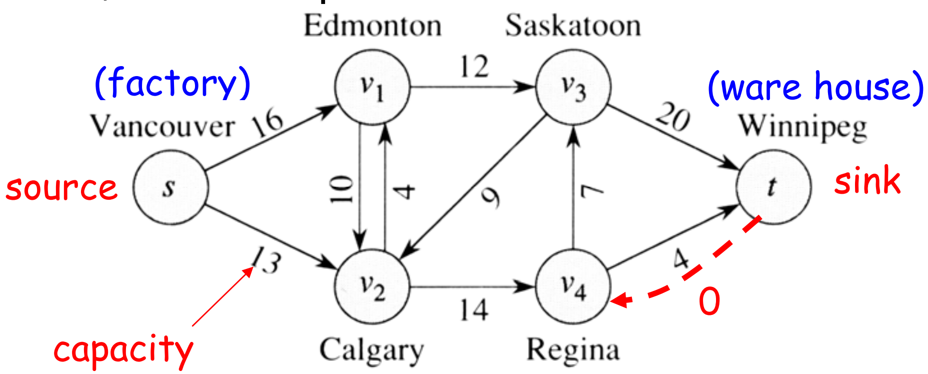
---------------------------------- 26 --------------------------------

Maximum Flow

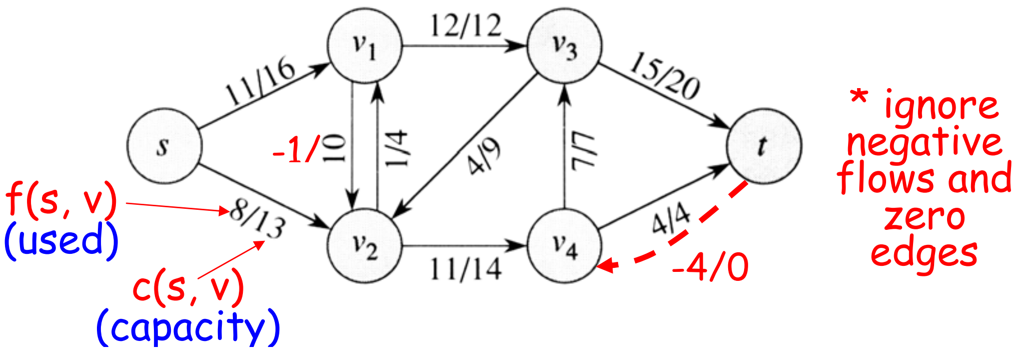
26.1 Flow networks

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 $s\rightarrow v\rightarrow t$.

26-1x



26-1a



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

Capacity constraint: For all $u,v\in V$, $f(u,v)\leq c(u,v)$

Skew symmetry: For all $u,v\in V$, $f(u,v)=-f(v,u)$

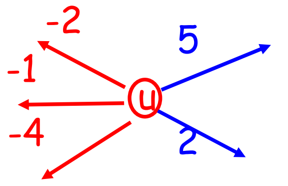
Flow conservation: For all $u\in V-\{s, t\}$,

26-2a

出 守 恒

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

total flow out off u = 0



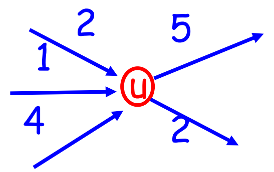
* ***Positive net flow*** entering (leaving) a vertex u:

入 出

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

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

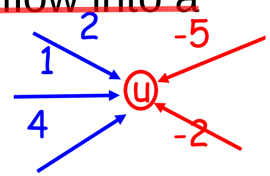
(flow conservation: positive in = positive out)



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

入

(flow conservation: total in = 0)



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

total out from s

total into t

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

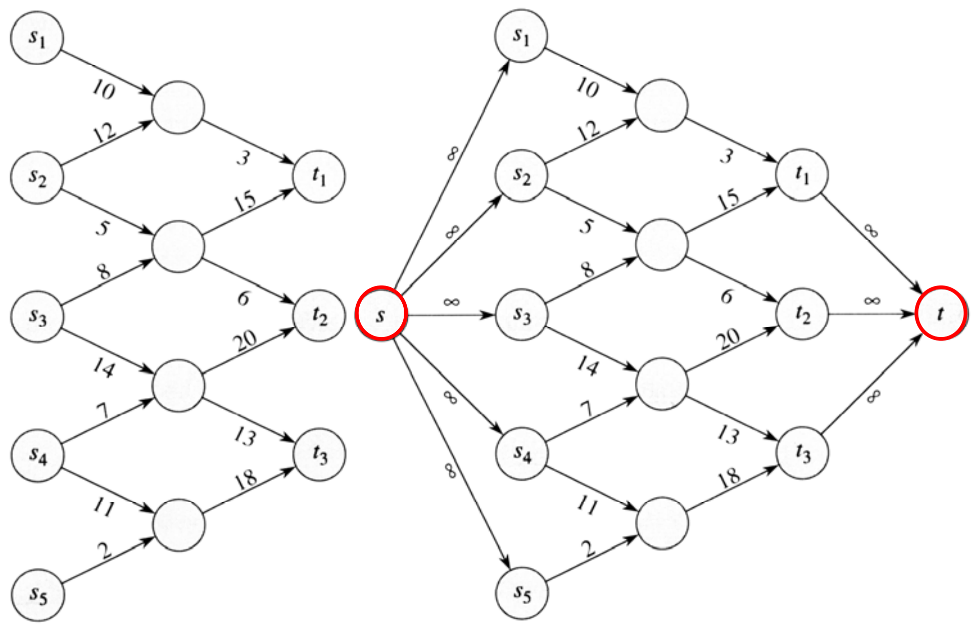
* ***Maximum-flow problem***: finding a flow of maximum value from *s* to *t*.

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

⇨ find a path: O(E) (not O(n^2)) BFS or DFS

* 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

See 26-6 example

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

26.2 The Ford-Fulkerson method

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

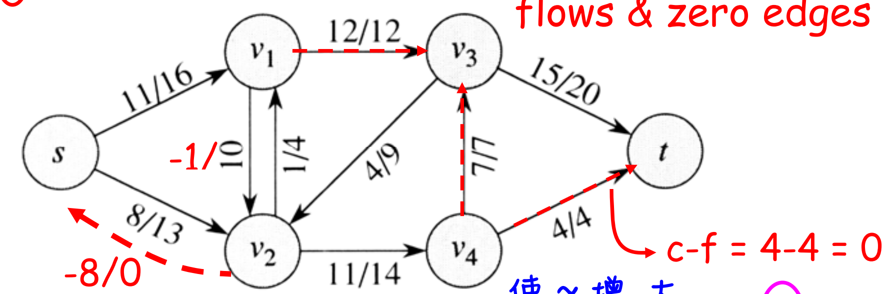
FORD-FULKERSON-METHOD(G, s, t)
1 initialize flow f to 0 $\quad f = 0$
2 **while** there exists an augmenting path p
3 **do** augment flow f along p
4 **return** f $\quad f = f + f_p$

Example:

26-4x

G and f

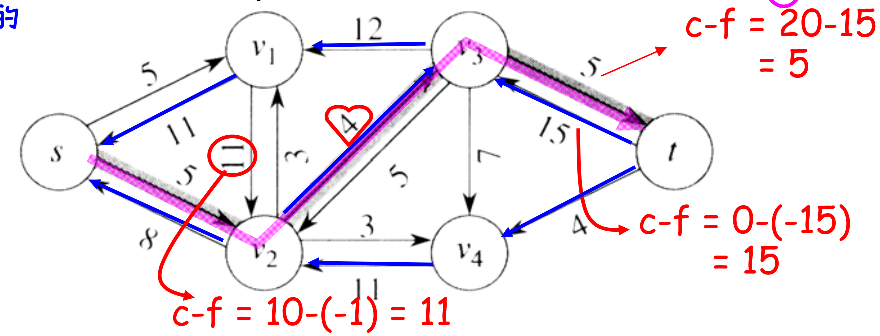
* ignore negative flows & zero edges



使 ~ 增 大

Residual network G_f with an augmenting path p

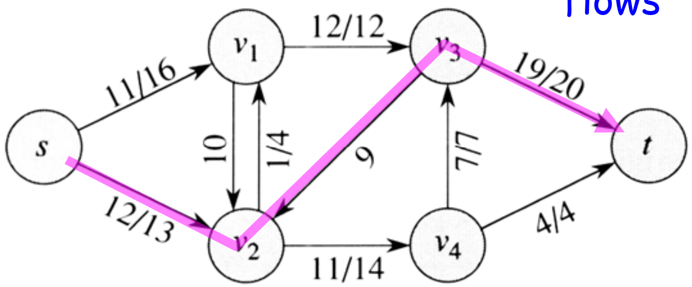
殘 餘 的



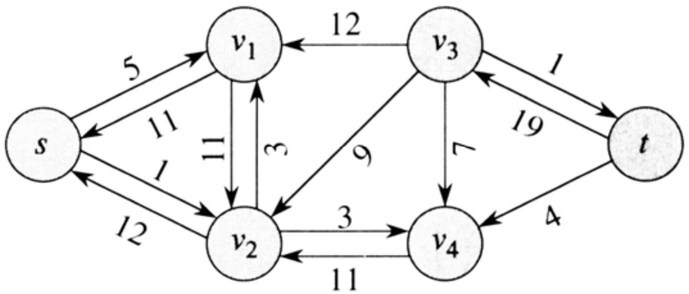
New $f \leftarrow f + f_p$

* ignore negative flows

26-5x



New G_f



Residual networks G_f

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

$$\underbrace{c_f(u,v)}_{} = \underbrace{c(u,v)}_{\text{capacity}} - \underbrace{f(u,v)}_{\text{used}}.$$

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

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

only non-negative edges

set of edges that still can be used!

Augmenting path: a simple path $s \rightarrow t$ 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: $f(S, T)$.

(See 26-3 for de

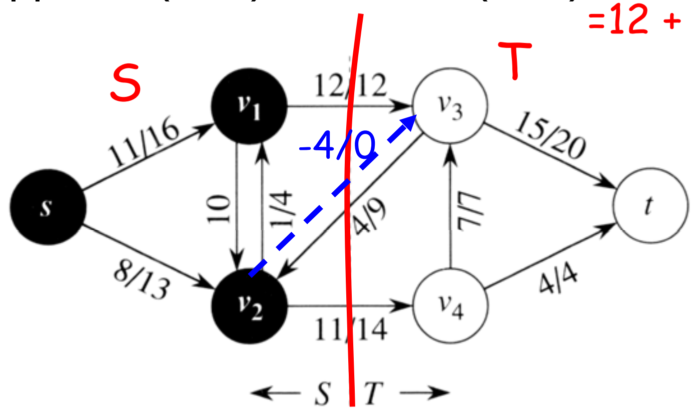
Capacity of a cut: $c(S, T)$.

26-6x

= 12 + (-4) + 11

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

=12 + 0 + 14



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

(flow conservation)

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

26-6a

(capacity constraint)

Every cut sets an upper bound on $|f^*|$.

Theorem 26.7: (Maximum flow minimum cut) 26-7a

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.

correctness

$|f| \le c(S, T)$

$c(S, T)$ is a minimum cut

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

26-7b

(2)→(3) Since (2), G_f contains no path from s to t. Define $S=\{v \mid$ there is a $s\to v$ 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 $f(\underset{\in S}{u},\underset{\in T}{v})= c(u,v)$, since otherwise $(u,v)\in E_f$ and v is in S. By lemma 26.5, $|f|=f(S, T)=c(S, T)$.

(3)→(1): By corollary 26.6, $|f|\le c(S, T)$ 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]
```

$f = 0$

f_p

$f = f+f_p$

$f[v,u] - c_f(p)$

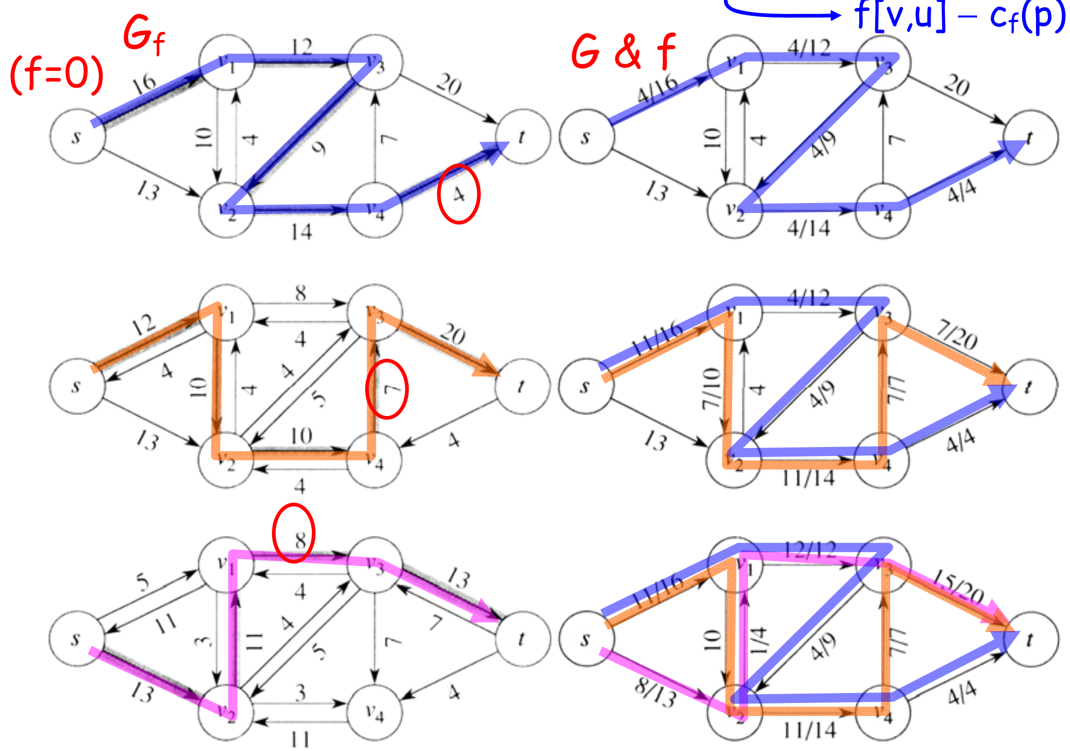
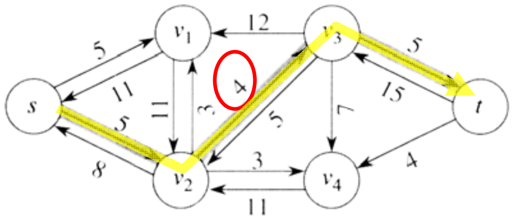


Fig 26-4

G_f



G & f

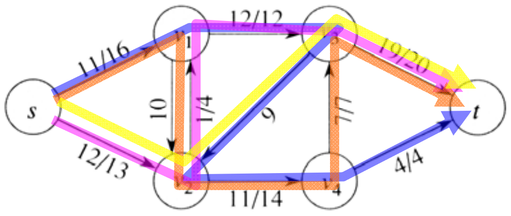
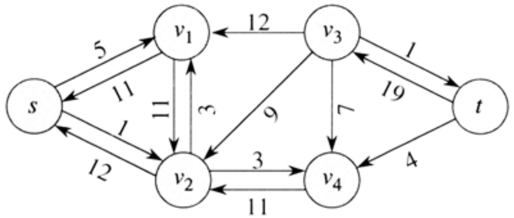


Fig 26-5



no augmenting paths
⇨ stop

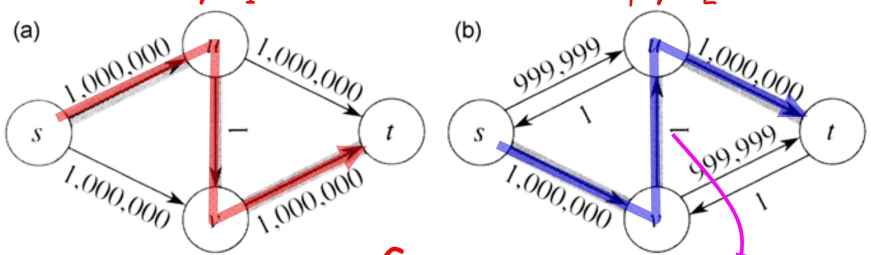
26-9x

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).

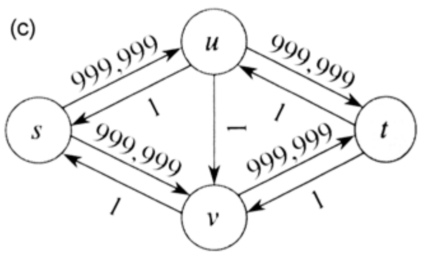
G , P_1 $\qquad$ G_f , P_2

26-9y



$c_f = 0 - (-1)$
$= 1$

G_f



find a path: $O(E)^{\text{BFS or DFS}}$

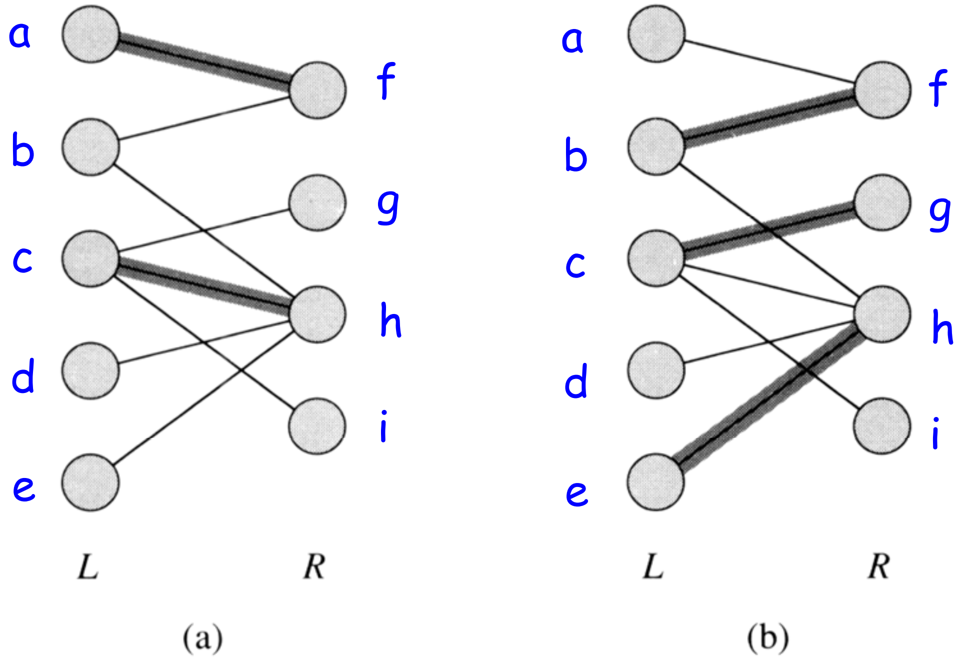
(2) If $c(u,v)$'s are integers, it performs in $O(E|f^*|)$ time, where f^* is the maximum flow. at most $|f^*|$ times

(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.) at most VE times

26.3 Maximum bipartite matching

A bipartite graph **(undirected)** ***G=(V=L∪R,E)*** **and two matchings**

Men $\qquad$ Women



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

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

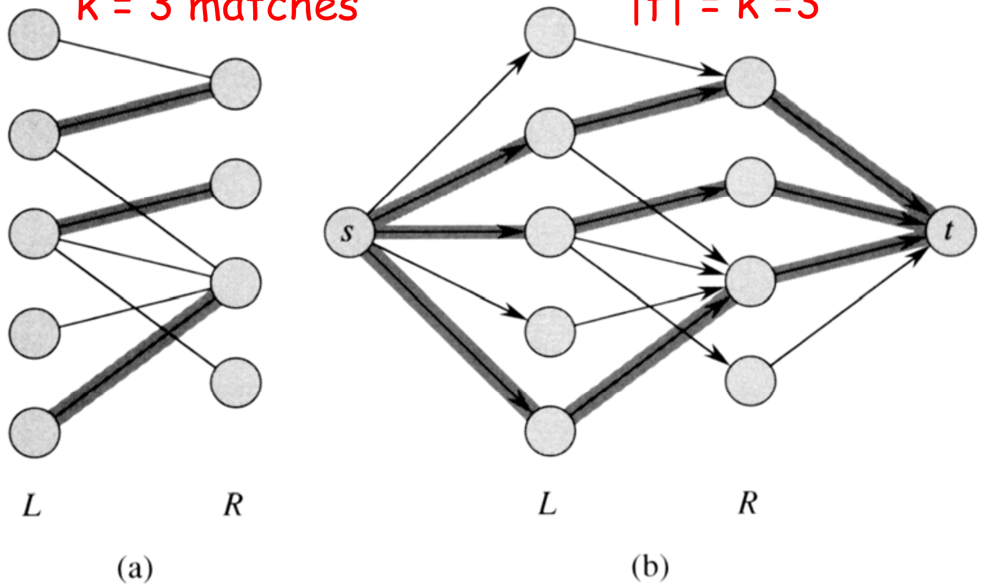
$E' = \{(s,u): u \in L\}$
$\cup \{(u,v): u \in L, v \in R, \text{ and } (u,v) \in E\}$
$\cup \{(v,t): v \in R\}$, and

each edge is assigned unit capacity.

26-11a

k = 3 matches

|f| = k =3



(a)

(b)

matching <--> integer-valued flow

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.) $G, f \dashrightarrow G_f \dashrightarrow f_p \dashrightarrow f + f_p$

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. $f^* \le |V|/2$

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

flow on undirected G: $\circ \overset{4}{\text{—}} \circ \Rightarrow \circ \underset{4}{\overset{4}{\rightleftarrows}} \circ$

26-12x

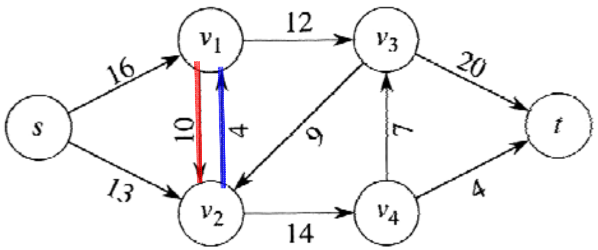
Differences in the 3rd Edition

(Consider only positive flows)

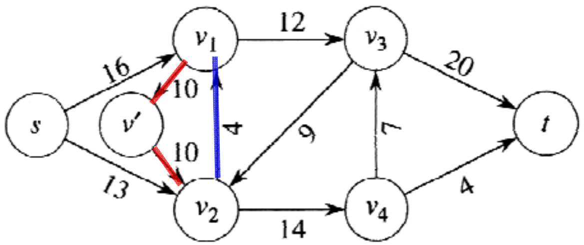
Flow networks:

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

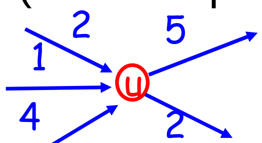
Flow: a real function f: $V \times V \rightarrow R$ satisfying the following TWO properties:

Capacity constraint: For all $u,v \in V$,
$0 \leq f(u,v) \leq c(u,v)$.
Only positive flows!!!

Flow conservation: For all $u \in V - \{s, t\}$, 26-2a

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

positive in = positive out



The residual capacity: ☹ (slightly complicated)

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

The basic Ford-Fulkerson algorithm

FORD-FULKERSON(G, s, t) ☹ (* needs converting)

```
for each edge (u, v) ∈ E[G]
    do f[u, v] ← 0                 one side (slightly simpler) ☺
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
            if (u, v) ∈ E[G]
                then f[u, v] ← f[u, v] + c_f(p)
                else  f[v, u] ← f[v, u] − c_f(p)
```

☹ (if-then-else)