Algorithm Theory

Chapter 6
Graph Algorithms

Part I:
Maximum Flow: Ford Fulkerson Algorithm

Fabian Kuhn

Graphs

Extremely important concept in computer science

Graph $\boldsymbol{G = (V, E)}$

- V: node (or vertex) set
- $E \subseteq V^2$: edge set
  - undirected graph: we often think of edges as sets of size 2 (e.g., $\{u, v\}$)
  - directed graph (digraph): edges are sometimes also called arcs
  - simple graph: no self-loops, no multiple edges
  - weighted graph: (positive) weight on edges (or nodes)
- (simple) path: sequence $v_0, \dots, v_k$ of nodes such that
  $(v_i, v_{i+1}) \in E$ for all $i \in \{0, \dots, k-1\}$
- …

Many real-world problems can be formulated as optimization problems on graphs.

Graph Optimization: Examples

Minimum spanning tree (MST):

- Compute min. weight spanning tree of a weighted undir. Graph

Shortest paths:

- Compute (length) of shortest paths (single source, all pairs, …)

Traveling salesperson (TSP):

- Compute shortest TSP path/tour in weighted graph

Vertex coloring:

- Color the nodes such that neighbors get different colors
- Goal: minimize the number of colors

Maximum matching:

- Matching: set of pair-wise non-adjacent edges
- Goal: maximize the size of the matching

Network Flow

Flow Network:

- Directed graph $G = (V, E), E \subseteq V^2$
- Each (directed) edge e has a capacity $c_e \geq 0$
  - Amount of flow (traffic) that the edge can carry
- A single source node $s \in V$ and a single sink node $t \in V$
  - Source s has only outgoing edges, sink t has only incoming edges

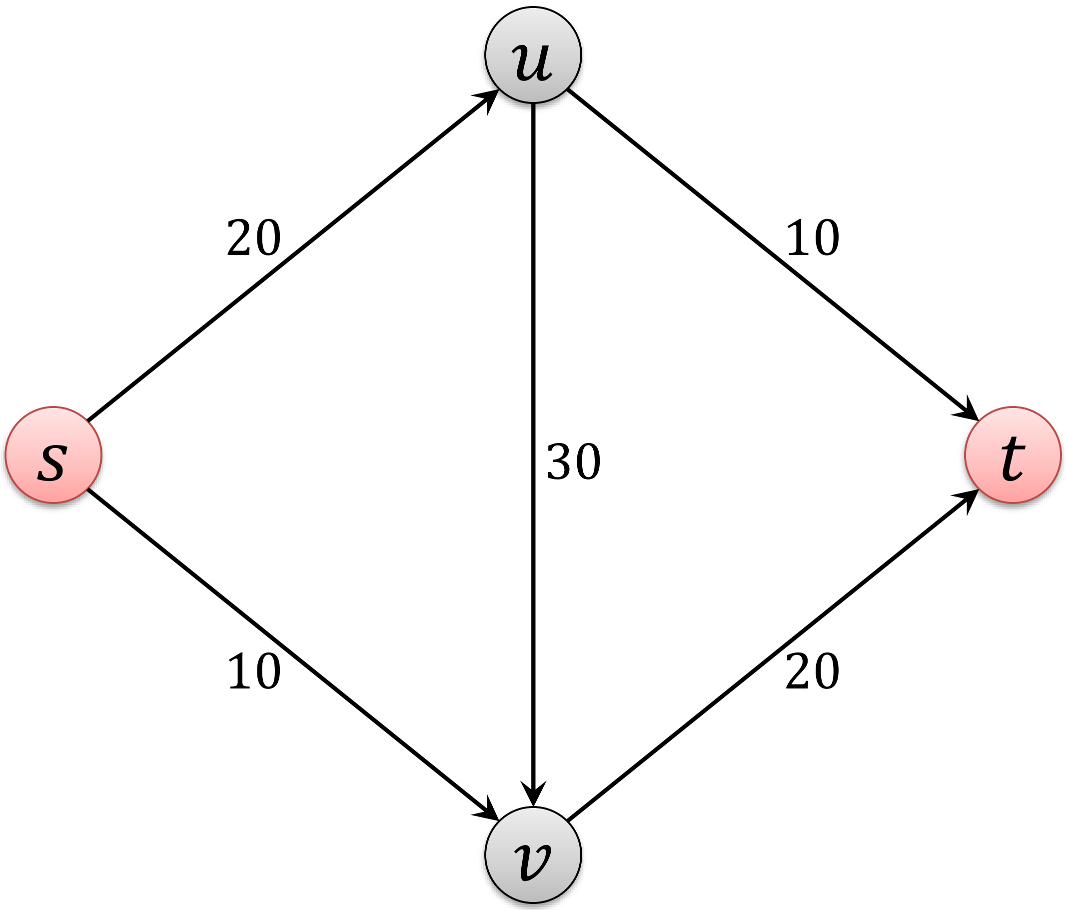
Flow: (informally)

- Traffic from s to t such that each edge carries at most its capacity

Examples:

- Highway system: edges are highways, flow is the traffic
- Computer network: edges are network links, flow is data
- Fluid network: edges are pipes that carry liquid

Example: Flow Network



Network Flow: Definition

Flow: function $\boldsymbol{f}: \boldsymbol{E} \to \mathbb{R}_{\geq 0}$

- $f(e)$ is the amount of flow carried by edge e

Capacity Constraints:

- For each edge $e \in E$, $f(e) \leq c_e$

Flow Conservation:

- For each node $v \in V \setminus \{s, t\}$,

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$

Flow Value:

$$|f| := \sum_{e \text{ out of } s} f\big((s, u)\big) = \sum_{e \text{ into } t} f\big((v, t)\big)$$

Notation

We define:

$$f^{\text{in}}(v) := \sum_{e \text{ into } v} f(e), \qquad f^{\text{out}}(v) := \sum_{e \text{ out of } v} f(e)$$

For a set $A \subseteq V$:

$$f^{\text{in}}(A) := \sum_{e \text{ into } A} f(e), \qquad f^{\text{out}}(A) := \sum_{e \text{ out of } S} f(e)$$

Flow conservation: $\forall v \in V \setminus \{s, t\}$: $f^{\text{in}}(v) = f^{\text{out}}(v)$

Flow value: $|f| = f^{\text{out}}(s) = f^{\text{in}}(t)$

For simplicity: Assume that all capacities are positive integers

The Maximum-Flow Problem

Maximum Flow:

Given a flow network, find a flow of maximum possible value

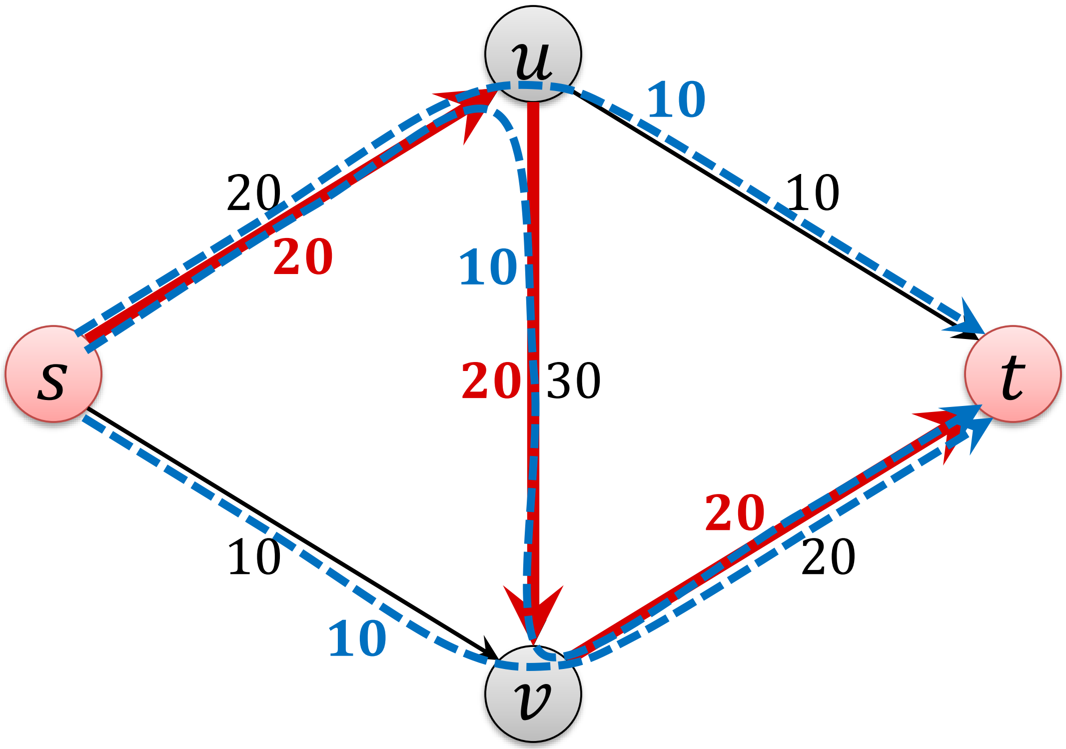
- Classic graph optimization problem
- Many applications (also beyond the obvious ones)
- Requires new algorithmic techniques

Maximum Flow: Greedy?

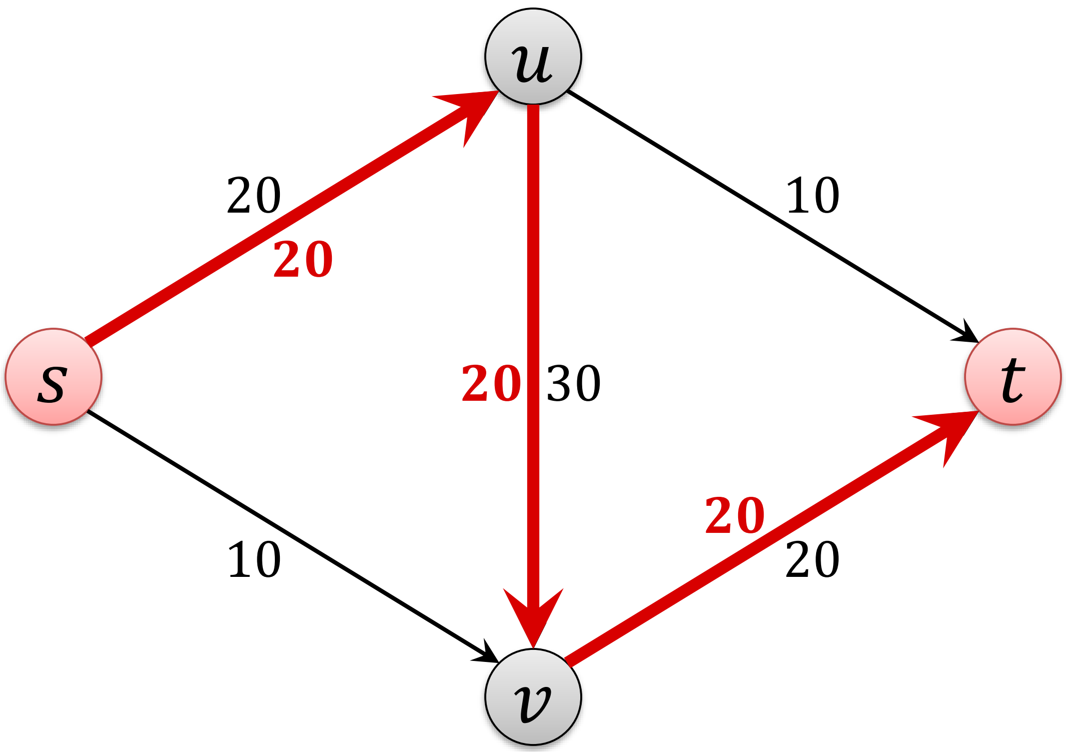
Does greedy work?

A natural greedy algorithm:

- As long as possible, find an s-t-path with free capacity and add as much flow as possible to the path



Improving the Greedy Solution



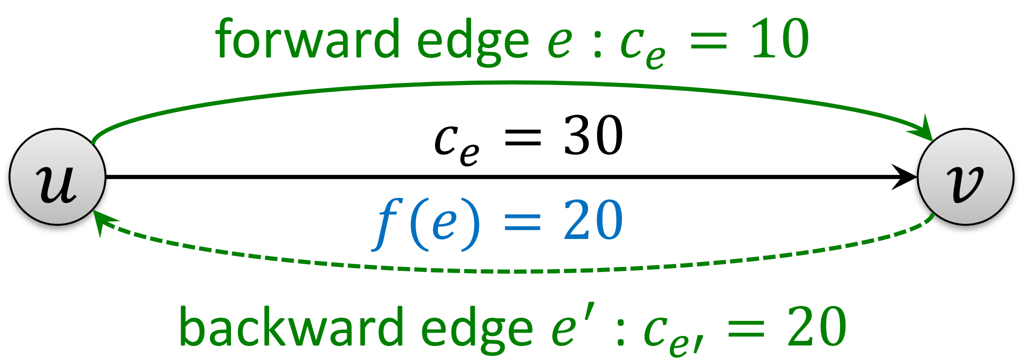
- Try to push 10 units of flow on edge (s, v)
- Too much incoming flow at v: reduce flow on edge (u, v)
- Add that flow on edge (u, t)

Residual Graph

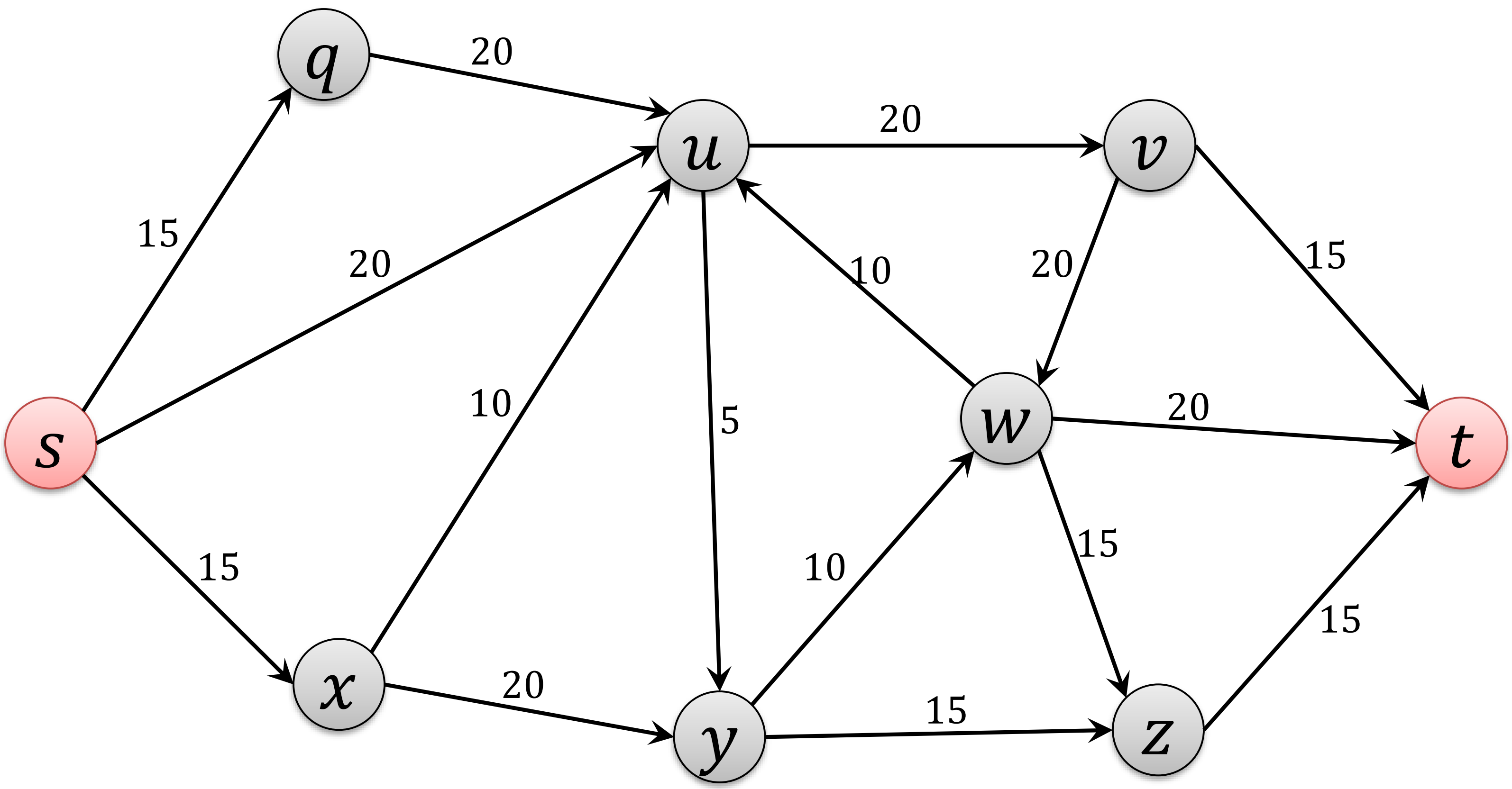
Given a flow network $G = (V, E)$ with capacities c_e (for $e \in E$)

For a flow f on G, define directed graph $G_f = (V_f, E_f)$ as follows:

- Node set $V_f = V$
- For each edge $e = (u, v)$ in E, there are two edges in E_f:
  – forward edge $e = (u, v)$ with residual capacity $c_e - f(e)$
  – backward edge $e' = (v, u)$ with residual capacity $f(e)$

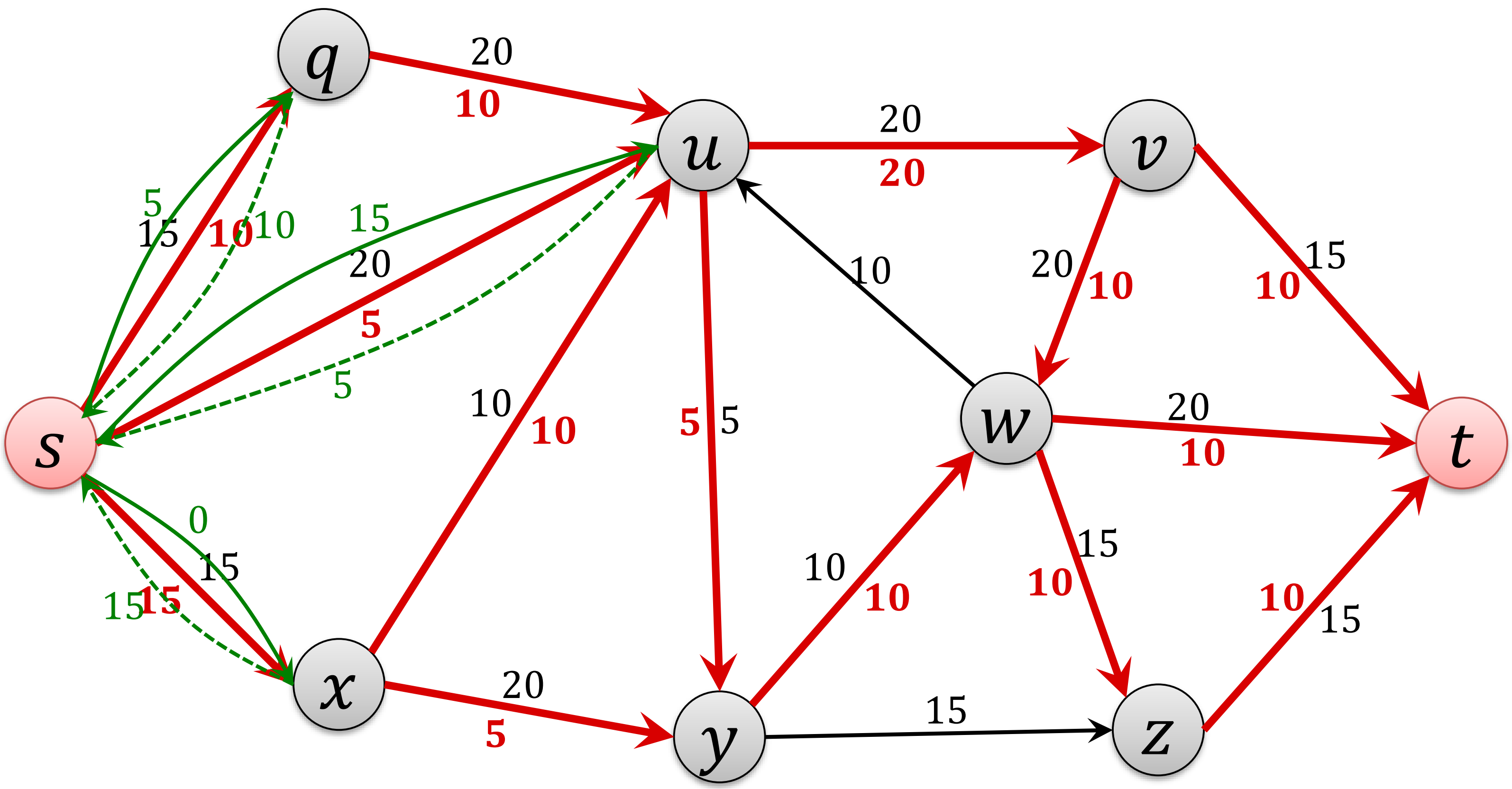


Residual Graph: Example



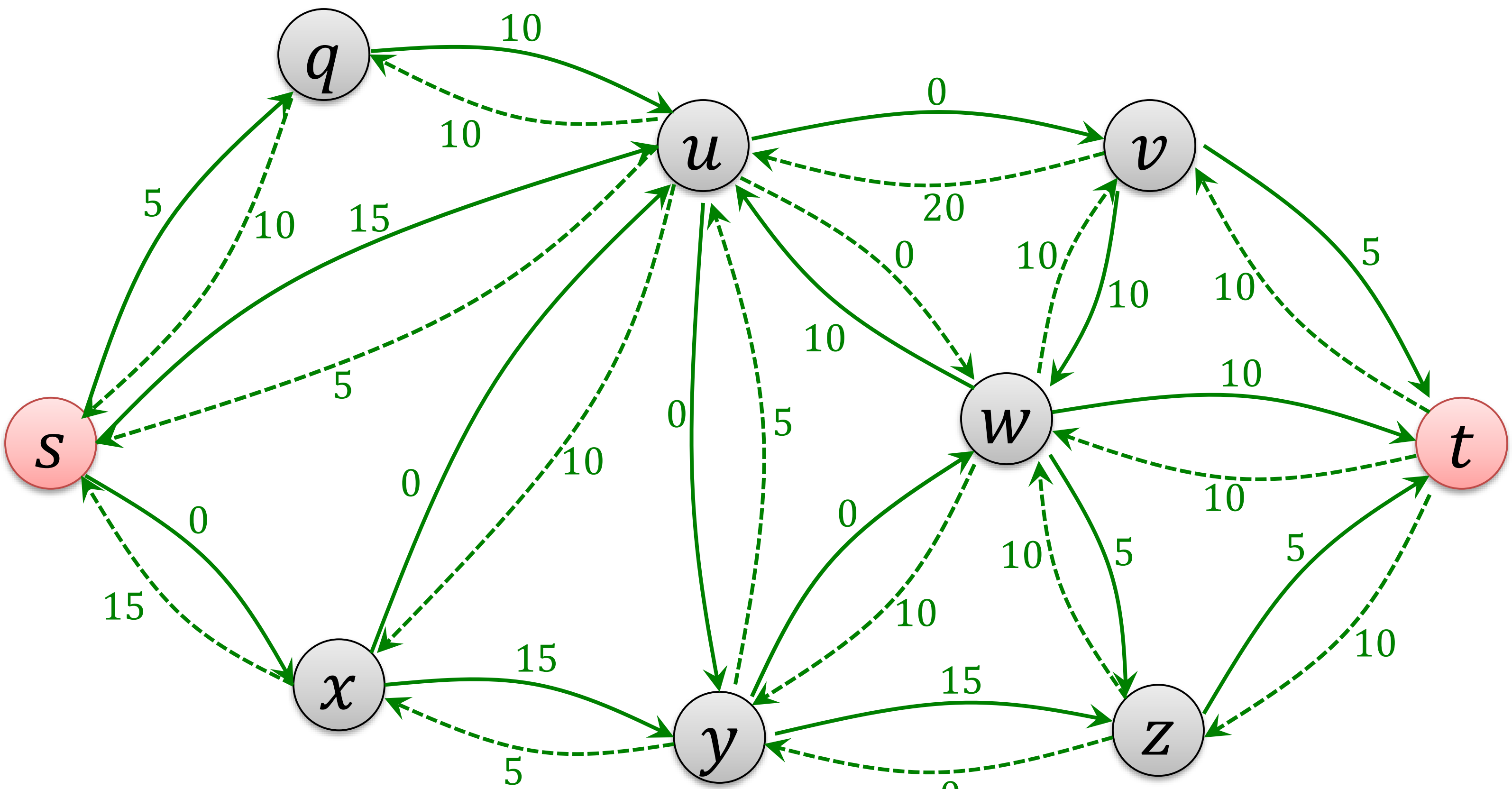
Residual Graph: Example

Flow f



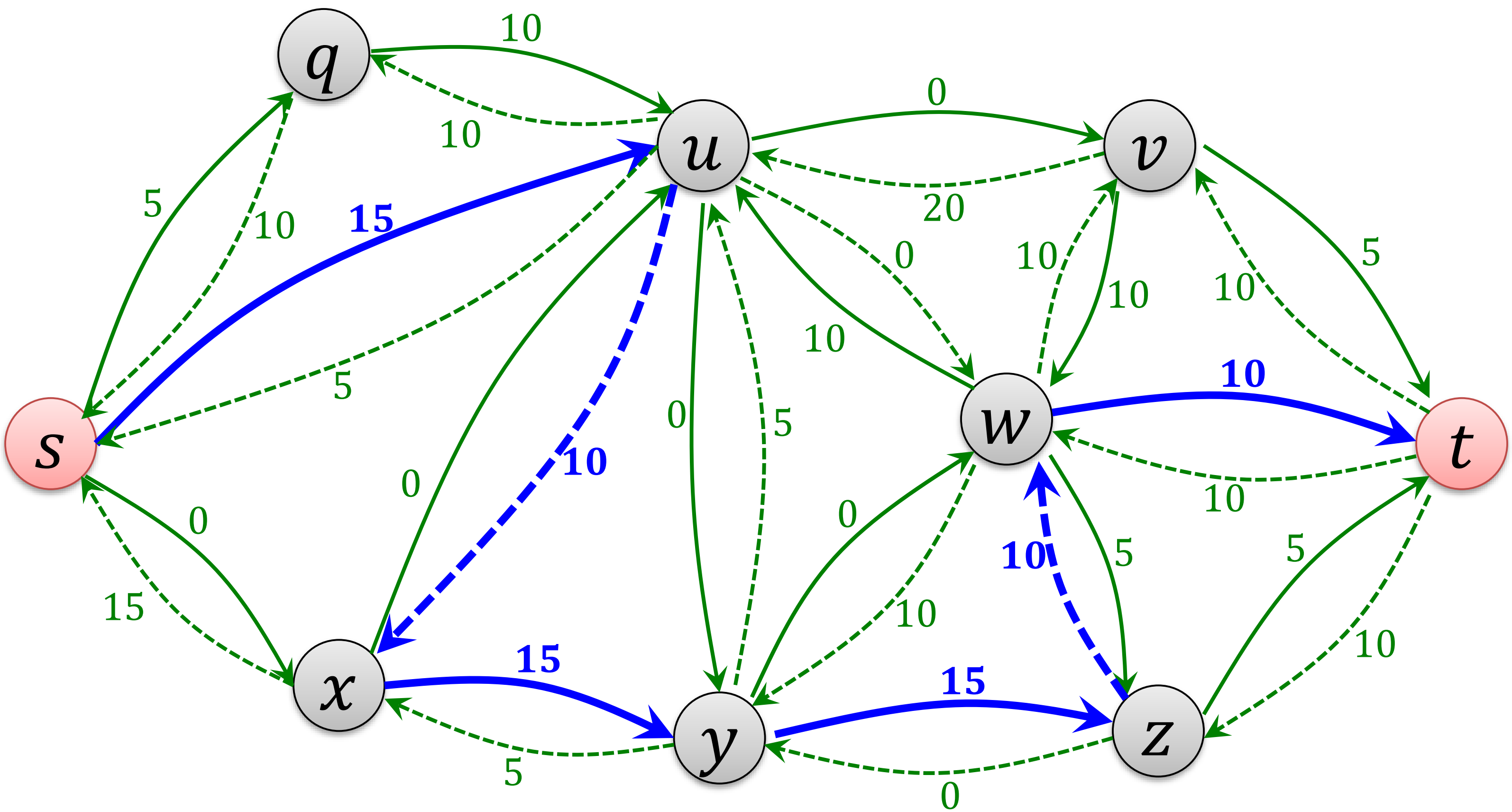
Residual Graph: Example

Residual Graph G_f



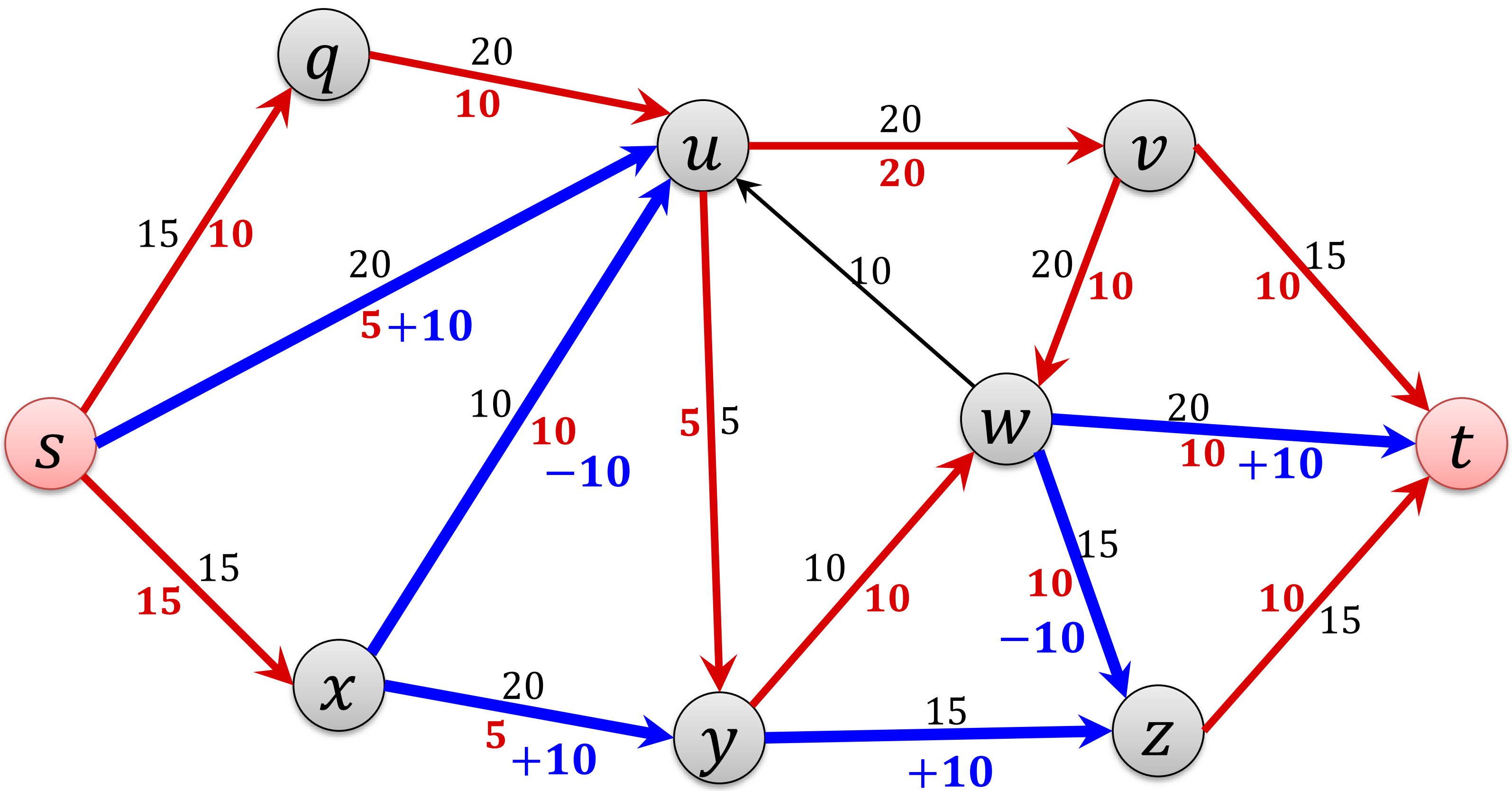
Augmenting Path

Residual Graph G_f



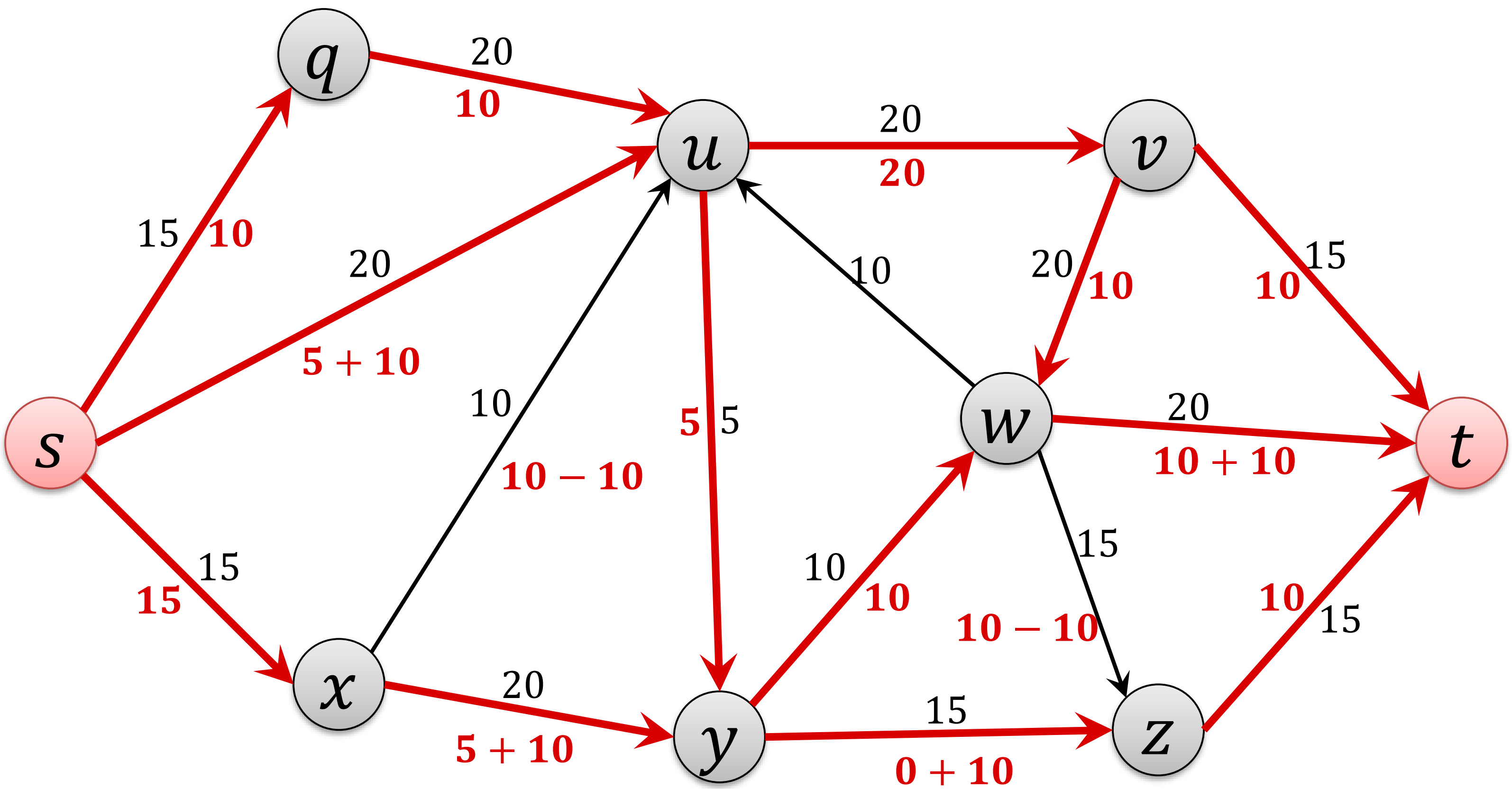
Augmenting Path

Augmenting Path



Augmenting Path

New Flow



Augmenting Path

Definition:
An augmenting path P is a (simple) s-t-path on the residual graph G_f on which each edge has residual capacity > 0.

$\text{bottleneck}(P, f)$: minimum residual capacity on any edge of the augmenting path P

Augment flow f to get flow f':

- For every forward edge (u, v) on P:

$$\boldsymbol{f'\big((u,v)\big) := f\big((u,v)\big) + \textbf{bottleneck}(P,f)}$$

- For every backward edge (u, v) on P:

$$\boldsymbol{f'\big((v,u)\big) := f\big((v,u)\big) - \textbf{bottleneck}(P,f)}$$

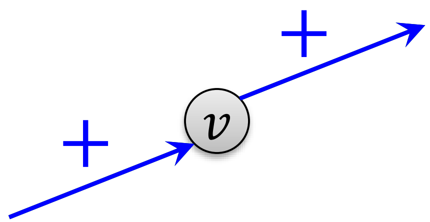
Augmented Flow

Lemma: Given a flow f and an augmenting path P, the resulting augmented flow f' is legal and its value is

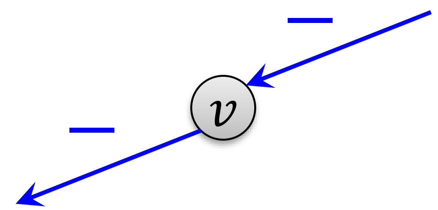
$$|\boldsymbol{f'}| = |\boldsymbol{f}| + \textbf{bottleneck}(\boldsymbol{P}, \boldsymbol{f}).$$

Proof:

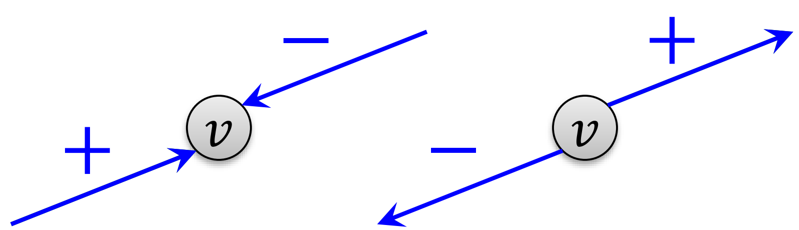
2 forward edges



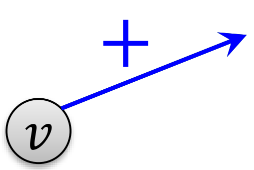
2 backward edges



forward & backward edge



flow value increases



Ford-Fulkerson Algorithm

- Improve flow using an augmenting path as long as possible:

1. Initially, $f(e) = 0$ for all edges $e \in E$, $G_f = G$
2. **while** there is an augmenting s-t-path P in G_f **do**
3. Let P be an augmenting s-t-path in G_f;
4. $f' \coloneqq \text{augment}(f, P)$;
5. update f to be f';
6. update the residual graph G_f
7. **end**;