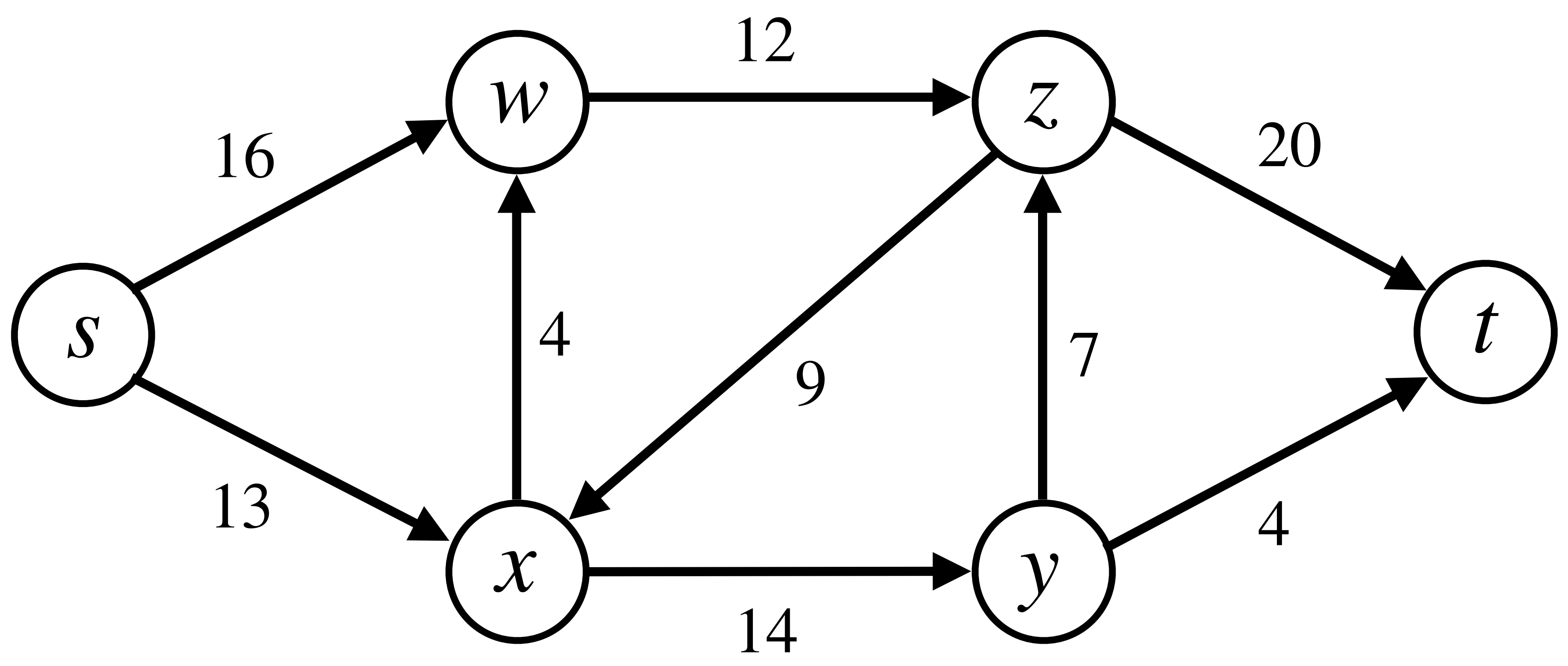
Flow Networks



(a)

Flow Networks

Figure (a) is flow network of a shipping company, where:

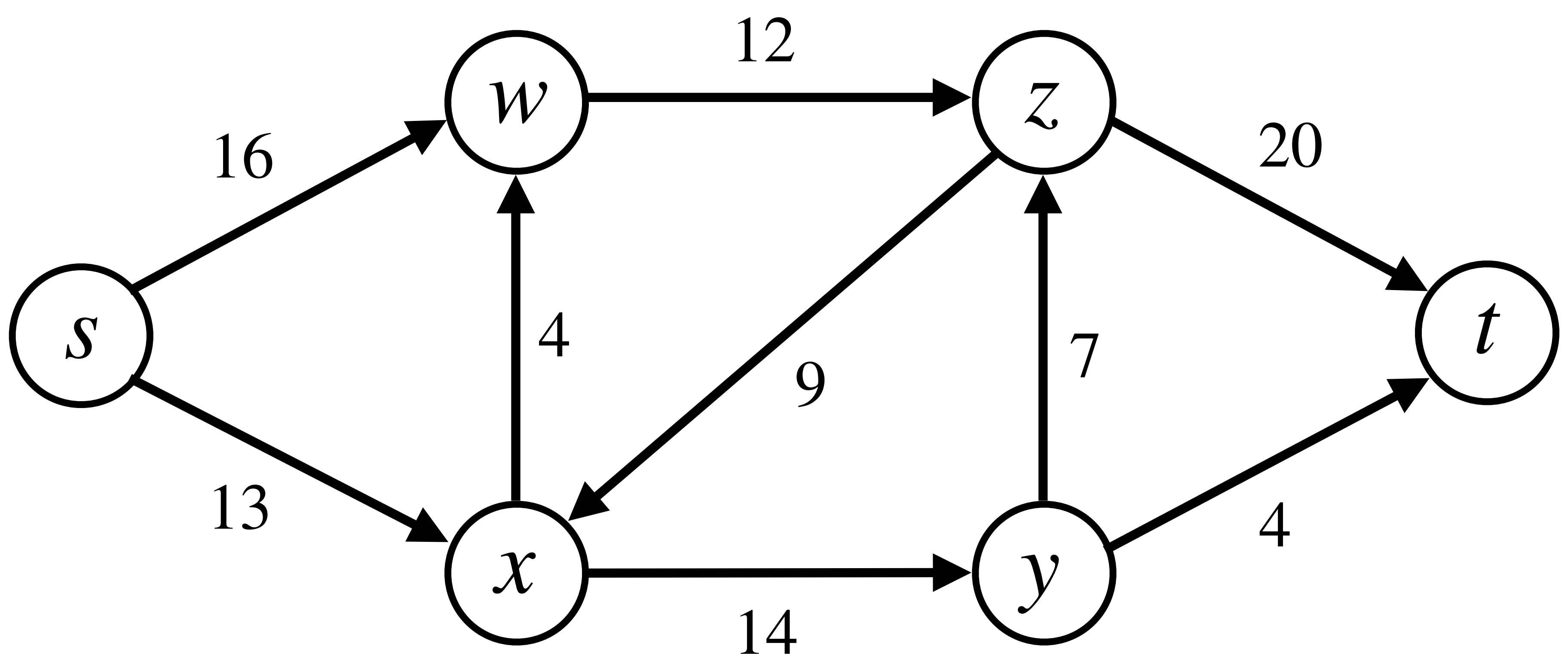


(a)

Flow Networks

Figure (a) is flow network of a shipping company, where:

- Vertices represent **cities**. s & t are the **source** & **sink** cities.

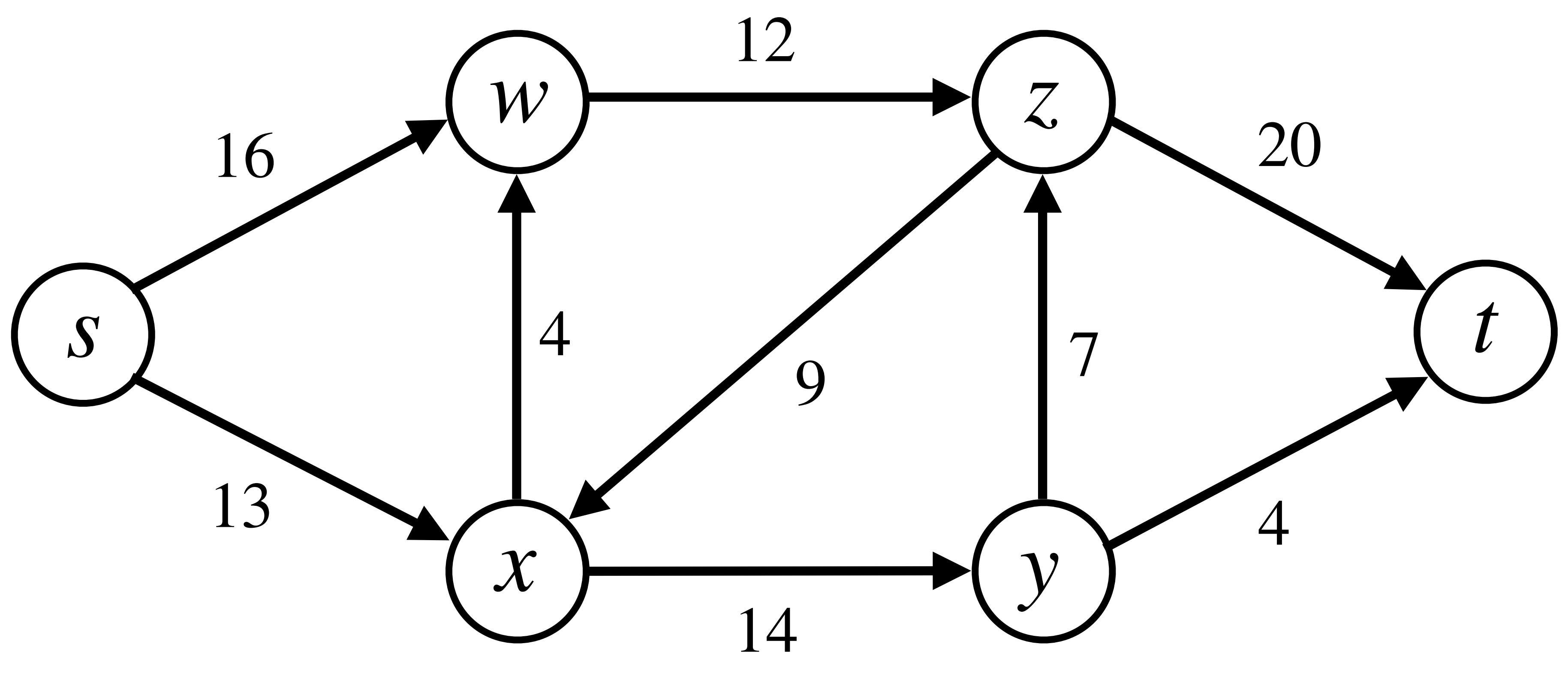


(a)

Flow Networks

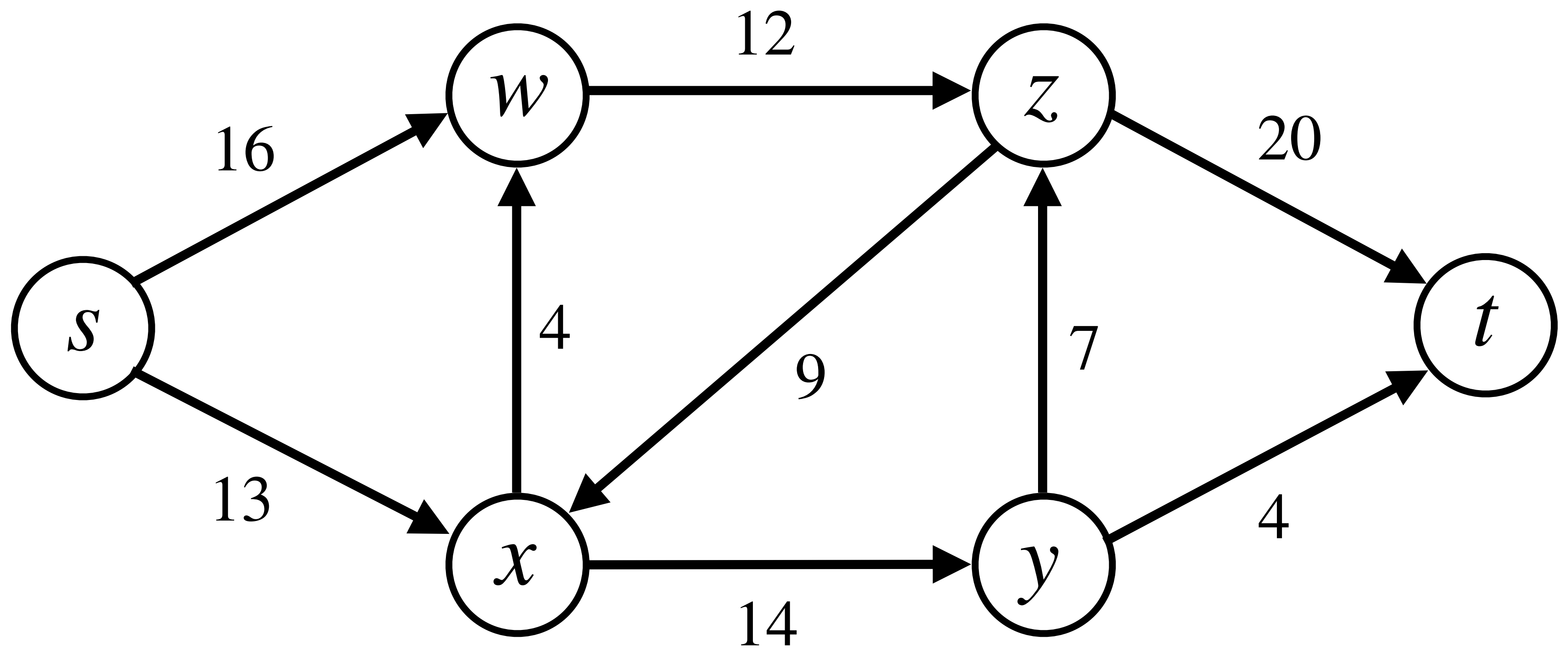
Figure (a) is flow network of a shipping company, where:

- Vertices represent **cities**. s & t are the **source** & **sink** cities.
- The number on any (u, v) edge is the **maximum number of packets** that can go from u to v per day.



(a)

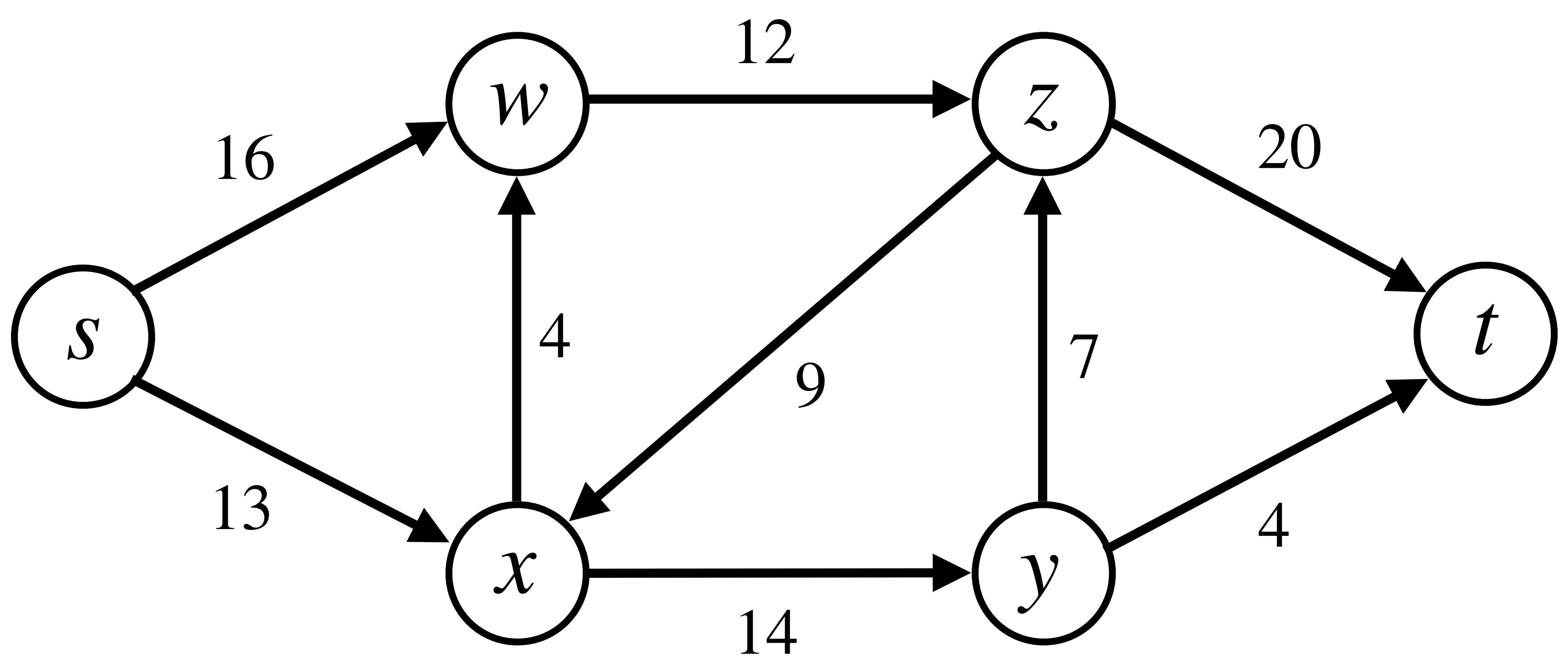
Flow Networks



(a)

Flow Networks

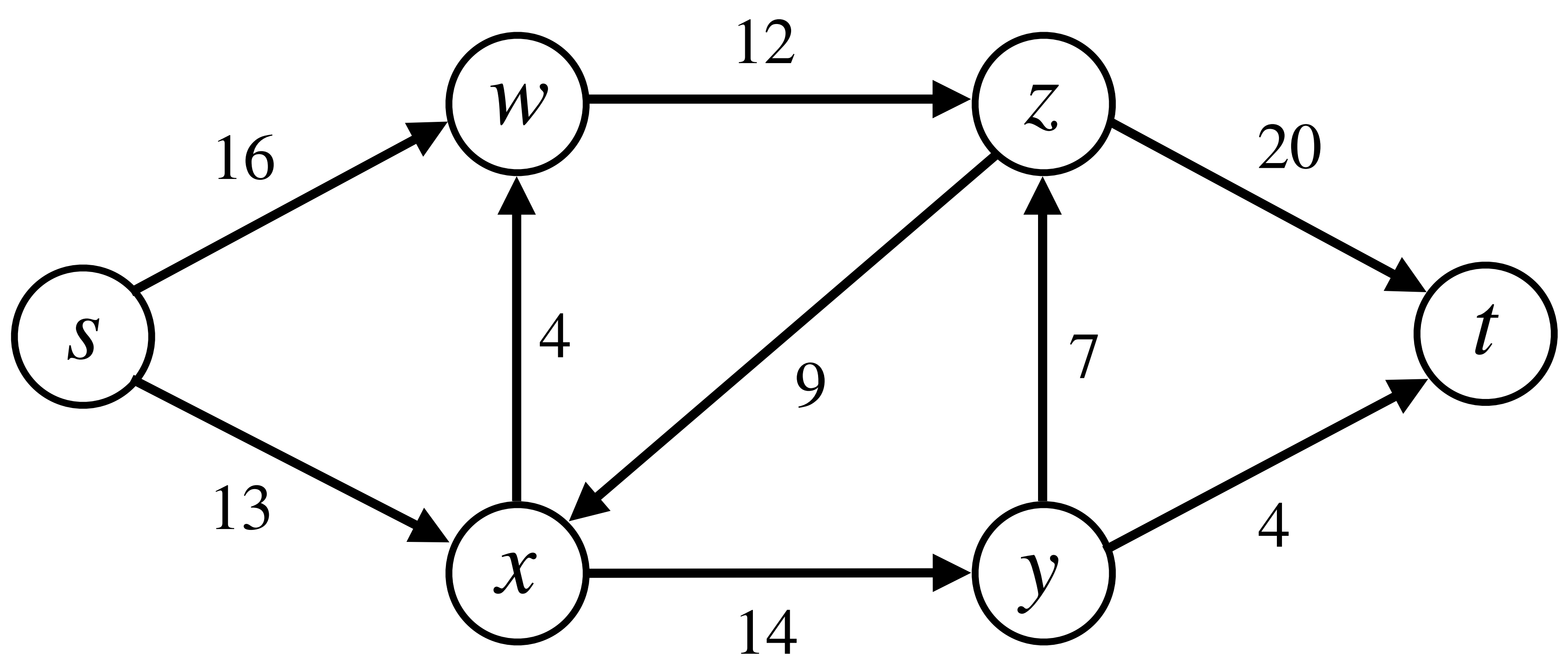
Goal: Find the **maximum number of packets** that can be shipped from s if the packets received and



(a)

Flow Networks

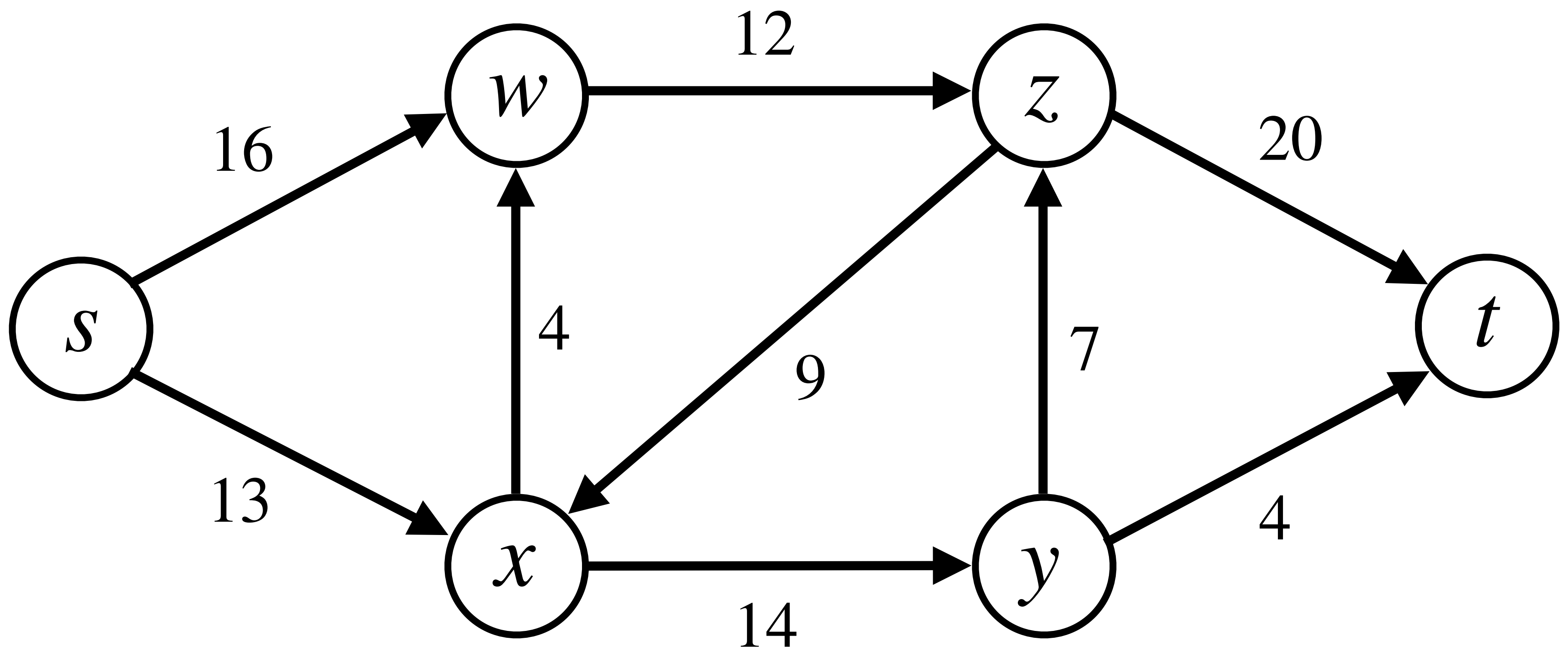
Goal: Find the **maximum number of packets** that can be shipped from s if the packets received and sent by intermediate cities are equal in numbers.



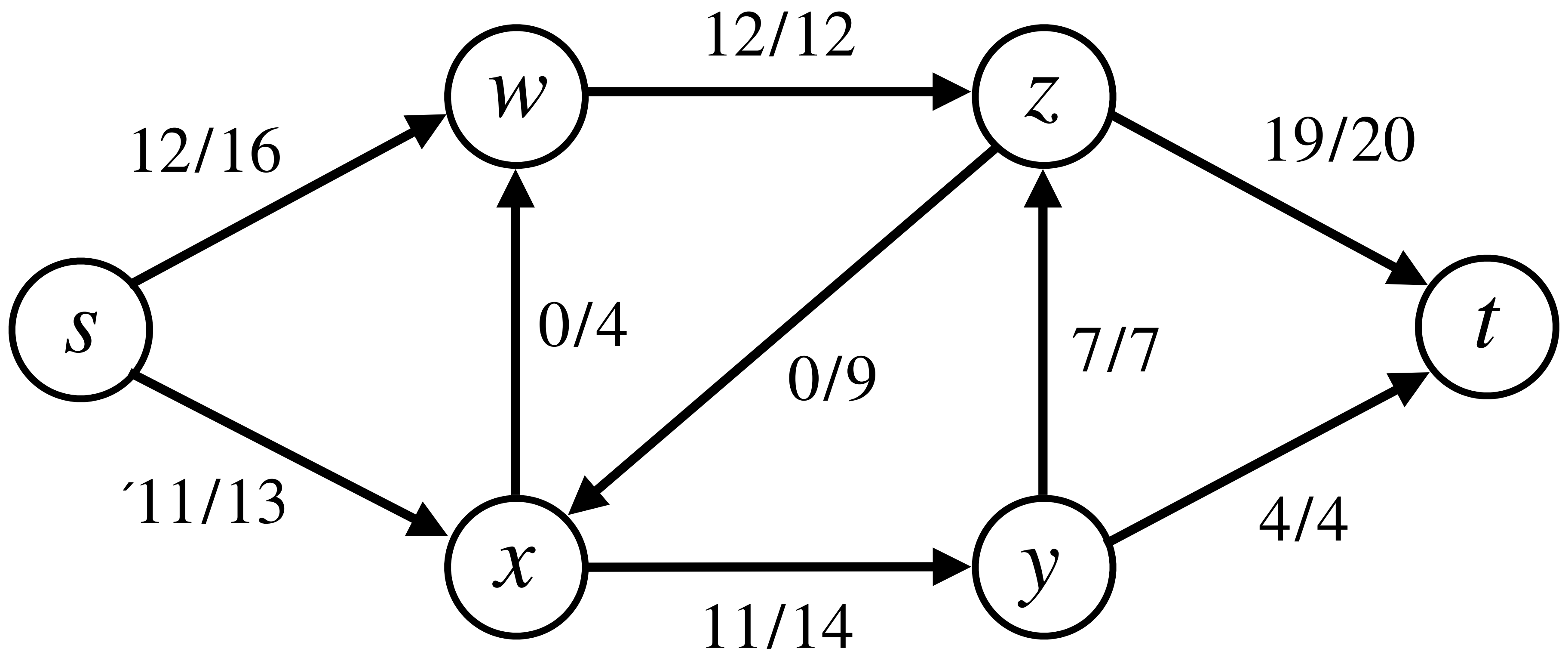
(a)

Flow Networks

Goal: Find the **maximum number of packets** that can be shipped from s if the packets received and sent by intermediate cities are equal in numbers.



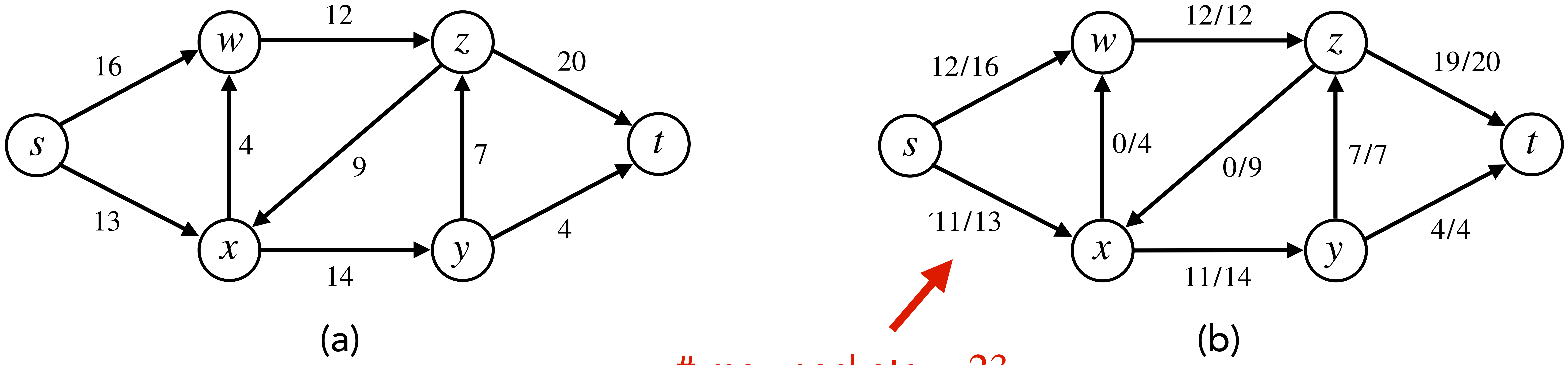
(a)



(b)

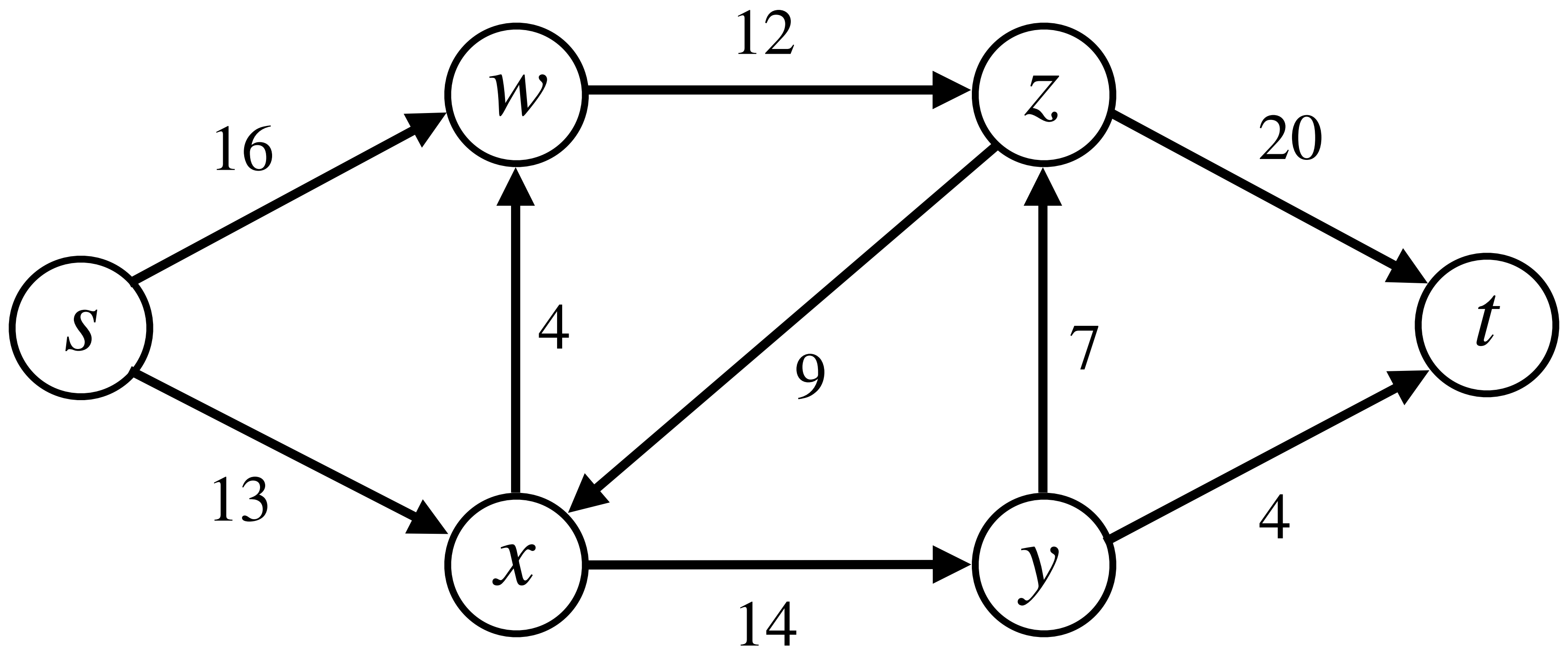
Flow Networks

Goal: Find the **maximum number of packets** that can be shipped from s if the packets received and sent by intermediate cities are equal in numbers.



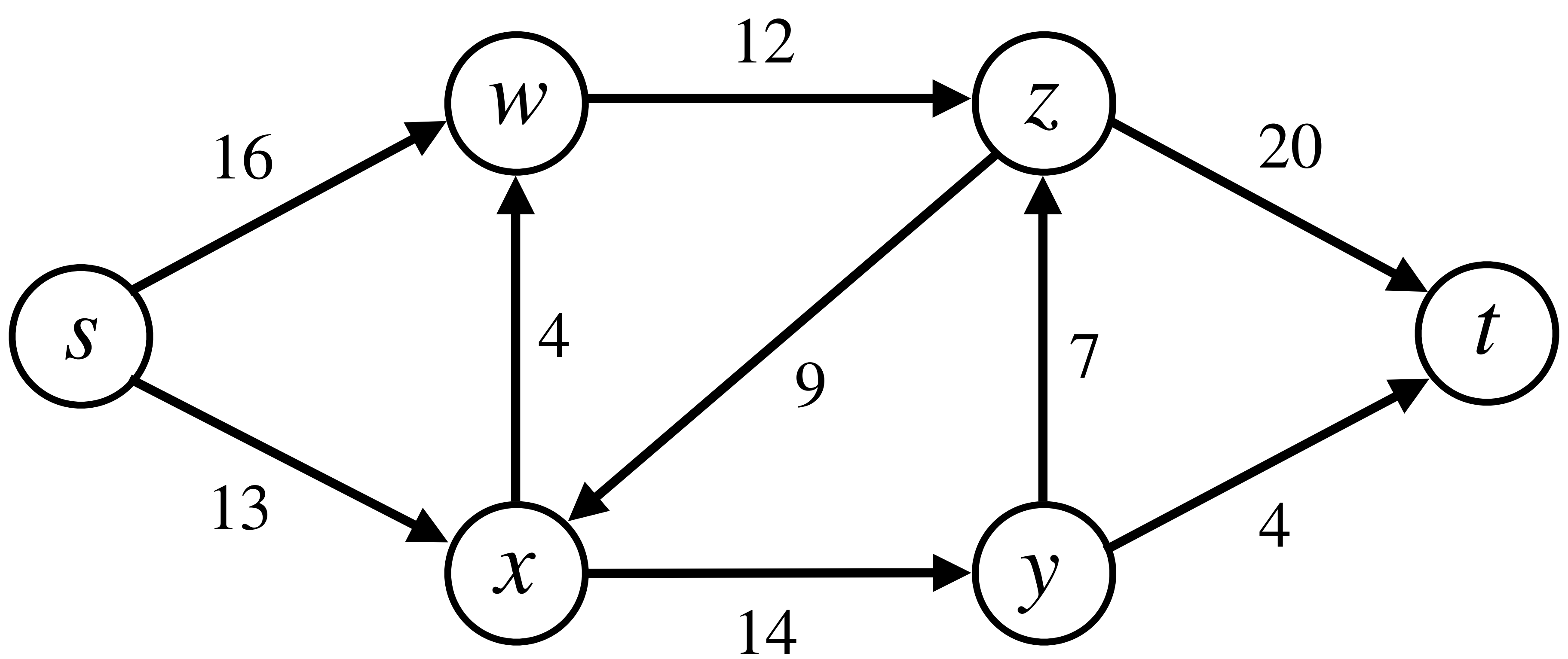
max packets $= 23$

Flow Networks



Flow Networks

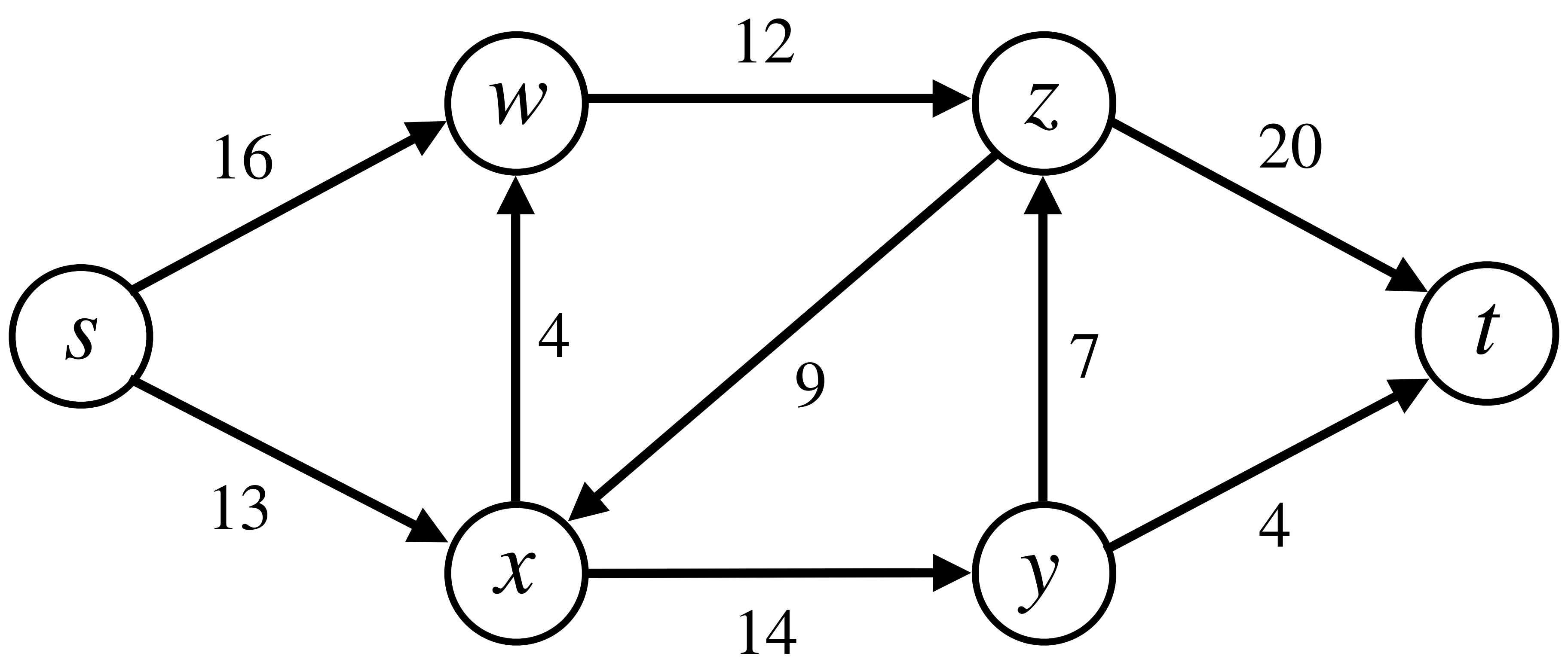
Defn: A flow network $G = (V, E)$ is a directed graph in which:



Flow Networks

Defn: A flow network $G = (V, E)$ is a directed graph in which:

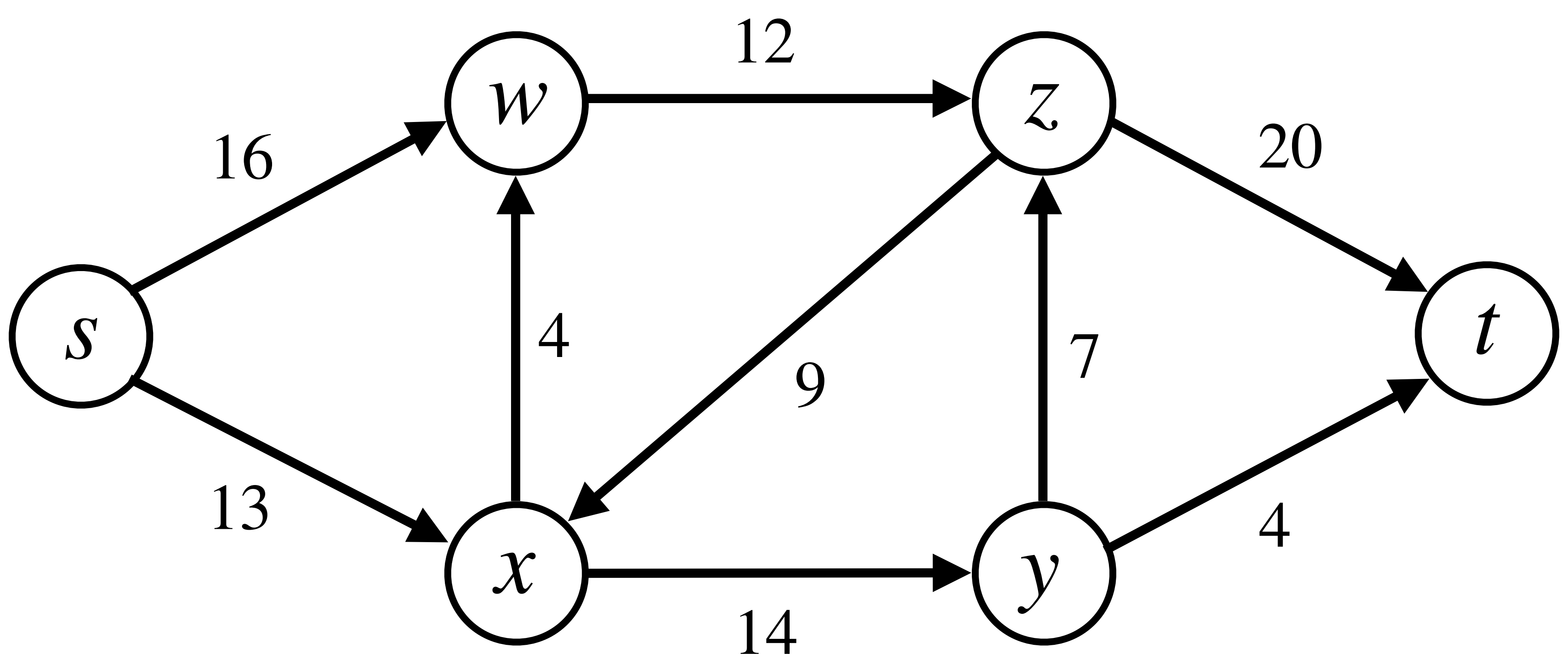
- Each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$.



Flow Networks

Defn: A flow network $G = (V, E)$ is a directed graph in which:

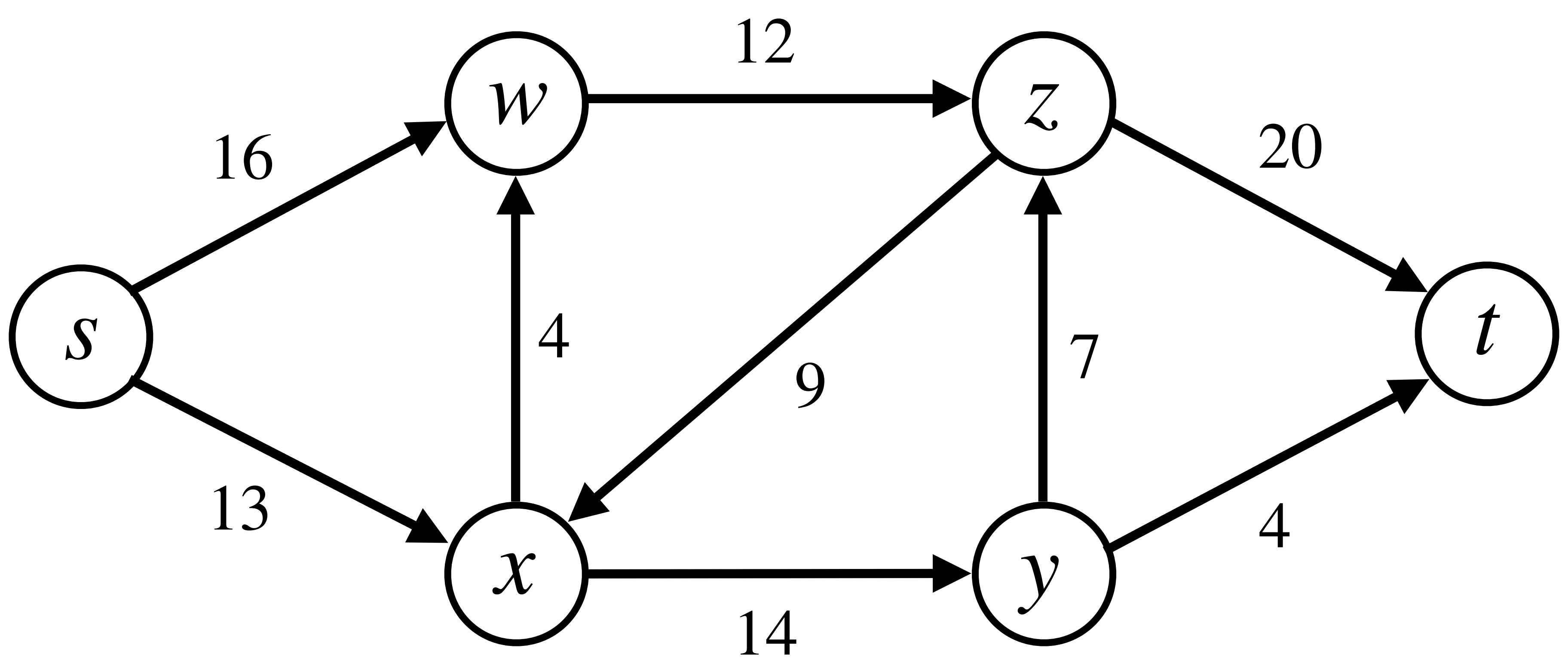
- Each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$.
- If $(u, v) \in E$, then $(v, u) \notin E$.



Flow Networks

Defn: A flow network $G = (V, E)$ is a directed graph in which:

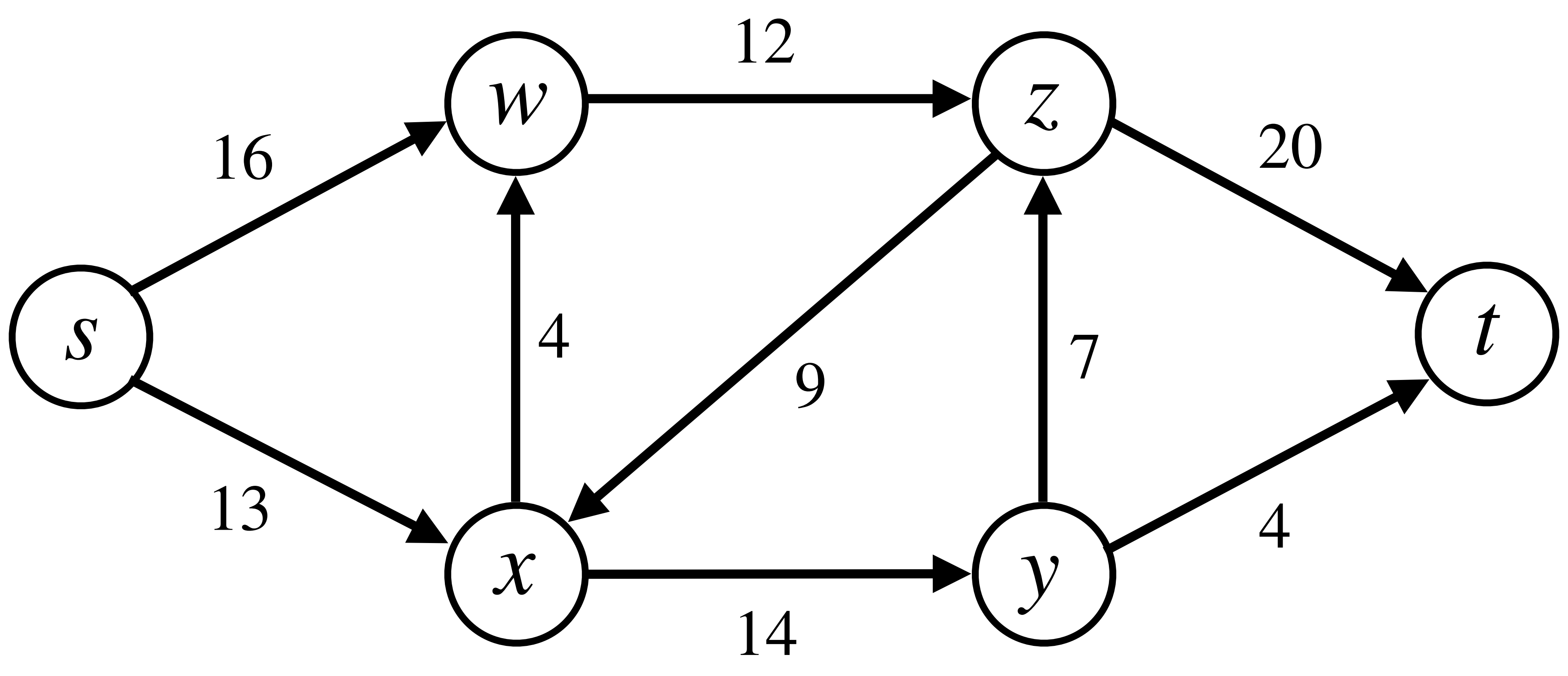
- Each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$.
- If $(u, v) \in E$, then $(v, u) \notin E$. (Reason will become clear soon.)



Flow Networks

Defn: A flow network $G = (V, E)$ is a directed graph in which:

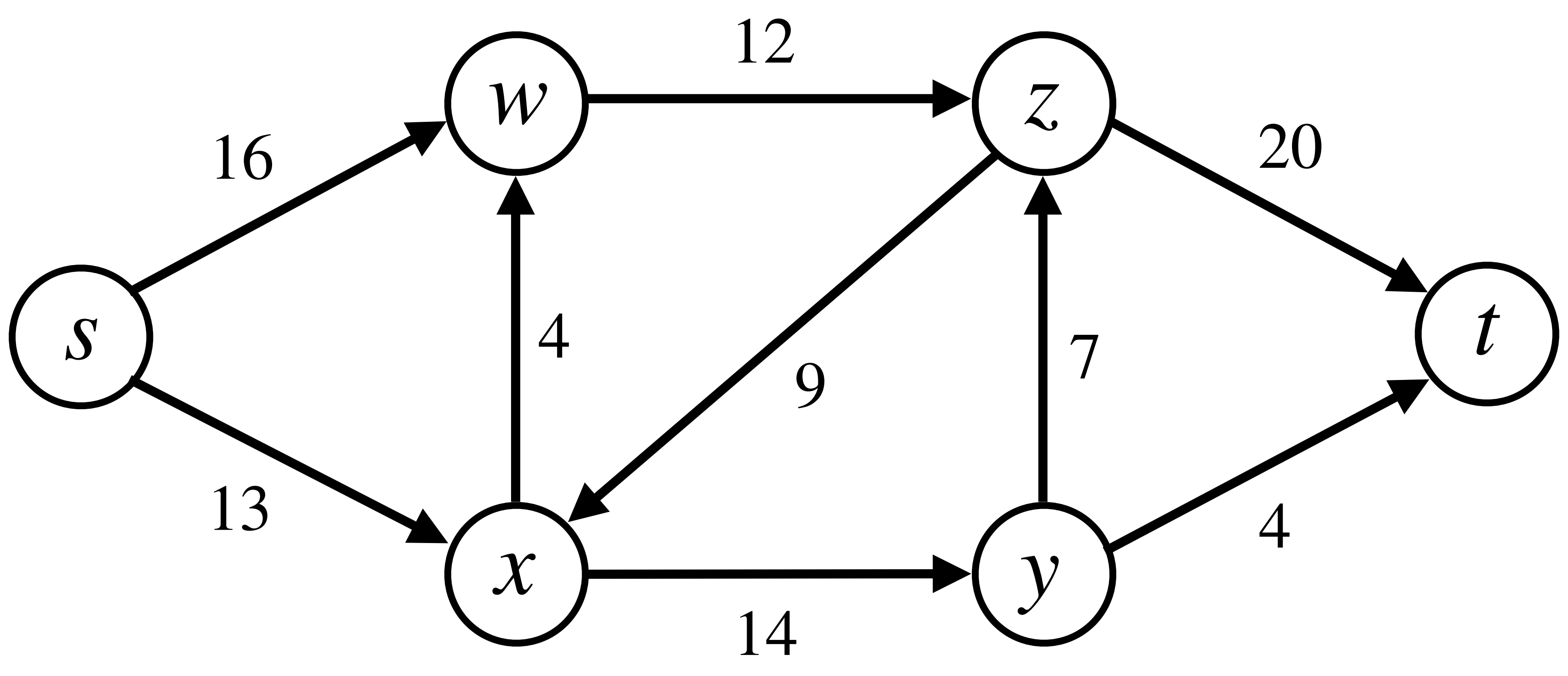
- Each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$.
- If $(u, v) \in E$, then $(v, u) \notin E$. (Reason will become clear soon.)
- If $(u, v) \notin E$, we define $c(u, v) = 0$. No self-loops are present.



Flow Networks

Defn: A flow network $G = (V, E)$ is a directed graph in which:

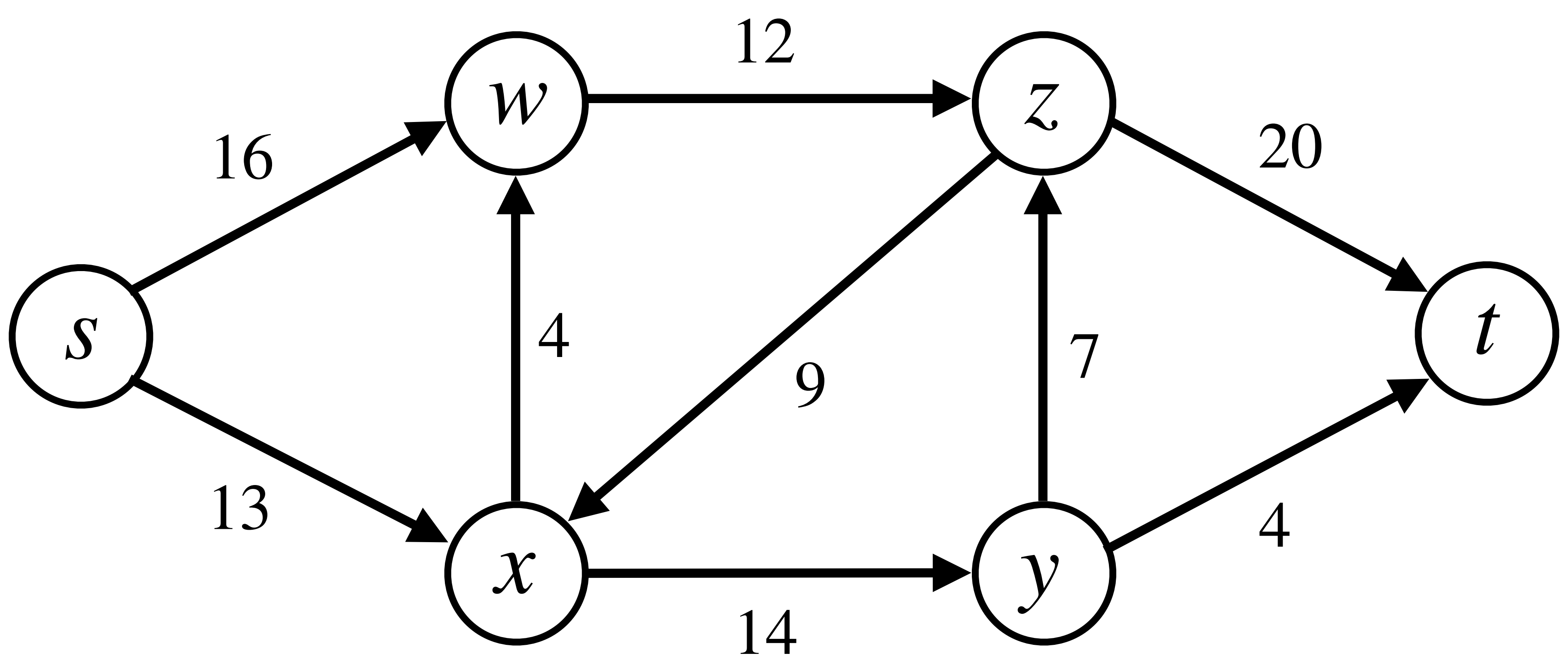
- Each edge $(u, v) \in E$ has a nonnegative **capacity** $c(u, v) \geq 0$.
- If $(u, v) \in E$, then $(v, u) \notin E$. (Reason will become clear soon.)
- If $(u, v) \notin E$, we define $c(u, v) = 0$. No self-loops are present.
- Two distinguished vertices: **source** s (no incoming edges) and **sink** t (no outgoing edges).



Flow Networks

Defn: A flow network $G = (V, E)$ is a directed graph in which:

- Each edge $(u, v) \in E$ has a nonnegative **capacity** $c(u, v) \geq 0$.
- If $(u, v) \in E$, then $(v, u) \notin E$. (Reason will become clear soon.)
- If $(u, v) \notin E$, we define $c(u, v) = 0$. No self-loops are present.
- Two distinguished vertices: **source** s (no incoming edges) and **sink** t (no outgoing edges).
- For every $v \in V$, some $s \rightsquigarrow v \rightsquigarrow t$ path exists. Hence, $|E| \geq |V| - 1$.



Flows

Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

- **Capacity constraint**: For all $u, v \in V$,

Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

- **Capacity constraint**: For all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$.

Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

- **Capacity constraint**: For all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$.
- **Flow conservation:** For all $u \in V \backslash \{s, t\}$,

Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

- **Capacity constraint**: For all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$.
- **Flow conservation:** For all $u \in V \backslash \{s, t\}$, $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.

Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

- **Capacity constraint**: For all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$.
- **Flow conservation:** For all $u \in V \setminus \{s, t\}$, $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.

Total flow in

Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

- **Capacity constraint**: For all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$.
- **Flow conservation:** For all $u \in V \setminus \{s, t\}$, $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.

Total flow in

Total flow out

Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

- **Capacity constraint**: For all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$.
- **Flow conservation:** For all $u \in V \backslash \{s, t\}$, $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.

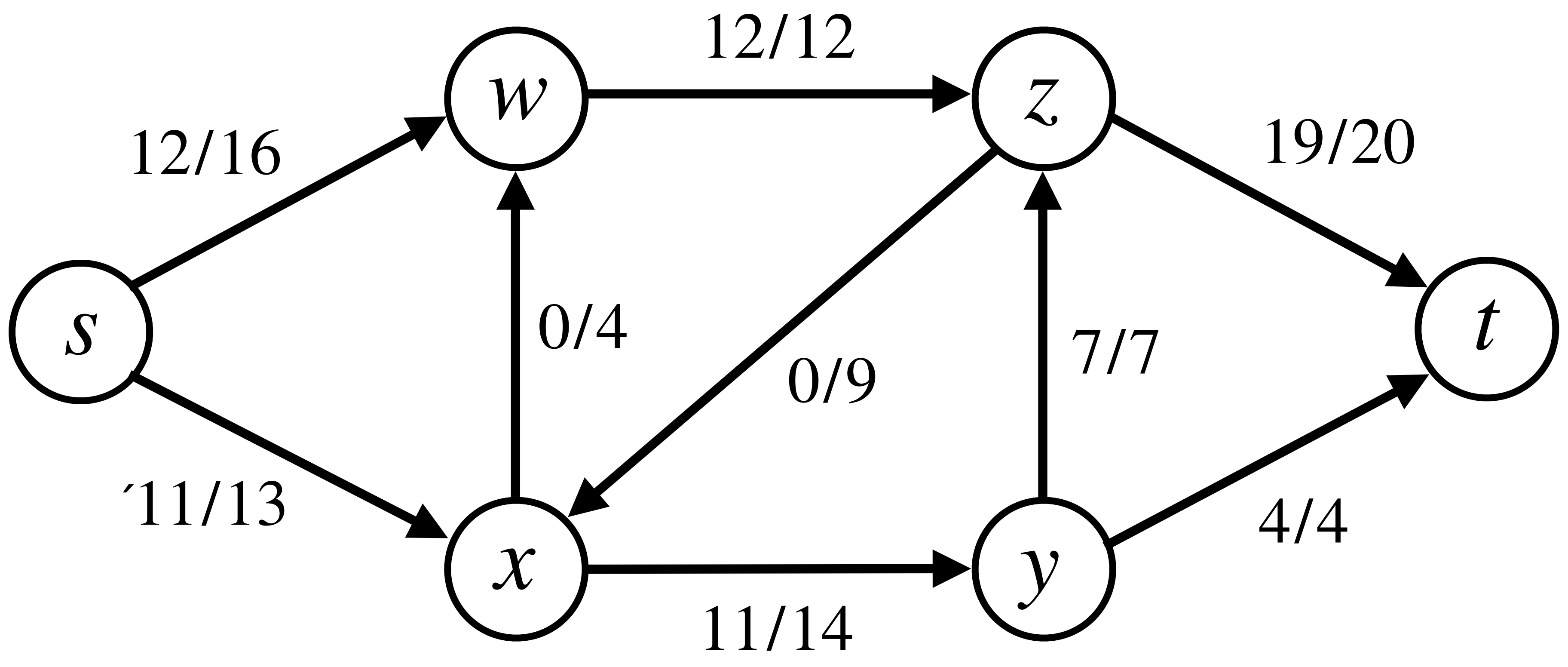
Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

A flow in G is a real-valued function $f : V \times V \rightarrow \mathbb{R}$ that satisfies the following two properties:

- **Capacity constraint**: For all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$.
- **Flow conservation:** For all $u \in V \backslash \{s, t\}$, $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.

Example:



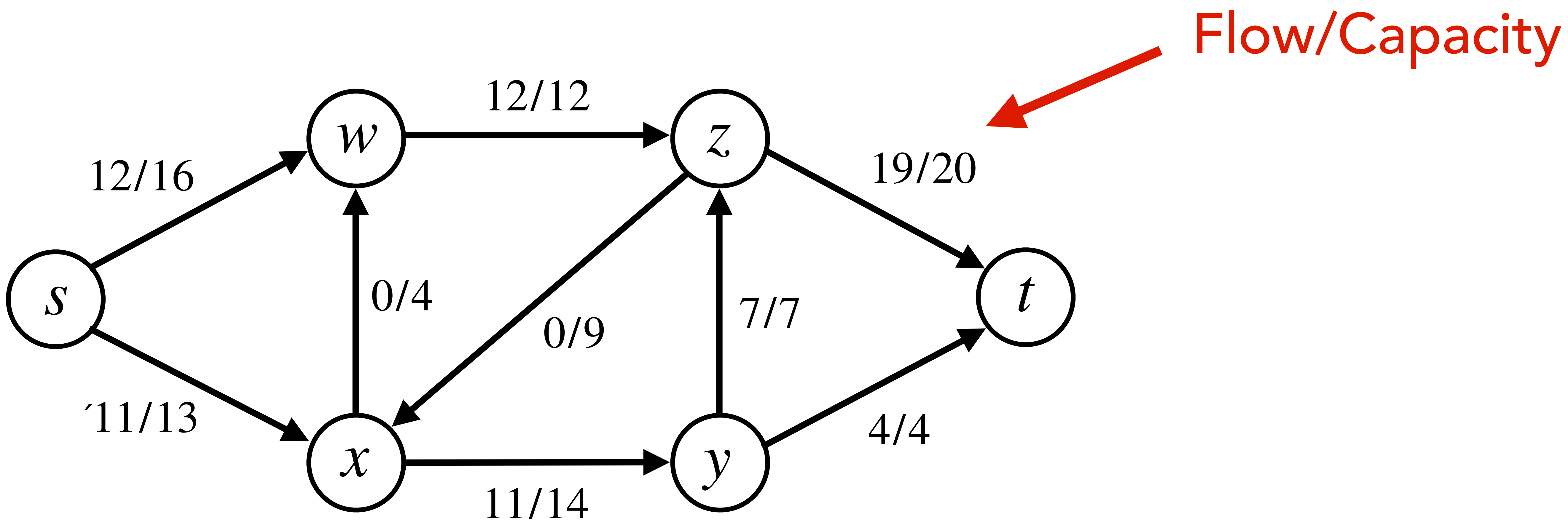
Flows

Defn: Let $G = (V, E)$ be flow network with a capacity function c and source s and sink t.

A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

- **Capacity constraint**: For all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$.
- **Flow conservation:** For all $u \in V \setminus \{s, t\}$, $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.

Example:



Flows

Flows

Defn: Value $|f|$ of flow f is defined as flow out of s, i.e.,

Flows

Defn: Value $|f|$ of flow f is defined as flow out of s, i.e., $|f| = \sum_{v \in V} f(s, v)$.

Flows

Defn: Value $|f|$ of flow f is defined as flow out of s, i.e., $|f| = \sum_{v \in V} f(s, v)$.

Maxflow:

Flows

Defn: Value $|f|$ of flow f is defined as flow out of s, i.e., $|f| = \sum_{v \in V} f(s, v)$.

Maxflow:

Input: A flow network G with source s and sink t.

Flows

Defn: Value $|f|$ of flow f is defined as flow out of s, i.e., $|f| = \sum_{v \in V} f(s, v)$.

Maxflow:

Input: A flow network G with source s and sink t.

Output: Flow of maximum value.

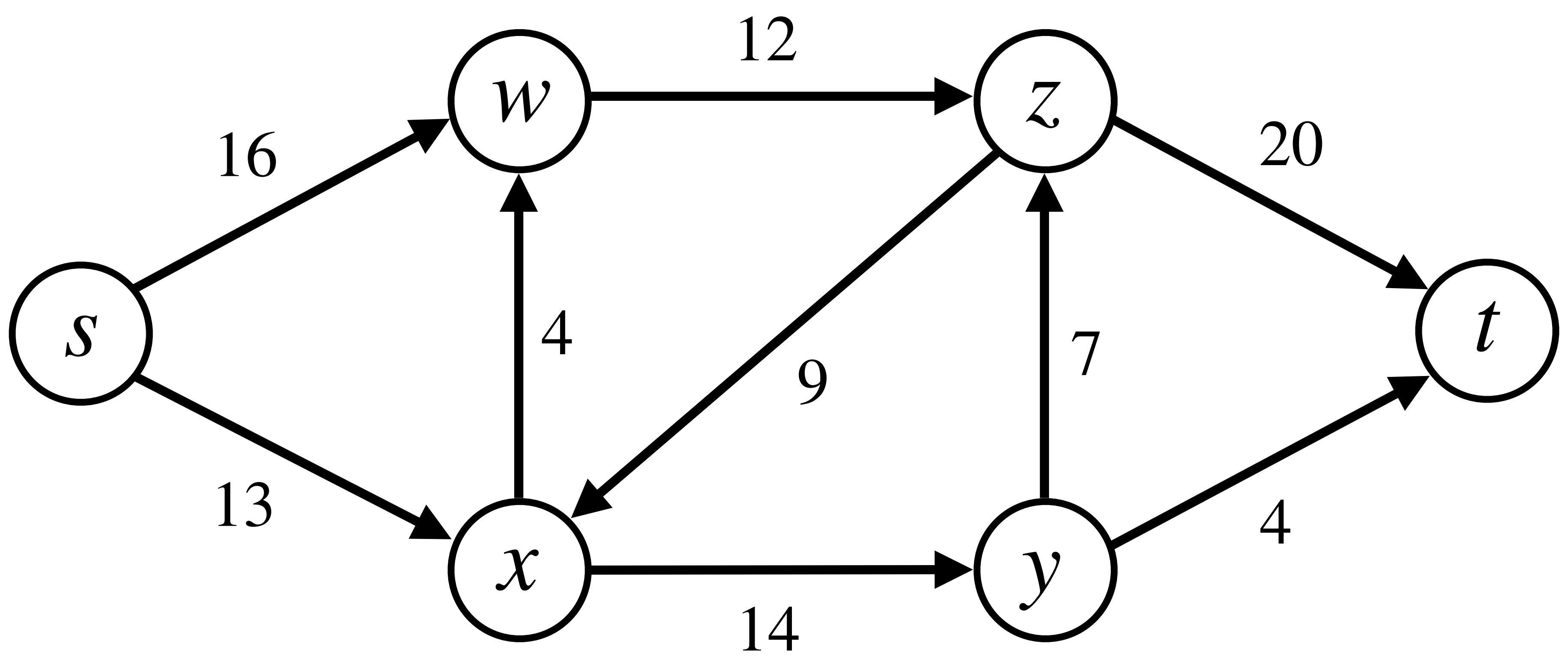
Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

Example:

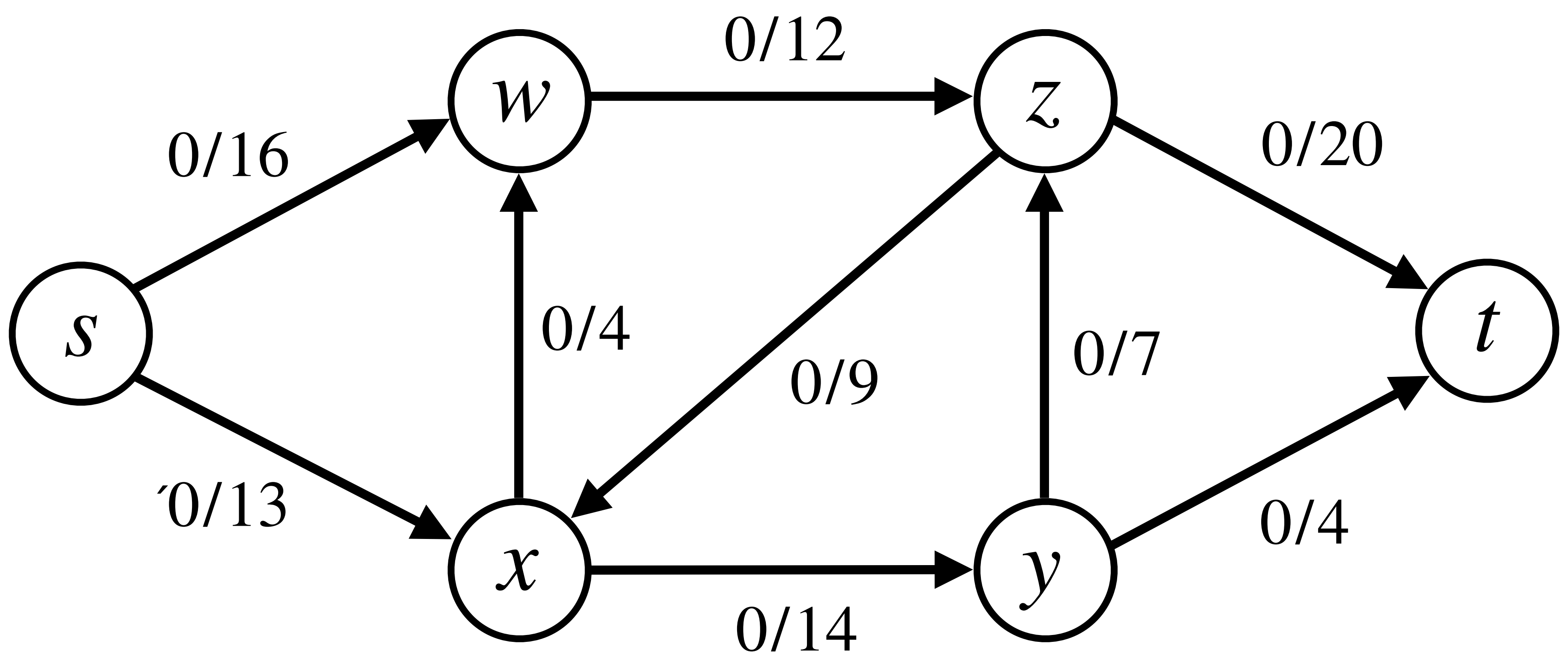


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge

Example:

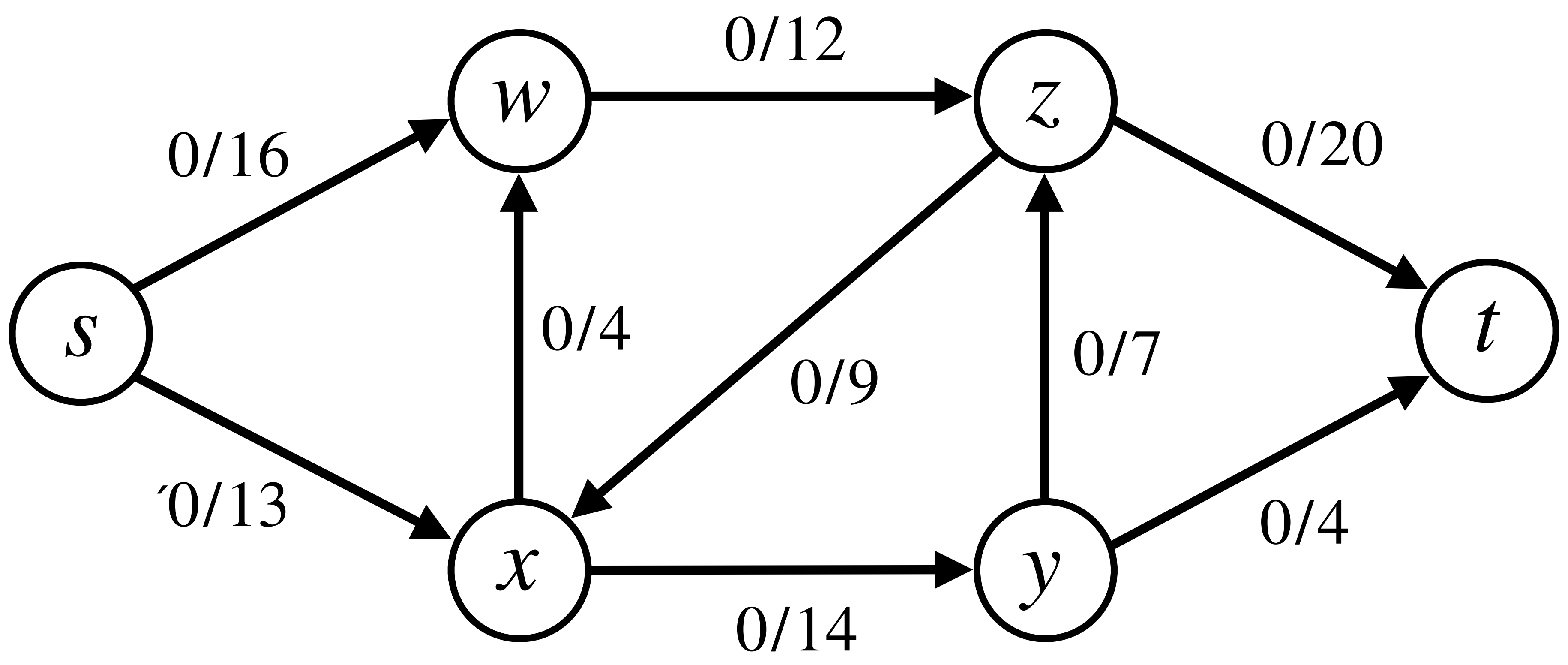


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$

Example:

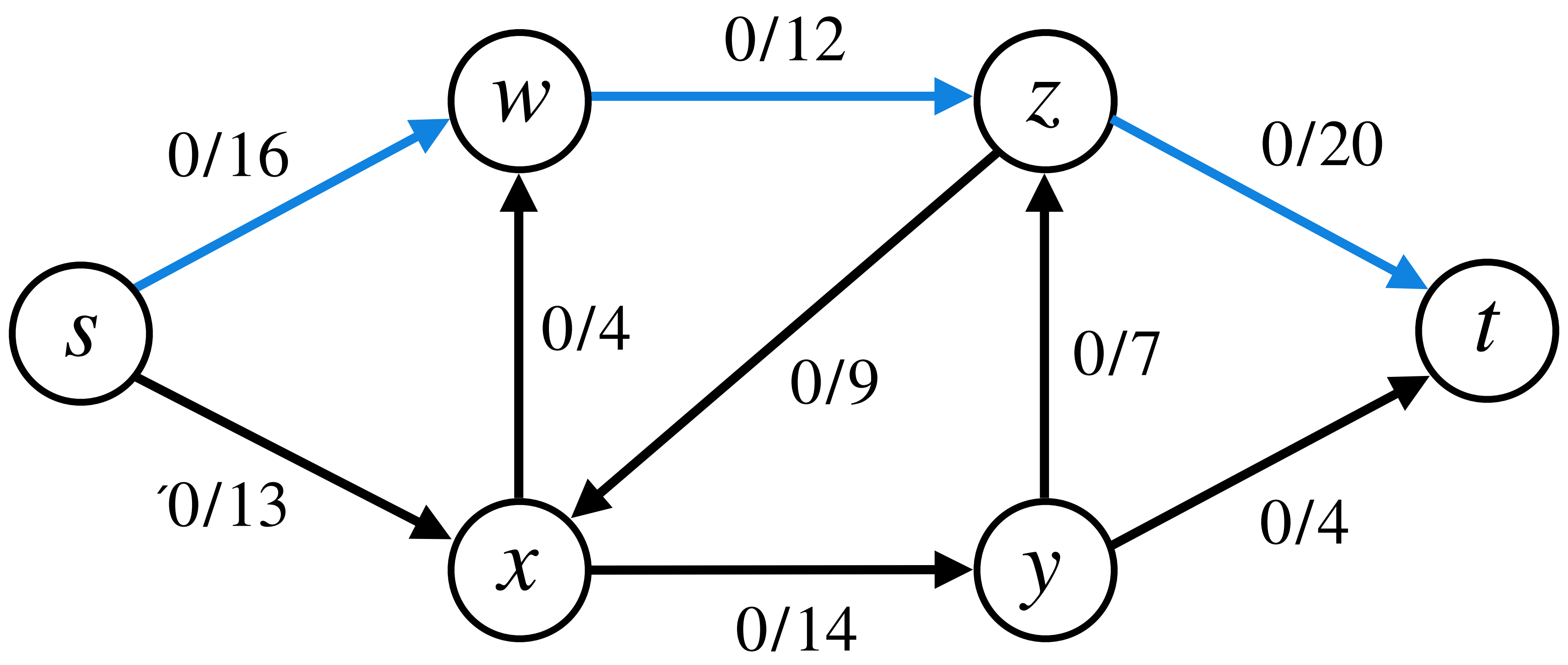


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$

Example:

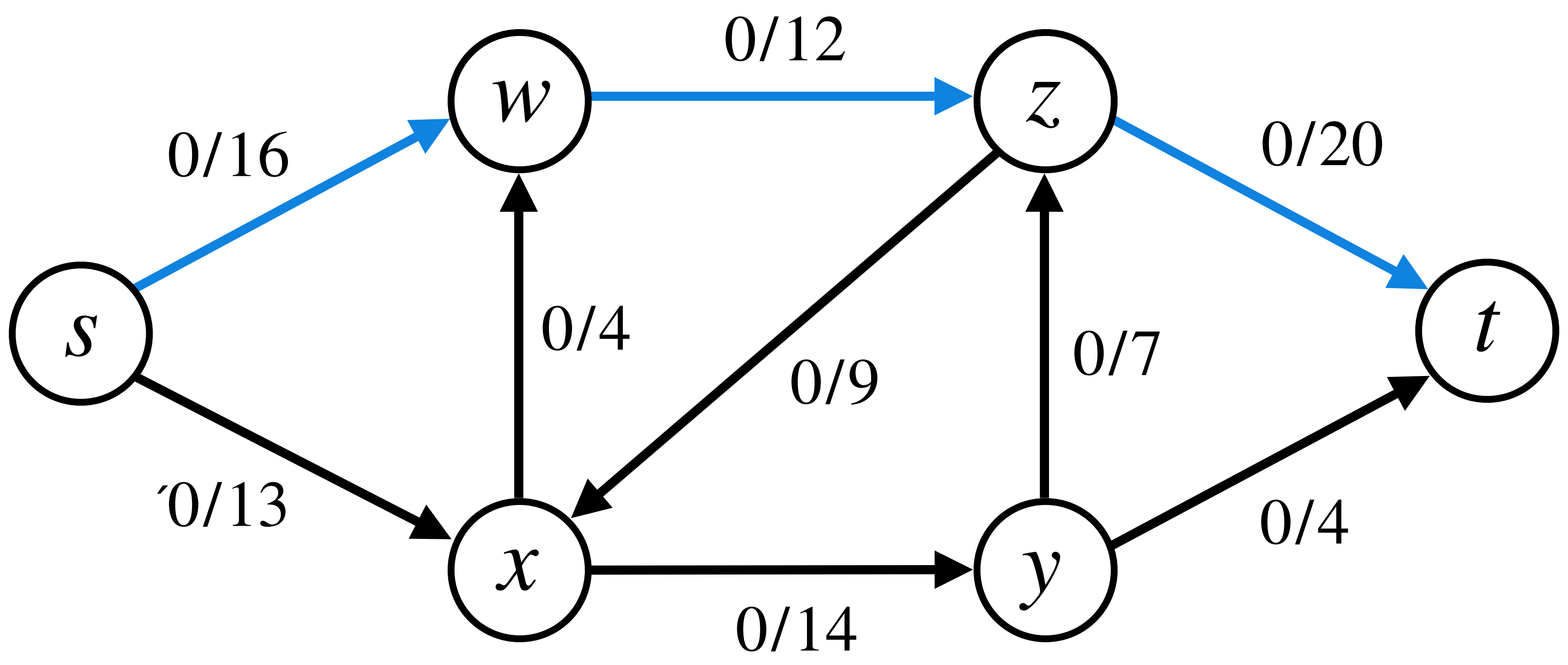


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P

Example:

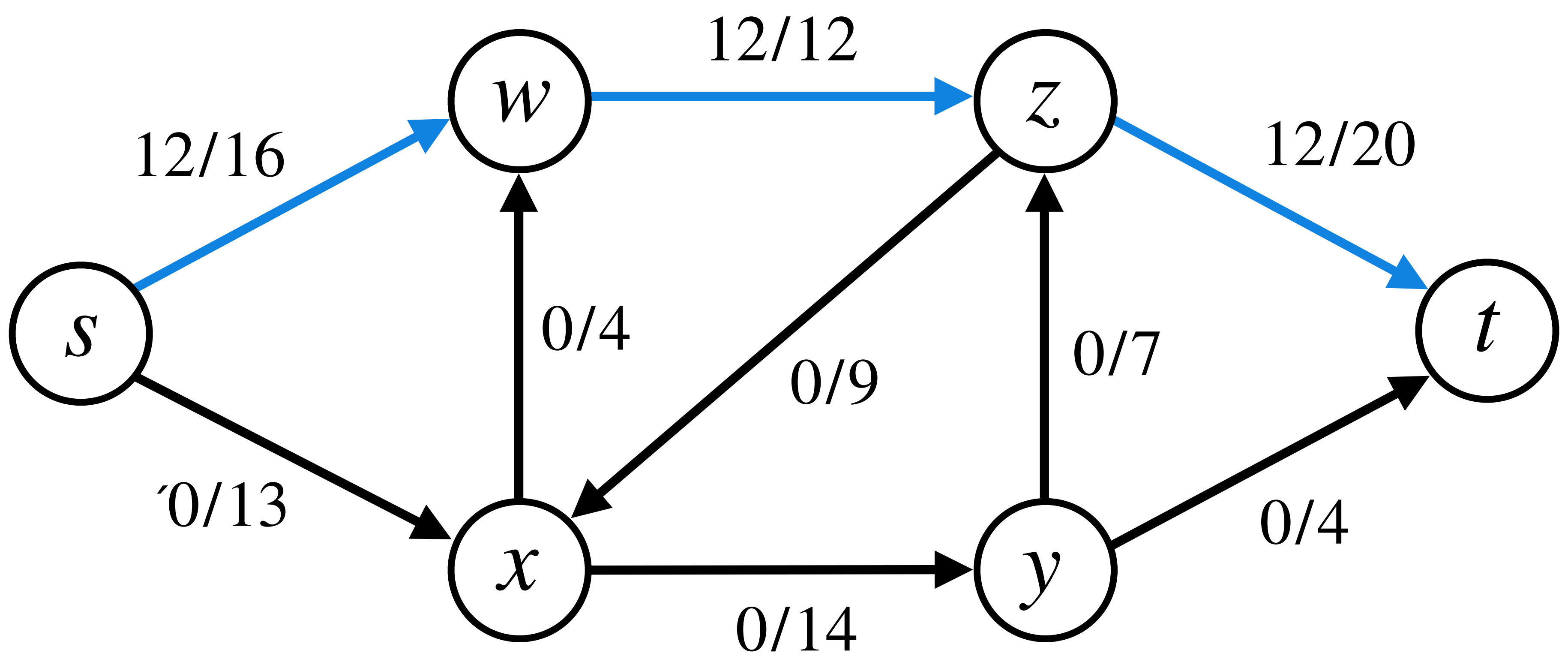


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P

Example:

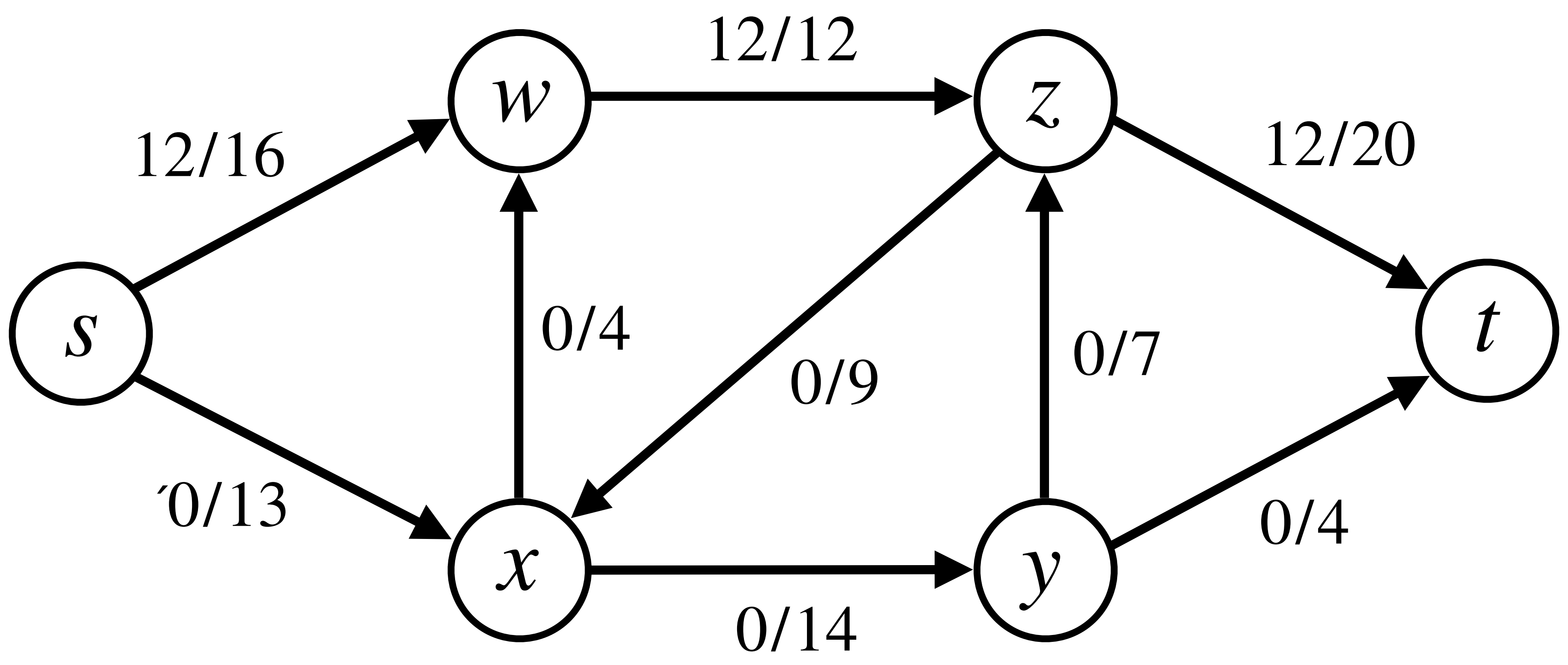


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck

Example:

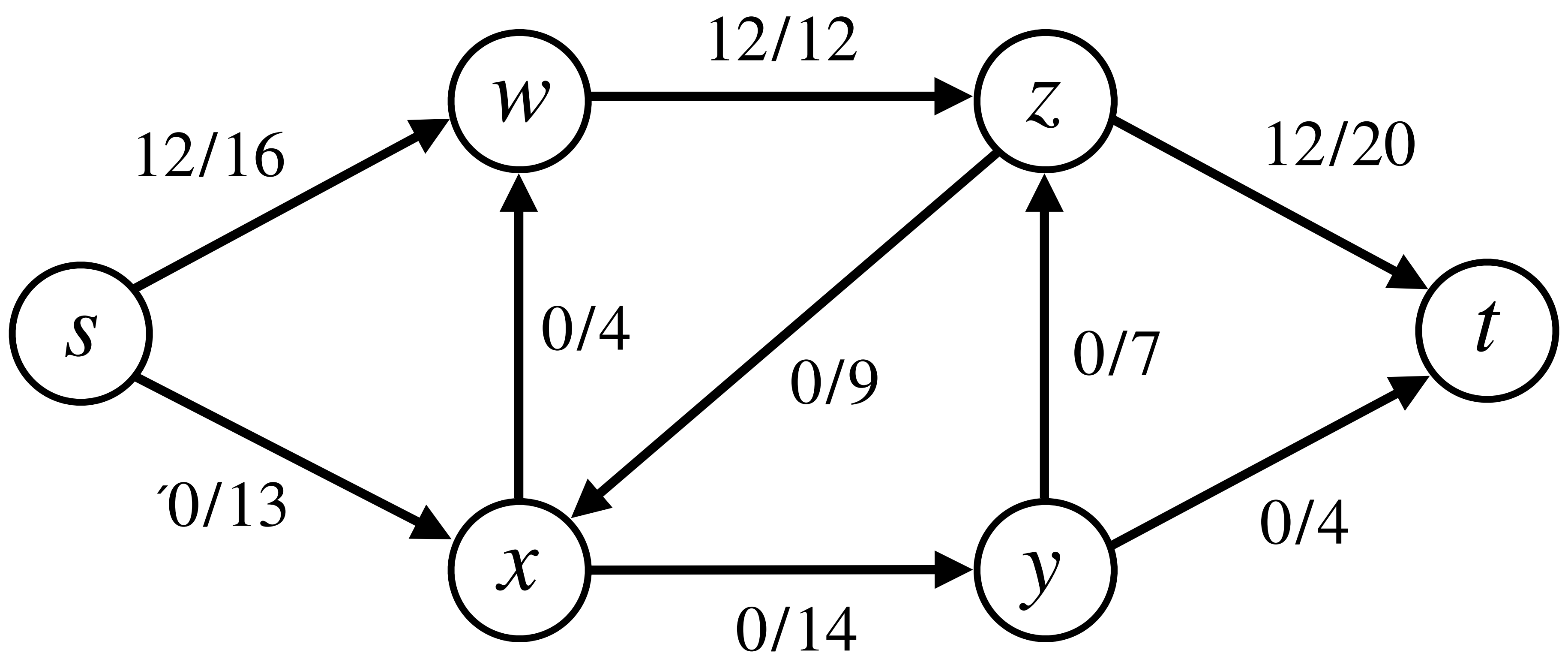


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Example:

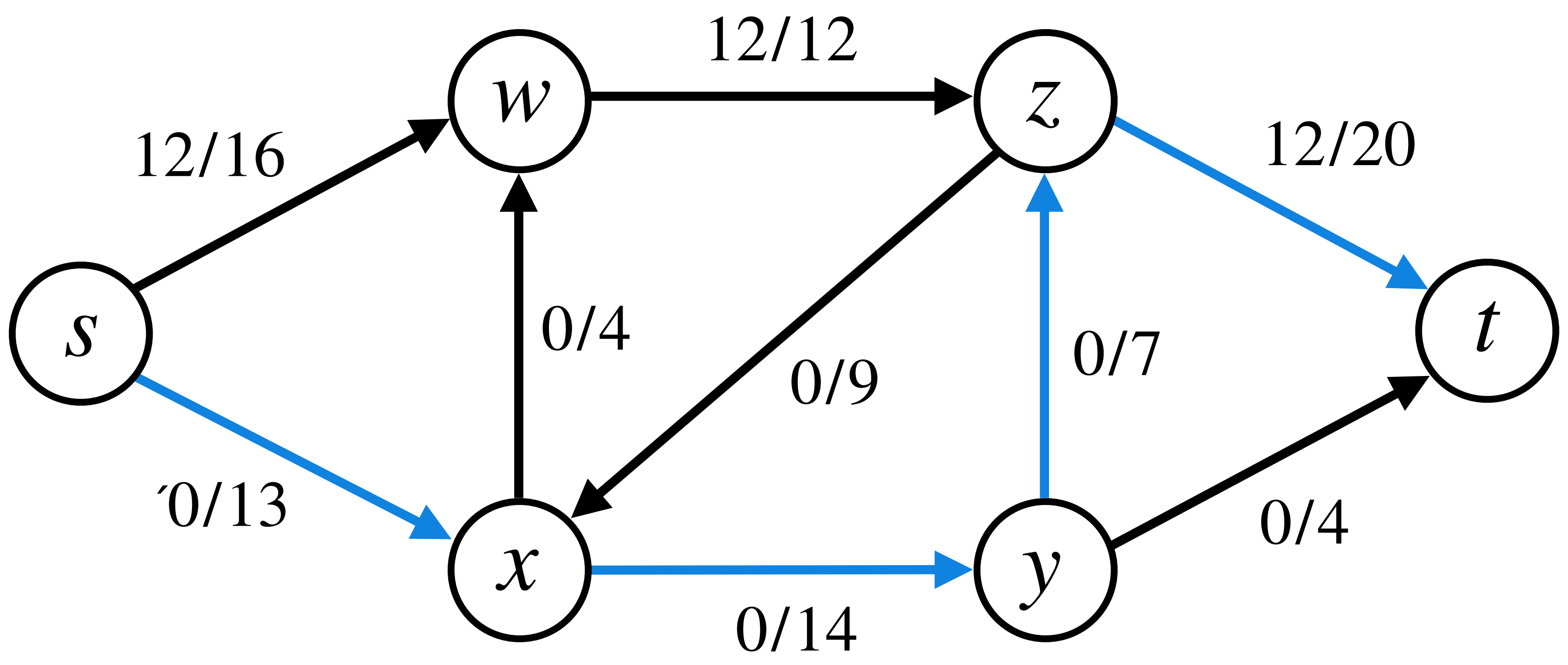


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Example:

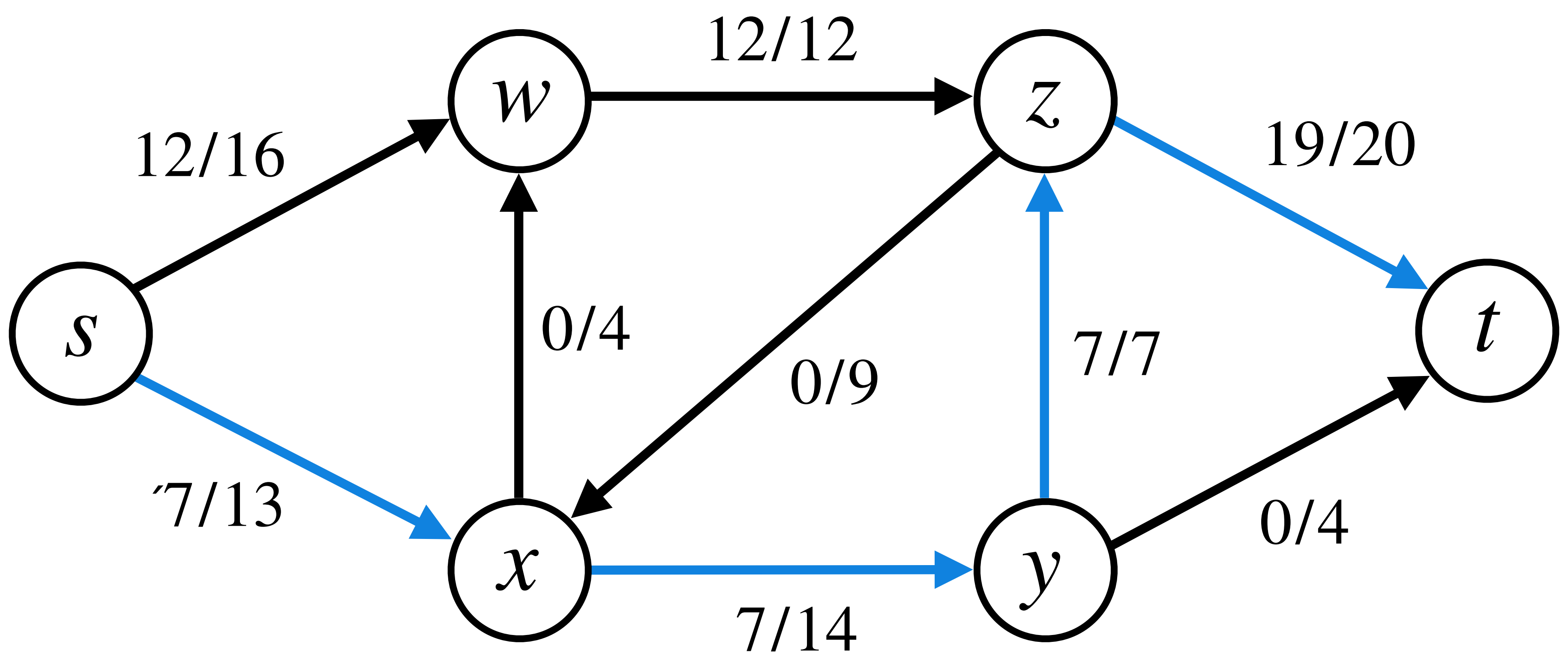


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Example:

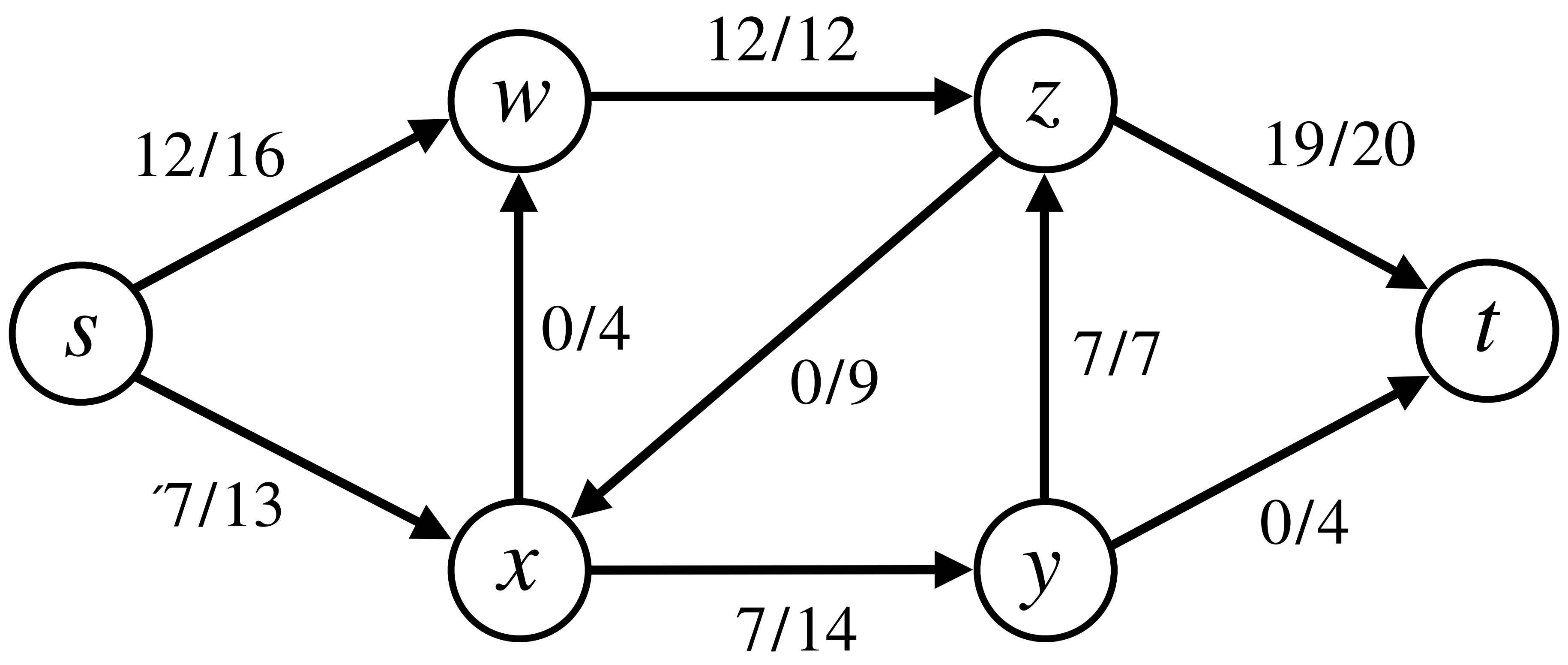


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Example:

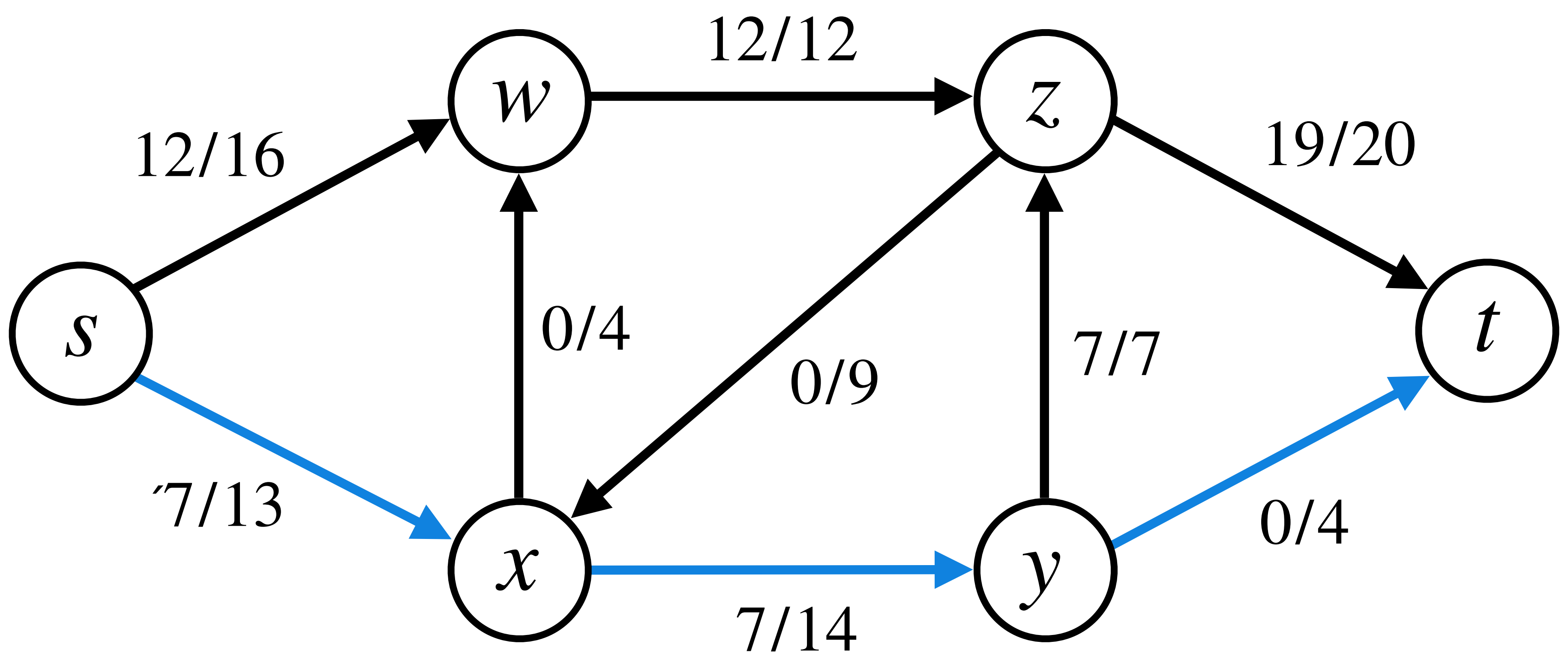


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Example:

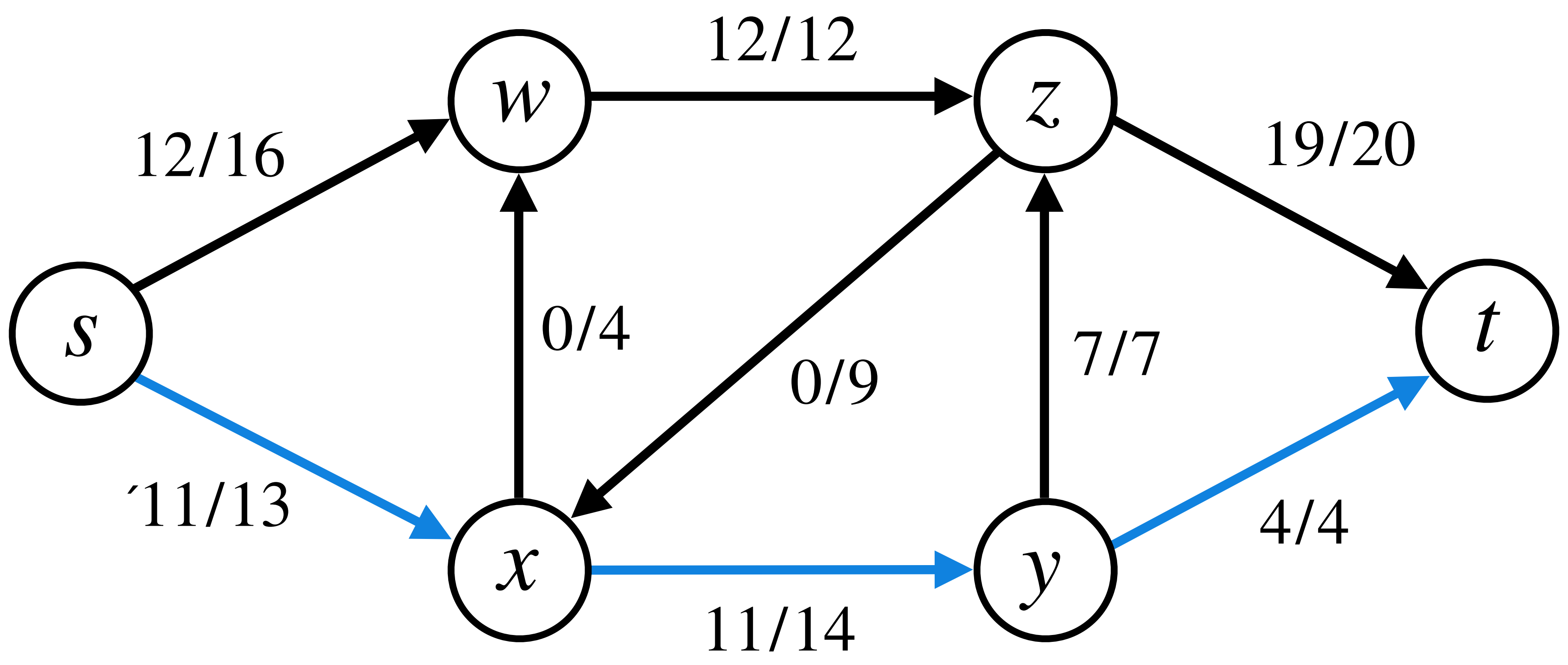


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Example:

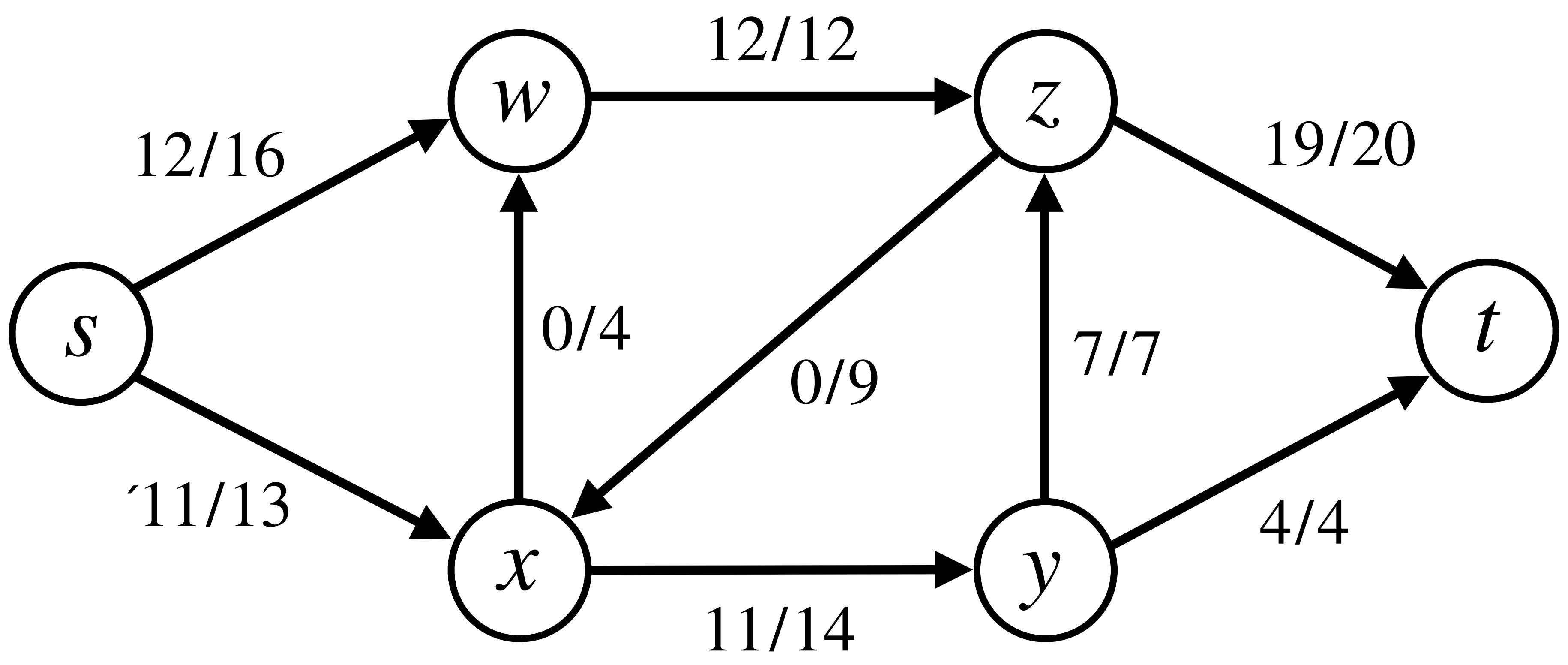


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Example:

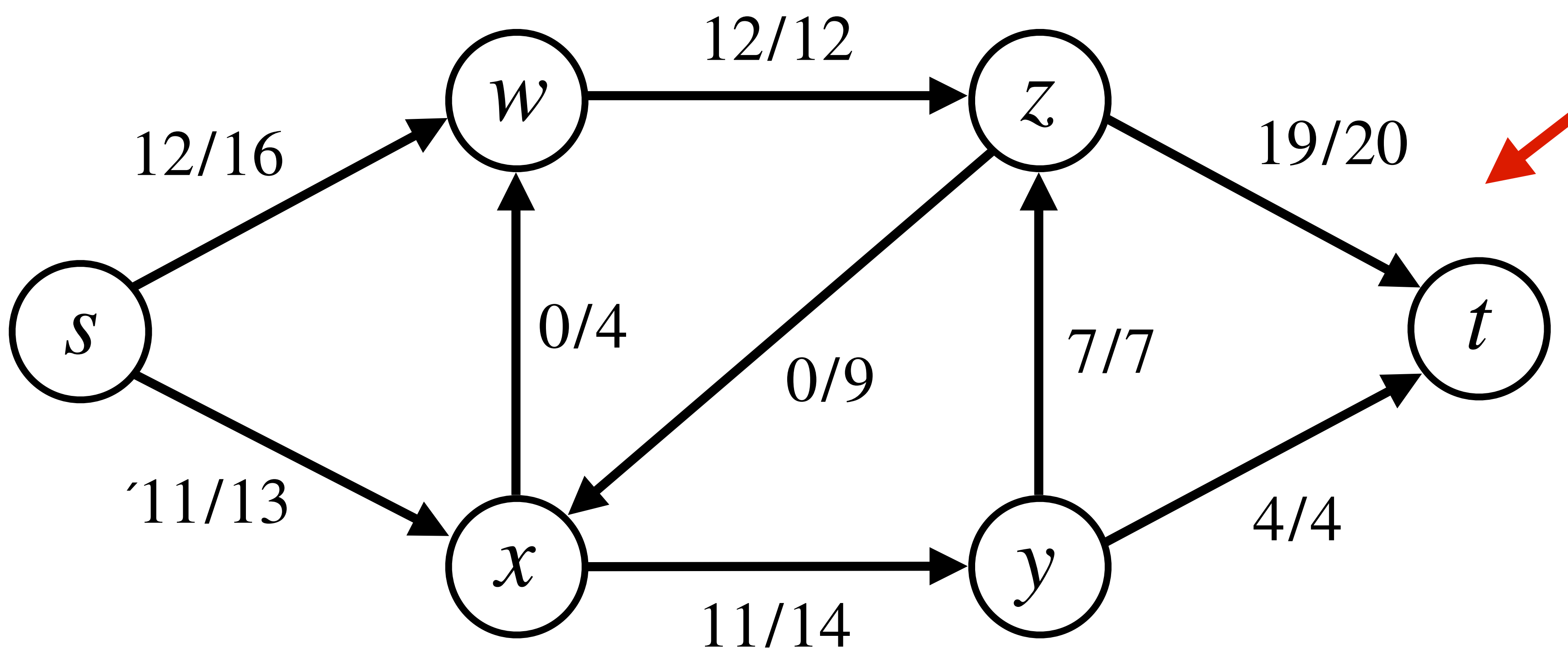


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Example:



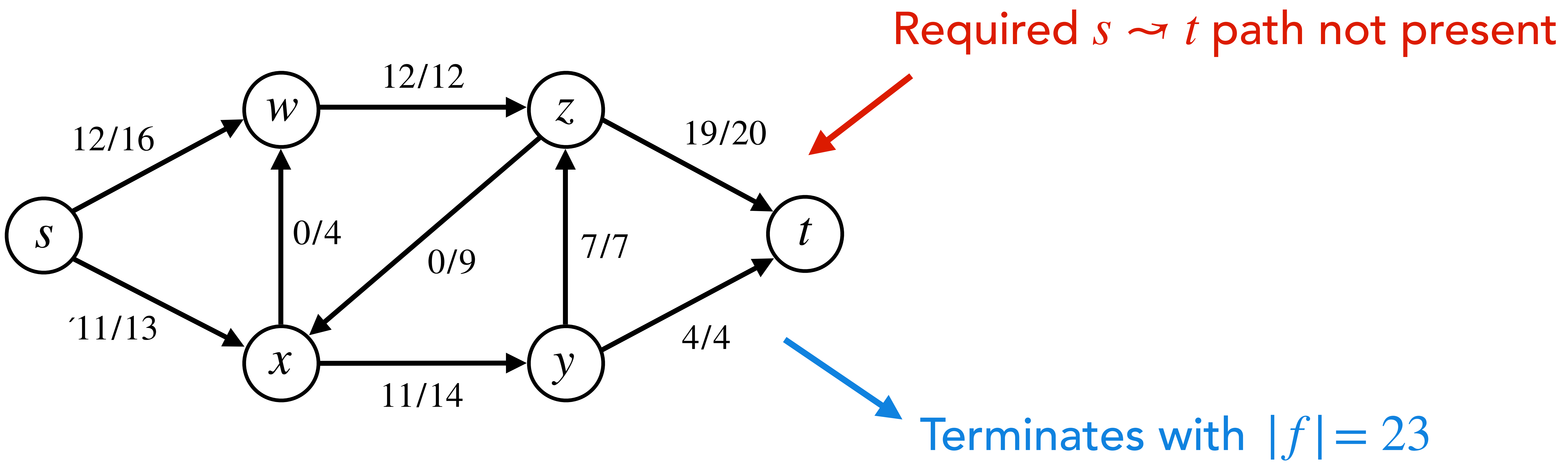
Required $s \rightsquigarrow t$ path not present

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Example:



Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Is the algorithm really correct?

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Is the algorithm really correct? No.

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

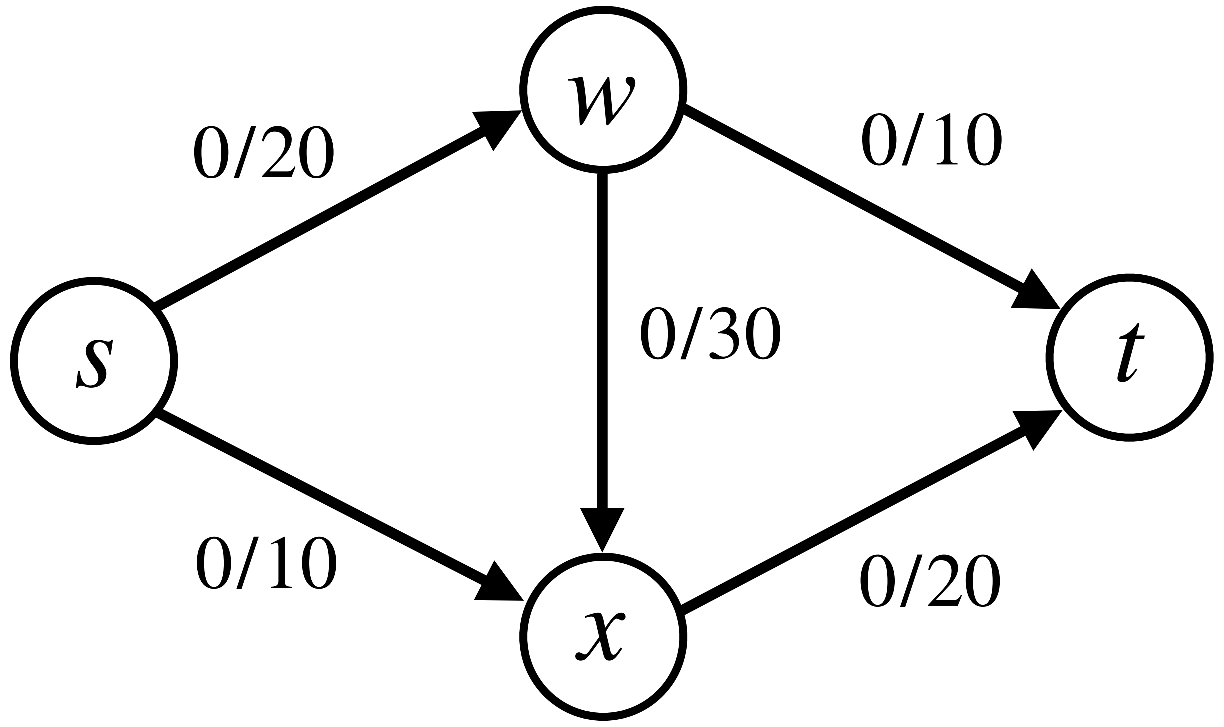
1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Counter-Example:

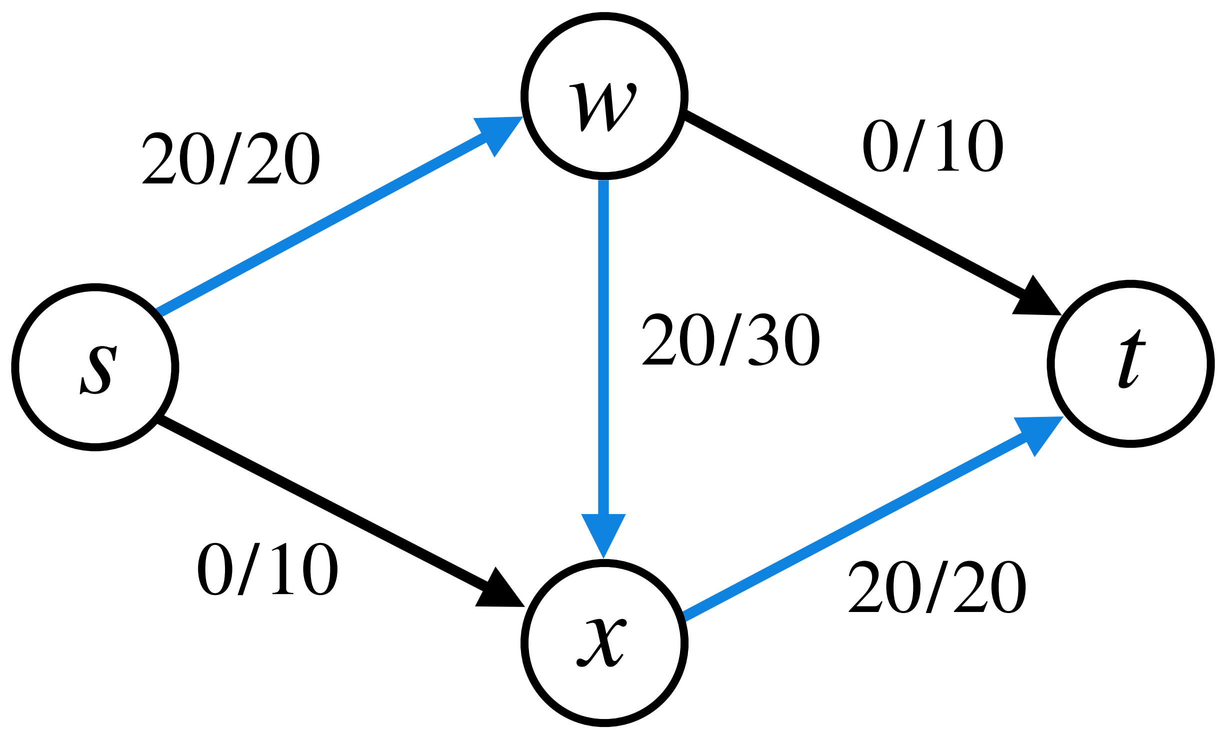


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Counter-Example:

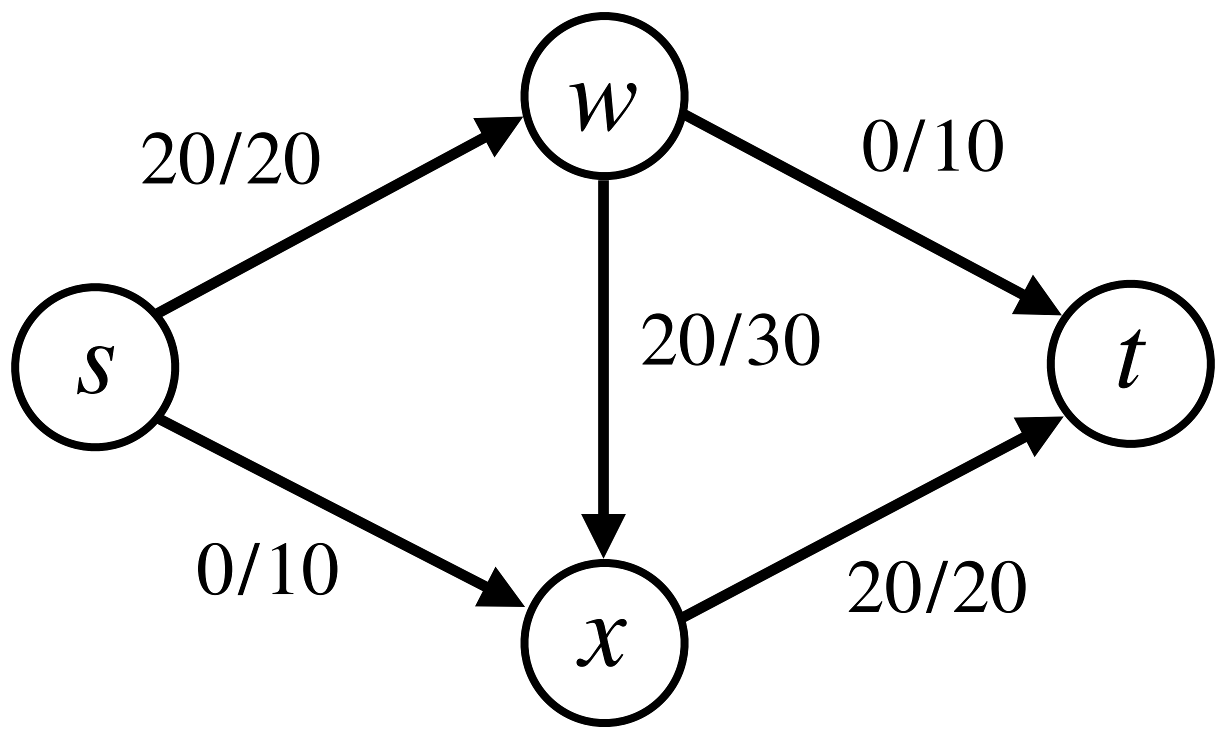


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Counter-Example:

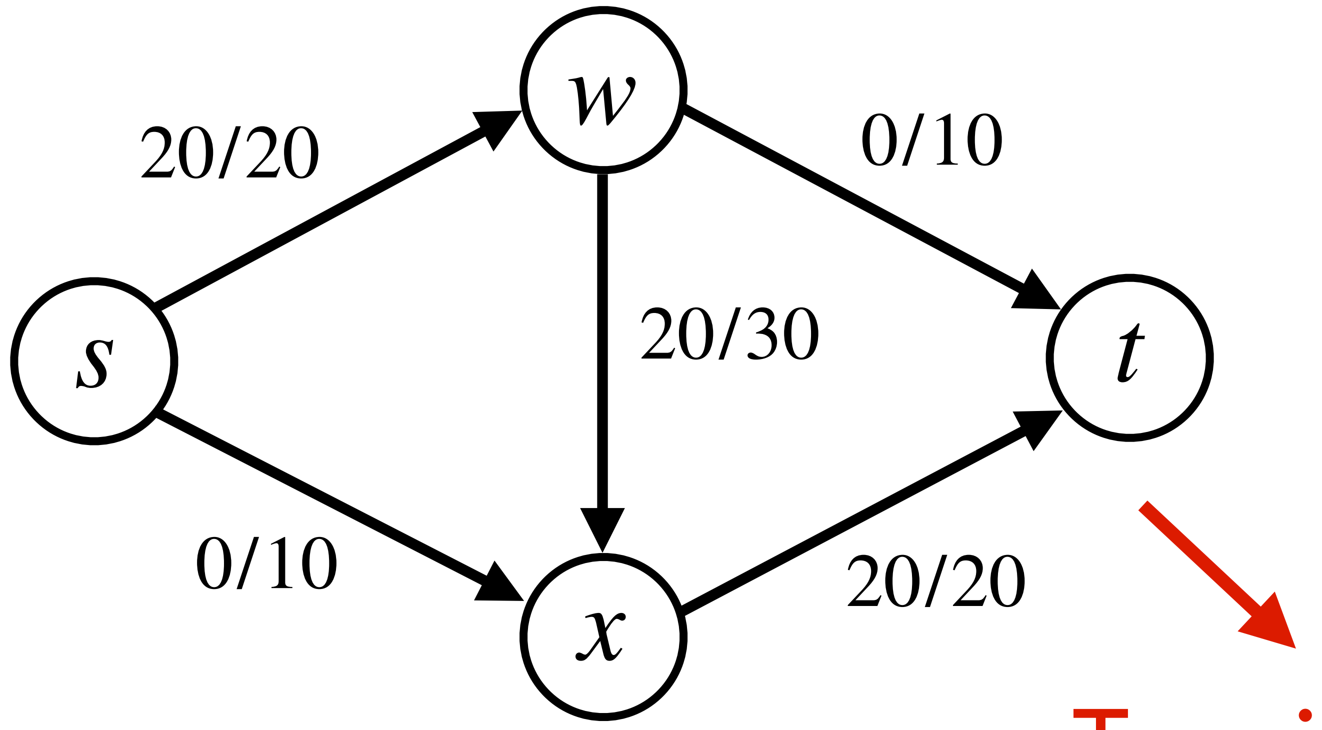


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Counter-Example:



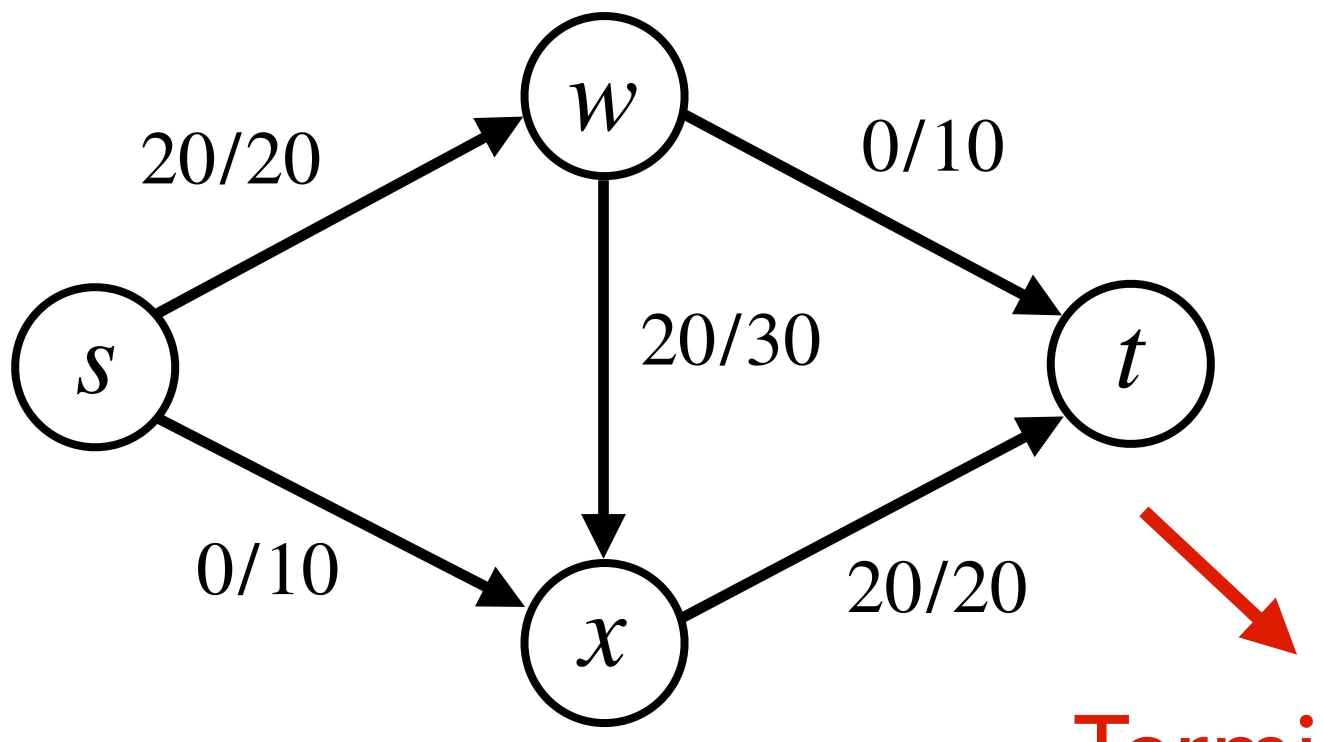
Terminates with $|f| = 20$

Finding Max Flow: An Attempt

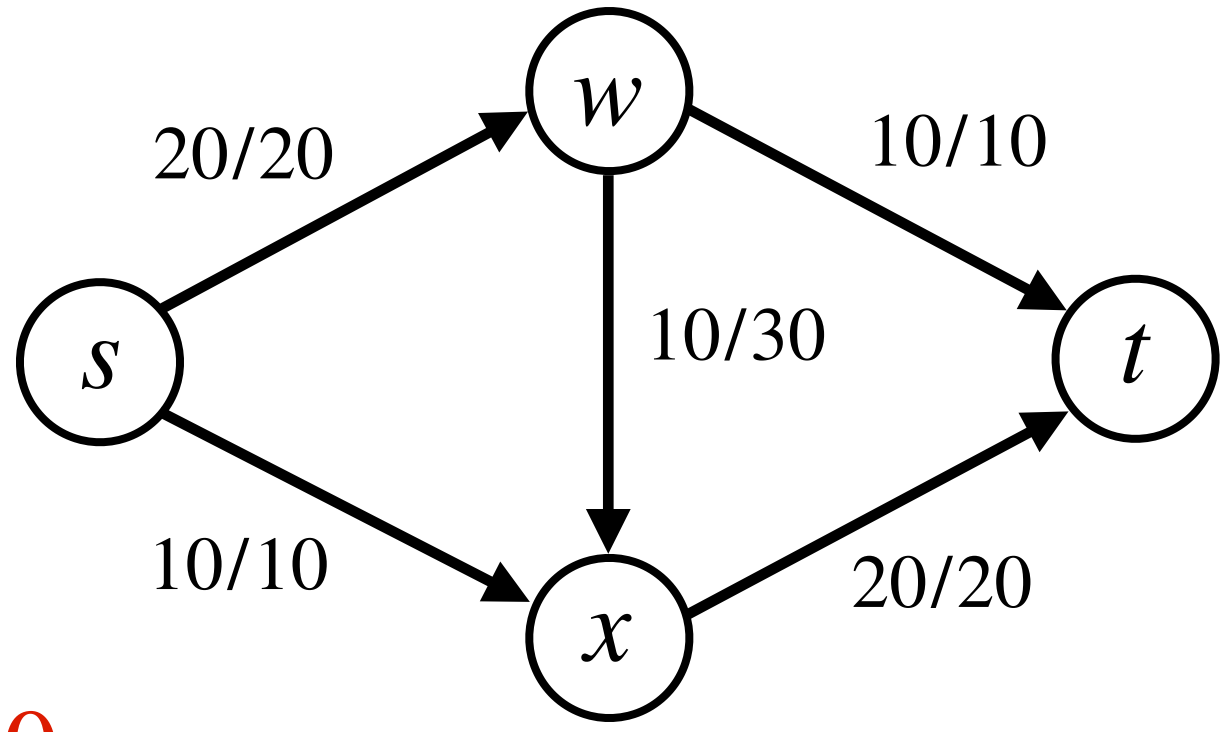
Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Counter-Example:



Terminates with $|f| = 20$

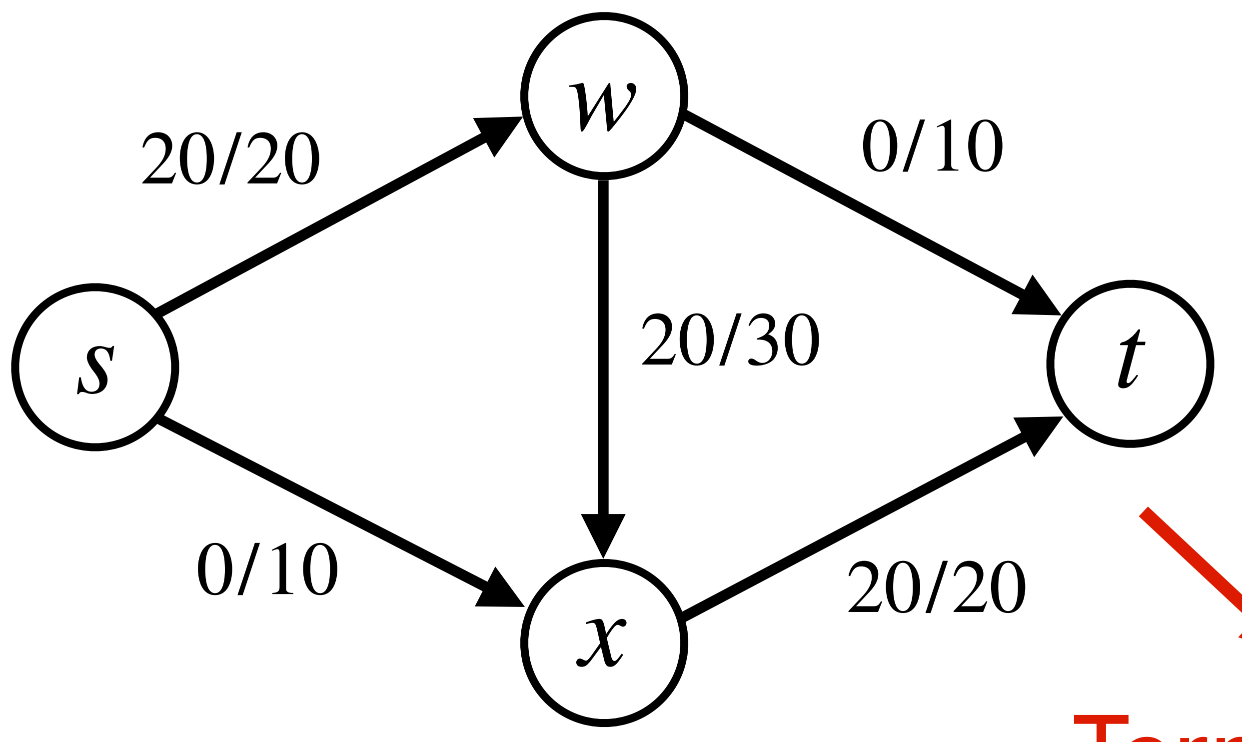


Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

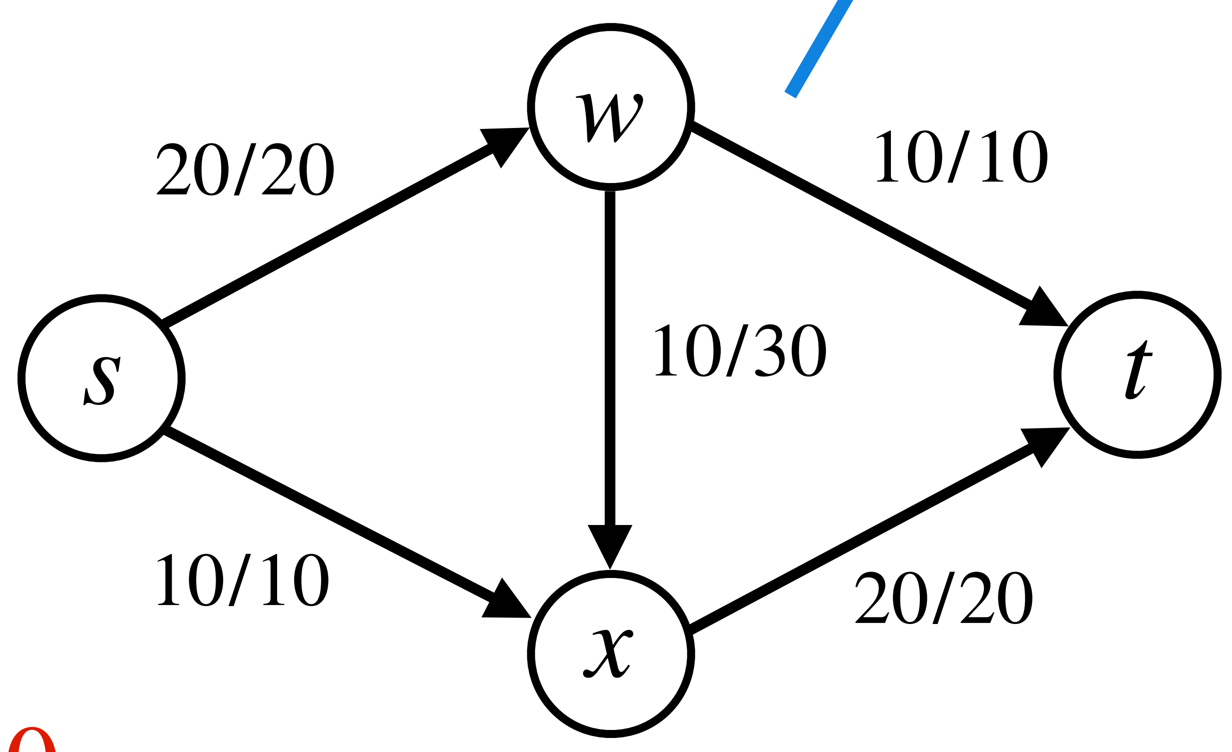
1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Counter-Example:



Terminates with $|f| = 20$

Max-flow is 30

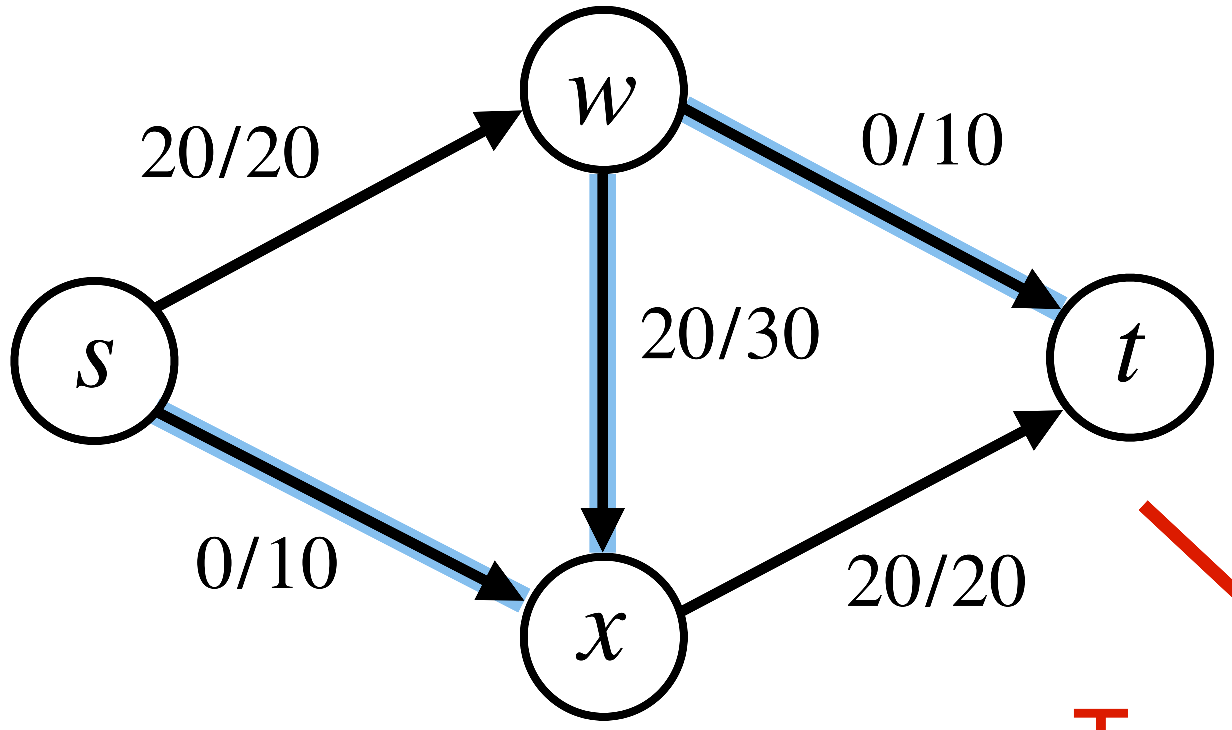


Finding Max Flow: An Attempt

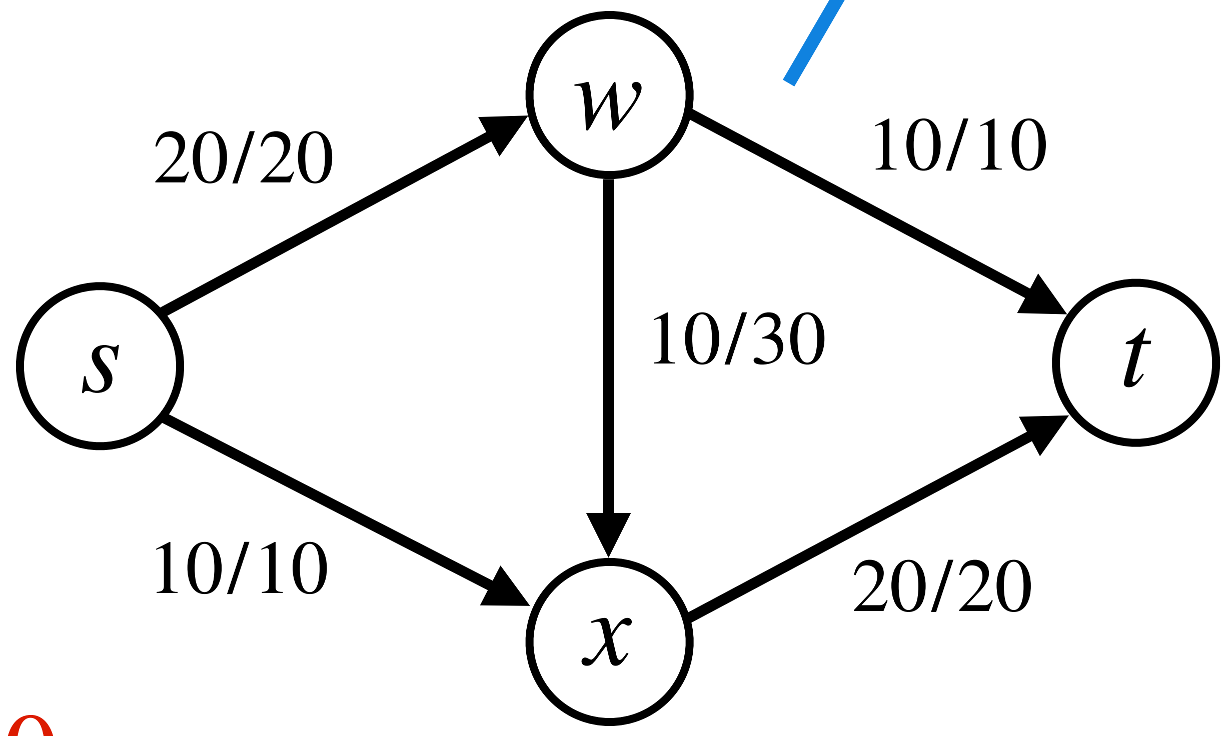
Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Counter-Example:



Terminates with $|f| = 20$



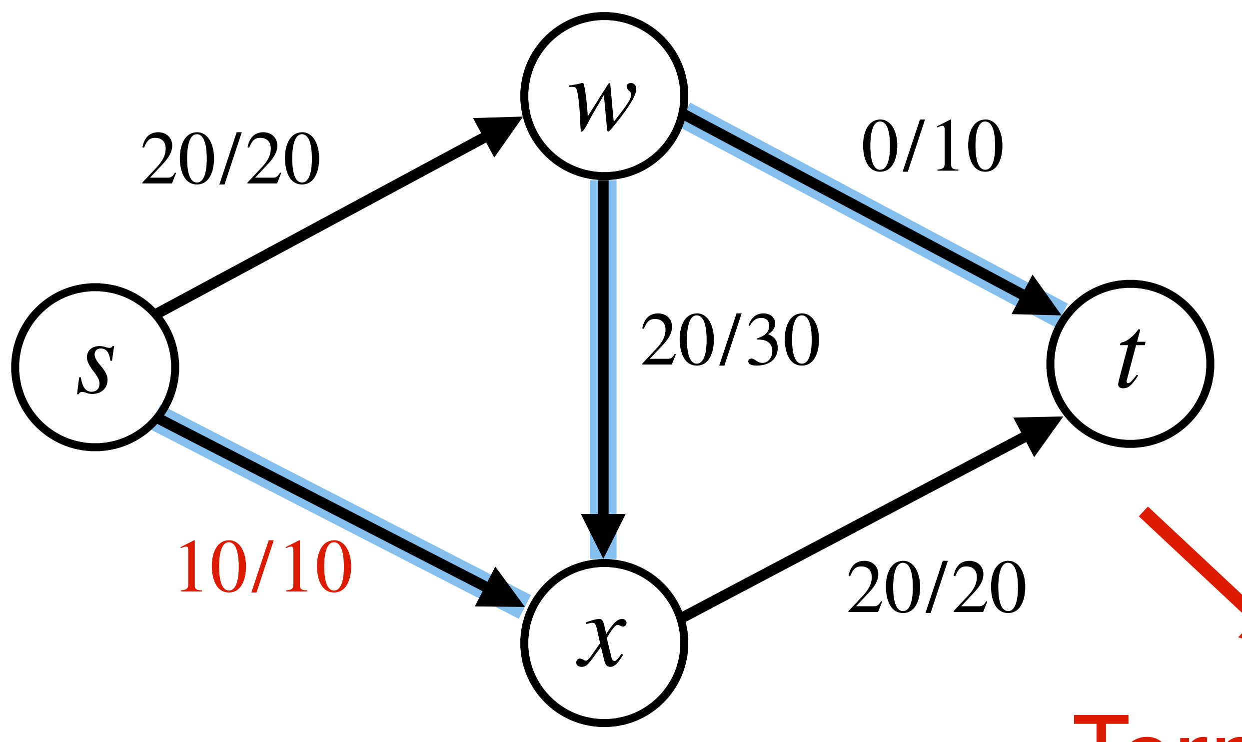
Max-flow is 30

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

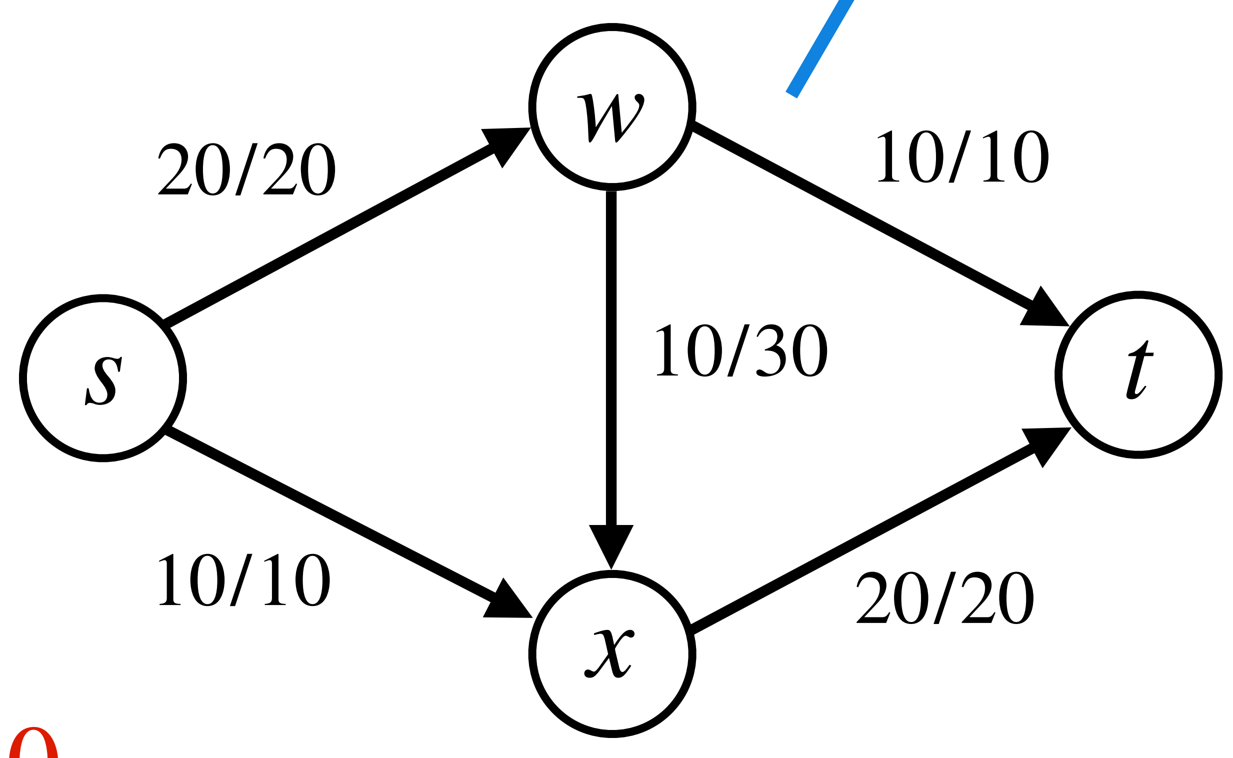
1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Counter-Example:



Terminates with $|f| = 20$

Max-flow is 30

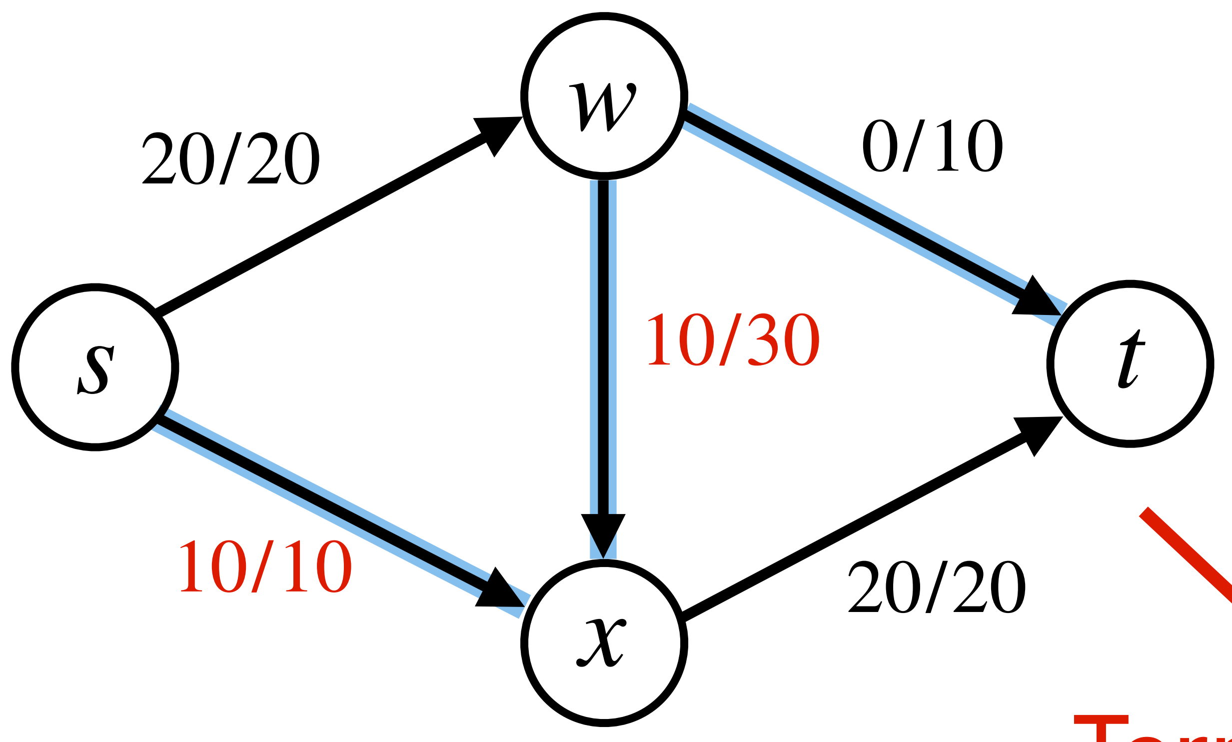


Finding Max Flow: An Attempt

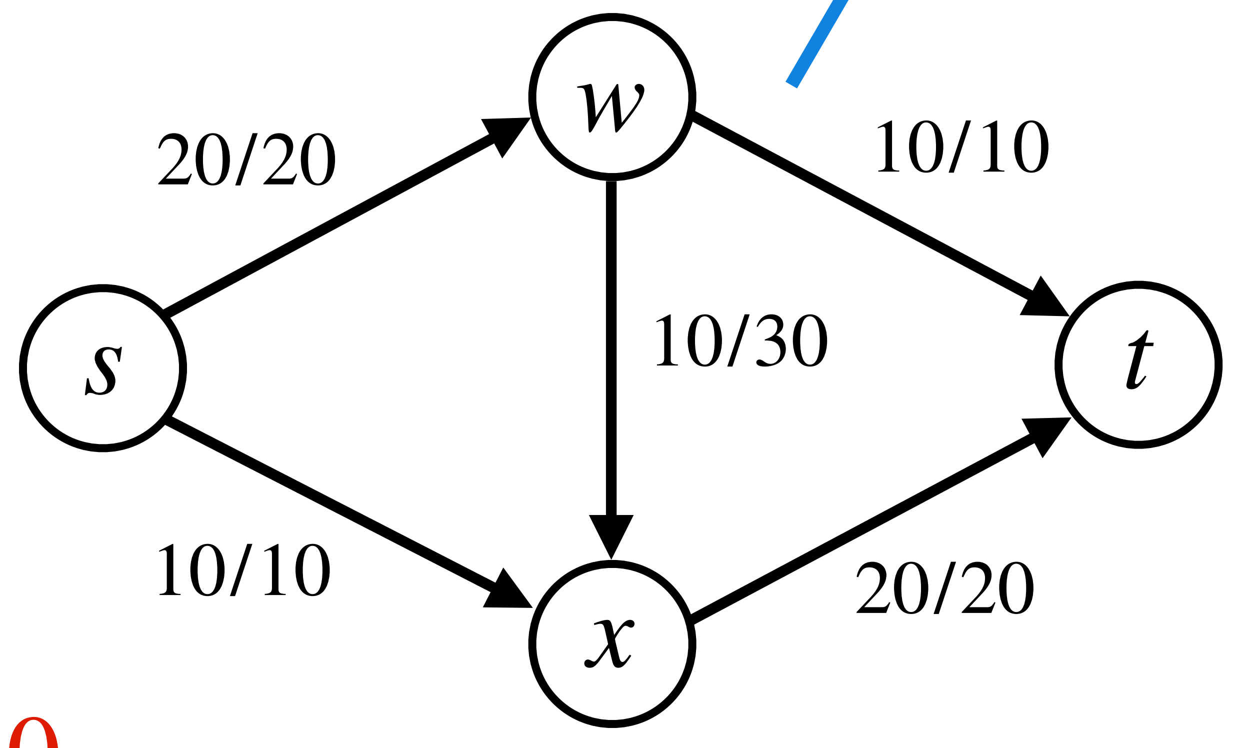
Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Counter-Example:



Terminates with $|f| = 20$



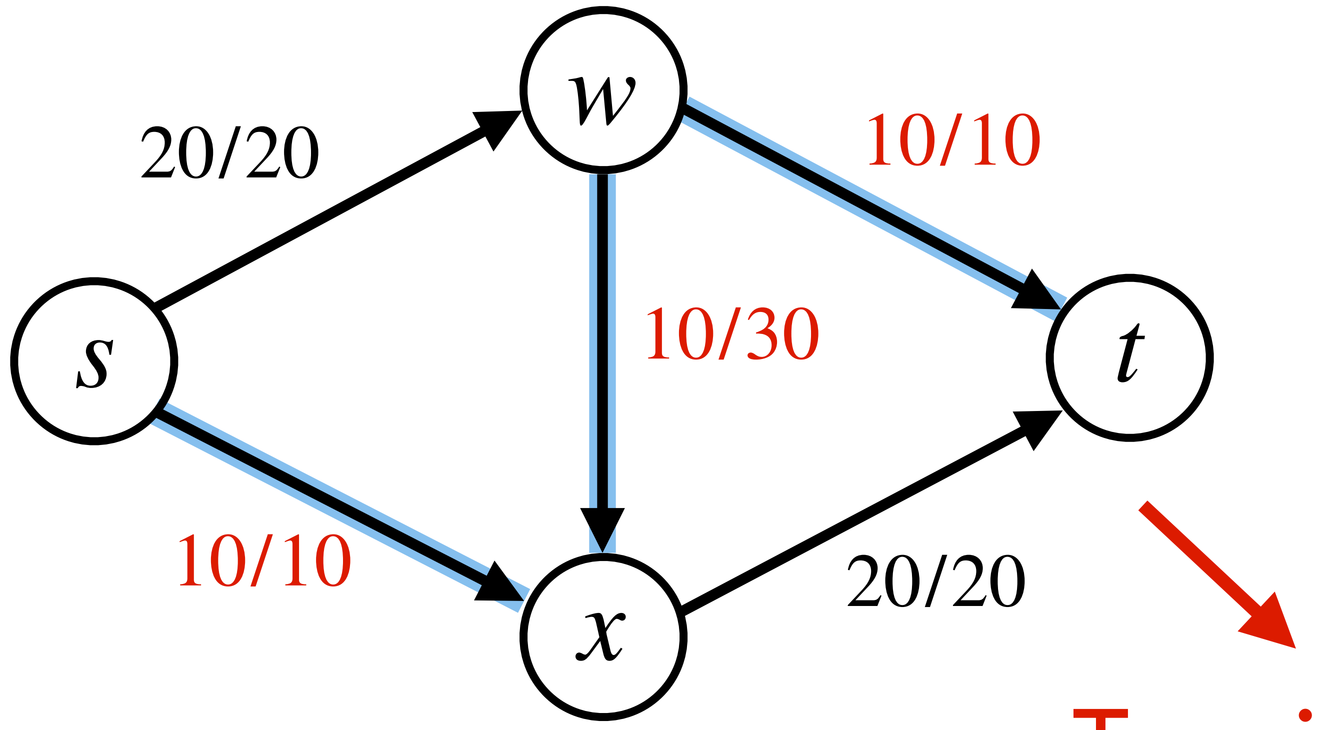
Max-flow is 30

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

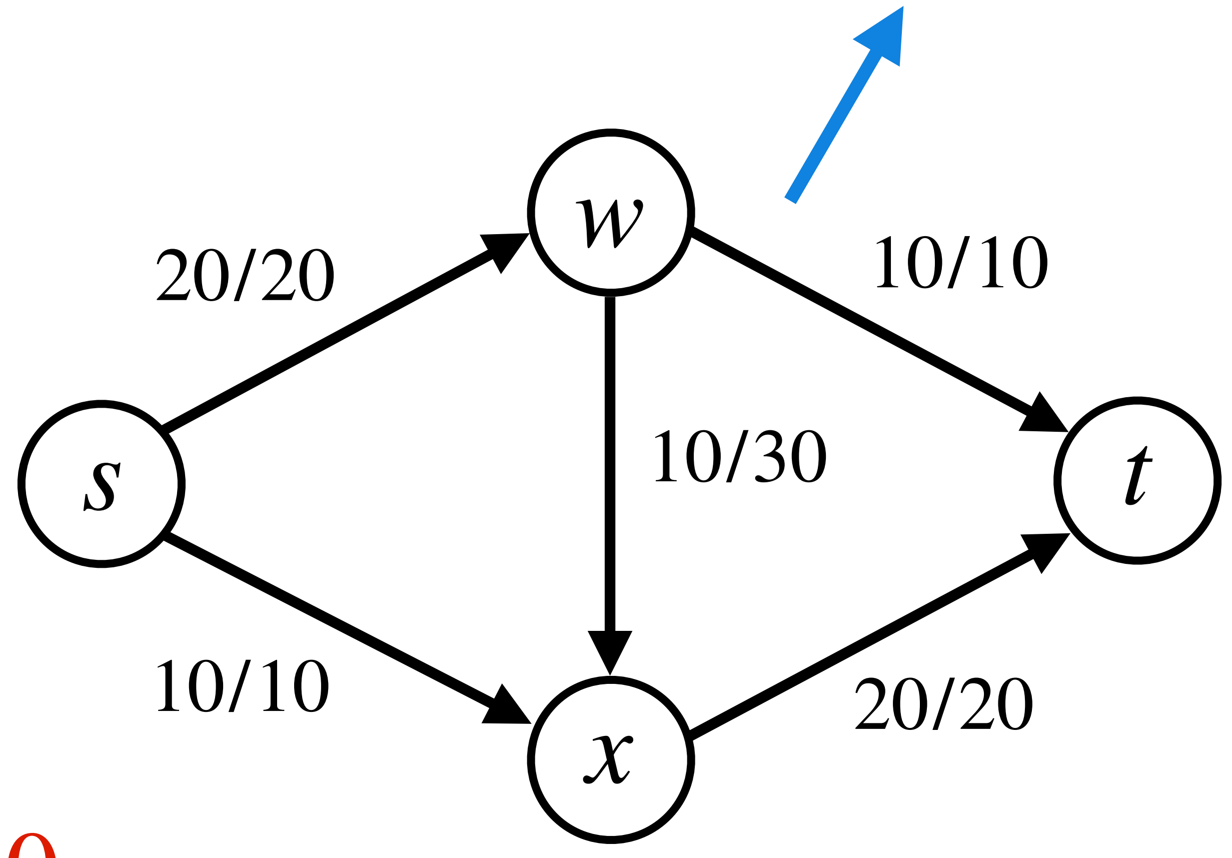
1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Counter-Example:



Terminates with $|f| = 20$

Max-flow is 30



Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Observation: The above algorithm never decreases flow along any edge.

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Observation: The above algorithm never decreases flow along any edge.

Possible Fix: May be we should allow decreasing/redistributing flows.

Finding Max Flow: An Attempt

Naive-Max-Flow(G, s, t)**:**

1. Start with flow $f = 0$ for every edge
2. Find an $s \rightsquigarrow t$ path P where every edge has $f < c$
3. Augment flow f with the least $c - f$ on P along P
4. Keep repeating line 2 & 3 until you get stuck
5. Return f

Observation: The above algorithm never decreases flow along any edge.

Possible Fix: May be we should allow decreasing/redistributing flows.

Need to learn a new structure for that!

Residual Networks

Residual Networks

Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$,

Residual Networks

Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

Residual Networks

Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$

Residual Networks

Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

Residual Networks

Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

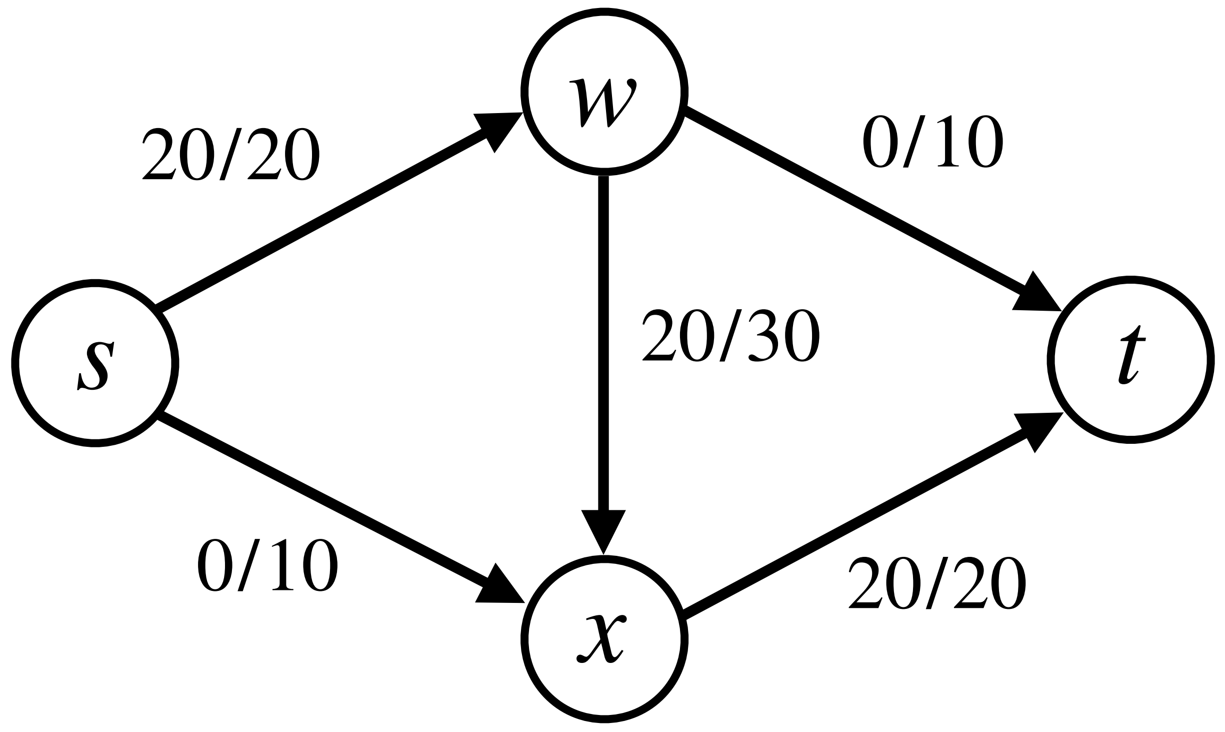
Example:

Residual Networks

Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

Example:



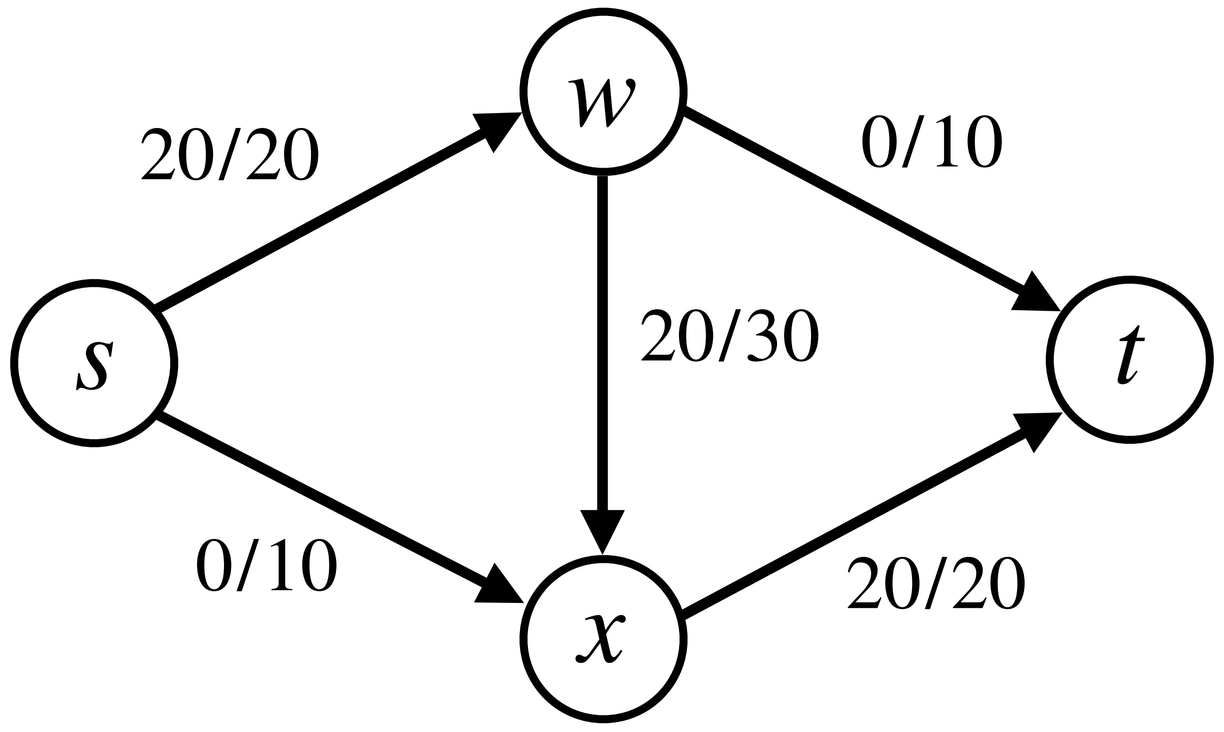
G

Residual Networks

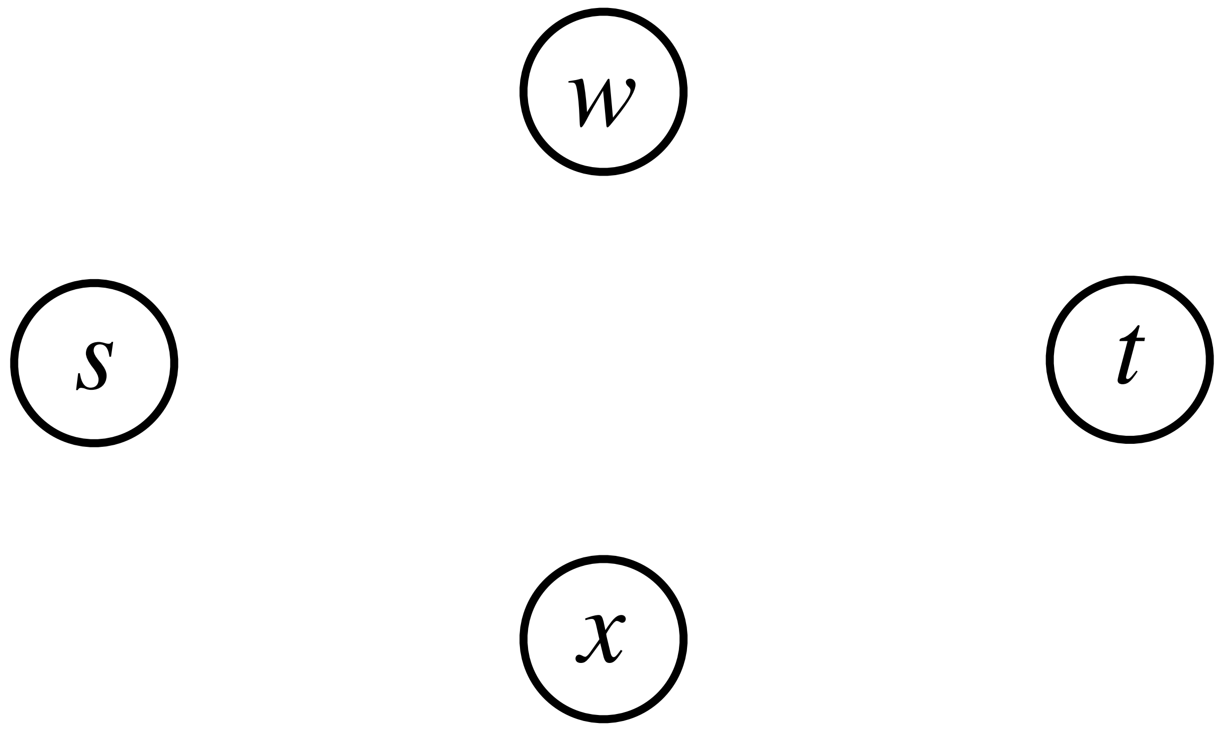
Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

Example:



G



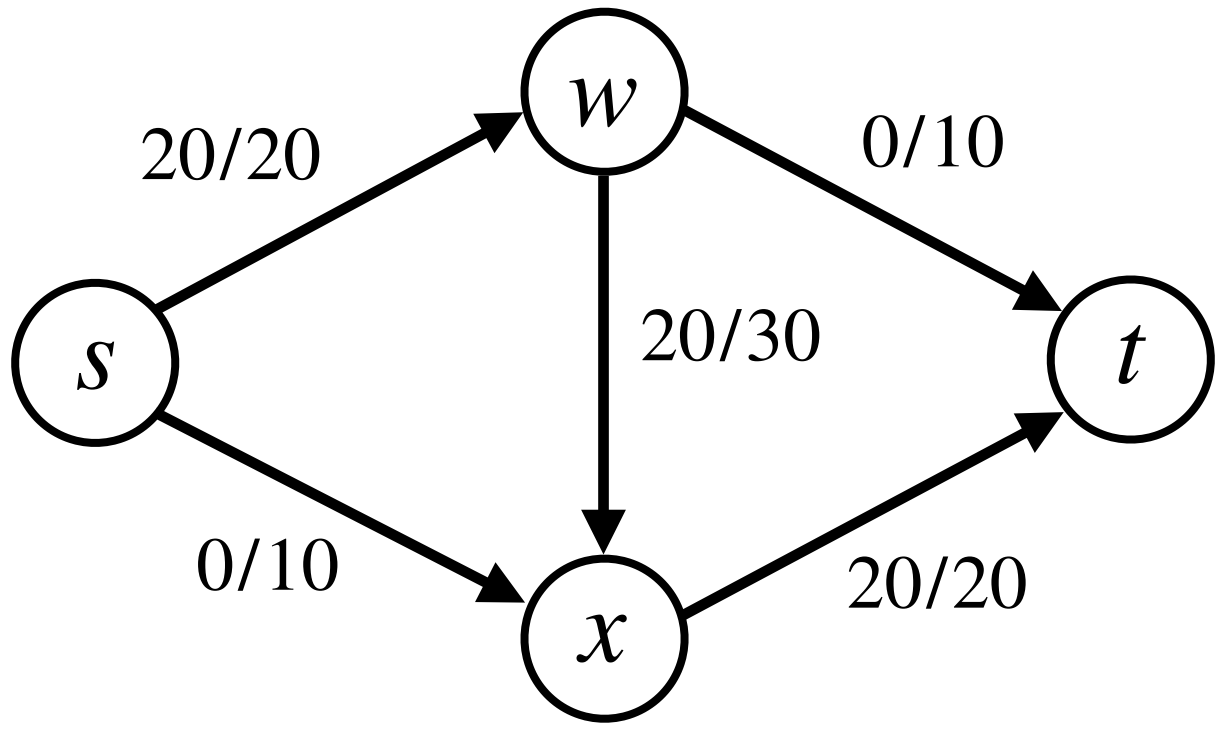
G_f

Residual Networks

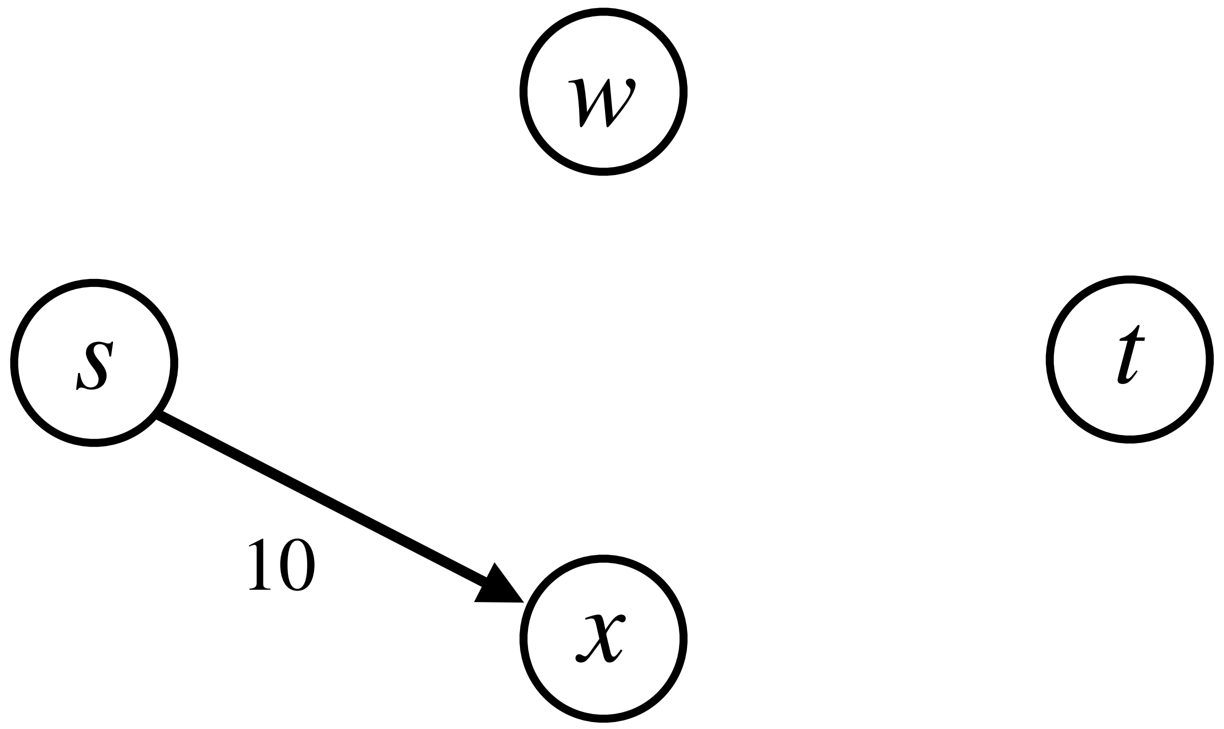
Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

Example:



G



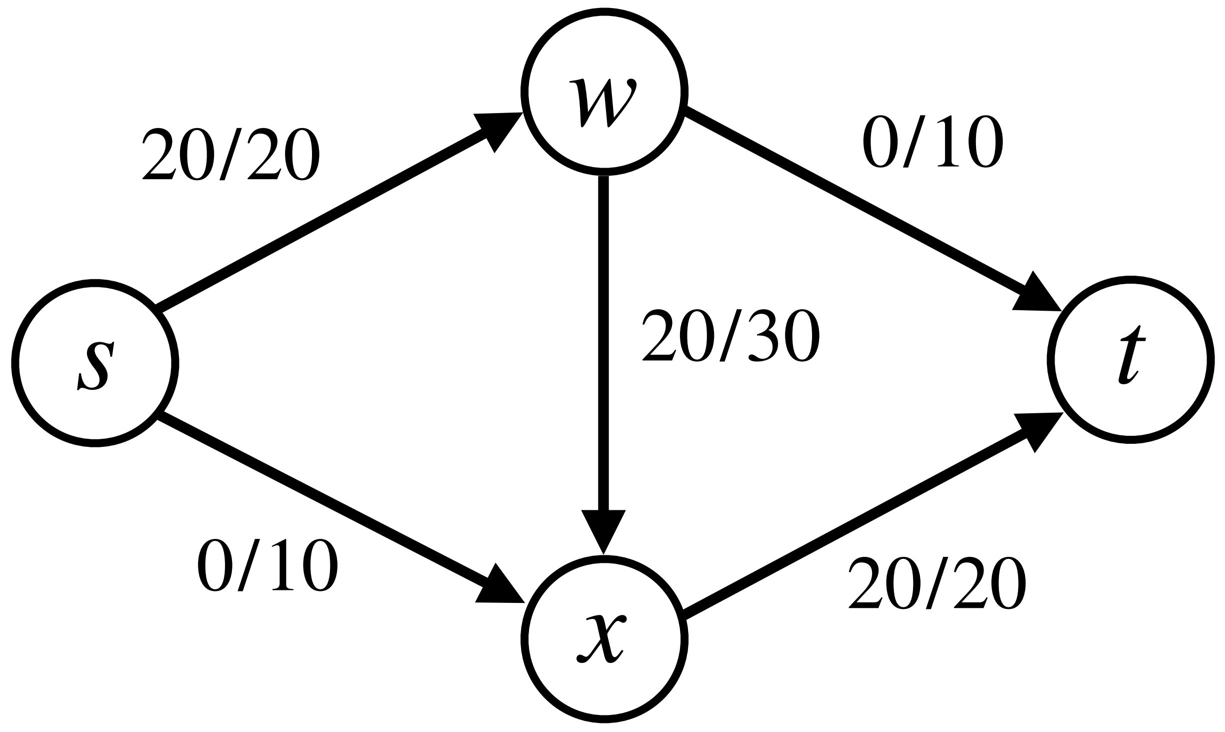
G_f

Residual Networks

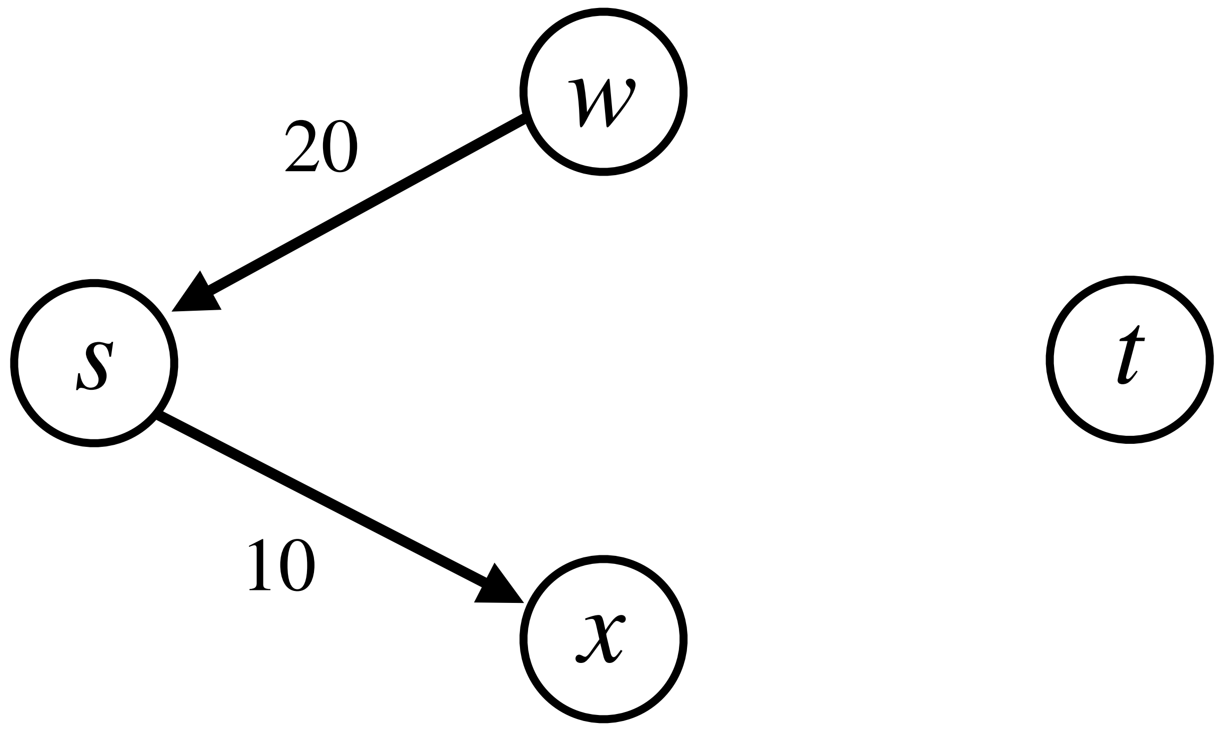
Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

Example:



G



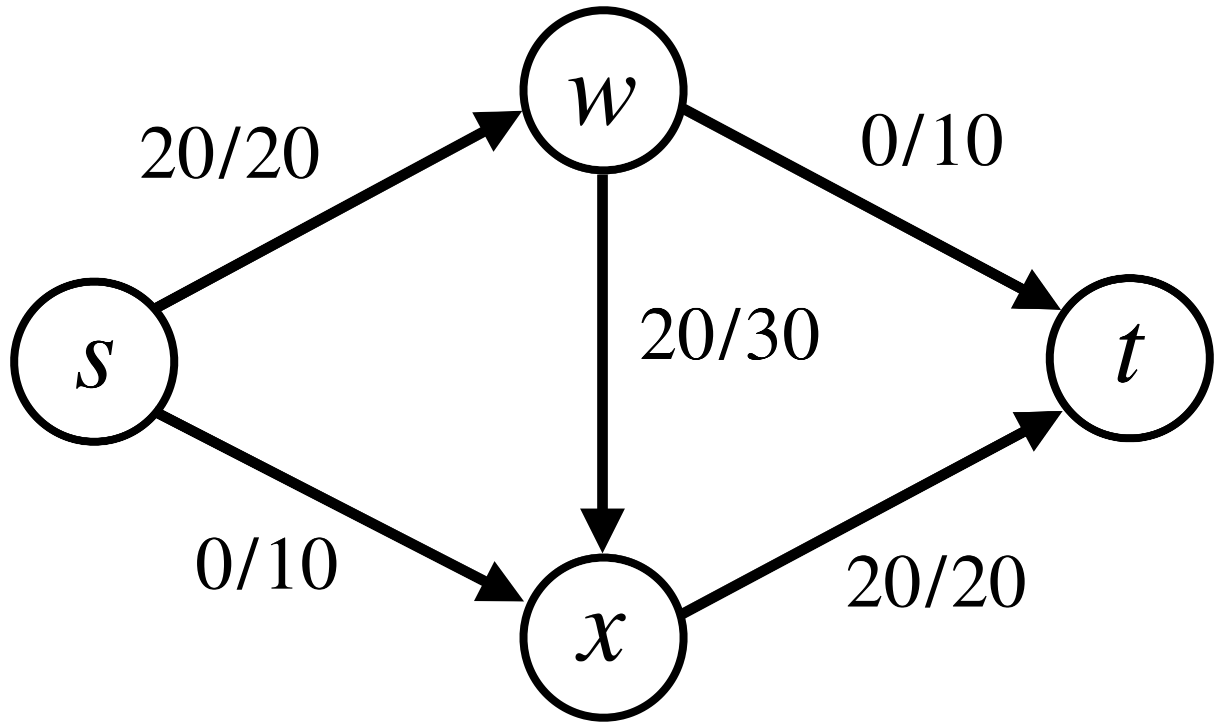
G_f

Residual Networks

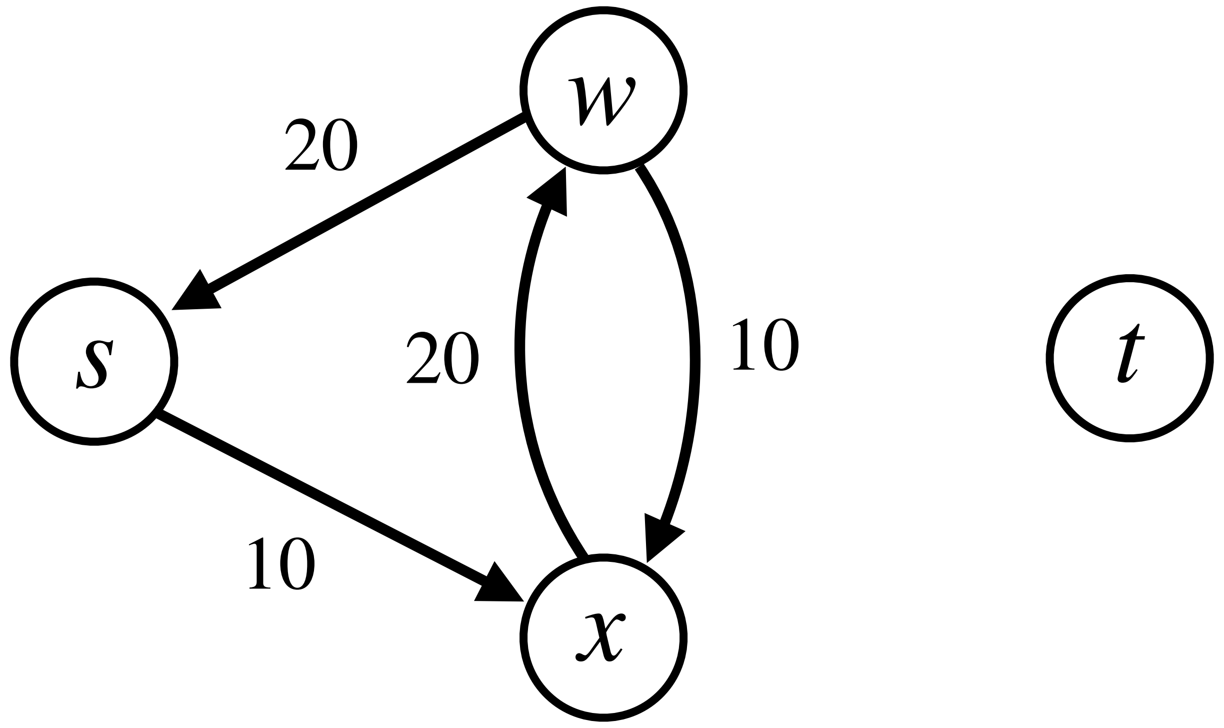
Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

Example:



G



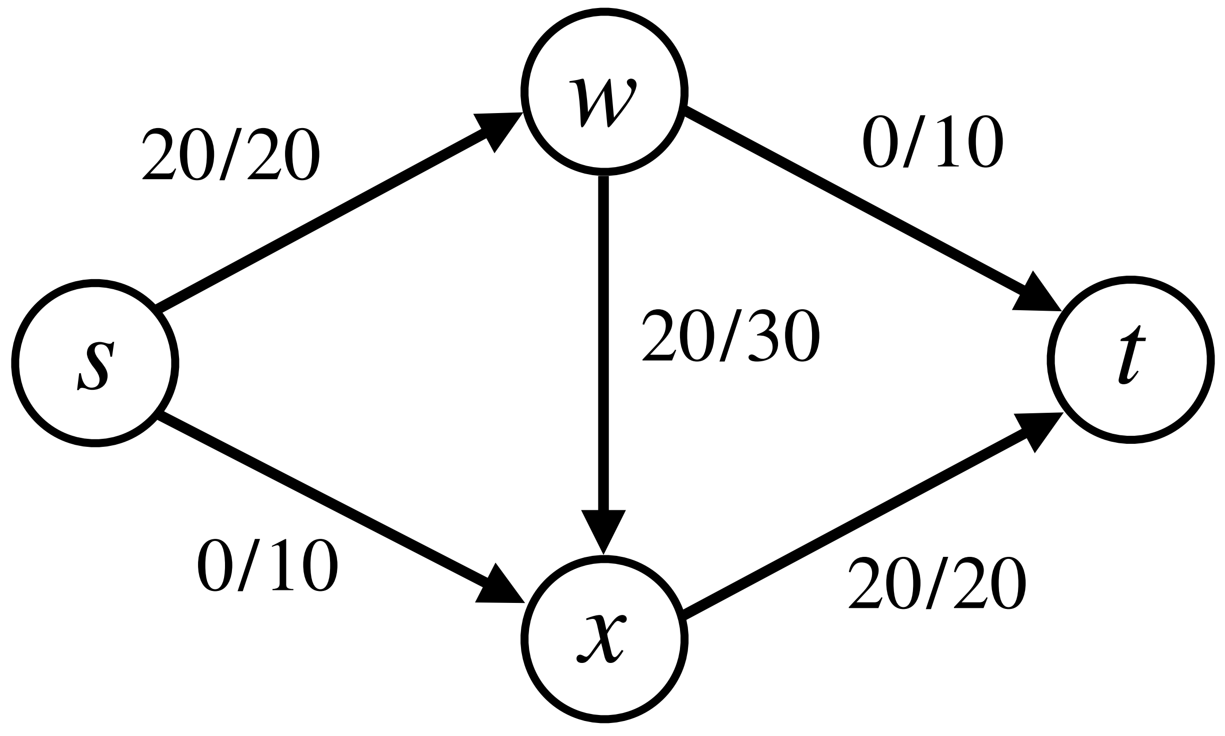
G_f

Residual Networks

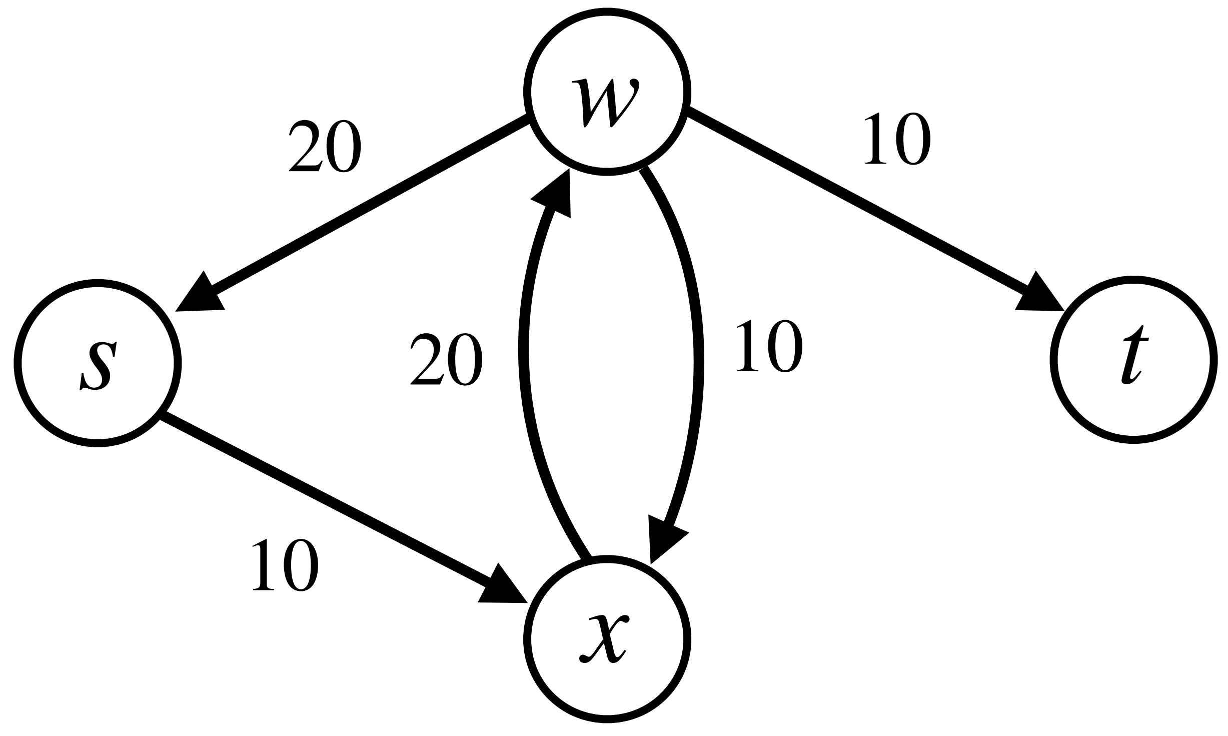
Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

Example:



G



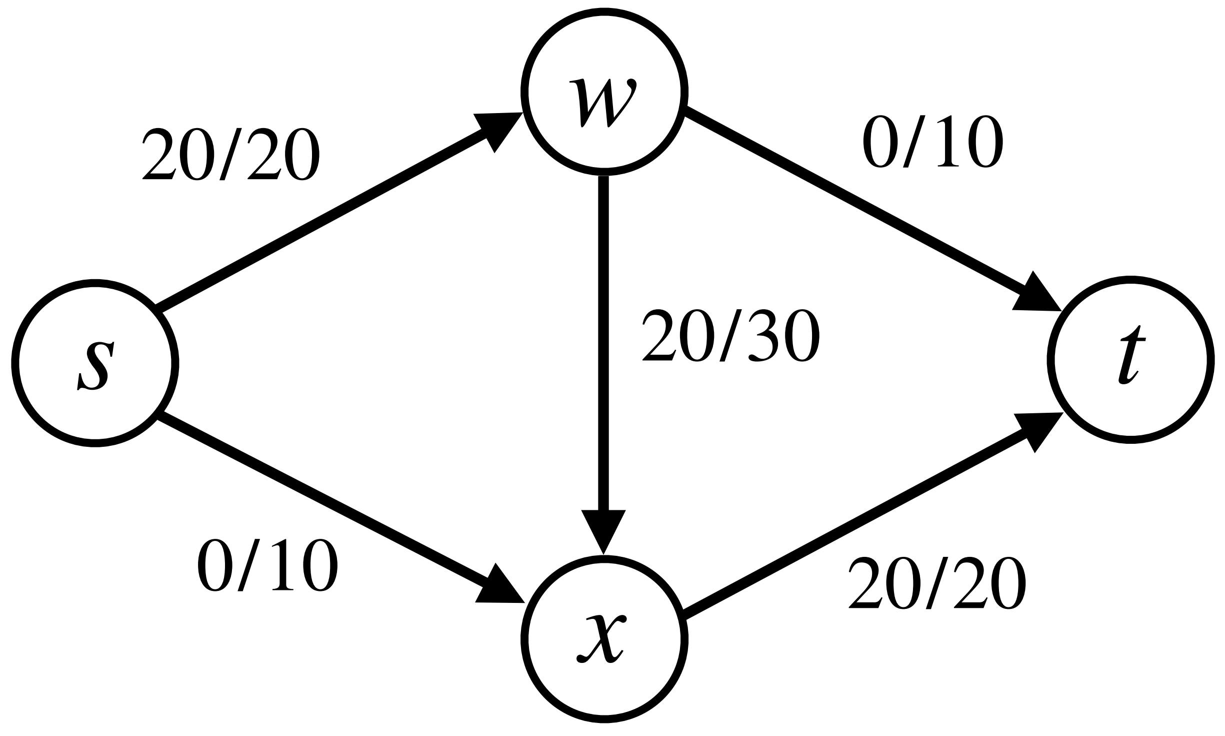
G_f

Residual Networks

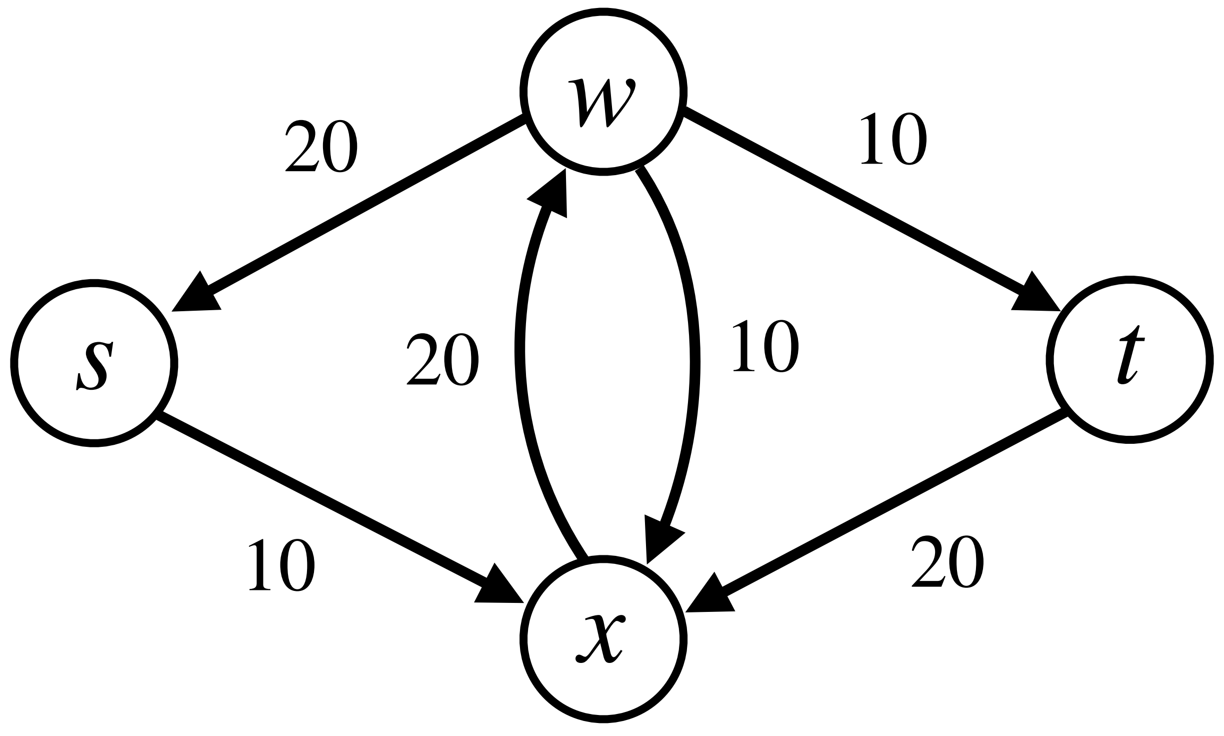
Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

Example:



G



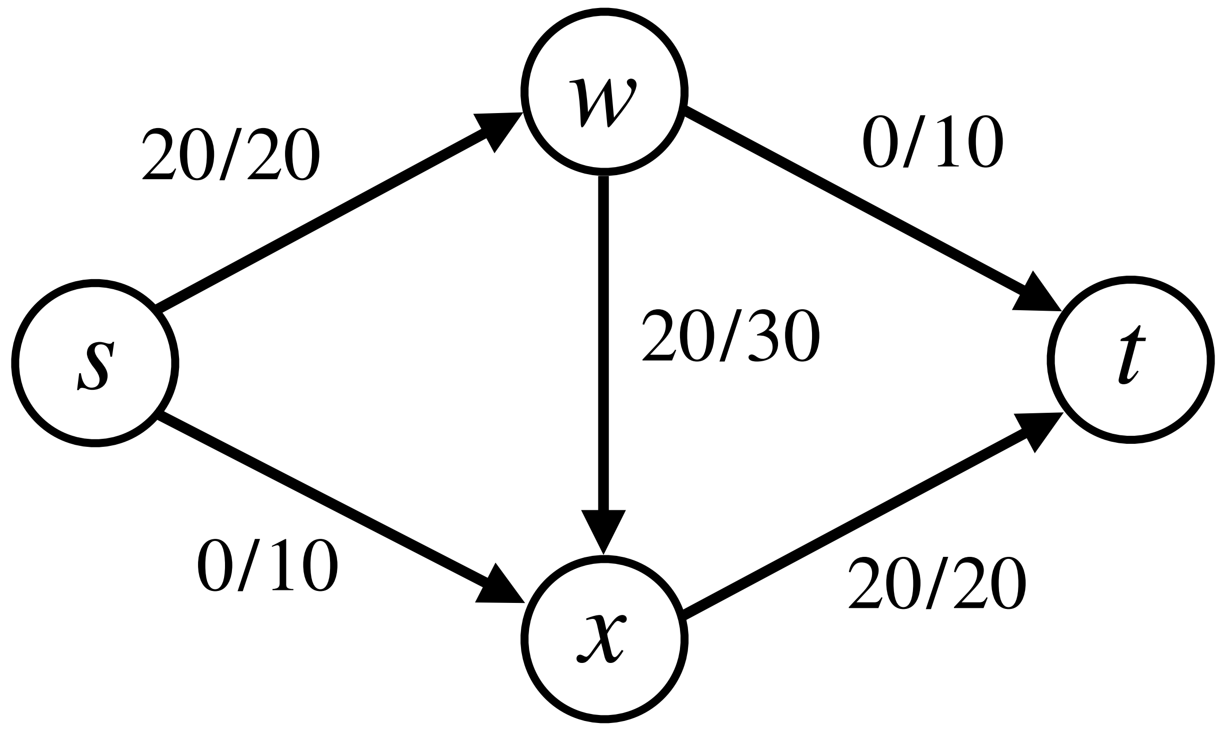
G_f

Residual Networks

Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

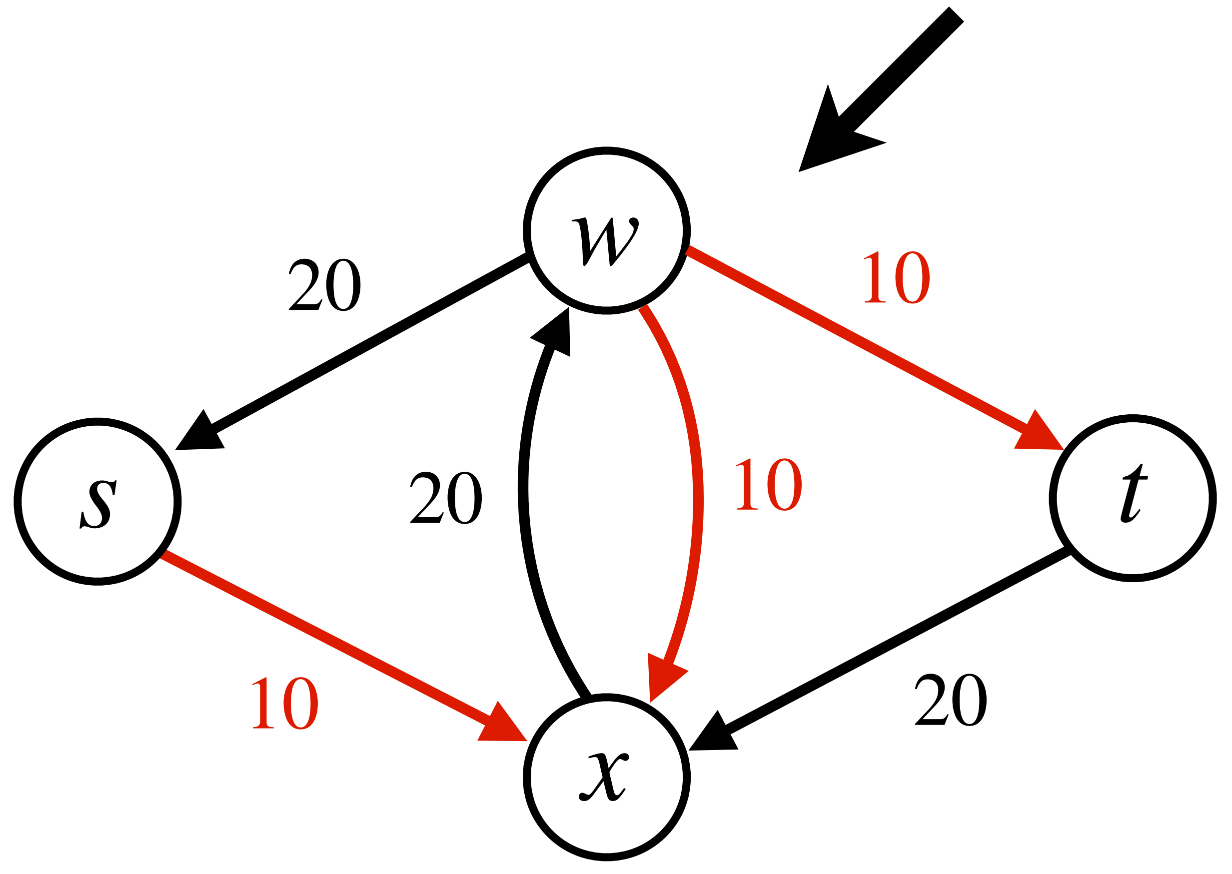
- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

Example:



G

Forward edges to use the remaining capacity



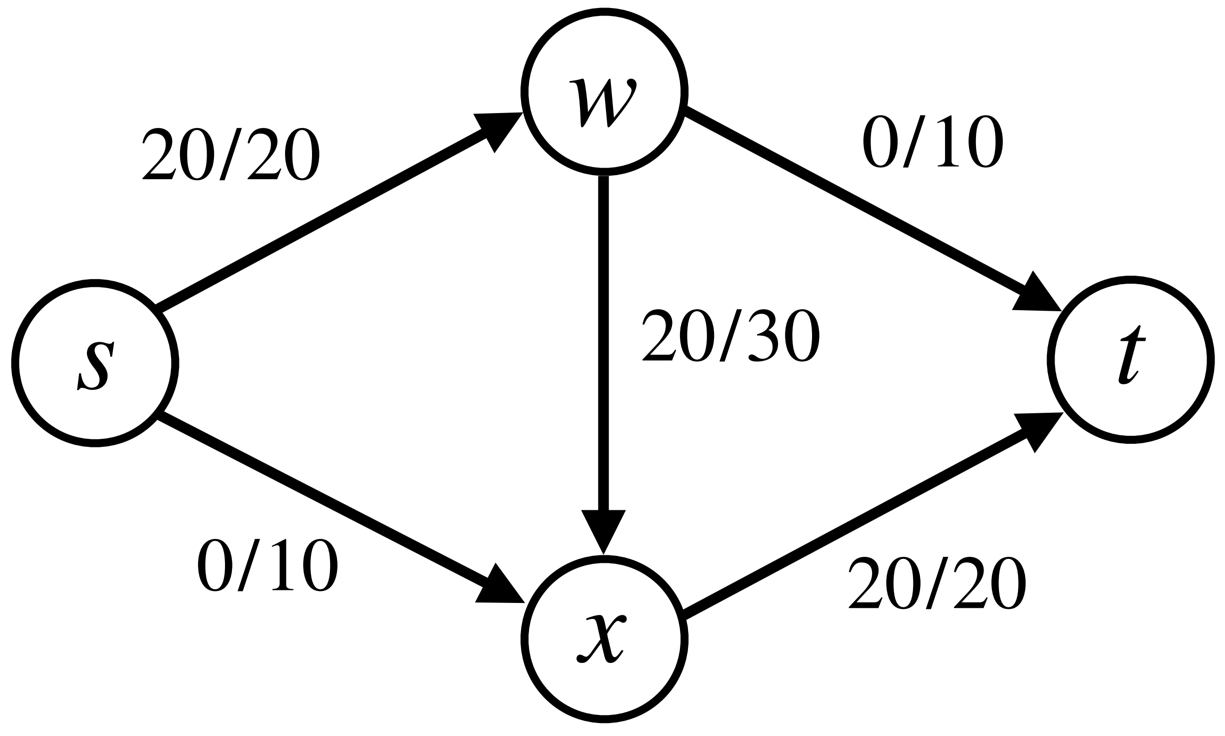
G_f

Residual Networks

Defn: For a given flow network $G = (V, E)$ and flow f, its residual network is $G_f = (V, E_f)$, where for every edge (u, v) in G, E_f contains:

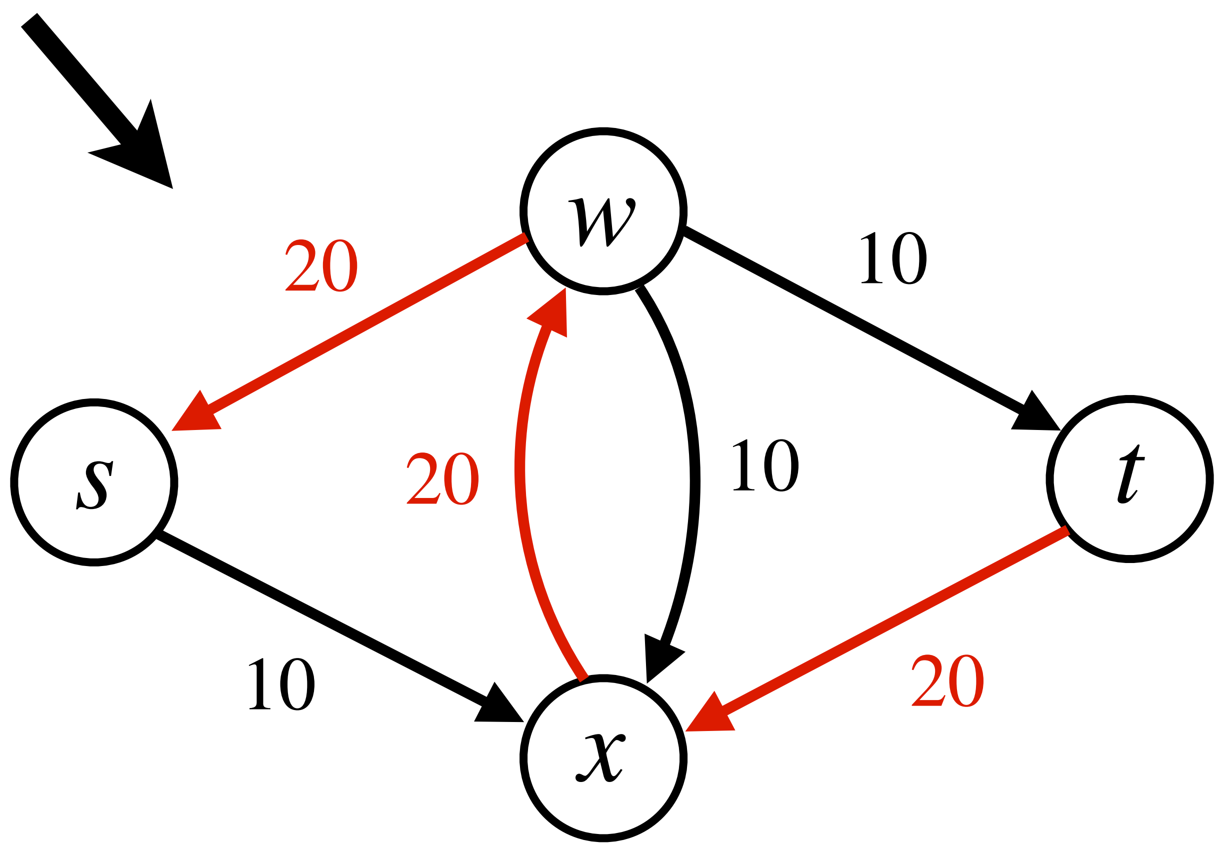
- **Forward edges:** Edge (u, v) with capacity $c_f(u, v) = c(u, v) - f(u, v) > 0$
- **Backward edges:** Edge (v, u) with capacity $c_f(v, u) = f(u, v) > 0$

Example:



G

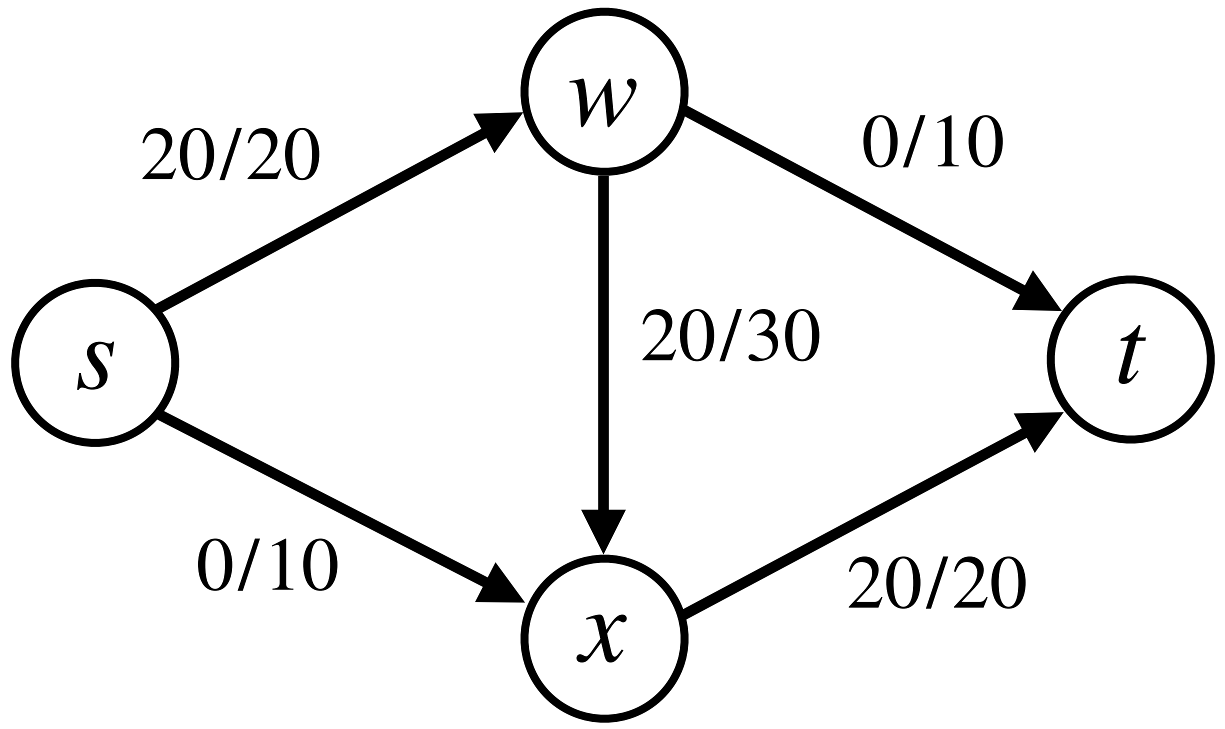
Backward edges to decrease the flow on some edges



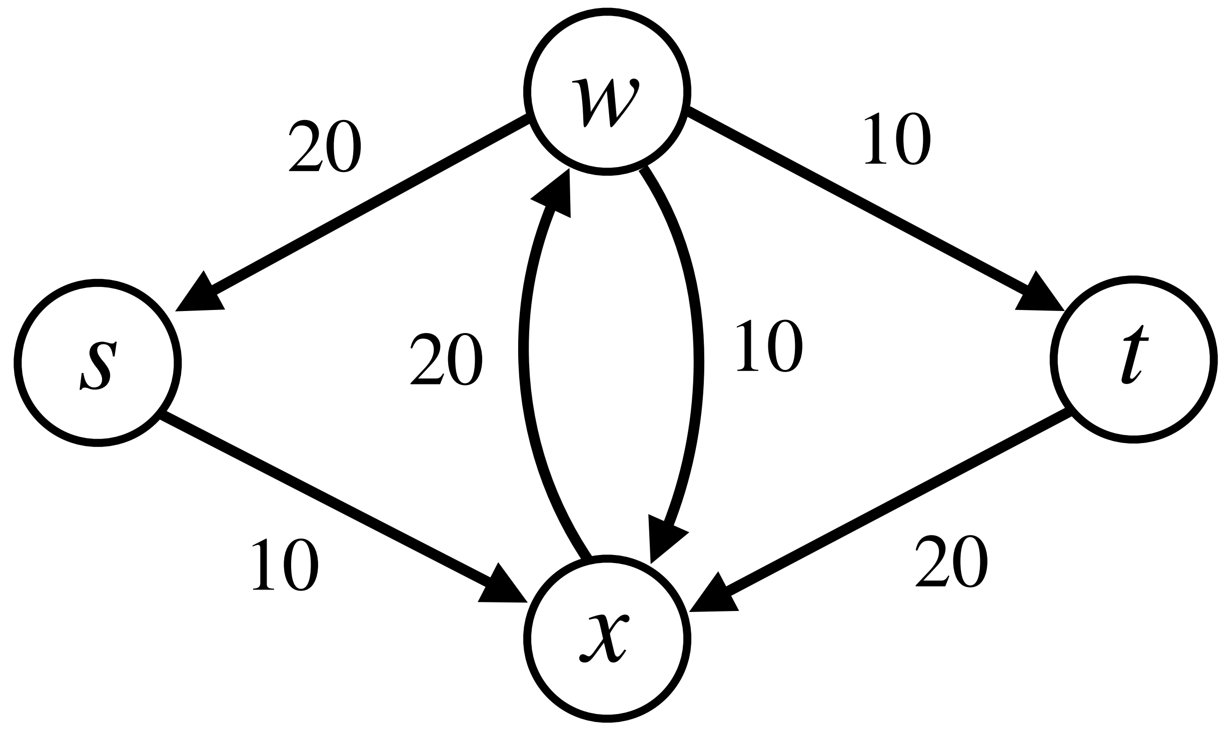
G_f

Augmenting Flows via Residual Networks

Example:



G

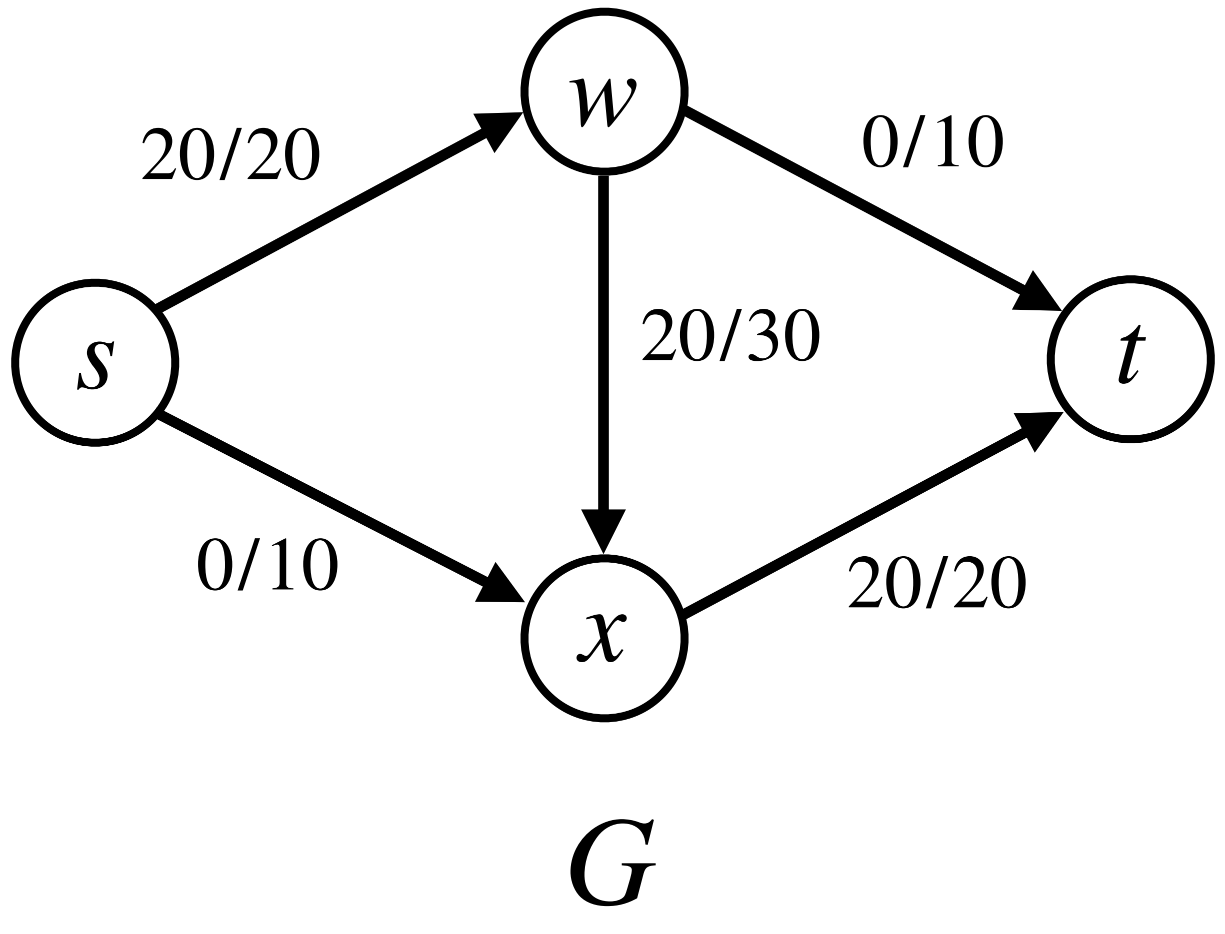


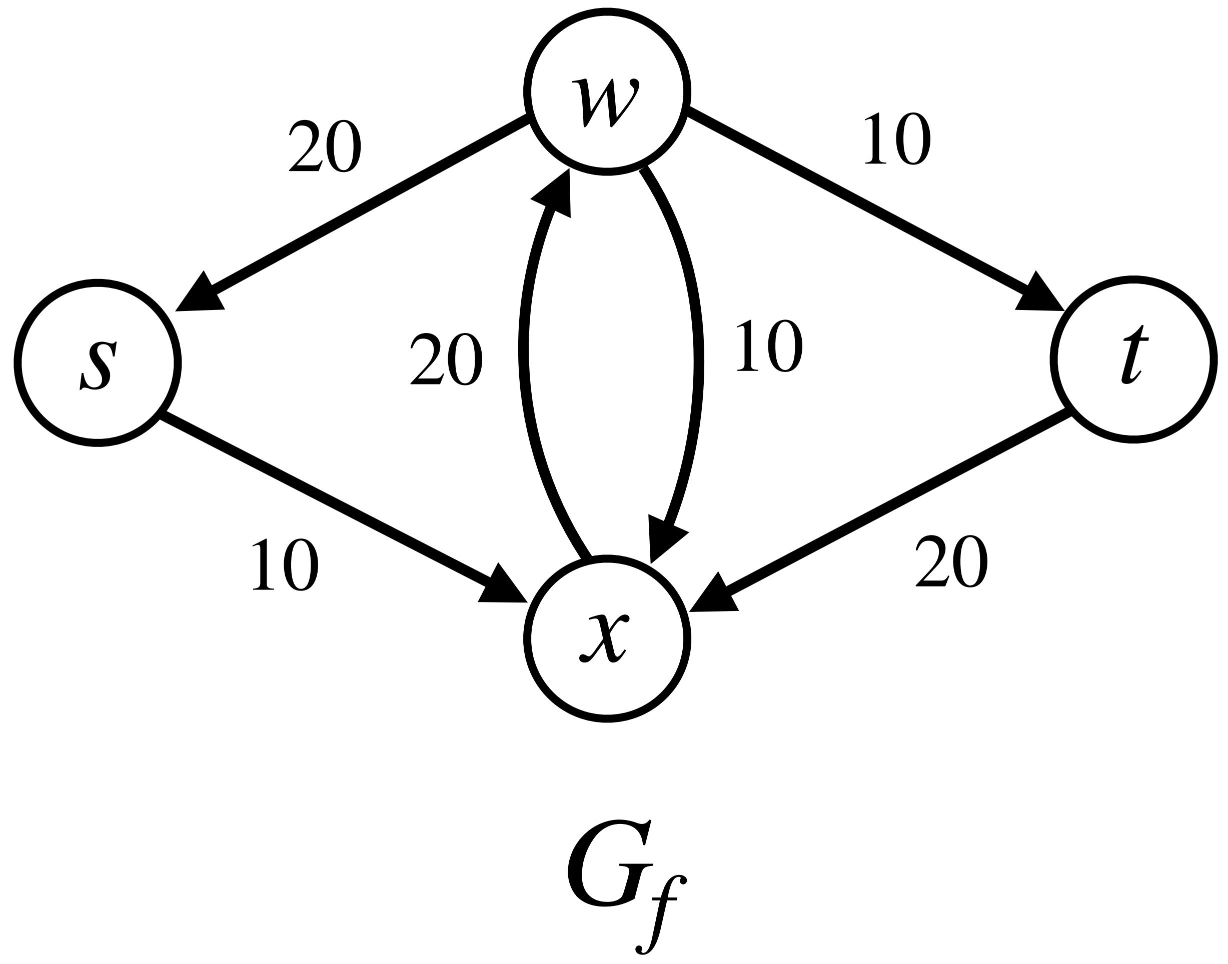
G_f

Augmenting Flows via Residual Networks

- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.

Example:

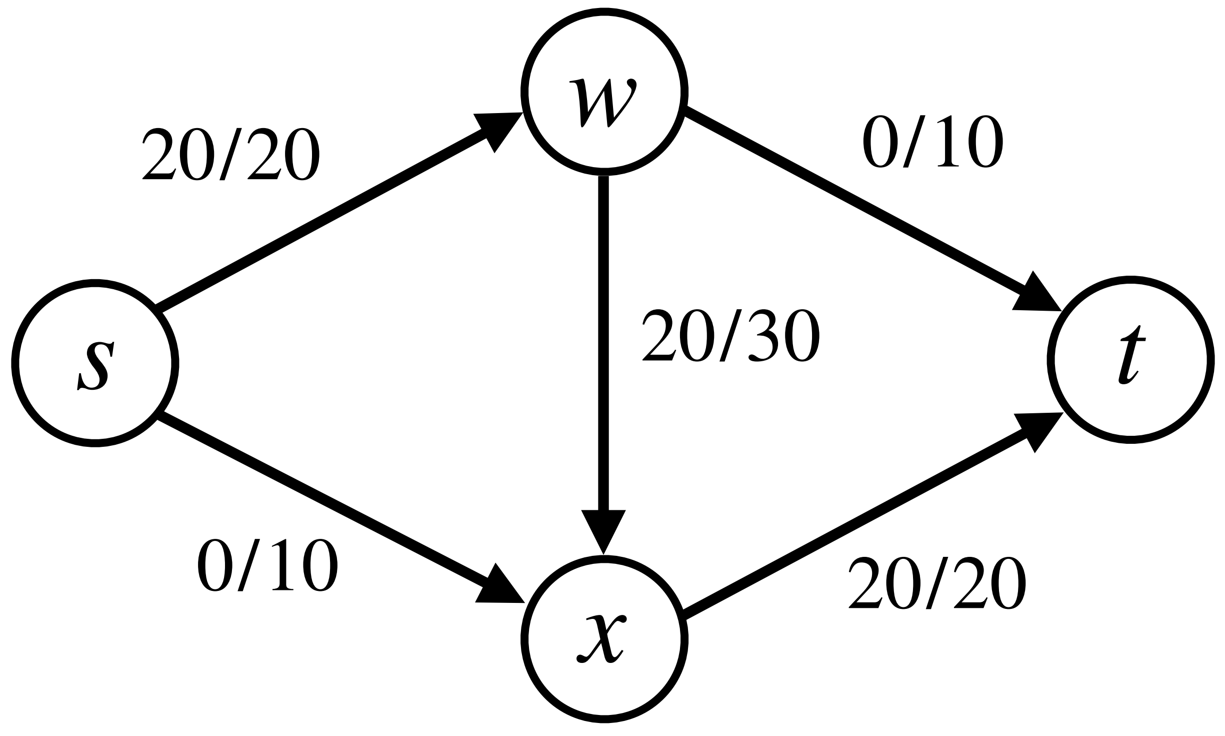




Augmenting Flows via Residual Networks

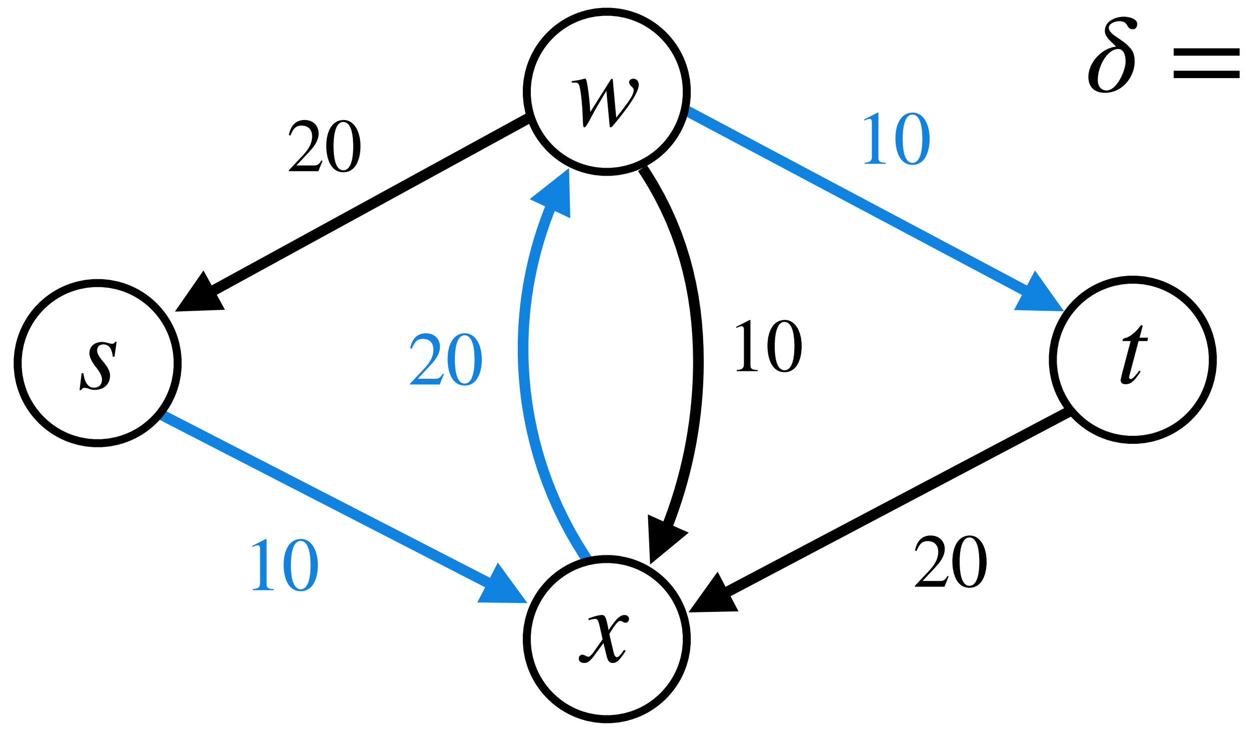
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.

Example:



G

$\delta = 10$

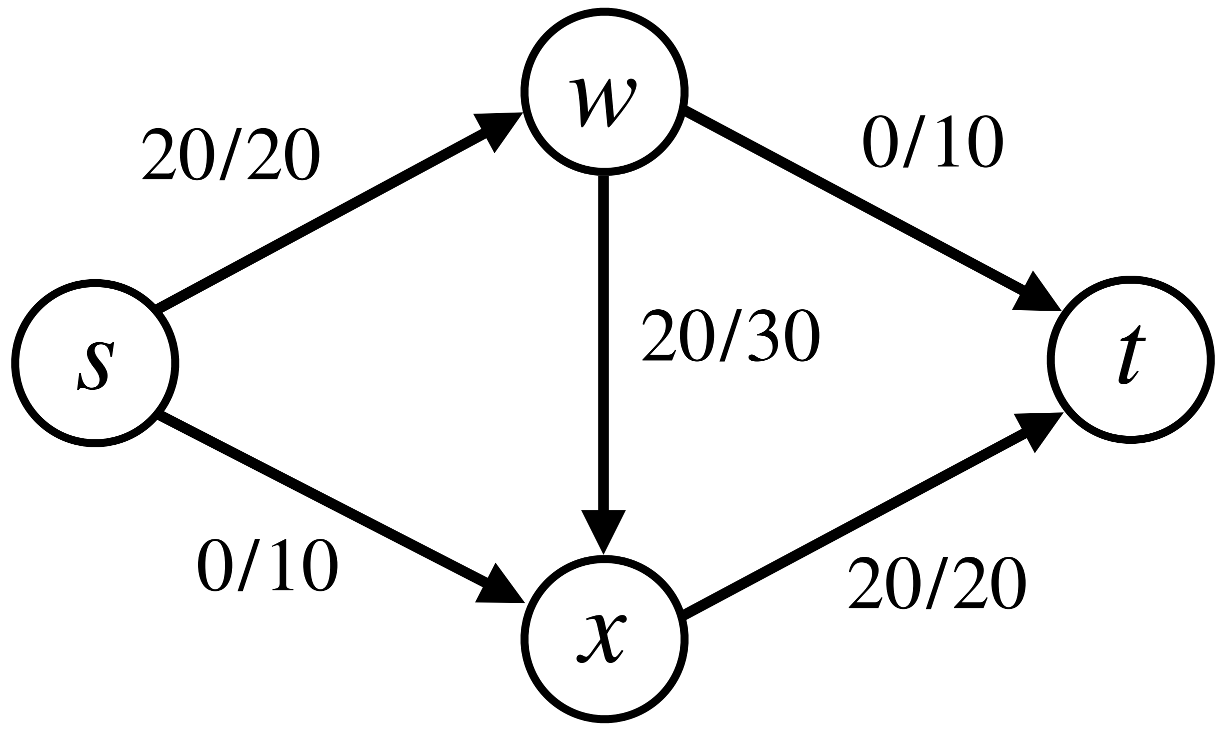


G_f

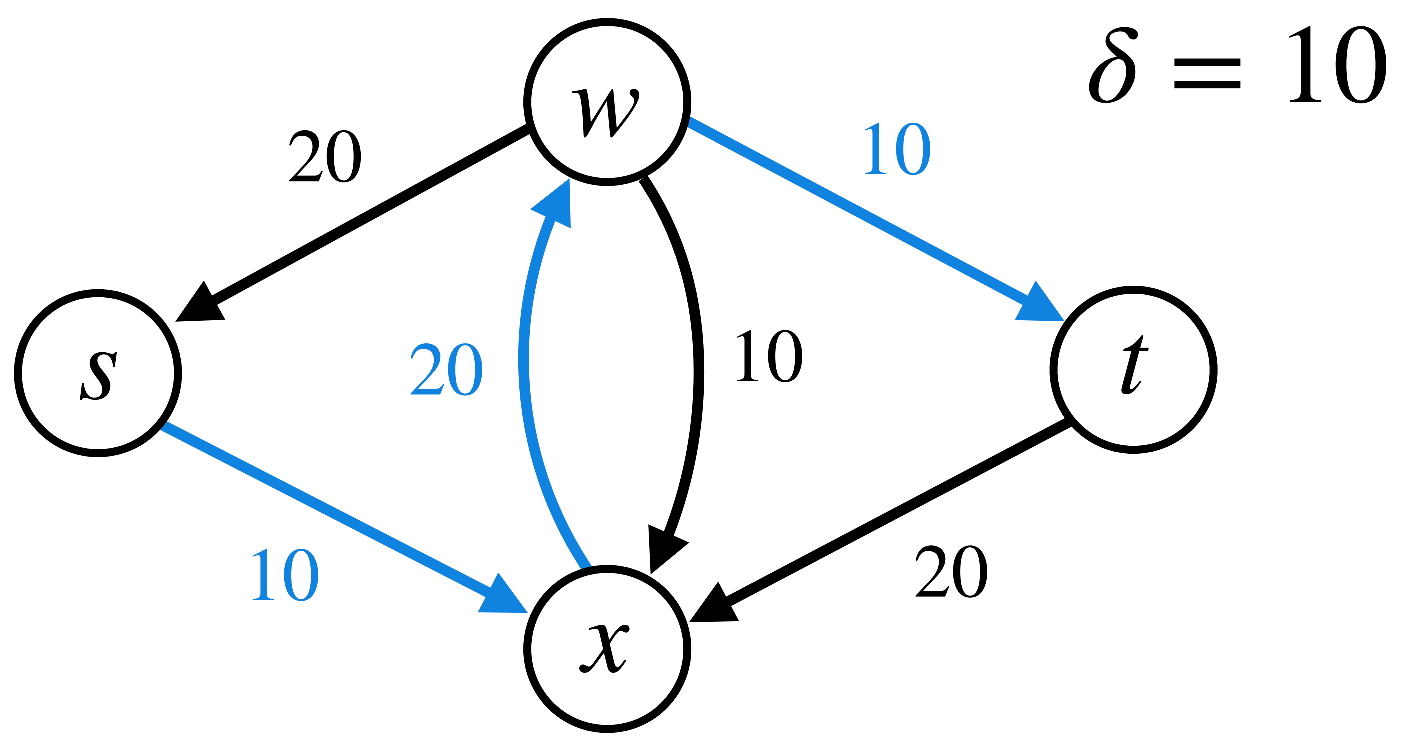
Augmenting Flows via Residual Networks

- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:

Example:



G

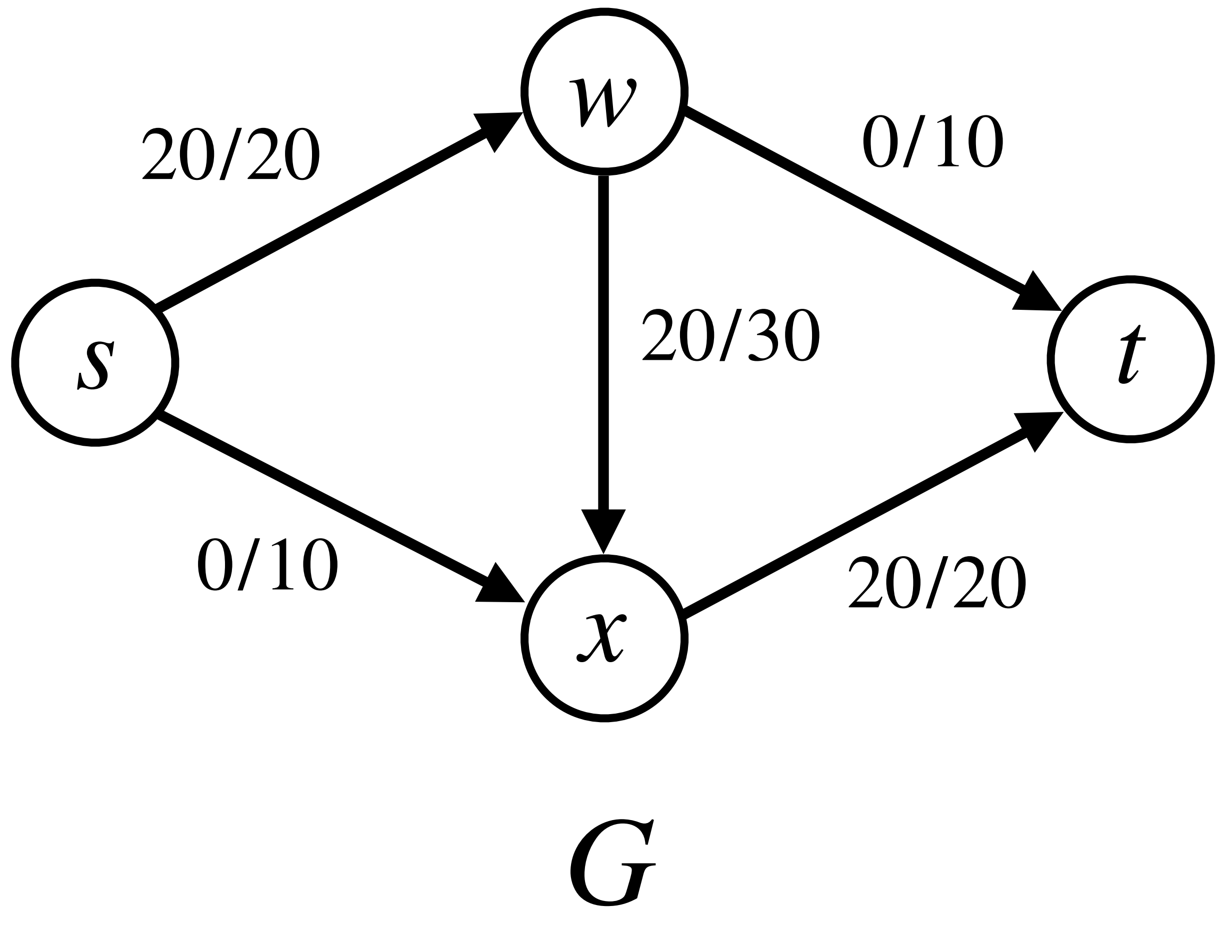


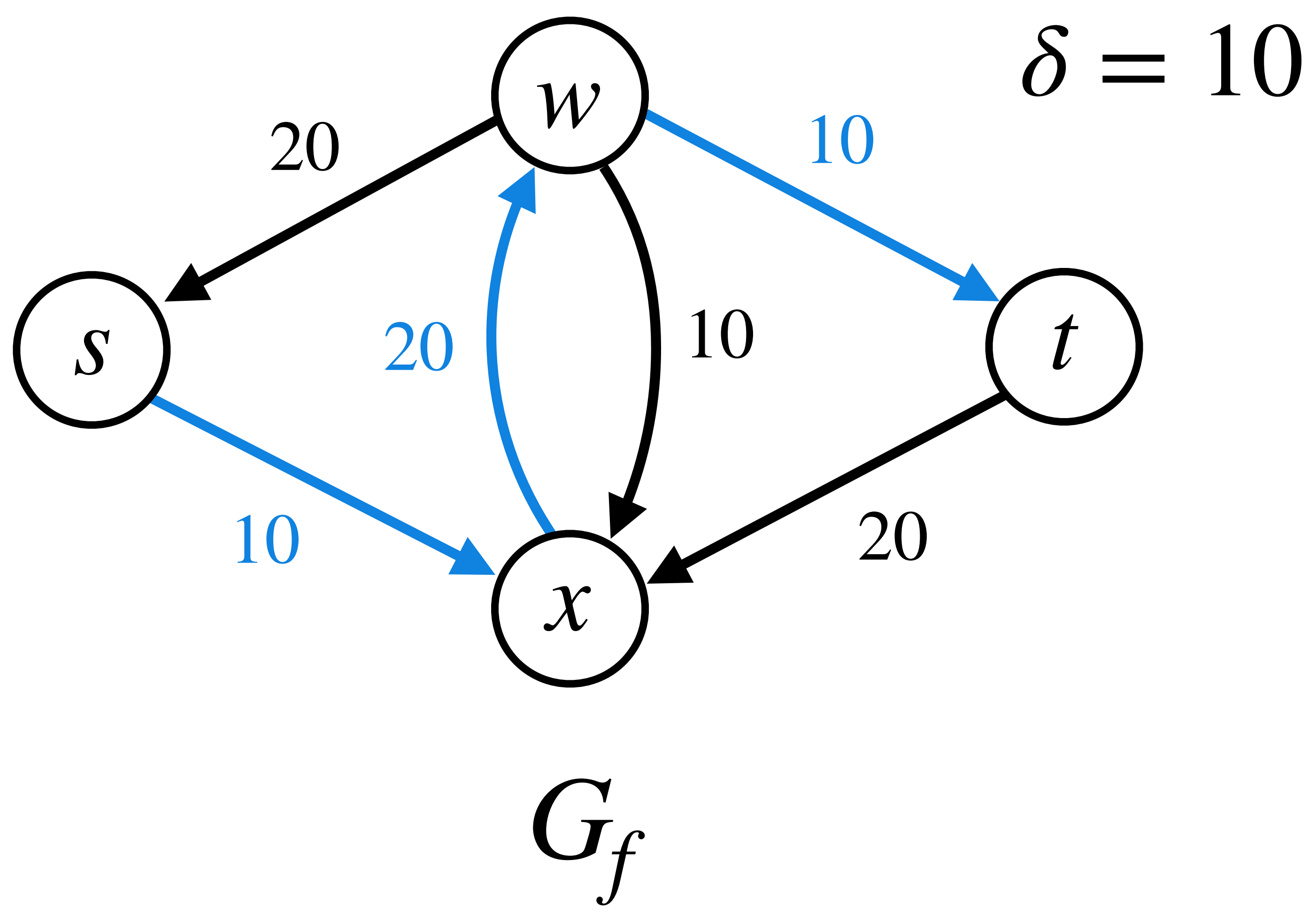
G_f

Augmenting Flows via Residual Networks

- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.

Example:

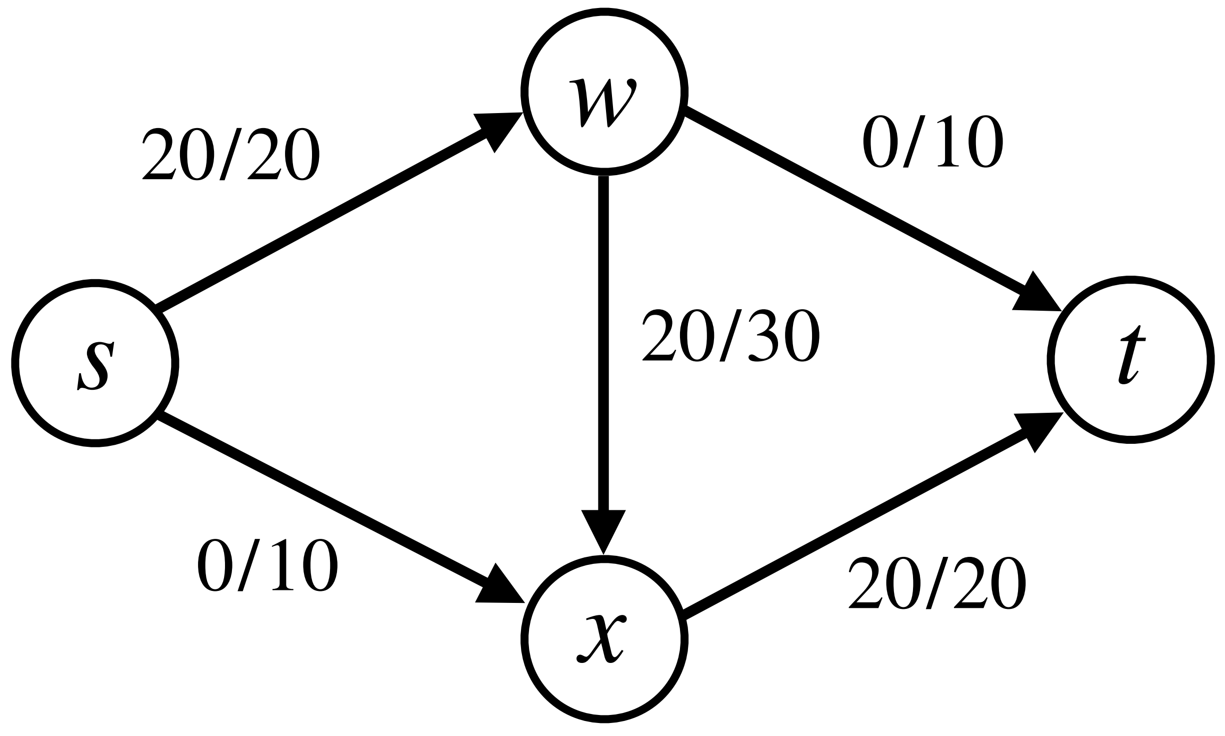




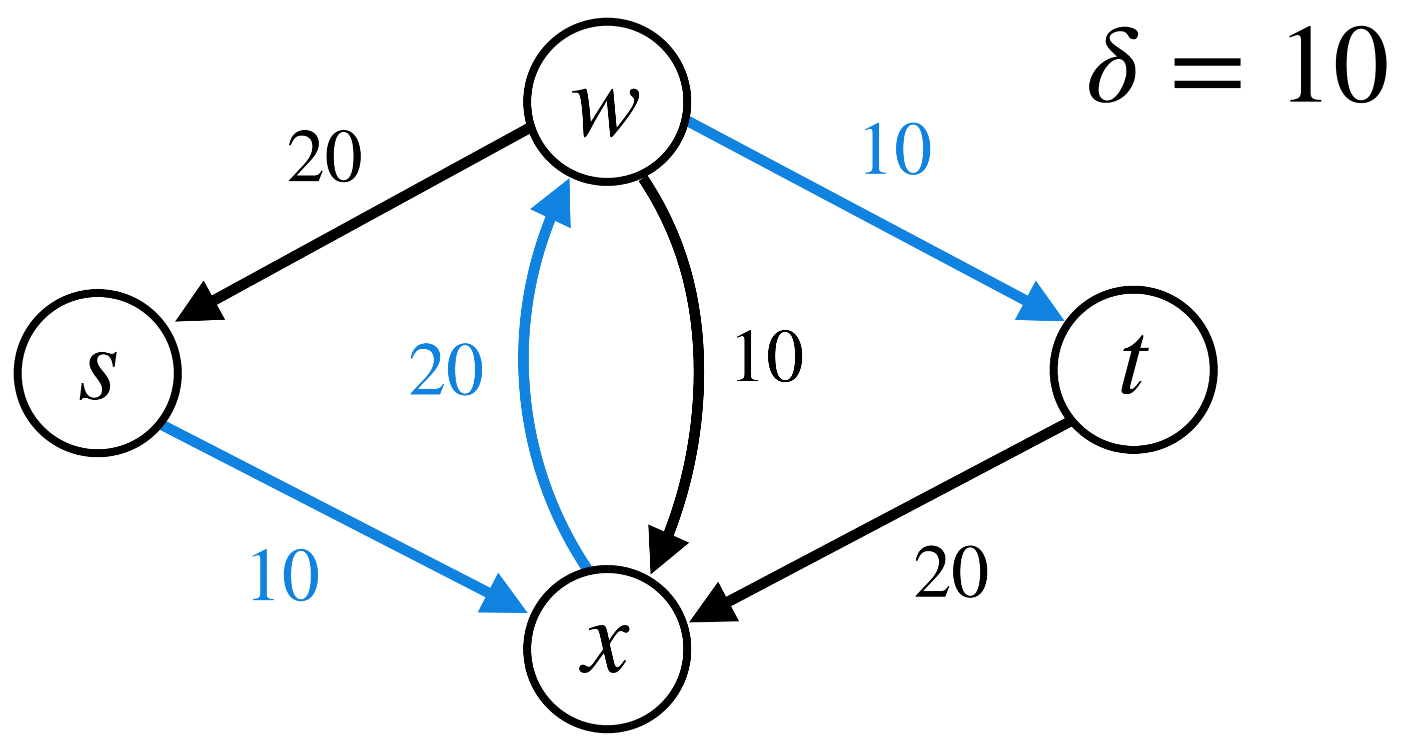
Augmenting Flows via Residual Networks

- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Example:



G

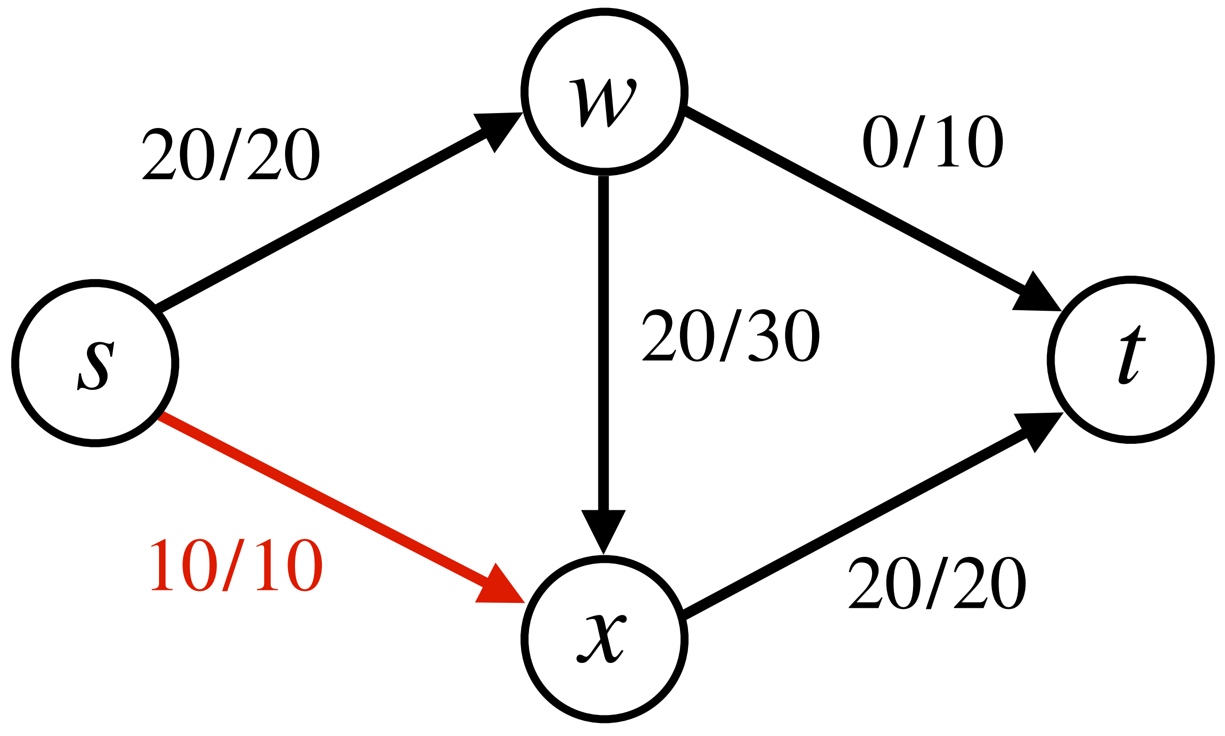


G_f

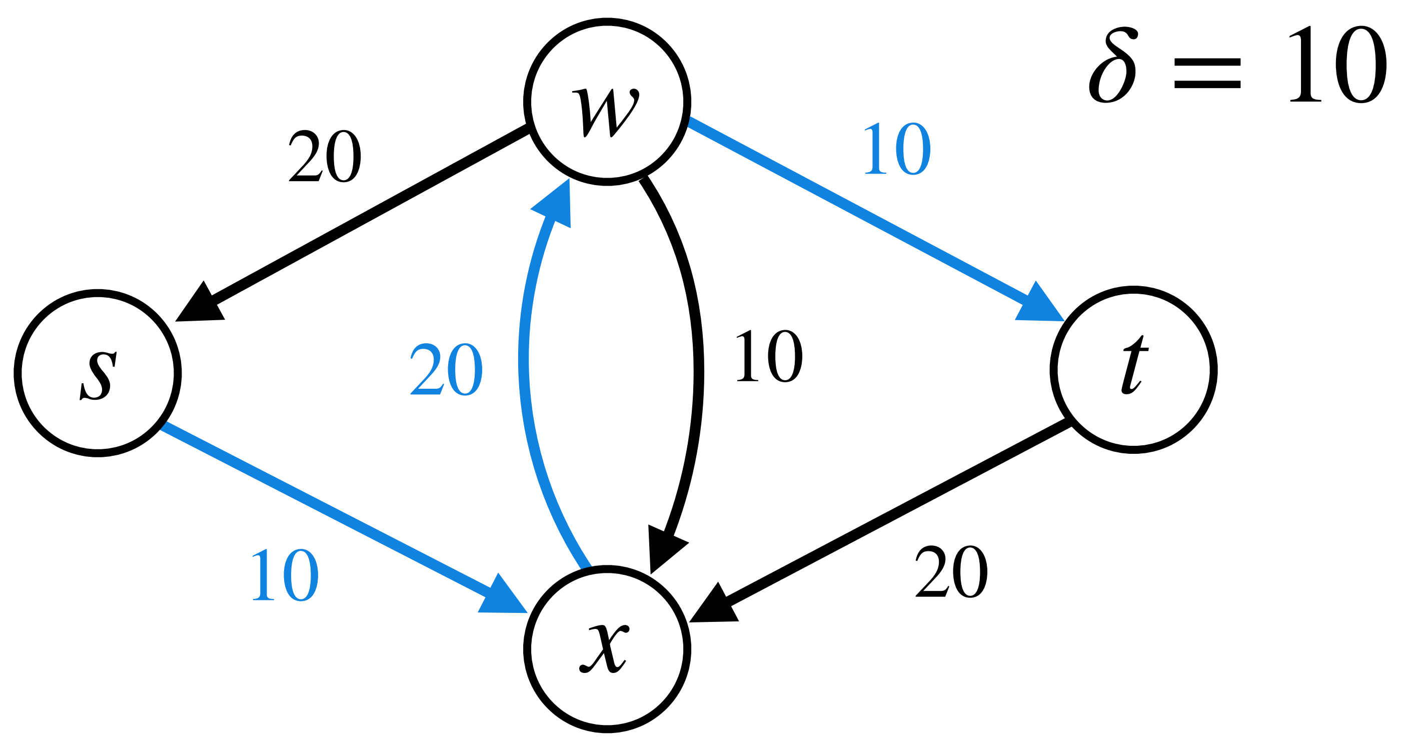
Augmenting Flows via Residual Networks

- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Example:



G

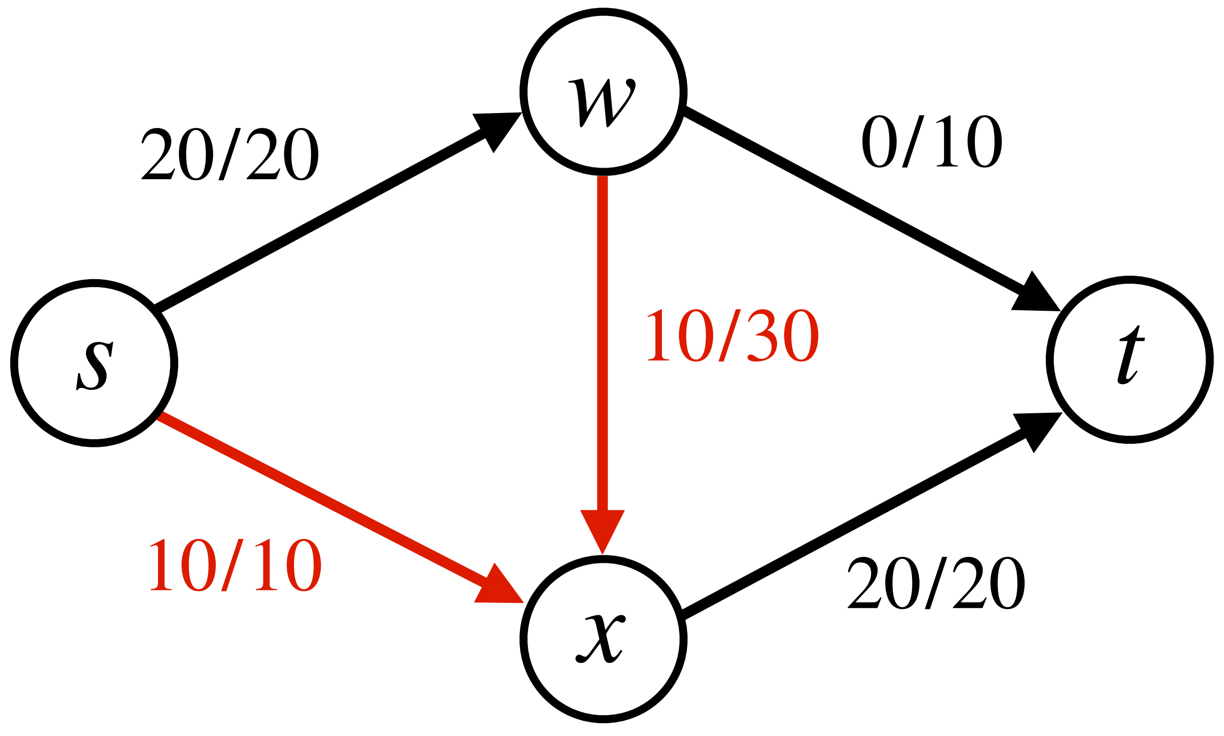


G_f

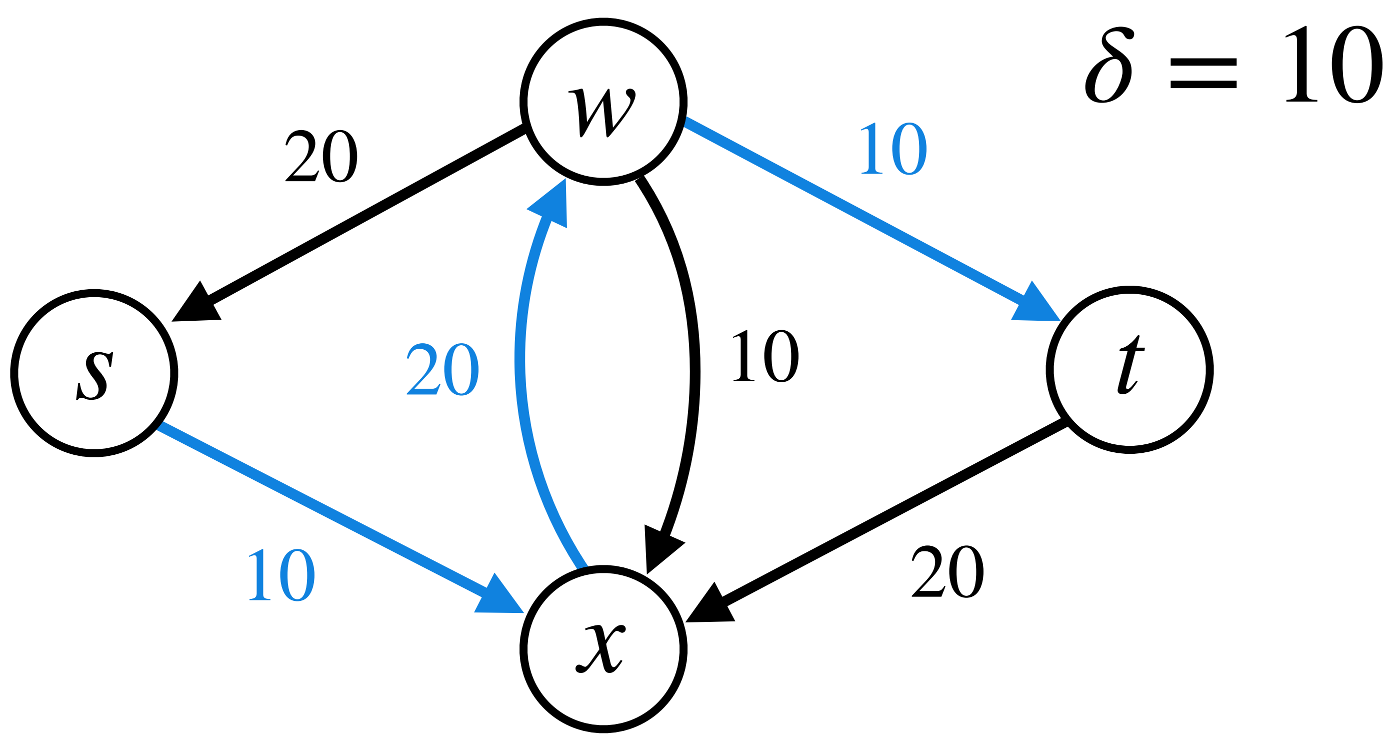
Augmenting Flows via Residual Networks

- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Example:



G

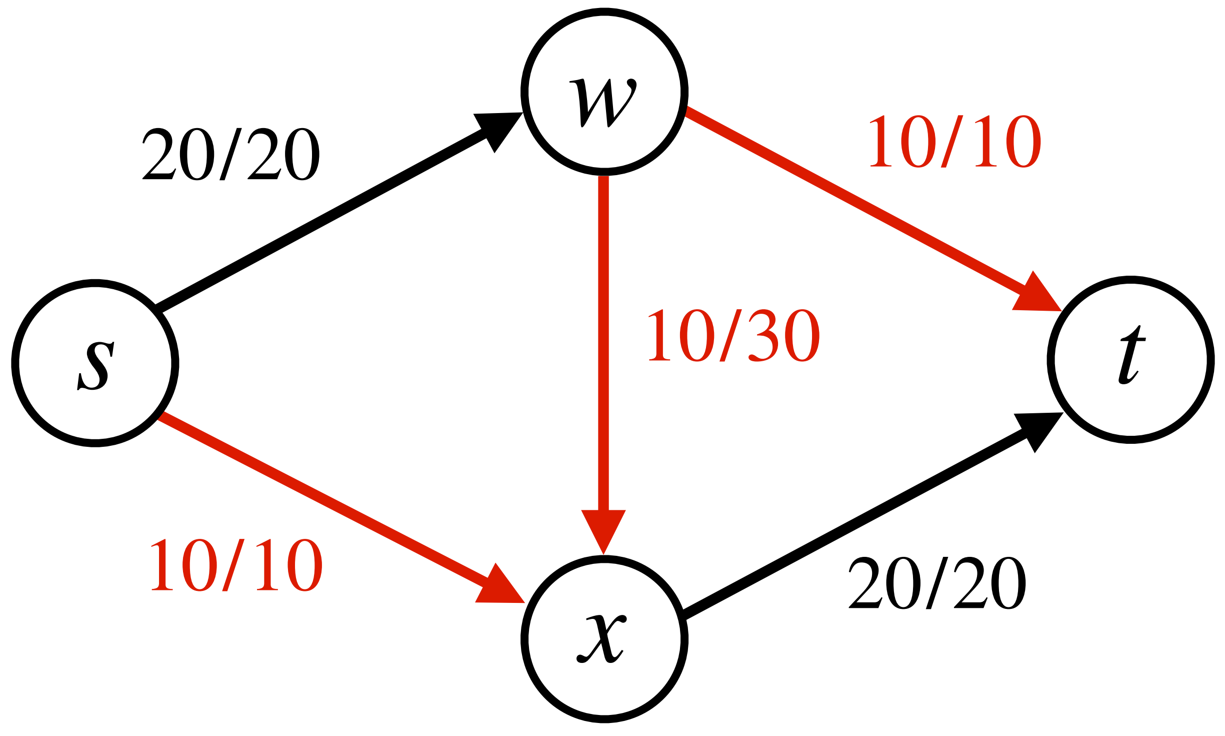


G_f

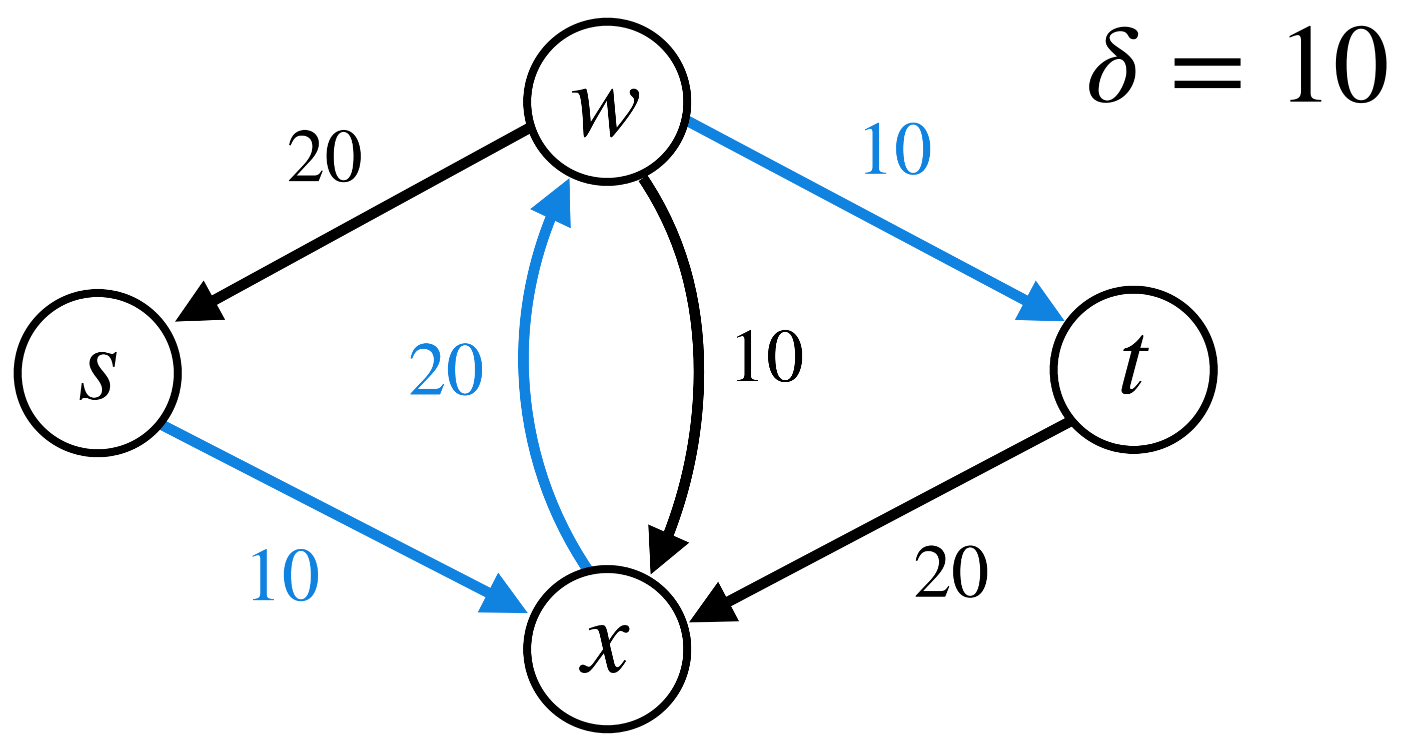
Augmenting Flows via Residual Networks

- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Example:



G



G_f

Augmenting Flows via Residual Networks

- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

What about capacity, conservation constraints?

Example: