Internet Routing Example

Acme Routing Company wants to route traffic over the internet from San Fransisco to New York. It owns some wires that go between San Francisco, Houston, Chicago and New York. The table below describes how many kilobytes can be routed on each wire in a second. Figure out a set of routes that maximizes the amount of traffic that goes from San Francisco to New York.

Cities	Maximum number of kbytes per second
S.F. - Chicago	3
S.F. - Houston	6
Houston - Chicago	2
Chicago - New York	7
Houston - New York	5

One commodity, one source, one sink

Maximum Flows

- A **flow network** $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative **capacity** .

- If $(u, v) \notin E$, we assume that $c(u, v) = 0$.

- We distinguish two vertices in a flow network: a **source** s and a **sink** t .

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

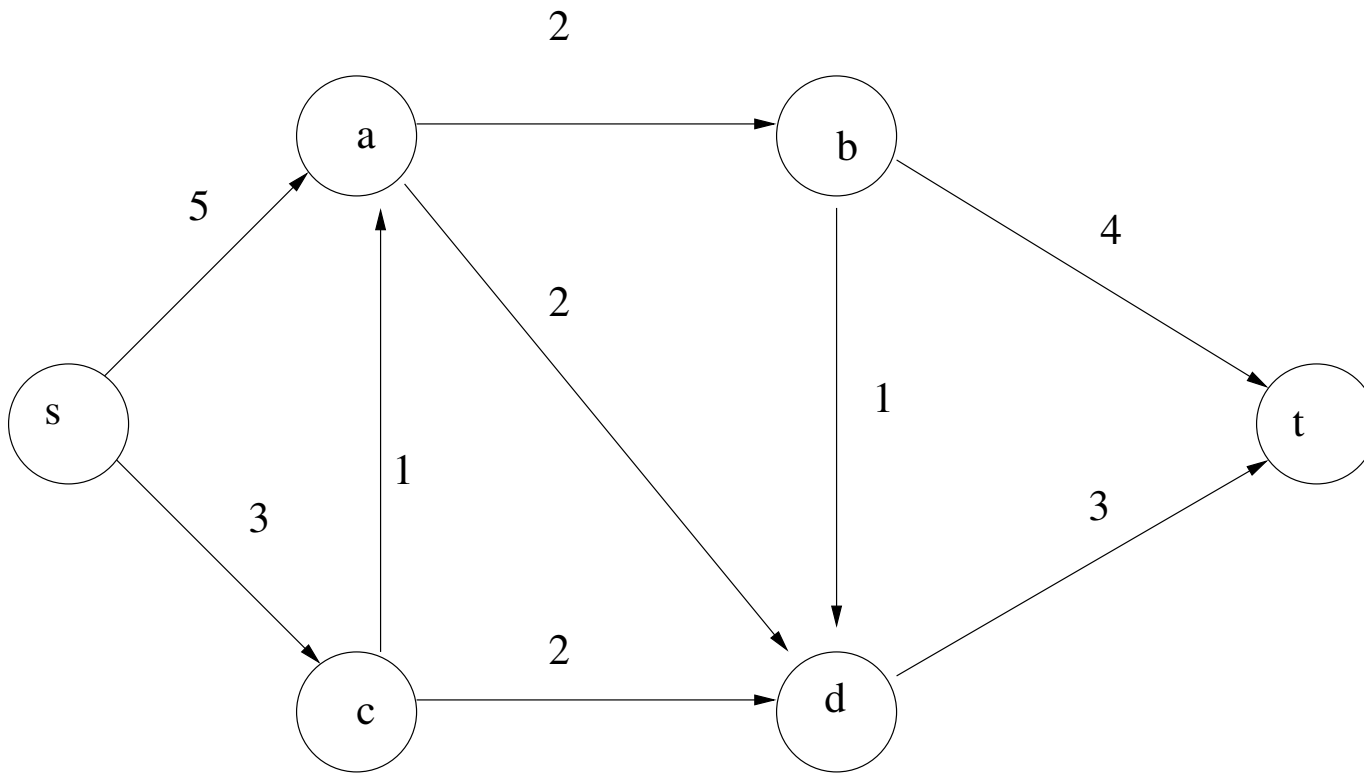Capacity constraint: For all $u, v \in V$, we require $0 \le f(u, v) \le c(u, v)$.

Flow conservation: For all $u \in V - \{s, t\}$, we require

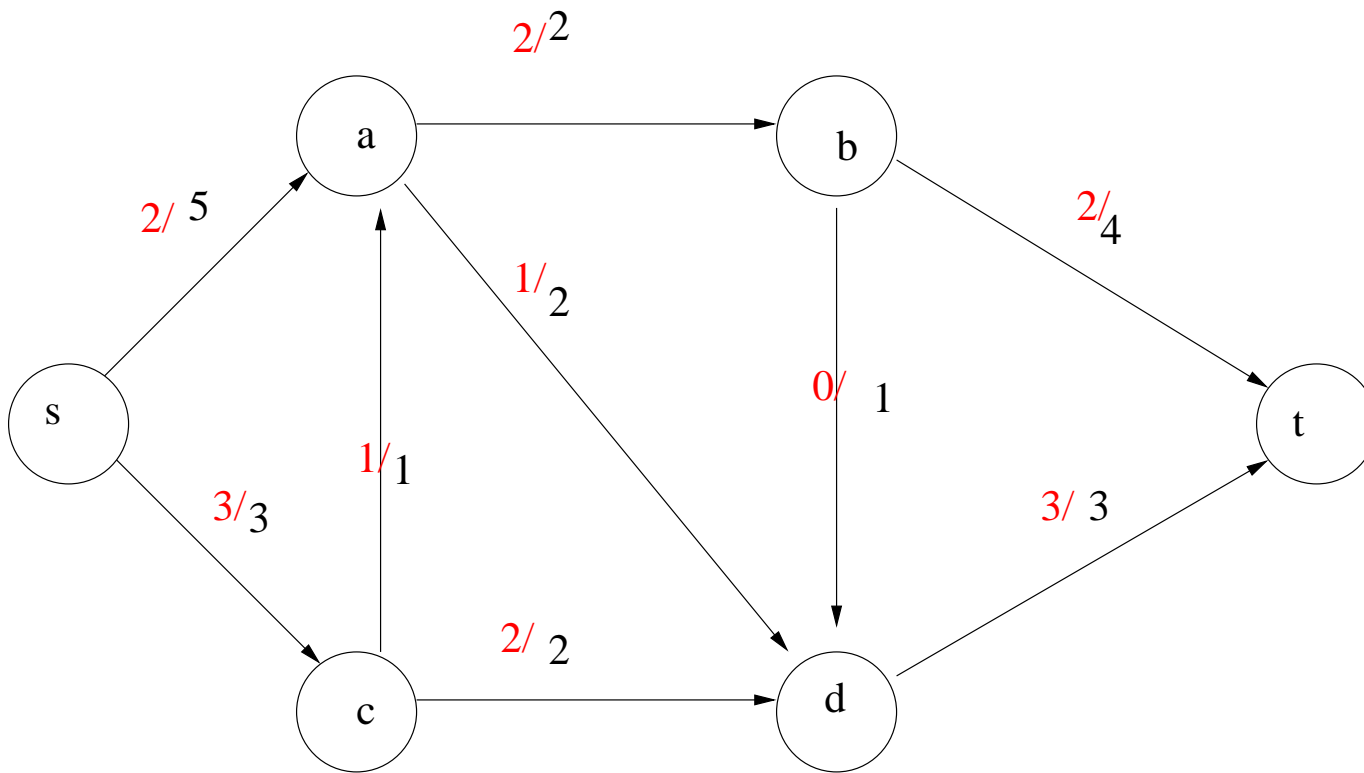$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v) \ .$$

The **value** of a flow f is defined as

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) \ , \tag{1}$$

Example

Solutions

Algorithm: Ford Fulkerson

Greedily send flow from source to sink.

Ford-Fulkerson-Method (G, s, t)

1   initialize flow f to 0
2   while there exists an augmenting path p
3   augment flow f along p
4   return f
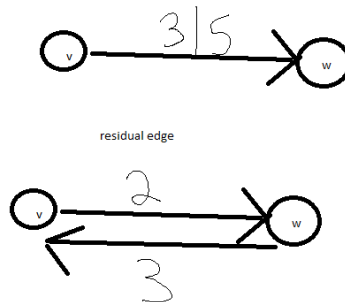
For this to work, we need a notion of a residual graph

Residual Graph

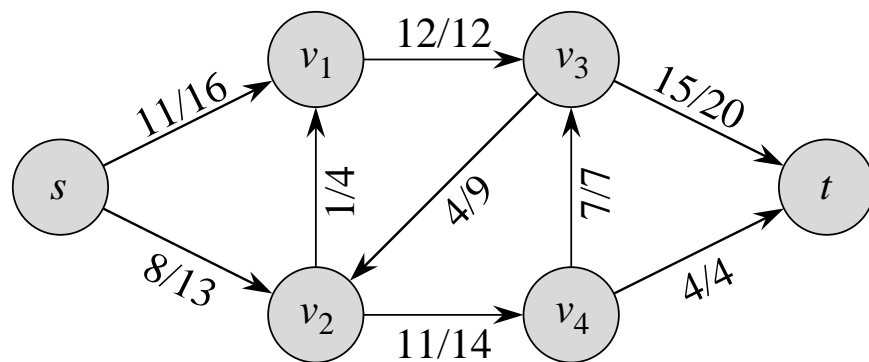The residual graph is the graph of edges on which it is possible to push flow from source to sink.

- The **residual capacity** of (u, v) , is

$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \textbf{if } (u, v) \in E , \\ f(v, u) & \textbf{if } (v, u) \in E , \\ 0 & \textbf{otherwise} . \end{cases} \qquad (2)$$
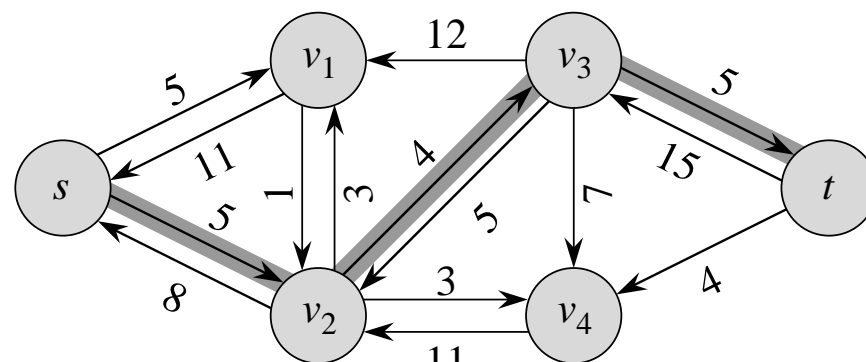
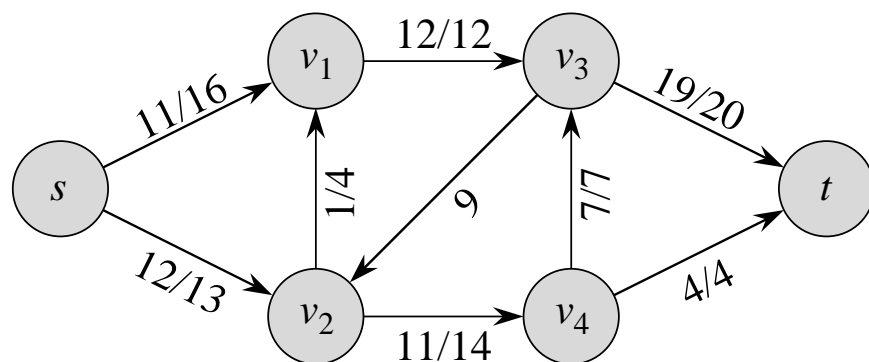- The residual graph G_f is the graph consisting of edges with positive residual capacity
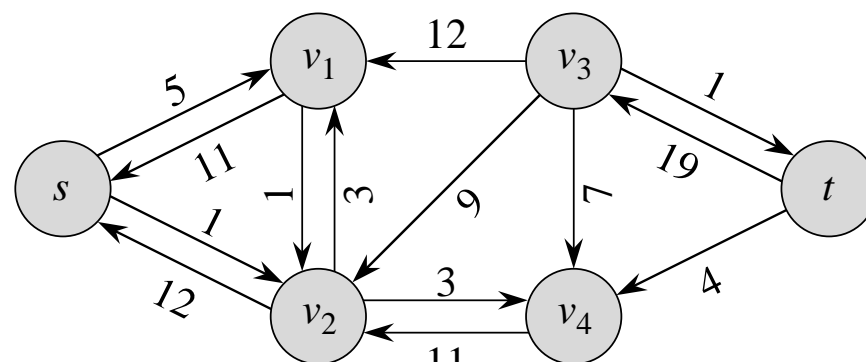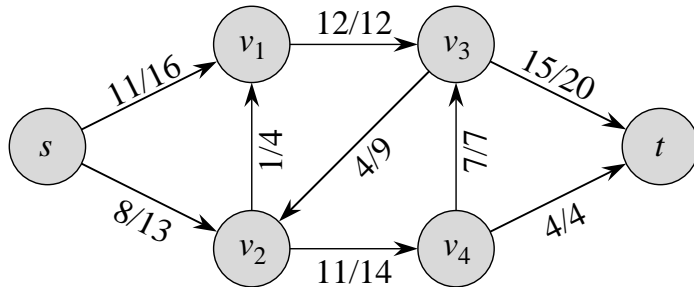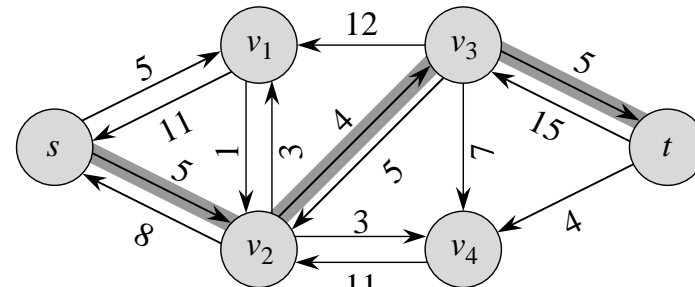
Residual Network
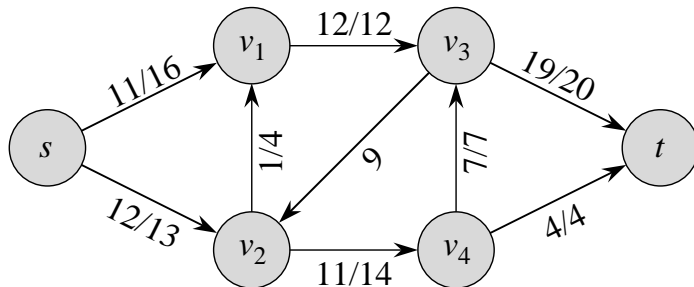

(a)

(b)

(c)

(d)

Updating a Flow

- Send flow along the path defined by the residual graph.

- Amount: minimum of capacity of all residual edges in the augmenting path.

- If a residual edge is a graph edge, then add the flow.

- If a residual edge is a reverse edge, then subtract the flow.

(a)
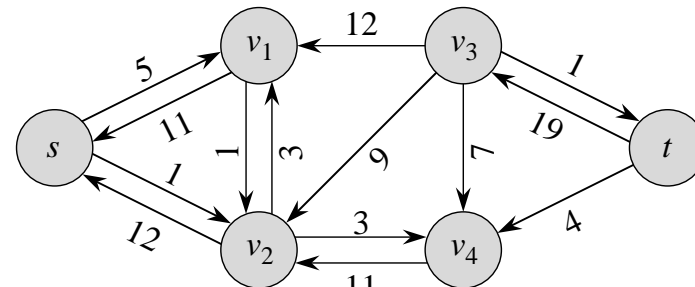
(b)

(c)

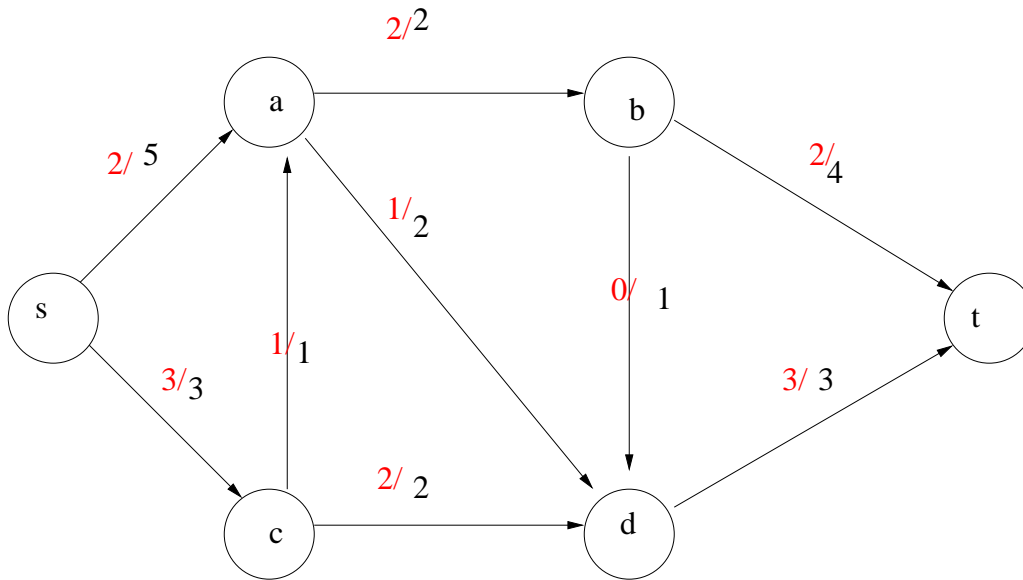(d)

An s - t cut satsfies

- $s \in S$, $t \in T$
- $S \cup T = V$, $S \cap T = \emptyset$
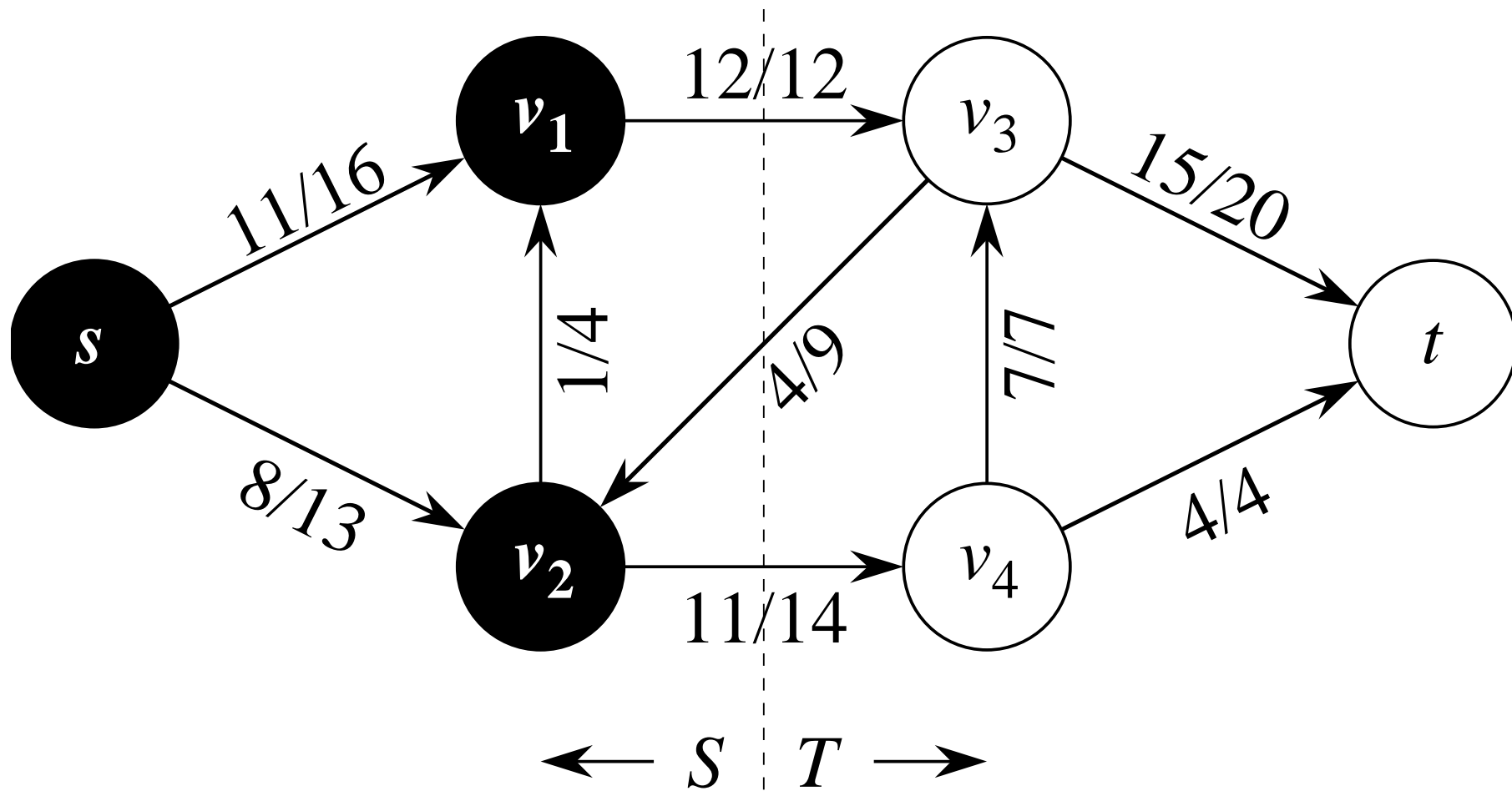

Capacity of a cut (only forward edges)

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

Flow crossing a cut (net flow)

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

Properties of cuts and flows

Capacity of a cut (only forward edges)

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

Flow crossing a cut (net flow)

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

- For all cuts (S,T) and all feasible flows f, $f(S,T) \le c(S,T)$ (weak duality).

- For all pairs of cuts (S_1, T_1) and (S_2, T_2), and all feasible flows f, $f(S_1, T_1) = f(S_2, T_2)$.

Examples of cuts

Max-flow min-cut theorem

If f is a flow in a flow network $G = (V, E)$ with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.

2. The residual network G_f contains no augmenting paths.

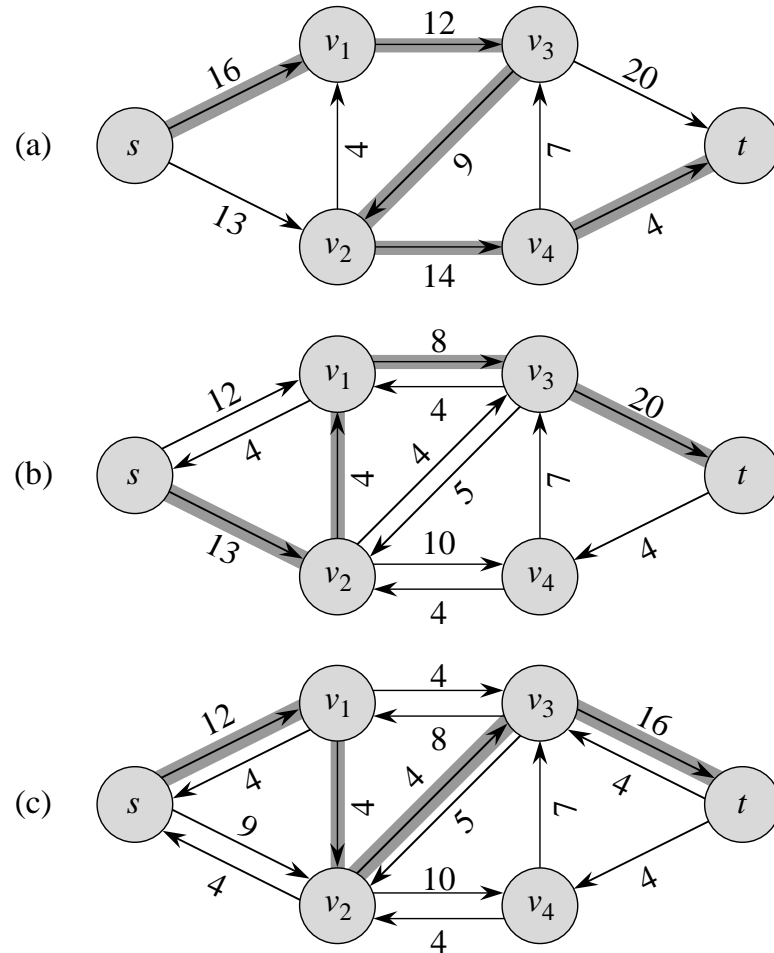3. $|f| = c(S, T)$ for some cut (S, T) of G.

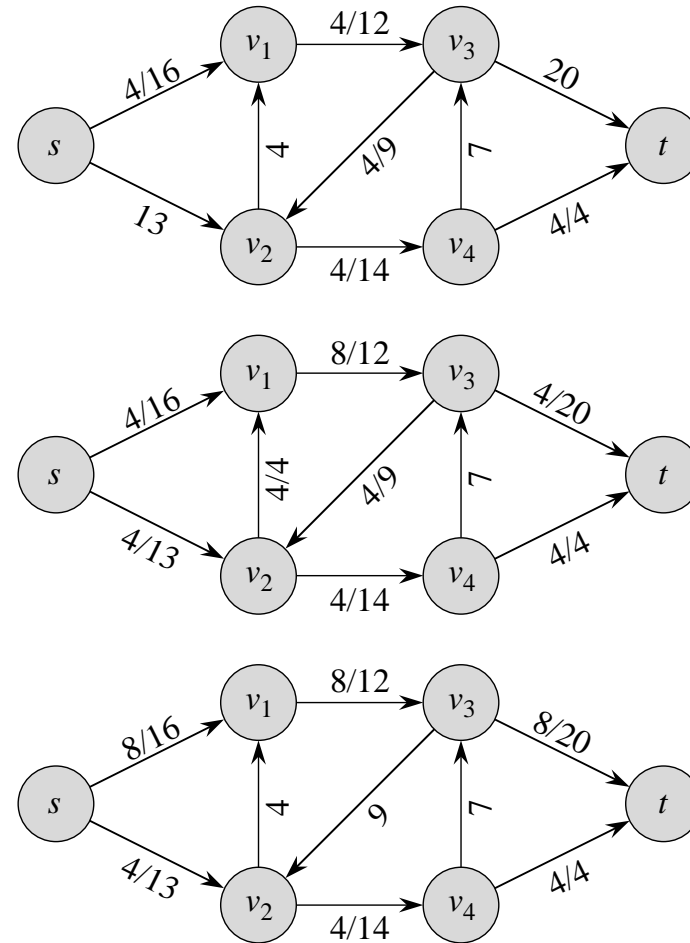# Proof

Ford Fulkerson expanded

$Ford - Fulkerson(G, s, t)$

1   **for each edge** $(u, v) \in E(G)$
2 $f(u, v) = 0$
3   **while there exists a path** p **from** s **to** t **in the residual network** G_f
4 $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
5   **for each edge** (u, v) **in** p
6   **if** $(u, v) \in E$
7 $f(u, v) = f(u, v) + c_f(p)$
8   **else** $f(v, u) = f(v, u) - c_f(p)$
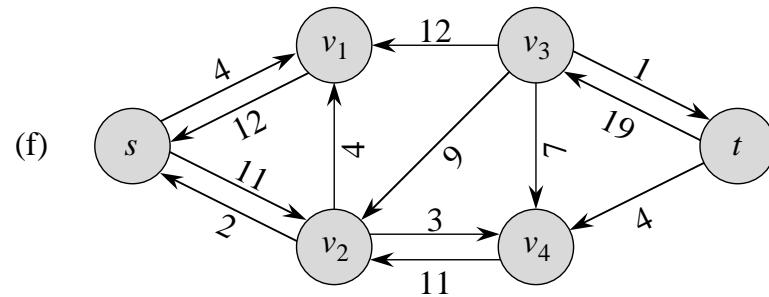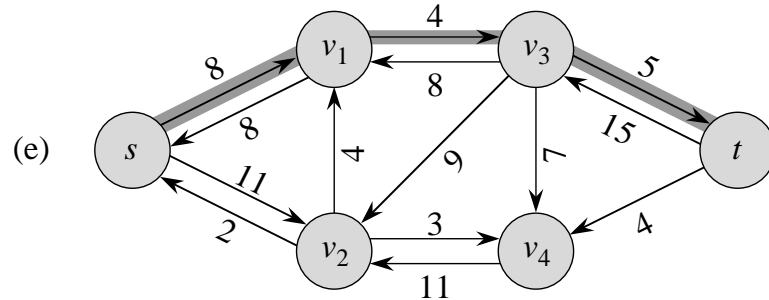
Algorithm

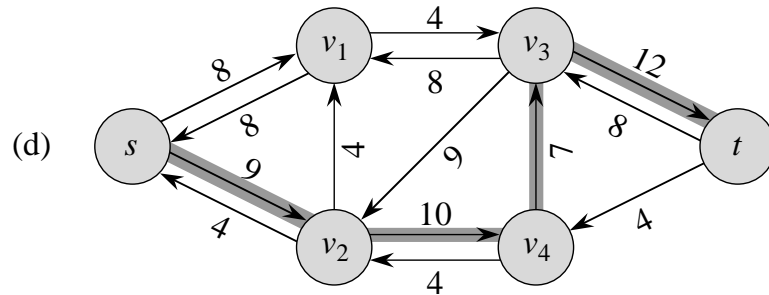**Residual graph (Capacities)**
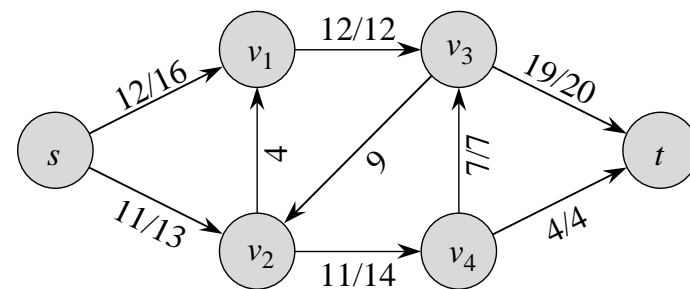
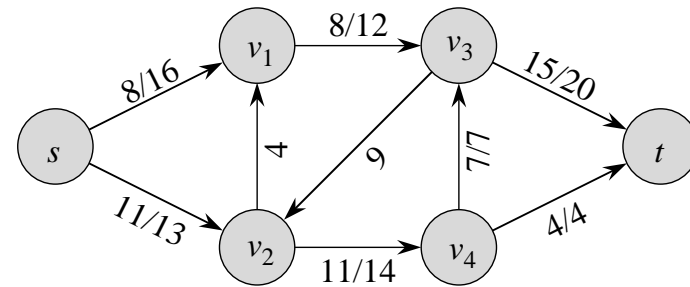**Graph (Flow/Capacity)**

Algorithm continued

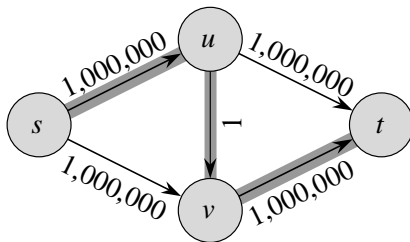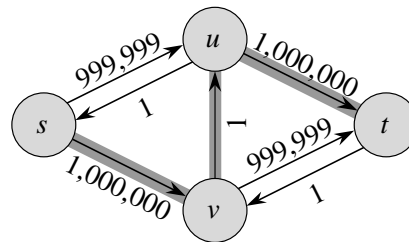**Residual graph (Capacities)**   **Graph (Flow/Capacity)**
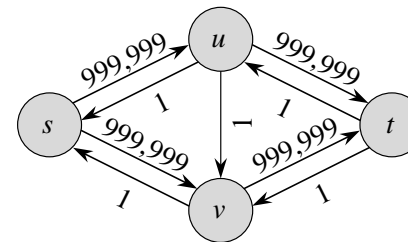
Analysis

- 1 iteration of FF takes $O(E + V)$ time (breadth-first search plus book-keeping).

- Each iteration sends at least one unit of flow.

- Total time $O(f^*E)$.

- This algorithm is only psuedo-polynomial.

(a)   (b)   (c)