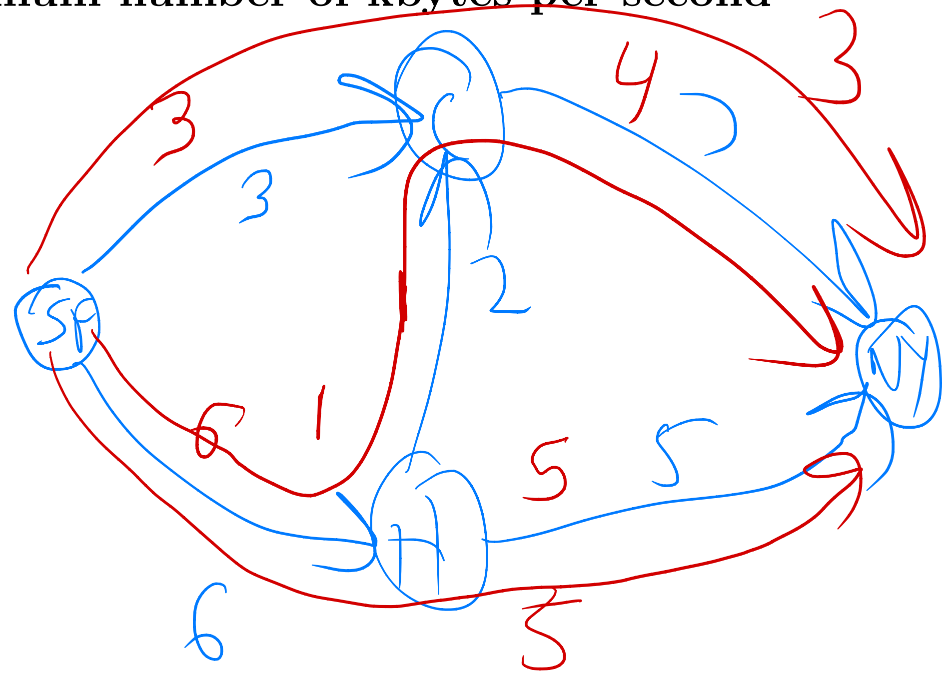
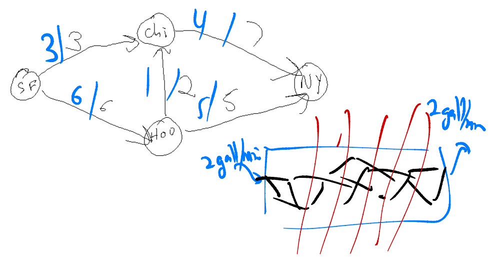
Internet Routing Example

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

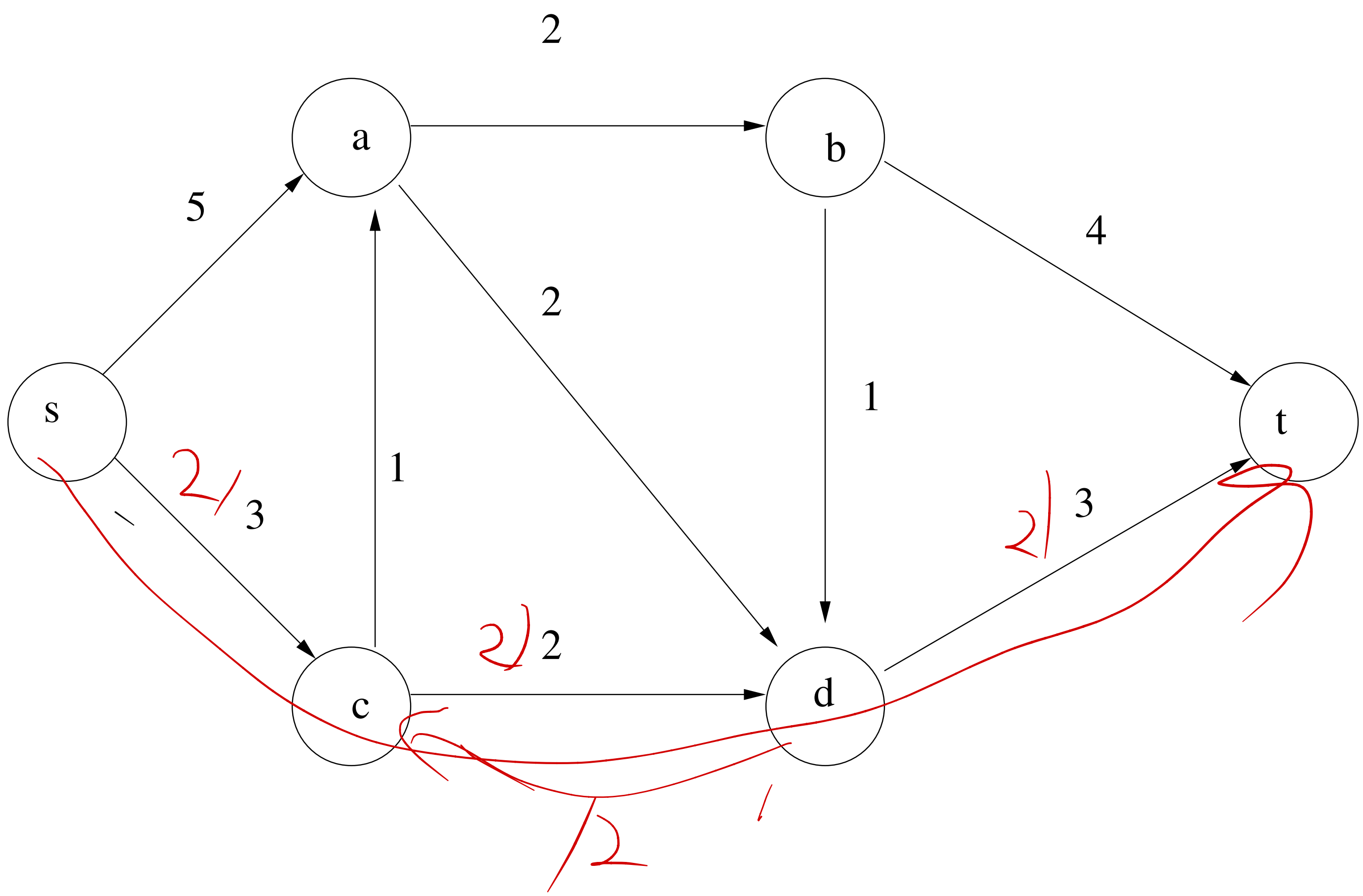
$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v) .$$

flow in = flow out

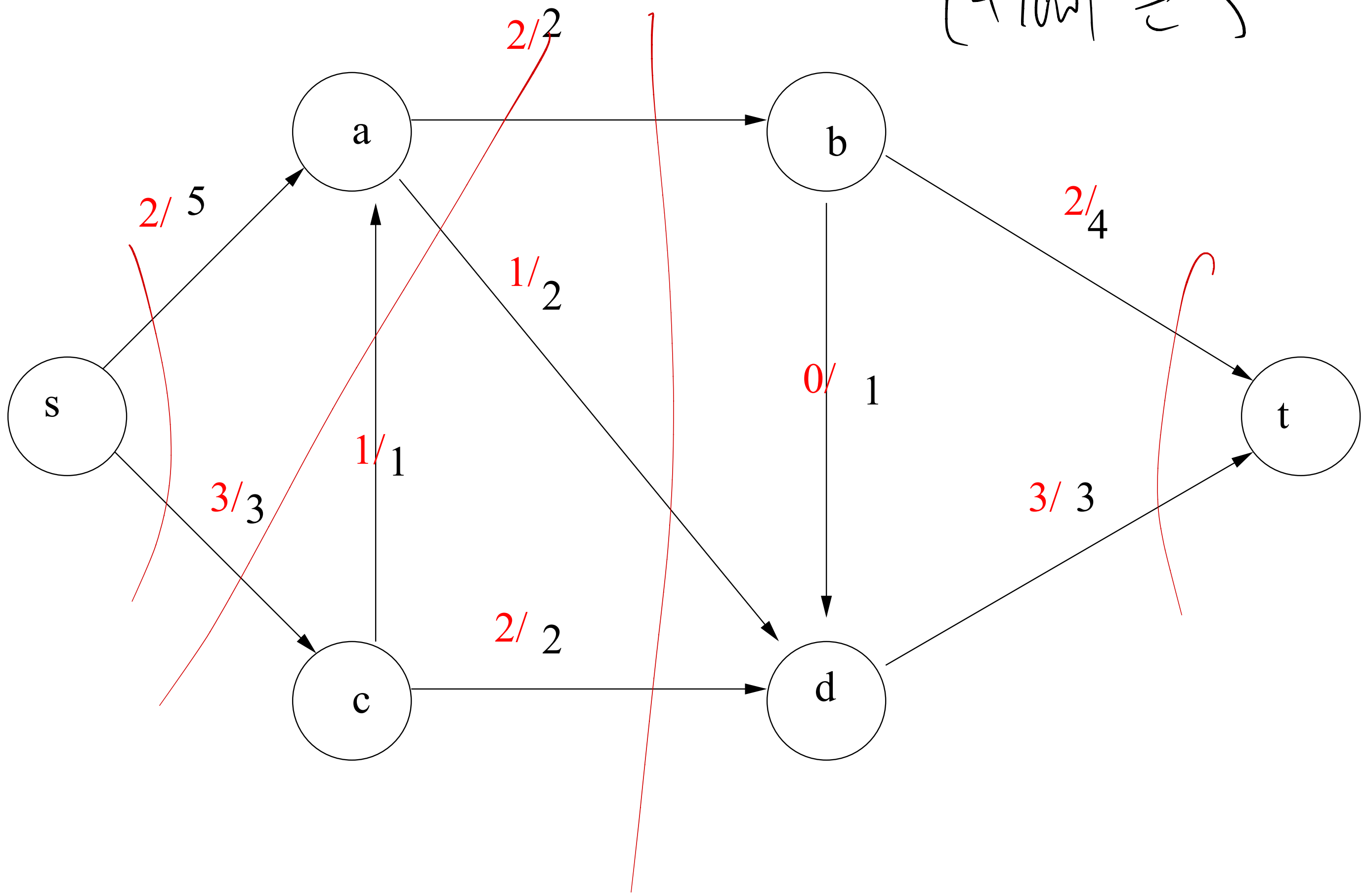
The **value** of a flow f is defined as

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

Example

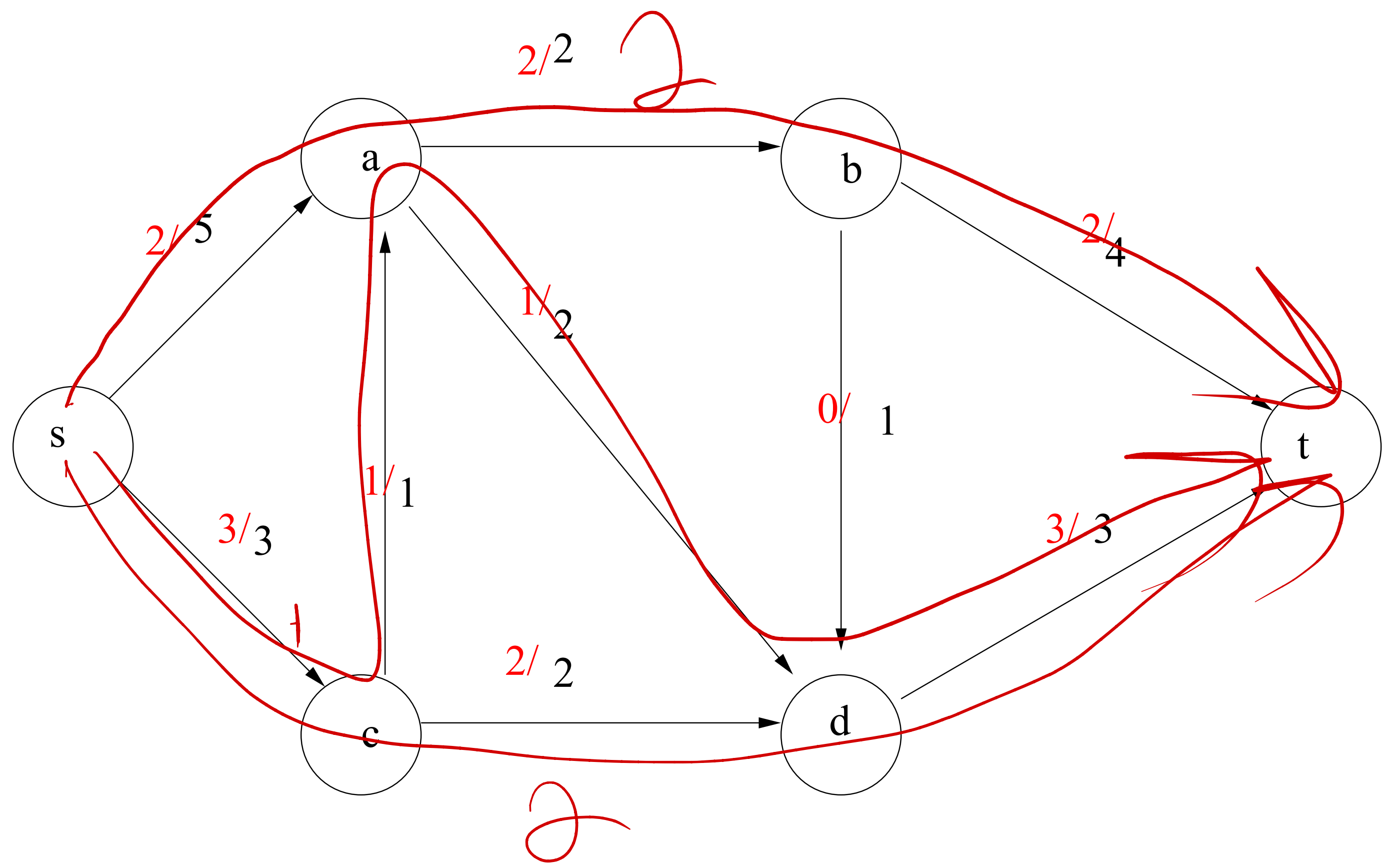


Solutions



|flow| = 5

Solutions



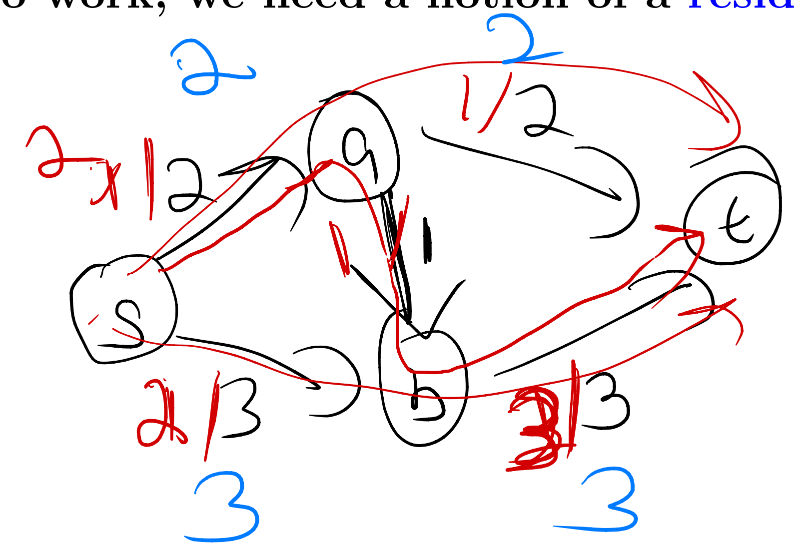
Algorithm: Ford Fulkerson

Greedily send flow from source to sink.

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

```
initialize flow f to 0
while there exists an augmenting path p
      augment flow f along p
return f
```

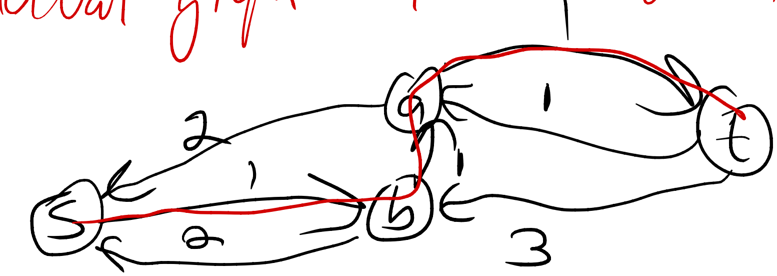
For this to work, we need a notion of a **residual graph**



flow = 4

|flow| = 5

Residual graph w.r.t. red flow



send 1 unit of flow on this path

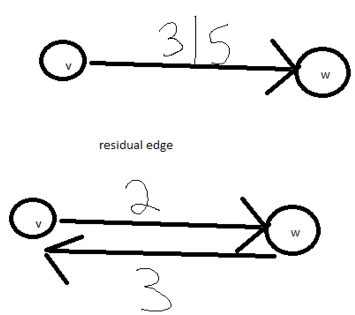
Residual Graph

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

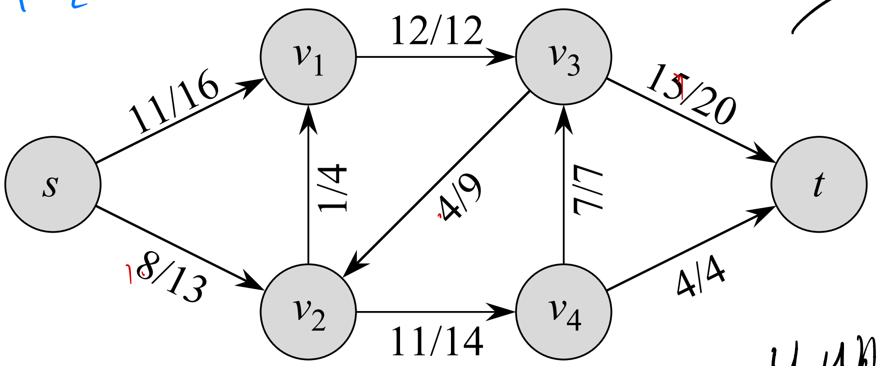
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \textbf{if } (u,v) \in E \ , \\ f(v,u) & \textbf{if } (v,u) \in E \ , \\ 0 & \textbf{otherwise} \ . \end{cases} \tag{2}$$

- The residual graph G_f is the graph consisting of edges with positive residual capacity



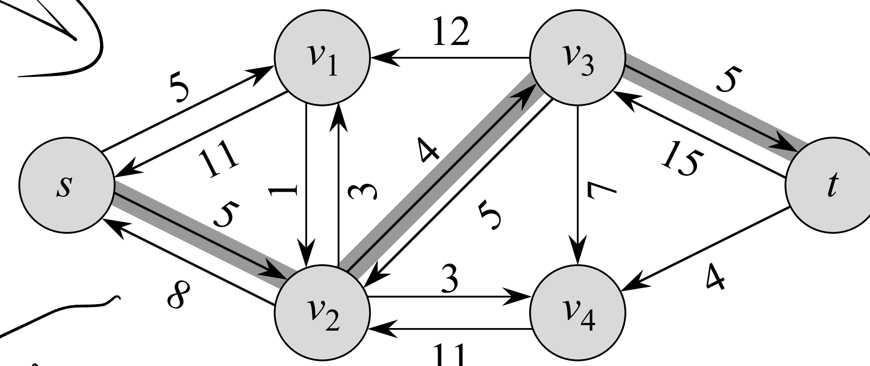
Residual Network

f/c



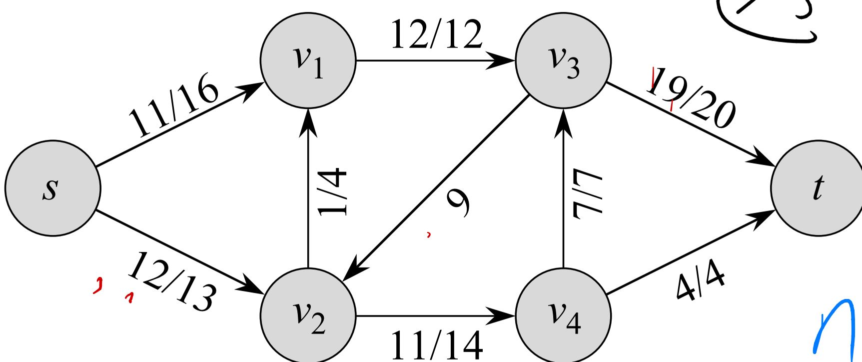
(a)

residual graph

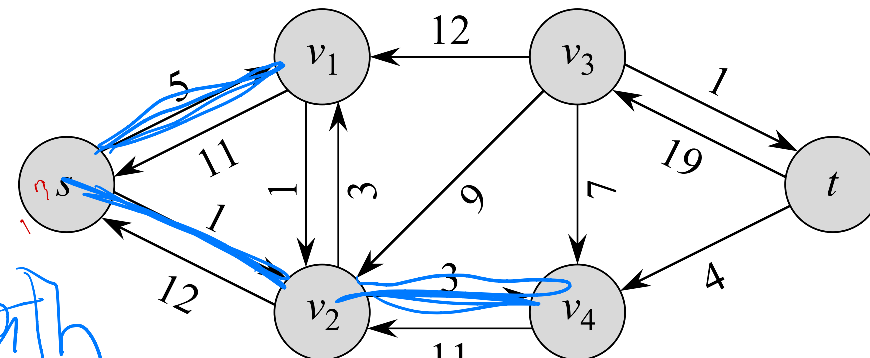


(b)

4 units of s-v2-v3-t



(c)

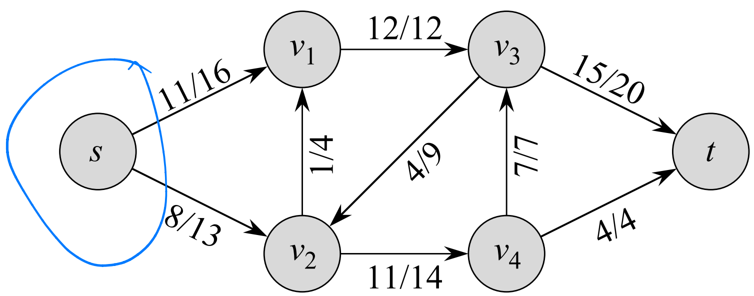


(d)

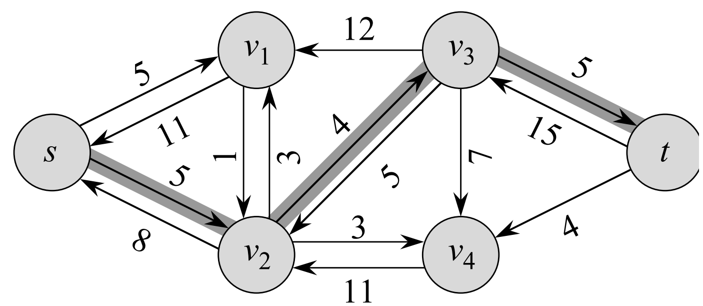
no path from s to t in G_f

Updating a Flow

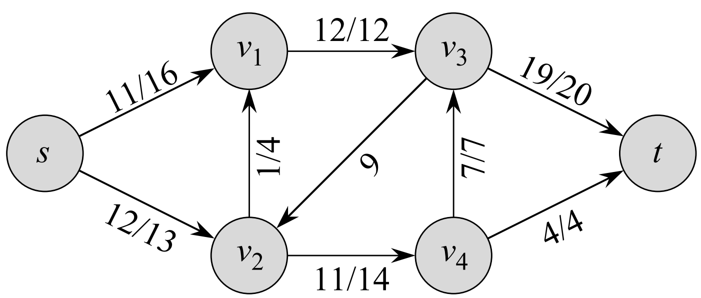
- Send flow along the path defined by the residual graph.
- Amount: minimum of capacity of all residual edges in the augmenting path.
- If a residual edge is a graph edge, then add the flow.
- If a residual edge is a reverse edge, then subtract the flow.



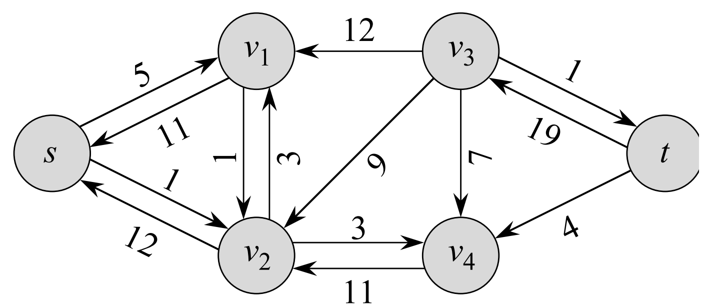
(a)



(b)



(c)



(d)

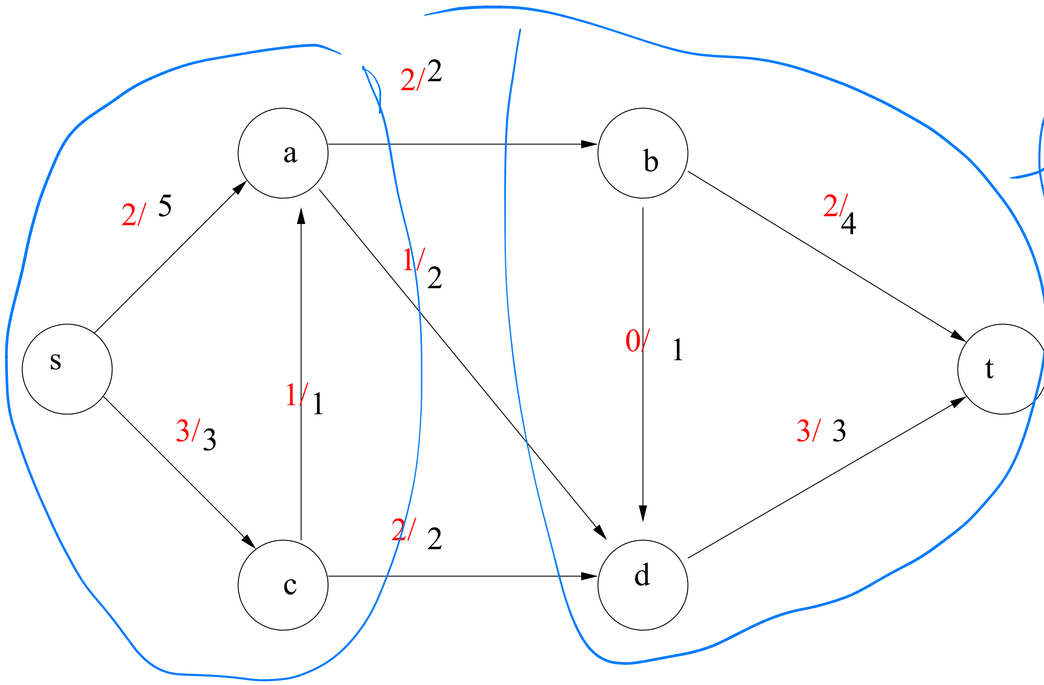
s - t Cuts

An s - t cut satsfies

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

Cap = 2+2+2 = 6

flow = 2+1+2 = 5



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)**

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

s - t Cuts

An s **-** t **cut satsfies**

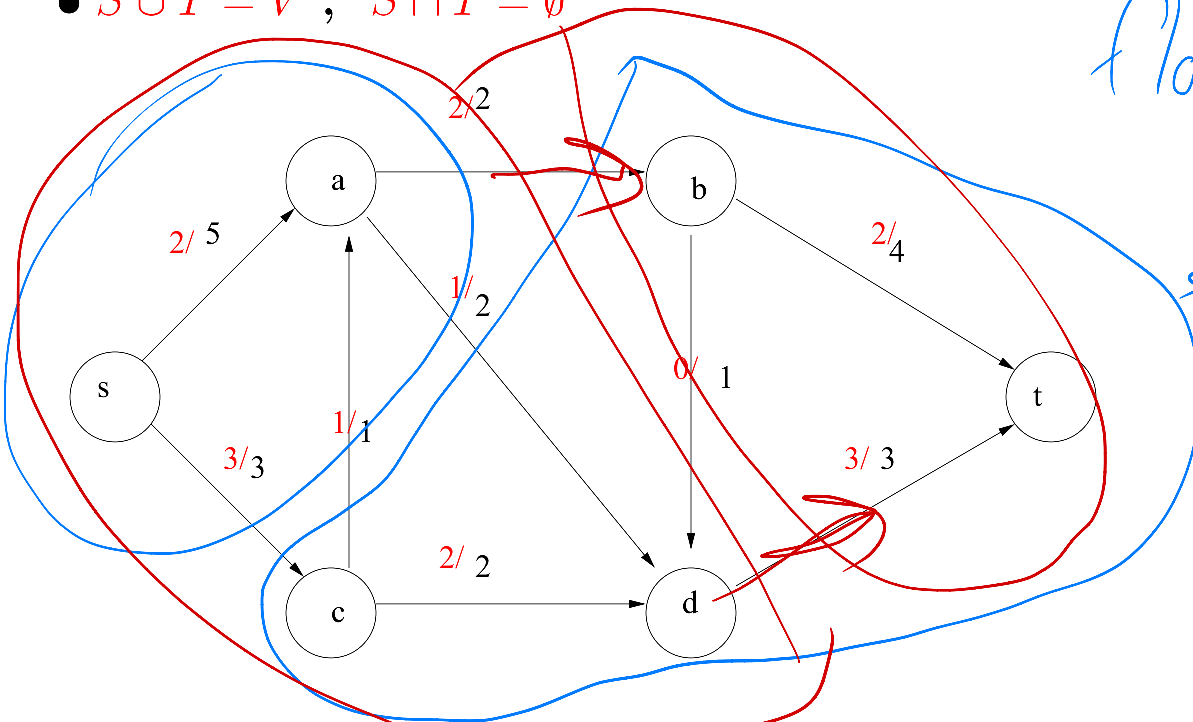
- $s \in S$, $t \in T$
- $S \cup T = V$, $S \cap T = \emptyset$

$$cap = 2+2+3 = 7$$

$$flow = 2+1-1+3 = 5$$

$$f(S, V-S) = 5$$

$$f(\{s,a\}, V-\{s,a\}) = 5$$



Capacity of a cut **(only forward edges)**

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

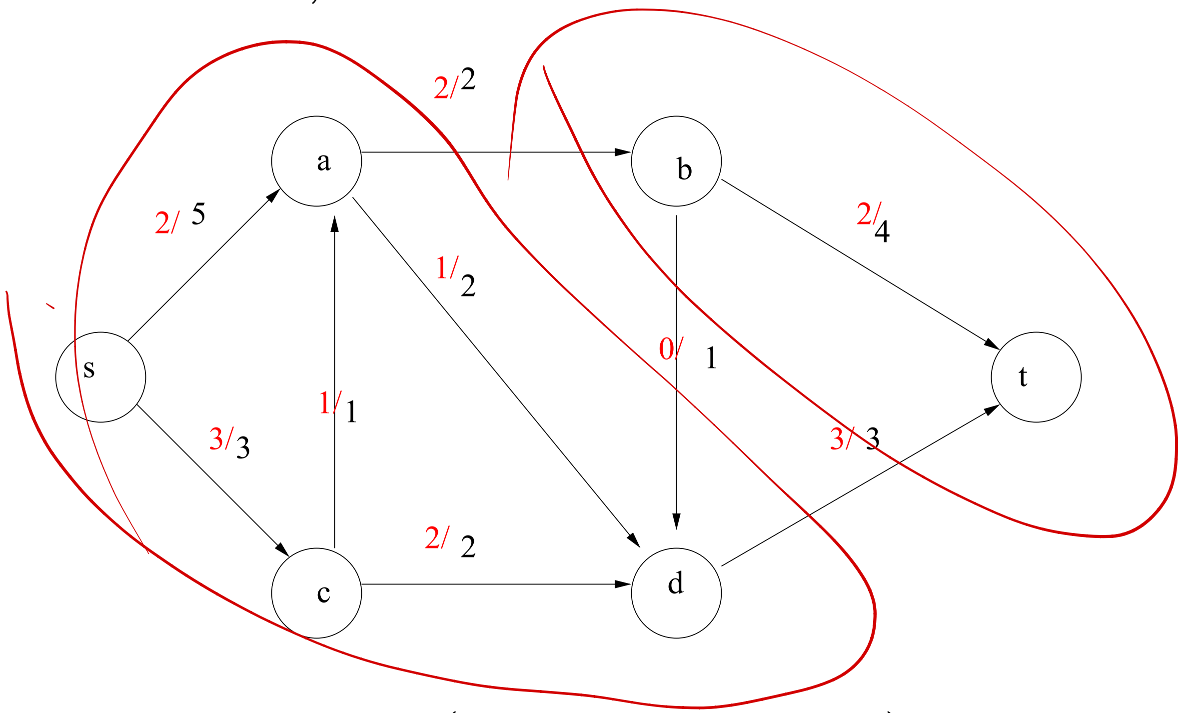
Flow crossing a cut **(net flow)**

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

s - t Cuts

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)**

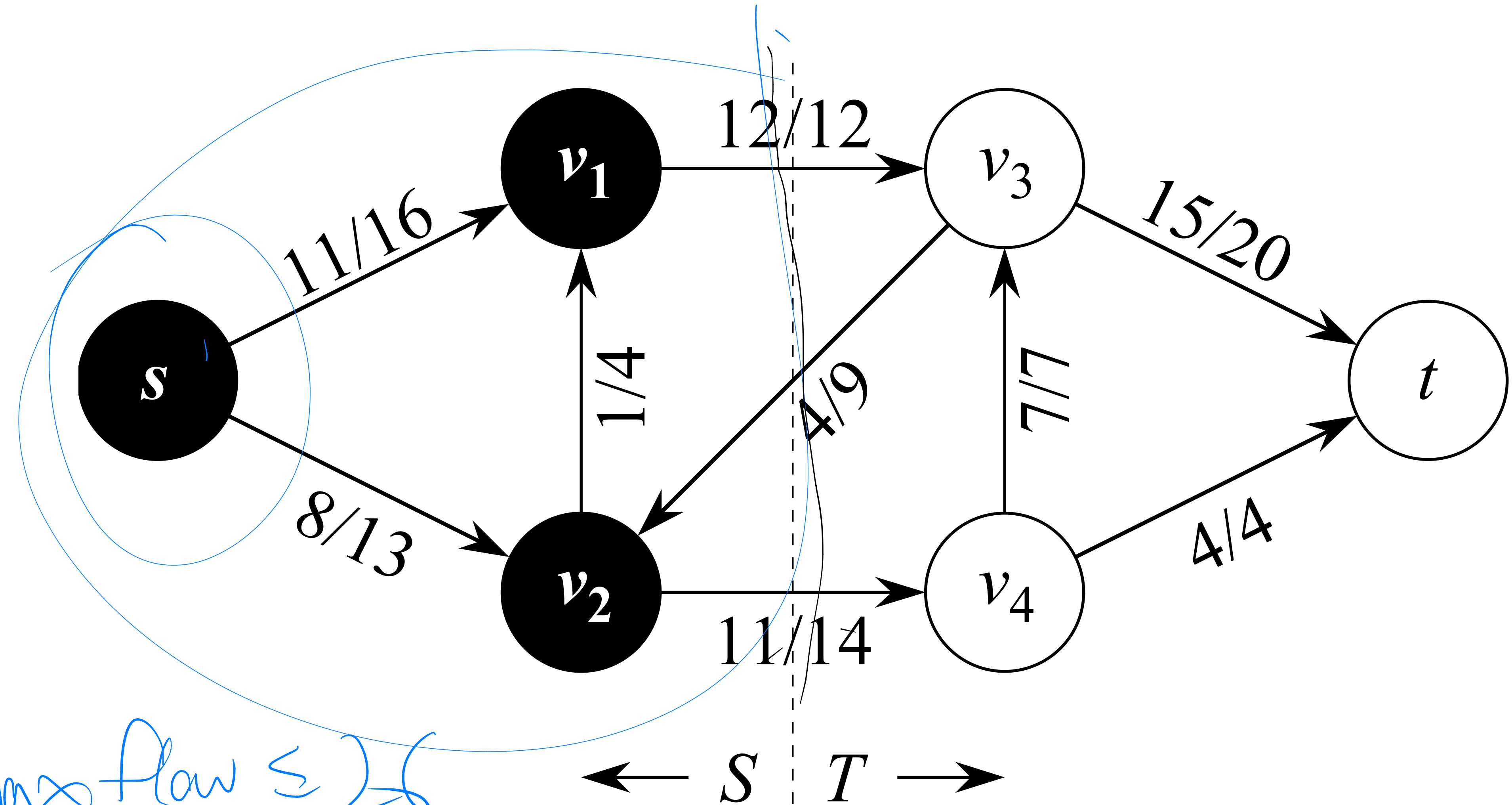
$$c(S,T) = \sum_{u \in S, v \in T} c(u,v)$$

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 **,** $f(S,T) \leq c(S,T)$ **(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 ≤ 26

Ford-Fulkerson
~1956

Max-flow min-cut theorem

flows cuts

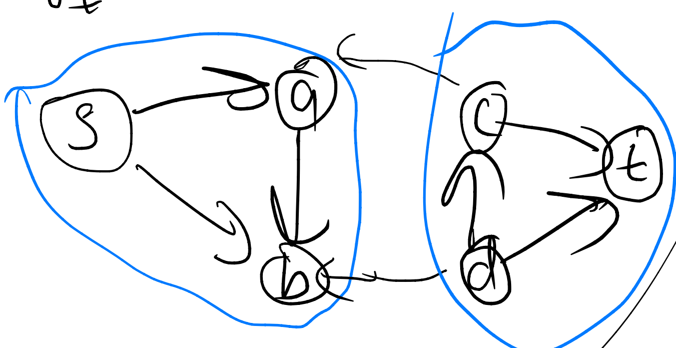
If f is a flow in a flow network $G = (V, E)$ with source s and sink t, then the following conditions are equivalent:

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.

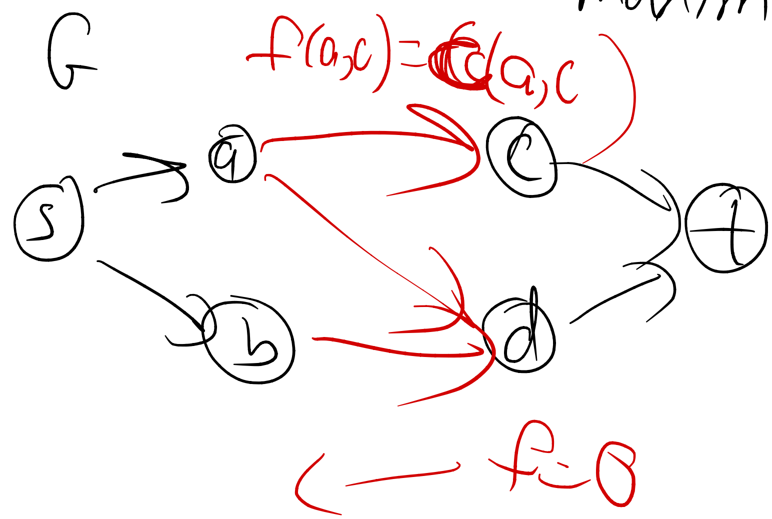
$1 \Rightarrow 2$. Consider $\neg 2 \Rightarrow \neg 1$

G_f has an aug. path $\Rightarrow$ f is not maximum.

G_f



G



$2 \Rightarrow 3$

$3 \Rightarrow 1$

Proof

Ford Fulkerson expanded

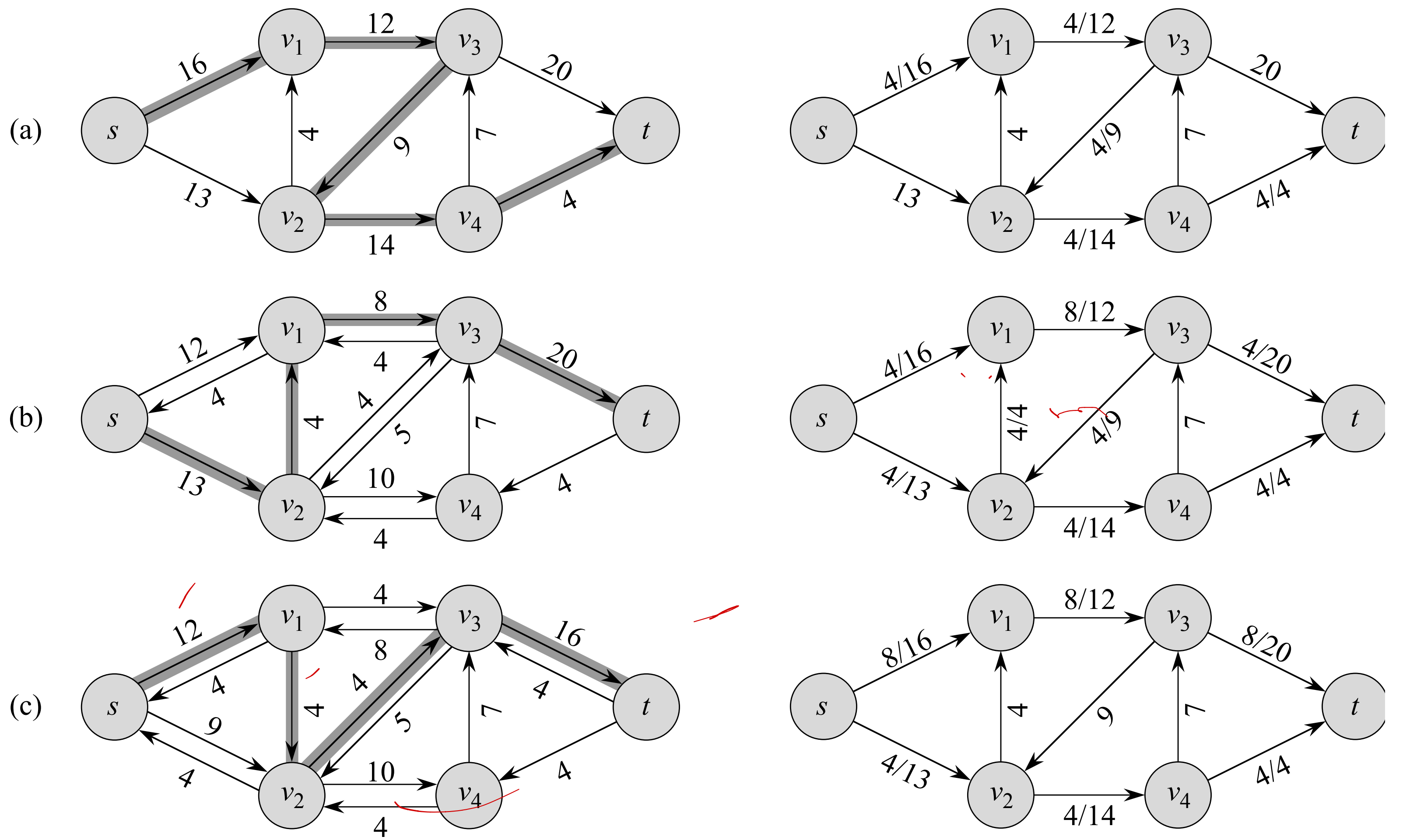
$Ford - Fulkerson(G, s, t)$

BFS

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

Algorithm

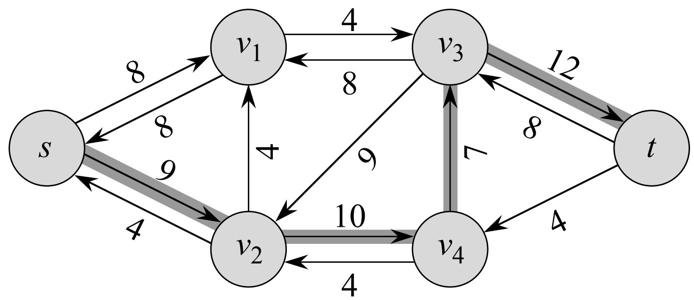
Residual graph (Capacities) | **Graph (Flow/Capacity)**

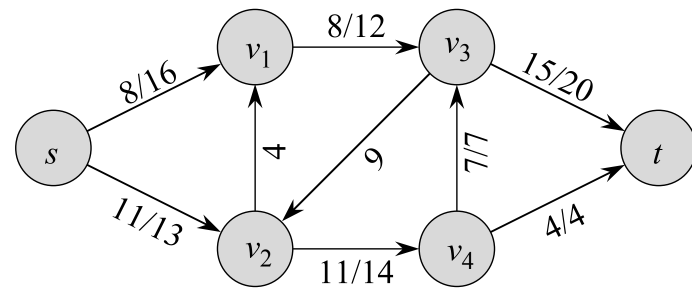


Algorithm continued

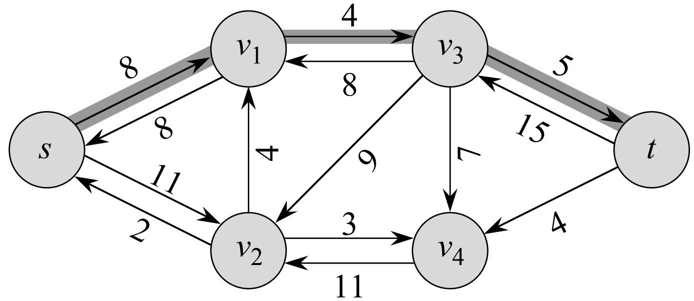
Residual graph (Capacities) | **Graph (Flow/Capacity)**

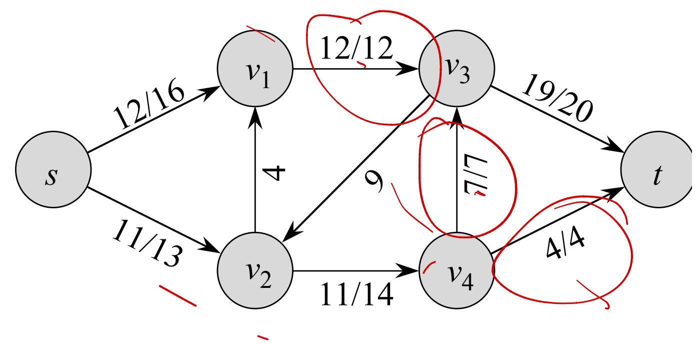
(d)



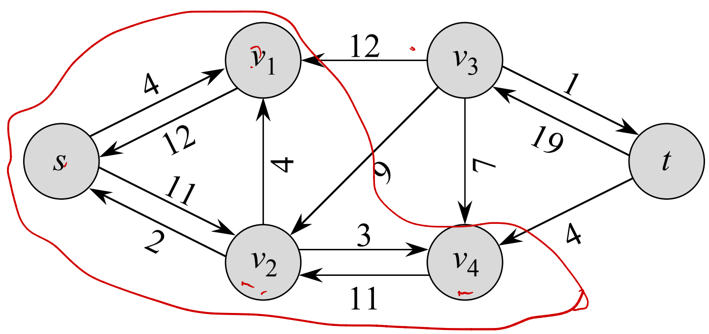


(e)





(f)



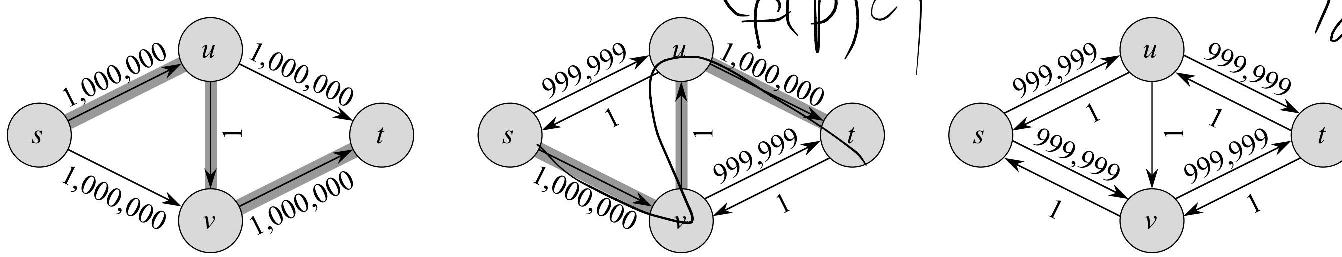
Analysis

- 1 iteration of FF takes $O(E+V)$ time (breadth-first search plus bookkeeping).
- Each iteration sends at least one unit of flow.
- Total time $O(f^*E)$. not polynomial
- This algorithm is only psuedo-polynomial.

value of max-flow

input
V nodes
E edges/capacities
$U = \max_e (c(e))$
$\log U$ bits for each capacity

$c_f(p) \geq 1$



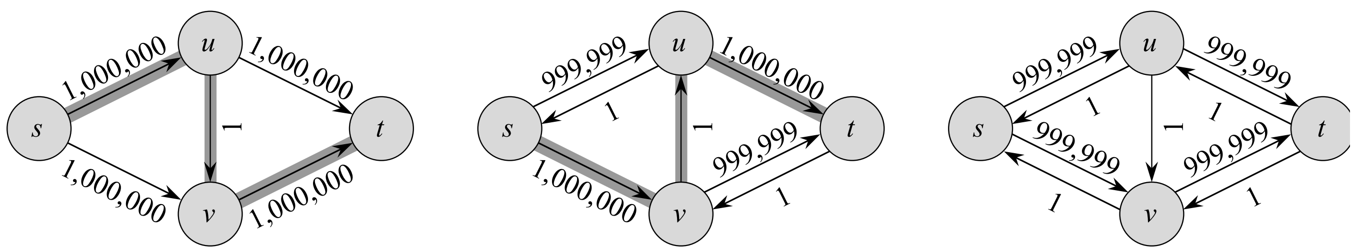
(a) (b) (c)

input size $\approx E \lg U$

flow value $\leq VU$
running time $\leq VUE$

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)