Lecture 18

Ford-Fulkerson Method (contd.), Max-Flow Min-Cut Theorem

Source: Introduction to Algorithms, CLRS and Kleinberg & Tardos

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.

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.

Note: Path P is called an augmenting path.

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.

Note: Path P is called an augmenting path.

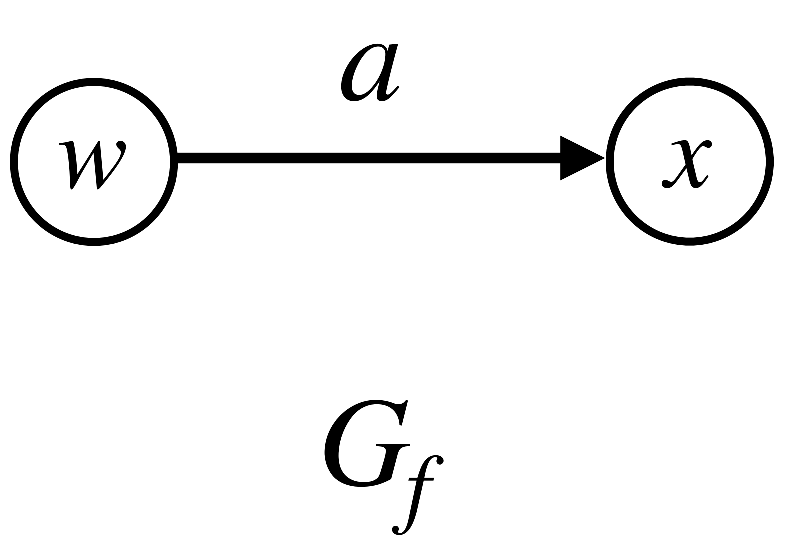
Satisfying Capacity Constraint:

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.

Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:

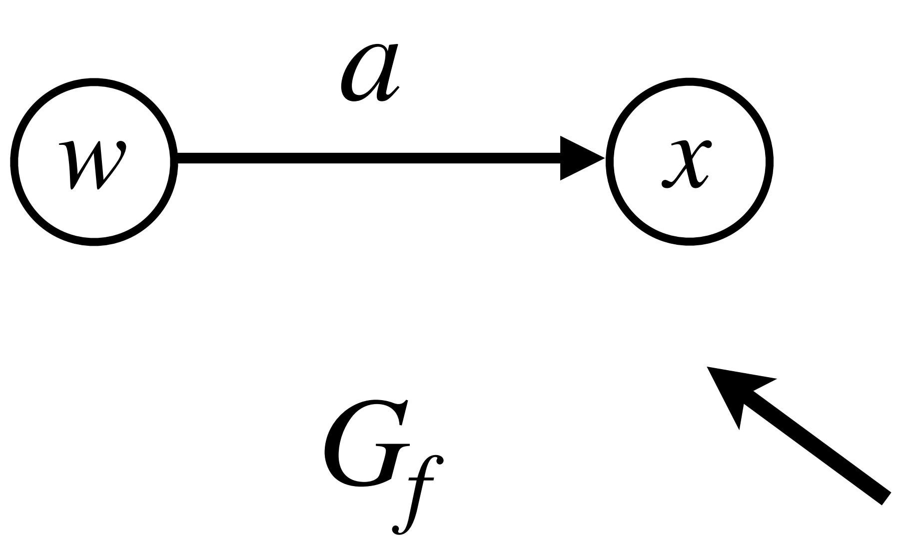


Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:



A forward edge
in the augmenting path

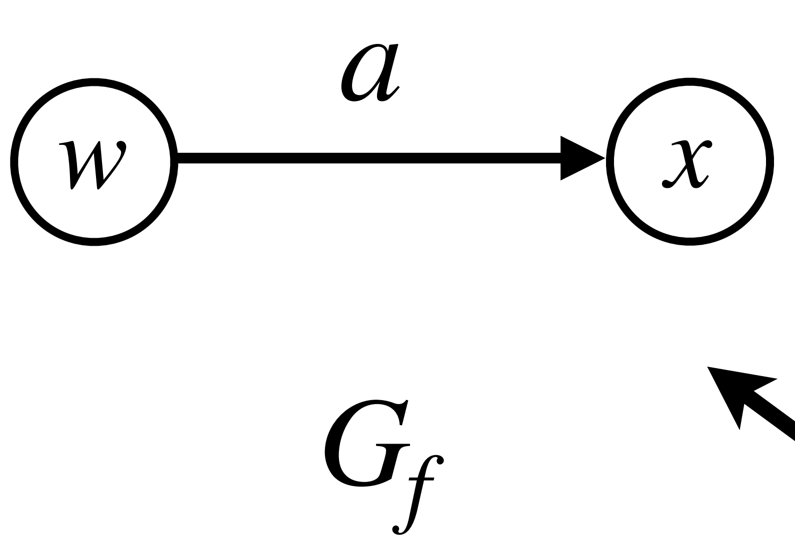
Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:

$\delta \leq a$



A forward edge in the augmenting path

Augmenting Flows via Residual Networks

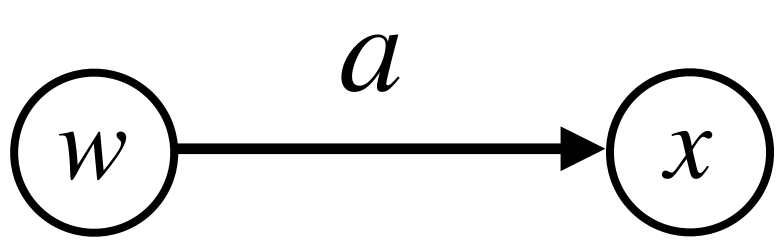
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:

$\delta \leq a$

G



G_f

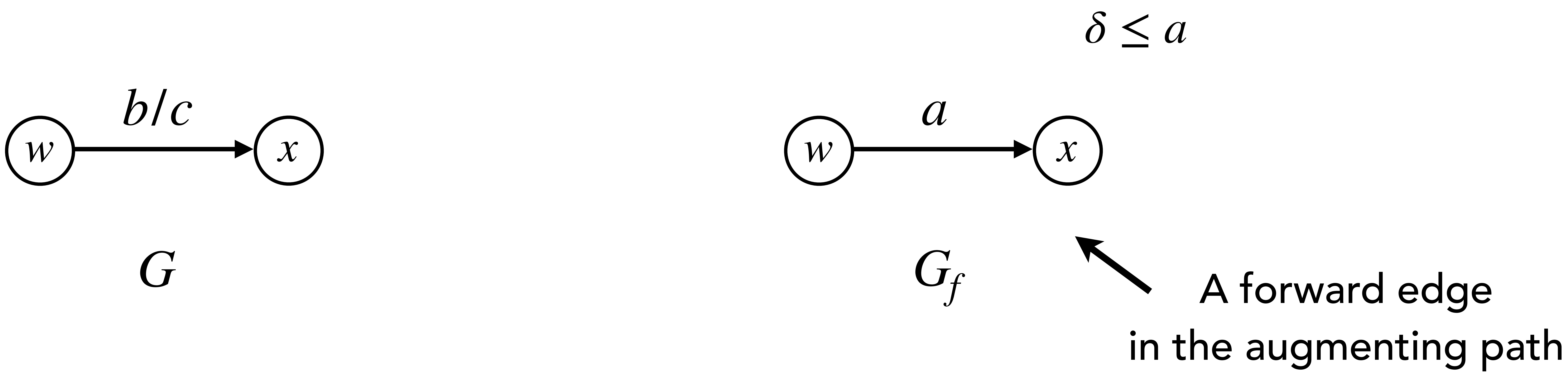
A forward edge in the augmenting path

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.

Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:

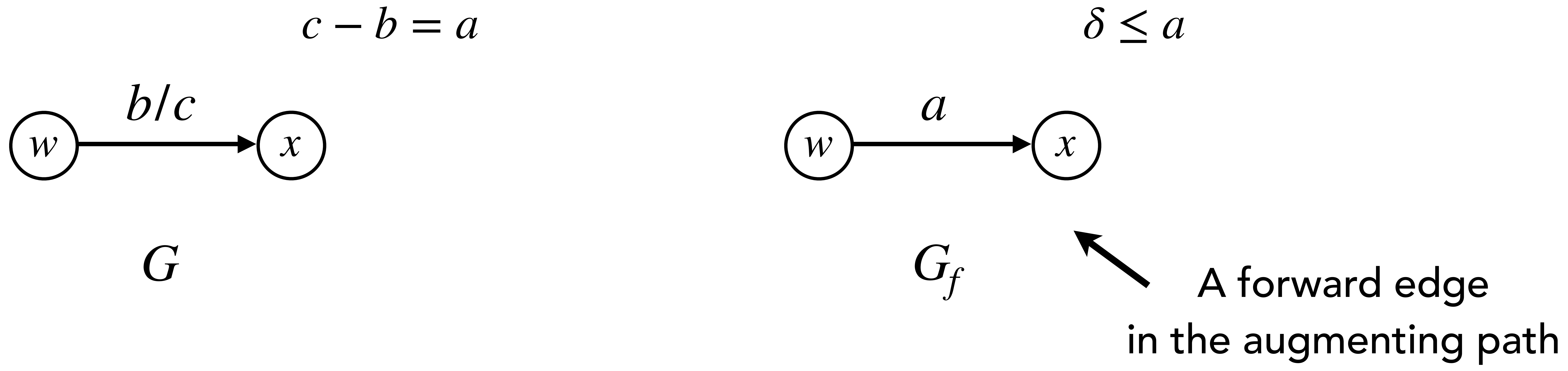


Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:

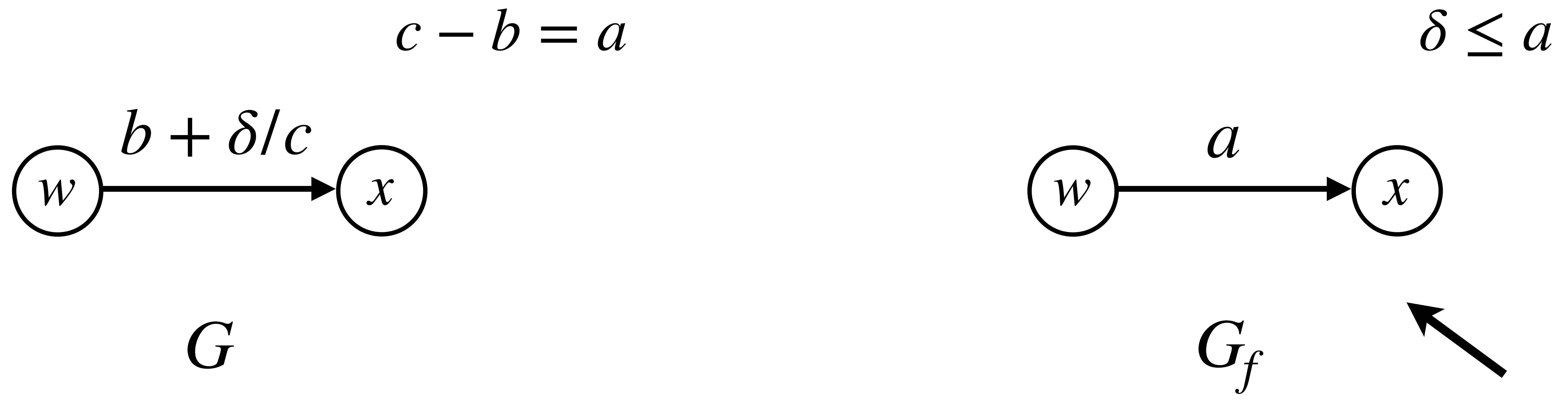


Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:



A forward edge
in the augmenting path

Augmenting Flows via Residual Networks

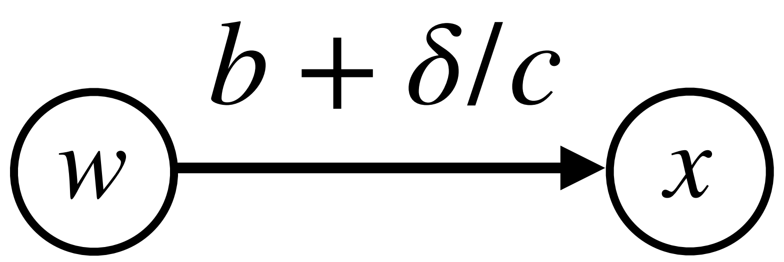
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

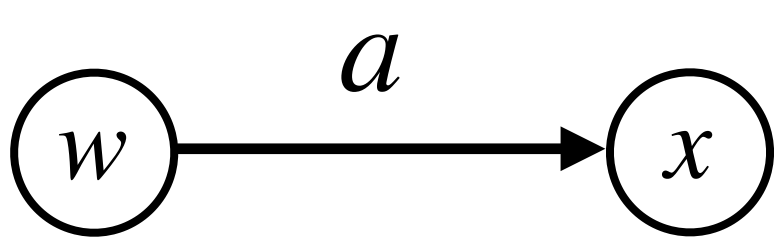
Satisfying Capacity Constraint:

$c - b = a,\ b + \delta \leq c$

$\delta \leq a$



G



G_f

A forward edge
in the augmenting path

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.

Note: Path P is called an augmenting path.

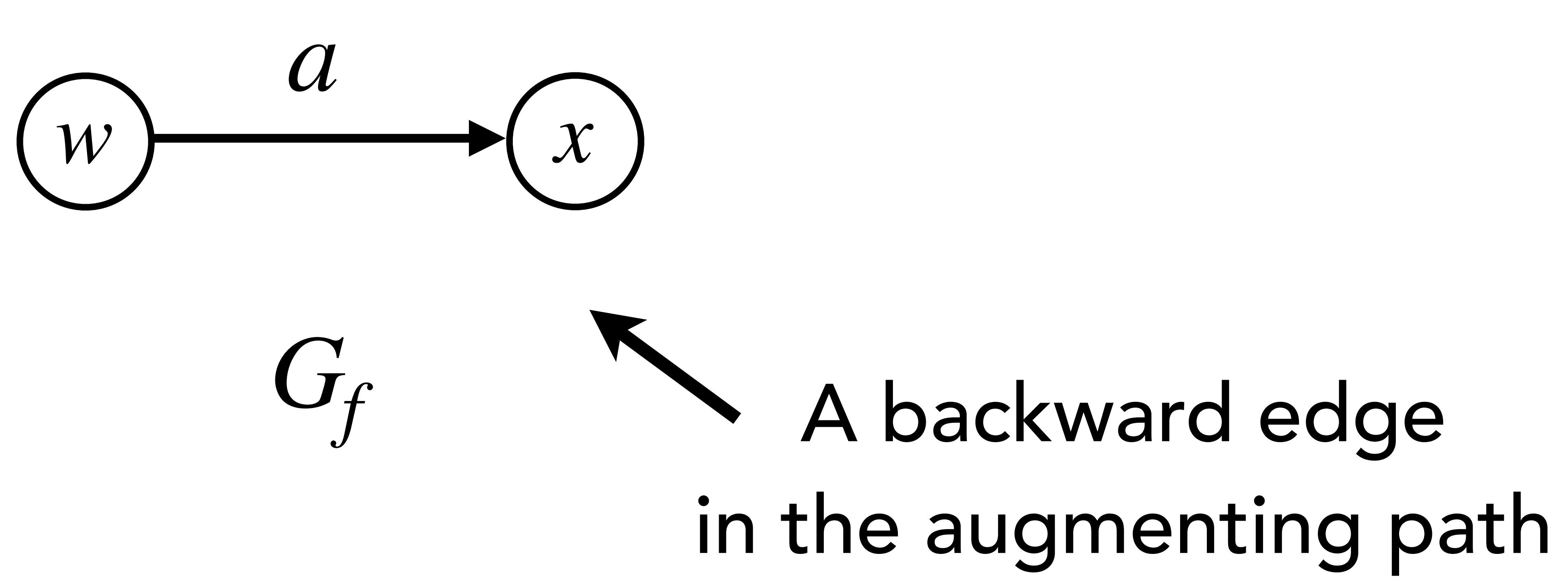
Satisfying Capacity Constraint:

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.

Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:



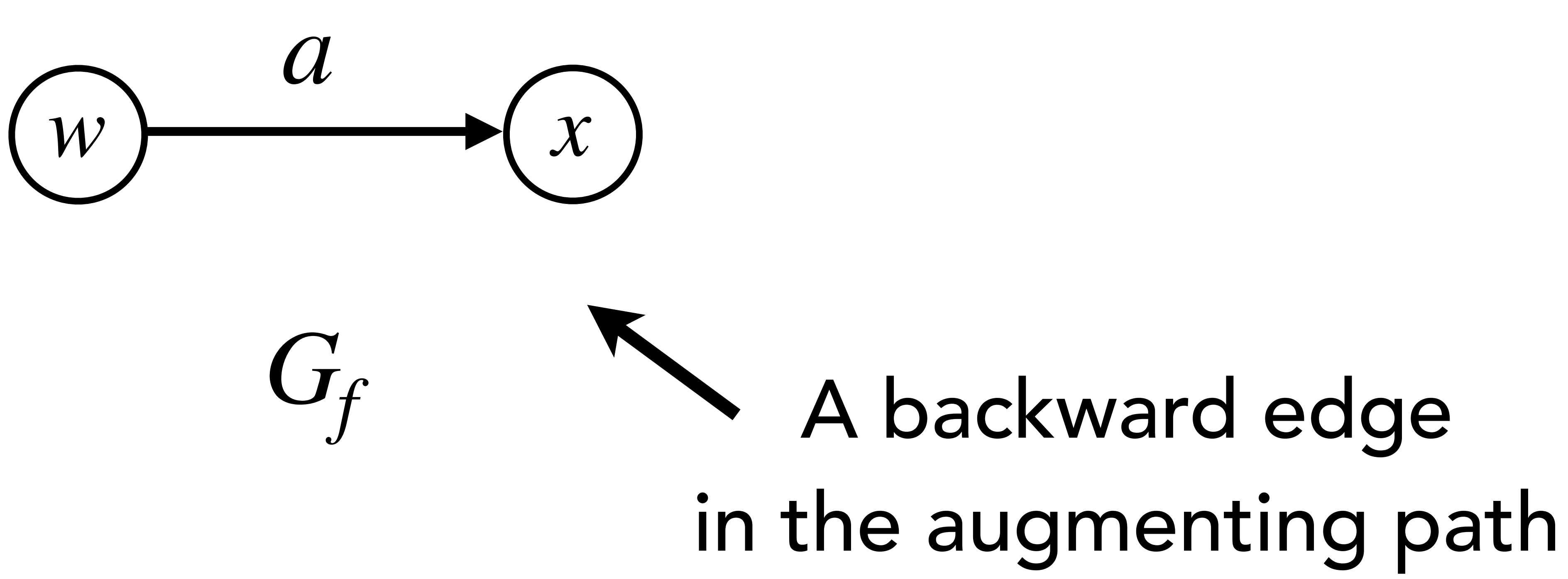
Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:

$\delta \leq a$

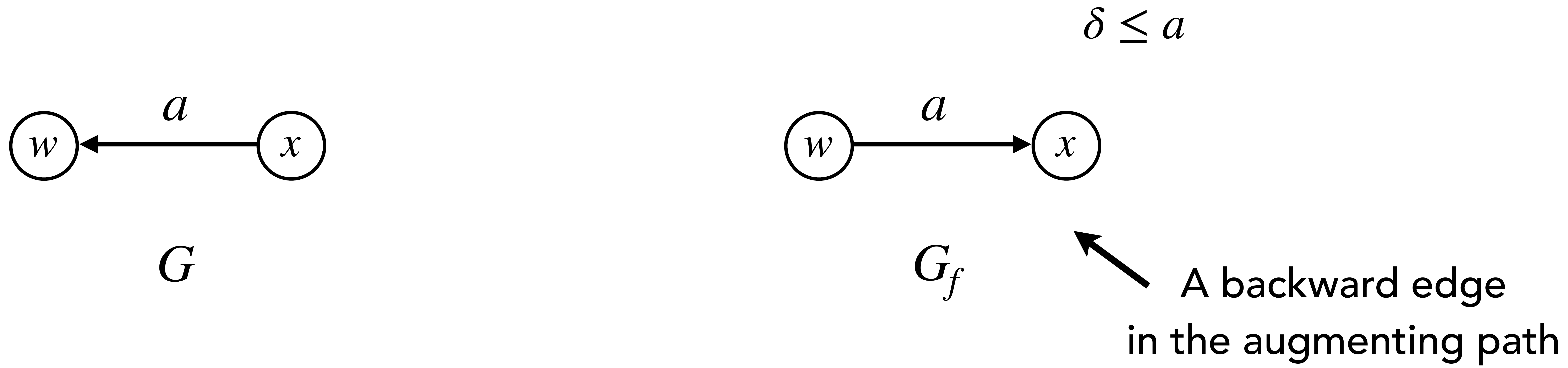


Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:



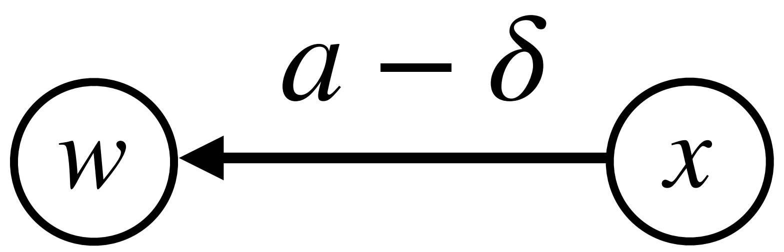
Augmenting Flows via Residual Networks

- 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.

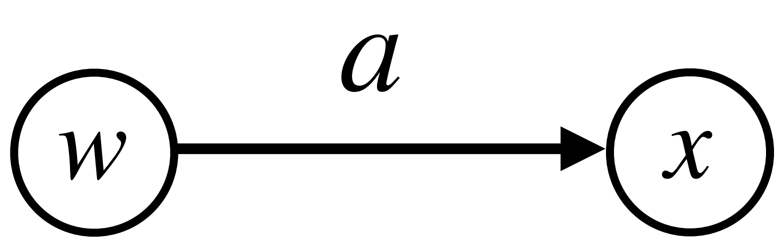
Note: Path P is called an augmenting path.

Satisfying Capacity Constraint:

$\delta \leq a$



G



G_f

A backward edge in the augmenting path

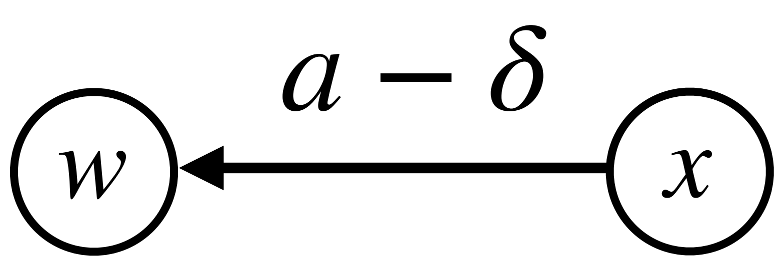
Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

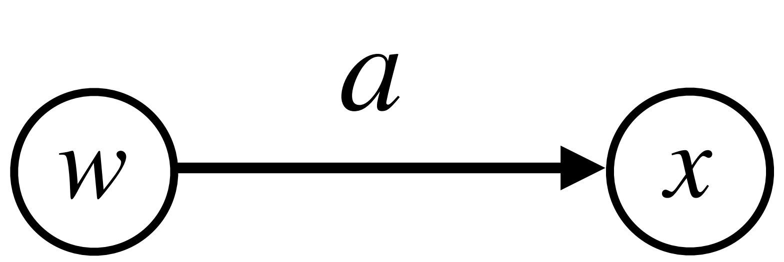
Satisfying Capacity Constraint:

$a - \delta \geq 0$



G

$\delta \leq a$



G_f

A backward edge in the augmenting path

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.

Note: Path P is called an augmenting path.

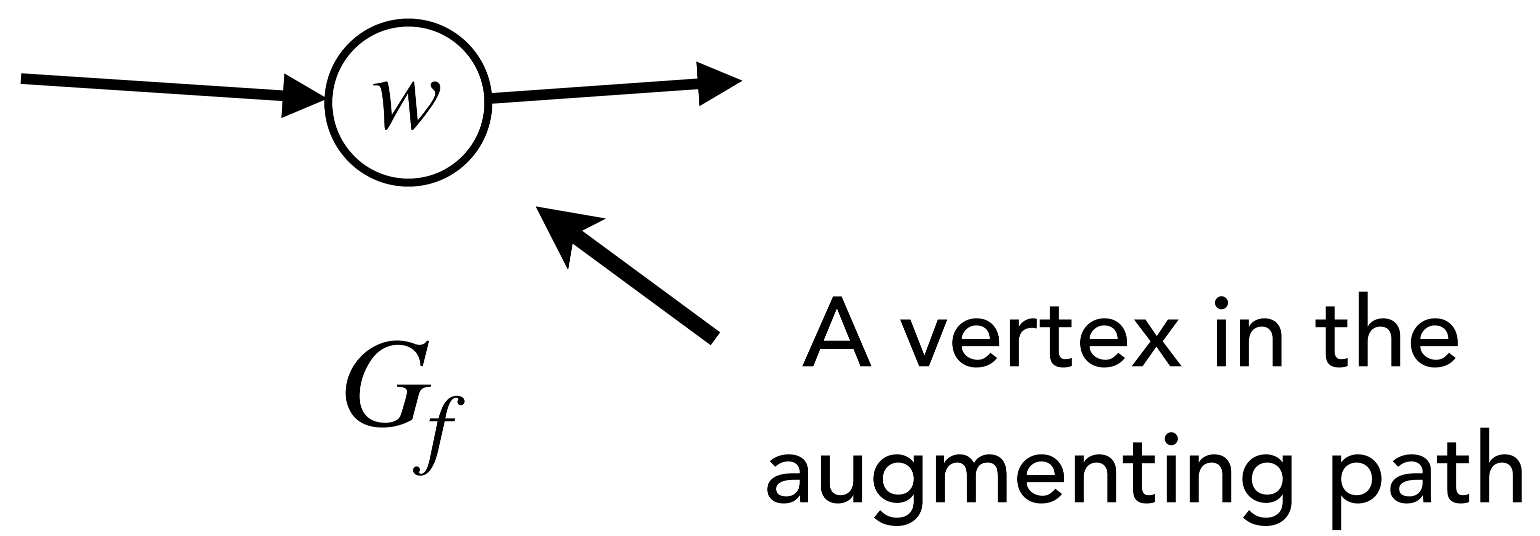
Satisfying Conservation Constraint:

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.

Note: Path P is called an augmenting path.

Satisfying Conservation Constraint:

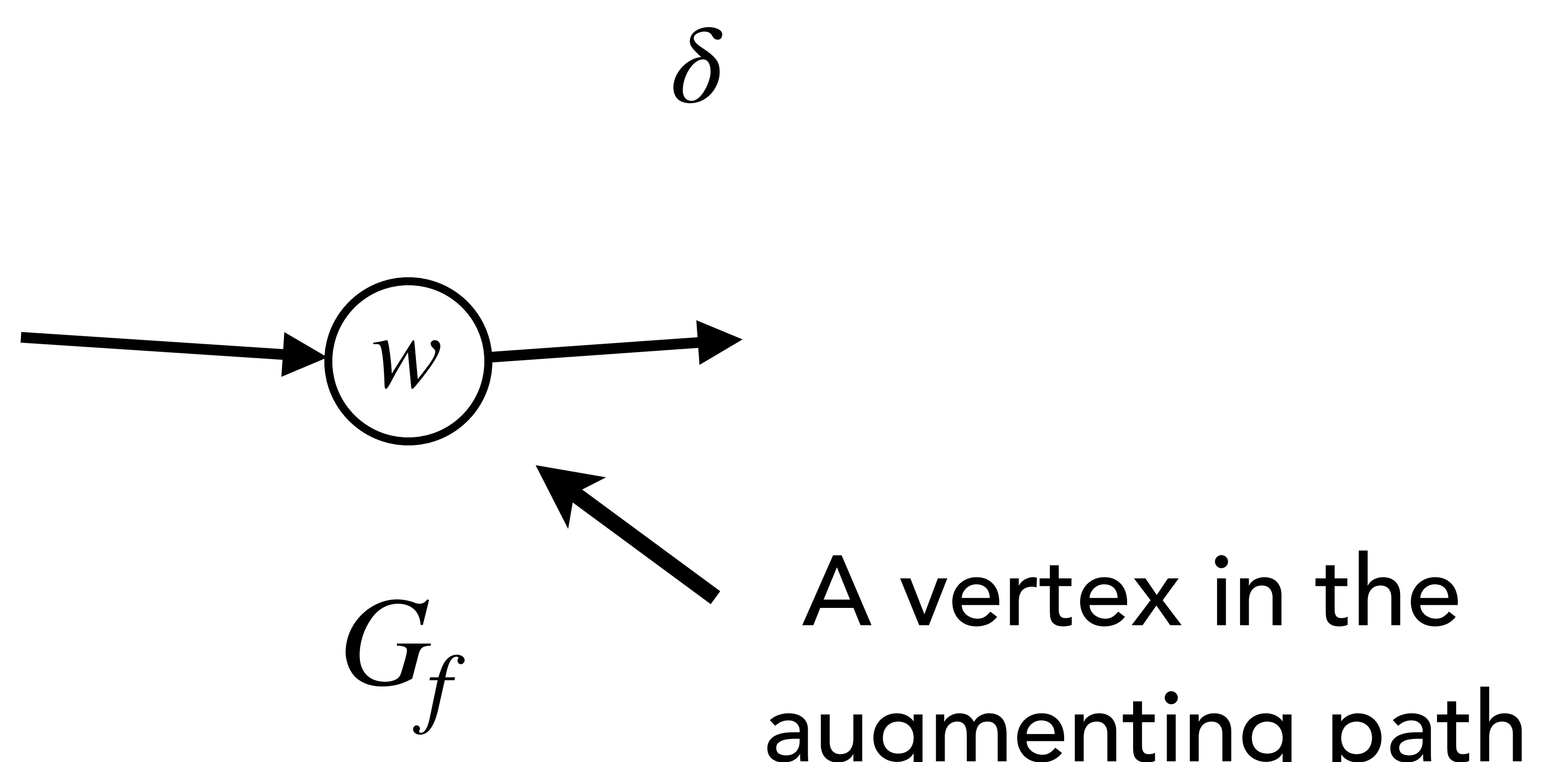


Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

Satisfying Conservation Constraint:

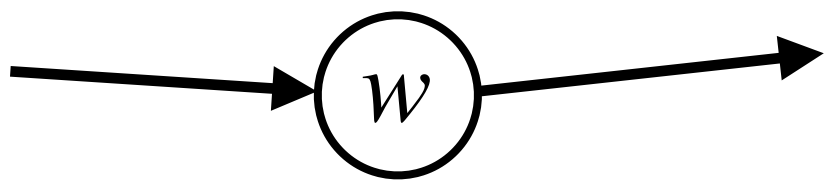


Augmenting Flows via Residual Networks

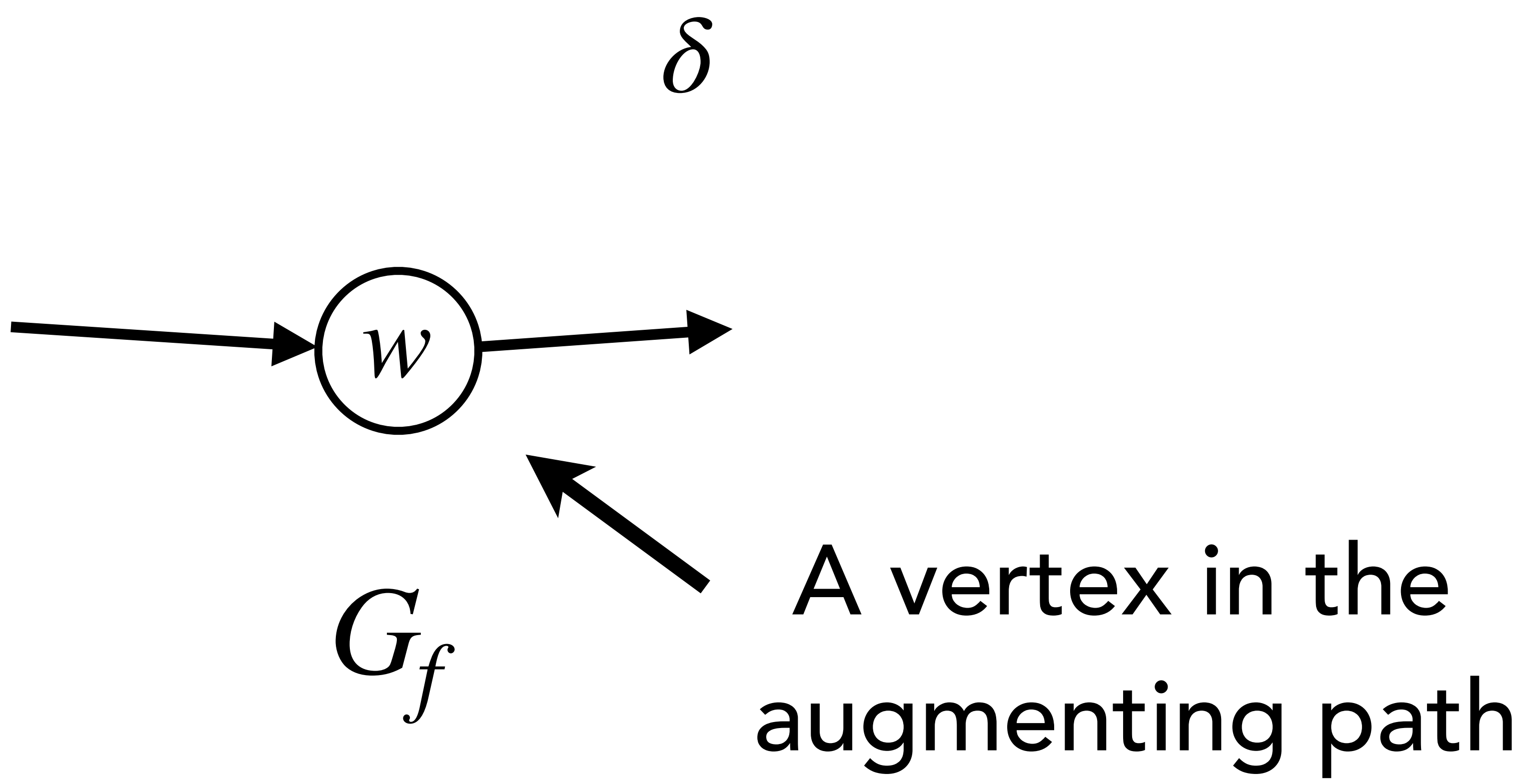
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

Satisfying Conservation Constraint:



Both edges are forward in G

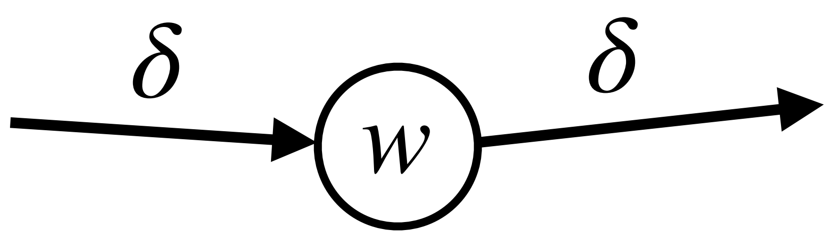


Augmenting Flows via Residual Networks

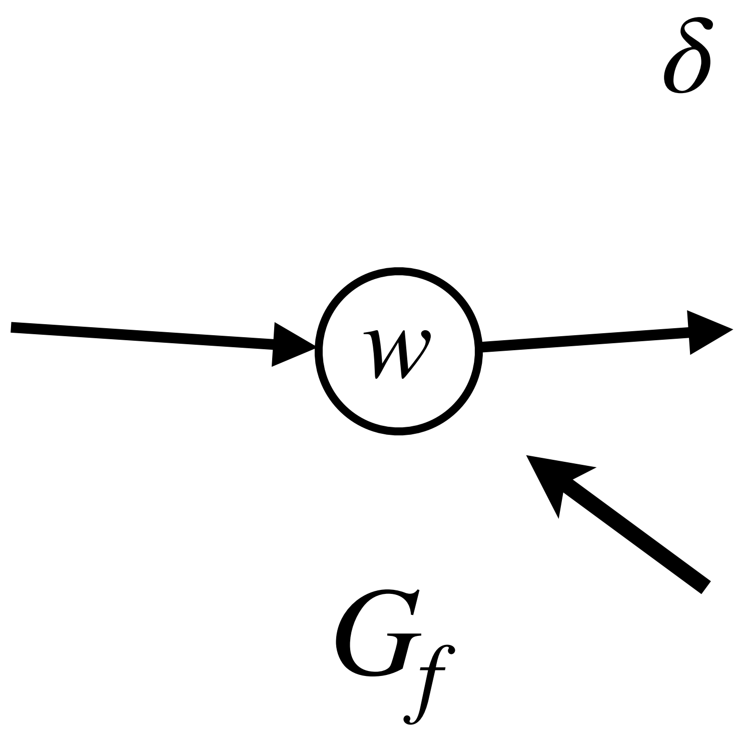
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

Satisfying Conservation Constraint:



Both edges are forward in G

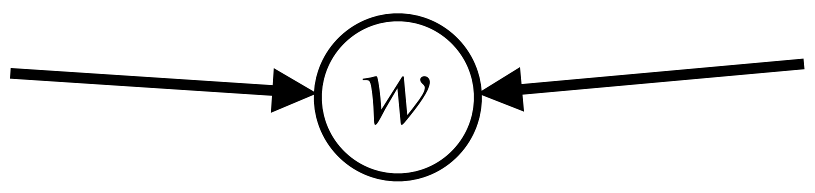


Augmenting Flows via Residual Networks

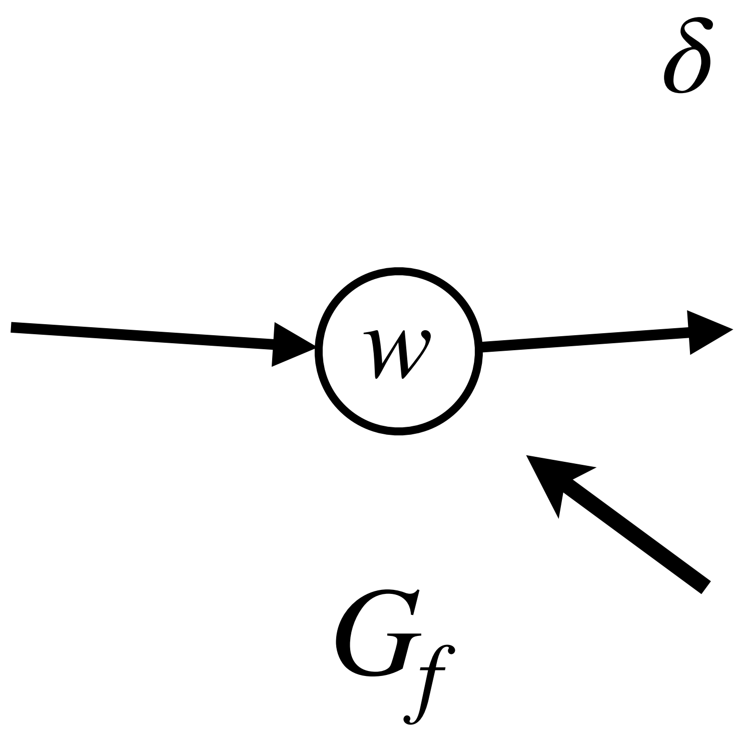
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

Satisfying Conservation Constraint:



First is forward, second is backward in G



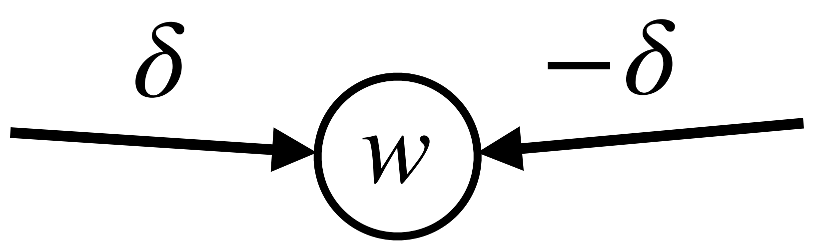
A vertex in the augmenting path

Augmenting Flows via Residual Networks

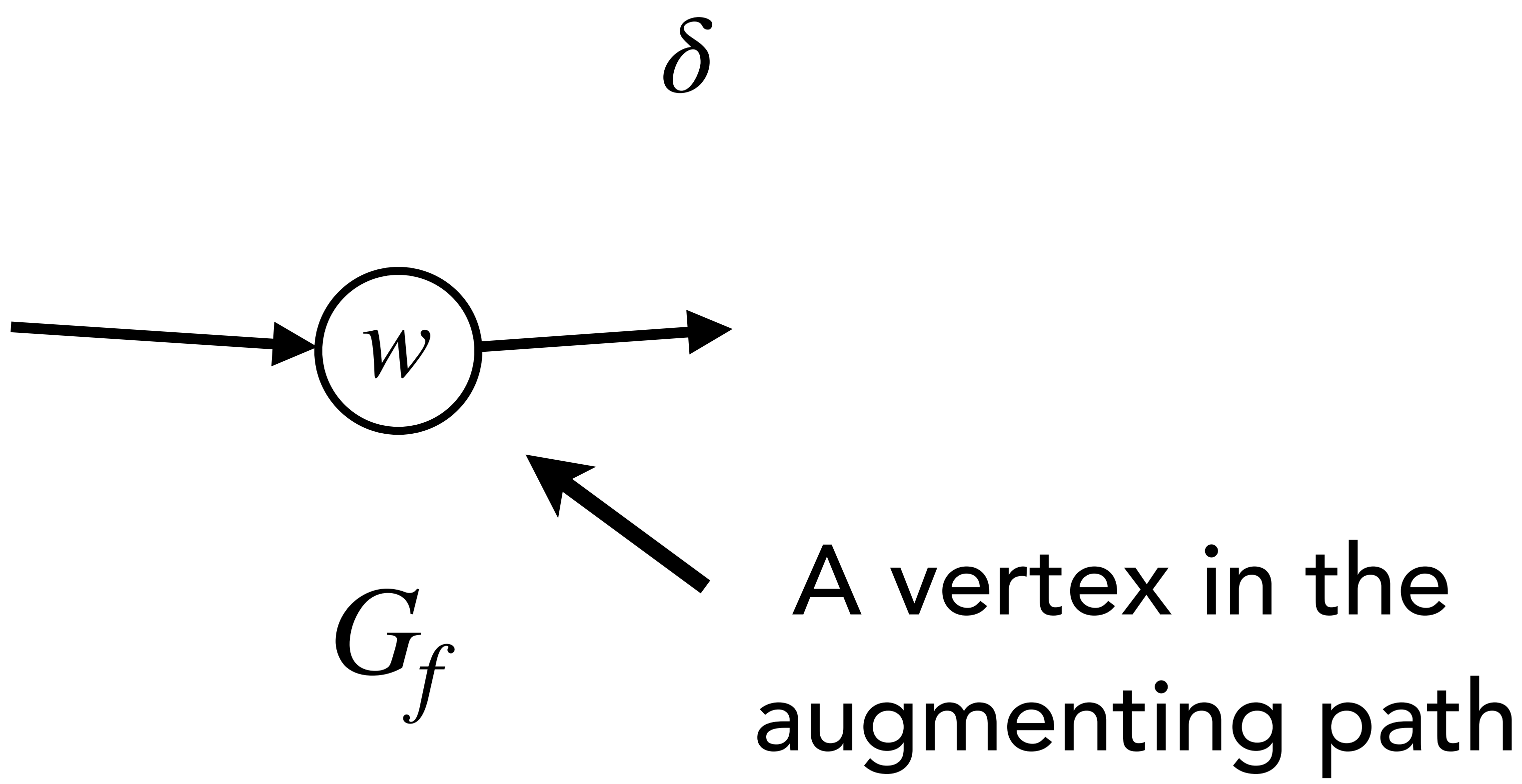
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

Satisfying Conservation Constraint:



First is forward, second is backward in G



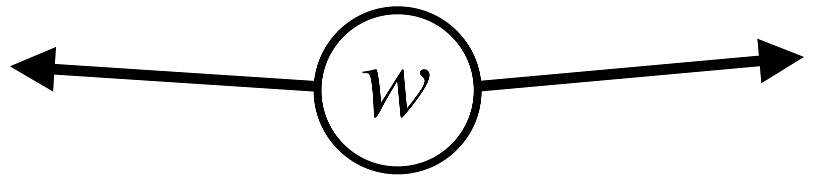
G_f

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.

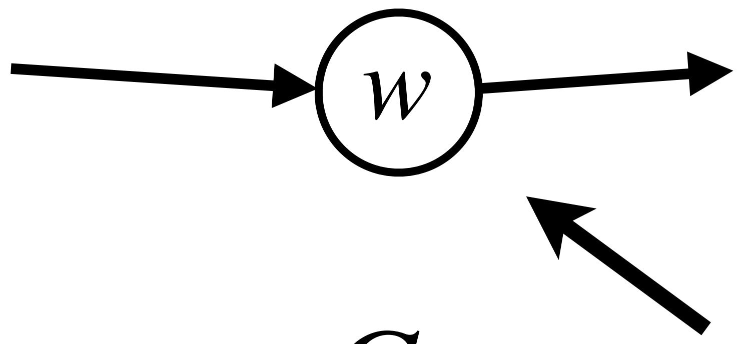
Note: Path P is called an augmenting path.

Satisfying Conservation Constraint:



First is backward, second is forward in G

δ



G_f

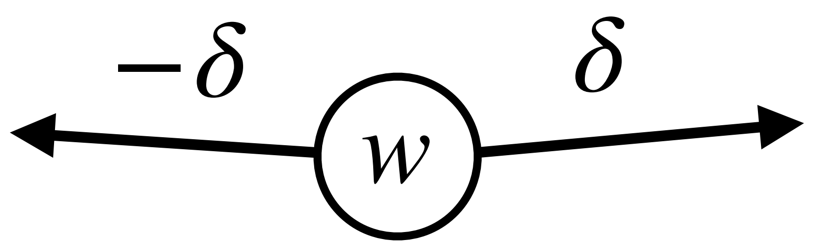
A vertex in the augmenting path

Augmenting Flows via Residual Networks

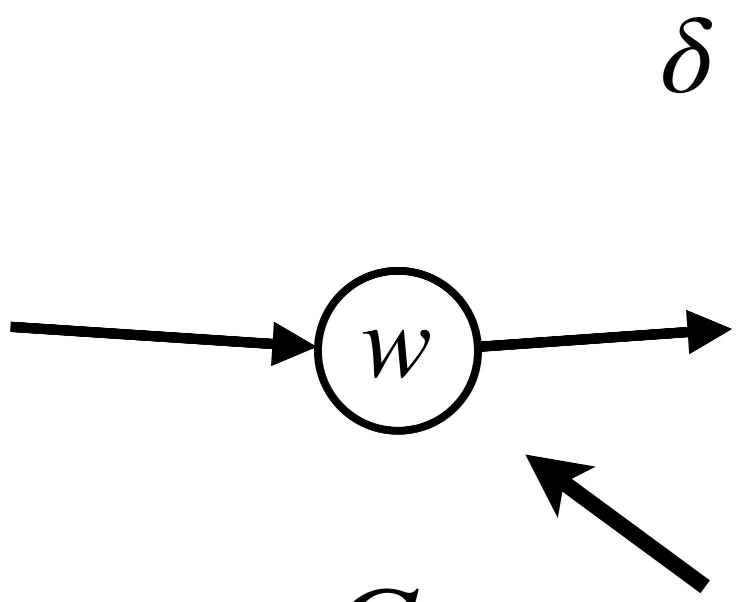
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

Satisfying Conservation Constraint:



First is backward, second is forward in G



G_f

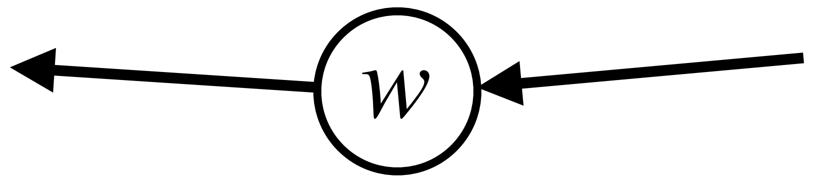
A vertex in the augmenting path

Augmenting Flows via Residual Networks

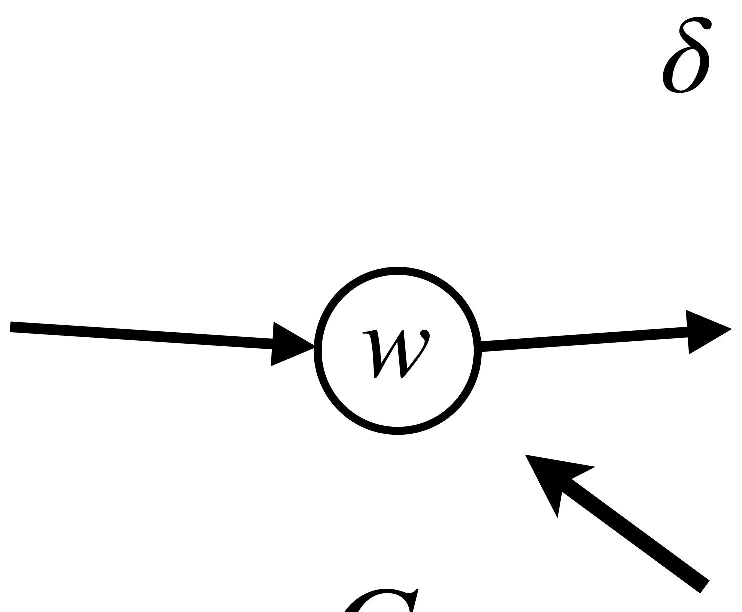
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

Satisfying Conservation Constraint:



Both edges are backward in G



G_f

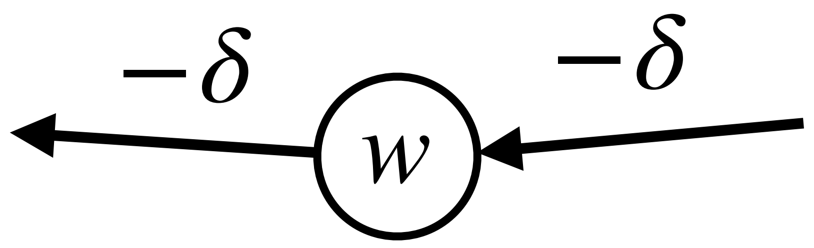
A vertex in the augmenting path

Augmenting Flows via Residual Networks

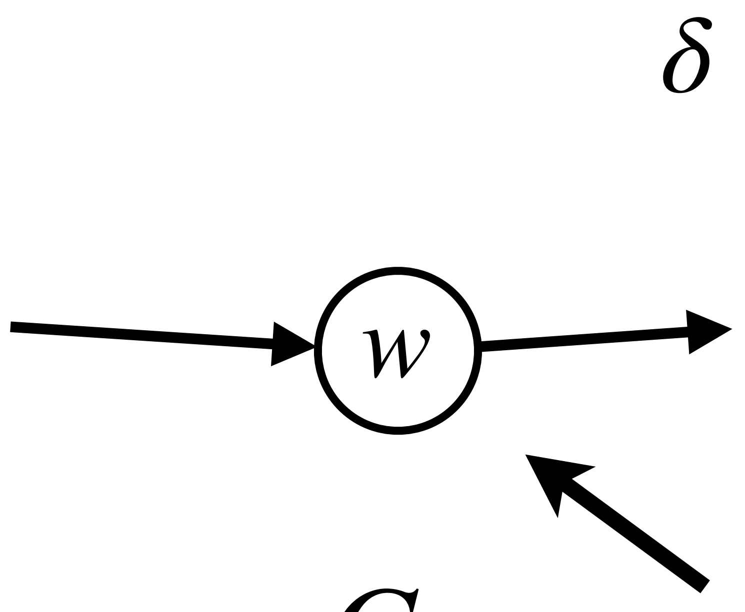
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

Satisfying Conservation Constraint:



Both edges are backward in G



G_f

A vertex in the augmenting path

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.

Note: Path P is called an augmenting path.

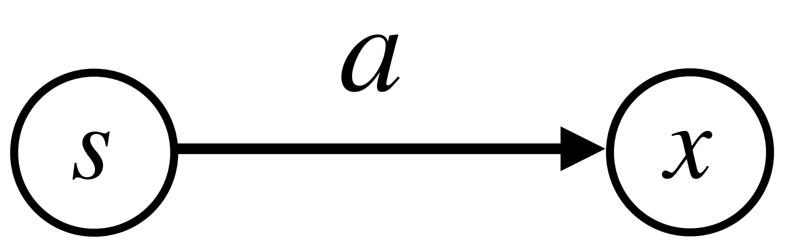
Why flow value increases?

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.

Note: Path P is called an augmenting path.

Why flow value increases?



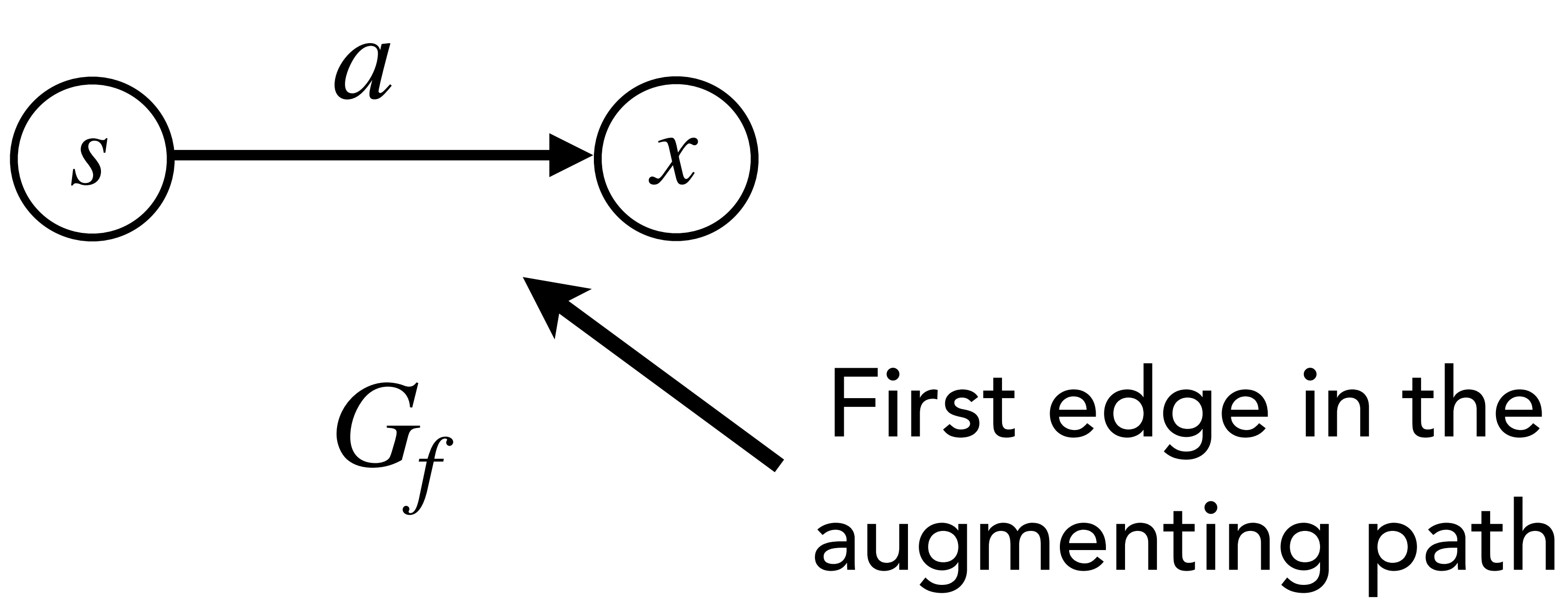
G_f

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.

Note: Path P is called an augmenting path.

Why flow value increases?



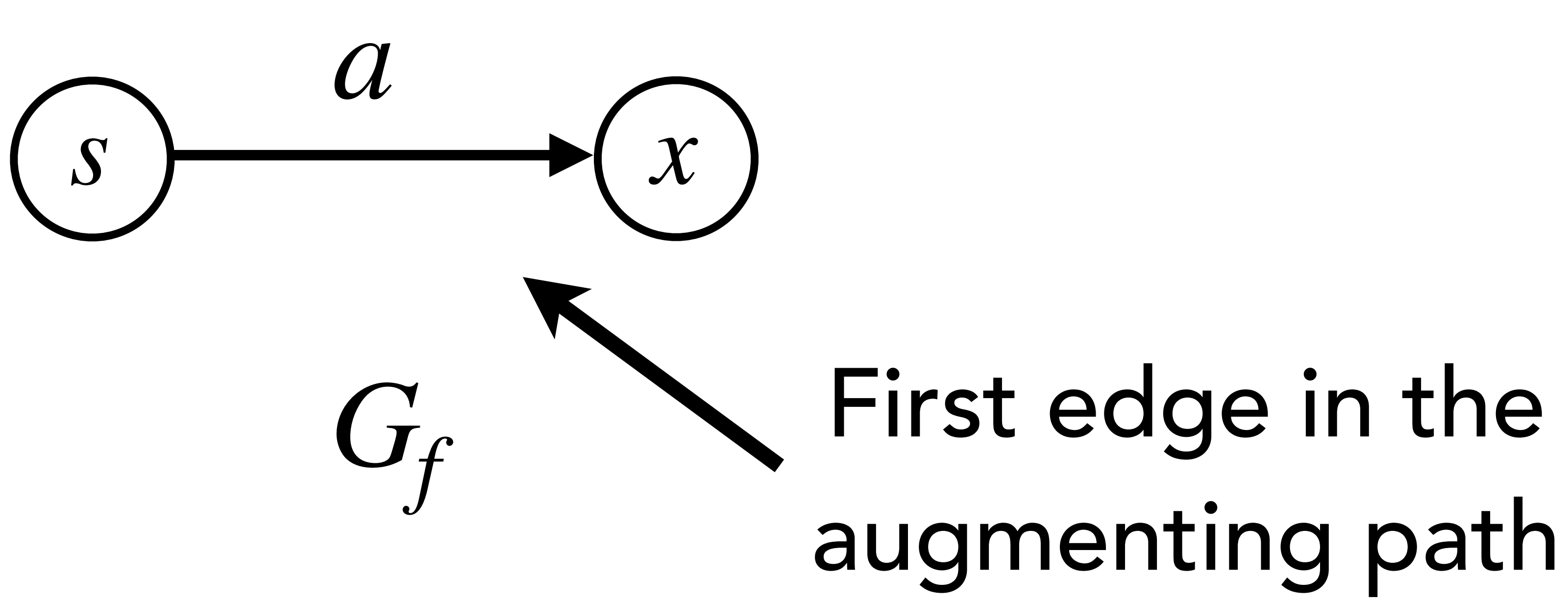
Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

Why flow value increases?

$\delta > 0$



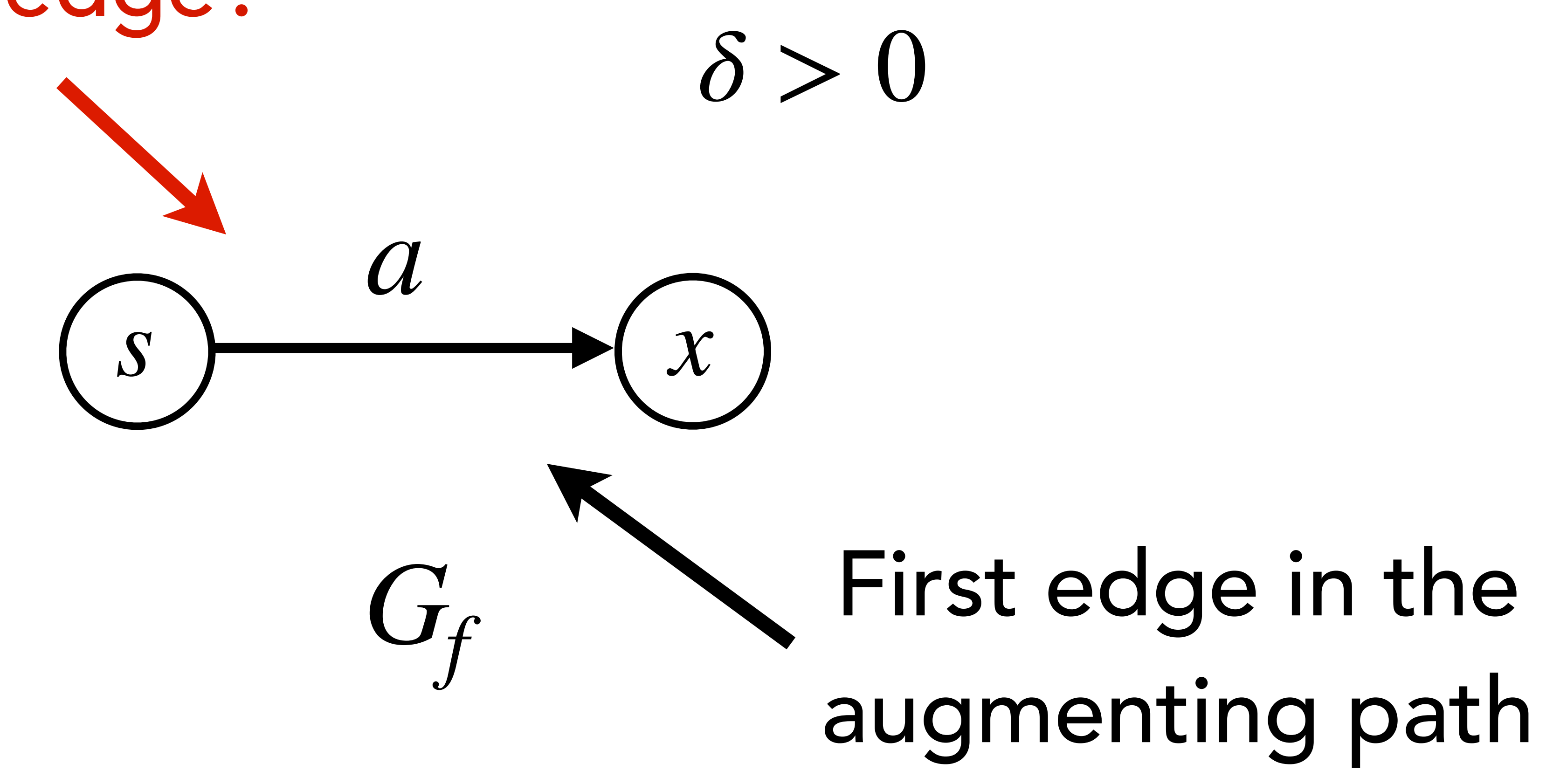
Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

Why flow value increases?

Can it be a backward edge?



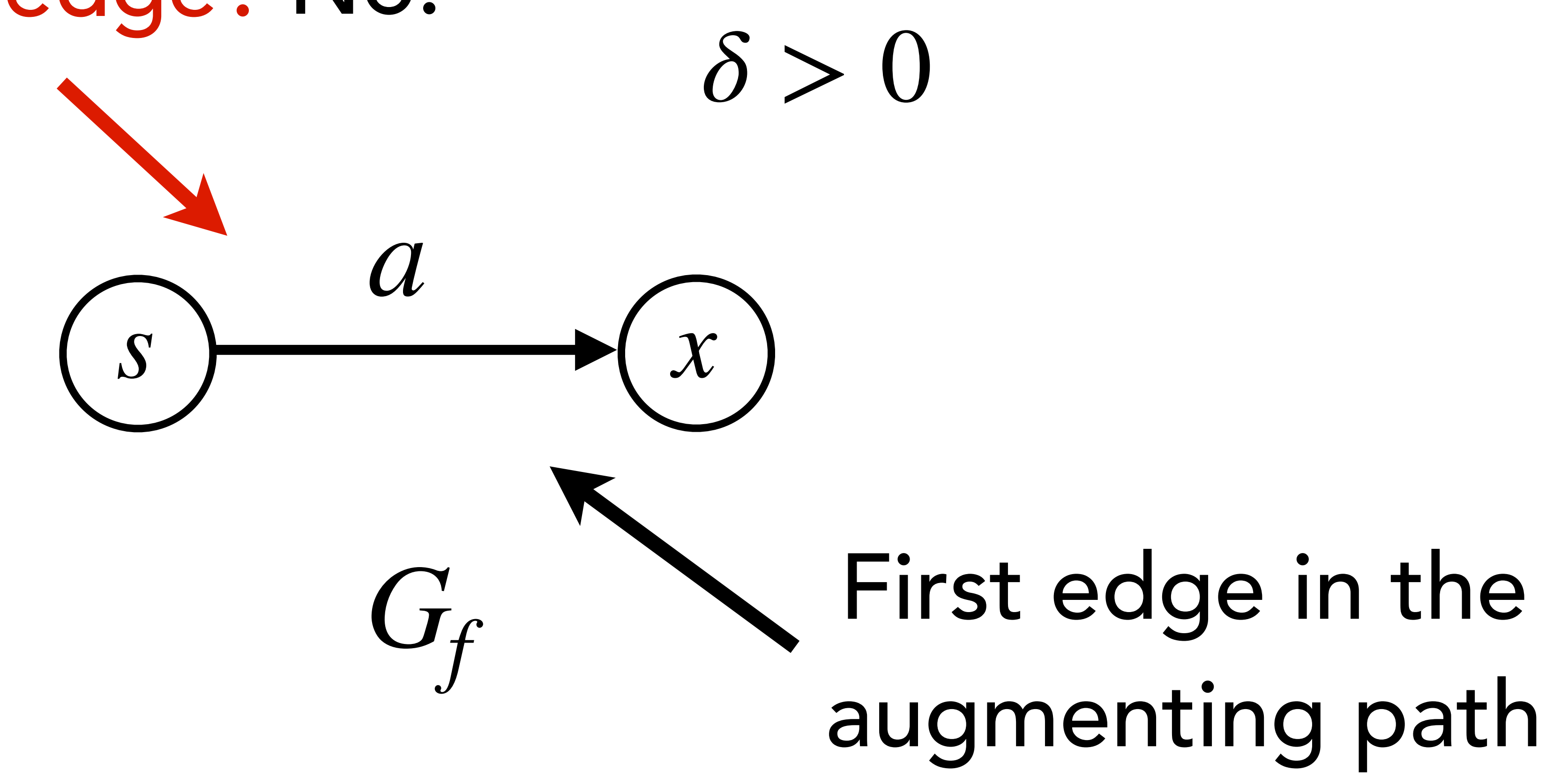
Augmenting Flows via Residual Networks

- 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.

Note: Path P is called an augmenting path.

Why flow value increases?

Can it be a backward edge? No.



Augmenting Flows via Residual Networks

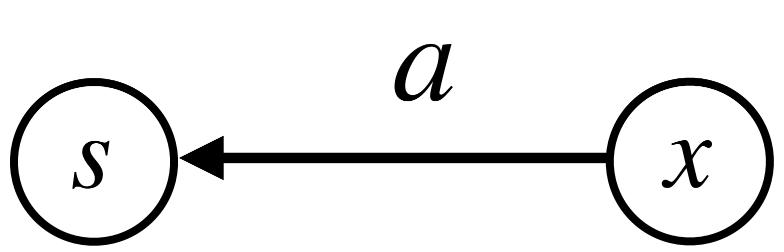
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

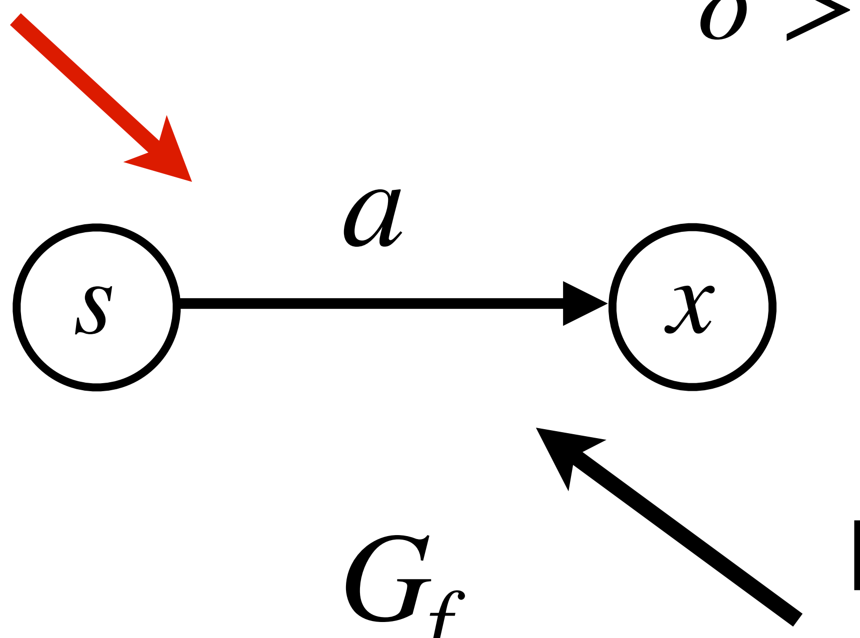
Why flow value increases?

Can it be a backward edge? No.

$\delta > 0$



G



G_f

First edge in the augmenting path

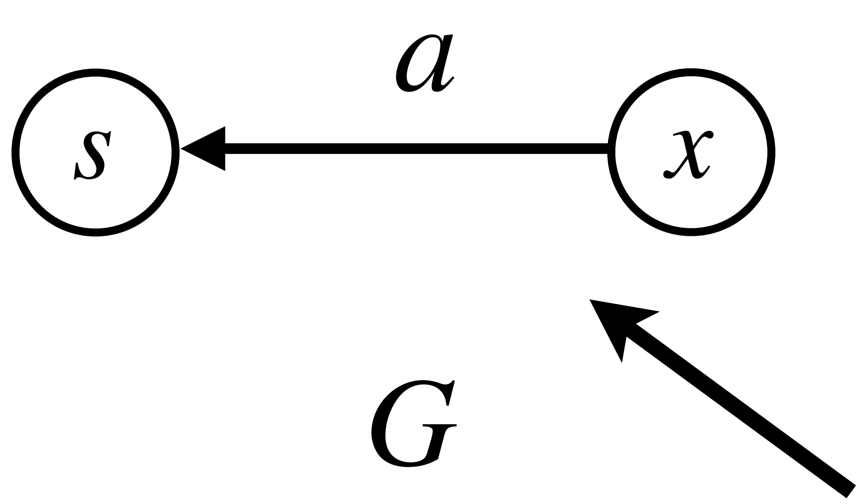
Augmenting Flows via Residual Networks

- 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.

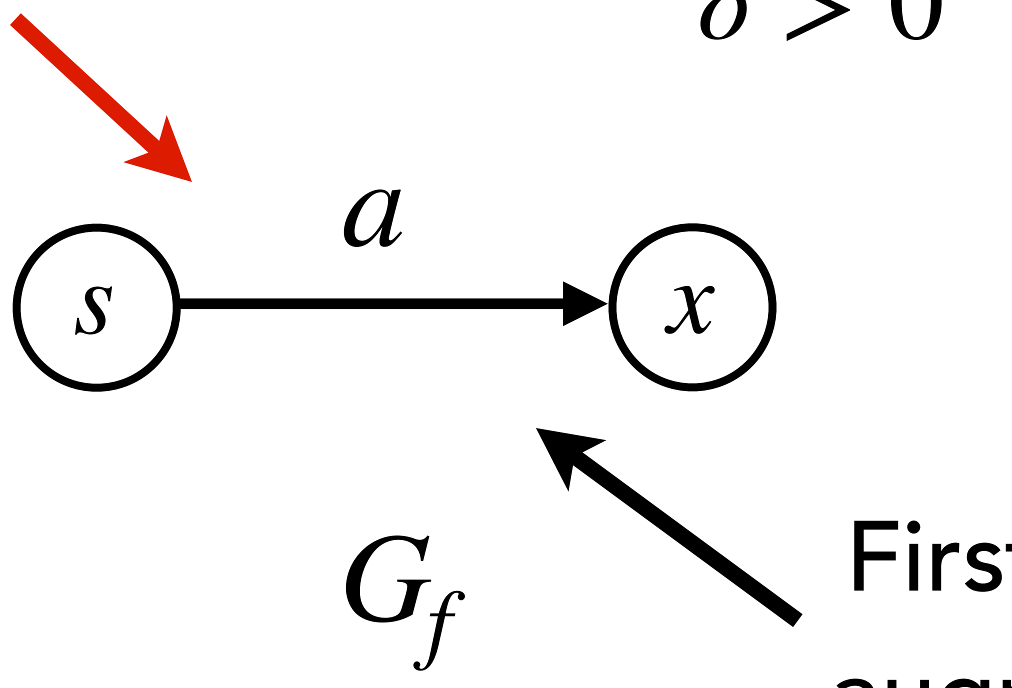
Note: Path P is called an augmenting path.

Why flow value increases?

Can it be a backward edge? No.



An incoming edge
to s is not possible



First edge in the
augmenting path

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.

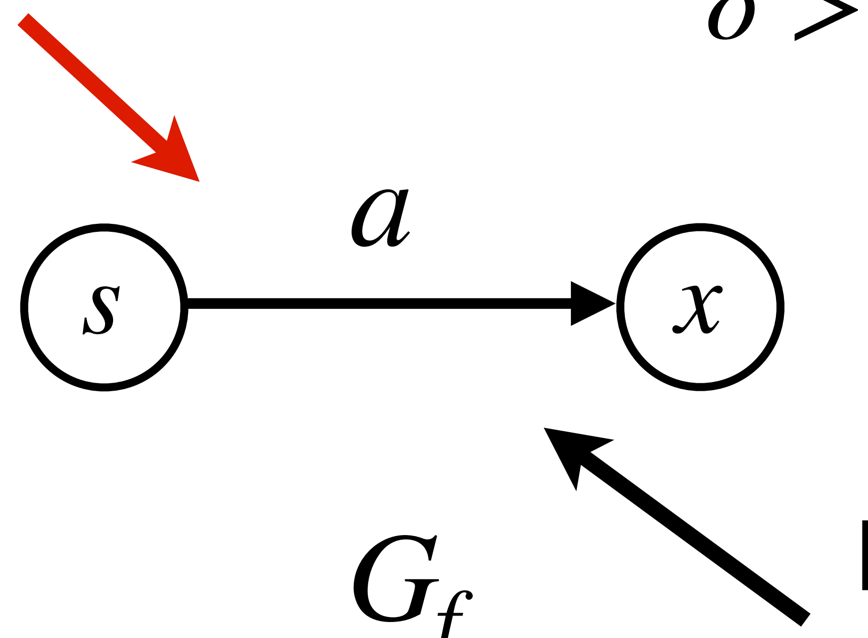
Note: Path P is called an augmenting path.

Why flow value increases?

Can it be a backward edge? No.

$\delta > 0$

G



G_f

First edge in the augmenting path

Augmenting Flows via Residual Networks

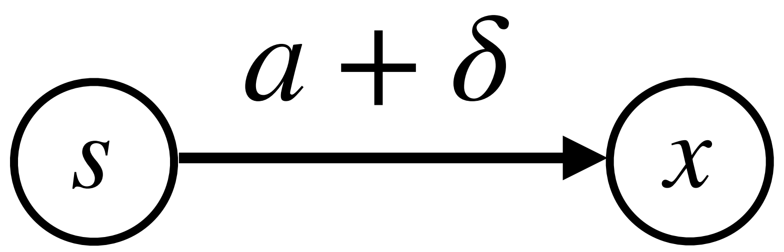
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

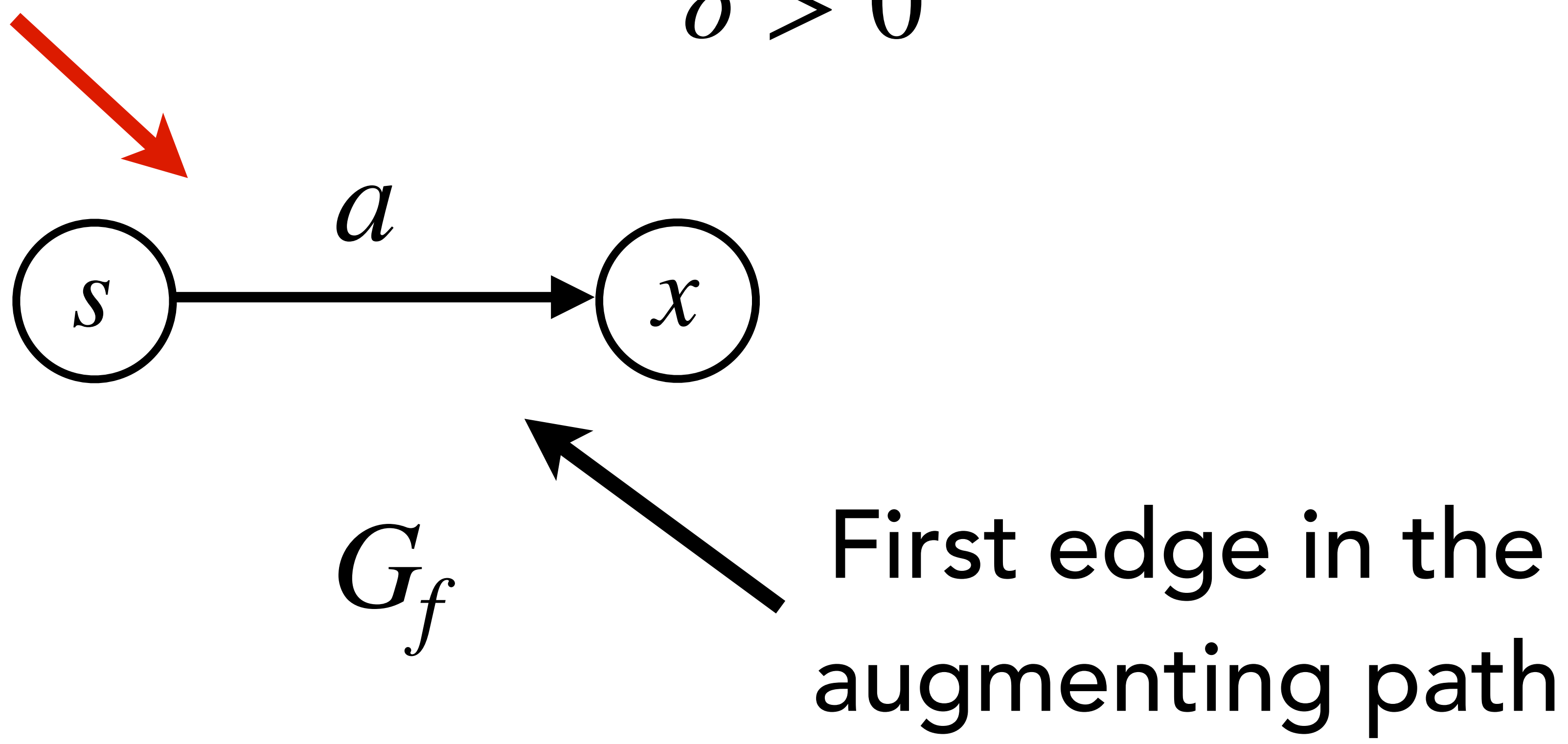
Why flow value increases?

Can it be a backward edge? No.

$\delta > 0$



G



G_f

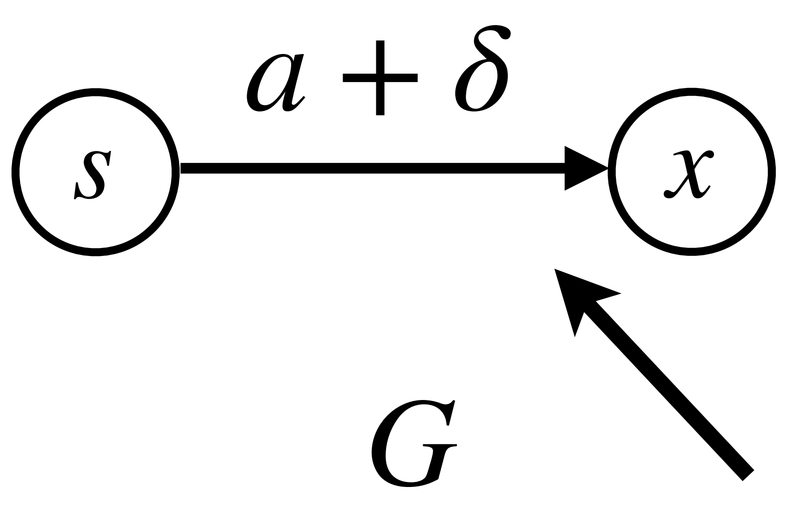
Augmenting Flows via Residual Networks

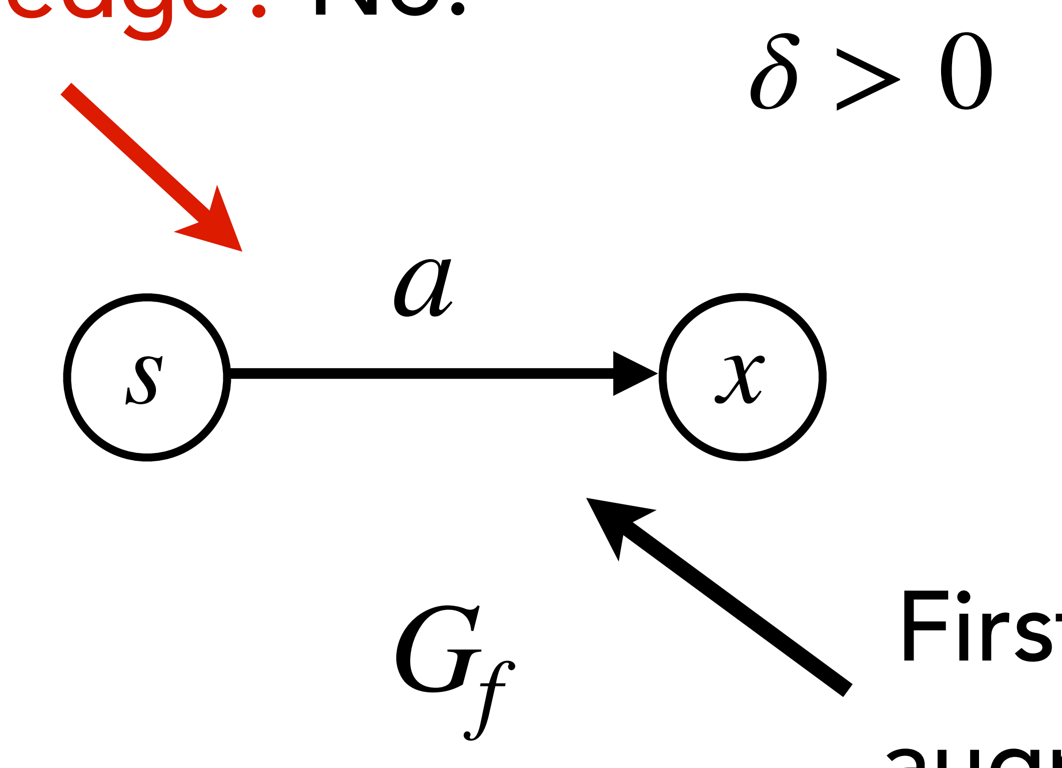
- Find $s \rightsquigarrow t$ path P in the **residual network** G_f and its bottleneck capacity δ.
- For every $(u, v) \in P$:
  - If $(u, v) \in E(G)$, add δ flow to (u, v) in f.
  - If $(v, u) \in E(G)$, subtract δ flow from (v, u) in f.

Note: Path P is called an augmenting path.

Why flow value increases?

Can it be a backward edge? No.





Ford-Fulkerson Method

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

1. **for** each edge $(u, v) \in E(G)$

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

1. **for** each edge $(u, v) \in E(G)$
2. $\quad f(u, v) = 0$

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

1. **for** each edge $(u, v) \in E(G)$
2. $\quad f(u, v) = 0$
3. **while** there exists an $s \rightsquigarrow t$ path P in the residual network G_f

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

1. **for** each edge $(u, v) \in E(G)$
2. $\quad f(u, v) = 0$
3. **while** there exists an $s \rightsquigarrow t$ path P in the residual network G_f
4. $\quad \delta = \textbf{Min}(c_f(u, v) : (u, v) \textbf{ in } P)$

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

1. **for** each edge $(u, v) \in E(G)$
2. $\quad f(u, v) = 0$
3. **while** there exists an $s \rightsquigarrow t$ path P in the residual network G_f
4. $\quad \delta = \textbf{Min}(c_f(u, v) : (u, v) \textbf{ in } P)$
5. $\quad$ **for** each edge (u, v) in P

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

for each edge $(u, v) \in E(G)$
    $f(u, v) = 0$
while there exists an $s \rightsquigarrow t$ path P in the residual network G_f
    $\delta =$ **Min**$(c_f(u, v) : (u, v)$ **in** $P)$
    for each edge (u, v) in P
        if $(u, v) \in E$

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

1. **for** each edge $(u, v) \in E(G)$
2. $\quad f(u, v) = 0$
3. **while** there exists an $s \rightsquigarrow t$ path P in the residual network G_f
4. $\quad \delta = \textbf{Min}(c_f(u, v) : (u, v)\ \textbf{in}\ P)$
5. $\quad$ **for** each edge (u, v) in P
6. $\quad\quad$ **if** $(u, v) \in E$
7. $\quad\quad\quad f(u, v) = f(u, v) + \delta$

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

for each edge $(u, v) \in E(G)$
    $f(u, v) = 0$
while there exists an $s \rightsquigarrow t$ path P in the residual network G_f
    $\delta =$ **Min**$(c_f(u, v) : (u, v)$ **in** $P)$
    for each edge (u, v) in P
        if $(u, v) \in E$
            $f(u, v) = f(u, v) + \delta$
        else

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

for each edge $(u, v) \in E(G)$
    $f(u, v) = 0$
while there exists an $s \rightsquigarrow t$ path P in the residual network G_f
    $\delta =$ **Min**$(c_f(u, v) : (u, v)$ **in** $P)$
    for each edge (u, v) in P
        if $(u, v) \in E$
            $f(u, v) = f(u, v) + \delta$
        else
            $f(v, u) = f(v, u) - \delta$

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t)**:**

for each edge $(u, v) \in E(G)$
    $f(u, v) = 0$
while there exists an $s \rightsquigarrow t$ path P in the residual network G_f
    $\delta = \textbf{Min}(c_f(u, v) : (u, v) \textbf{ in } P)$
    for each edge (u, v) in P
        if $(u, v) \in E$
            $f(u, v) = f(u, v) + \delta$
        else
            $f(v, u) = f(v, u) - \delta$
return f

Ford-Fulkerson Method

Ford-Fulkerson(G, s, t):

for each edge $(u, v) \in E(G)$
  $f(u, v) = 0$
while there exists an $s \rightsquigarrow t$ path P in the residual network G_f
  $\delta =$ **Min**$(c_f(u, v) : (u, v)$ **in** $P)$
  for each edge (u, v) in P
    if $(u, v) \in E$
      $f(u, v) = f(u, v) + \delta$
    else
      $f(v, u) = f(v, u) - \delta$
return f

Why f will be maximum when loop breaks?

Ford-Fulkerson Method: Correctness

Ford-Fulkerson Method: Correctness

Theorem: If there is no augmenting path in the residual network G_f, then f is a maximum flow.

Ford-Fulkerson Method: Correctness

Theorem: If there is no augmenting path in the residual network G_f, then f is a maximum flow.

Proof: We need to study cuts and max-flow, min-cut theorem for the proof.

Ford-Fulkerson Method: Analysis

Ford-Fulkerson(G, s, t)**:**

for each edge $(u, v) \in E(G)$
    $f(u, v) = 0$
while there exists an $s \rightsquigarrow t$ path P in the residual network G_f
    $\delta =$ **Min**$(c_f(u, v) : (u, v)$ **in** $P)$
    for each edge (u, v) in P
        if $(u, v) \in E$
            $f(u, v) = f(u, v) + \delta$
        else
            $f(v, u) = f(v, u) - \delta$
return f

Ford-Fulkerson Method: Analysis

Ford-Fulkerson(G, s, t):

for each edge $(u, v) \in E(G)$ ← $O(|E|)$

  $f(u, v) = 0$

while there exists an $s \rightsquigarrow t$ path P in the residual network G_f

  $\delta =$ **Min**$(c_f(u, v) : (u, v)$ **in** $P)$

  for each edge (u, v) in P

    if $(u, v) \in E$

      $f(u, v) = f(u, v) + \delta$

    else

      $f(v, u) = f(v, u) - \delta$

return f

Ford-Fulkerson Method: Analysis

Ford-Fulkerson(G, s, t)**:**

for each edge $(u, v) \in E(G)$ — $O(|E|)$

  $f(u, v) = 0$

while there exists an $s \rightsquigarrow t$ path P in the residual network G_f — Loop may run for $|f^*|$ time, where f^* is a max-flow, as flow may increase by one with every iteration

  $\delta =$ **Min**$(c_f(u, v) : (u, v)$ **in** $P)$

  for each edge (u, v) in P

    if $(u, v) \in E$

      $f(u, v) = f(u, v) + \delta$

    else

      $f(v, u) = f(v, u) - \delta$

return f

Ford-Fulkerson Method: Analysis

```
Ford-Fulkerson(G, s, t):
    for each edge (u, v) ∈ E(G)
        f(u, v) = 0
    while there exists an s ⇝ t path P in the residual network G_f
        δ = Min(c_f(u, v) : (u, v) in P)
        for each edge (u, v) in P
            if (u, v) ∈ E
                f(u, v) = f(u, v) + δ
            else
                f(v, u) = f(v, u) − δ
    return f
```

$O(|E|)$

Loop may run for $|f^*|$ time, where f^* is a max-flow, as flow may increase by one with every iteration

$O(|E|)$ because $|E| \geq |V| - 1$

Ford-Fulkerson Method: Analysis

Ford-Fulkerson(G, s, t):

- **for** each edge $(u, v) \in E(G)$
  - $f(u, v) = 0$
- **while** there exists an $s \rightsquigarrow t$ path P in the residual network G_f
  - $\delta =$ **Min**$(c_f(u, v) : (u, v)$ **in** $P)$
  - **for** each edge (u, v) in P
    - **if** $(u, v) \in E$
      - $f(u, v) = f(u, v) + \delta$
    - **else**
      - $f(v, u) = f(v, u) - \delta$
- **return** f

$O(|E|)$

Loop may run for $|f^*|$ time, where f^* is a max-flow, as flow may increase by one with every iteration

$O(|E|)$ because $|E| \geq |V| - 1$

Time Complexity: $O(|E| \cdot |f^*|)$

Ford-Fulkerson Method: Analysis

Ford-Fulkerson(G, s, t)**:**

1. **for** each edge $(u, v) \in E(G)$ — $O(|E|)$
2. $\quad f(u, v) = 0$
3. **while** there exists an $s \rightsquigarrow t$ path P in the residual network G_f
4. $\quad \delta =$ **Min**$(c_f(u, v) : (u, v)$ **in** $P)$
5. $\quad$ **for** each edge (u, v) in P
6. $\quad\quad$ **if** $(u, v) \in E$
7. $\quad\quad\quad f(u, v) = f(u, v) + \delta$
8. $\quad\quad$ **else**
9. $\quad\quad\quad f(v, u) = f(v, u) - \delta$
10. **return** f

Loop may run for $|f^*|$ time, where f^* is a max-flow, as flow may increase by one with every iteration

Lines 4–9: $O(|E|)$ because $|E| \geq |V| - 1$

Time Complexity: $O(|E| \cdot |f^*|)$

Note: Analysis is valid when capacities are integer.

Ford-Fulkerson Method: A Non-terminating Case

Ford-Fulkerson Method: A Non-terminating Case

Ford-Fulkerson may not terminate when some capacities are irrational.

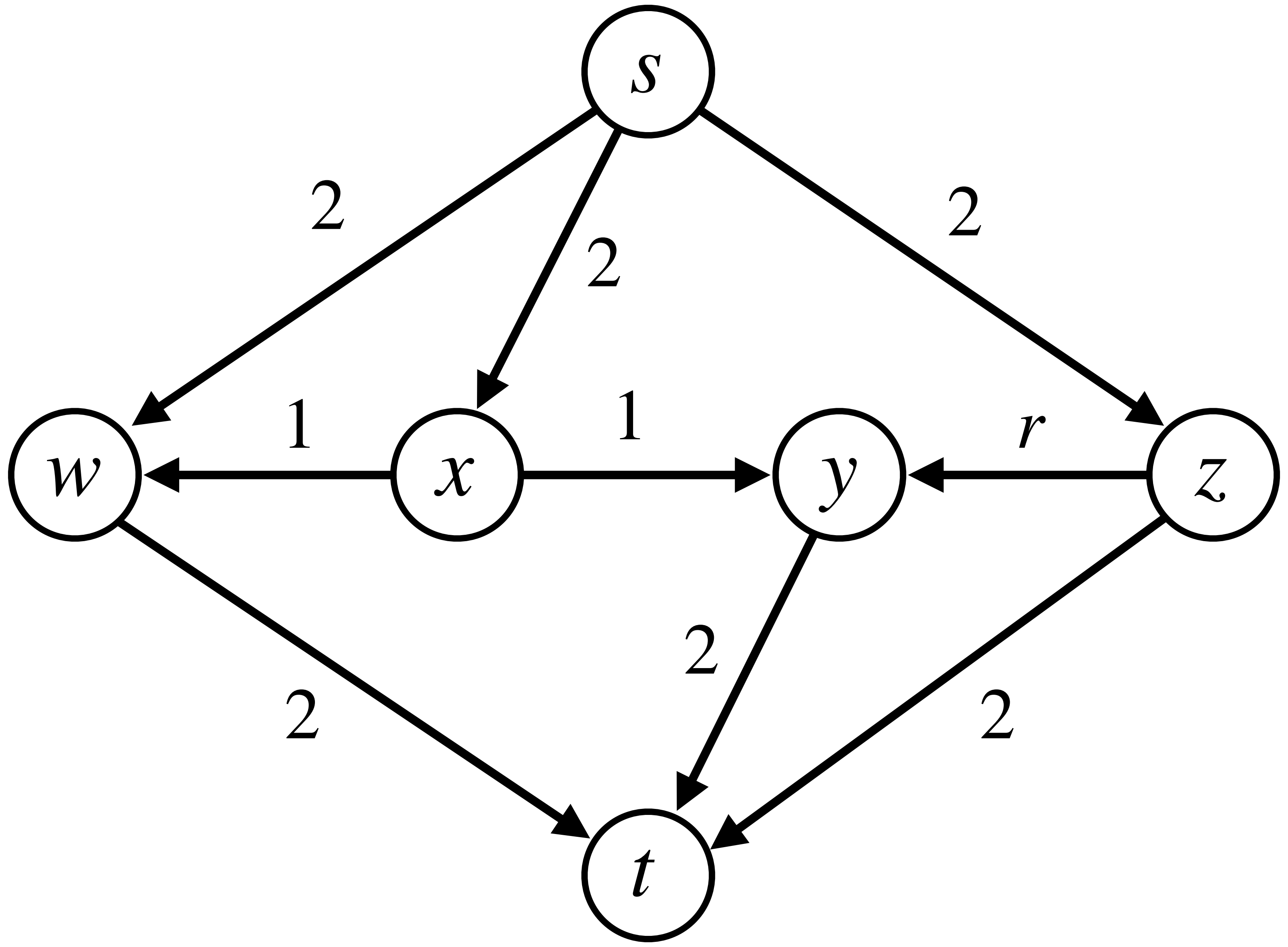
Ford-Fulkerson Method: A Non-terminating Case

Ford-Fulkerson may not terminate when some capacities are irrational.

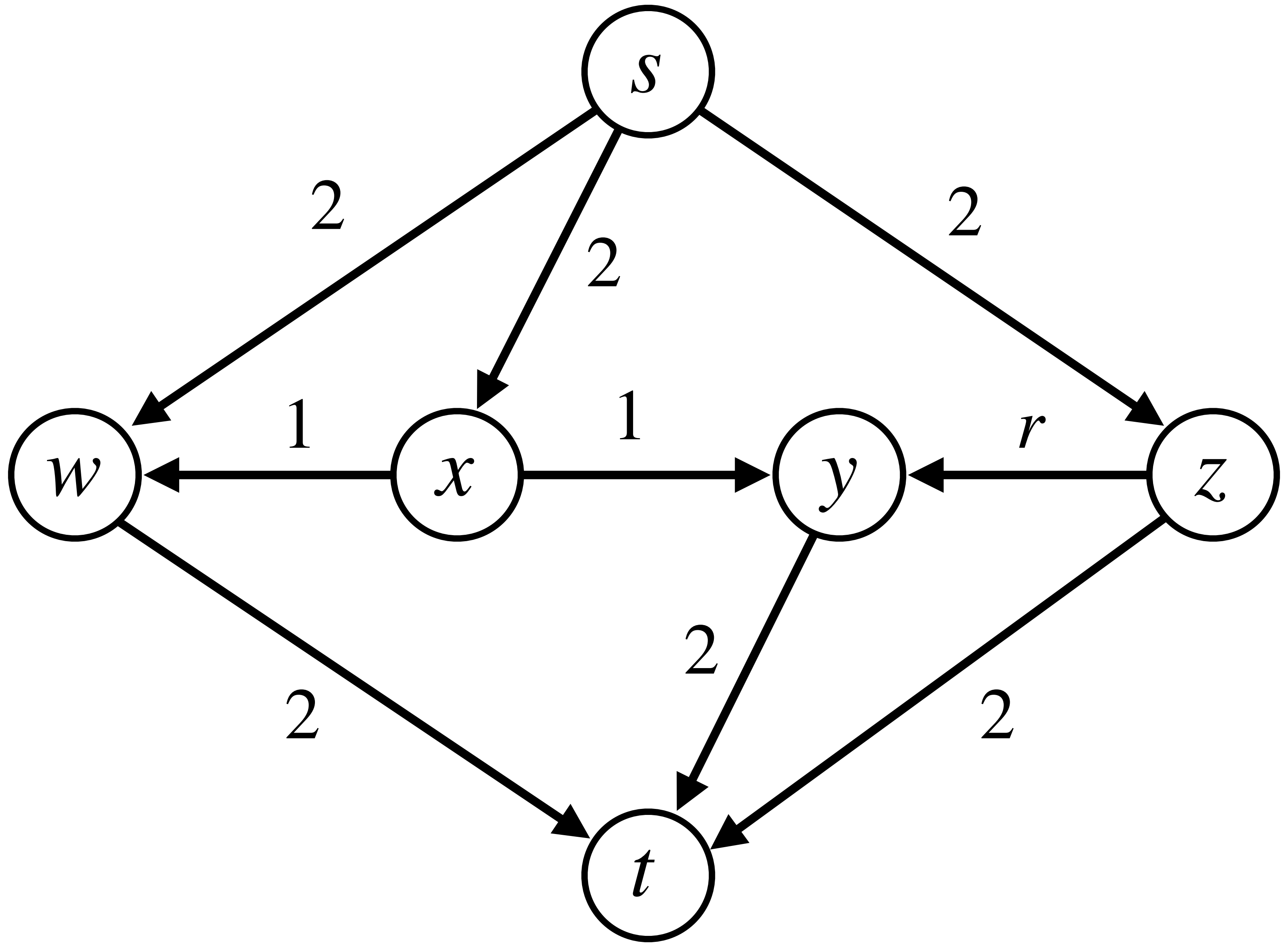
A famous example is on the next slide.

Ford-Fulkerson Method: A Non-terminating Case

Ford-Fulkerson Method: A Non-terminating Case



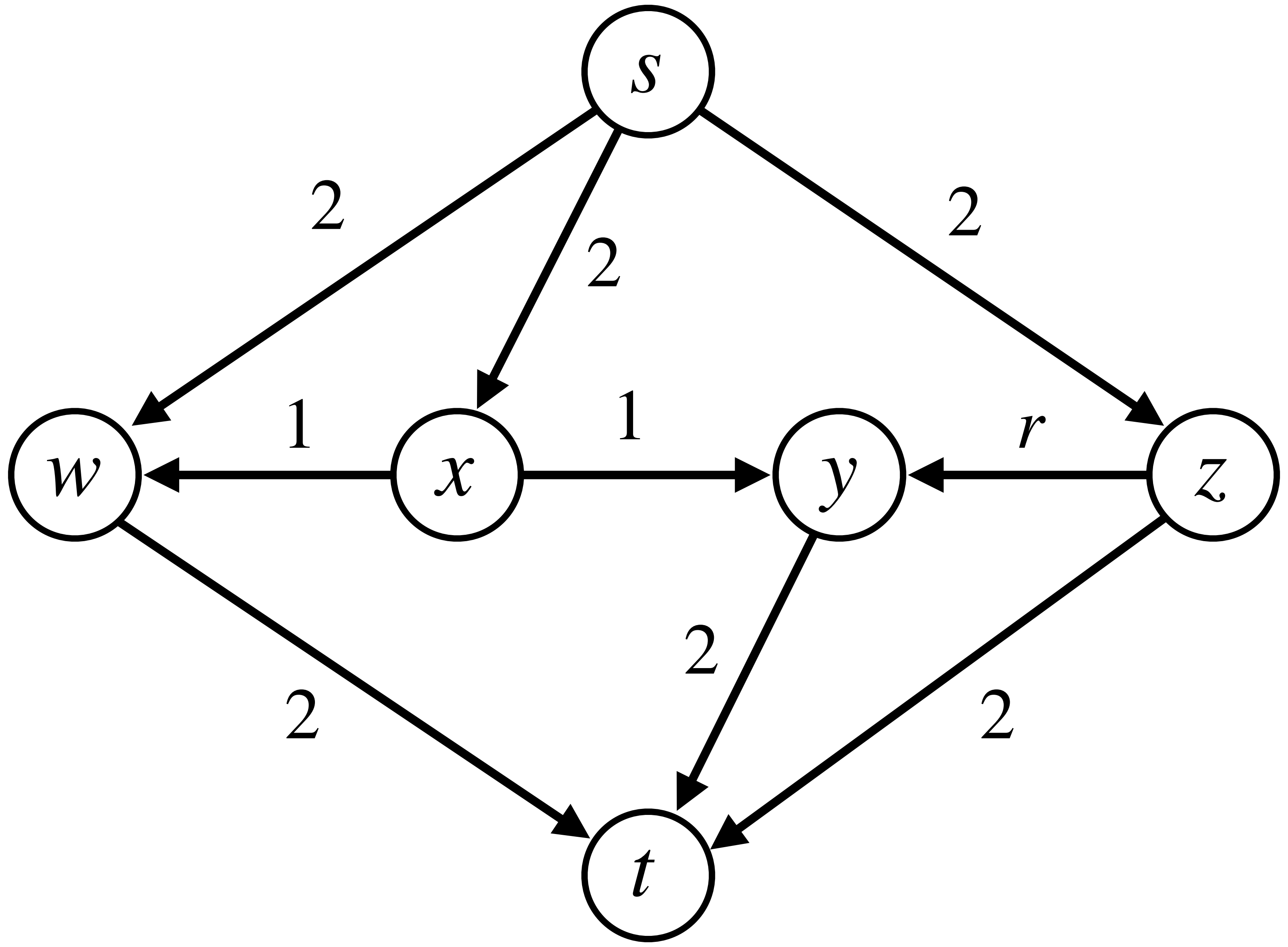
Ford-Fulkerson Method: A Non-terminating Case



$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

Let $P = \langle s, x, y, z \rangle$, $P_1 = \langle s, z, y, x, w \rangle$, $P_2 = \langle s, w, y, z, t \rangle$, $P_3 = \langle w, x, y, t \rangle$ in residual networks.

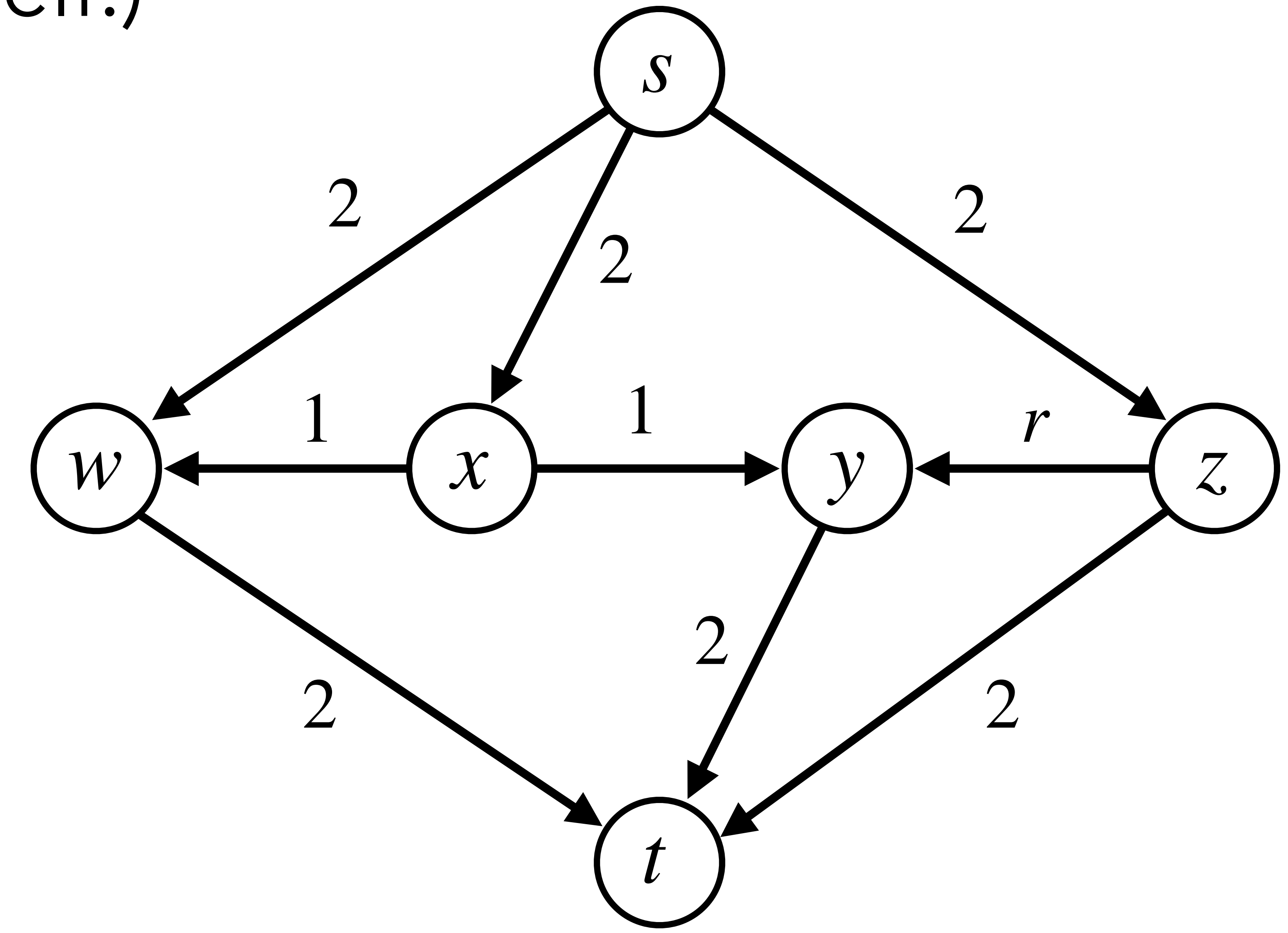


$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

Let $P = \langle s, x, y, z \rangle$, $P_1 = \langle s, z, y, x, w \rangle$, $P_2 = \langle s, w, y, z, t \rangle$, $P_3 = \langle w, x, y, t \rangle$ in residual networks.

Then we can perform the following 5 steps: (Verify it yourself.)



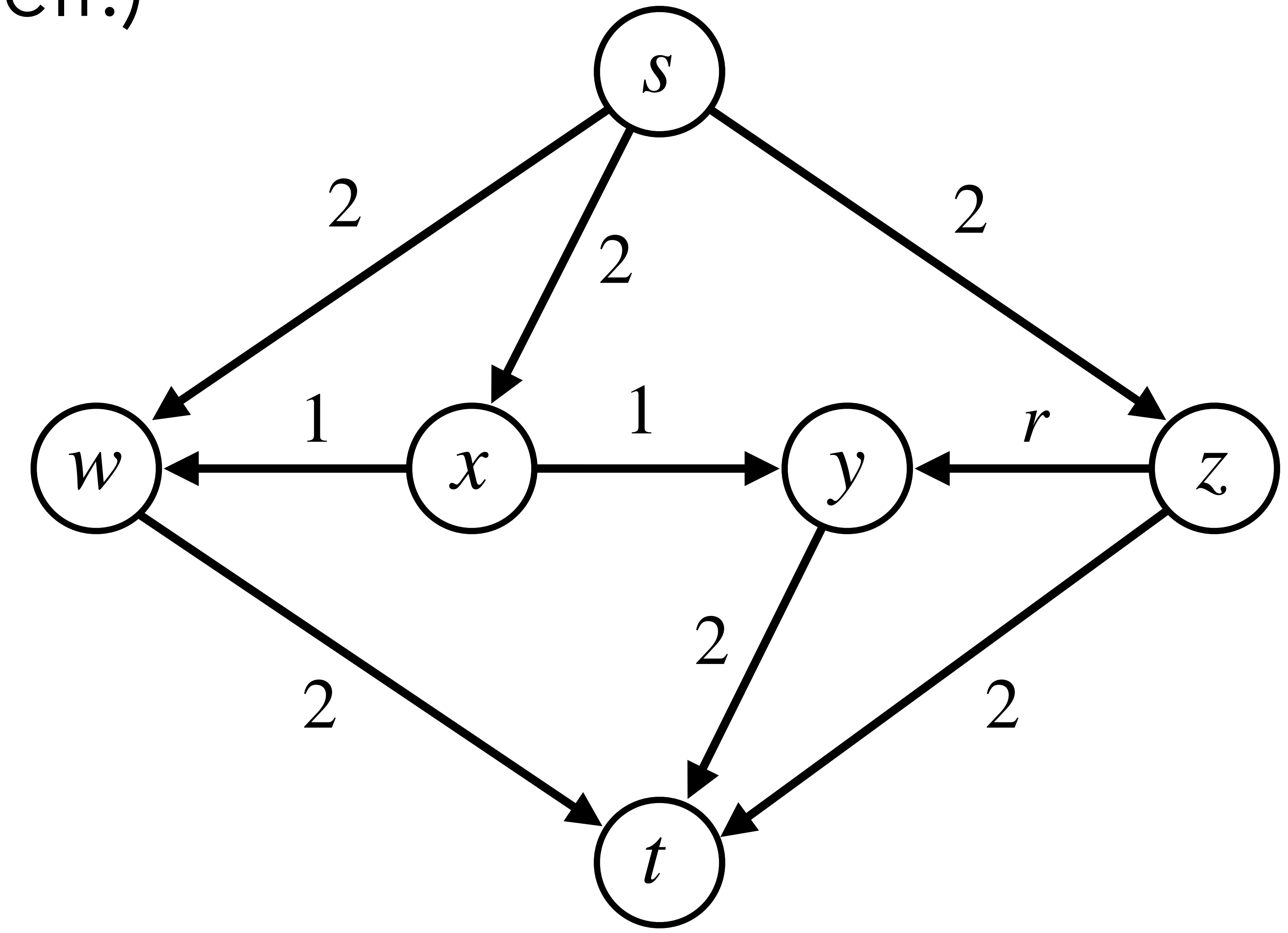
$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

Let $P = \langle s, x, y, z \rangle$, $P_1 = \langle s, z, y, x, w \rangle$, $P_2 = \langle s, w, y, z, t \rangle$, $P_3 = \langle w, x, y, t \rangle$ in residual networks.

Then we can perform the following 5 steps: (Verify it yourself.)

Step	Augmenting Path	Sent Flow
1		
2		
3		
4		
5		



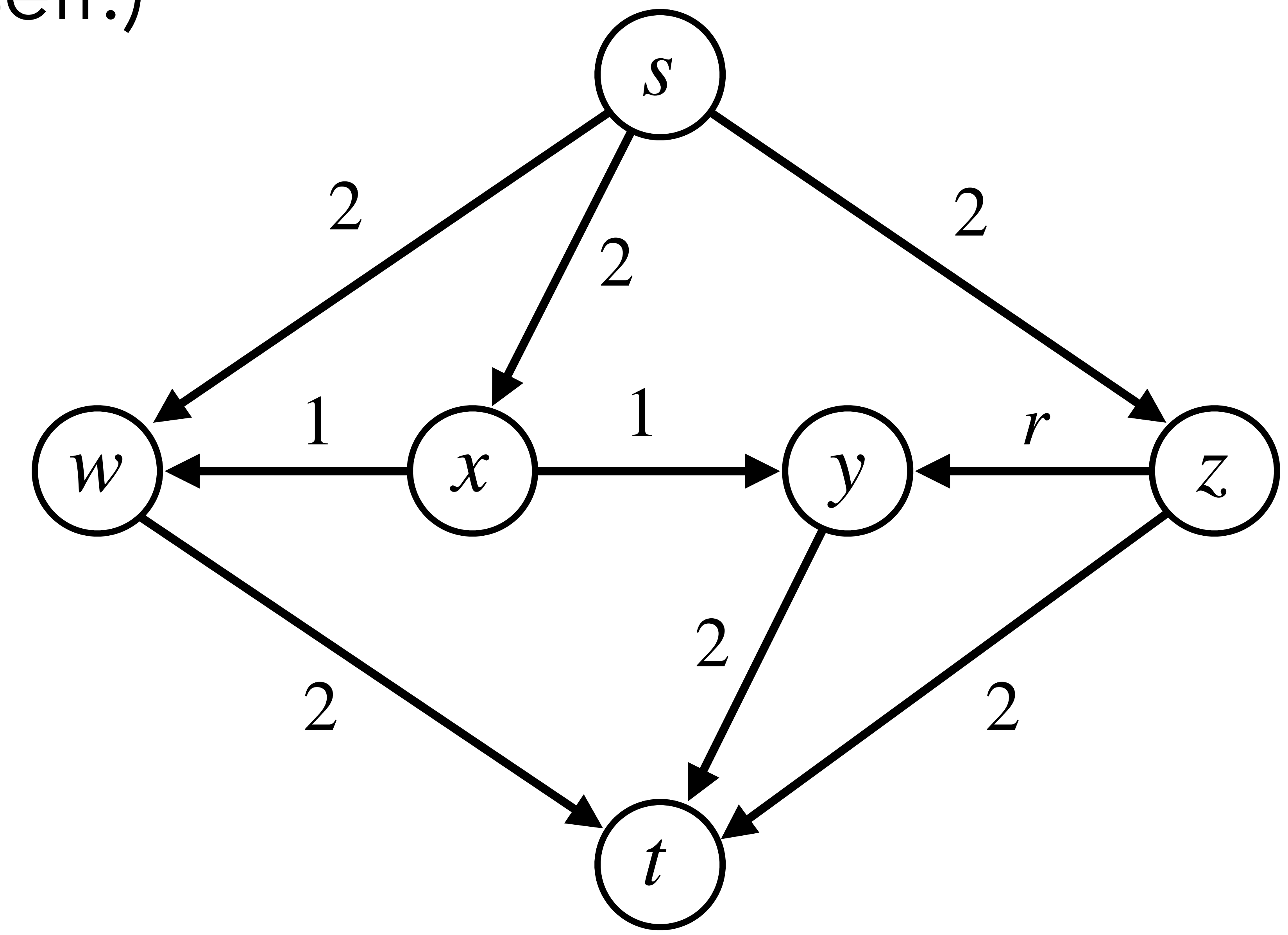
$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

Let $P = \langle s, x, y, z \rangle$, $P_1 = \langle s, z, y, x, w \rangle$, $P_2 = \langle s, w, y, z, t \rangle$, $P_3 = \langle w, x, y, t \rangle$ in residual networks.

Then we can perform the following 5 steps: (Verify it yourself.)

Step	Augmenting Path	Sent Flow
1	P	1
2		
3		
4		
5		



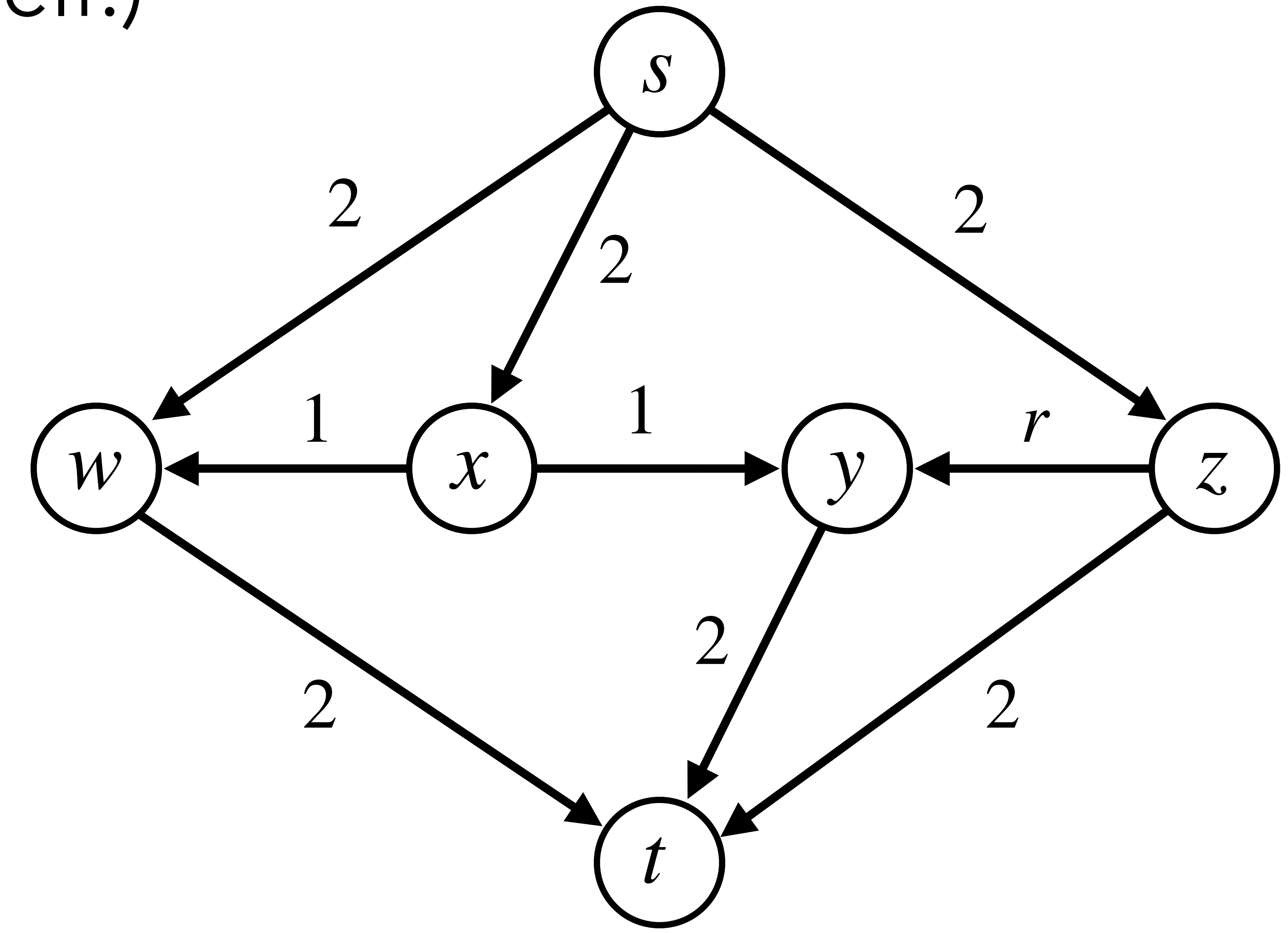
$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

Let $P = \langle s, x, y, z \rangle$, $P_1 = \langle s, z, y, x, w \rangle$, $P_2 = \langle s, w, y, z, t \rangle$, $P_3 = \langle w, x, y, t \rangle$ in residual networks.

Then we can perform the following 5 steps: (Verify it yourself.)

Step	Augmenting Path	Sent Flow
1	P	1
2	P_1	r
3		
4		
5		



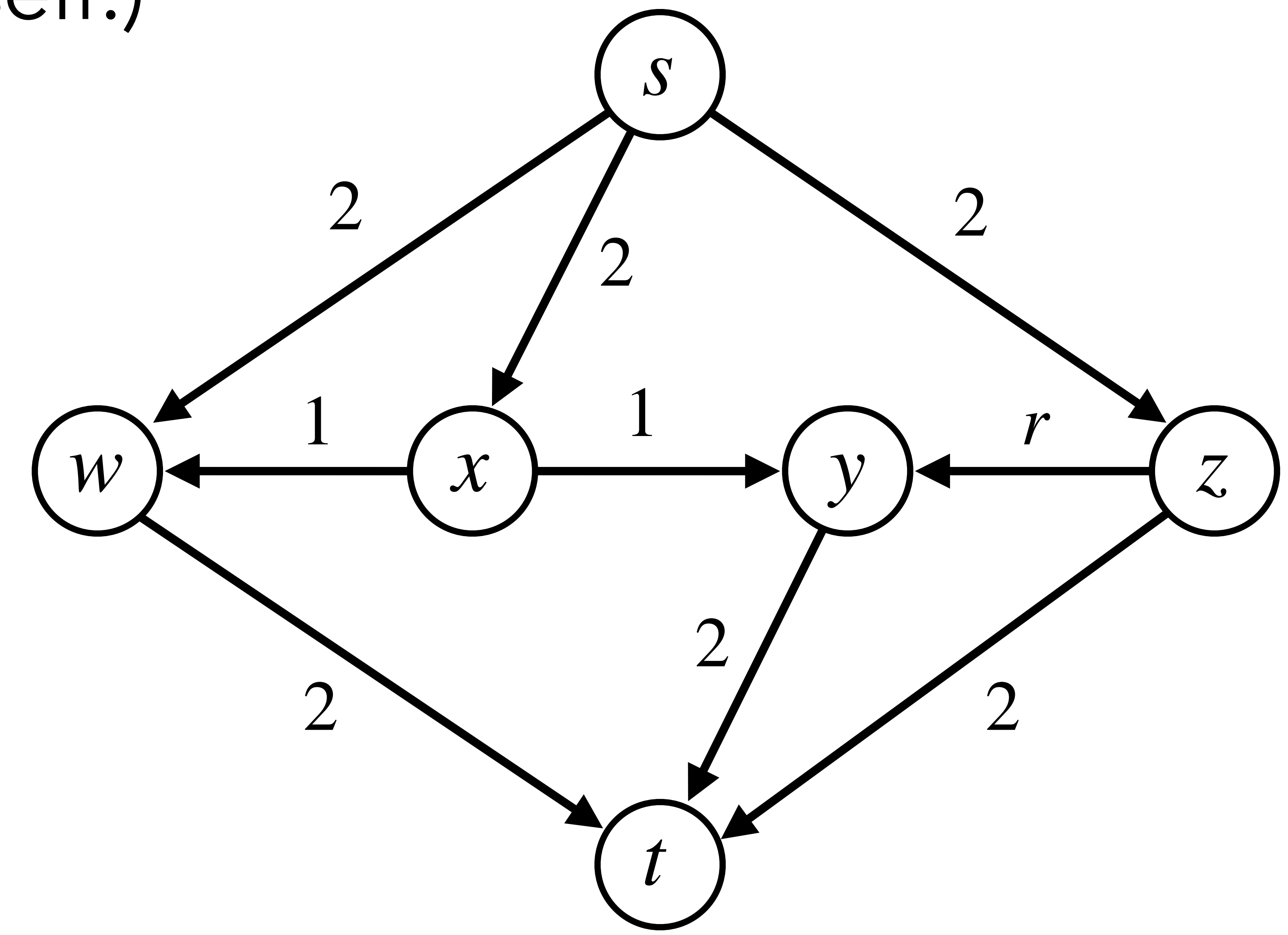
$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

Let $P = \langle s, x, y, z \rangle$, $P_1 = \langle s, z, y, x, w \rangle$, $P_2 = \langle s, w, y, z, t \rangle$, $P_3 = \langle w, x, y, t \rangle$ in residual networks.

Then we can perform the following 5 steps: (Verify it yourself.)

Step	Augmenting Path	Sent Flow
1	P	1
2	P_1	r
3	P_2	r
4		
5		



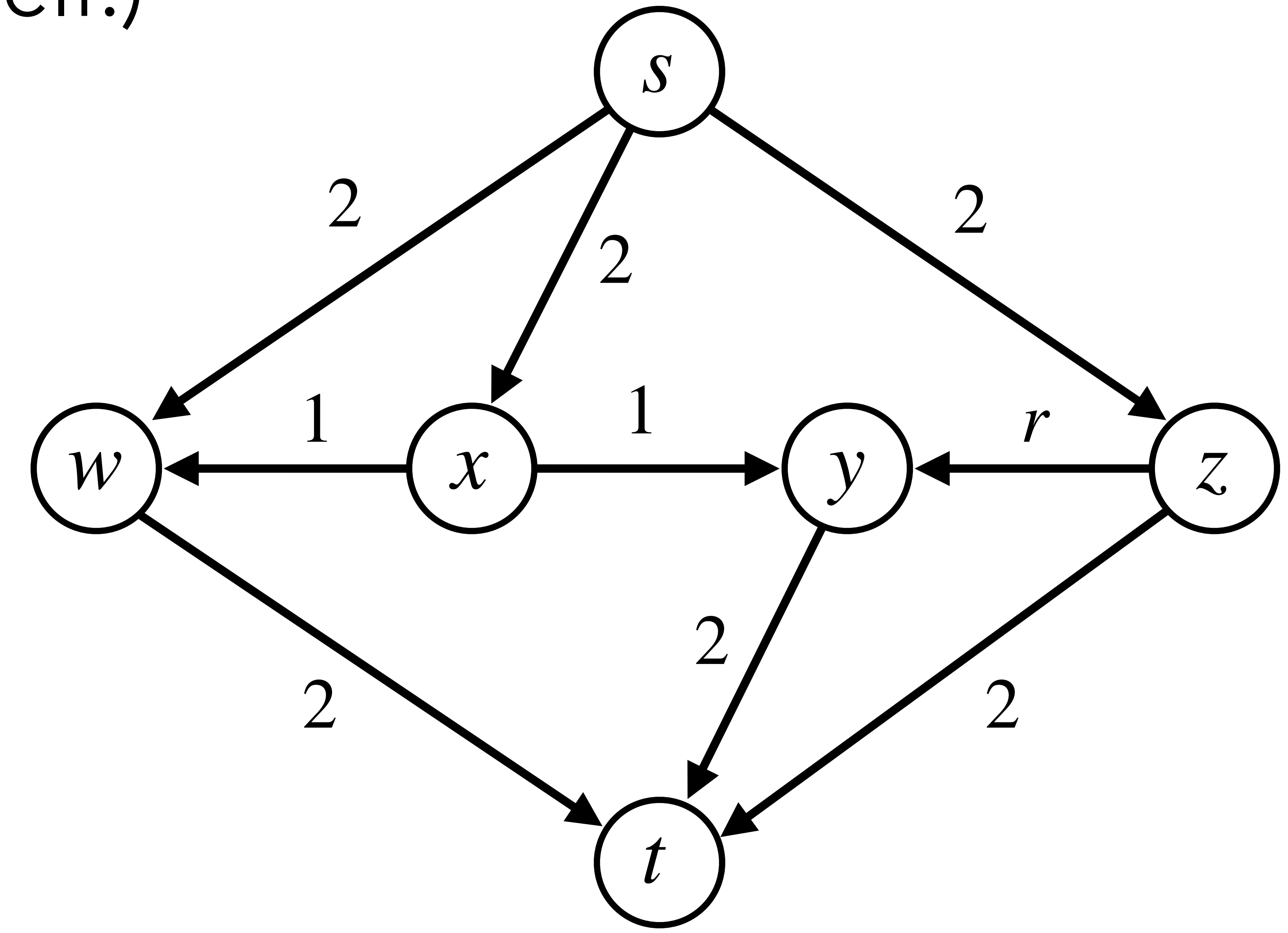
$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

Let $P = \langle s, x, y, z \rangle$, $P_1 = \langle s, z, y, x, w \rangle$, $P_2 = \langle s, w, y, z, t \rangle$, $P_3 = \langle w, x, y, t \rangle$ in residual networks.

Then we can perform the following 5 steps: (Verify it yourself.)

Step	Augmenting Path	Sent Flow
1	P	1
2	P_1	r
3	P_2	r
4	P_1	r^2
5		



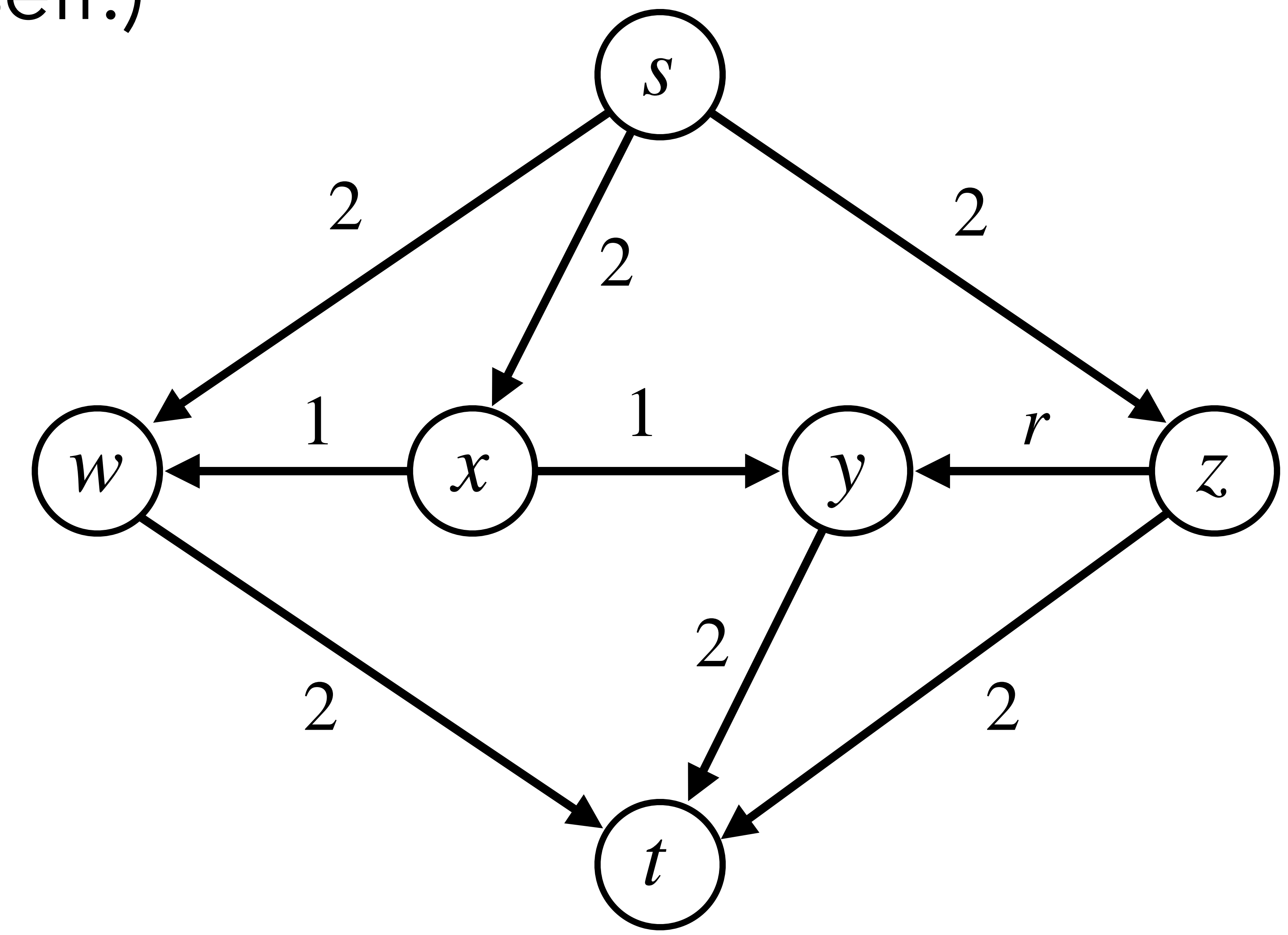
$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

Let $P = \langle s, x, y, z \rangle$, $P_1 = \langle s, z, y, x, w \rangle$, $P_2 = \langle s, w, y, z, t \rangle$, $P_3 = \langle w, x, y, t \rangle$ in residual networks.

Then we can perform the following 5 steps: (Verify it yourself.)

Step	Augmenting Path	Sent Flow
1	P	1
2	P_1	r
3	P_2	r
4	P_1	r^2
5	P_3	r^2



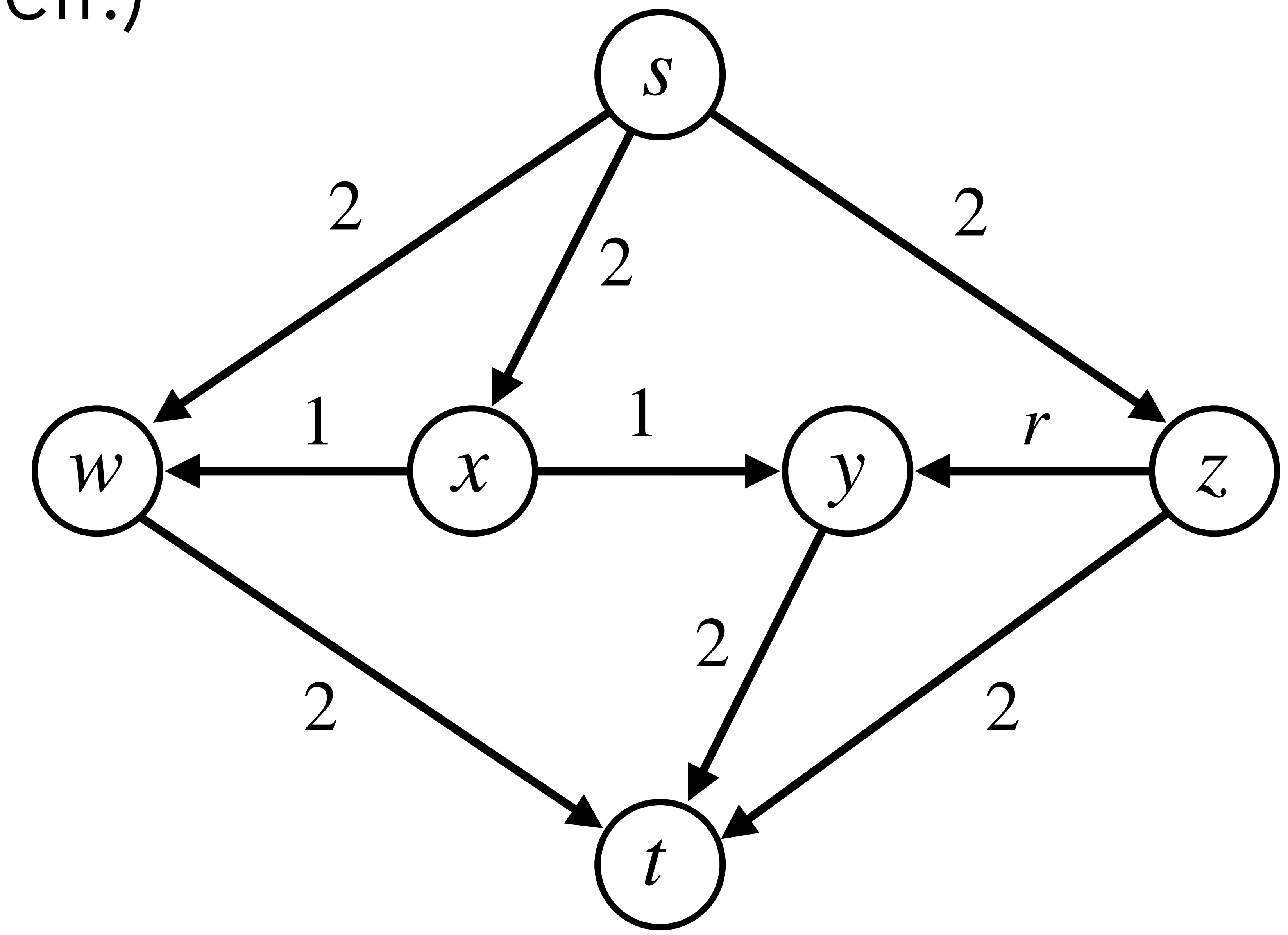
$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

Let $P = \langle s, x, y, z \rangle$, $P_1 = \langle s, z, y, x, w \rangle$, $P_2 = \langle s, w, y, z, t \rangle$, $P_3 = \langle w, x, y, t \rangle$ in residual networks.

Then we can perform the following 5 steps: (Verify it yourself.)

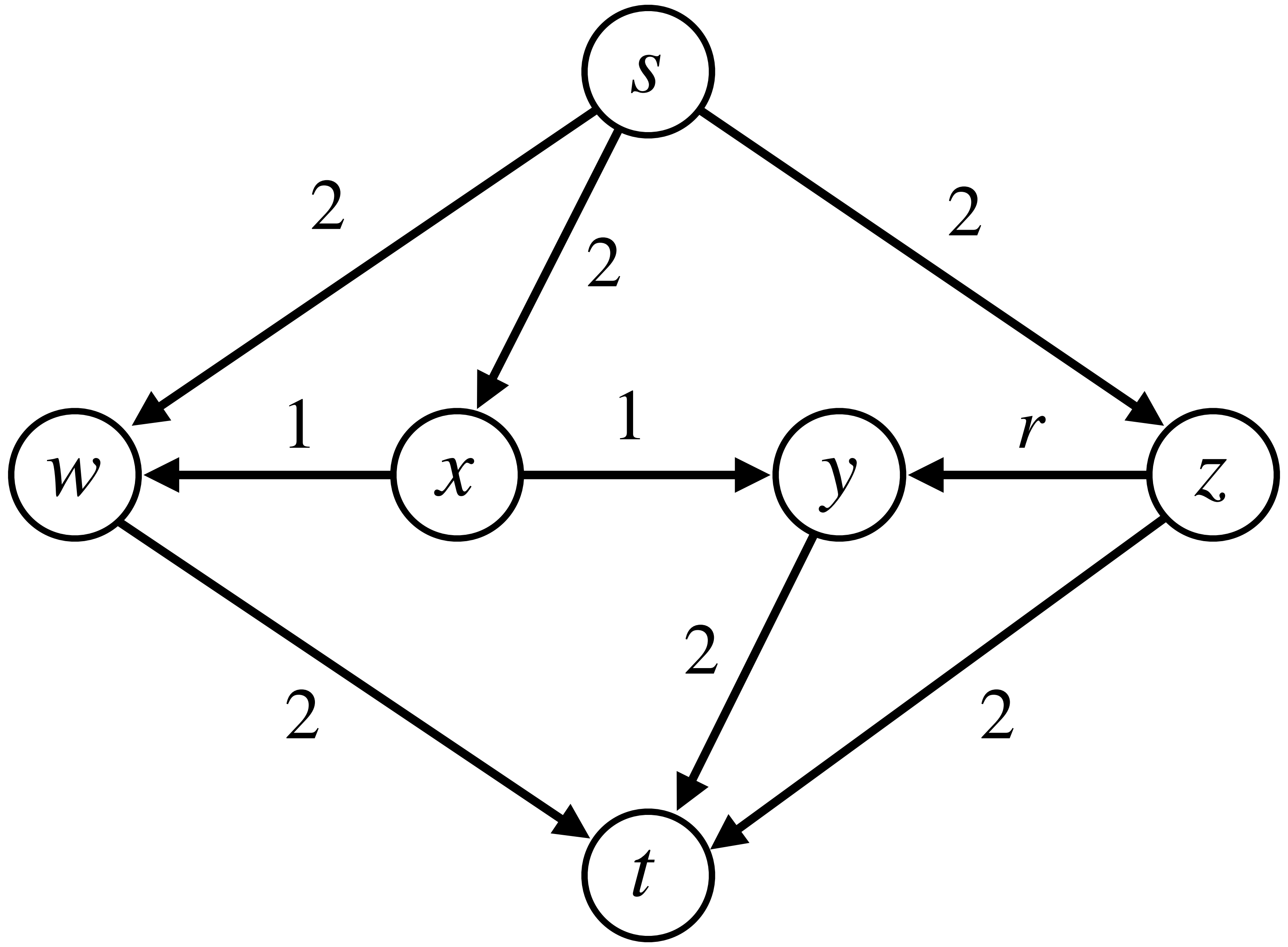
Step	Augmenting Path	Sent Flow
1	P	1
2	P_1	r
3	P_2	r
4	P_1	r^2
5	P_3	r^2



$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

We can repeat **steps $2 - 5$** with flows r^3, r^3, r^4, and r^4 and keep doing so…

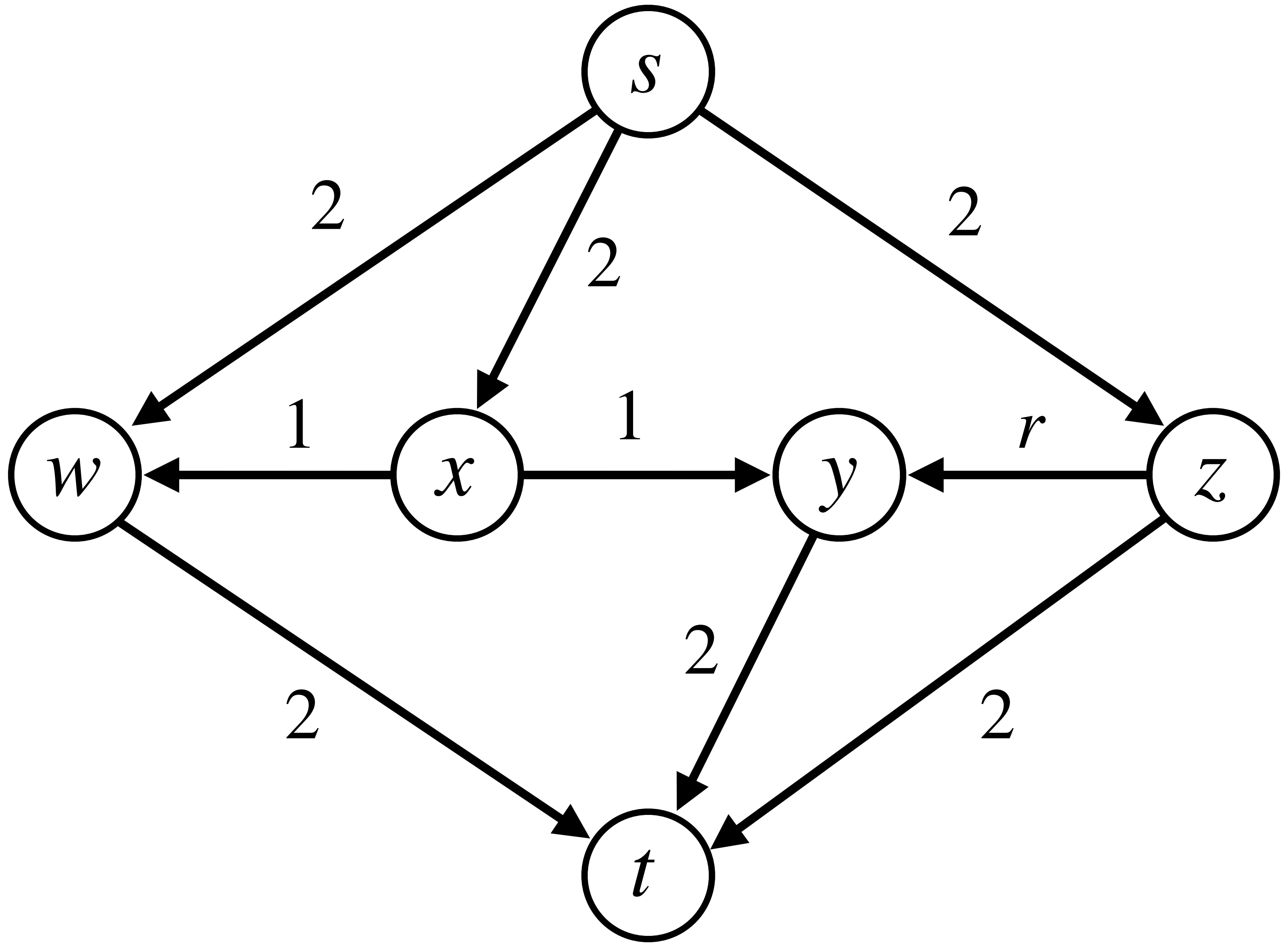
Ford-Fulkerson Method: A Non-terminating Case



$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

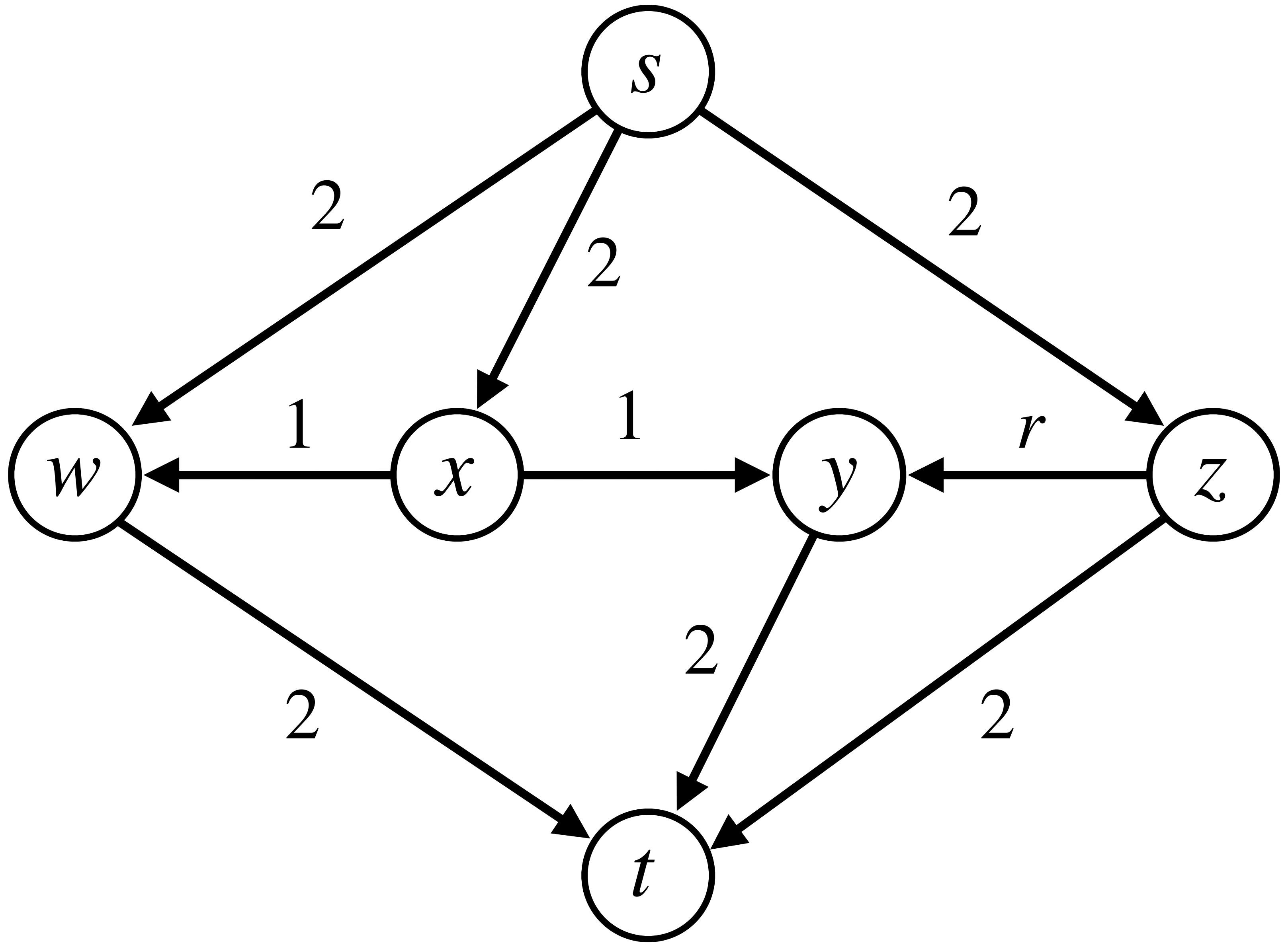
The total flow will converge to



$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

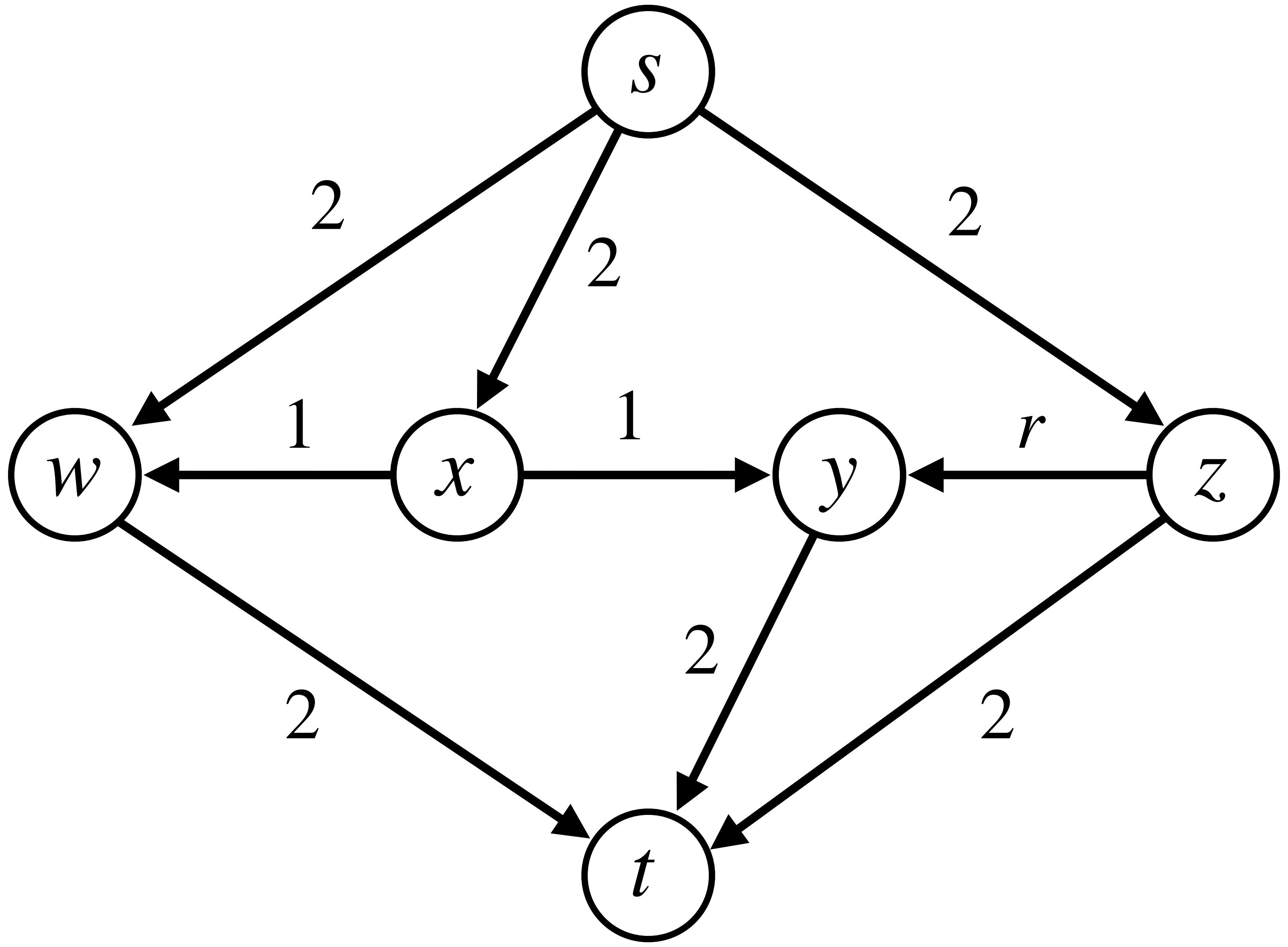
The total flow will converge to $1 + 2\sum_{i=1}^{\infty} r^i =$



$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

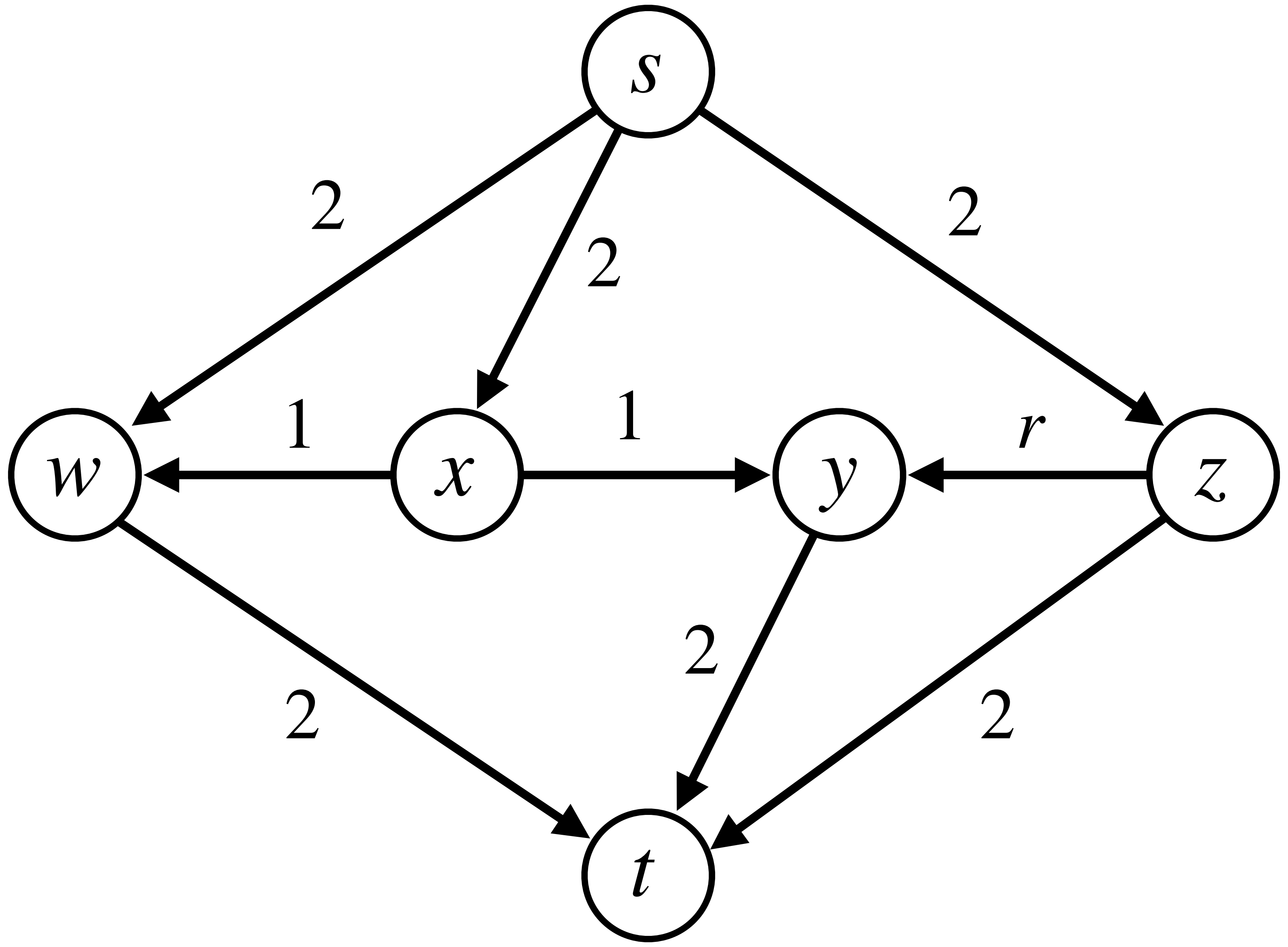
The total flow will converge to $1 + 2\sum_{i=1}^{\infty} r^i = 2 + 3r$



$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

The total flow will converge to $1 + 2\sum_{i=1}^{\infty} r^i = 2 + 3r < 5$.

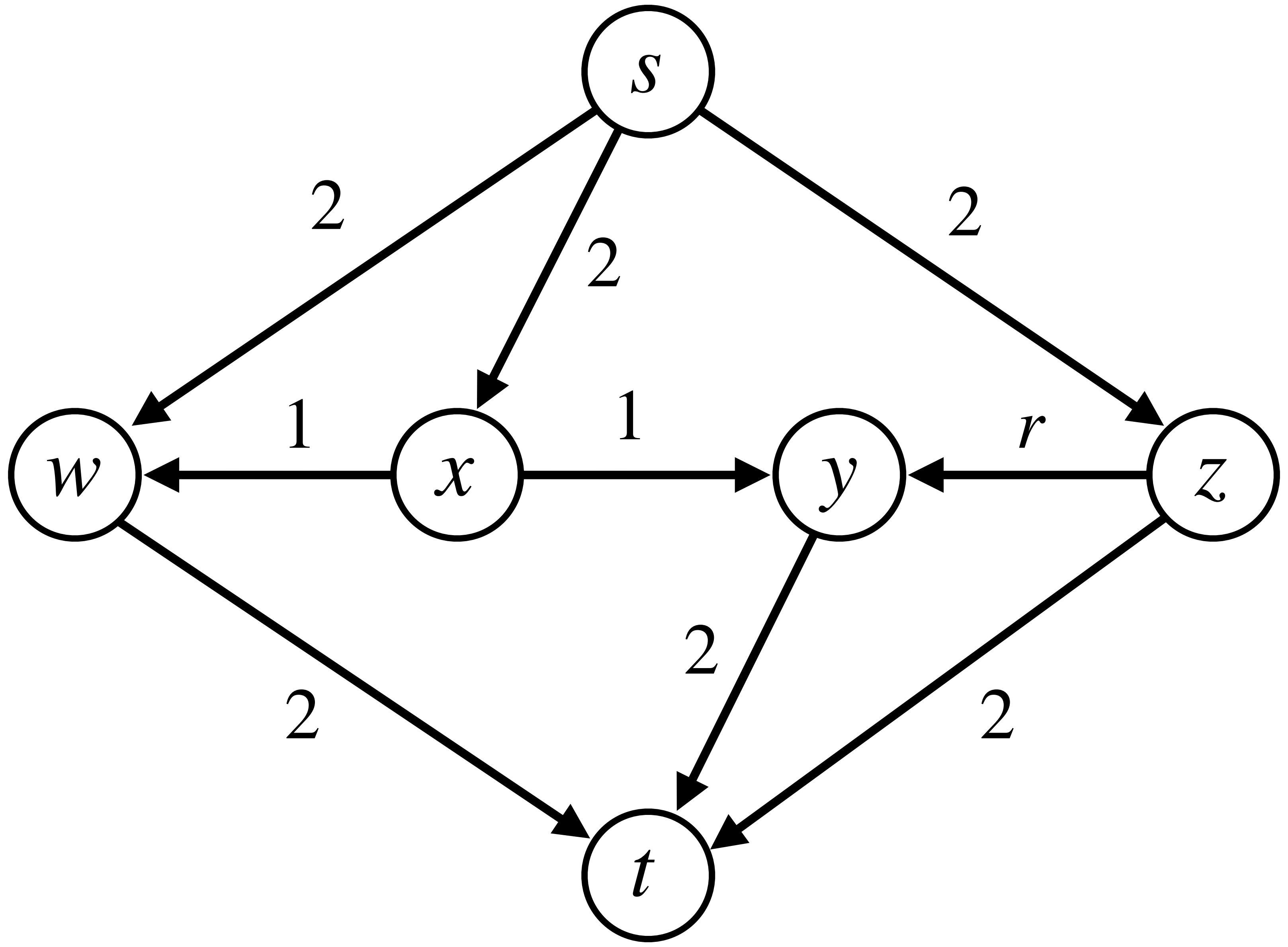


$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

The total flow will converge to $1 + 2\sum_{i=1}^{\infty} r^i = 2 + 3r < 5$.

There is a flow with value 5 in the given network.



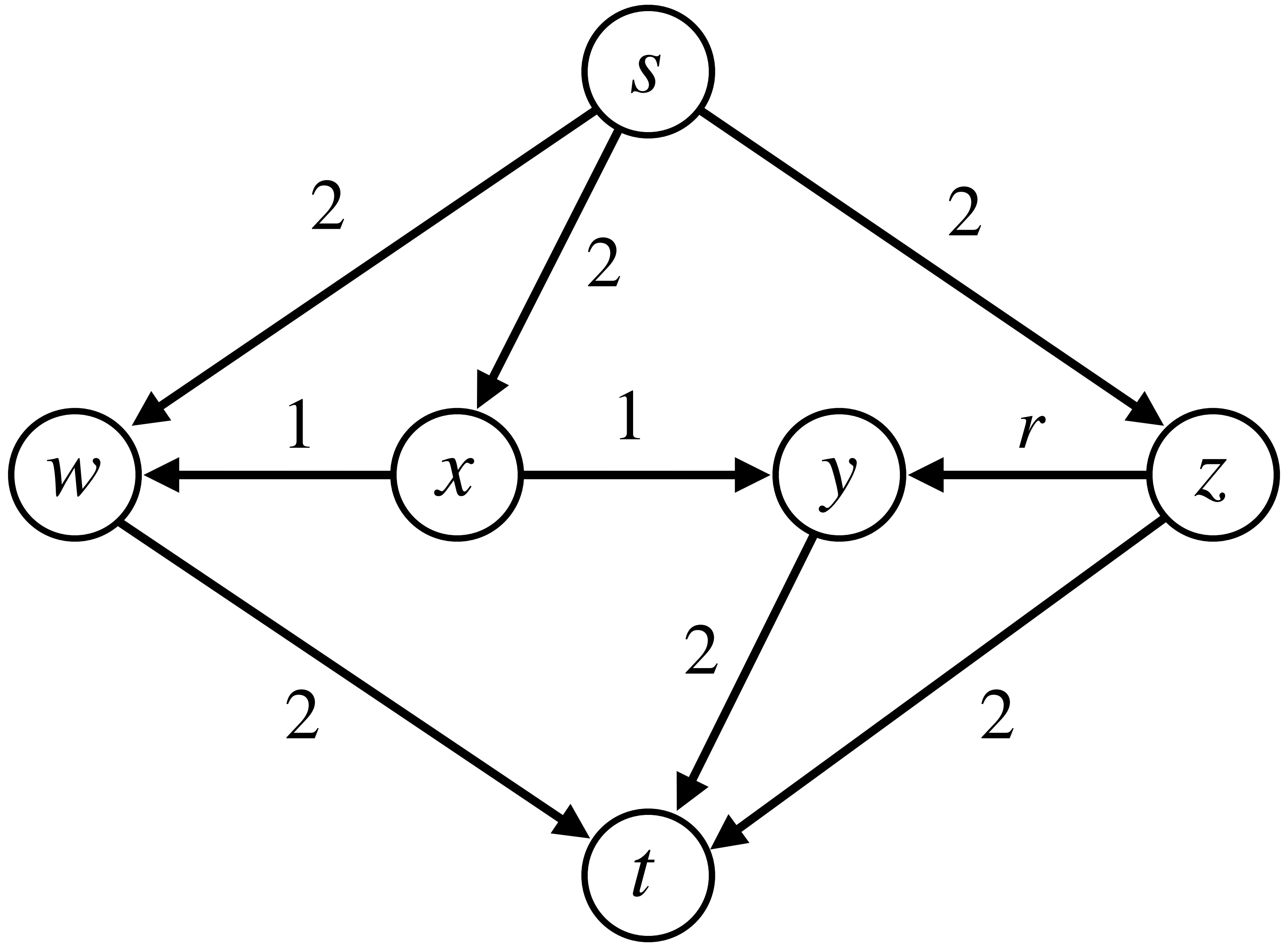
$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method: A Non-terminating Case

The total flow will converge to $1 + 2\sum_{i=1}^{\infty} r^i = 2 + 3r < 5$.

There is a flow with value 5 in the given network.

Hence, the algorithm will never terminate.



$r = (\sqrt{5} - 1)/2$ is chosen so that $r^2 = 1 - r$

Ford-Fulkerson Method

Will Ford-Fulkerson terminate when capacities are rationals?

Ford-Fulkerson Method

Will Ford-Fulkerson terminate when capacities are rationals?

Yes, prove it yourself.

Ford-Fulkerson Method: Correctness

Ford-Fulkerson Method: Correctness

Defn: A cut (S, T) of a flow network $G = (V, E)$

Ford-Fulkerson Method: Correctness

Defn: A cut (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ so that

Ford-Fulkerson Method: Correctness

Defn: A cut (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ so that $s \in S$ and $t \in T$.

Ford-Fulkerson Method: Correctness

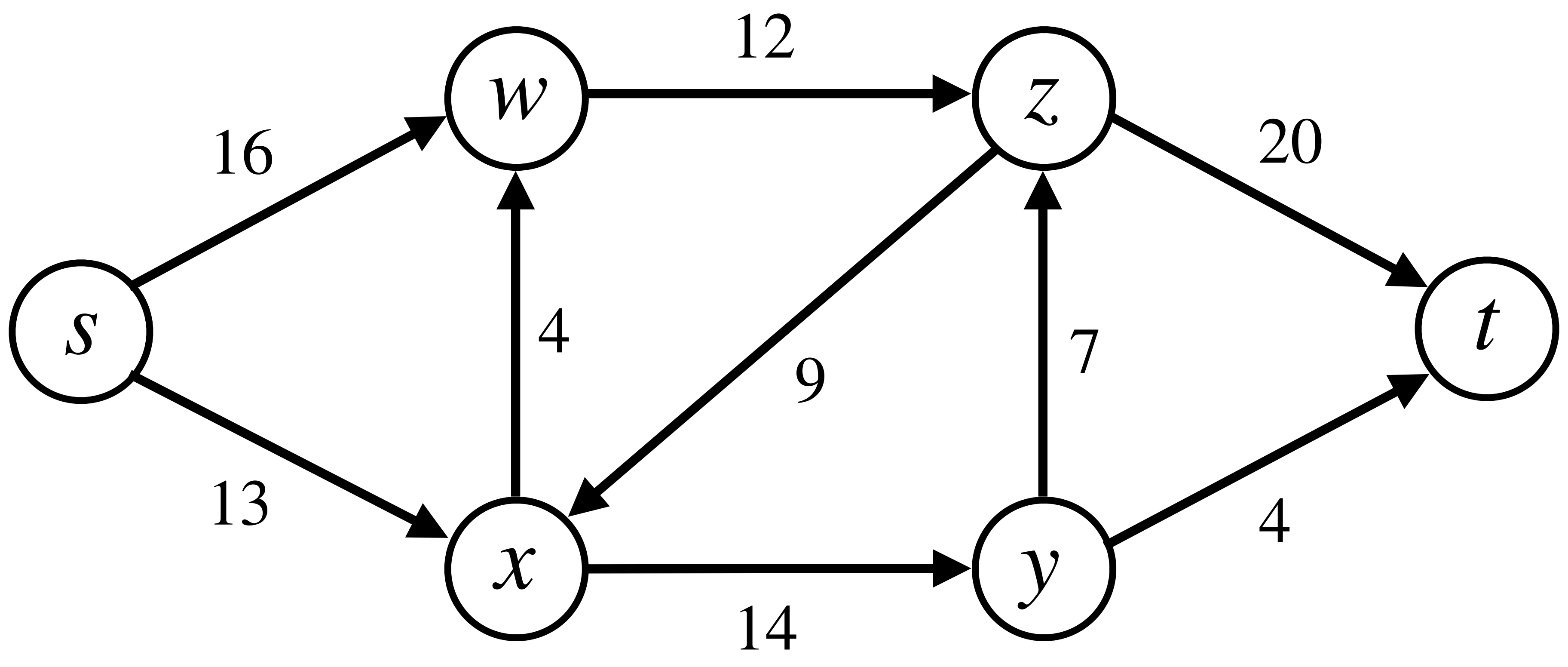
Defn: A cut (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ so that $s \in S$ and $t \in T$.

Examples:

Ford-Fulkerson Method: Correctness

Defn: A cut (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ so that $s \in S$ and $t \in T$.

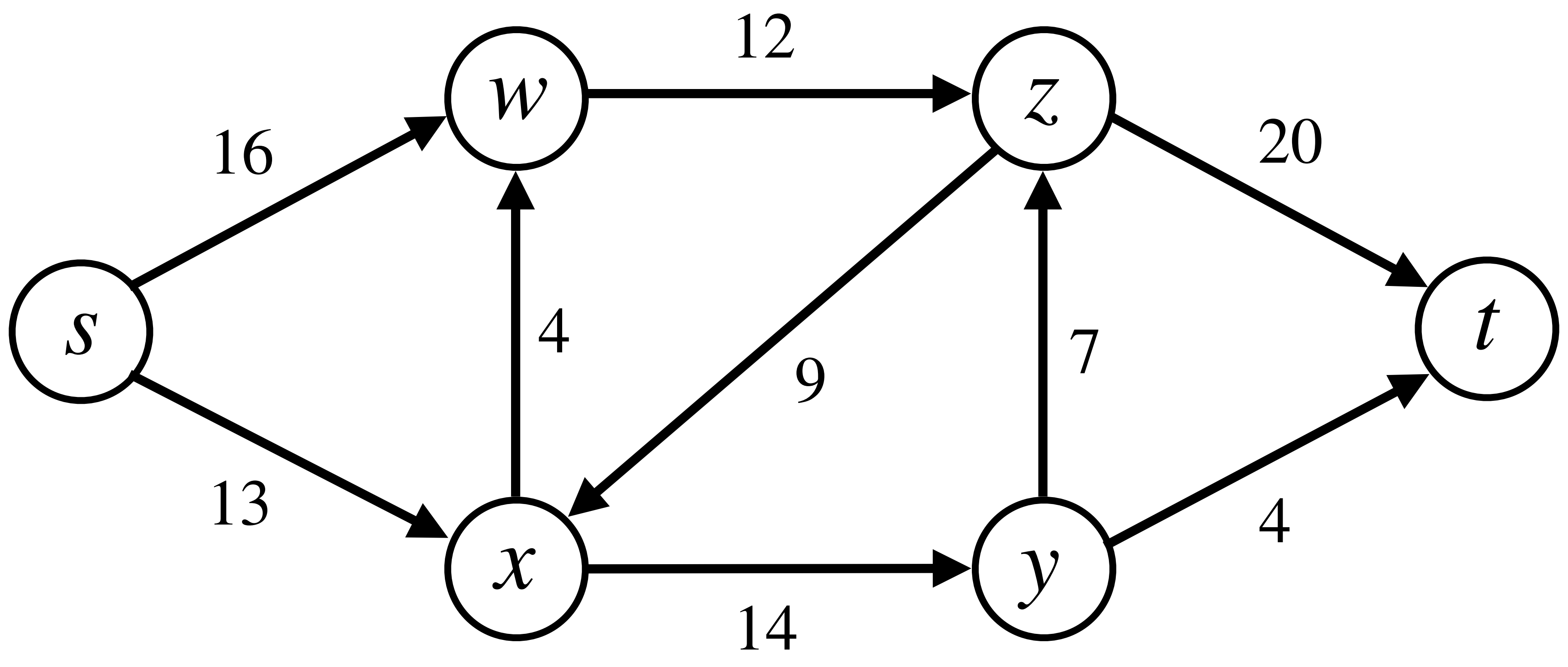
Examples:

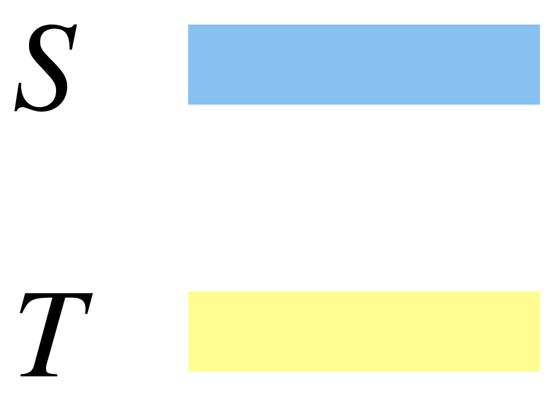


Ford-Fulkerson Method: Correctness

Defn: A cut (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ so that $s \in S$ and $t \in T$.

Examples:

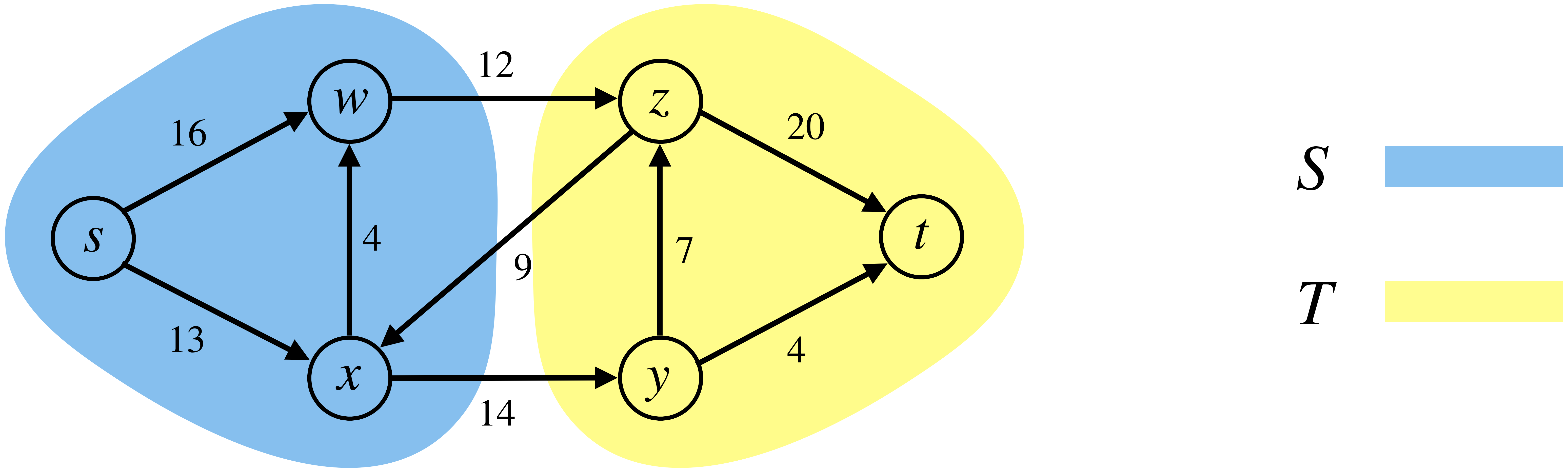




Ford-Fulkerson Method: Correctness

Defn: A cut (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ so that $s \in S$ and $t \in T$.

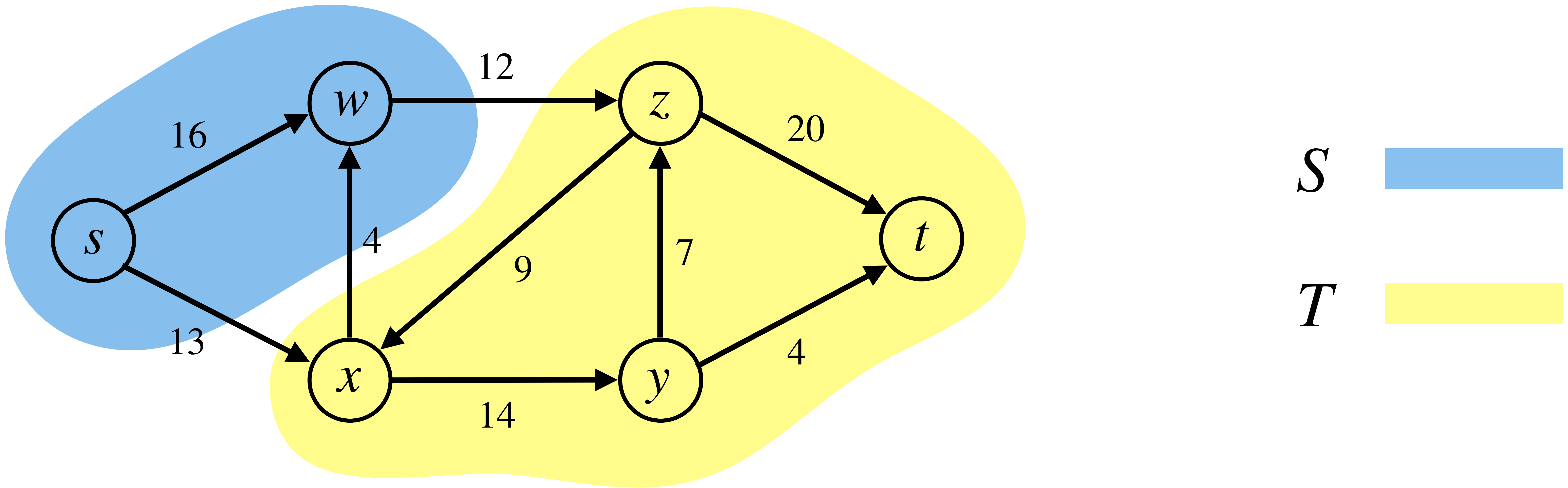
Examples:



Ford-Fulkerson Method: Correctness

Defn: A cut (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ so that $s \in S$ and $t \in T$.

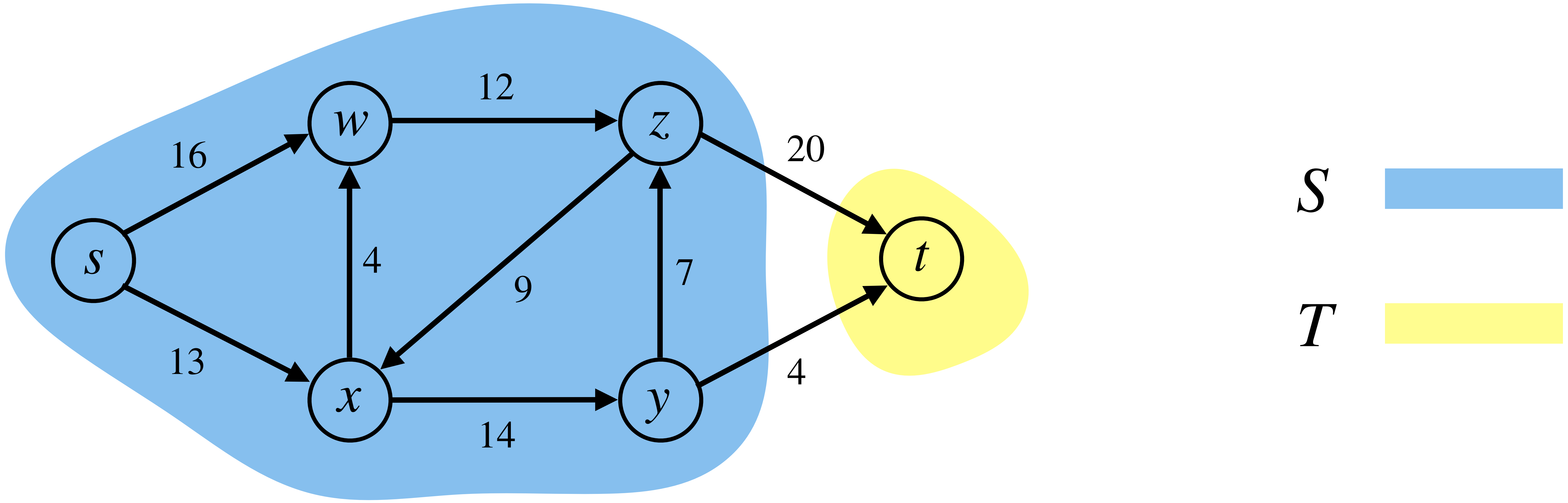
Examples:



Ford-Fulkerson Method: Correctness

Defn: A cut (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ so that $s \in S$ and $t \in T$.

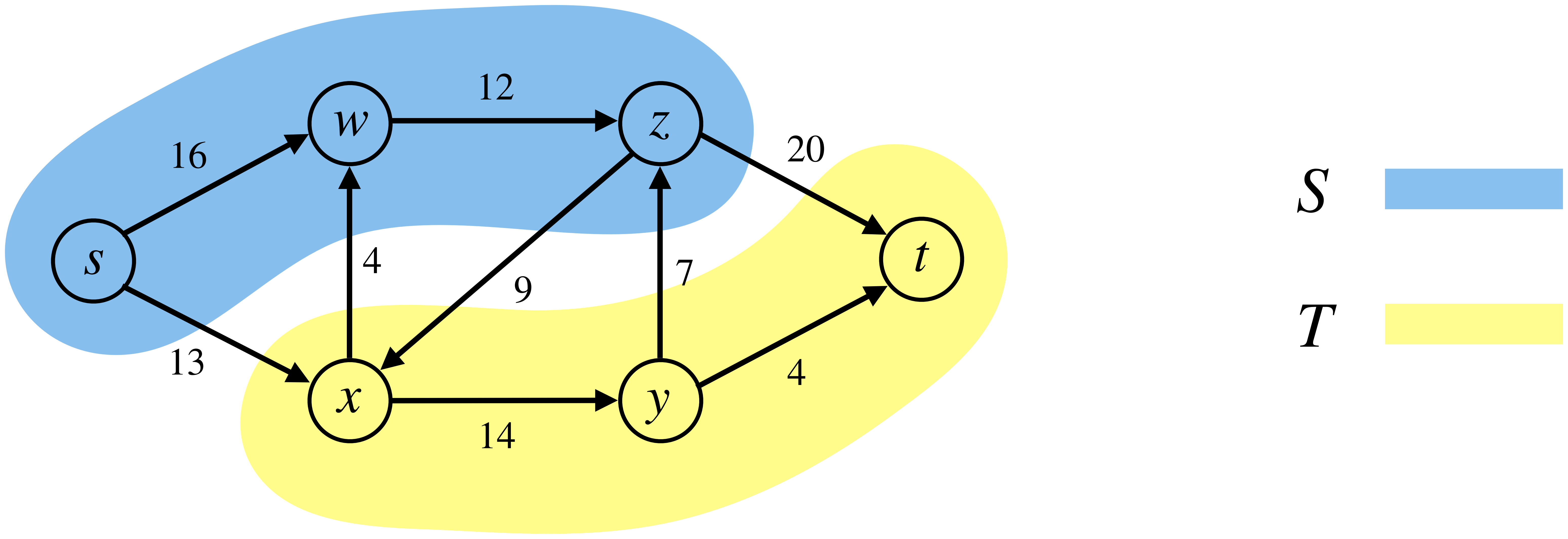
Examples:



Ford-Fulkerson Method: Correctness

Defn: A cut (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ so that $s \in S$ and $t \in T$.

Examples:



Ford-Fulkerson Method: Correctness

Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f,

Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u)$$

Total flow from S to T

Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Total flow from S to T

Total flow from T to S

Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u)$$

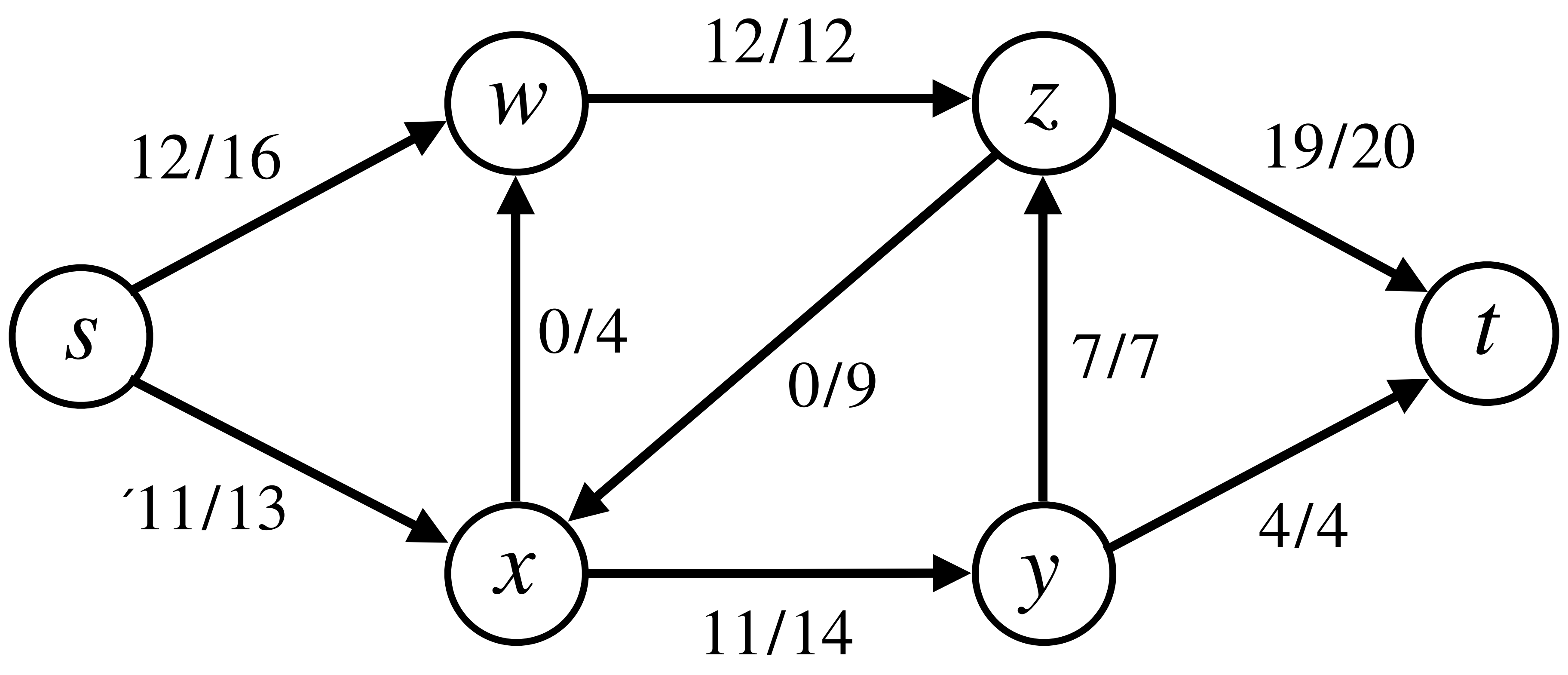
Examples:

Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:

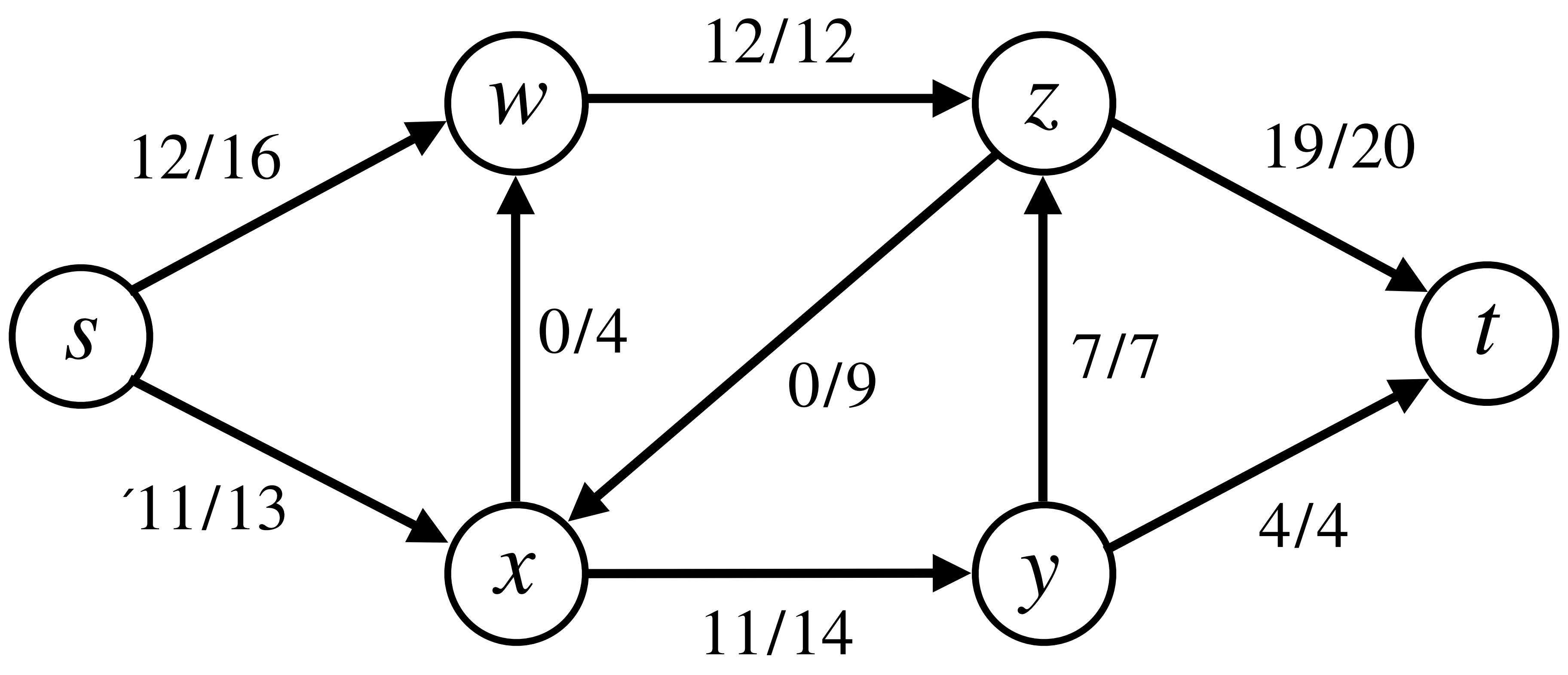


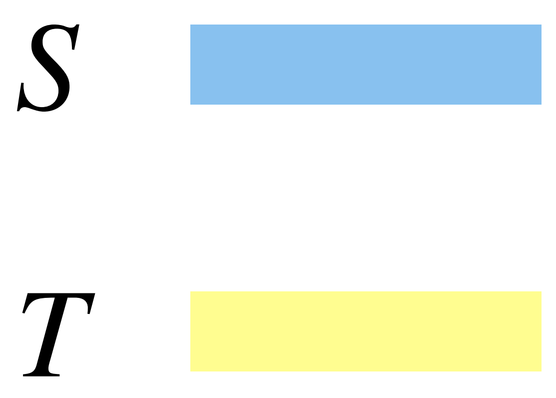
Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:



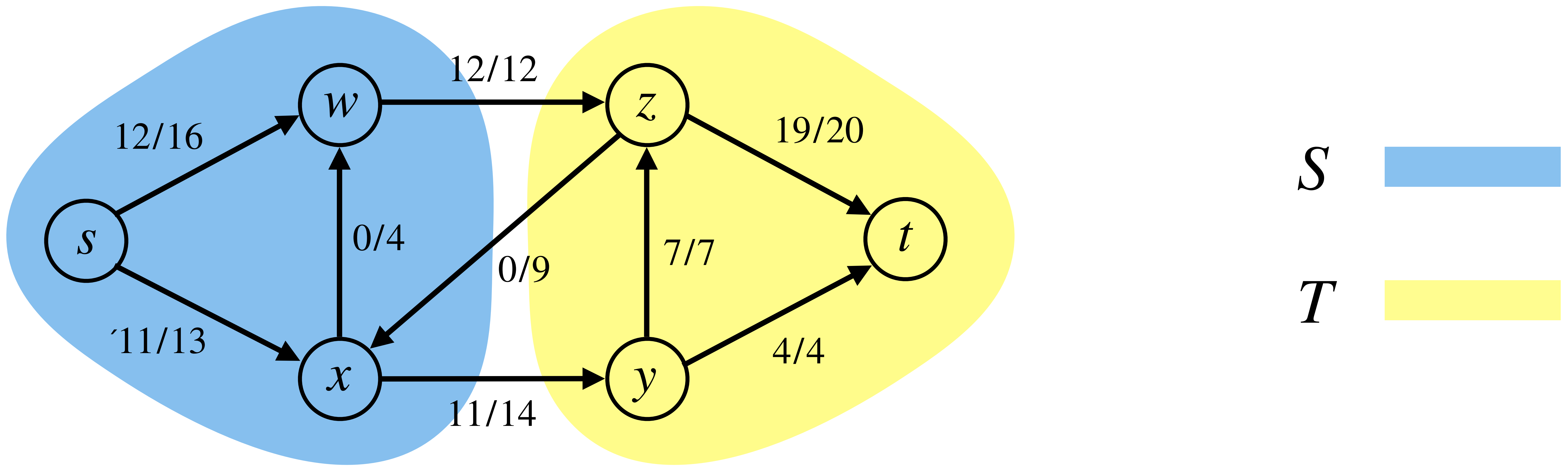


Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:



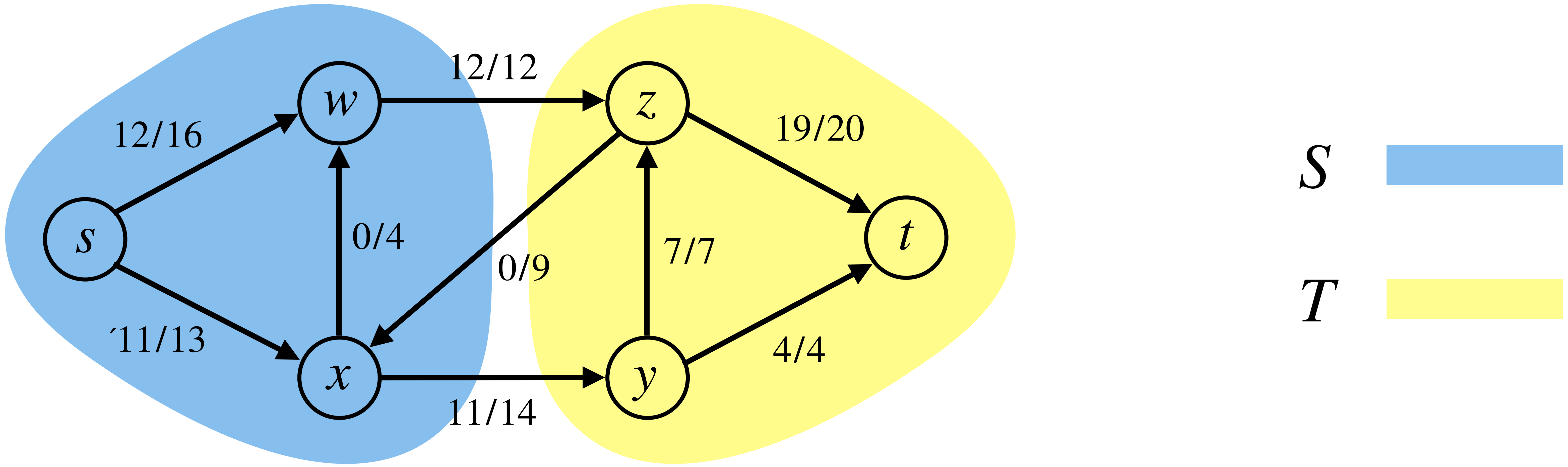
Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:

$f(S,T) = 23$

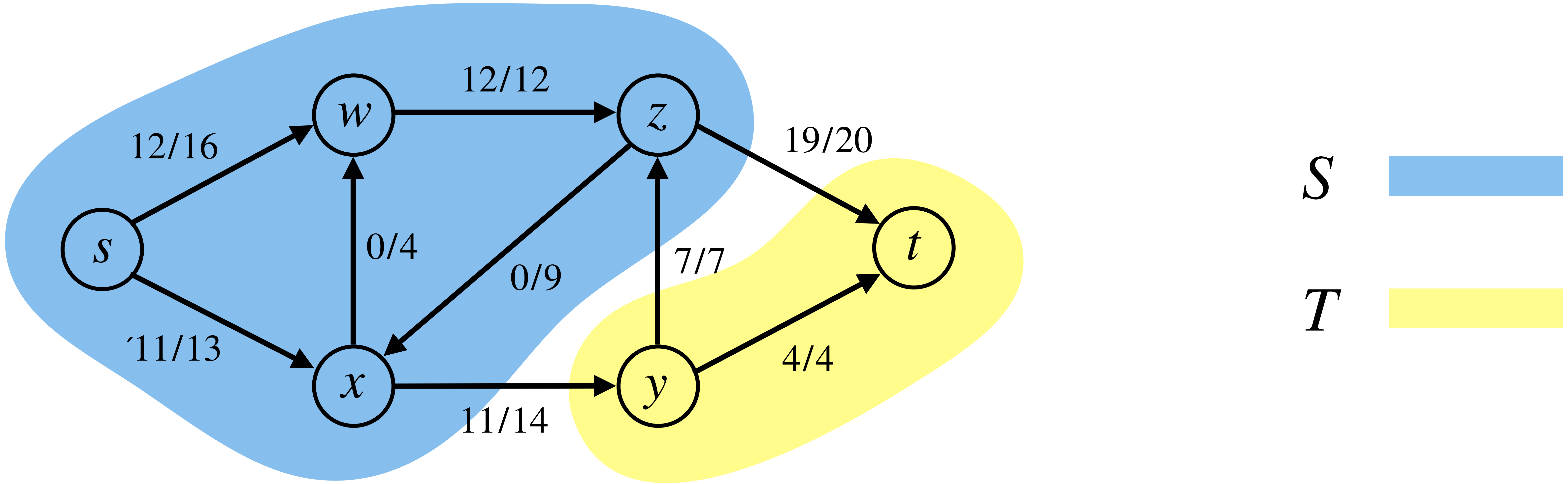


Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:



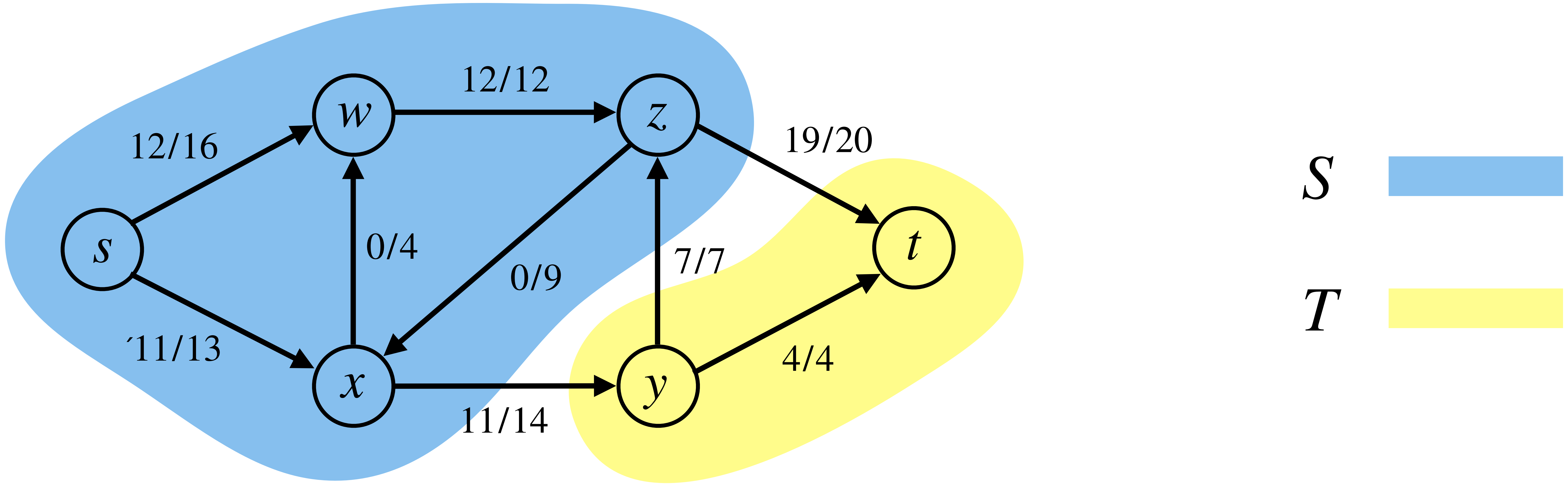
Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:

$f(S, T) = 23$

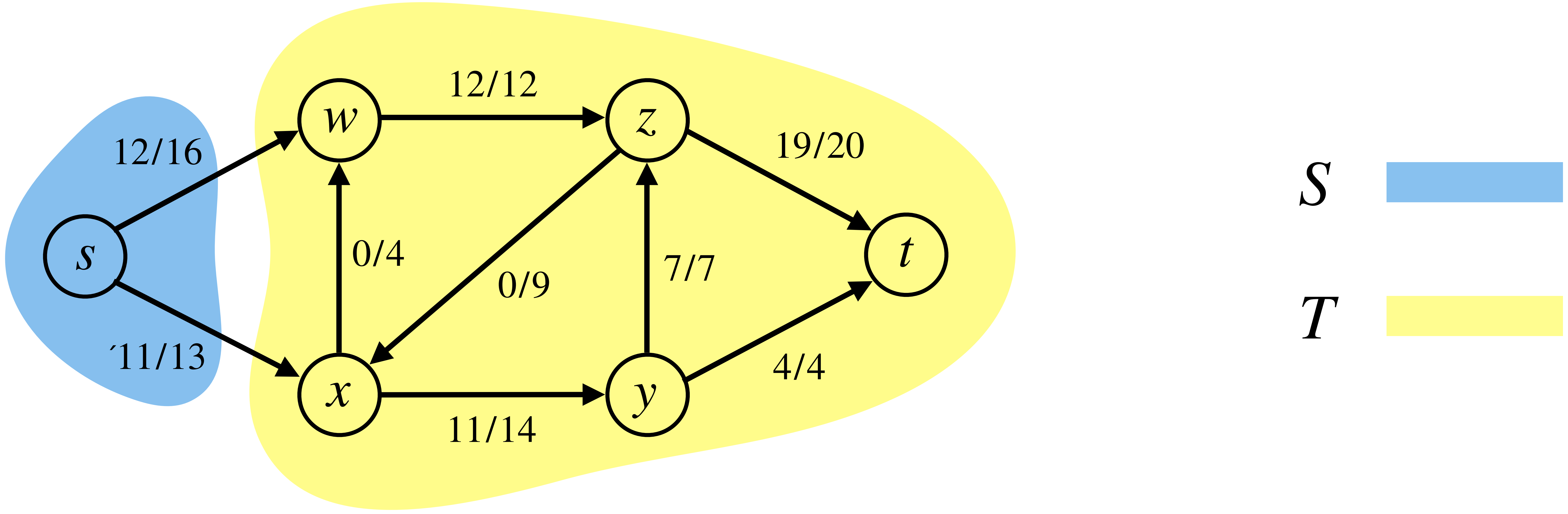


Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:



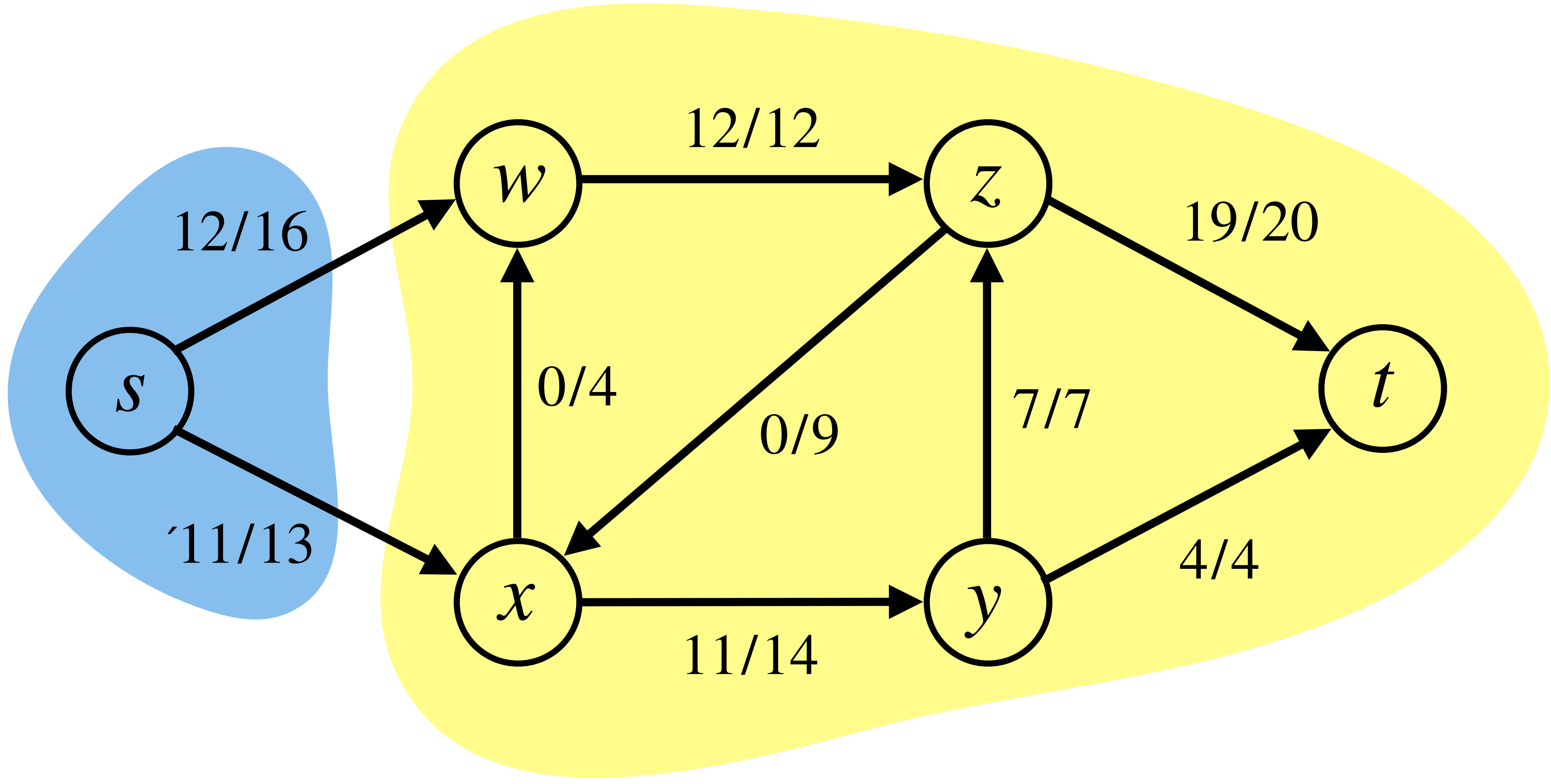
Ford-Fulkerson Method: Correctness

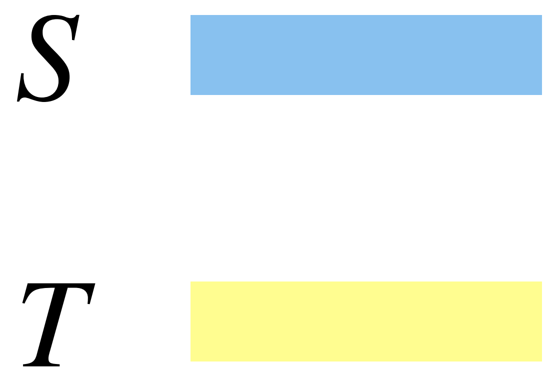
Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:

$f(S, T) = 23$



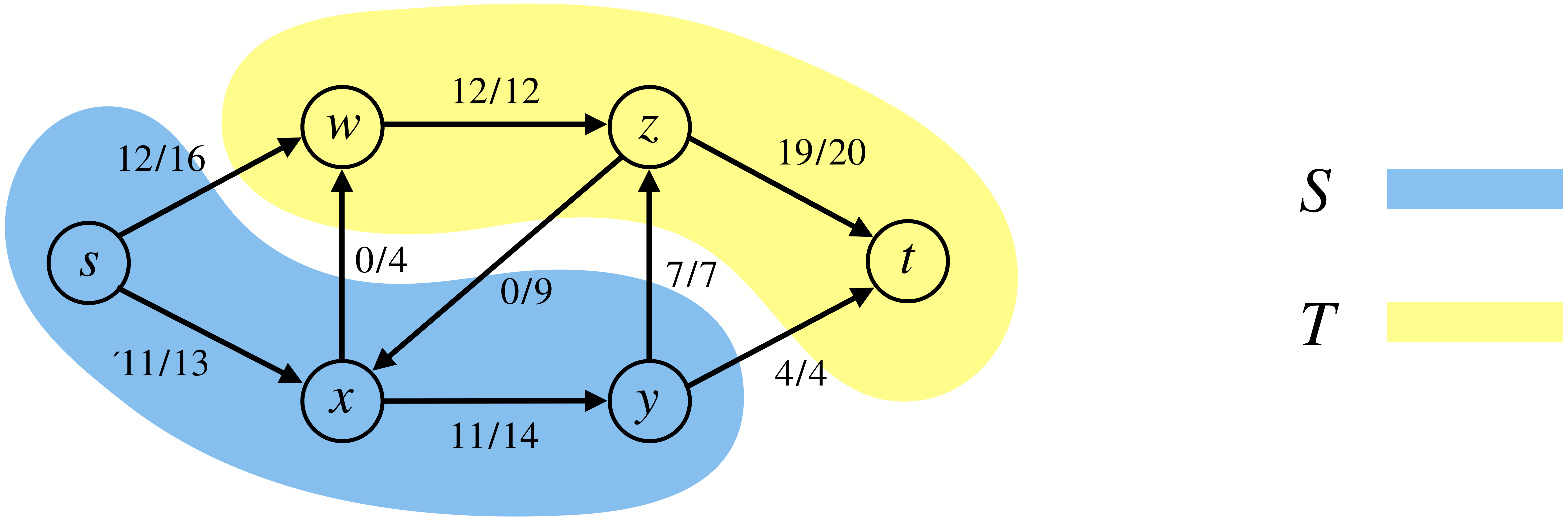


Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:



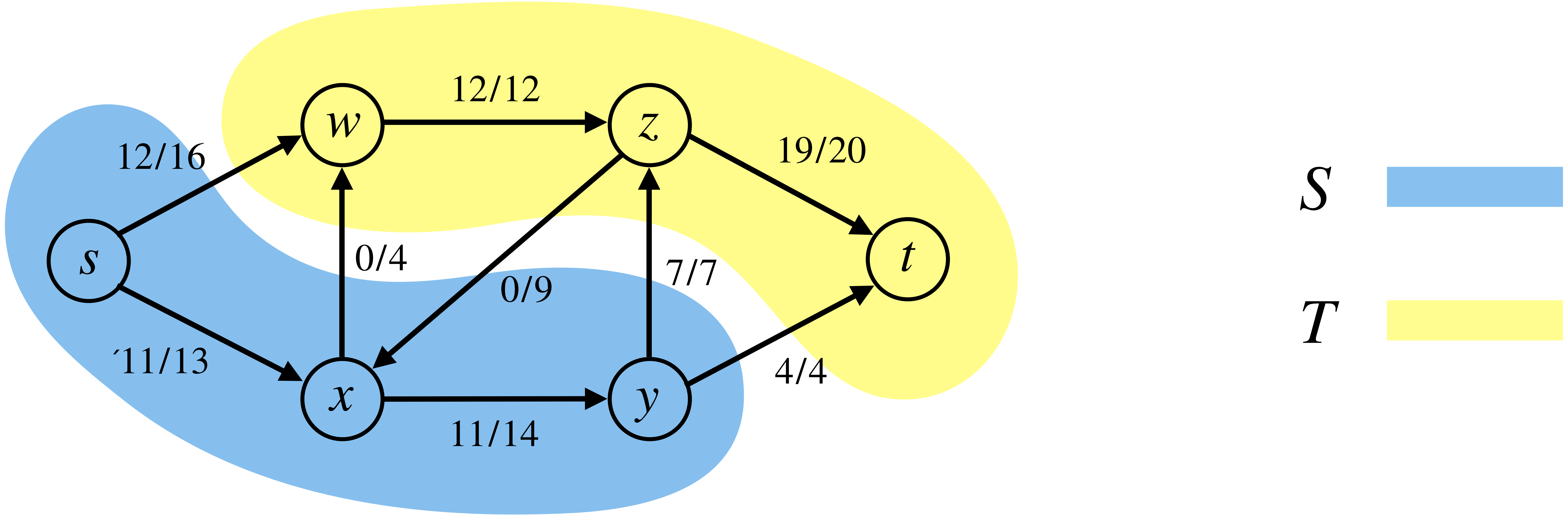
Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:

$f(S, T) = 23$



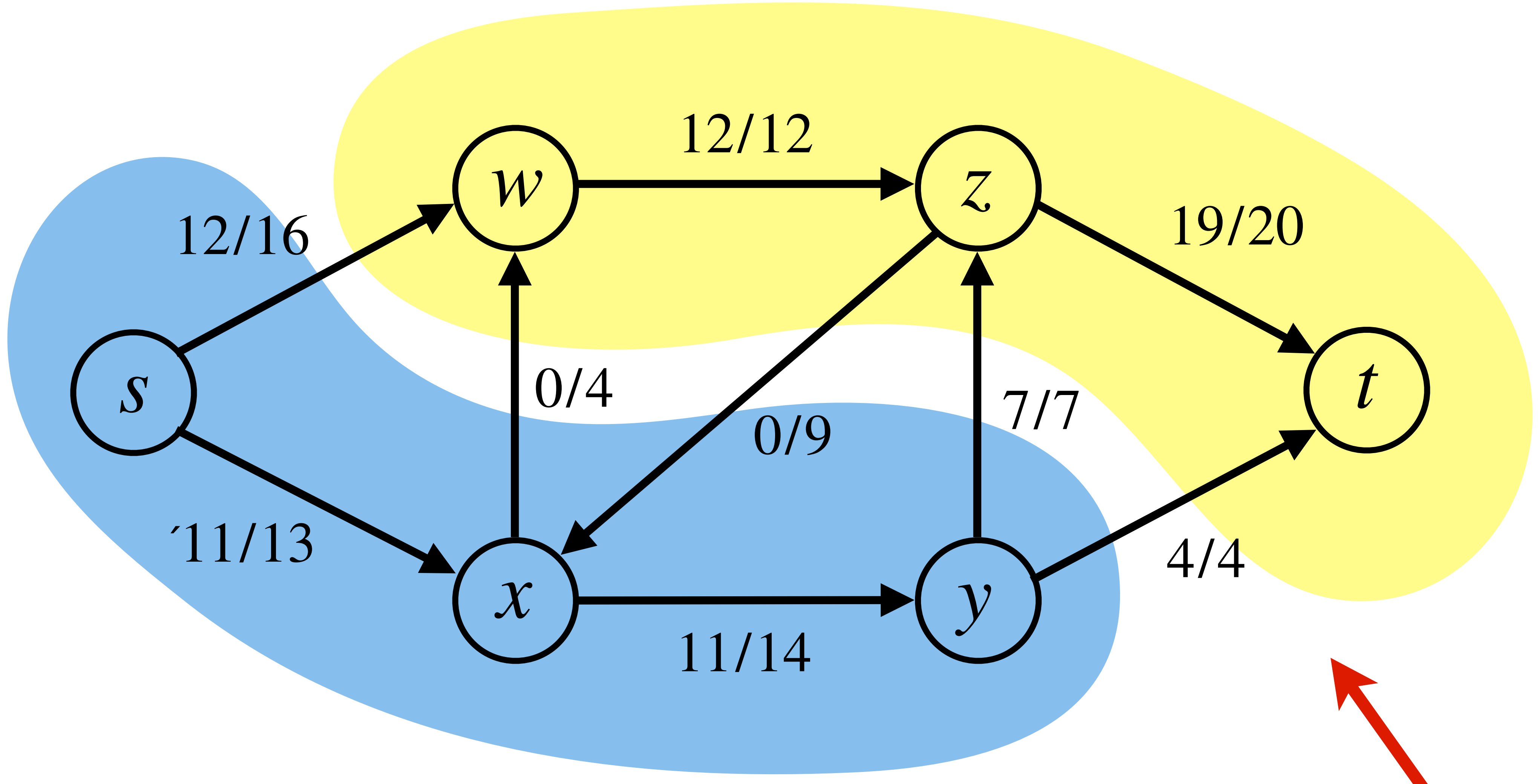
Ford-Fulkerson Method: Correctness

Defn: In a flow network $G = (V, E)$ with flow f, the net flow $f(S, T)$ across the cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

Examples:

$f(S, T) = 23$



S

T

Why the net flow is the same for every cut?