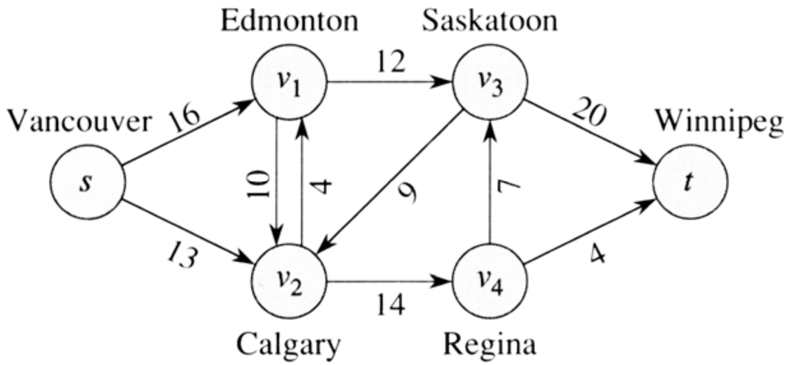
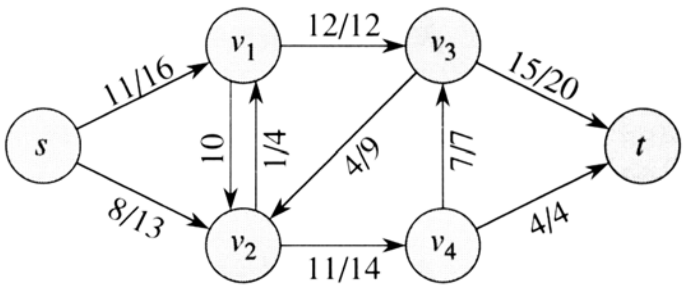
------------------------------ 26 ------------------------------

Maximum Flow

26.1 Flow networks

***Flow networks*:** a directed graph $G=(V, E)$, in which each $(u,v)\in E$ has a capacity $c(u,v)\geq 0$. If $(u,v)\notin E$, we assume $c(u,v)=0$. There are a *source vertex s* and a *sink vertex t* in G. For every vertex v in G, there is a path $s\to v\to t$.





***Flow*:** a real function f: $V\times V \to R$ satisfying the following three properties.

***Capacity constraint*:** For all $u,v\in V$, $f(u,v)\leq c(u,v)$
***Skew symmetry*:** For all $u,v\in V$, $f(u,v)=-f(v,u)$
***Flow conservation*:** For all $u\in V-\{s, t\}$,

$$\sum_{v\in V} f(u,v)=0.$$

* ***Positive net flow*** entering (leaving) a vertex u:

$$\sum_{v\in V \text{ and } f(v,u)>0} f(v,u) \quad \left(\sum_{v\in V \text{ and } f(u,v)>0} f(u,v) \right).$$

* For all $u\in V-\{s, t\}$, we have
Positive net flow entering u
$=$ Positive net flow leaving u.

* For all $u\in V-\{s, t\}$, $\sum_{v\in V} f(v,u)=0$. (Total flow into a vertex is 0.

* $f(u,v)$ is called the ***net flow*** from u to v. It can be positive or negative.

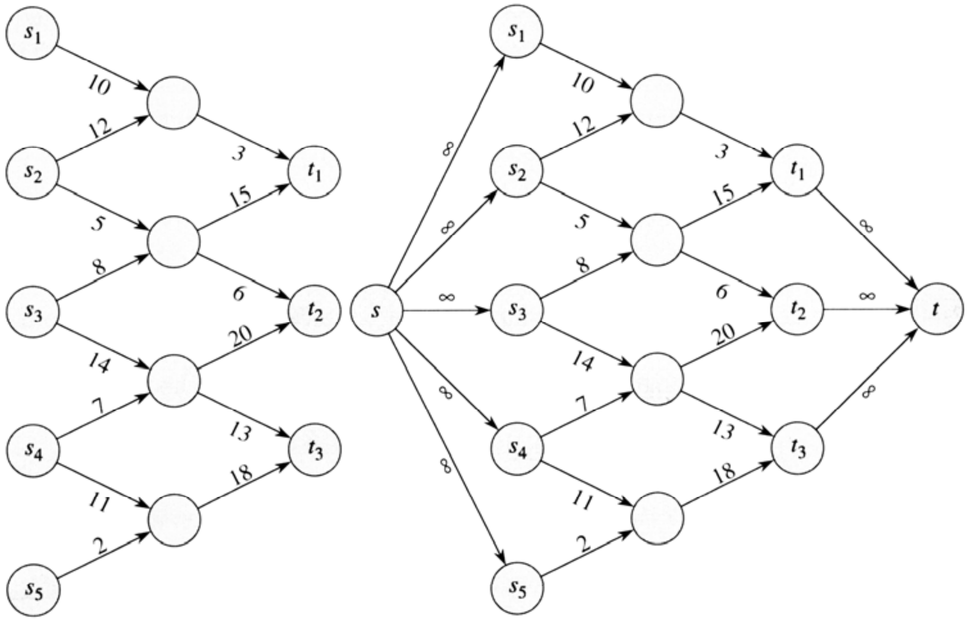
* The value of a flow f is $|f|=\sum_{v\in V} f(s,v)$.

* ***Maximum-flow problem*:** finding a flow of maximum value from s to t.

If $(u,v) \notin E$ and $(v,u) \notin E$, $f(v,u)=f(u,v)=0$.

* Nonzero net flow from u to v implies $(u,v) \in E$ or $(v,u) \in E$.

* ***Networks with multiple sources and sinks***



(a) (b)

* Let X and Y be sets of vertices. For simplicity, define

$$f(X,Y)=\sum_{x \in X}\sum_{y \in Y} f(x,y) \text{ and } c(X,Y)=\sum_{x \in X}\sum_{y \in Y} c(x,y).$$

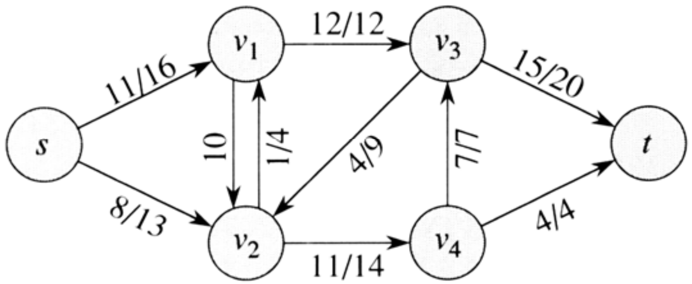
26.2 The Ford-Fulkerson method

* We call it a method instead of algorithm, because it encompasses several implementations.

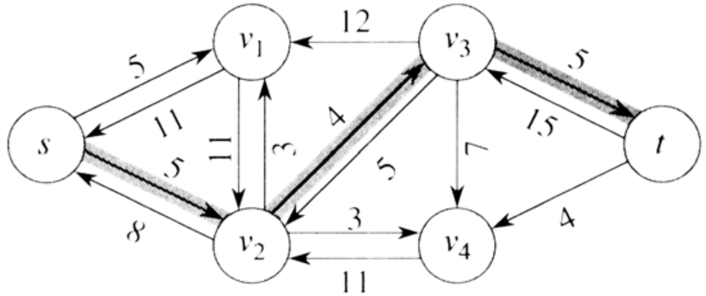
```
FORD-FULKERSON-METHOD(G, s, t)
initialize flow f to 0
while there exists an augmenting path p
      do augment flow f along p
return f
```

Example:

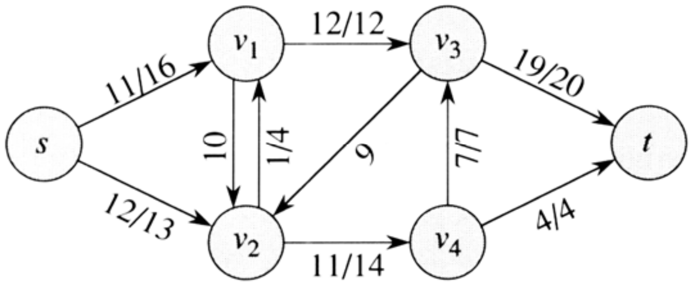
G and f



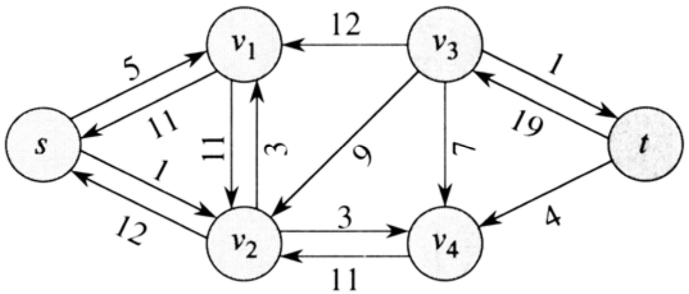
Residual network G_f with an augmenting path p



New f



New G_f



Residual networks G_f

(1) *residual capacity* of (u,v) is given as

$$c_f(u,v) = c(u,v) - f(u,v).$$

(2) $G_f = (V, E_f)$, where

$$E_f = \{(u,v) \in V \times V: c_f(u,v) > 0\}.$$

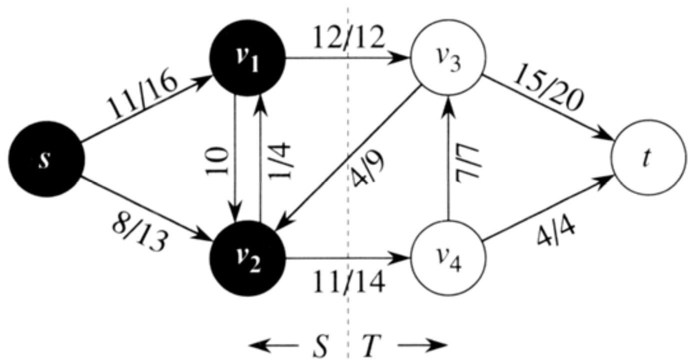
***Augmenting path*:** a simple path $s \to t$ in G_f.

***Cut of a flow network*:** a partition of V into S and $T=V-S$ such that $s \in S$ and $t \in T$.

Net flow across a cut: $f(S, T)$.

Capacity of a cut: $c(S, T)$.

Example: $|f|=19$, $f(S, T)=19$, and $c(S, T)=26$.



Lemma 26.5: For any cut (S, T), $f(S, T)=|f|$.

Corollary 26.6: For any f, $|f| \le c(S, T)$.

Theorem 26.7: (Maximum flow minimum cut)
The following are equivalent:

1. f is a maximum flow

2. G_f contains no augmenting paths

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

Proof: (1)$\rightarrow$(2) (By contraction) Suppose there is an augmenting path p. We have $|f+f_p|>|f|$, which contradicts to "f is a maximum flow."

(2)$\rightarrow$(3) Since (2), G_f contains no path from s to t. Define $S=\{v \mid$ there is a $s\rightarrow v$ in $G_f\}$ and $T=V-S$. Note that $t\in T$. Thus, (S, T) is a cut.

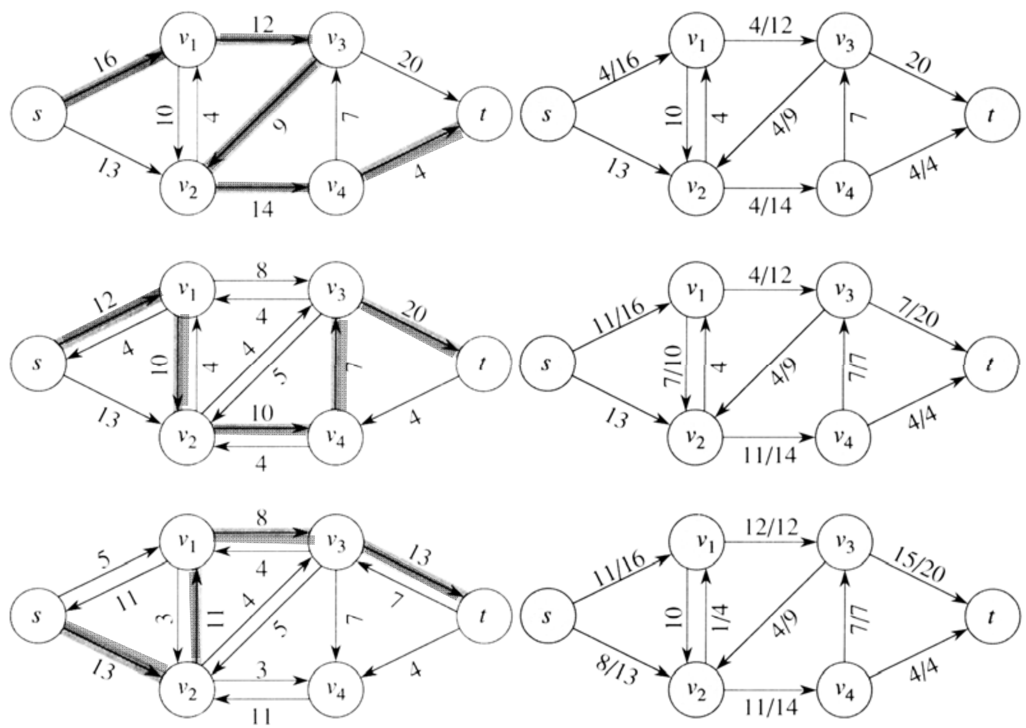
For each pair $u\in S$ and $v\in T$, we have $f(u,v)=c(u,v)$, since otherwise $(u,v)\in E_f$ and v is in S. By lemma 26.5, $|f|=f(S, T)=c(S, T)$.

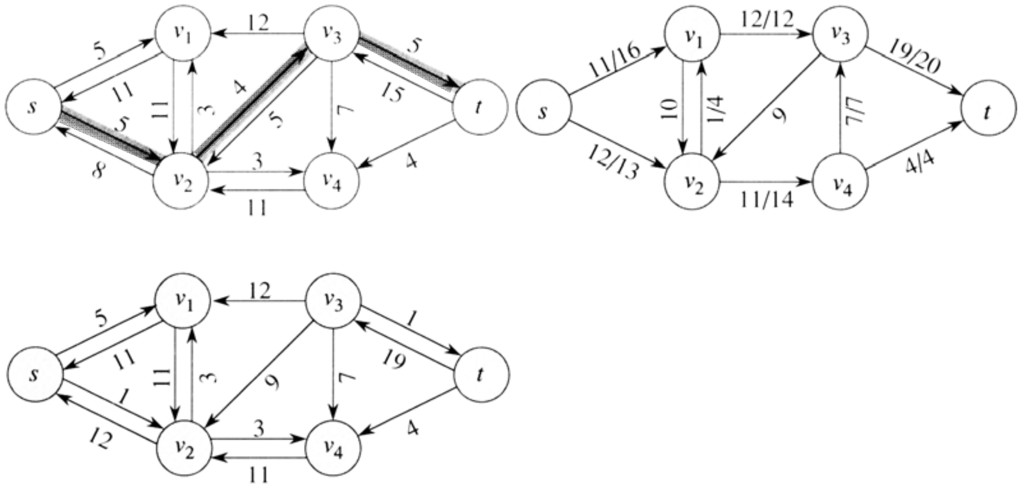
(3)$\rightarrow$(1): By corollary 26.6, $|f|\leq c(S, T)$ for all cuts. The condition $|f|=c(S, T)$ thus implies f is a maximum flow.

Q.E.D.

The basic algorithm

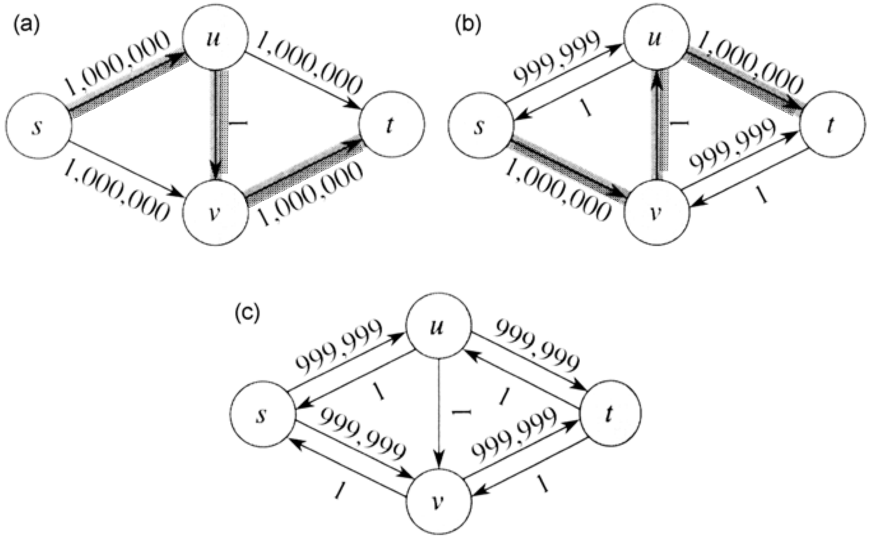
```
FORD-FULKERSON(G, s, t)
for each edge (u, v) ∈ E[G]
     do f[u, v] ← 0
        f[v, u] ← 0
while there exists a path p from s to t in G_f
     do c_f(p) ← min {c_f(u, v) : (u, v) is in p}
        for each edge (u, v) in p
             do f[u, v] ← f[u, v] + c_f(p)
                f[v, u] ← −f[u, v]
```





Analysis:

(1) $|f|$ is increasing. But, if p is chosen poorly, the algorithm might not even terminate (while $c(u,v)$'s are irrational numbers).

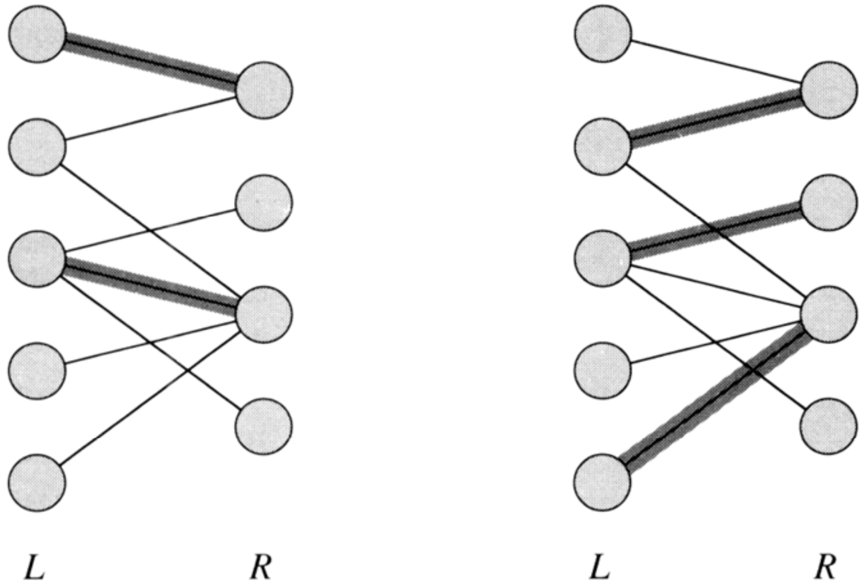


(2) If $c(u,v)$'s are integers, it performs in $O(E|f^*|)$ time, where f^* is the maximum flow.

(3) If p is chosen by using breadth-first search, the algorithm is called the ***Edmonds-Karp algorithm***. It performs in $O(VE^2)$ time. (We are not going to prove this.)

26.3 Maximum bipartite matching

A bipartite graph **(undirected)** $\boldsymbol{G=(V=L\cup R,E)}$ **and two matchings**



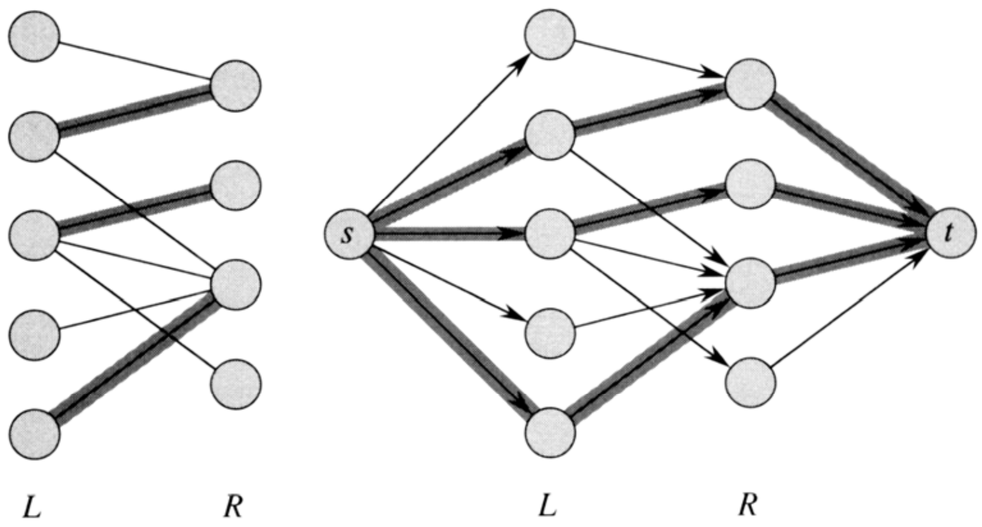
(a) (b)

***Corresponding flow network*:** $G'=(V',E')$, where

$V' = V \cup \{s, t\}$,

$E' = \{(s,u): u \in L\}$
$\cup \{(u,v): u \in L, v \in R, \text{ and } (u,v) \in E\}$
$\cup \{(v,t): v \in R\}$, and

each edge is assigned unit capacity.



(a) (b)

Lemma 26.10: If M is a matching in G, then there is an integer-valued flow f in G' with $|M|=|f|$. Conversely, if f is an integer-valued flow f in G', then there is a matching M in G with $|M|=|f|$.

Theorem 26.11: If all $c(u,v)$'s are integer, all $f^*(u,v)$'s produced by Ford-Fulkerson method are integers. (by induction.)

Corollary 26.12: $|f^*|$ of G' is equal to the cardinality of a maximum matching in G.

* The maximum bipartite matching problem can be solved in $O(Ef^*)=O(EV)$ time.

Homework: Ex. 26.2-6, 26.2-11, Pro. 26-1, 26-2.

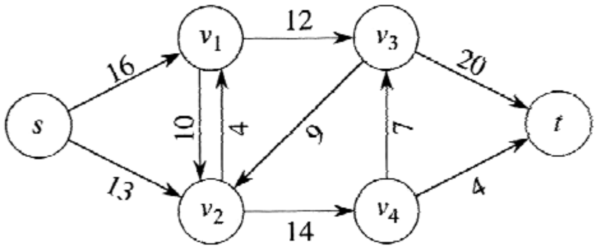
Differences in the 3rd Edition

(Consider only positive flows)

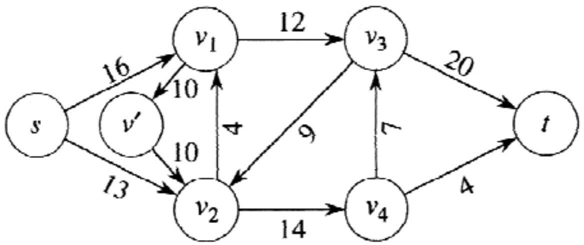
Flow networks:

Assume that G contains no *antiparallel* edges.
(If $(u, v) \in E$, then $(v, u) \notin E$.)

Handling antiparallel edges:



(a)



(b)

Converting a network with antiparallel edges into one with no antiparallel edges

***Flow*:** a real function f: $V \times V \rightarrow R$ satisfying the following TWO properties:

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

***Flow conservation*:** For all $u \in V - \{s, t\}$,
$\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$. (flow in equals flow out)

The residual capacity:

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

The basic Ford-Fulkerson algorithm

FORD-FULKERSON(G, s, t)

for each edge $(u, v) \in E[G]$
  do $f[u, v] \leftarrow 0$
while there exists a path p from s to t in G_f
  do $c_f(p) \leftarrow \min\{c_f(u, v): (u, v) \text{ is in } p\}$
    for each edge (u, v) in p **do**
      if $(u, v) \in E[G]$
        then $f[u, v] \leftarrow f[u, v] + c_f(p)$
        else $f[v, u] \leftarrow f[v, u] - c_f(p)$