算法设计与分析

Computer Algorithm Design & Analysis

2024.3

王多强

QQ：1097412466

Chapter 26

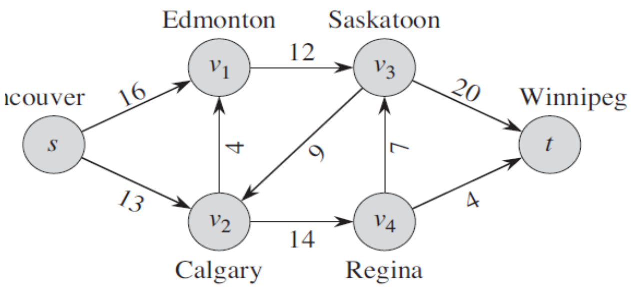
Maximum Flow

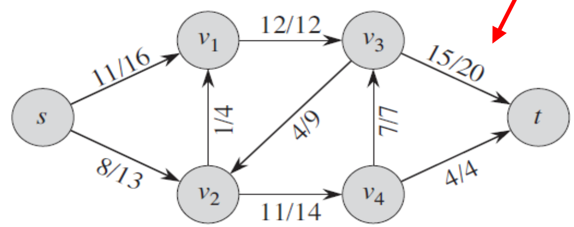
最大流

◆ **最大流问题**是**网络流**理论研究的一个基本问题。

问题引出：实例 **物流网络**

实际运量/容量





最优解

问题建模：带权有向图

结点代表城市，边代表道路，箭头代表货物运输的方向，边上的权重代表道路的运量限制 (**容量**)。有一个**源点** *s* 和一个**汇点** *t*，**从 s 向 t"运输"货物。**

问题设定：在不违反任何道路**容量限制**的条件下，求**从 s 向 t 运送货物的最大速率。**

——这一问题的抽象称为**最大流问题**。

26.1 流网络

流网络是一个有向图，记为 $G = (V, E)$，有向边表示**流向**，边上定义有**容量函数**：$c: E \rightarrow R^{+} \cup \{0\}$，每条边 $(u, v) \in E$ 上有一个**非负的容量值** $c(u, v) \geq 0$，表示该边的最大通量。如果 $(u, v) \notin E$，不失一般性，令 $c(u, v) = 0$。

G 中有一个称为**源点**的出发点和一个称为**汇点**的汇集点。而 G 是一个**从源点到汇点的连通图**：每个结点都在从源点到汇点的某条路径上，且除源结点外，每个结点至少有一条流入的边；除汇点外，每个结点至少有一条流出的边；**不允许有自循环**，并根据图的一般性质有 $|E| \geq |V| - 1$。

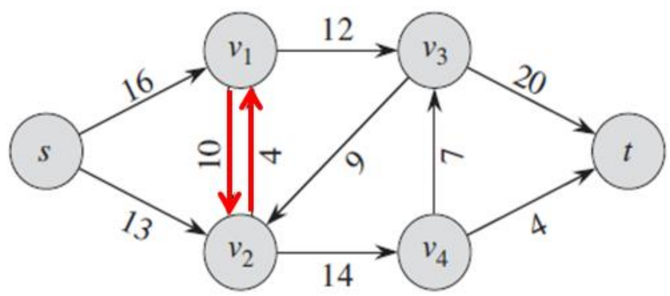
(1) 标准流网络：

约定：只有单一的源点和汇点，且没有反向边

- **反向边**：若 $(u,v) \in E$ 同时 $(v,u) \in E$，则 (u,v) 和 (v,u) 互为**反向边，也称为反向平行边**。
- **无反向边**：如果 $(u,v) \in E$，则 $(v,u) \notin E$，反之亦然。

2）非标准流网络：

不满足上述标准流网络要求的流网络是**非标准流网络**：图中有多于1个的源点或汇点、有反向边。



如，一个非标准流网络**有反向边**

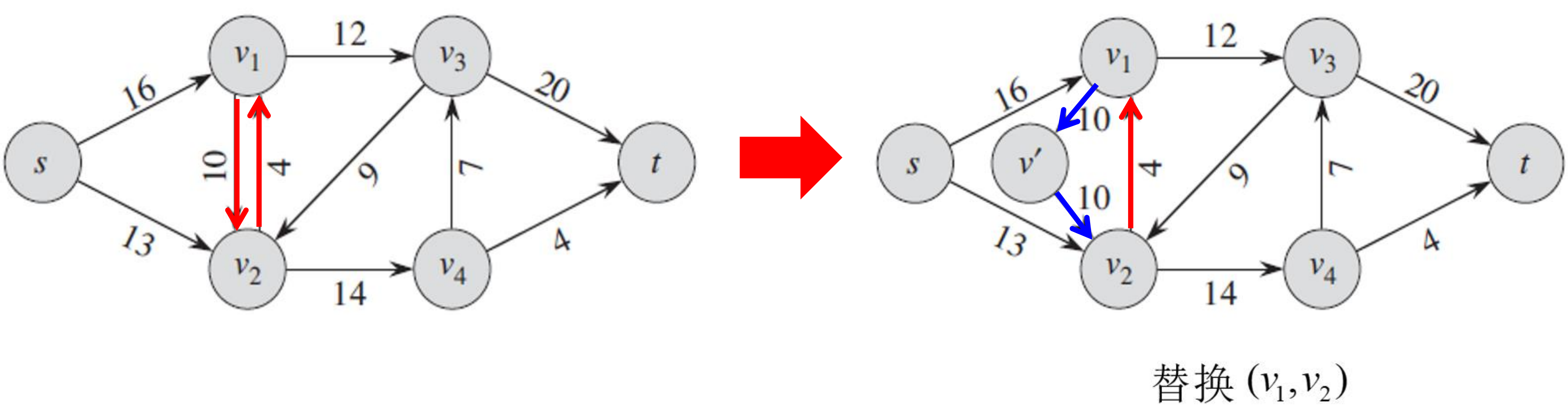
◆ 非标准流网络可以转化为标准流网络

(1) 若有反向平行边 (u, v) 和 (v, u)

- 选择其中一条，然后**加入一个新结点**，将该边分为两条边。

如图，选择边 (v_1, v_2)，然后增加一个新结点 v'，从而将原来的边 (v_1, v_2) 分成 (v_1, v') 和 (v', v_2) 两条，并置：

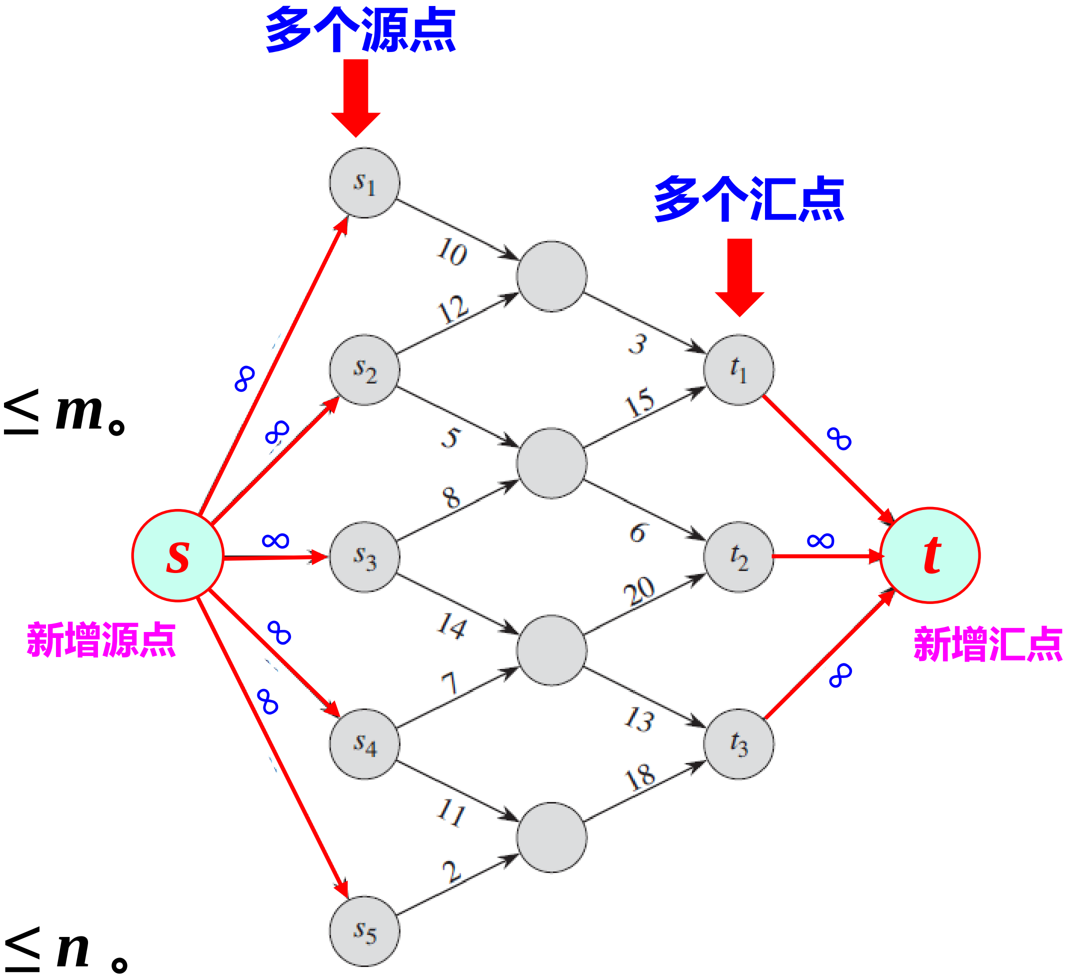
$$c(v_1, v') = c(v', v_2) = c(v_1, v_2)$$



(2) 若有多个源点或多个汇点

- **对多个源点**
  - 加入一个**超级源点** s，并加入有向边 (s, s_i)，然后令 $c(s, s_i) = \infty,\ 1 \le i \le m$。

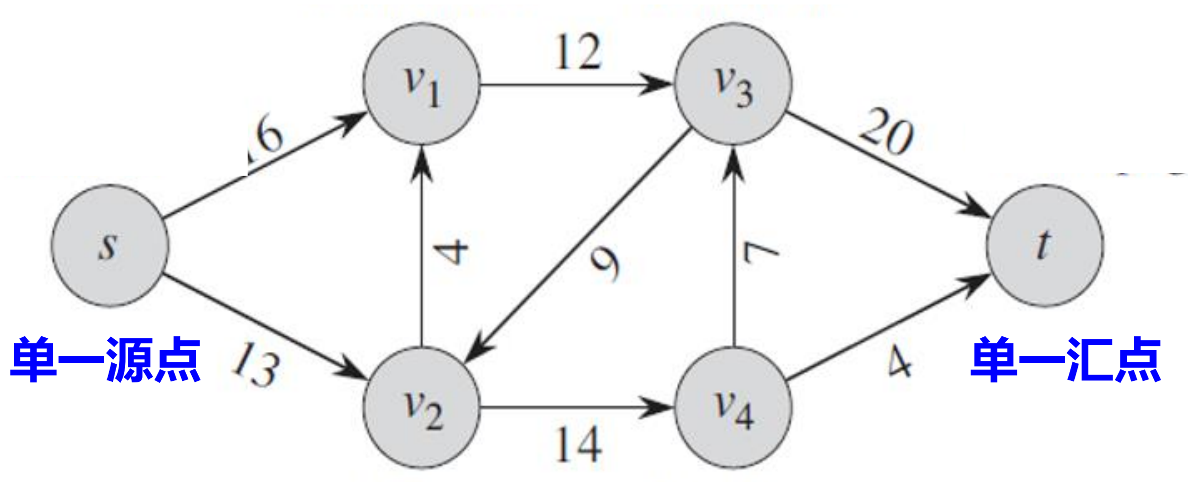
- **对多个汇点**
  - 加入一个**超级汇点** t，并加入有向边 (t_i, t)，然后令 $c(t_i, t) = \infty,\ 1 \le i \le n$ 。



可以证明：转化前后的两个问题是等价的，即具有相同的流。

- **不失一般性，本章只讨论标准流网络**

一个（标准）流网络 *G* 示例如下：



***G* 的容量函数定义**

$$c(s,v_1)=16$$
$$c(s,v_2)=13$$
$$c(v_1,v_3)=12$$
$$c(v_2,v_1)=4$$
$$c(v_2,v_4)=14$$
$$c(v_3,v_2)=9$$
$$c(v_4,v_3)=7$$
$$c(v_3,t)=20$$
$$c(v_4,t)=4$$

设 $G=(V,E)$ 是一个流网络，s 为源结点，t 为汇点。定义在 G 上的**容量函数**为：$c: E \to R^{+} \cup \{0\}$

1、流的概念

流是 G 中**边上**的一个**实值函数映射**，记为 $f: V \times V \to R$，并满足以下性质：

（1）**容量限制:** 对于所有的结点 $u, v \in V$，有

$$0 \le f(u,v) \le c(u,v)$$

若 $(u,v) \notin E$，记 $f(u,v)=0$

（2）**流量守恒**：对于所有结点 $u \in V-\{s,t\}$，有

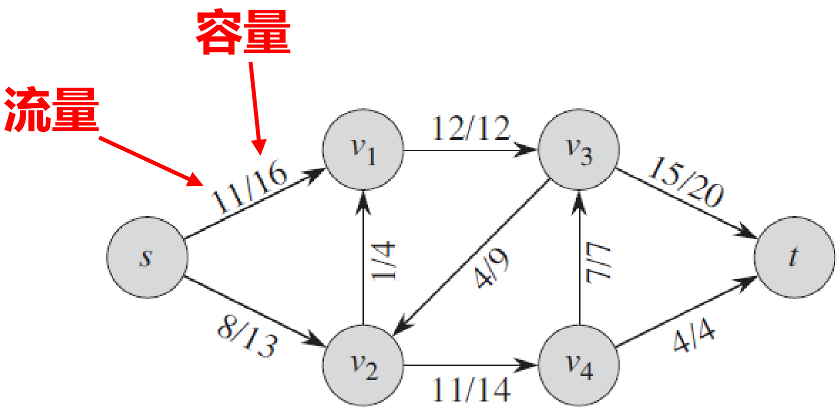
$$\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$$

即：任何结点流入的流量等于流出的流量

◆ 流的值

一个流网络 G 上的**流 f 的值**是**流出源点 s 的总流量**减去**流入源点 s 的总流量**，即**源点的净流出量**，用 $|f|$ 表示：

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



一个带有流 f 的流网络 G

G 中各条边上的流：

$f(s, v_1) = 11$， $f(s, v_2) = 8$，

$f(v_1, v_3) = 12$， $f(v_2, v_1) = 1$，

$f(v_2, v_4) = 11$， $f(v_3, v_2) = 4$，

$f(v_4, v_3) = 7$， $f(v_3, t) = 15$，

$f(v_4, t) = 4$

f 的流值： $$|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) = 19$$

2、最大流问题:

在一个给定的流网络 G 中找一个**流值最大的流。**

- **求流网络最大流的基本思路:**

从**最小流值**开始，**一点一点地增加**，**直到最大值**。

需要解决以下问题:

（1）开始的时候，最小流值定义为多少?

（2）如何一点一点地增加流值?

（3）何时结束，即怎么判断得到了最大流?

其它约定：

（1）除源点和汇点外，其它结点上物料只是**流过**， 即**物料进入的速率等于离开的速率**（即满足**流量守恒**，**不储存**）；

（2）源点物料的生成速率和其它结点物料的接收速率**恒定**，且足够快、足够多，满足相关需要；

（3）**每条边上的容量是物料通过该边的最大速率**，不能突破（即满足**容量限制**）。

26.2 Ford-Fulkerson方法

1. 基本思想

- **通过不断增加可行流值的方式找到最大流**

（1）从**最小流值**为 **0** 的初始流开始；

（2）通过**某种方法**，对流值进行增加；

（3）当确认**无法再增加流值**时，就得到最大流；

1955年由Lester R. Ford, Jr. 和 Delbert R. Fulkerson提出

Ford-Fulkerson方法的要点：

（1）提出**残存网络** G_f (residual network) 和 **增广路径** p (augmenting path) 。

（2）用**增广路径**对路径上边的流量进行修改，以**增加流网络的流量**。

（3）判断是否得到最大流的条件是**最大流最小切割定理**。

- 该定理给出了**算法的终止条件，并证明算法终止时可以获得一个最大流**。

Ford-Fulkerson方法的过程描述

FORD-FULFERSON-MENTHOD(G, s, t){

1 initialize flow f to 0

2. **while** there exists an augmenting path p in the residual network G_f

3. augment flow f along p

4. **return** f

2、残存网络（Residual Network）

对给定的流网络 G 和流 f，G 的**残存网络**记为 $G_f = (V, E_f)$。

G_f 也是一个**有向图**，由 G 中的**结点集** V 和以下的边组成的边集 E_f 组成，并在边上定义**残存容量** $c_f(u,v)$ *for all* $(u,v) \in Ef$ 。

- **残存网络的构造**：对于 G 中的一条任意边 (u,v)：

1）若 $f(u, v) < c(u, v)$ ，则将 (u, v) 和它的**反向边** (v, u) 都加入 G_f，并记 (u, v) 和 (v, u) 的**残存容量**为：

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

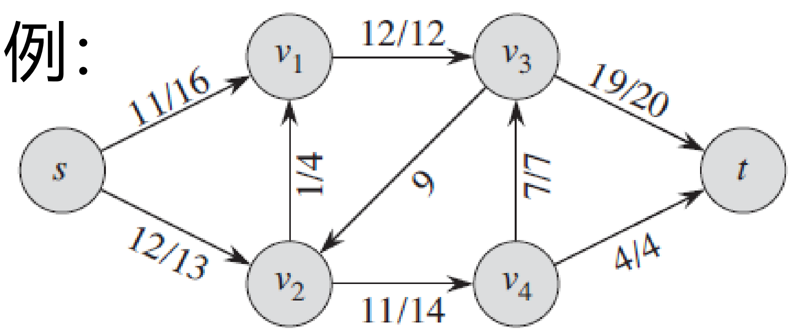
$$c_f(v,u) = f(u,v)$$

2）如果 $f(u, v) = c(u, v)$，则 (u, v) 不加入 G_f，但其**反向边** (v, u) 加入G_f，并置 $c_f(v, u) = f(u, v)$。

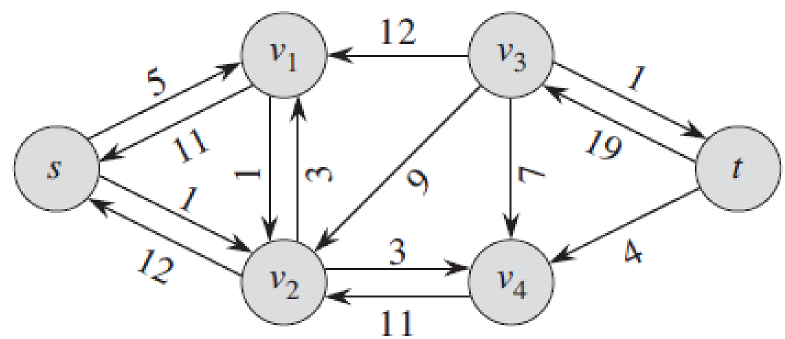
（注：由于(u, v) 不加入 G_f，所以也可视为 $c_f(u, v) = 0$ ）

3）如果 $f(u, v) = 0$，则 (u, v) 加入 G_f，其**反向边** (v, u) 不加入G_f，并置 $c_f(u, v) = f(u, v)$。（同理，由于 (v, u) 不加入 G_f，所以可视为 $c_f(v, u) = 0$ ）

例:



一个带有流 f 的流网络 G



G 的残存网络 G_f

- (v_1, v_3) 是“满”流量的边
- (v_3, v_2) 是没有流量的边

- (v_1, v_3) 不加入G_f，反向边 (v_3, v_1) 加入G_f
- (v_2, v_3) 加入 G_f，反向边 (v_3, v_2) 不加入G_f

残存网络： 设流网络 $G = (V, E)$，f 是 G 中的一个流，由 f 所**诱导**的图 G 的残存网络记为 G_f:

仅残存容量大于0的边

$$G_f = (V, E_f)$$

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\} \text{ 且 } |E_f| \le 2|E|$$

G_f的边的残存容量 $c_f: V \times V \to R$ 定义为:

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

- 边 (u, v) 的残存容量 $c_f(u, v)$ 反映了(u, v)上可以增加的流量的空间。
- 边 (v, u) 的残存容量 $c_f(v, u)$ 反映了如果从边 (v, u) **反向流回去**可“**回流**”的最大流量。

3、流量的递增（augmentation）

设 $G=(V,E)$ 为一个流网络，源结点为 s，汇点为 t，f 为 G 中的一个流。设 G_f 为由流 f 所诱导的 G 的残存网络。

将 G_f 视为一个**“非标准的流网络”**(存在反向边)。记 f' 为 G_f 上的一个流，这意味着 f' **在 G_f 中满足容量限制和流量守恒**。

(如何找 f' 见后)

定义 $f\uparrow f'$ 为用流 f' 对流 f 递增后得到的一个**“新”流**：

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

理解：$f'(u,v)$ 可视为 (u,v) 边上流量的**正向增加**，而 $f'(v,u)$ 可视为 (u,v) 边上流量因“**回流**”造成的**反向减少**。

引理26.1 设 $G=(V,E)$ 为一个流网络，源点为 s，汇点为 t。设 f 为 G 中的一个流。设 G_f 为由流 f 所诱导的 G 的残存网络，设 f' 为 G_f 中的一个流。那么 $f\uparrow f'$ **是 G 的一个流**，其值为：

$$|f\uparrow f'| = |f| + |f'|$$

证明：思考：如何才能称为一个流?

（1）**满足容量限制**，即 $|f\uparrow f'|(u,v)\le c(u,v)$；

（2）**满足流量守恒**，即 对所有的 $u\in V-\{s,t\}$，有

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

证明

证明1： 证明$f\uparrow f'$ 是图G的一个流，即满足**容量限制**和**流量守恒.**

(1) 对容量限制的证明

根据定义，如果边 $(u, v)\in E$，则根据 G_f 的定义，(u, v) 的 **反向边** (v, u) 有：$c_f(v, u) = f(u, v)$ ，且因为 f' 是 G_f的一个流，所以流量 $f'(v, u) \le c_f(v, u) = f(u, v)$ 。

因此，

$$
\begin{aligned}
(f\uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \quad \text{（根据定义）}\\
&\ge f(u, v) + f'(u, v) - f(u, v)\\
&= f'(u, v)
\end{aligned}
$$

即，流$f\uparrow f'$ 在每条边上的流量**都不为负**

再有，

$$
\begin{aligned}
(f\uparrow f')(u,v) &= \boxed{f(u,v)+f'(u,v)} - f'(v,u) && \text{（根据定义）}\\
&\le \boxed{f(u,v)+f'(u,v)} \\
&\le f(u,v)+c_f(u,v) && \text{（ } f' \text{ 的容量限制）}\\
&= f(u,v)+c(u,v)-f(u,v) && \text{（ } c_f \text{ 的定义）}\\
&= c(u,v)
\end{aligned}
$$

所以递增后，**流 $f\uparrow f'$ 满足容量限制**。

(2) 对流量守恒的证明

对所有的 $u \in V - \{s, t\}$，有：

$$
\begin{aligned}
\sum_{v \in V} (f \uparrow f')(u, v) &= \sum_{v \in V} (f(u, v) + f'(u, v) - f'(v, u)) \\
&= \sum_{v \in V} f(u, v) + \sum_{v \in V} f'(u, v) - \sum_{v \in V} f'(v, u) \\
&= \sum_{v \in V} f(v, u) + \sum_{v \in V} f'(v, u) - \sum_{v \in V} f'(u, v) \\
&= \sum_{v \in V} (f(v, u) + f'(v, u) - f'(u, v)) \\
&= \sum_{v \in V} (f \uparrow f')(v, u) ,
\end{aligned}
$$

f 和 f' 都满足流量守恒

即 流 $f \uparrow f'$ 满足流量守恒

证明2：证明流值 $|f \uparrow f'| = |f| + |f'|$

定义 $V_1 = \{ v: (s, v) \in E \}$ 所有s有边能到达的结点

$V_2 = \{ v: (v, s) \in E \}$ 所有有边能到达s的结点

由于**流网络** G **中没有反向边**，所以任意 (s, v) 和 (v, s) 不可能同时存在于 G 中，则：

$$V_1 \cap V_2 = \emptyset \text{ 且 } V_1 \cup V_2 \subseteq V。$$

则有，

$$\begin{aligned} |f \uparrow f'| &= \sum_{v \in V} (f \uparrow f')(s, v) - \sum_{v \in V} (f \uparrow f')(v, s) \quad \text{(根据流值的定义)} \\ &= \sum_{v \in V_1} (f \uparrow f')(s, v) - \sum_{v \in V_2} (f \uparrow f')(v, s) , \end{aligned}$$

$$|f \uparrow f'|$$

$$= \sum_{v \in V_1} (f \uparrow f')(s,v) - \sum_{v \in V_2} (f \uparrow f')(v,s)$$

(根据 $|f \uparrow f'|$ 的定义展开)

$$= \sum_{v \in V_1} \underline{(f(s,v) + f'(s,v) - f'(v,s))} - \sum_{v \in V_2} \underline{(f(v,s) + f'(v,s) - f'(s,v))}$$

$$= \sum_{v \in V_1} f(s,v) + \sum_{v \in V_1} f'(s,v) - \sum_{v \in V_1} f'(v,s) - \sum_{v \in V_2} f(v,s) - \sum_{v \in V_2} f'(v,s) + \sum_{v \in V_2} f'(s,v)$$

$$= \sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) + \sum_{v \in V_1} f'(s,v) + \sum_{v \in V_2} f'(s,v) - \sum_{v \in V_1} f'(v,s) - \sum_{v \in V_2} f'(v,s)$$

$$= \sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) + \sum_{v \in V_1 \cup V_2} f'(s,v) - \sum_{v \in V_1 \cup V_2} f'(v,s)$$

$$= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{v \in V} f'(s,v) - \sum_{v \in V} f'(v,s)$$

$$= |f| + |f'|$$

(根据流值的定义)

注：$V_1 \cup V_2 \subseteq V$

$V_1 \cap V_2 = \emptyset$

证毕

下一个问题：在 G_f 中怎么找流 f' 来实现对流 f 的递增?

4、增广路径

对给定的流网络 $G=(V, E)$ 和 G 上的一个流 f，由 f 所诱导的**增广路径** p (*Augmenting Path*) 是**残存网络** G_f **中一条从源结点** s **到汇点** t **的简单路径。**

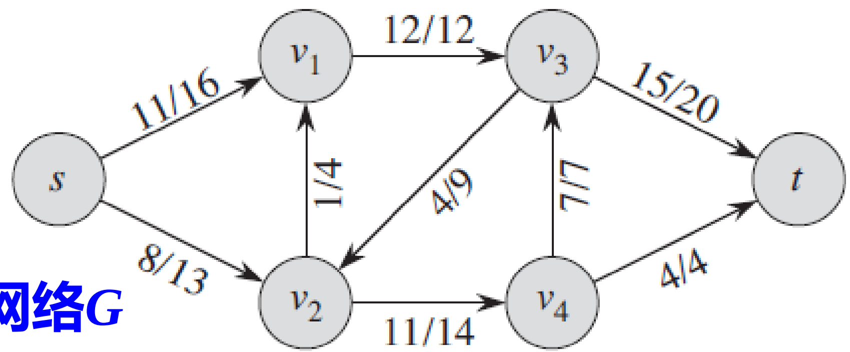
- 对于增广路径上的**一条边** (u, v)，其可用于"**扩增**"容量的**最大值**为该边的**残存容量** $c_f(u, v)$ 。
- **而对于整条增广路径** p，其能用于**扩增流** f 的**最大流值**，称为该路径的**残存容量**，记为 $c_f(p)$：

$$c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is on } p\}$$

$$> 0$$

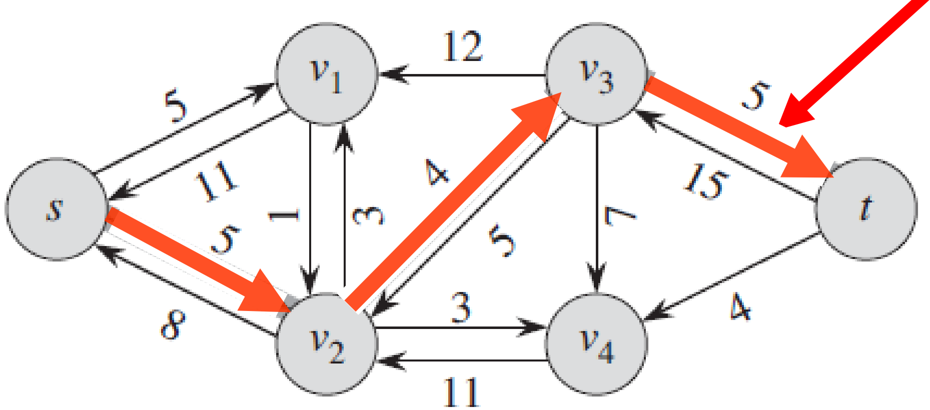
p 上所有边残存容量的最小者

示例:



流网络G

残存网络G_f



红色路径是一条增广路径

$p = \{s, v_2, v_3, t\}$

$c_f(s, v_2) = 5$

$c_f(v_2, v_3) = 4$

$c_f(v_3, t) = 5$

则增广路径 p 的残存容量 $c_f(p)$ 是:

$$c_f(p) = \min\{c_f(s, v_2), c_f(v_2, v_3), c_f(v_3, t)\}$$
$$= 4$$

引理26.2 设$G=(V,E)$为一个流网络，f是图G的一个流。

记 p **为残存网络** G_f **中的一条增广路径。**

定义一个函数 $f_p: V \times V \to R$ 如下：

$$f_p(u,v) = \begin{cases} c_f(p) & if (u,v) \text{ is on } p \\ 0 & otherwise \end{cases}$$

则 f_p **是残存网络** G_f **中的一个流**，其值为 $|f_p| = c_f(p) > 0$。

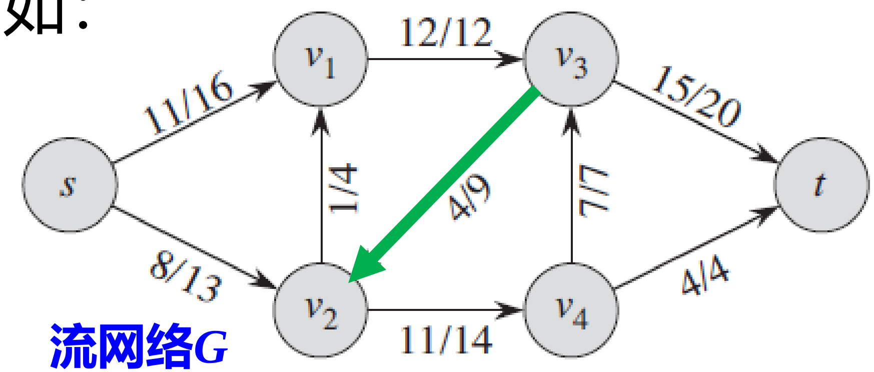
证明：略（参见26.2-7）。

流 f_p **有大于 0 的递增流量**

找到了G_f中的一个流： f_p

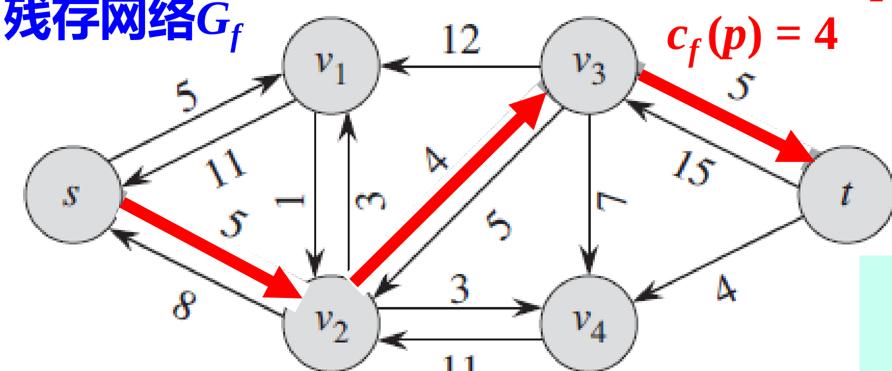
则：**用 f_p 递增流 f**，根据引理26.1，$f \uparrow f_p$ 仍是 G 的一个流，且其流值 $|f|+|f_p|>|f|$，从而更加接近最大值。

如：



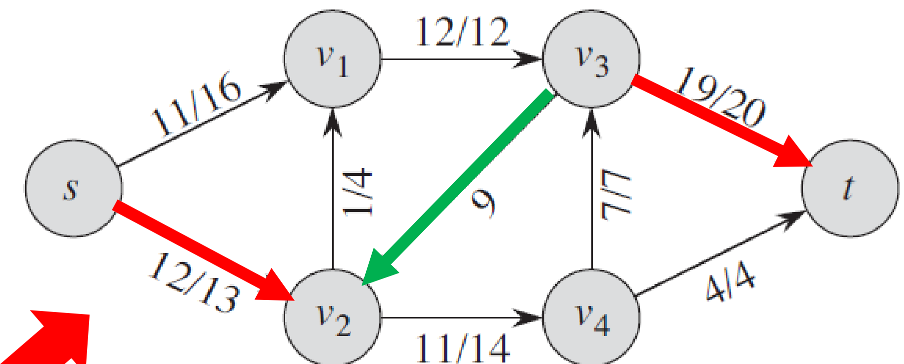
流网络G

残存网络G_f



增广路径 p

$c_f(p)=4$



用 f_p 增加 f 的流量后的流网络

注：边 (v_3, v_2) 原来有 v_3 到 v_2 的流量，但递增后，该流量从 v_2 “**流回**”了 v_3，所以 G 中**边 (v_3, v_2) 在递增后没有流量了**。

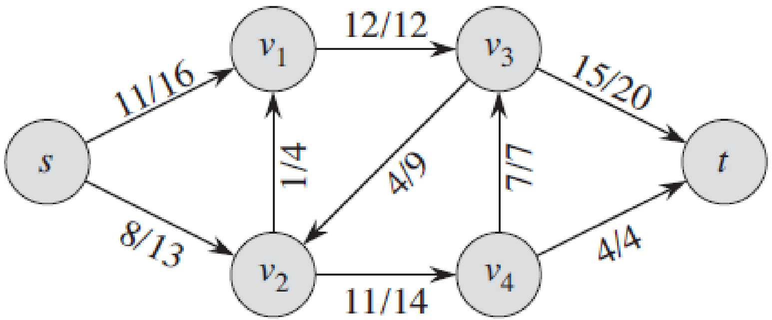
推论26.3：设 $G = (V, E)$ 为一个流网络，f 是图 G 的一个流，p 为残存网络 G_f 中的一条增广路径。

设 f_p 是上面定义的残存网络的流。用 f_p 增加 f 的流量，则函数 $f \uparrow f_p$ 是图 G 的一个流，其值为：

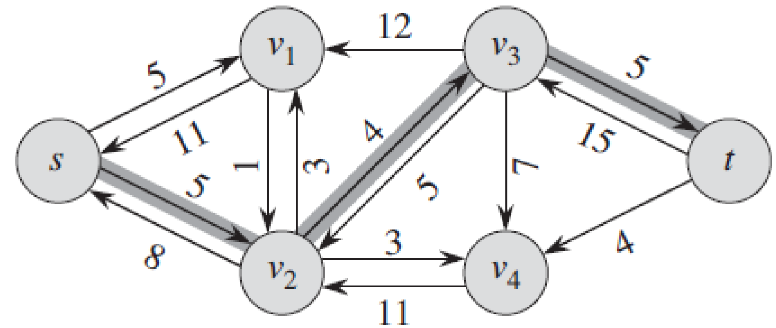
$$|f \uparrow f_p| = |f| + |f_p| > |f|。$$

证明：根据**引理**26.1和**引理**26.2可得证。

◆ **利用残存网络和增广路径实现 *Ford-Fulkerson* 方法：**

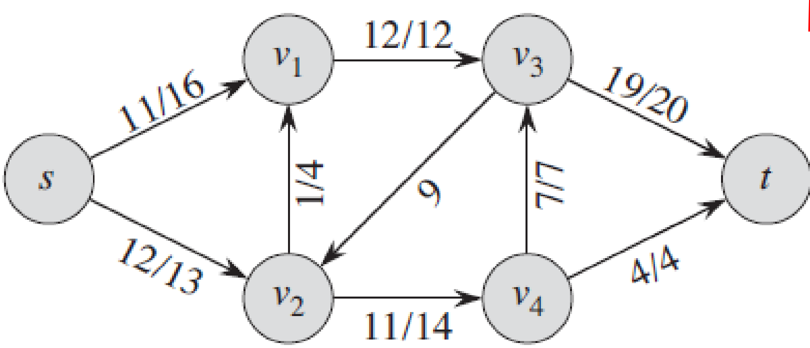


流网络 *G*，当前有一个流 *f*

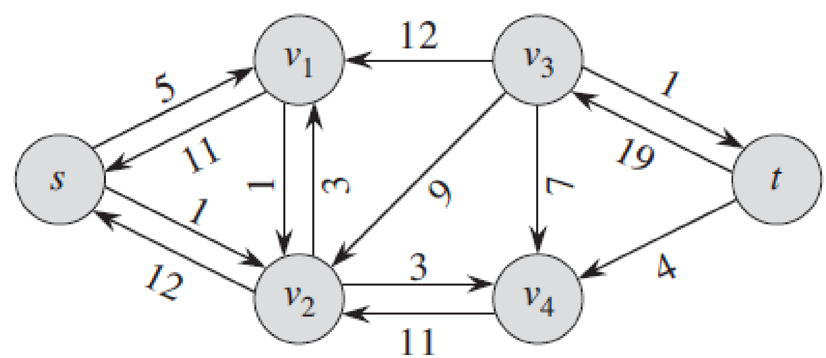


构造残存网络，找到一条增广路径

用增广路径递增流值



第一次递增后的流网络 *G*，流变大了



再构造残存网络，再找下一条增广路径，再次递增 *G* 中的流 ……

下一个问题：递增过程何时结束?

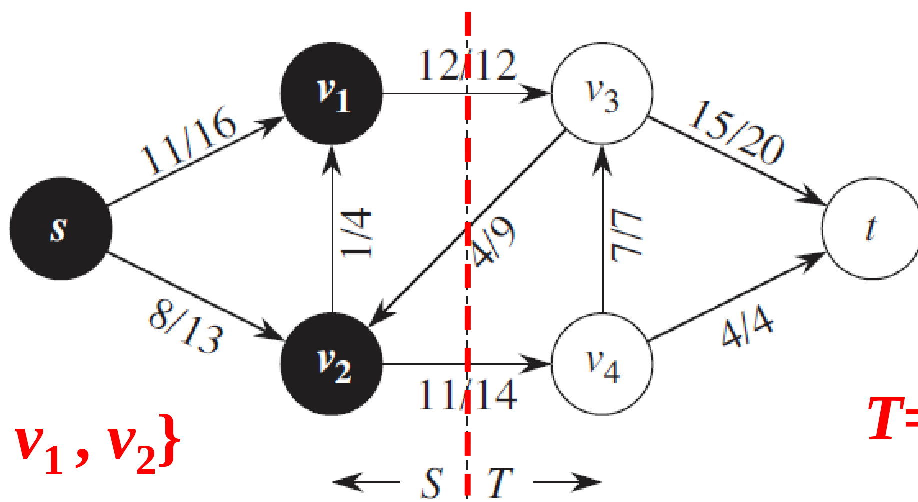
最大流最小切割定理

5、最大流最小切割定理

（1）切割

图 $G = (V, E)$ 的一个**切割** (S, T) 是结点集 V 的一个划分，使得 $S \subseteq V$，$T \subseteq V$，$S \cap T=\Phi$，$S \cup T=V$，即 $T = V - S$。

如：



$S =\{s, v_1, v_2\}$

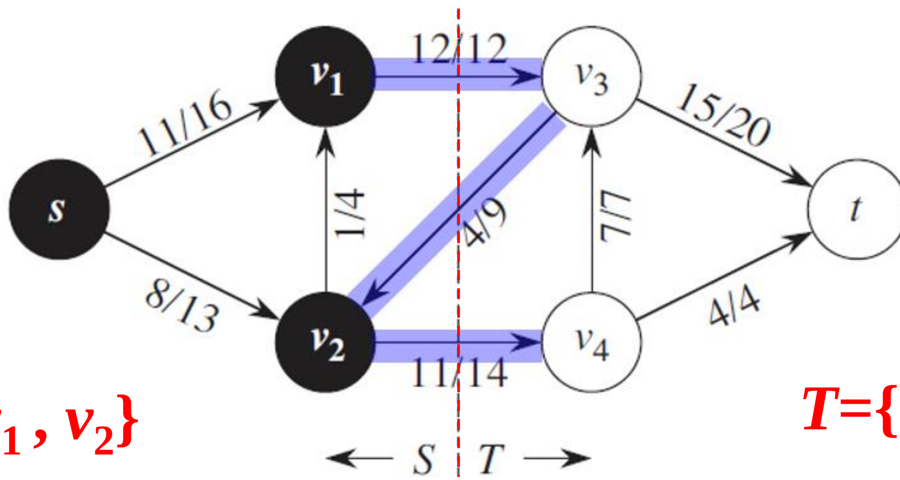
$T=\{v_3, v_4, t\}$

一个图 G 及 G 上的一个切割

（2）横跨切割：

对于图 $G=(V, E)$ 及 G 的一个切割 (S, T)，如果一条边 $(u, v)\in E$ 的一个端点在集合 S 中，而另一个端点在集合 T 中，则称该边**横跨切割 (S, T)**。

如：



$S=\{s, v_1, v_2\}$　　　　$T=\{v_3, v_4, t\}$

一个图 G 及 G 上的一个切割

(v_1, v_3)、(v_3, v_2)、(v_2, v_4) **是横跨切割 (S, T) 的边。**

（3）流和切割

给定流网络 $G = (V, E)$，源结点为 s，汇点为 t。定义 G 的一个**切割** (S, T)，将结点集合 V 分成 S 和 $T = V - S$ 两部分，并使得 $s \in S$，$t \in T$。设 f 是 G 上的一个流，则

（a）**流 f 横跨切割 (S, T)**

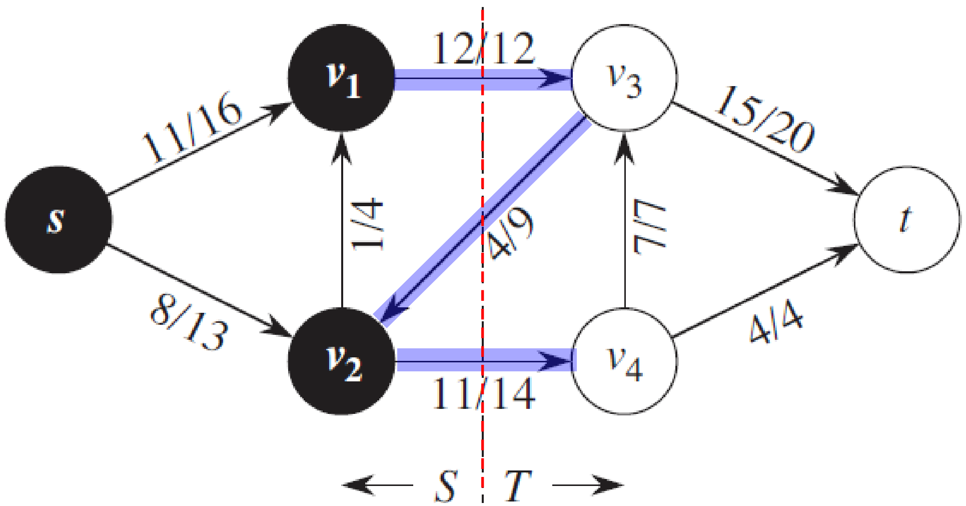
（b）**流 f 横跨切割 (S, T) 的净流量 $f(S, T)$** 为：

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

- **再定义切割 (S, T) 的容量**为：$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$

即从 S 到 T 的**横跨切割**的边的容量之和

例，设一个流网络 G 和流 f，以及 G 上的一个切割 (S, T)如下：



则，

切割 (S, T) 是：$S = \{s, v_1, v_2\}$，$T = \{v_3, v_4, t\}$

横跨切割 (S, T) 的**净流量** $f(S, T) = f(v_1, v_3) + f(v_2, v_4) - f(v_3, v_2)$

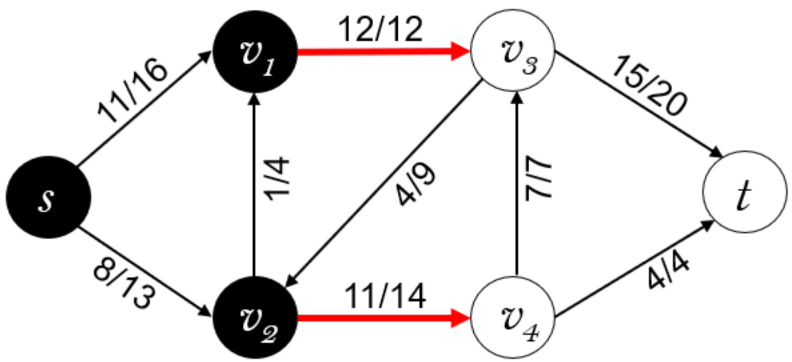
$= 12 + 11 – 4 = 19$

切割 (S, T) 的**容量** $c(S, T) = c(v_1, v_3) + c(v_2, v_4)$

$= 12 + 14 = 26$

(4) 最小切割

不同的切割，结点集的划分和横跨切割的边集都不同，横跨切割的容量也就可能不同。将网络中**容量最小的切割**称为**最小切割**。

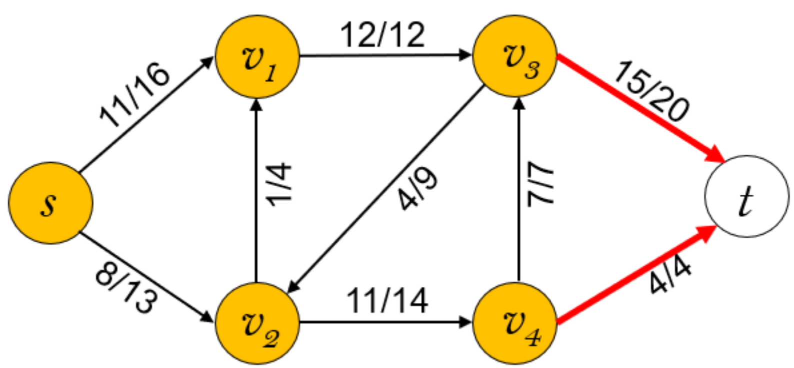


切割1：

S_1 ={s, v_1, v_2 }， T_1 ={v_3, v_4 , t }

$c\ (S_1, T_1) = c\ (v_1, v_3) + c\ (v_2, v_4)$

$=12+14=$**26**



切割2：

S_2={s, v_1, v_2, v_3, v_4, }， T_2={ t }

$c\ (S_2 ,T_2) = c\ (v_3, t)+c\ (v_4, t)$

$=20+4=$**24**

引理26.4 设 f 为流网络 $G = (V, E)$ 的一个流，设该流网络的源点为 s，汇点为 t，再设 (S, T) 为流网络 G 如上的任意一个切割，则**横跨切割 (S, T) 的净流量**为 $f(S, T) = |f|$。

证明：根据**流量守恒**，对任意结点 $u \in V - \{s, t\}$ 都有：

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

所以对 $S - \{s\}$ 中所有结点，它们的总流出量也必等于总流入量，即：

$$\sum_{u \in S - \{s\}} \left(\sum_{v \in V} f(u, v) - \sum_{v \in V} f(v, u) \right) = 0$$

而流 f 的**流值**定义是 $|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$，则进一步推导有：

$$|f| = \sum_{v\in V} f(s,v) - \sum_{v\in V} f(v,s) + \underbrace{\sum_{u\in S-\{s\}} \left(\sum_{v\in V} f(u,v) - \sum_{v\in V} f(v,u) \right)}_{=0}$$

$$= \sum_{v\in V} f(s,v) - \sum_{v\in V} f(v,s) + \sum_{u\in S-\{s\}} \sum_{v\in V} f(u,v) - \sum_{u\in S-\{s\}} \sum_{v\in V} f(v,u)$$

$$= \sum_{v\in V} \left(f(s,v) + \sum_{u\in S-\{s\}} f(u,v) \right) - \sum_{v\in V} \left(f(v,s) + \sum_{u\in S-\{s\}} f(v,u) \right)$$

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

将 V 分成 S 和 T 单独处理

$S \cap T = \Phi$，$S \cup T = V$

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

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

进一步整理：

$$
\begin{aligned}
|f| &= \sum_{v\in S}\sum_{u\in S} f(u,v) + \sum_{v\in T}\sum_{u\in S} f(u,v) \\
&\quad - \sum_{v\in S}\sum_{u\in S} f(v,u) - \sum_{v\in T}\sum_{u\in S} f(v,u) \\
&= \sum_{v\in T}\sum_{u\in S} f(u,v) - \sum_{v\in T}\sum_{u\in S} f(v,u) + \underline{\left(\sum_{v\in S}\sum_{u\in S} f(u,v) - \sum_{v\in S}\sum_{u\in S} f(v,u)\right)} \\
&= \sum_{u\in S}\sum_{v\in T} f(u,v) - \sum_{u\in S}\sum_{v\in T} f(v,u) \\
&= f(S,T)
\end{aligned}
$$

两项均包含了 S 中的所有结点对，最终**相互抵消**，**等于0**。

证毕，即**横跨（任何）切割的净流量**都等于**流的值 $|f|$**。

推论26.5： 流网络 G 中任意**流 f 的值不超过** G **的任意切割的容量**。

证明： 设 (S, T) 为流网络 G 的任意切割，设 f 为 G 中的任意流。

$$|f| = f(S,T) \quad \text{(引理26.4)}$$

$$= \boxed{\sum_{u\in S}\sum_{v\in T} f(u,v)} - \sum_{u\in S}\sum_{v\in T} f(v,u) \quad \text{(根据流横跨切割 }(S, T)\text{的净流量 } f(S, T)\text{ 的定义)}$$

$$\le \boxed{\sum_{u\in S}\sum_{v\in T} f(u,v)}$$

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

$$= c(S,T) \quad \text{(根据切割容量的定义)}$$

证毕

(6）最大流最小切割定理

定理 26.6：设 f 为流网络 $G=(V, E)$ 中的一个流，该流网络的源点为 s，汇点为 t，则下面的条件是等价的：

（1） **f 是 G 的一个最大流；**

（2）**残存网络 G_f 不包括任何增广路径；**

（3）$|f| = c(S, T)$，**其中 (S, T) 是 G 的某个切割。**

最大流最小切割定理说明，一个流网络 G 的**流 f 在流值等于某个切割的容量时达到最大**，此时在对应的**残存网络 G_f 中不再有增广路径**。

定理的证明

证明思路：为了证明上述**3条等价**，只需依次证明 **由(1)可得(2)**、**由(2)可得(3)** 以及 **由(3)可得(1)**。如果得证，则说明三者可以相互导出，即等价。

(1) f 是 G 的一个最大流；
(2) 残存网络 G_f 不包括任何增广路径；
(3) $\|f\| = c(S, T)$，其中 (S, T) 是 G 的某个切割。

证明 由(1)可导出(2)：**反证法**

假定 f 是 G 的一个最大流，但残存网络 G_f 同时**包含一条增广路径** p。设 f_p 是该增广路径 p 上定义的流，$|f_p| > 0$。

根据**推论26.3**，用 f_p 递增 f 形成的“新流” $f \uparrow f_p$ 将是 G 中的一个流，且流值严格大于 $|f|$，即 $|f \uparrow f_p| = |f| + |f_p| > |f|$。

这与 f 是最大流冲突。所以 G_f **中不可能再包含任何增广路径。**

(1) f 是 G 的一个最大流；

(2) 残存网络 G_f 不包括任何增广路径；

(3) $|f| = c(S, T)$，其中 (S, T) 是 G 的某个切割。

证明 由(2)可导出(3)

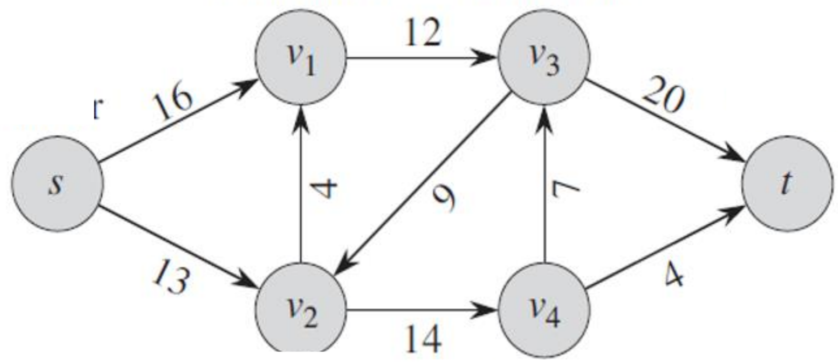
因为 G_f 中不包含任何增广路径，所以 **G_f 中不存在从源点 s 到汇点 t 的路径**。

定义这样一个**切割 (S, T)** 如下

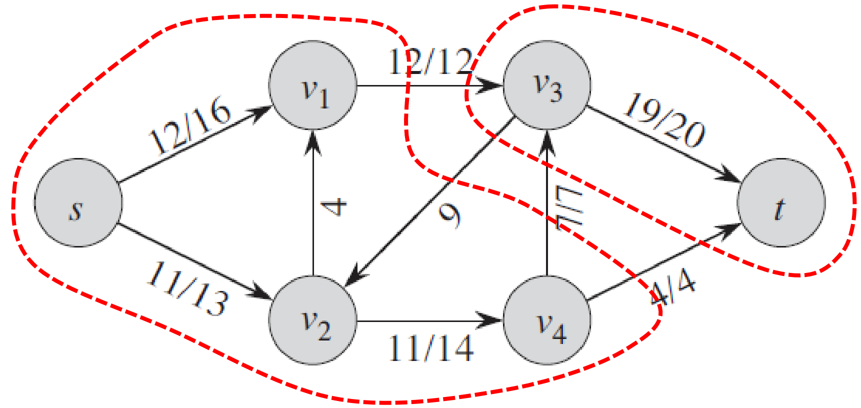
$$S = \{v \in V: \text{在} G_f \text{中存在一条从} s \text{到} v \text{的路径}\},$$

$$T = V - S.$$

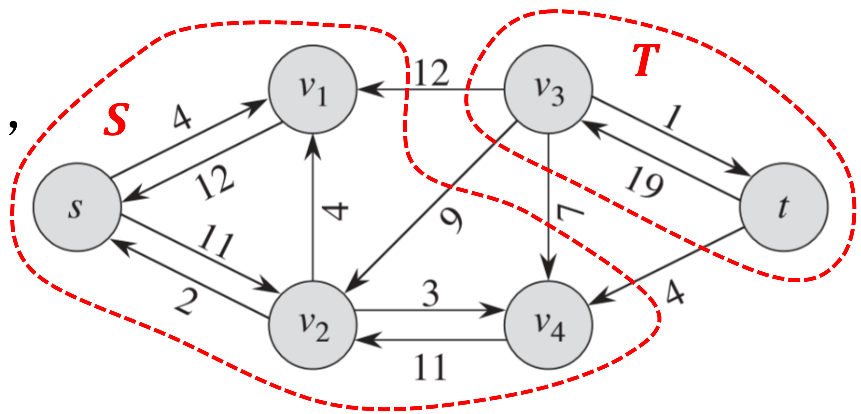
原始流网络



最大流时的流网络



最大流的流网络对应的残存网络



定义切割 (S, T)：

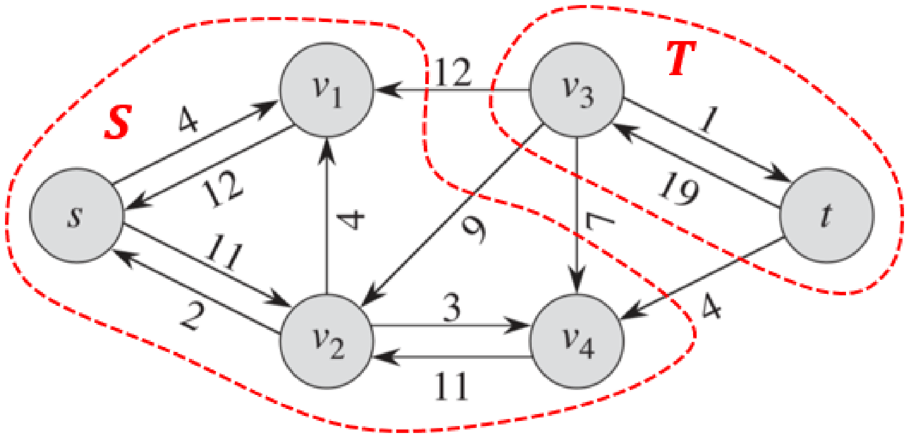
$S = \{v \in V$: 在G_f中存在一条从s到v的路径$\}$,

$T = V - S$

无增广路径：G_f中不存在从 s 到 t 的**有向**路径。

无增广路径

最大流的流网络对应的残存网络



(1) f 是 G 的一个最大流;

(2) 残存网络 G_f 不包括任何增广路径;

(3) $|f| = c(S, T)$，其中 (S, T) 是 G 的某个切割。

现证明：对于**某个切割**，**包括对上述** (S, T)，**将有** $|f| = c(S, T)$。

根据以上切割 (S, T) 的设定，显然流 f 是横跨该切割的流。

根据**引理26.4**，**流值** $|f|$ 等于横跨切割 (S, T) 的净流量 $f(S, T)$，

即：$|f| = f(S, T)$。

所以为证 $|f| = c(S, T)$，只需要证明：$f(S, T) = c(S, T)$。

即：当 G_f **中不存在增广路径**时，流 f 横跨切割 (S, T) 的净流量 $f(S, T)$ 等于该切割的容量 $c(S, T)$。

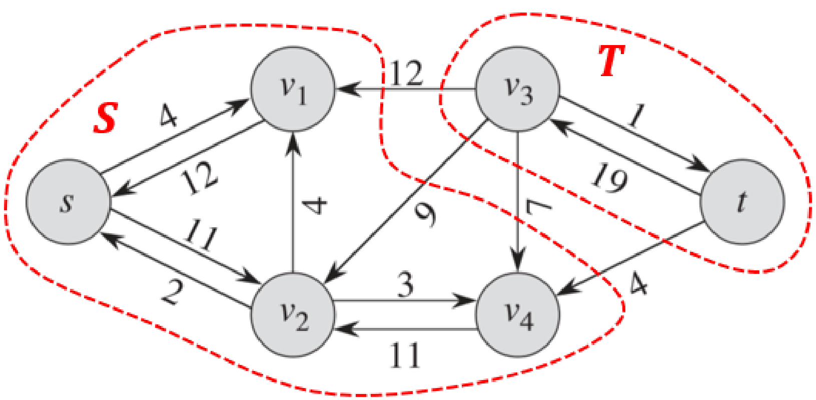
考虑任意一对结点 $u \in S, v \in T$，有两种情况：

情况1：**若** $(u, v) \in E$，**则必有** $f(u, v) = c(u, v)$ 。

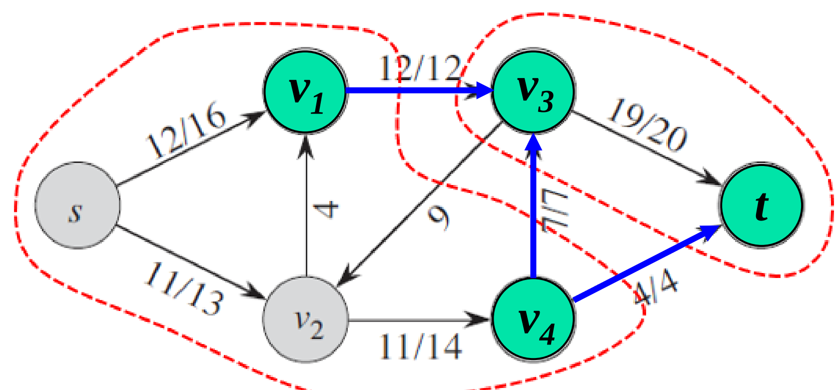
否则，就有 $f(u, v) < c(u, v)$ 。而如此的话，(u, v)上就会**有残存容量**（即 $c_f(u, v) = c(u, v) - f(u, v) > 0$ ），而因有残存容量，所以就有 $(u, v) \in E_f$。那么在 G_f 中就可以**从 s 经过 u 而到达 v**，这与 $v \in T$ 相矛盾（S 集合中的结点都是s可达的结点，但 T 集合的结点都是 s 不可达的。前提 $v \in T$ 已约定 v 是 s 不可达的结点，现在又推出可达，所以相矛盾)。

另外，若 $(u, v) \notin E$ **，则** $f(u, v) = 0$ 。

最大流的流网络对应的残存网络



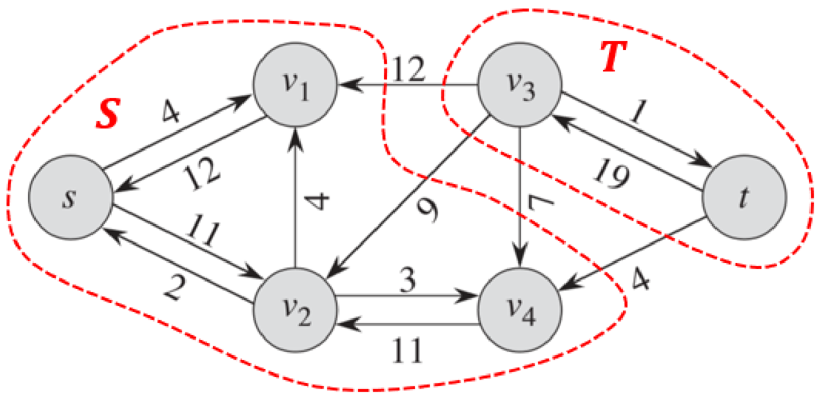
最大流时的流网络



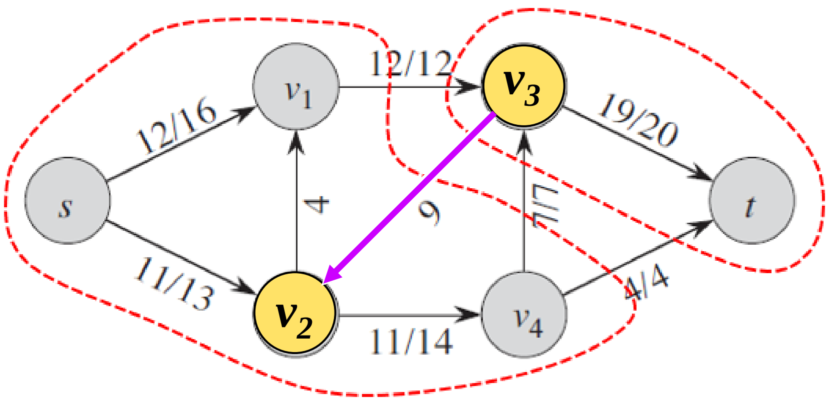
$u \in S, v \in T$

情况2：若有 $(v,u) \in E$，则必有 $f(v,u) = 0$。

最大流的流网络对应的残存网络



最大流时的流网络



否则，就有 $f(v,u) > 0$。如果这样的话，在 G_f 中就存在它的**反向边** (u,v)，即 $(u,v) \in E_f$，这样就推出与情况1相同的矛盾。

另外，若 $(v,u) \notin E$，则 $f(v,u) = 0$。

综上，可以得到

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

情况1

情况2，这样的 (v,u) 或者$\notin E$
或者$f(v,u)=0$

$$=\sum_{u\in S}\sum_{v\in T}c(u,v)-\sum_{v\in T}\sum_{u\in S}0$$

$$=c(S,T)$$

即当 G_f 中不包含增广路径时，横跨切割的**净流量**就恰好等于**切割的容量**

$|f| = f(S, T) = c(S, T)$

证毕

(1) f 是 G 的一个最大流；

(2) 残存网络 G_f 不包括任何增广路径；

(3) $|f| = c(S, T)$，其中 (S, T) 是 G 的某个切割。

证明 由(3)可导出(1)

根据推论26.5（**任何流 f 的值不超过任意切割的容量**），**对于所有切割 (S,T)，$|f| \le c(S, T)$**，因此，$|f| = c(S, T)$ 意味着 f 是一个最大流。

同时也根据推论26.5：**流 f 的值不超过任意切割的容量，所以，这样的 (S, T) 将是 G 的一个最小切割。**

定理证毕

◆ 回到Ford-Fulkerson方法

FORD-FULFERSON-MENTHOD(G, s, t){

1 initialize flow f to 0

2. **while** there exists an augmenting path p in the residual network G_f

3. augment flow f along p

4. **return** f

如何具体实现最大流的求解?

- **基本思想**：通过**不断增加可行流的流值**找最大流
- **技术路线**：

（1）从流值为0的初始流开始

（2）**对流值进行增加** ➡ **在 G_f 中找增广路径**

（3）**确认无法增加流值，即得到最大流。** ➡ **G_f 不再包括增广路径**

6、Ford-Fulkerson算法的细化

FORD-FULFERSON(G, s, t){

for each edge $(u,v) \in G.E$

$(u, v).f = 0$ 初始流值设为0

找一条增广路径

while there exists a path p from s to t in the residual network G_f

$c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$

求增广路径的残存容量$c_f(p)$

for each edge (u, v) in p

if $(u, v) \in E$

$(u, v).f = (u, v).f + c_f(p)$

若$(u, v) \in E$，则用 $c_f(p)$ 扩增(u, v)上的流量

else

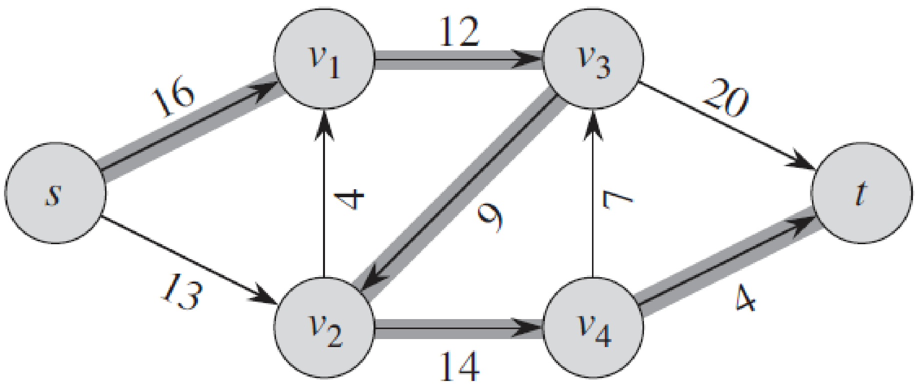
$(v, u).f = (v, u).f - c_f(p)$

若 $(u, v) \notin E$ ，其反向边 $(v, u) \in E$ 且(v, u)上是**满流量**，此时用 $c_f(p)$ 减少(v, u) 上的流（**回流**）。

如何找一条增广路径？*BFS* 或者 *DFS*

（1）基本的Ford-Fulkerson算法

算法运行示例：Iteration 1



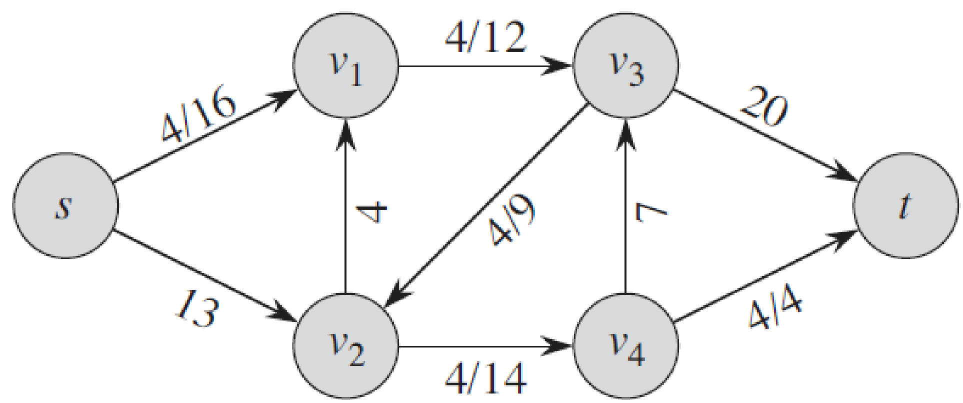
流网络 G 初始状态，此时G上的流 $f = 0$，$G_f == G$，直接在其中找一条增广路径 p。

注：增广路径可能不止一条，选其中一条即可

$$p = \{sv_1, v_1v_3, v_3v_2, v_2v_4, v_4t\},$$

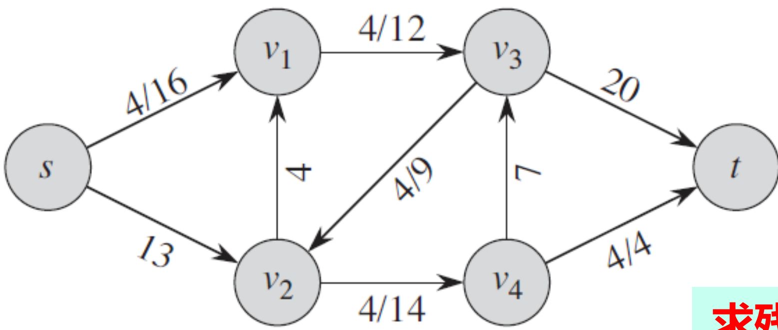
$$c_f(p) = 4$$

用 $c_f(p)$ 递增 G 上的流



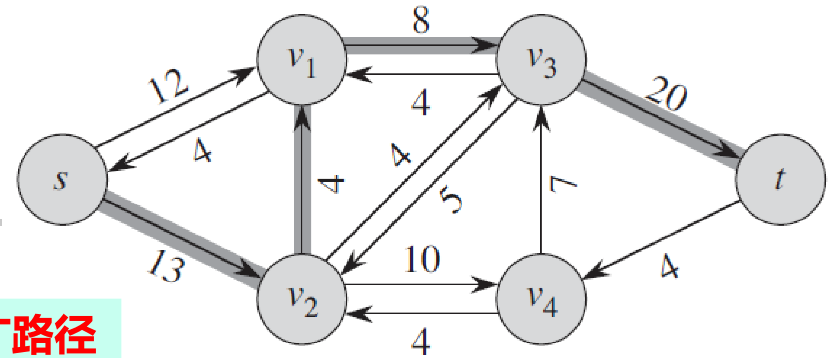
第一次扩增后的流网络

Iteration 2



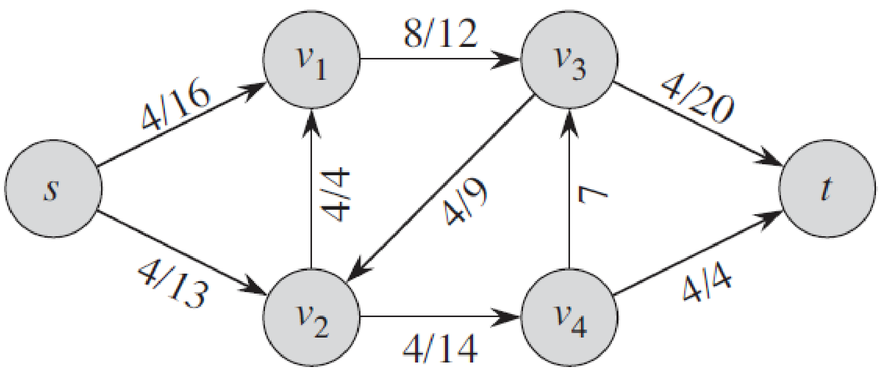
第一次扩增后的流网络

求残余网络和增广路径



$$p = \{sv_2, v_2v_1, v_1v_3, v_3t\}$$

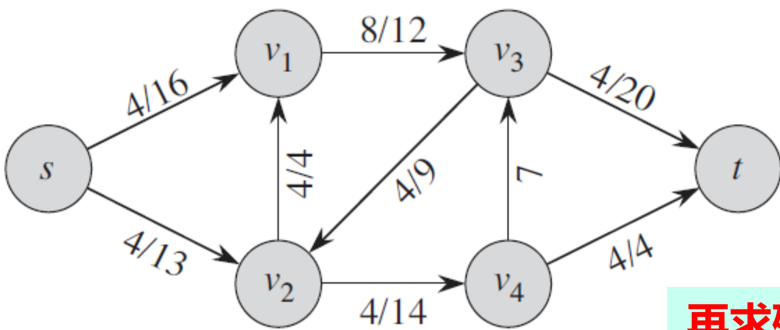
$$c_f(p) = 4$$



用 $c_f(p)$ 继续递增 G 上的流

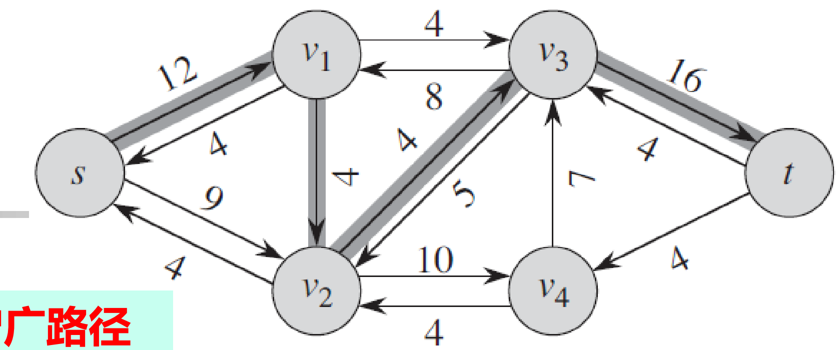
第二次扩增后的流网络

Iteration 3



第二次扩增后的流网络

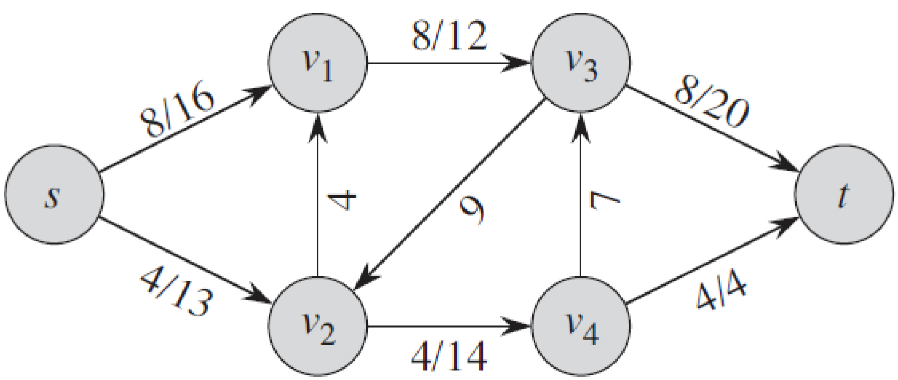
再求残余网络和增广路径



$$p = \{sv_1, v_1v_2, v_2v_3, v_3t\}$$

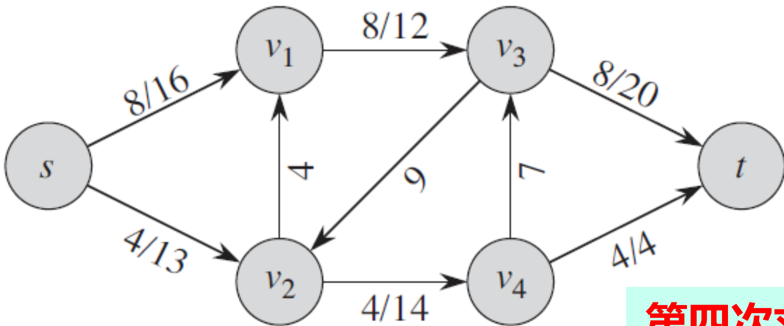
$$c_f(p) = 4$$

用 $c_f(p)$ 继续递增 G 上的流



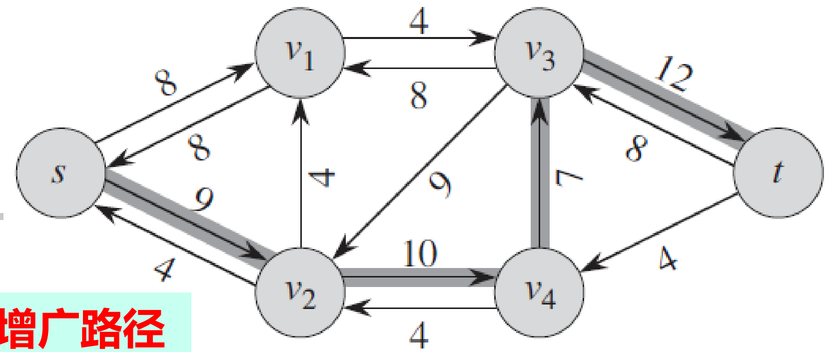
第三次扩增后的流网络

Iteration 4



第三次扩增后的流网络

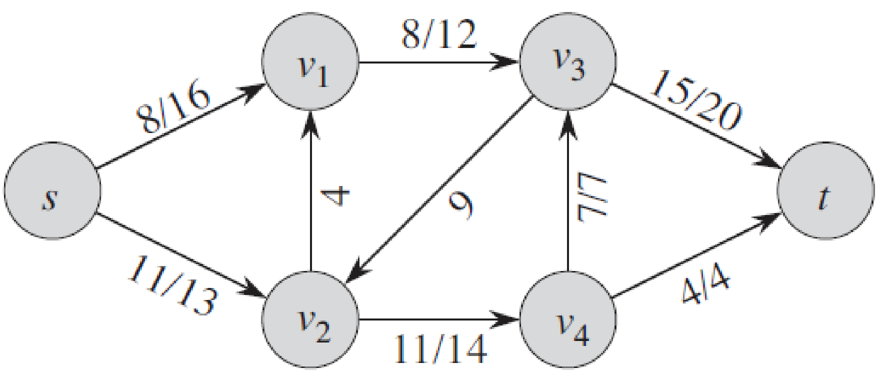
第四次求残余网络和增广路径



$$p = \{sv_2, v_2v_4, v_4v_3, v_3t\}$$

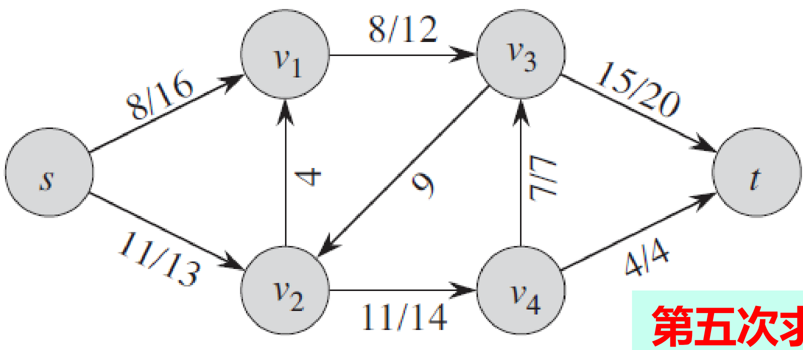
$$c_f(p) = 7$$

用 $c_f(p)$ 继续递增 G 上的流



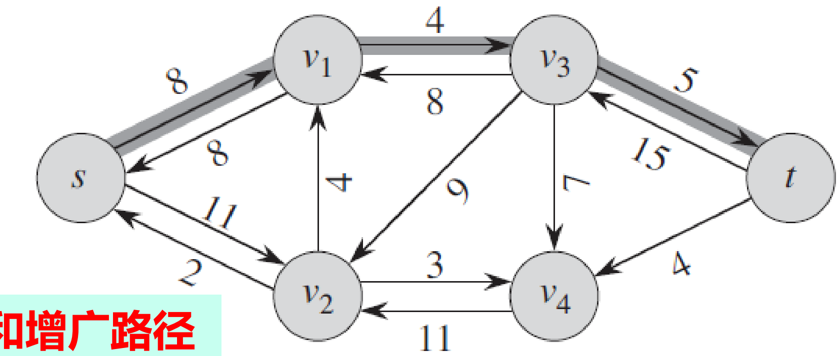
第四次扩增后的流网络

Iteration 5



第四次扩增后的流网络

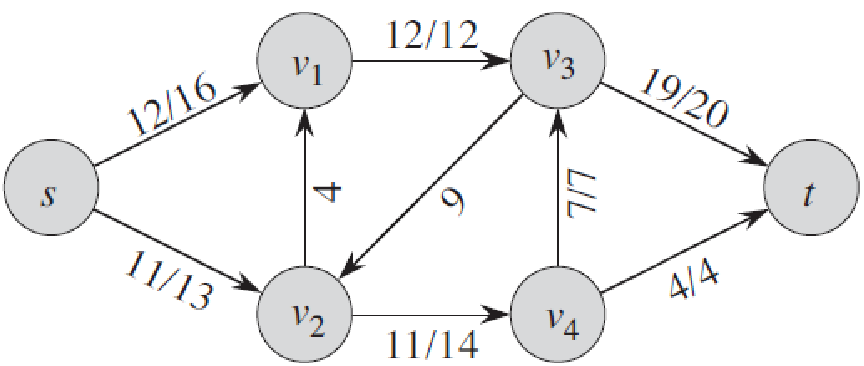
第五次求残余网络和增广路径



$$p = \{sv_1, v_1v_3, v_3t\}$$

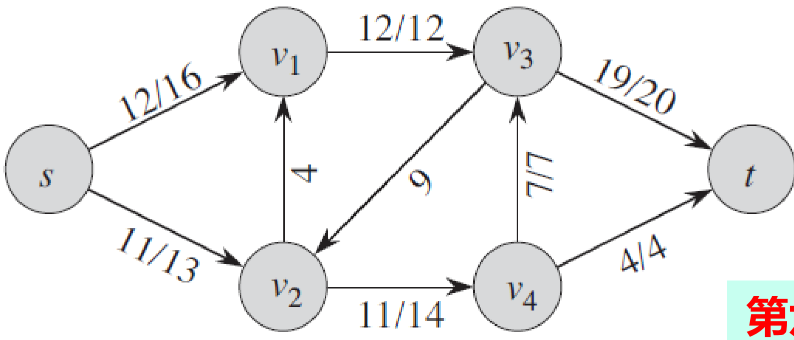
$$c_f(p) = 4$$

用 $c_f(p)$ 继续递增 G 上的流



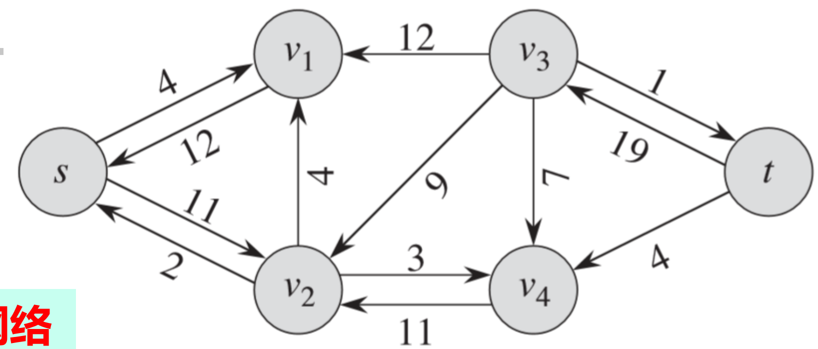
第五次扩增后的流网络

Iteration 6



第五次扩增后的流网络

第六次求残余网络



无增广路径，得到最大流

$|f^*| = 23$

注：这里记最大流为 f^*

无增广路径：G_f 中不存在从 s 到 t 的有向路径。

（2）Ford-Fulkerson算法复杂性分析

- **假定容量为整数。**

  注：如果边的容量是无理数，Ford-Fulkerson方法可能不能终止（**不收敛**）。

- **Ford-Fulkerson**算法的时间复杂度是 $O(|E| \times |f^*|)$ 。

运行时间分析：

① **while**：因为每一次循环，**流值至少增加1**，所以最多有 $O(|f^*|)$次迭代。

```
FORD-FULFERSON(G, s, t){
  for each edge (u,v) ∈G.E
      (u,v).f=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) ∈ E
              (u, v).f = (u, v).f + c_f(p)
          else
              (v, u).f = (v, u).f − c_f(p)
```

算法的主体

② 每次循环时间

每次循环做三个主要操作，**计算残存网络**、**寻找增广路径**和**更新每条边的流值**。

- 计算残存网络： $O(|E|)$
- 寻找增广路径： $O(|V|+|E|) \in O(|E|)$
- 用 $c_f(p)$ 更新流： $O(|E|)$

BFS 或 *DFS* 的时间复杂度都是$O(n+e)$，且 $|E|>|V|-1$。

综合： $O(|E| \times |f^*|)$

```
FORD-FULFERSON(G, s, t){
   for each edge (u,v) ∈G.E
       (u,v).f=0
   while there exists a path p from s to t in the residual network G_f      ← O(|f*|)
       c_f(p) = min {c_f(u, v) : (u, v) is in p }                           ┐
       for each edge (u, v) in p                                            │
           if (u, v) ∈ E                                                    │
               (u, v).f = (u, v).f + c_f(p)                                 ├ O(|E|)
           else                                                             │
               (v, u).f = (v, u).f − c_f(p)                                 ┘
```

7、Edmonds-Karp算法

Edmonds-Karp算法是Ford-Fulkerson算法的一种具体实现：

使用**广度优先搜索**寻找源点到汇点的**最短路径作为增广路径**。

FORD-FULKERSON(G, s, t)

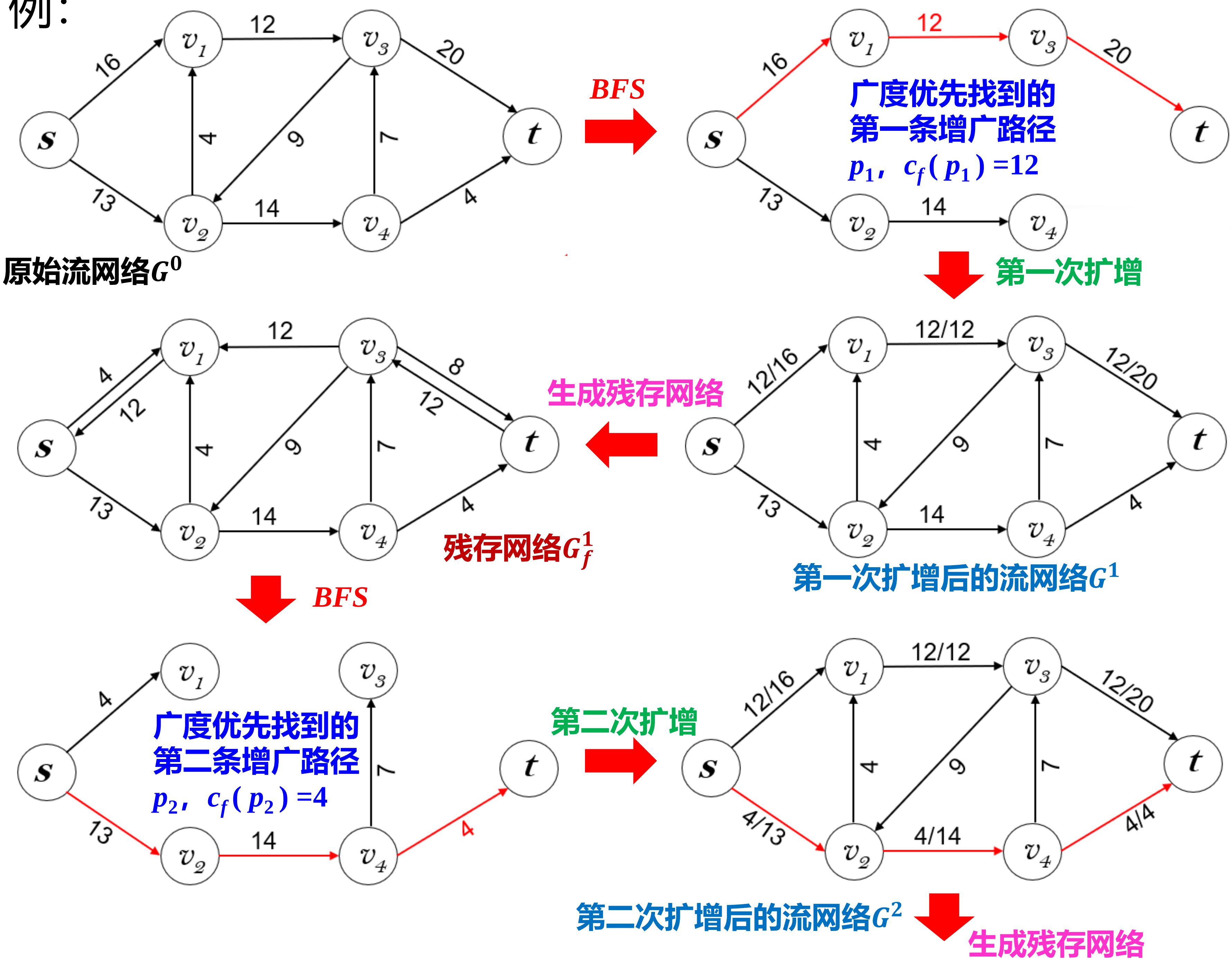
for each edge $(u, v) \in G.E$
    $(u, v).f = 0$
while there exists a path p from s to t in the residual network G_f
    $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
    for each edge (u, v) in p
        if $(u, v) \in E$
            $(u, v).f = (u, v).f + c_f(p)$
        else $(v, u).f =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

Edmonds-Karp**算法**用**广度优先搜索**找到的**最短路径作为增广路径**

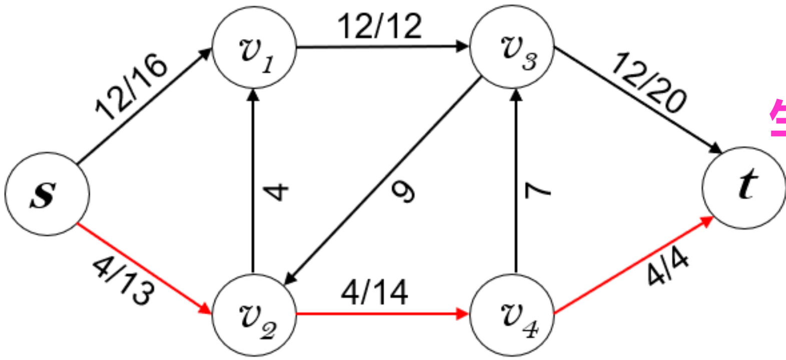
Edmonds-Karp算法的时间复杂度： $O(VE^2)$

Ford-Fulkerson算法: $O(|E| \times |f^*|)$

例：

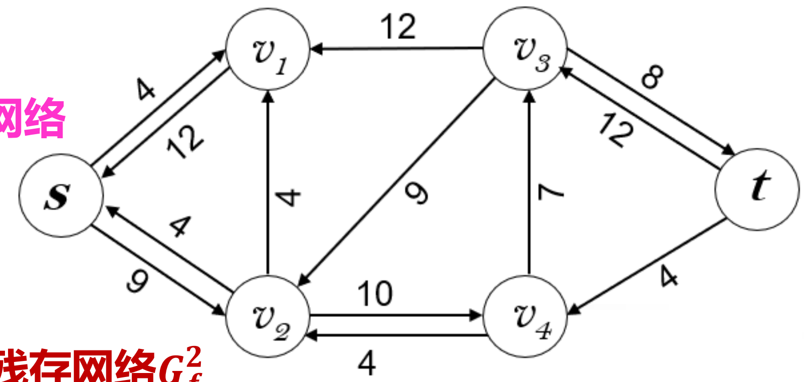


原始流网络G^0
BFS
广度优先找到的
第一条增广路径
p_1，$c_f(p_1)=12$
第一次扩增
第一次扩增后的流网络G^1
生成残存网络
残存网络G_f^1
BFS
广度优先找到的
第二条增广路径
p_2，$c_f(p_2)=4$
第二次扩增
第二次扩增后的流网络G^2
生成残存网络



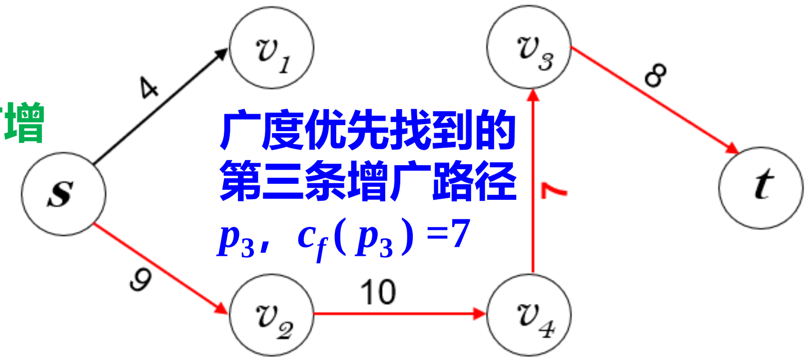
第二次扩增后的流网络G^2

生成残存网络



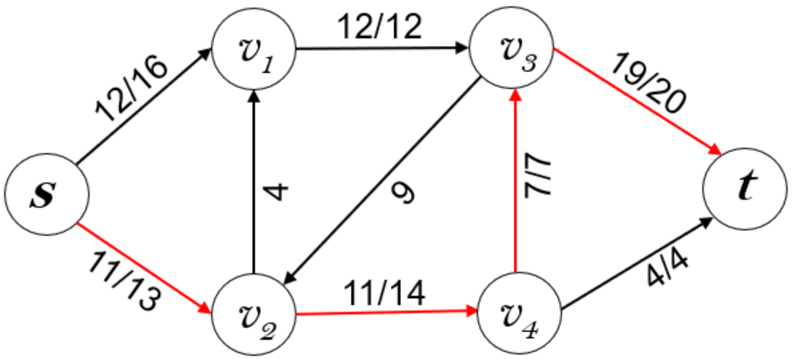
残存网络G_f^2

BFS



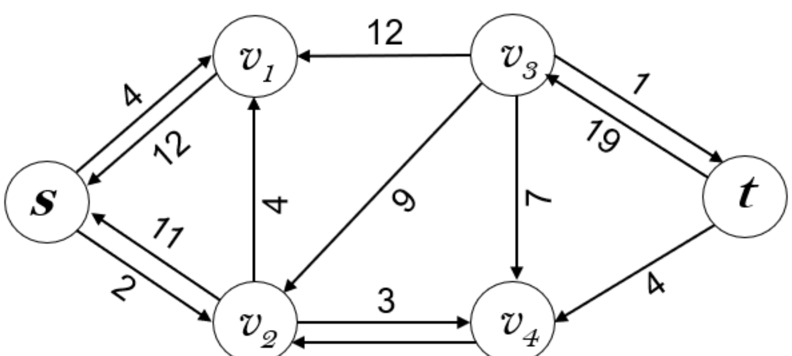
广度优先找到的第三条增广路径 p_3，$c_f(p_3)=7$

第三次扩增



第三次扩增后的流网络G^3

生成残存网络



残存网络G_f^3

没有增广路径了，算法结束，得到最大流。

BFS 的时间复杂度是$O(n+e)$

Edmonds-Karp算法的时间复杂度：

- **分析Edmonds-Karp算法的时间复杂度主要看以下问题：**

(1) 求 G_f、用 *BFS* 找最短路径（增广路径）及扩增流：$O(E)$。

(2) 迭代（**while**循环）中总共会找到多少条最短路径（**循环多少次**）？

- O(VE)：**找关键边问题。**（见后）

综合： $O(VE^2)$。

证明过程如下：

令 $\delta_f(u,v)$ 表示残存网络中 G_f 中从结点 u 到结点 v 的**最短路径长度**。**注：** 路径长度等于路径上的边数。

引理26.7：如果Edmonds-Karp算法运行在流网络 $G = (V, E)$ 上，源点为 s，汇点为 t，则对于所有结点 $v \in V - \{s, t\}$，残存网络 G_f 中的最短路径距离 $\delta_f(s, v)$ **随着每次流量的递增而单调递增。**

证明：

假设对于某个结点 $v \in V - \{s, t\}$，存在一个流量递增操作，导致残存网络 G_f 中从源结点 s 到 v 的**最短路径长度减小**。

设 f 是流量递增操作中第一个导致 G_f 中某条最短路径长度减少的**流量递增操作之前的流量**，f' 是**流量递增操作之后的流量**。

现假设 v 是在流递增操作中，**最短路径被减小的结点中 $\delta_{f'}(s, \bullet)$ 最小的结点**。则，根据此假设就应有：

$$\delta_{f'}(s, v) < \delta_f(s, v)。$$

下面证明该式不成立

$\delta_{f'}(s, v)$ 是 $G_{f'}$ 中 s 到 v 的最短路径长度

$\delta_f(s, v)$ 是 G_f 中 s 到 v 的最短路径长度

设 $p = s \cdots\rightarrow u \rightarrow v$ 为残存网络 $G_{f'}$ 中从源点 s 到结点 v 的一条最短路径，设 u 是这条路径上 v 的直接前驱，显然 $(u, v) \in E_{f'}$。

而且根据最短路径性质，有：

$$\delta_{f'}(s,u) = \delta_{f'}(s,v) - 1$$

路径长度等于路径上的边数

想想为什么?

再看这个 u：应有 $\delta_f(s,u) \leq \delta_{f'}(s,u)$

并且还有 $(u,v) \notin E_f$，这是因为，如果 $(u,v) \in E_f$，就有：

E_f是 G_f的边集

$$\delta_f(s,v) \leq \delta_f(s,u) + 1$$

三角不等关系

$$\leq \delta_{f'}(s,u) + 1$$

$$= \delta_{f'}(s,v)$$

即从 f 到 f'，s 到v 的最短路径变长了，而不是变短.

就与“v 是最短路径被减小的结点**”的假设矛盾了。**

所以就有 $(u,v) \notin E_f$。

由上可得:

① 若 $(u,v) \in E_{f'}$，则 $(u,v) \notin E_f$；

② 而 $(u,v) \in E_{f'}$ 说明 G 中 u 和 v 之间有边（即相邻接）。

进一步思考：为什么 $(u,v) \notin E_f$ 但 $(u,v) \in E_{f'}$?

这是因为：f 中没有 v 到 u 的流量，但在由 f 至 f' 的递增操作中，增加了 v 到 u 的流量（而使得在 f' 中存在 $v \rightarrow u$ 的流量，进而在 $G_{f'}$ 中存在边 (u, v)）。

故：正是由 f 至 f' 增加了 v 到 u 的流量，所以**在 G_f 中，边 (v, u) 在所选增广路径上**。即 G_f 中存在 $s \sim v \rightarrow u \sim t$ 的增广路径。

而 **Edmonds-Karp** 算法是按最短路径寻找增广路径，所以这条增广路径中的**子路径 $s \sim v \rightarrow u$ 是从 s 到 u 的一条最短路径**，v 是 u 的前驱。

(**注**：在 G_f 的增广路径子路径 $s \sim v \to u$中，v **是** u **的直接前驱**，而在前面设定的$G_{f'}$ 中的最短路径 $p = s \cdots \to u \to v$中，u **是** v **的直接前驱**)

再根据最短路径的性质，在 G_f 中应有：

$$\begin{aligned}\delta_f(s,v) &= \delta_f(s,u) - 1 \\ &\le \delta_{f'}(s,u) - 1 \\ &= \delta_{f'}(s,v) - 2\end{aligned}$$

可见：流量递增操作不可能使残存网络中 s 到 v 的最短路径长度减小，所以前面的假设不成立。

从而有：对于任意结点 $v \in V - \{s, t\}$，残存网络 G_f 中的最短路径长度 $\delta_f(s,v)$ 随着每次流量的递增而单调递增。引理得证。

定理26.8：如果Edmonds-Karp算法运行在源结点为 s 汇点为 t 的流网络 $G = (V, E)$ 上，则**算法执行的流量递增操作的次数为** $O(VE)$。

关键边：在残存网络 G_f 中，如果一条增广路径 p 的**残存容量**等于该条路径上边 (u, v) 的**残存容量**，即 $c_f(p) = c_f(u, v)$，那么边 (u, v) 称为增广路径 p 上的**关键边**。（注：每条增广路径上至少有一条关键边，思考关键边带来的影响）

关键边的一个关键特性：对当前流 f，一条边 (u,v) 若成为了关键边，则在递增操作后，因残存容量为0，所以在其后的残存网络中 (u,v) **将消失**；这一状态一直维持到后面某次递增产生了 v 至 u 的流量后，(u,v) **才可能再次出现在残存网络中**。

- **现在证明**：**一条边 (u,v) 成为关键边之后，在下次再成为关键边的时候，s 到 u 的最短距离至少会增加2。**

设 $u, v \in V$，且 $(u, v) \in E$。

首先考虑 (u, v) **第一次成为关键边时**：此时 (u, v) 处于增广路径上，而**增广路径是最短路径**，所以有：$\delta_f(s, v) = \delta_f(s, u) + 1$。

之后，**(u, v) 会从下一个残存网络中消失**。

如果 (u, v)**能再次成为关键边**，这必有之前某一步**从 u 到 v 的流量减小了**。设这一步之前的一步的流是f'，正是因为f'，使得 (u, v) 的反向边 (v, u) 出现在 $G_{f'}$ 的增广路径上，而在这条增广路径上有：$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$。

且根据引理26.7，有：$\delta_f(s, v) \le \delta_{f'}(s, v)$，

进而有 $\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$

$\ge \delta_f(s, v) + 1 = \delta_f(s, u) + 2$

即：一条边(u, v)成为关键边之后，在下次再次成为关键边的时候，s 到 u 的最短距离至少会增加2。

因此，从 (u, v) **第一次成为关键边**后算起，(u, v) 还能成为关键边的次数至多是 $(|V|-2)/2 = |V|/2-1$。 即一条边 (u, v) 能成为关键边的总次数最多为 $|V|/2$。

而算法中，**每条增广路径上至少有一条关键边**，且**每条边都可能成为关键边**。所以所有边能成为关键边的总次数最多是：$E \times |V|/2$，所以**出现的关键边总数至多为** $O(VE)$，亦即**算法最多迭代 $O(VE)$ 次**。 ■

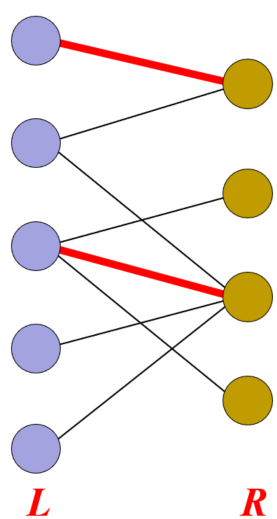
由于在用**广度优先搜索**寻找增广路径时，**Ford-Fulkerson**方法的每次迭代（构造残存网络、找增广路径和增加流值等）都可以在 $O(E)$ 时间内实现，所以**Edmonds-Karp算法的总运行时间是** $O(VE^2)$。

26.3 最大流算法的一个应用：寻找最大二分匹配

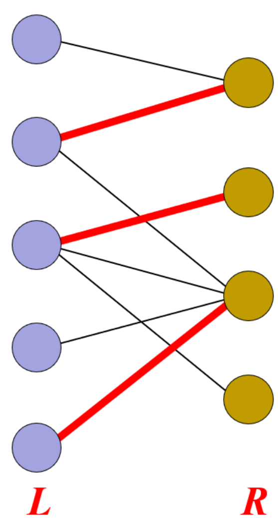
1、匹配

无向图 $G = (V, E)$ 的一个**匹配是边的一个子集** $M \subseteq E$，使得对所有结点 $v \in V$，M 中最多只有的一条边与其相连。M **中边的数量**称为 M 的**基数**，记为 $|M|$。

最大匹配：图 G 的基数最大的匹配，称为**最大匹配。**



一个匹配



最大匹配 (最大匹配不唯一)

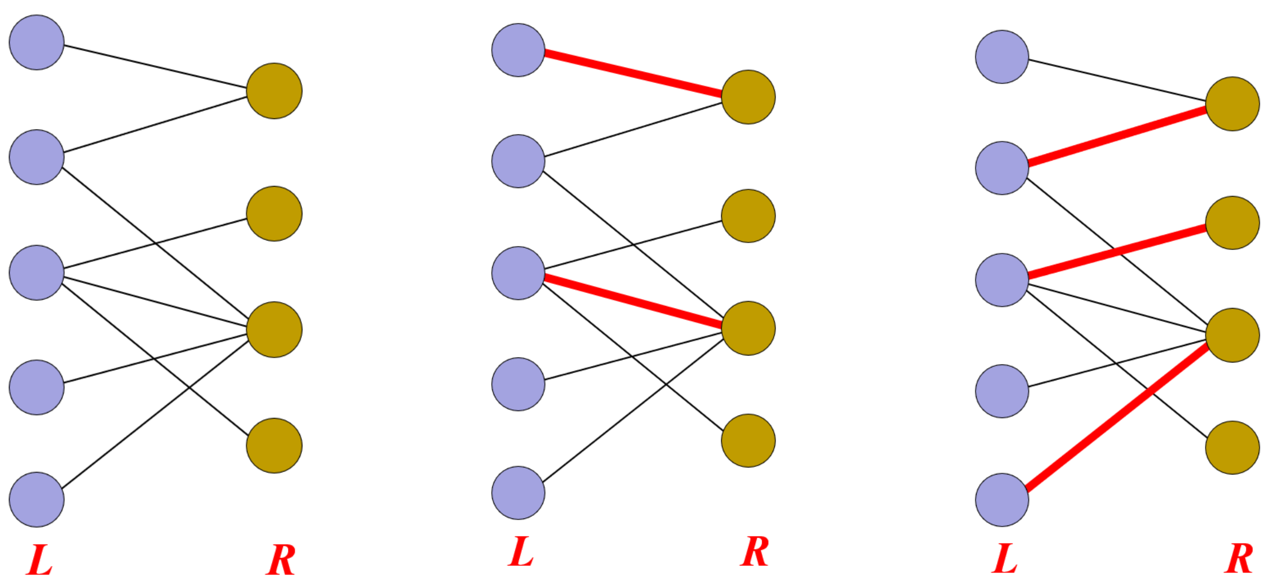
2、二分图：

二分图中的结点集 V 可以**划分**为两部分 L 和 R（即切割），使得 $L \cup R = V, L \cap R = \emptyset$，且所有的边**横跨该划分**，即对于任意的 $(u, v) \in E$，有 $u \in L, v \in R$ 或者 $v \in L, u \in R$。

二分匹配问题：求二分图中的匹配。

最大二分匹配问题：求二分图中的最大匹配。

如：



二分图　　二分匹配　　最大二分匹配

3、利用网络流算法求最大二分匹配

将二分图改造成流网络，然后利用**Ford-Fulkerson**算法求解。

（1）将二分图改造成流网络

设二分图 $G=(V,E)$，按照下述方式，将其改造成一个流网络 $G' = (V', E')$：

- **新增 源点** s 和 **汇点** t，令 $V' = V \cup \{s,t\}$；
- **新增** s **到** L **中所有结点的边**和 R **中所有结点到** t **的边，即令：**

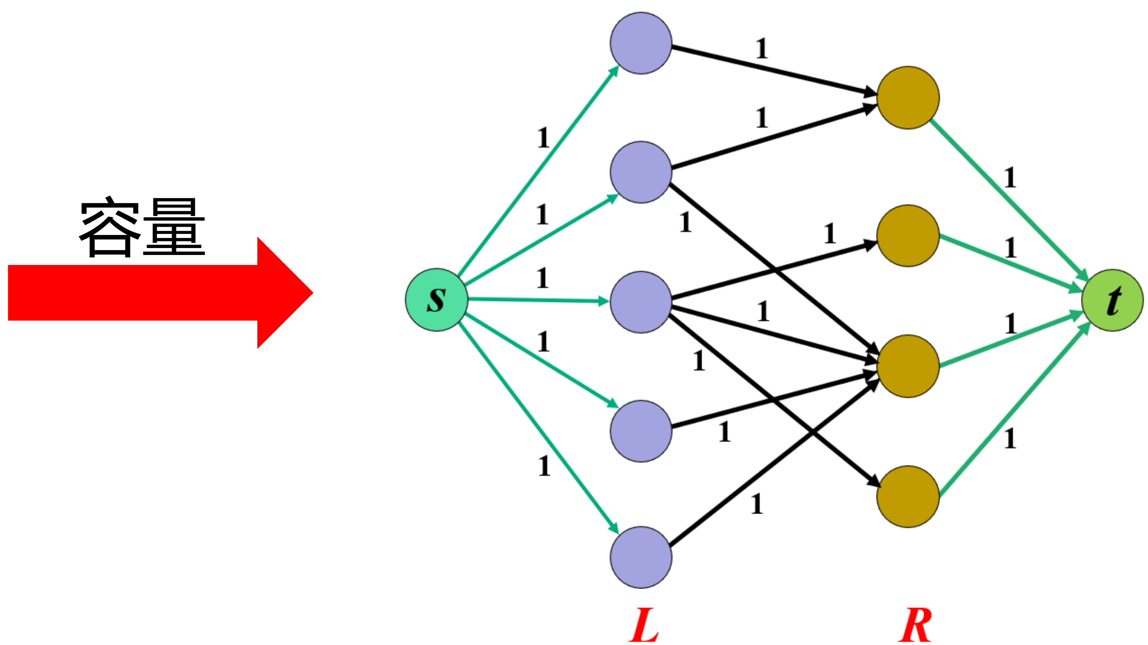
$$E' = \{(s,u): u \in L\} \cup \{(u,v): (u,v) \in E\} \cup \{(v,t): v \in R\}$$

- 定义 E' 每条边上的容量为**单位容量**：

$$c(u,v) = 1 \quad for\ all\ \ (u,v) \in E'$$

不失一般性，这里将 G' 中的边全部视为从“左”至“右”的**有向边**。

如：



并且，不失一般性，假定 G 的结点集 V 中的**每个结点至少有一条相连的边**。则有：$|E| \geq |V|/2$，且有 $|E| \leq |E'|$，并且 $|E'| = |E| + |V| \leq 3|E|$，所以 $|E'| \in \Theta(|E|)$（**数据量的数量级相等**）。

（2）寻找最大二分匹配

利用**Ford-Fulkerson算法**求 G' **中的最大流**。

问题的解是：流值大于0且在原图中的边将构成最大匹配，最大匹配的基数等于最大流的流值。

为此需要证明以下三点：

（a）原图中的匹配和转化后流网络中流一一对应，并且匹配集中的边数对应于流值。

（b）证明在容量限制是整数的前提下，**Ford-Fulkerson**方法产生的流是整数值的流。

（c）证明最大流的流值等于最大匹配的基数。

其中：（a）由以下引理给出

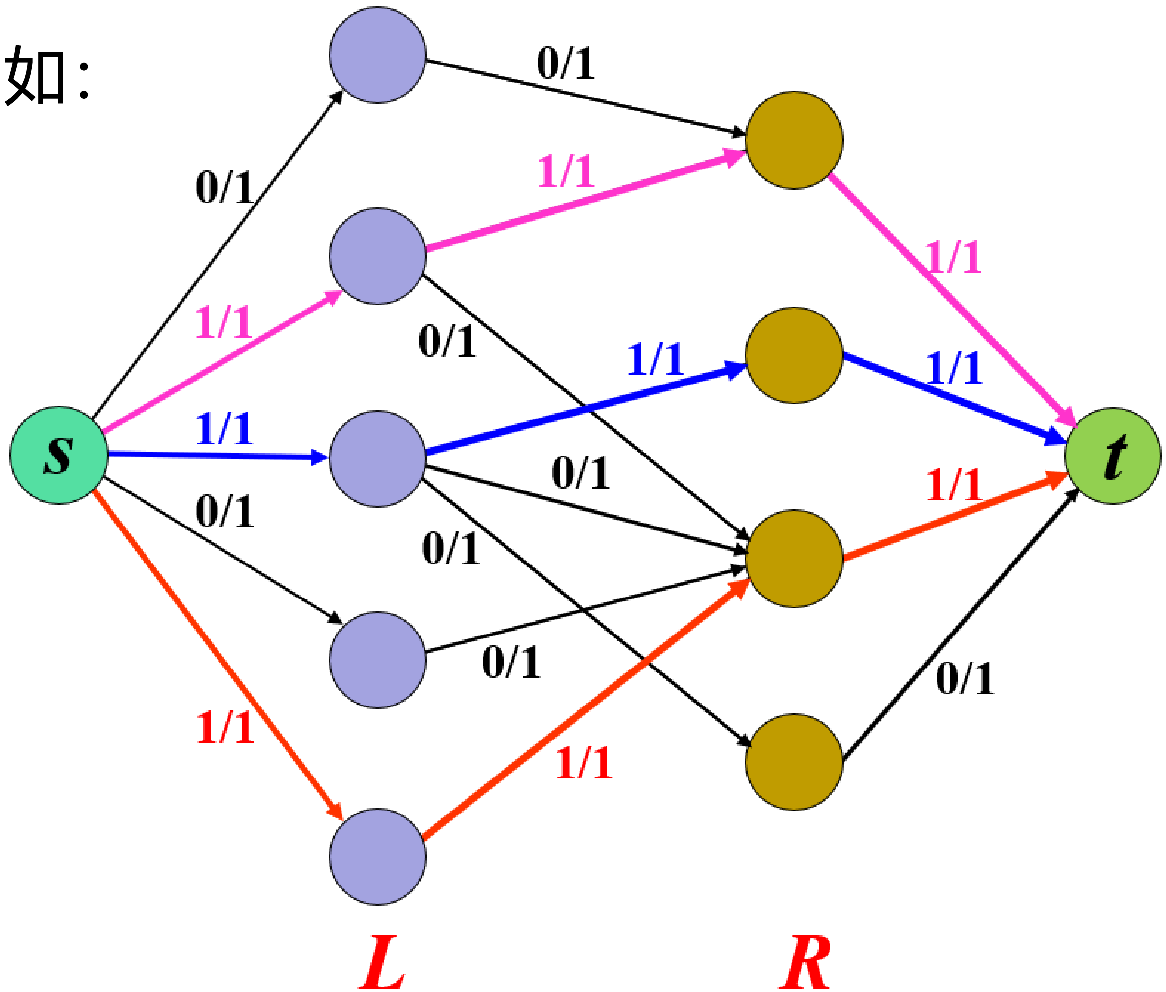
引理26.9：如果 M 是 G 中的一个匹配，则流网络 G' 中存在一个整数值的流 f，使得 $|f| = |M|$。**反之**，如果 f 是 G' 中的一个整数流，则 G 中存在一个匹配 M，使得 $|M| = |f|$。

证明： 假定 M 是 G 中一个匹配，在 G' 中定义一个流 f 与 M 对应：

- 若 $(u, v) \in M$，$f(s, u) = f(u, v) = f(v, t) = 1$
- 而 E' 中的其它边 (x, y)，$(x, y) \in E', (x, y) \notin M$，$f(x, y) = 0$。

(可以验证 f 满足**容量限制**和**流量守恒**性质，自行验证)

如:



反之，假定 f 是G' 的一个**整数值流**，则在 G 中定义一个边子集 M 与 f 对应：

$$M = \{(u, v): u \in L, v \in R, and\ f(u, v) > 0\}$$

(即 M 由 G 中对应于 G' 中的那些**流量大于0**且**横跨划分** ***(L, R)*** 的边组成)

根据 G' 的结构特点，① 对 L 中的每个结点 $u \in L$，由于只有一条入边 (s, u)，且**容量为1**，而这里 f **被定义为整数值的流**（即流值必须为整数），所以 u **最多只能**从 (s, u) 接收**1个整数单位流量**，而根据流量守恒性质，如果有流入的1单位流量，则 u 同时也必有**1个单位流量**沿唯一的一条边流出至其对端结点。

② 同样，对于 R 中的每个结点 $v \in R$，**最多只能**有1个整数单位流量通过一条入边流入并通过 (v, t) 流出至 t 。

所以，对 V 中的任何结点，如果在 G' 中连接有一条流值大于0 的边，它只能连接一条（不管是流入还是流出）。

根据 M 的定义： M 由那些 $f(u,v)>0$ 的“边”组成，所以 M 是 G 的一个匹配（即：由于有 G' 中**流 f 的前提约束**，使得 G 的结点集 V 中的每个结点最多只能和 M 中的一条边相连）。

而且，事实上，由于“f **是整数流、且边的容量都为1**”的设定，使得 $f(u,v)>0$ 实际就意味着 $f(u,v)=1$。而且，那些流量大于 0 边 (u,v) 都是 $u\in L$ 且 $v\in R$ 的横跨划分的边，流仅从 u 流向 v，不存在反向流，这就使得 $|M|=|f|$。

综上所述，对于一个二分图 $G=(V,E)$ 和它对应的流网络 $G'=(V',E')$，以及如前定义的 G 的匹配 M 和 G' 上的流 f，有：

（1）对于 E 中任何属于 M 中的边 (u,v)，有 $<u,v>\in E'$ 且 $<v,u>\notin E'$，且由于 $u\in L$，$v\in R$，所以在 G' 中，边 (u,v) **横跨划分**$(L\cup\{s\}$，$R\cup\{t\})$，且 $f(u,v)=1$；

（2）而对任何不属于 M 中的所有边，其上的流值都等于0。

（3）流 f **横跨划分** $(L\cup\{s\}$，$R\cup\{t\})$。

根据引理26.4（$|f|=f(L\cup\{s\},R\cup\{t\})$），即有：

$$|f|==|M|。$$

（b）由以下定理给出

（即在容量限制是整数的前提下，Ford-Fulkerson 方法产生的流是整数值的流）

定理26.10（完整性定理）如果容量函数 c 只能取整数值，则 **Ford-Fulkerson** 方法所生成的最大流 f 满足 $|f|$ 是整数值的性质。而且，对于所有的结点 u 和 v，$f(u,v)$ 的值都是整数。

证明：可以通过对迭代次数的归纳进行证明（自学）。

（c）**由以下推论给出**

（即证明最大流的流值等于最大匹配的基数）

推论26.11 二分图 G 的一个最大匹配 M 的边数等于其对应的流网络 G' 中某一最大流 f 的值。

证明：（**反正法**）

假定 M 是图 G 中的一个最大匹配，但流网络 G' 中对应的流 f 不是最大流。那么 G' 中就有一个更大的流 f'：$|f'| > |f|$。

由于 G' 的容量都是整数值，根据**定理26.10**，f' 的值也是整数值。再根据**引理26.9**，G 中就存在一个与 f' 对应的匹配 M'，且有 $|M'| = |f'| > |f| = |M|$，这与 M 是最大匹配相矛盾。

同理可证，如果 f 是 G' 中的一个最大流，则其对应的匹配是 G 的一个最大匹配。

■

时间分析：

二分图任何匹配的基数的**最大值**为 $\boldsymbol{min(L,\ R)} = \boldsymbol{O(V)}$，所以 G' 中的最大流的值为 $\boldsymbol{O(V)}$ 。

所以可以在 $O(VE') = O(VE)$ 时间内找到一个二分图的最大匹配。

注：*Ford-Fulkerson*算法的时间复杂度是：$O(|E| * f^*)$

$|E'| = \Theta(E)$

本章小结

1. 基本概念：流网络、流、最大流等；

2. Ford-Fulkerson方法的基本思想；

3. 残存网络、增广路径、最大流最小切割定理；

4. Ford-Fulkerson算法和Edmonds-Karp算法，相关性质和一些证明；

4. 最大流算法的应用：最大二分匹配问题。

- 作业：26.1-1，26.2-3，26.3-1