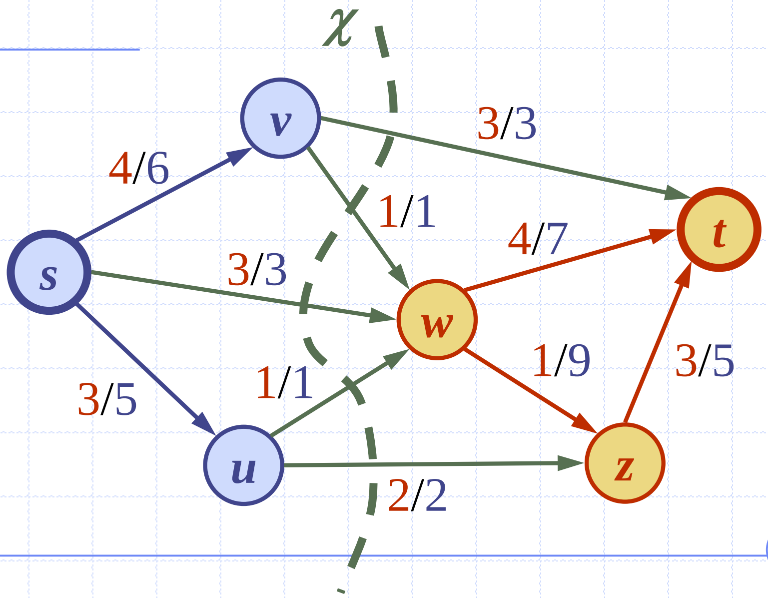
Chapter 8: Network Flow

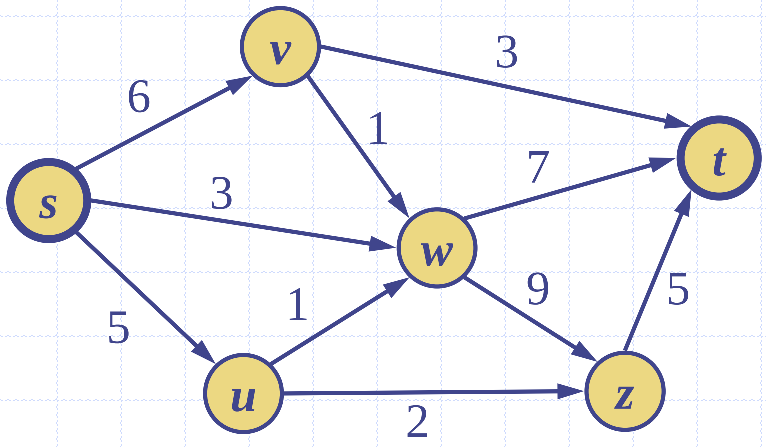


Outline and Reading

- Flow Networks
  - Flow (§8.1.1)
  - Cut (§8.1.2)
- Maximum flow
  - Augmenting Path (§8.2.1)
  - Maximum Flow and Minimum Cut (§8.2.1)
  - Ford-Fulkerson's Algorithm (§8.2.2-8.2.3)
- Sections §8.2.4-8.5 on Matching and Minimum Flow are optional.

Flow Network

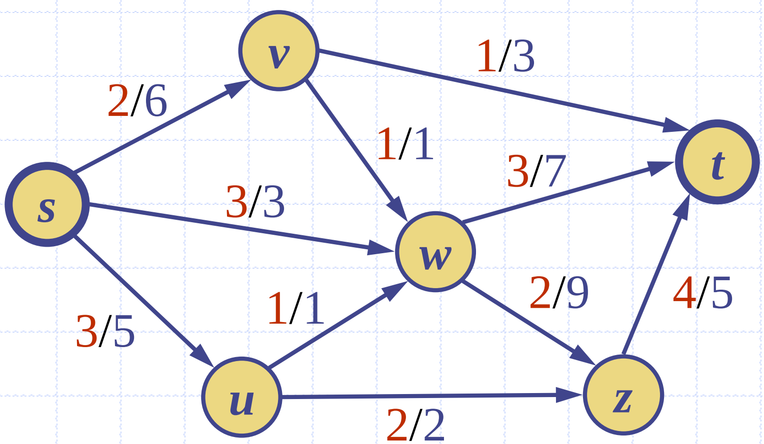
- A flow network (or just network) N consists of
  - A weighted digraph G with nonnegative integer edge weights, where the weight of an edge e is called the capacity $c(e)$ of e
  - Two distinguished vertices, s and t of G, called the source and sink, respectively, such that s has no incoming edges and t has no outgoing edges.
- Example:



Flow

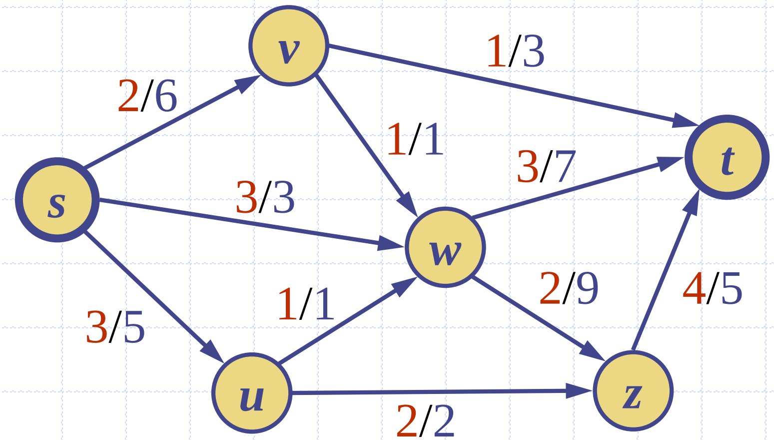
- A flow f for a network N is is an assignment of an integer value $f(e)$ to each edge e that satisfies the following properties:
  - Capacity Rule: For each edge e, $0 \le f(e) \le c(e)$
  - Conservation Rule: For each vertex $v \neq s,t$ $$\sum_{e \in E^-(v)} f(e) = \sum_{e \in E^+(v)} f(e)$$

    where $E^-(v)$ and $E^+(v)$ are the incoming and outgoing edges of v, resp.
- The value of a flow f, denoted $|f|$, is the total flow from the source, which is the same as the total flow into the sink
- Example:

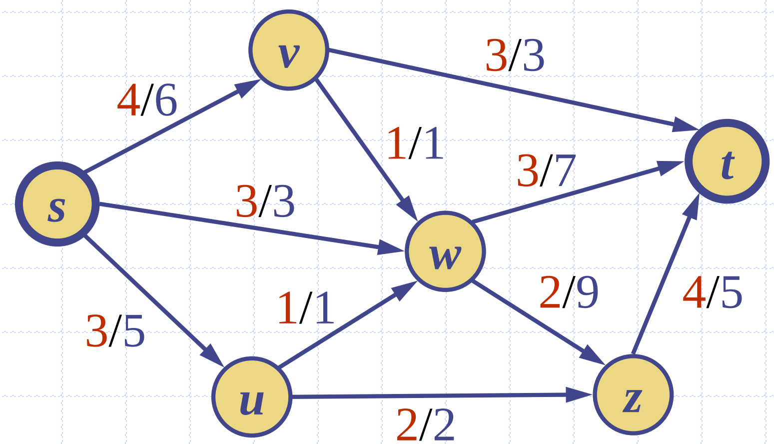


Maximum Flow

- A flow for a network N is said to be maximum if its value is the largest of all flows for N
- The maximum flow problem consists of finding a maximum flow for a given network N
- Applications
  - Hydraulic systems
  - Electrical circuits
  - Traffic movements
  - Freight transportation



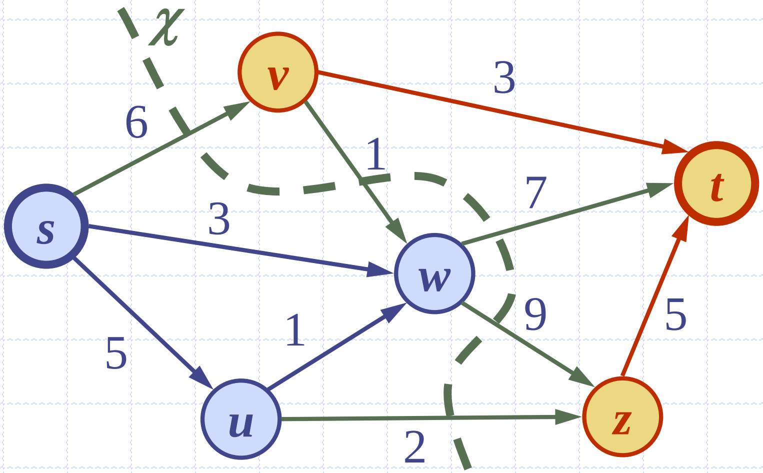
Flow of value 8 = 2 + 3 + 3 = 1 + 3 + 4

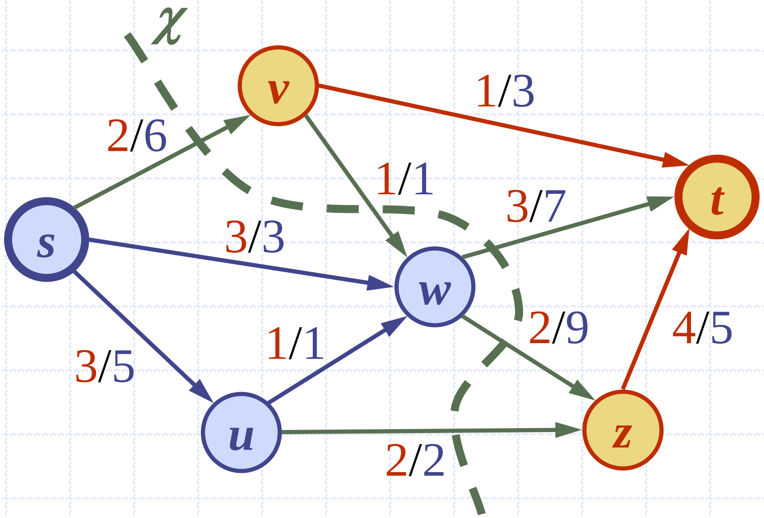


Maximum flow of value 10 = 4 + 3 + 3 = 3 + 3 + 4

Cut

- A cut of a network N with source s and sink t is a partition $\chi = (V_s, V_t)$ of the vertices of N such that $s \in V_s$ and $t \in V_t$
  - Forward edge of cut χ: origin in V_s and destination in V_t
  - Backward edge of cut χ: origin in V_t and destination in V_s
- Flow $f(\chi)$ across a cut χ: total flow of forward edges minus total flow of backward edges
- Capacity $c(\chi)$ of a cut χ: total capacity of forward edges
- Example:
  - $c(\chi) = 24$
  - $f(\chi) = 8$





Flow and Cut

Lemma:

The flow $f(\chi)$ across any cut χ is equal to the flow value $|f|$

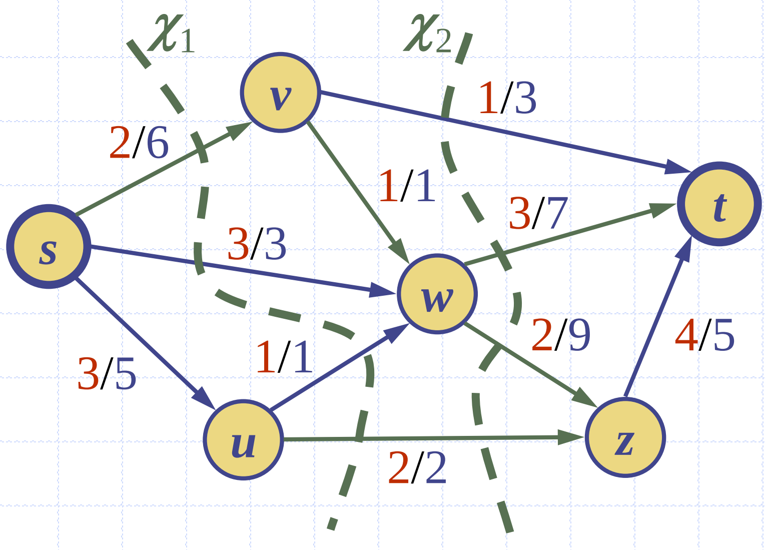
Lemma:

The flow $f(\chi)$ across a cut χ is less than or equal to the capacity $c(\chi)$ of the cut

Theorem:

The value of any flow is less than or equal to the capacity of any cut, i.e., for any flow f and any cut χ, we have

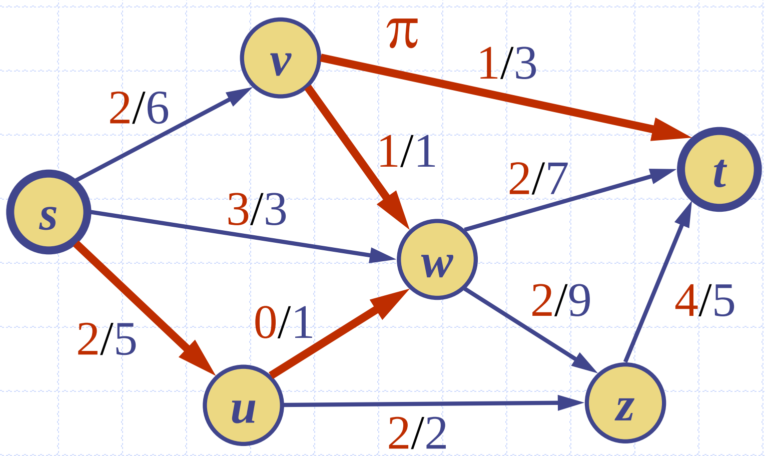
$$|f| \leq c(\chi)$$



$$c(\chi_1) = 12 = 6 + 3 + 1 + 2$$
$$c(\chi_2) = 21 = 3 + 7 + 9 + 2$$
$$|f| = 8$$

Augmenting Path

- Consider a flow f for a network N
- Let e be an edge from u to v:
  - Residual capacity of e from u to v: $\Delta_f(u, v) = c(e) - f(e)$
  - Residual capacity of e from v to u: $\Delta_f(v, u) = f(e)$
- Let π be a path from s to t
  - The residual capacity $\Delta_f(\pi)$ of π is the smallest of the residual capacities of the edges of π in the direction from s to t
- A path π from s to t is an augmenting path if $\Delta_f(\pi) > 0$



$$\Delta_f(s,u) = 3$$
$$\Delta_f(u,w) = 1$$
$$\Delta_f(w,v) = 1$$
$$\Delta_f(v,t) = 2$$
$$\Delta_f(\pi) = 1$$
$$|f| = 7$$

Flow Augmentation

Lemma:

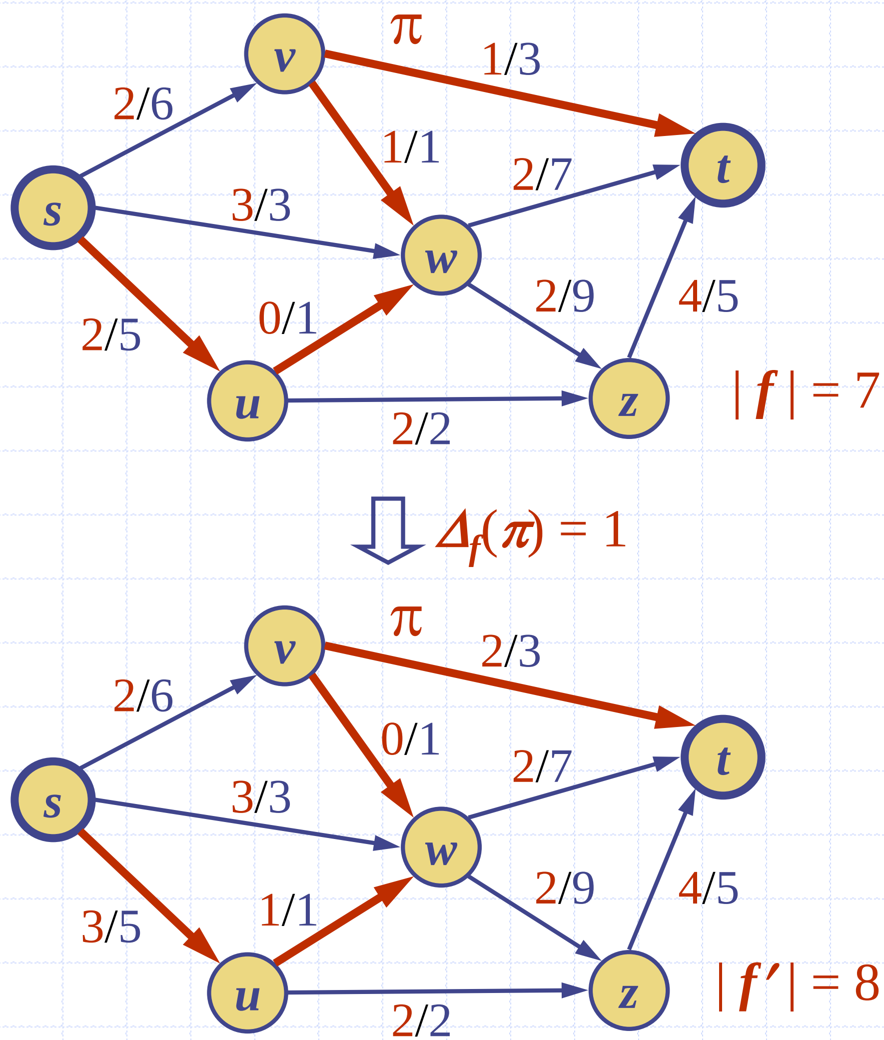
Let π be an augmenting path for flow f in network N. There exists a flow f' for N of value

$$|f'| = |f| + \Delta_f(\pi)$$

Proof:

We compute flow f' by modifying the flow on the edges of π

- Forward edge:
  $$f'(e) = f(e) + \Delta_f(\pi)$$
- Backward edge:
  $$f'(e) = f(e) - \Delta_f(\pi)$$



Ford-Fulkerson's Algorithm

- Initially, $f(e) = 0$ for each edge e
- Repeatedly
  - Search for an augmenting path π
  - Augment by $\Delta_f(\pi)$ the flow along the edges of π
- A specialization of DFS (or BFS) searches for an augmenting path
  - An edge e is traversed from u to v provided $\Delta_f(u, v) > 0$

```
Algorithm FordFulkersonMaxFlow(N)
  for all e ∈ G.edges()
    setFlow(e, 0)
  while G has an augmenting path π
    { compute residual capacity Δ of π }
    Δ ← ∞
    for all edges e ∈ π
      { compute residual capacity δ of e }
      if e is a forward edge of π
        δ ← getCapacity(e) − getFlow(e)
      else { e is a backward edge }
        δ ← getFlow(e)
      if δ < Δ
        Δ ← δ
    { augment flow along π }
    for all edges e ∈ π
      if e is a forward edge of π
        setFlow(e, getFlow(e) + Δ)
      else { e is a backward edge }
        setFlow(e, getFlow(e) − Δ)
```

Max-Flow and Min-Cut

- Termination of Ford-Fulkerson's algorithm
  - There is no augmenting path from s to t with respect to the current flow f
- Define

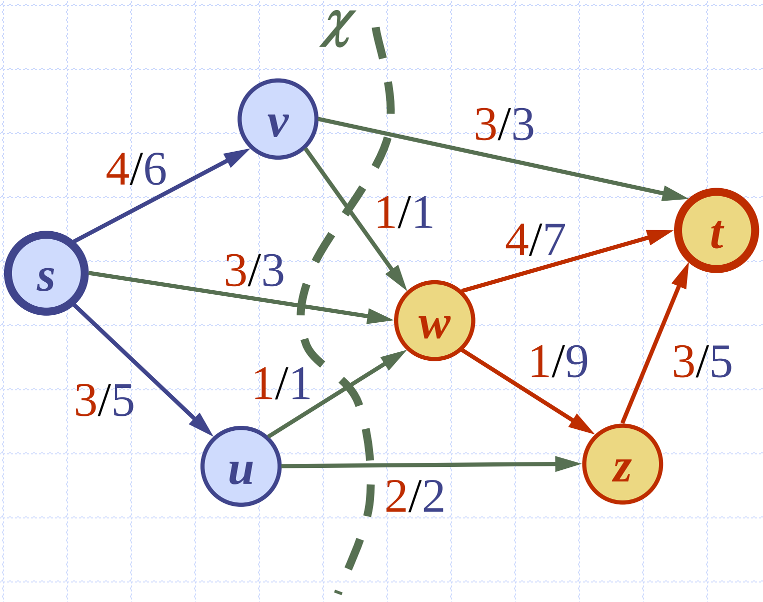
  V_s set of vertices reachable from s by augmenting paths

  V_t set of remaining vertices
- Cut $\chi = (V_s, V_t)$ has capacity

  $c(\chi) = |f|$
  - Forward edge: $f(e) = c(e)$
  - Backward edge: $f(e) = 0$
- Thus, flow f has maximum value and cut χ has minimum capacity

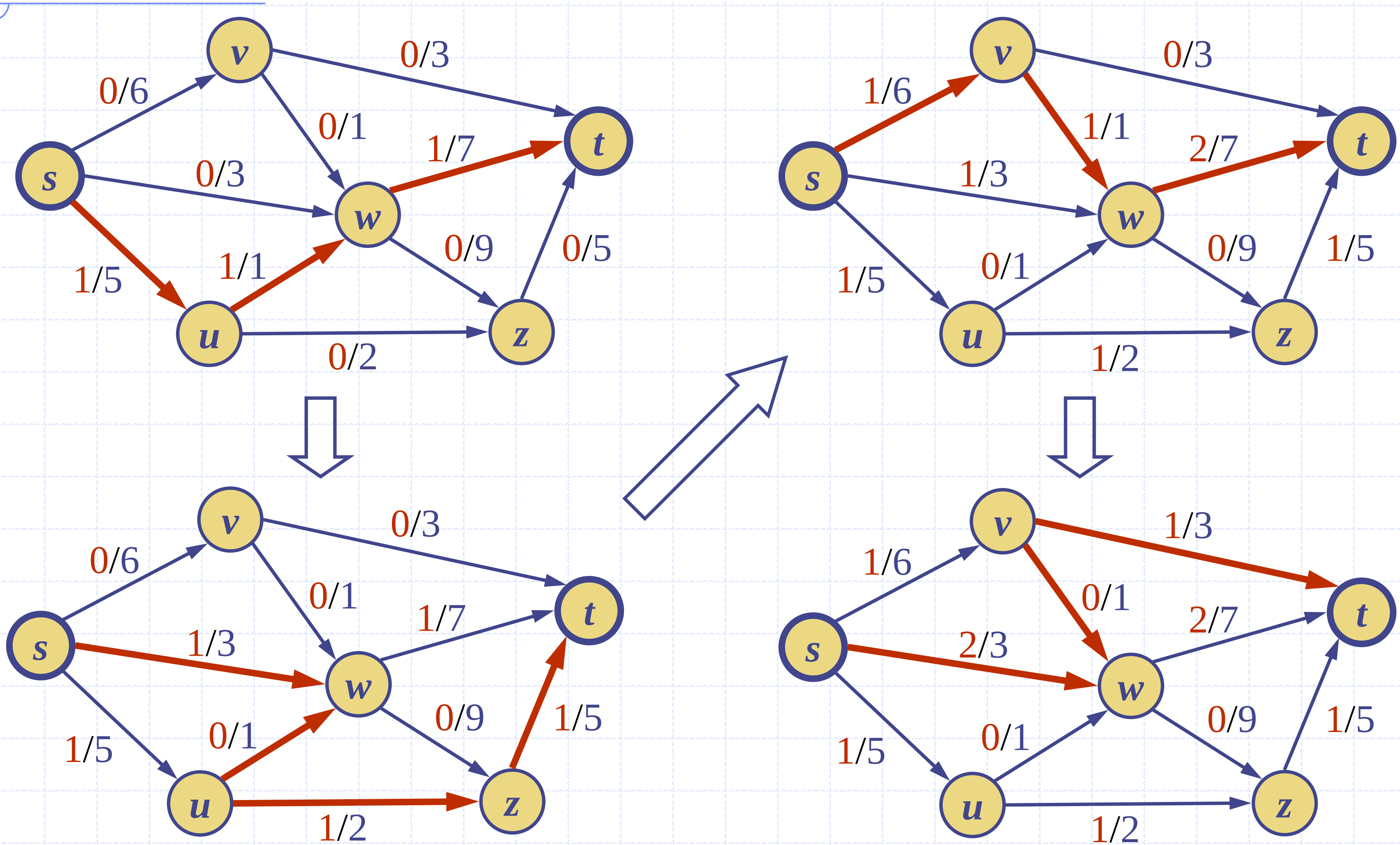
Theorem:

The value of a maximum flow is equal to the capacity of a minimum cut



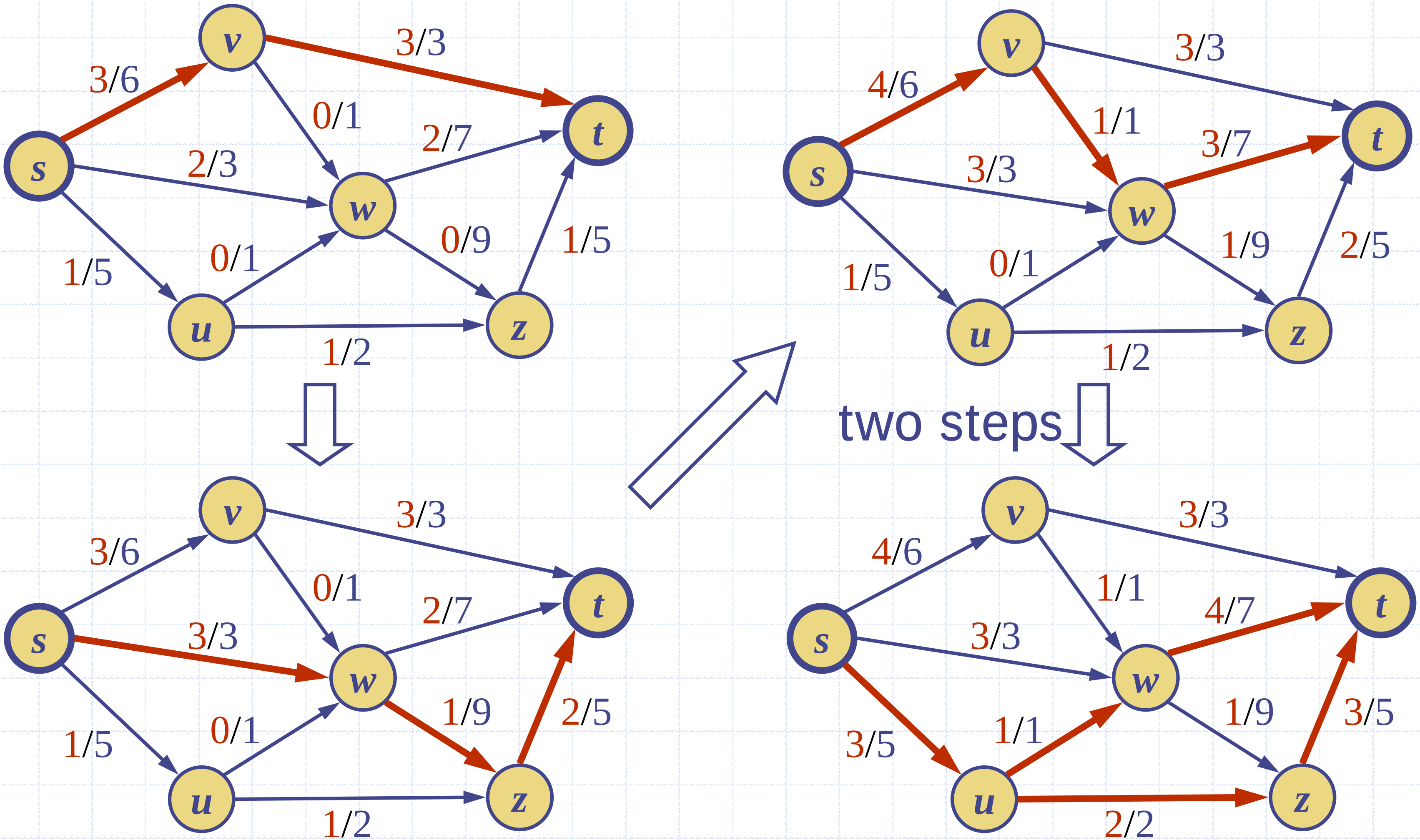
$c(\chi) = |f| = 10$

Example (1)



v
0/6
0/3
0/1
1/7
t
s
0/3
w
1/5
1/1
0/9
0/5
u
z
0/2
v
1/6
0/3
1/1
2/7
t
s
1/3
w
1/5
0/1
0/9
1/5
u
z
1/2
v
0/6
0/3
0/1
1/7
t
s
1/3
w
1/5
0/1
0/9
1/5
u
z
1/2
v
1/6
1/3
0/1
2/7
t
s
2/3
w
1/5
0/1
0/9
1/5
u
z
1/2

Example (2)



v
s
t
w
u
z
3/6
3/3
0/1
2/7
2/3
1/5
0/1
0/9
1/5
1/2
3/6
3/3
0/1
2/7
3/3
1/5
0/1
1/9
2/5
1/2
4/6
3/3
1/1
3/7
3/3
1/5
0/1
1/9
2/5
1/2
two steps
4/6
3/3
1/1
4/7
3/3
3/5
1/1
1/9
3/5
2/2

Analysis

- In the worst case, Ford-Fulkerson's algorithm performs $|f^*|$ flow augmentations, where f^* is a maximum flow
- Example
  - The augmenting paths found alternate between π_1 and π_2
  - The algorithm performs 100 augmentations
- Finding an augmenting path and augmenting the flow takes $O(n + m)$ time
- The running time of Ford-Fulkerson's algorithm is $O(|f^*|(n + m))$