

算法设计与分析

Computer Algorithm Design & Analysis
2019.11

王多强

dqwang@mail.hust.edu.cn

群名称: 2019-算法
群 号: 835135560



群名称: 2019-算法
群 号: 835135560

A decorative graphic on the left side of the slide, consisting of overlapping blue, red, and yellow squares with a black crosshair.

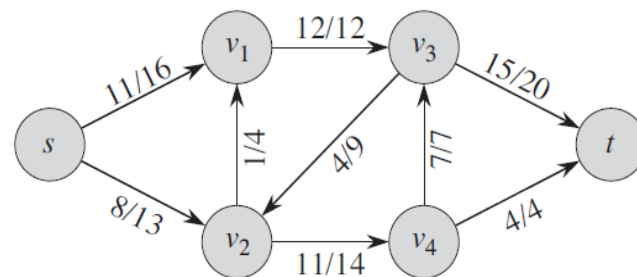
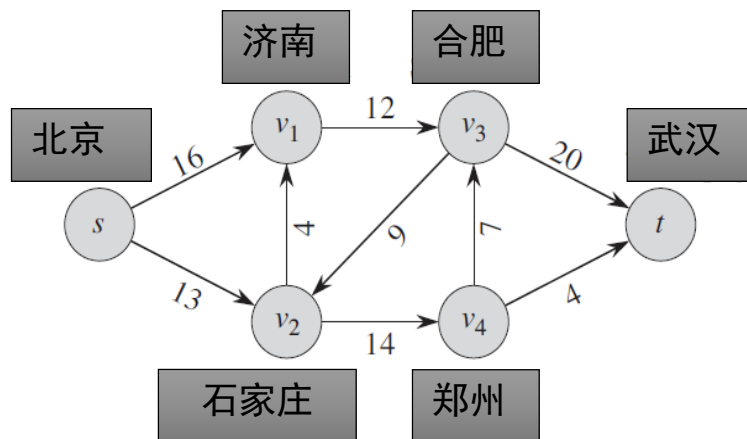
Chapter 26

Maximum Flow

最大流

最大流问题由 T. E. Harris and F. S. Ross于1954年提出

问题的引出：实例**物流网络**



最优解

有一个**源结点**和一个**汇点**，从源结点向汇点“运输”货物。

在不违反任何**路径容量限制**的条件下，从源结点到汇点运送

货物的**最大速率**是多少——这一问题的抽象称为**最大流问题**。

26.1 流网络

带权有向图：网络

□ **结 点**：表示城市

□ **有向边**：表示运输路径和物流的方向

□ **权 重**：表示运量限制

□ **流**：一条从源点到汇点的路径即路径上的流量

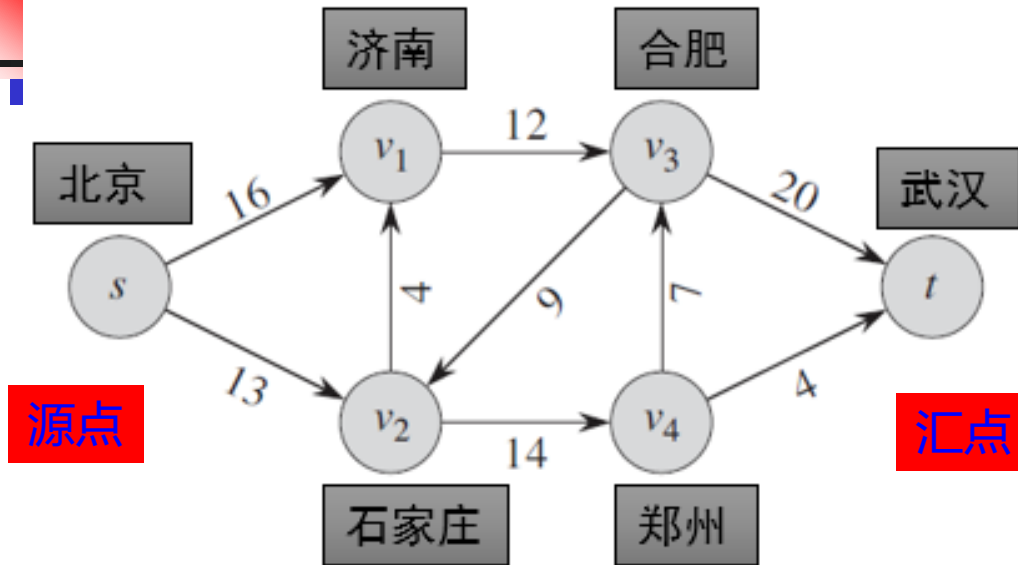
—— 这种用来表示“**流 (flow)**”的图称为 “**流网络**”。

- (1) **流量守恒**：除源结点和汇点外，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；
- (2) 物料的生成速率和接收速率恒定且足够快、足够多，满足需要；
- (3) 每条边上的容量是物料通过该边的最大速率，不能突破。

流网络是一个有向图 $G = (V, E)$ ，边上定义有**容量函数**： $c: E \rightarrow R^+ \cup \{0\}$

- (1) 有一个**源结点** s 和**汇点** t ;
- (2) 有向边表示流向;
- (3) 每条边 $(u, v) \in E$ 上有一个非负的**容量值** $c(u, v) \geq 0$;
如果 $(u, v) \notin E$ ，为方便起见，定义 $c(u, v) = 0$;
- (4) 如果边集合 E 中包含边 (u, v) ，则图中不包含其反向边 (v, u) ;
- (5) 图中不允许有自循环;
- (6) **流网络是连通图**，每个结点都在从 s 到 t 的某条路径上;
- (7) 除源结点外，每个结点至少有一条流入的边;
- (8) 除汇点外，每个结点至少有一条流出的边;
- (9) $|E| \geq |V| - 1$

流网络示例



容量函数

$$c(s, v_1) = 16$$

$$c(s, v_2) = 13$$

$$c(v_1, v_3) = 12$$

$$c(v_2, v_1) = 4$$

$$c(v_2, v_4) = 14$$

$$c(v_3, v_2) = 9$$

$$c(v_4, v_3) = 7$$

$$c(v_3, t) = 20$$

$$c(v_4, t) = 4$$

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

$$E = \{sv_1, sv_2, v_1v_3, v_2v_1, v_2v_4, v_3v_2, v_4v_3, v_3t, v_4t\}$$

流网络遵循以下基本性质：

- (1) **流量守恒**：除源结点和汇点外，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；
- (2) 物料的生成速率和接收速率恒定且足够快、足够多，满足需要（包括源结点的输出和所有结点的输入）；
- (3) 每条边上的容量是物料通过该边的最大速率，不能突破。

设 $G = (V, E)$ 是一个流网络, 其容量函数为 $c: E \rightarrow R^+ \cup \{0\}$ 。

设 s 为源结点, t 为汇点。

流 是定义在 G 上的一个实值函数映射, 记为 $f: V \times V \rightarrow R$, 并满足:

(1) **容量限制**: 对于所有的结点 $u, v \in V$, 有

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

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

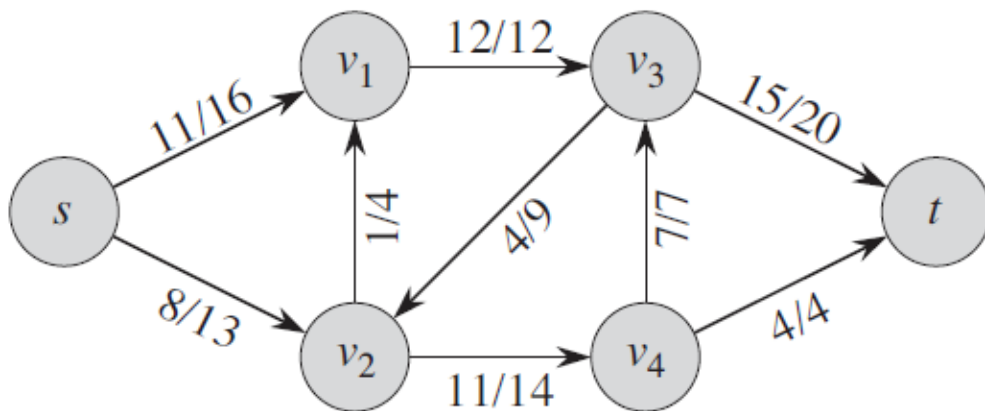
$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$

若 $(u, v) \notin E$, 记 $f(u, v) = 0$ 。

流的值:

一个流 f 的值定义为流出源结点 s 的总流量减去流入源结点 s 的总流量, 即源结点的净流出量, 用 $|f|$ 表示:

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

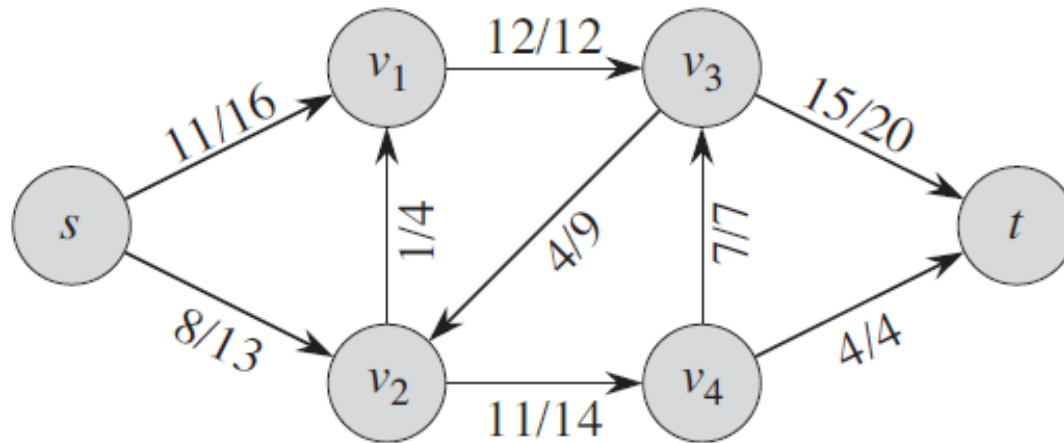


$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) = 19$$

$$\begin{aligned} f(s, v_1) &= 11 \\ f(s, v_2) &= 8 \\ f(v_1, v_3) &= 12 \\ f(v_2, v_1) &= 1 \\ f(v_2, v_4) &= 11 \\ f(v_3, v_2) &= 4 \\ f(v_4, v_3) &= 7 \\ f(v_3, t) &= 15 \\ f(v_4, t) &= 4 \end{aligned}$$

最大流问题：在给定的流网络G中找一个**流值最大的流**

最大流示例



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

标准流网络:

(1) 无反向边

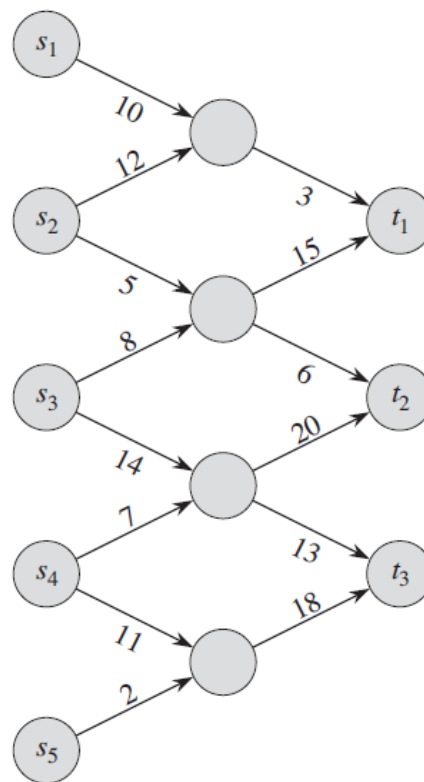
反向边: 也称为**反向平行边**。一个有向图中, (v, u) (u, v) 互为反向平行边。

无反向边: 如果 $(u, v) \in E$, 则 $(v, u) \notin E$ 。

(2) 只有单一的源结点和汇点。

非标准流网络:

不满足上述要求的流网络**是非标准**
的流网络。对于非标准的流网络可转化
为标准流网络。



非
标
准
流
网
络

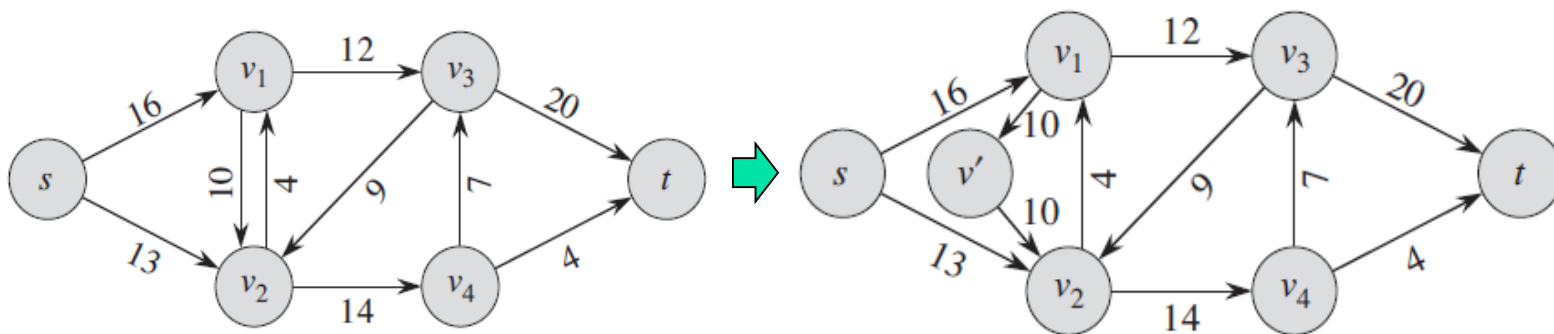
将非标准的流网络转化为标准流网络

(1) 反向平行边

转化方法：反向平行边 (u, v) 和 (v, u) ，选择其中的一条，然后加入一个新的结点 v' ，将其分为两条边 (u, v') 和 (v', v) ，并置：

$$c(u, v') = c(v', u) = c(u, v)$$

如：



替换 (v_1, v_2)

(2) 多个源结点和多个汇点

◆ 多个源结点

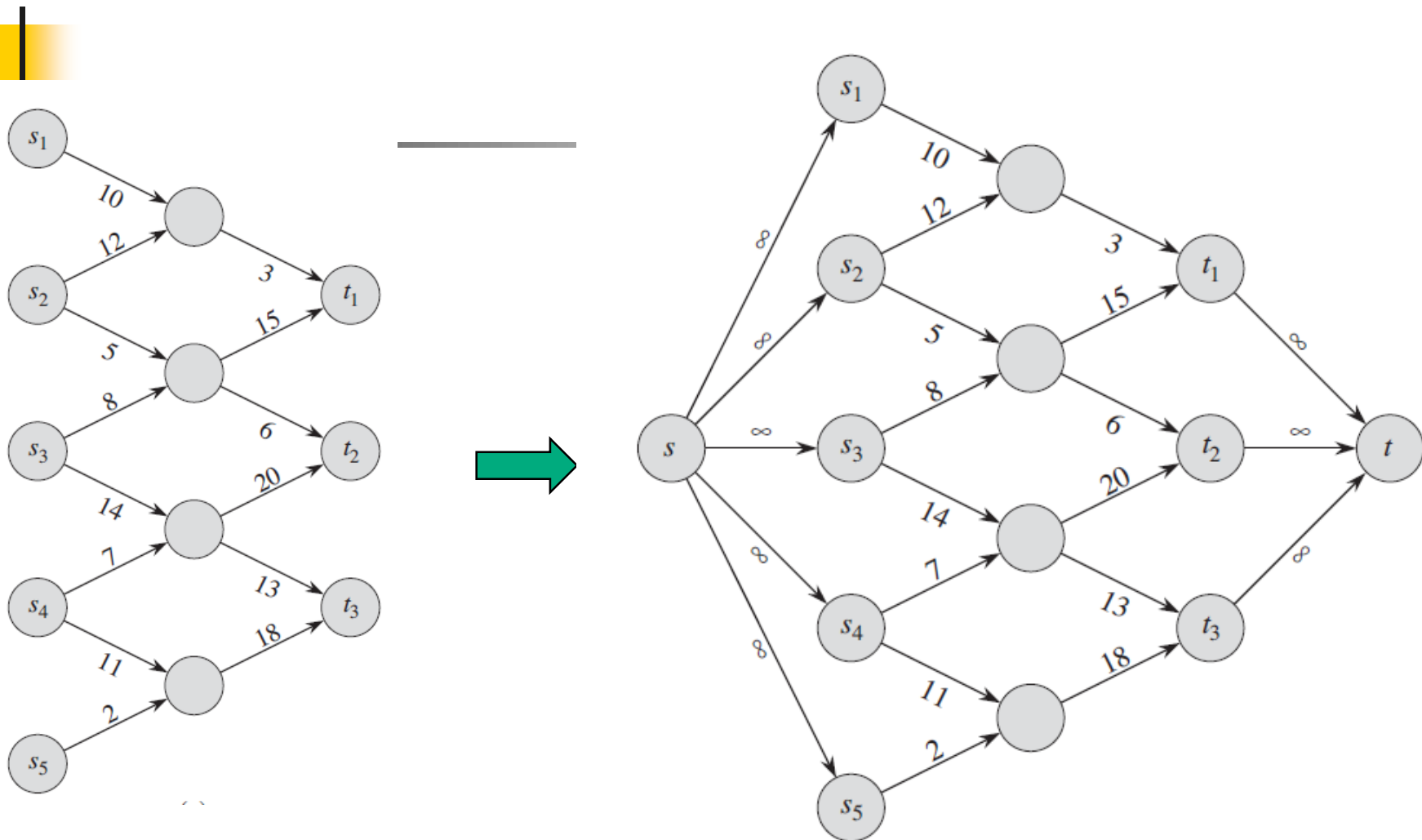
转化方法： 加入一个**超级源结点s**，并加入有向边 (s, s_i) ，
然后令 $c(s, s_i) = \infty$ ， $1 \leq i \leq m$ ；

◆ 多个汇点

转化方法： 加入一个**超级汇点t**，并加入有向边 (t_i, t) ，
然后令 $c(t_i, t) = \infty$ ， $1 \leq i \leq n$ 。

可以证明：转化后的两个网络是等价的，具有相同的流。

例:



求流网络的最大流：

从最小流值开始，一点一点地增加，直到最大值。

从而引出以下问题：

- (1) 最小流值是多少？**
- (2) 怎么增加流值？**
- (3) 何时能得到最大流？**

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**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

方法的要点:

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

◆ 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。

判断是否得到最大流的条件是**最大流最小切割定理**，该定理将说明**算法何时终止，并在终止时获得一个最大流**。

A decorative horizontal bar with a yellow-to-white gradient and a vertical black line is positioned to the left of the title.

Ford-Fulkerson方法的过程描述

FORD-FULKERSON-METHOD(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)

对给定流网络 G 和流量 f , G 的**残存网络**记为 G_f , 也是一个有向图, 由 G 中的结点和以下的边组成:

对于 G 中的一条任意边 (u,v) ,

(1) 若 $f(u,v) < c(u,v)$, 则将**边 (u,v)** 和它的**反向边 (v,u)** 加入 G_f , 并设它们的**残存容量**为:

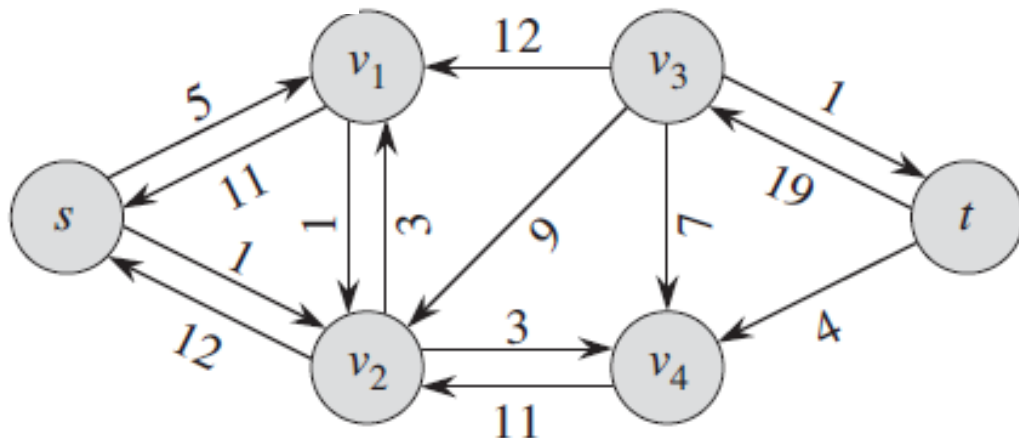
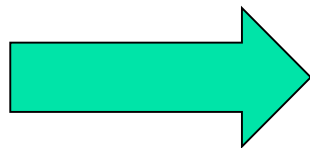
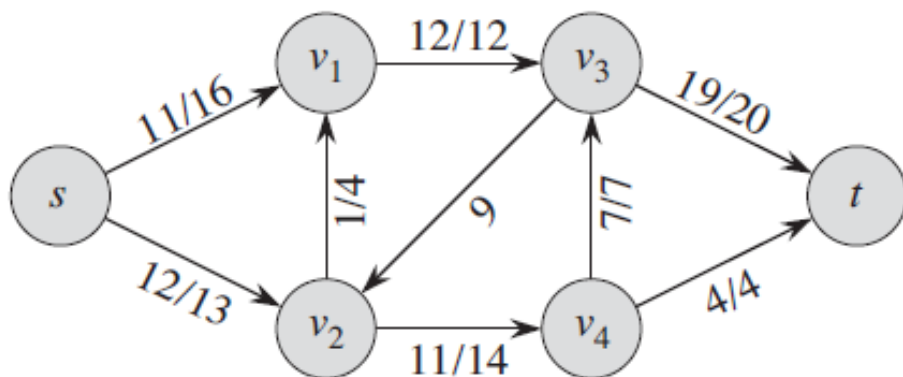
$$c_f(u, v) = c(u, v) - f(u, v)$$

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

残存容量 $c_f(u, v)$
反映了边上可以增加流量的空间。

(2) 如果边 (u,v) 的流量等于其容量, 则 $c_f(u,v) = 0$, (u,v) 不加入 G_f 。但**反向边 (v,u)** 加入 G_f , 并置 $c_f(v,u) = f(u,v)$

例:



残存网络

残存容量

设流网络 $G = (V, E)$, f 是 G 中的一个流

◆ 残存容量 :

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

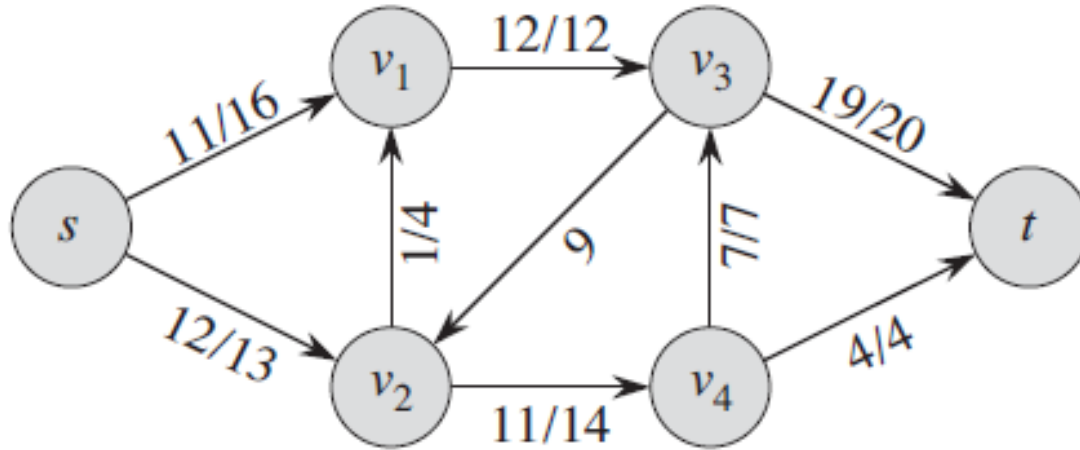
由 f 所诱导的图 G 的残存网络记为 G_f :

$$G_f = (V, E_f)$$

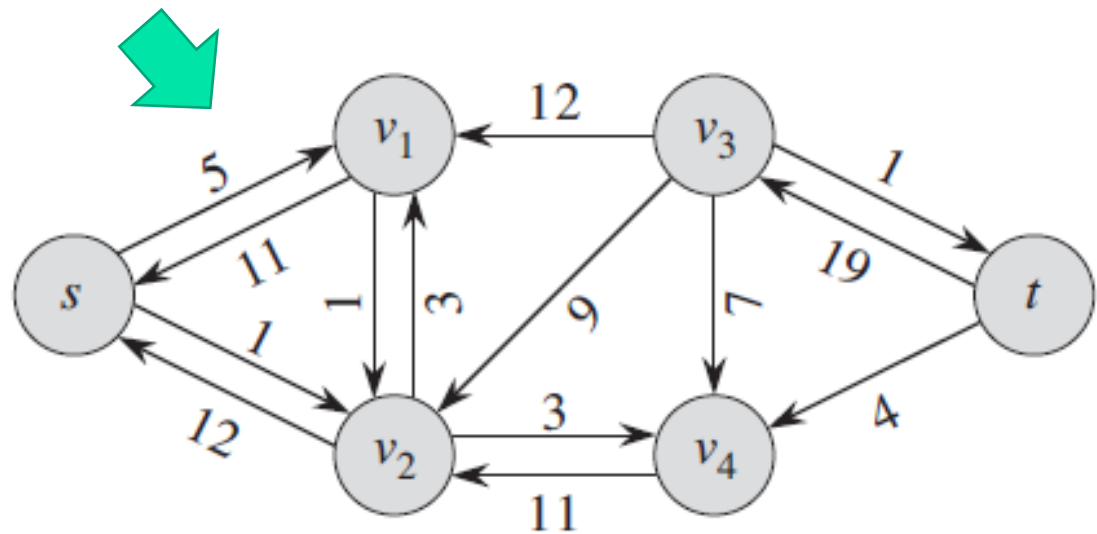
$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\} \text{ 且 } |E_f| \leq 2 |E|.$$

仅由残存容量大于0的边组成

残存网络示例:



流网络 G



残存网络 G_f

定义 $f \uparrow f'$ 为用流 f' **对流 f 递增** (augmentation) 后得到的流:

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

其中,

$f'(u, v)$ 表示的是边 (u, v) 上流量的增加,

$f'(v, u)$ 表示的是边 (u, v) 上流量的减少

引理26.1 设 $G=(V, E)$ 为一个流网络, 源结点为 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

1.证明 $f \uparrow f'$ 是图G的一个流: (满足容量限制和流量守恒)

(1) 容量限制

根据定义, 如果边 $(u, v) \in E$, 则 $c_f(v, u) = f(u, v)$ 。而且

因为 f' 是 G_f 的一个流, 所以 $f'(v, u) \leq c_f(v, u) = f(u, v)$ 。

因此,

$$\begin{aligned}(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \quad (\text{根据定义}) \\ &\geq f(u, v) + f'(u, v) - f(u, v) \\ &= f'(u, v) \\ &\geq 0.\end{aligned}$$

每条边上的流量不为负

此外,

$$\begin{aligned}(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \\ &\leq f(u, v) + f'(u, v) \\ &\leq f(u, v) + c_f(u, v) \\ &= f(u, v) + c(u, v) - f(u, v) \\ &= c(u, v) .\end{aligned}$$

满足容量限制

所以递增后, 流 $f \uparrow f'$ 不超过容量的限制。

(2) 流量守恒

对所有的 $u \in V - \{s, t\}$ 有:

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

注: f' 是 G_f 的一个流, f 和 f' 都满足流量守恒

2. 证明: $|f \uparrow f'| = |f| + |f'|$

定义 $V_1 = \{v : (s, v) \in E\}$

$V_2 = \{v : (v, s) \in E\}$

任意 (s, v) 和 (v, s) 不可能同时存在于流网络中, 则:

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

则有,

$$\begin{aligned} |f \uparrow f'| &= \sum_{v \in V} (f \uparrow f')(s, v) - \sum_{v \in V} (f \uparrow f')(v, s) \\ &= \sum_{v \in V_1} (f \uparrow f')(s, v) - \sum_{v \in V_2} (f \uparrow f')(v, s), \end{aligned}$$

$$|f \uparrow f'|$$

$$\begin{aligned}
 &= \sum_{v \in V_1} (f \uparrow f')(s, v) - \sum_{v \in V_2} (f \uparrow f')(v, s) \\
 &= \sum_{v \in V_1} (f(s, v) + f'(s, v) - f'(v, s)) - \sum_{v \in V_2} (f(v, s) + f'(v, s) - f'(s, v)) \\
 &= \sum_{v \in V_1} f(s, v) + \sum_{v \in V_1} f'(s, v) - \sum_{v \in V_1} f'(v, s) - \sum_{v \in V_2} f(v, s) - \sum_{v \in V_2} f'(v, s) + \sum_{v \in V_2} f'(s, v) \\
 &= \sum_{v \in V_1} f(s, v) - \sum_{v \in V_2} f(v, s) + \sum_{v \in V_1} f'(s, v) + \sum_{v \in V_2} f'(s, v) - \sum_{v \in V_1} f'(v, s) - \sum_{v \in V_2} f'(v, s) \\
 &= \sum_{v \in V_1} f(s, v) - \sum_{v \in V_2} f(v, s) + \sum_{v \in V_1 \cup V_2} f'(s, v) - \sum_{v \in V_1 \cup V_2} f'(v, s) \\
 &= \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) + \sum_{v \in V} f'(s, v) - \sum_{v \in V} f'(v, s) \\
 &= |f| + |f'|
 \end{aligned}$$

注: $V_1 \cup V_2 \subseteq V$
 $V_1 \cap V_2 = \emptyset$

证毕

3. 增广路径

对给定流网络 $G=(V,E)$ 和流 f ，**增广路径** p (Augmenting Path) 是其**残存网络** G_f 中一条从源结点 s 到汇点 t 的简单路径。

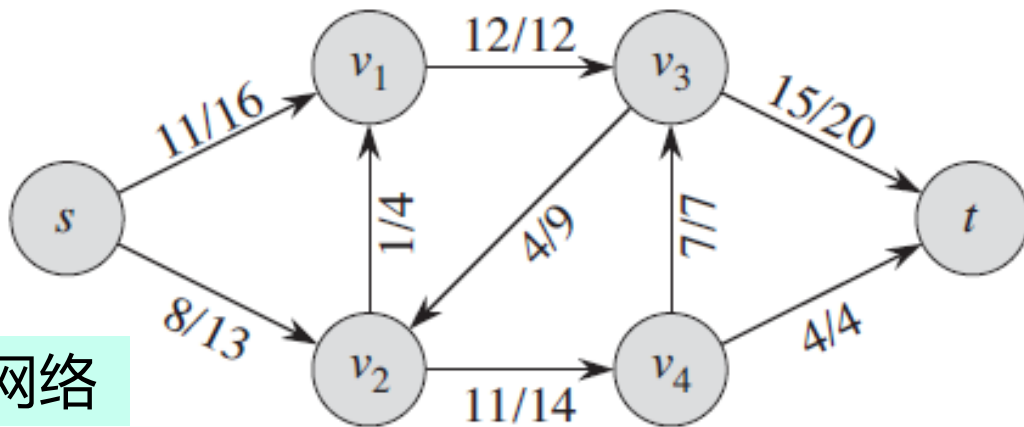
- 对于增广路径上的一条边 (u,v) ，其可增加的流值最大为该边的残存容量 $c_f(u,v)$ 。
- 对于增广路径 p ，能增加的最大流值称为该路径的**残存容量**，记为 $c_f(p)$ ：

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

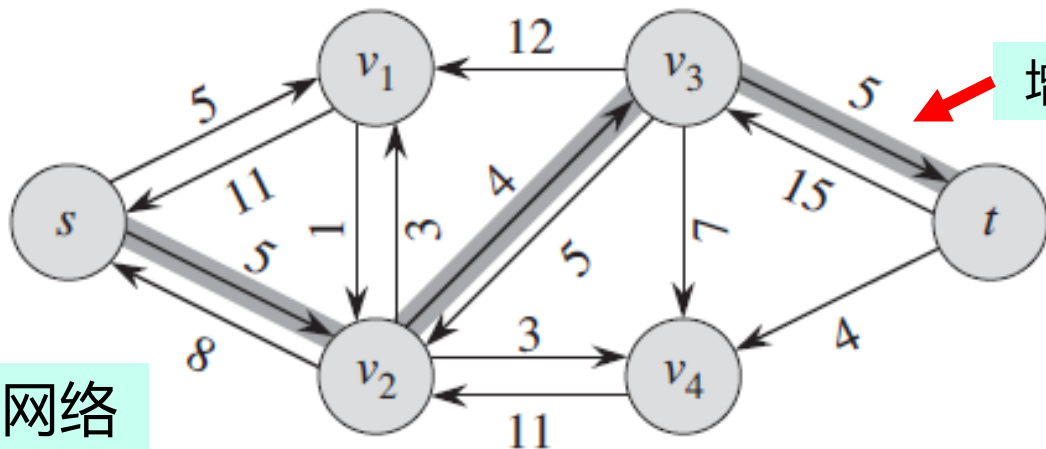
$$> 0$$

示例:

流网络



残存网络



增广路径

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

$$c_f(s, v_2) = 5$$

$$c_f(v_2, v_3) = 4$$

$$c_f(v_3, t) = 5$$

增广路径的残存容量

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

$$= 4$$

引理26.2 设 $G = (V, E)$ 为一个流网络, f 是图 G 的一个流。

记 p 为其残存网络 G_f 中的一条增广路径。

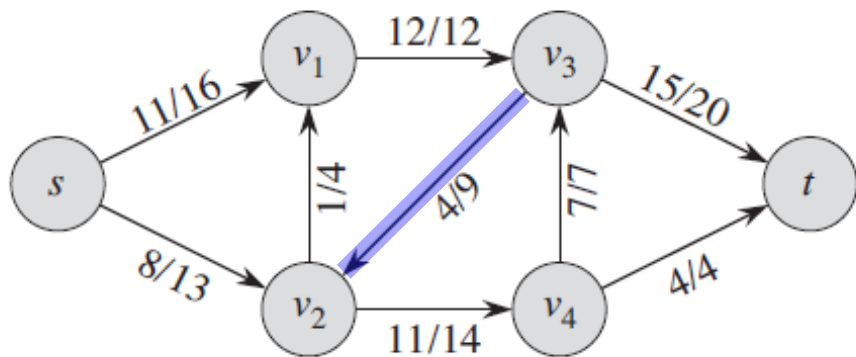
定义一个函数 $f_p : V \times V \rightarrow R$ 如下:

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

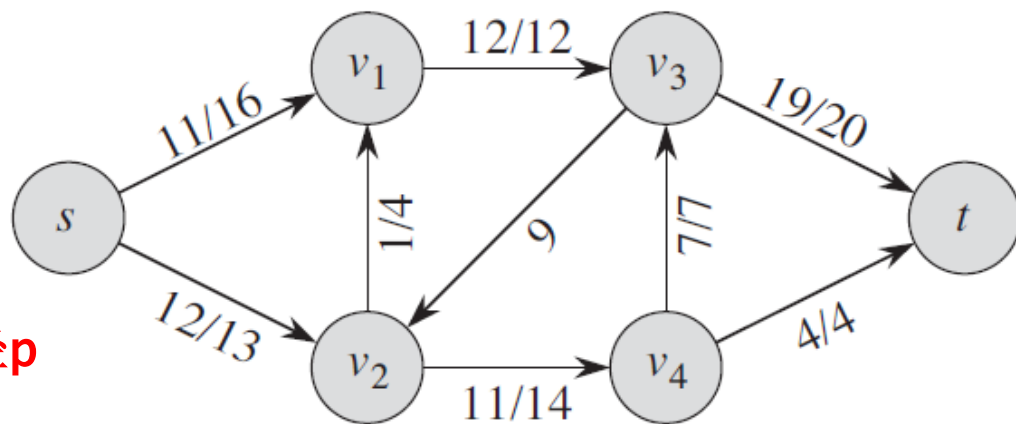
则 f_p 是残存网络 G_f 中的一个流, 其值为 $|f_p| = c_f(p) > 0$ 。

证明: 略 (参见26.2-7)。

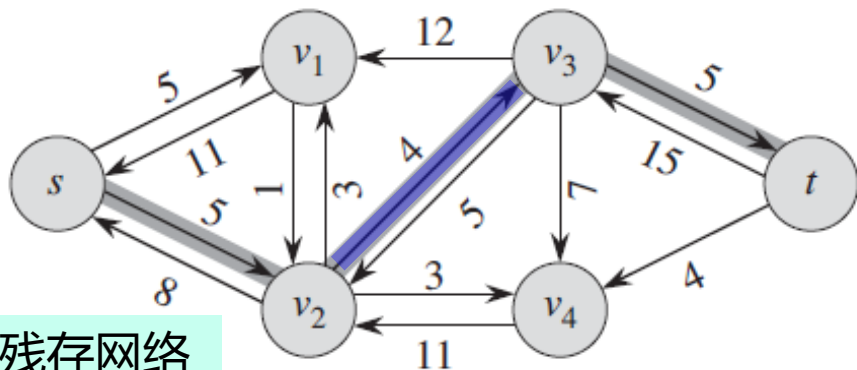
将流 f 增加 f_p 的量，得到的仍是 G 的一个流，且该流
 的值更加接近最大值。



流网络



用 f_p 增加流量后的流网络

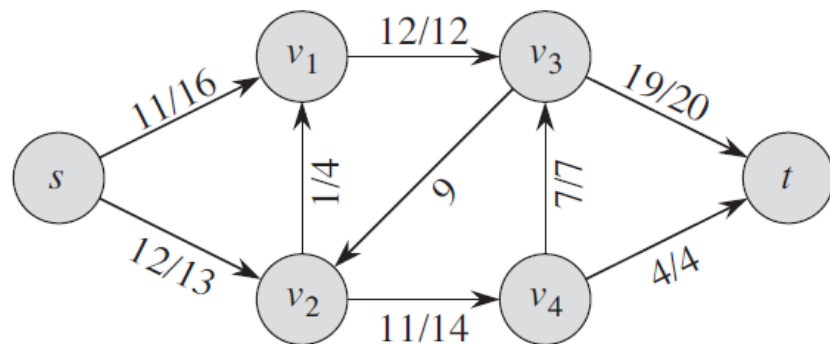
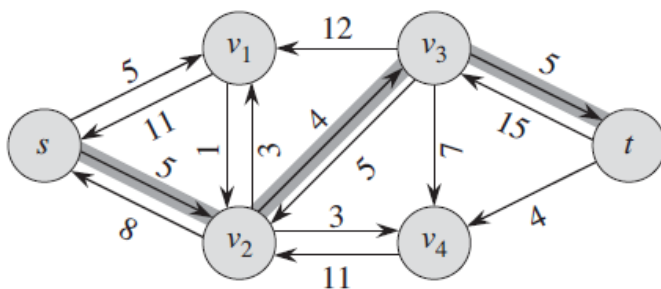
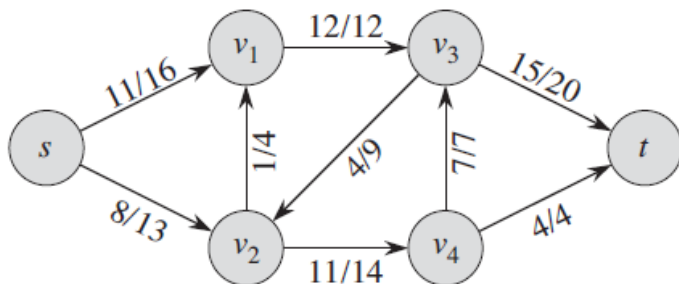


残存网络

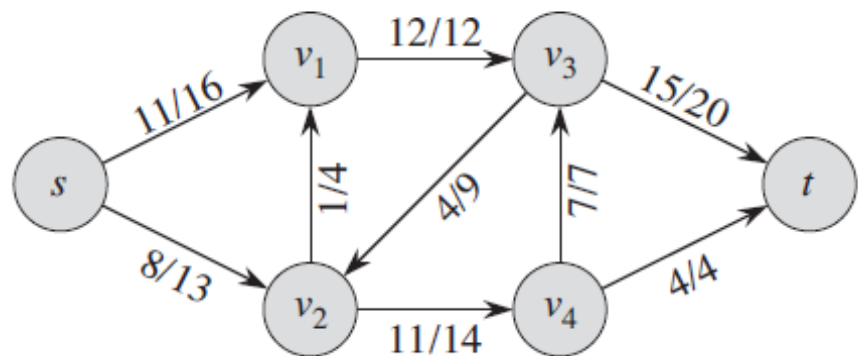
增广路径 p

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 是上式定义的残存网络的流, 假定将 f 增加 f_p 的量, 则函数 $f \uparrow f_p$ 是图 G 的一个流, 其值为 $|f \uparrow f_p| = |f| + |f_p| > |f|$ 。

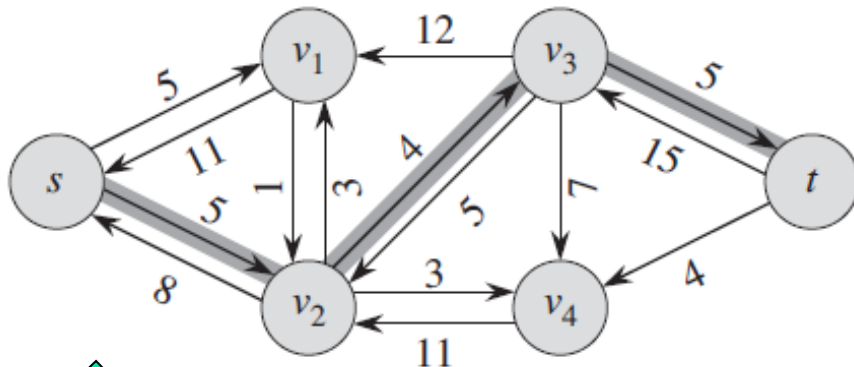
证明: 根据引理26.1和引理26.2可得证。



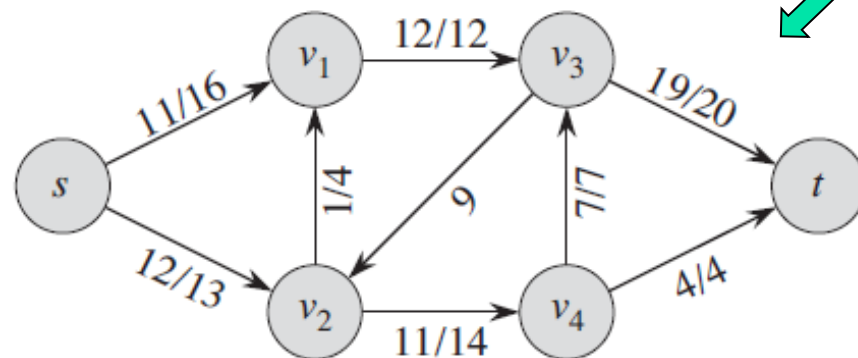
利用残存网络和增广路径实现Ford-Fulkerson方法:



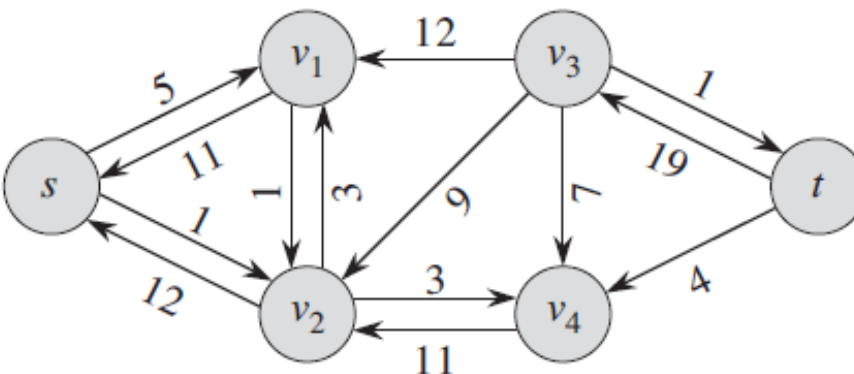
流网络



残存网络1与一条增广路径



残存网络2



增加流值

4. 流网络的切割

给定流网络 $G = (V, E)$ ，源结点为 s ，汇点为 t 。

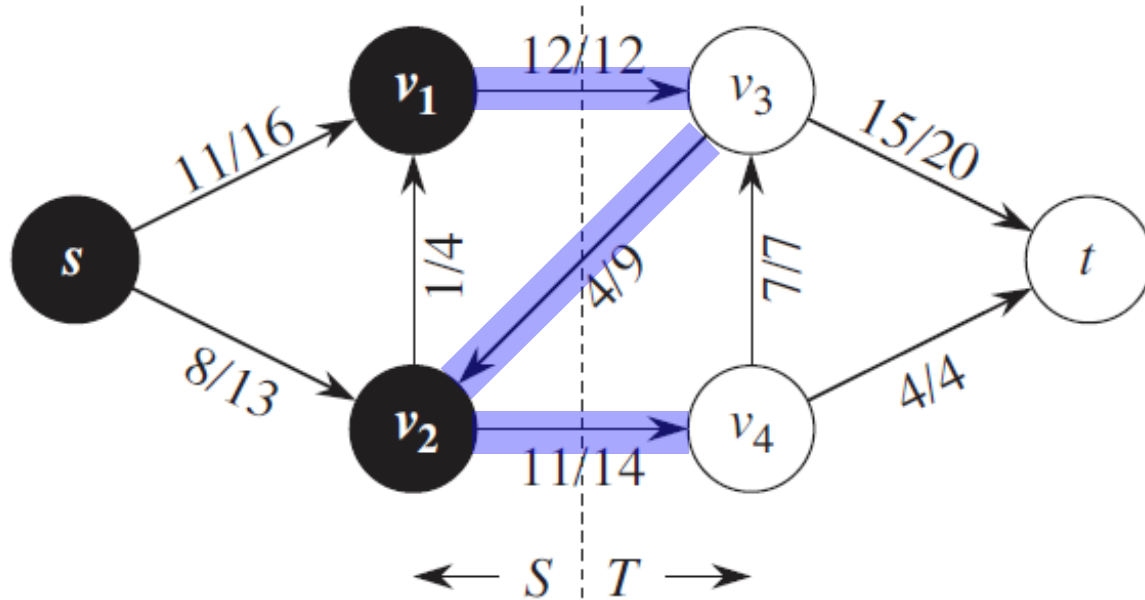
定义一个**切割**(S, T)，将结点集合 V 分成两部分 S 和 $T = V - S$ ，使得 $s \in S, t \in T$ 。

若 f 是 G 上的一个流，**横跨切割**(S, T) 的**净流量** $f(S, T)$ 为：

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

切割(S, T) 的**容量**为： $c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$

最小切割：网络中容量最小的切割。



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

$$f(S, T) = f(v_1, v_3) + f(v_2, v_4) - f(v_3, v_2) = 12 + 11 - 4 = 19$$

$$c(S, T) = c(v_1, v_3) + c(v_2, v_4) = 12 + 14 = 26$$

引理26.4 设 f 为流网络 G 的一个流, 该流网络的源结点为 s , 汇点为 t , 设 (S,T) 为流网络 G 的任意切割, 则**横跨切割 (S,T) 的净流量**为。

证明: 对于任意 $u \in V - \{s, t\}$:

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

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

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

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

将上式针对S和T进行分解，得：

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

化简有：

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

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的值不能超过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。

$$\begin{aligned} |f| &= f(S, T) \\ &= \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u) \\ &\leq \sum_{u \in S} \sum_{v \in T} f(u, v) \\ &\leq \sum_{u \in S} \sum_{v \in T} c(u, v) \\ &= c(S, T) . \end{aligned}$$

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, 该流网络的源结点为 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) 是流网络 G 的某个切割

最大流最小切割定理说明, 流网络 G 的最大流 f 在流的值等于某个切割的容量时达到, 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

(1)→(2)证明: 使用反证法

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 是残存网络 G_f 中由增广路径 p 定义的流。

根据推论1, 对 f 增加流 f_p 所形成的流是 G 中的一个流, 且流值严格大于 $|f|$: $|f \uparrow f_p| = |f| + |f_p| > |f|$, **与 f 是最大流冲突。**

(2)→(3) 证明：假定 G_f 中不包含任何增广路径，即**残存网络 G_f 中不存在从源结点 s 到汇点 t 的路径。**

定义**切割 (S, T)** 如下

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

为证明定理，只需证明 $|f| = c(S, T)$ 。

根据引理26.4 , $f(S, T) = |f|$,

所以只需要证明 $f(S, T) = c(S, T)$

情况1: $(u, v) \in E$, 则有 $f(u, v) = c(u, v)$ 。

否则 $c_f(u, v) = c(u, v) - f(u, v) > 0$, 从而得到 $(u, v) \in E_f$,
并推出 $v \in S$, 与 v 的定义矛盾。

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

否则 $c_f(u, v) = f(v, u) > 0$, 从而得到 $(u, v) \in E_f$,
并推出 $v \in S$, 与 v 的定义矛盾。

情况3: $(v, u) \notin E$ 且 $(u, v) \notin E$, 则 $f(u, v) = f(v, u) = 0$

综上,

$$\begin{aligned} f(S, T) &= \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{v \in T} \sum_{u \in S} f(v, u) \\ &= \sum_{u \in S} \sum_{v \in T} c(u, v) - \sum_{v \in T} \sum_{u \in S} 0 \\ &= c(S, T). \end{aligned}$$

证毕

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black crosshair is positioned on the left side of the slide.

(3)→(1)证明: 根据推论26.5, 对于所有切割 (S,T) , $|f| \leq c(S,T)$ 。

因此, $|f| = c(S,T)$ 意味着 f 是一个最大流。

定理证毕

找最大流:

FORD-FULKERSON-METHOD(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

基本思想: 通过不断增加**可行流值**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**
- (3) **确认无法增加流值, 即得到最大流。**

Ford-Fulkerson算法的细化

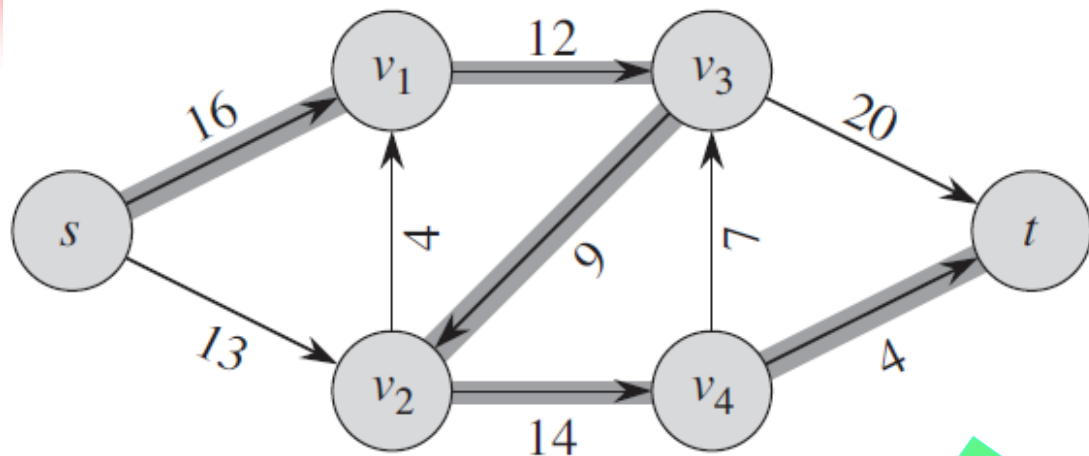
FORD-FULKERSON(G, s, t)

```
1  for each edge  $(u, v) \in G.E$ 
2       $(u, v).f = 0$ 
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               $(u, v).f = (u, v).f + c_f(p)$ 
8          else  $(v, u).f = (v, u).f - c_f(p)$ 
```

➤ 深度优先搜索或者广度优先搜索

基本的Ford-Fulkerson算法

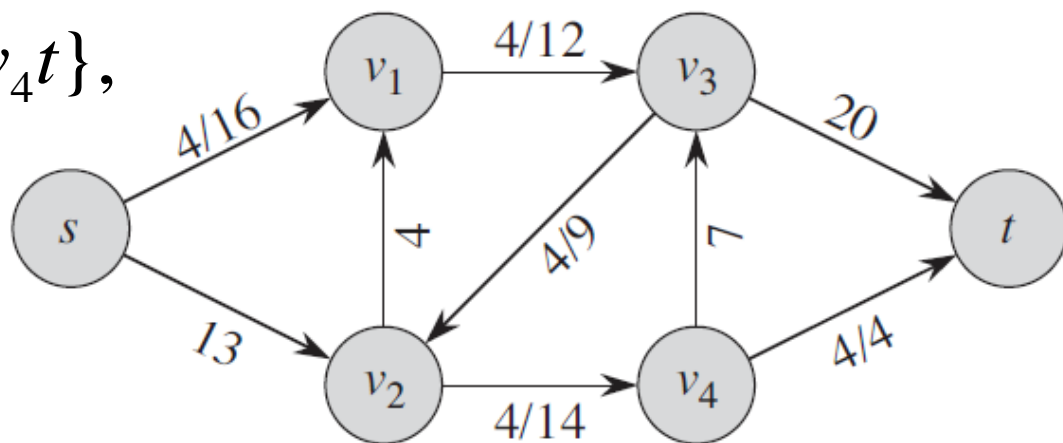
算法运行示例：Iteration 1



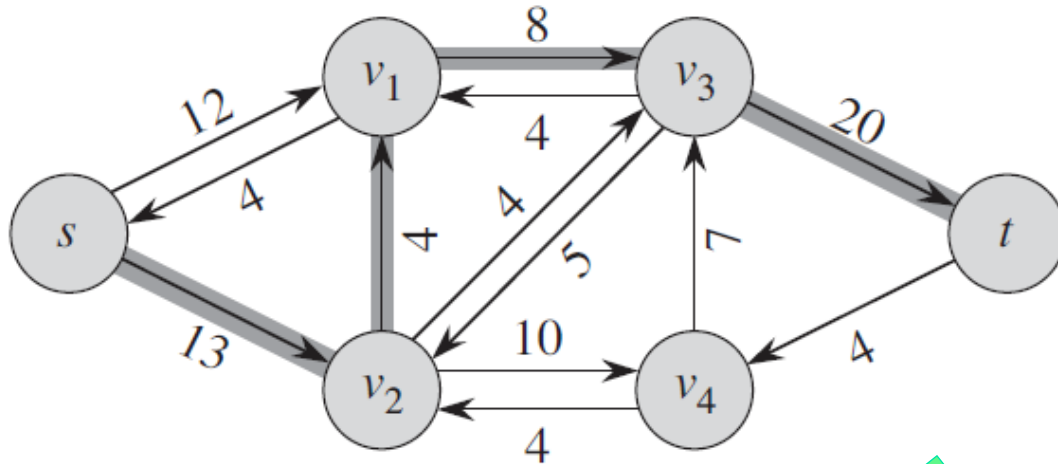
流网络和增广路径

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

$$c_f(p) = 4$$



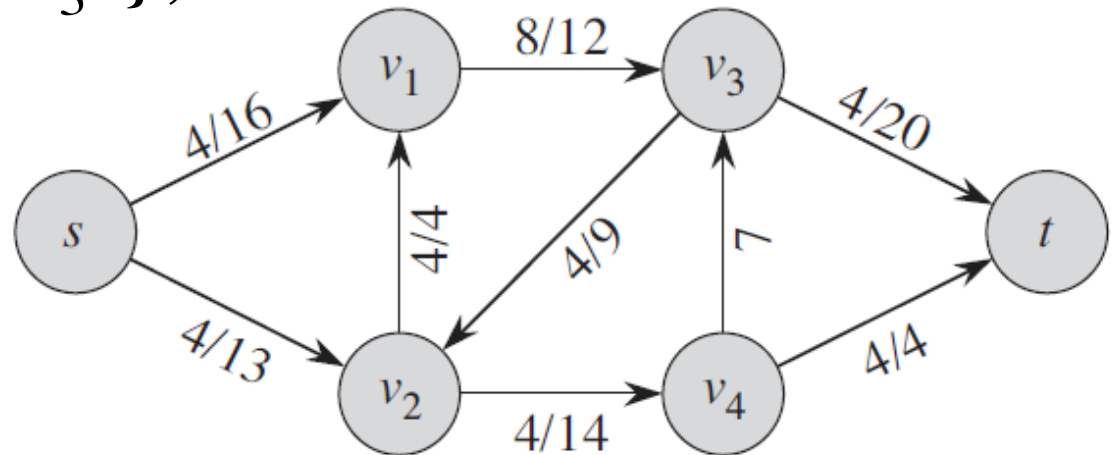
Iteration 2



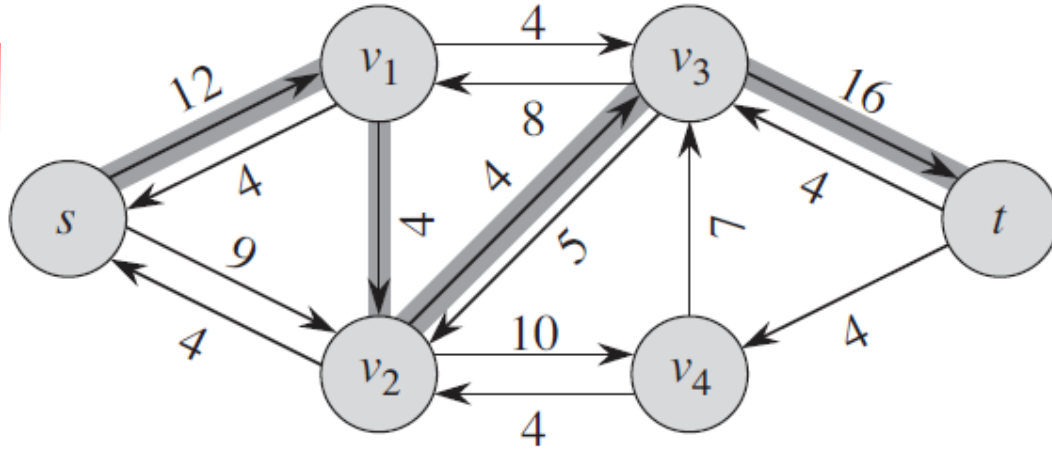
残余网络和增广路径

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

$$c_f(p) = 4$$



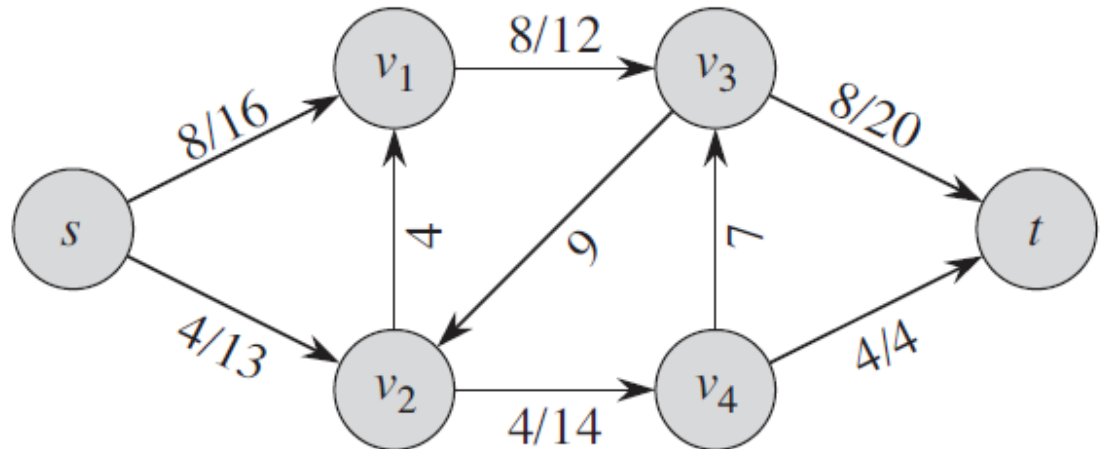
Iteration 3



残余网络和增广路径

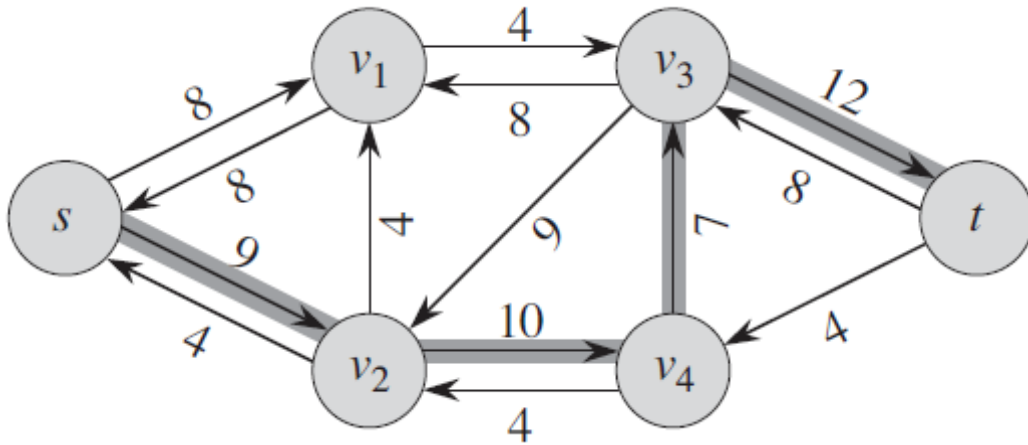
$$p = \{sv_1, v_1v_2, v_2v_3, v_3t\},$$

$$c_f(p) = 4$$



Iteration 4

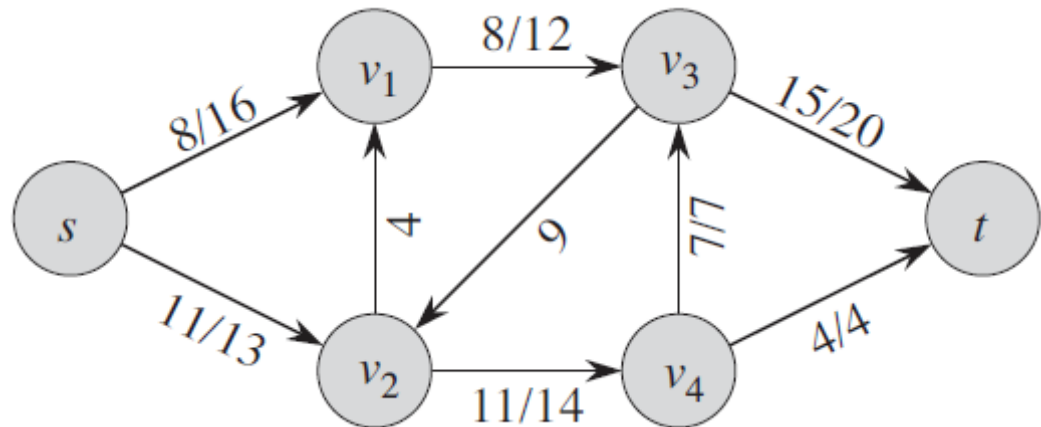
流网



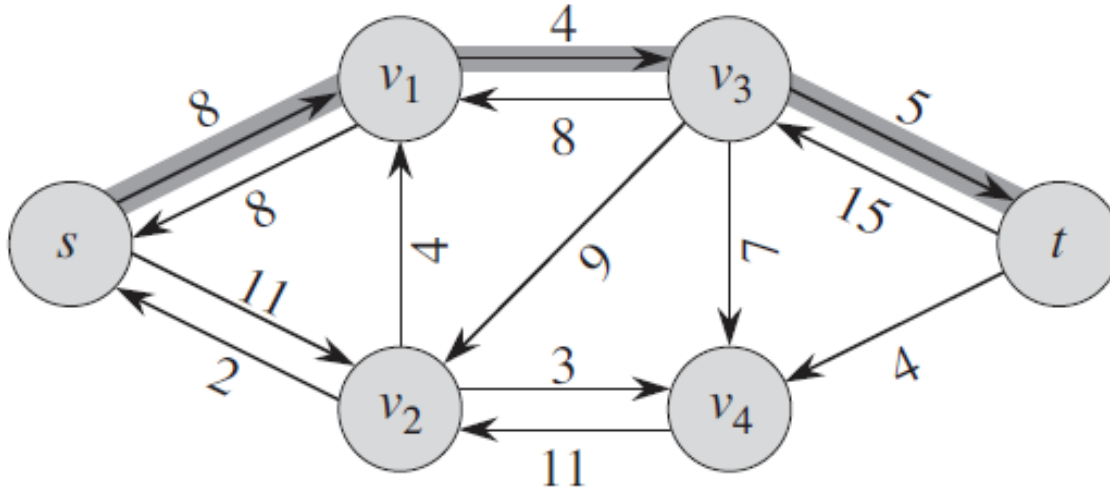
残余网络和增广路径

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

$$c_f(p) = 7$$



Iteration 5

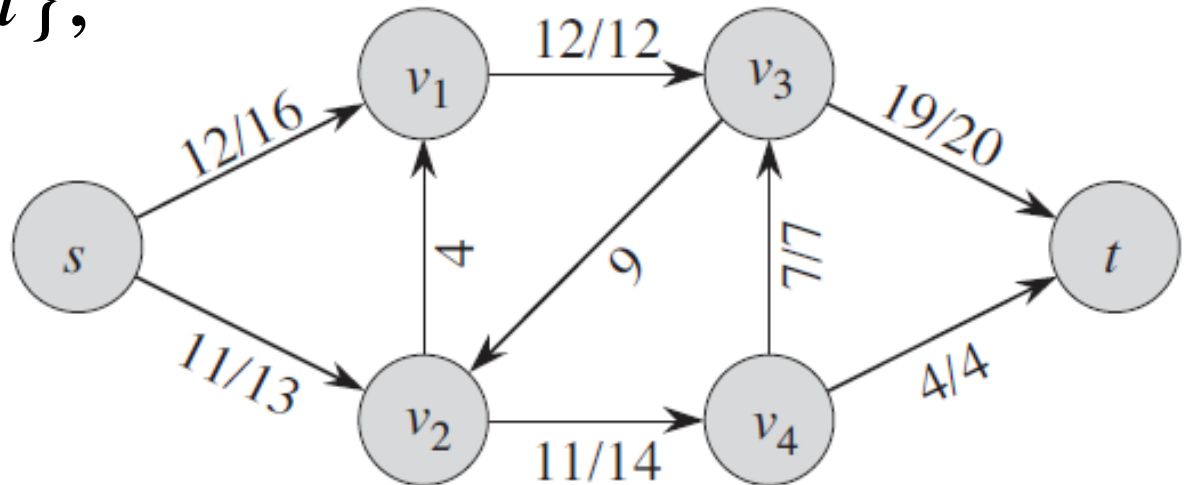


残余网络和增广路径

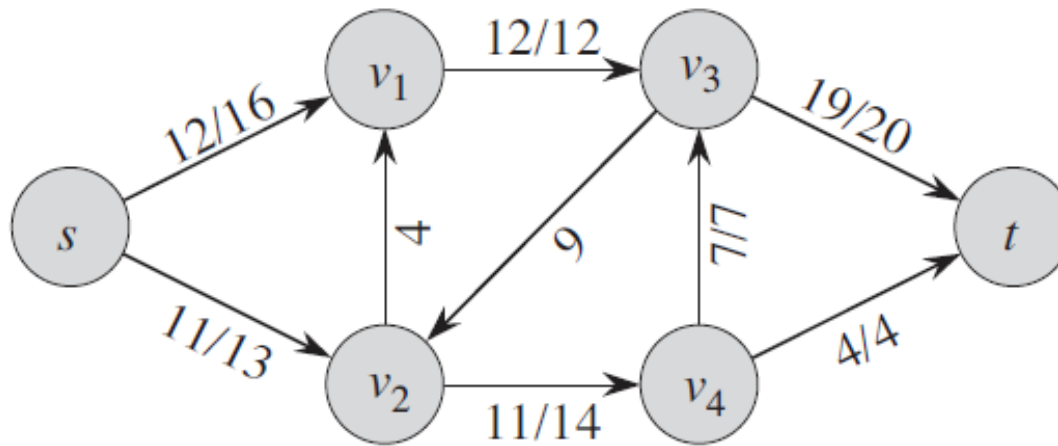


$$p = \{sv_1, v_1v_3, v_3t\},$$

$$c_f(p) = 4$$



无增广路径，得到最大流 $|f| = 23$



Ford-Fulkerson算法复杂性分析

假定：容量为整数。

时间复杂度： $O(|E| \times |f^*|)$,

注：如边的容量是无理数时，Ford-Fulkerson方法可能不能终止，也就是不会得到最大流。

运行时间分析：

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

(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: $O(|E|)$

◆ 计算增广路径: $O(|V| + |E|) \in O(|E|)$

◆ 更新: $O(|E|)$

综和: $O(|E| \times |f^*|)$

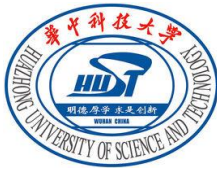
Edmonds-Karp算法

Edmonds-Karp算法的不同：使用广度优先搜索寻找源结点到汇点的最短路径作为增广路径。

FORD-FULKERSON(G, s, t)

```
1  for each edge  $(u, v) \in G.E$ 
2       $(u, v).f = 0$ 
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               $(u, v).f = (u, v).f + c_f(p)$ 
8          else  $(v, u).f = (v, u).f - c_f(p)$ 
```

运行时间： $O(VE^2)$



Edmonds-Karp算法运行时间分析

(1) 找最短路径: $O(E)$ 。

(2) 关键边: $O(VE)$ 。

综合: $O(VE^2)$ 。

证明过程:

令 $\delta_f(u, v)$ 表示残存网络中 G_f 中从结点 u 到结点 v 的最短路径距离。

引理4: 如果Edmonds-Karp算法运行的流网络 $G = (V, E)$ 上, 该网络的源结点是 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, 导致从源结点 s 到 v 的路径距离减小。

设 f 是第一个导致某条最短路径距离减少的流量递增操作之前的流量, f' 是递增操作之后的流量。

设 v 是在流递增操作中最短路径被减小的结点中 $\delta_{f'}(s, v)$ 最小的结点, 根据假设, 应有 $\delta_{f'}(s, v) < \delta_f(s, v)$ 。

下面证明这个不等式不成立。

设 $p = s \cdots \rightarrow u \rightarrow v$ 为残存网络 $G_{f'}$ 中从源结点 s 到结点 v 的一条最短路径, 则:

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

$$\text{对于 } u, \text{ 应有 } \delta_{f'}(s, u) \geq \delta_f(s, u)$$

且 $(u, v) \notin E_f$ 。否则

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

与假设矛盾。



由上可得： $(u, v) \in E_{f'}$ 时， $(u, v) \notin E_f$ 。

(1) $(u, v) \notin E_f$ ：意味着相对流 f ， $f(u, v) = c(u, v)$ ；

(2) $(u, v) \in E_{f'}$ ：意味着相对流 f' ， $f'(u, v) < c(u, v)$ 。

根据**Edmonds-Karp算法**，增广路径是最短路径，增广路径上的一条边 (v,u) 存在于此最短路径上。根据最短路径的性质，应有：

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

所以流量递增操作导致从源结点s到v的路径距离减小不成立。



定理2： 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 (u,v) 称为增广路径 p 上的关键边。

- ◆ 由前一引理已知： **Edmonds-Karp算法中，当流值增加的时候，从s到每个结点的距离会增加。**
- ◆ 证明： **一条边 (u,v) 成为关键边之后，下次再成为关键边的時候，s到u的最短距离至少会增加2。**

设 $u, v \in V$, 且 $(u, v) \in E$ 。

考虑 (u, v) 第一次成为**关键边**时: (u, v) 处于增广路径上, 而


增广路径是最短路径, 所以有: $\delta_f(s, v) = \delta_f(s, u) + 1$

对流增加后, (u, v) 就会从下面的残存网络中消失, 直到某一步从 u 到 v 的流量减小了: 此时, (v, u) 是增广路径上的边, 记此时的流为 f' , 则有

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

根据引理26.7, $\delta_f(s, v) \leq \delta_{f'}(s, v)$, 所以有

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

A decorative graphic consisting of overlapping yellow, red, and blue squares is located in the top left corner.

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

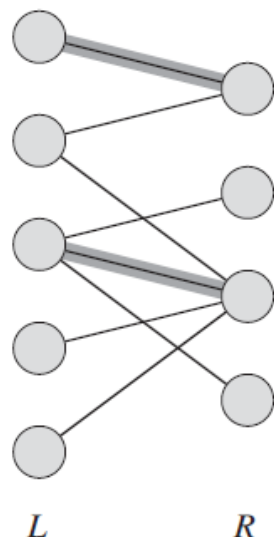
而每次流值增加，都存在一条增广路径，每条增广路径上至少有一条关键边，而每条边都可能成为关键边，所以Edmonds-Karp算法执行流值递增操作的次数至多为 $O(VE)$ 。 ■

最大流算法应用：寻找最大二分匹配

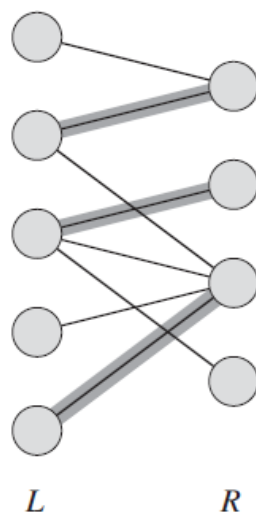
匹配：对无向图 $G=(V,E)$ 的一个**匹配是边的一个子集** $M \subseteq E$ ，使得对于所有的结点 v ，子集 M 中最多有一条边与结点 v 相连。

M 中边的数量称为 M 的**基数**，记为 $|M|$ 。

最大匹配：基数最大的匹配。即，如果 M 是一个最大匹配，则对于任意匹配 M' ，有 $|M| \geq |M'|$



一个匹配

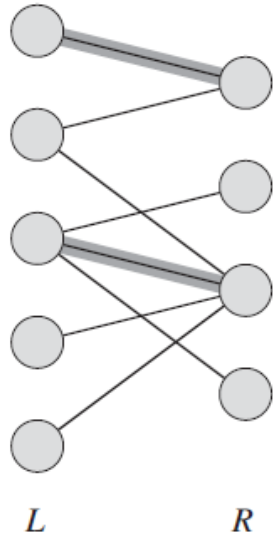


最大匹配

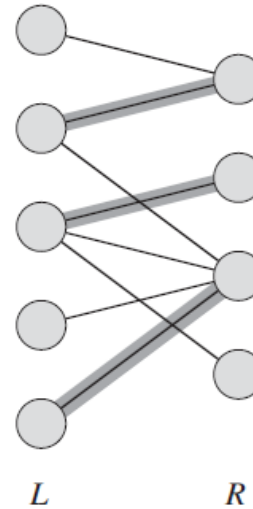
二分图:

结点集合 V 可以划分为两部分 L 和 R , $L \cup R = V, L \cap R = \emptyset$,
边集 E 中所有边横跨 L 和 R , 即对于任意的 $(u, v) \in E$, 有

$$u \in L, v \in R \text{ 或 } v \in L, u \in R$$



二分匹配



最大二分匹配

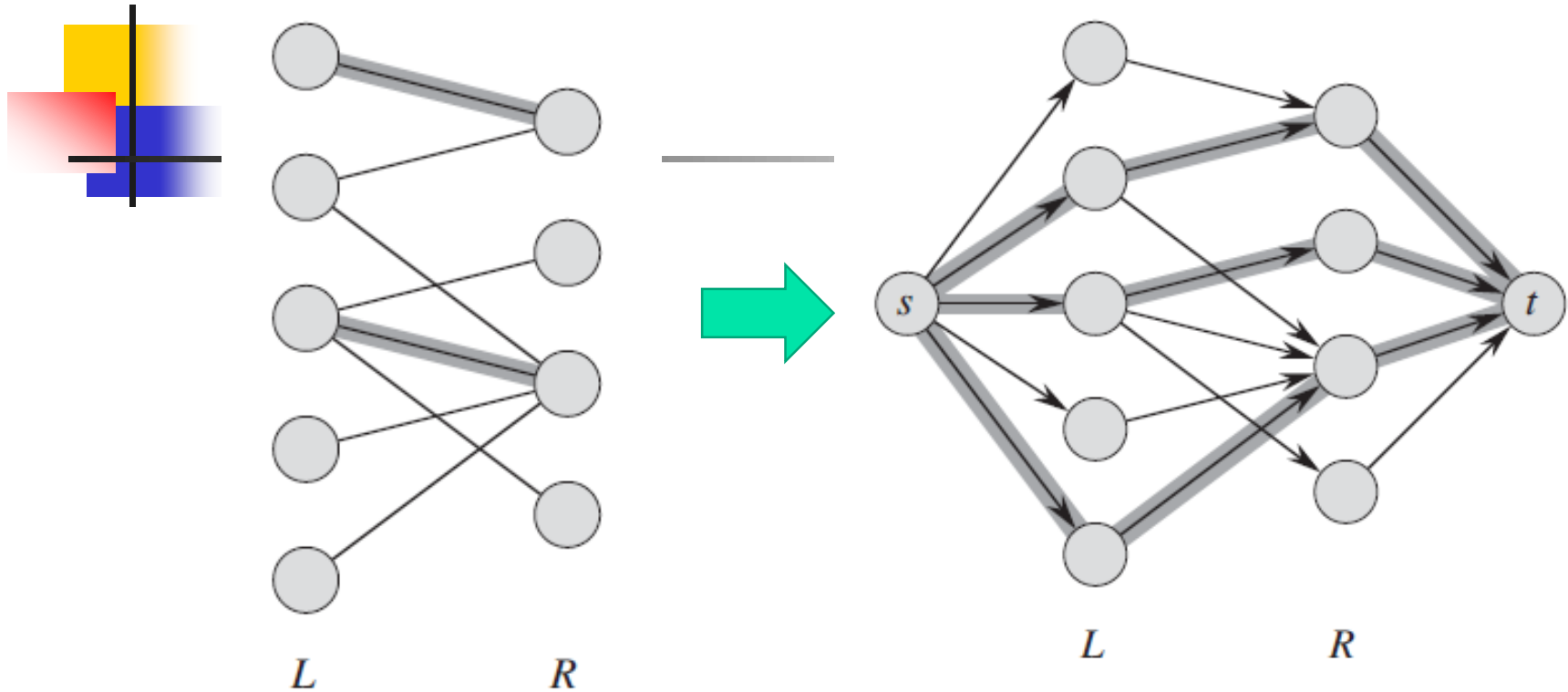
寻找最大二分匹配

转化:

- 新增源结点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

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

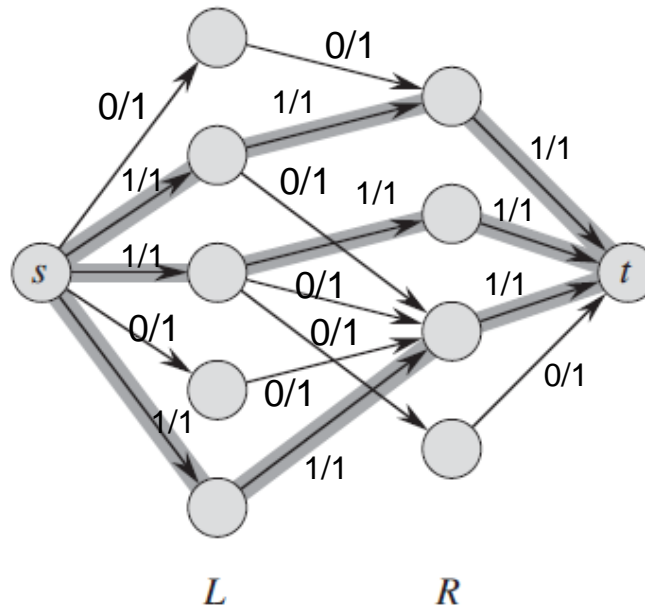
- 定义每条边上的容量为单位容量: $(u, v) \in E'$
 $c(u, v) = 1$



- 不失一般性，假定结点集 V 中的每个结点至少有一条相连的边，
 $|E| \geq |V|/2$ ，则有 $|E| \leq |E'| = |E| + |V| \leq 3|E|$
 所以 $|E'| \in \Theta(|E|)$

寻找最大二分匹配算法

利用Ford-Fulkerson算法求得 G' 中的最大流。流值大于0且在原图中的边将构成最大匹配，而最大匹配的边数就是最大流的流值。



最大匹配=3

(1) 证明原图和转化后的流网络中，匹配和流一一对应，并且匹配的边数对应于流值

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 中匹配，定义 G' 中对应的流 f ：

如果 $(u, v) \in M$ ， $f(s, u) = f(u, v) = f(v, t) = 1$ ；
所有其它属于 E' 的边 (u, v) ， $f(u, v) = 0$ 。

- (1) 可以验证 f 满足容量限制和流量守恒性质（自行验证）。
- (2) $|M|=|f|$?

假定 f 是 G' 中如上所定义的一个整数值流，并设

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

根据 G' 的构造，每个结点 $u \in L$ 只有一条进入的边，即 (s, u) ，其容量为1：根据能量守恒性质，必有一个单位的流出。由于 f 是整数值流，所以 u 不仅最多只能从一个单边流入1单位流量，而且也只能最多从一条边流出。

同样的讨论可应用于 $v \in R$ ， **v 最多有一条带有正值流的入边。**

所以 M 是一个匹配（即对于所有的结点 v ， M 中最多有一条边与之相连）。

从而有 $|M|=|f|$:

根据 f 的定义, 对每个匹配的结点 $u \in L$, 有 $f(s, u)=1$, 而对于每条边 $(u, v) \in E-M$, 有 $f(u, v)=0$ 。

因此, 横跨切割 $(L \cup \{s\}, R \cup \{t\})$ 的净流量 $f(L \cup \{s\}, R \cup \{t\})$ 等于 $|M|$ 。根据引理26.4, 流的值 $|f|=|M|$ 。证毕

(2) 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。

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, 留作练习。

(3) 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:

假定 M 是图 G 中的一个最大匹配, 但流网络 G' 中的流 f 不是最大流。那么 G' 中存在一个最大流 f' , 满足 $|f'| > |f|$ 。

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。同时, f' 有一个对应的匹配 M' , 使得 $|M'| = |f'| > |f| = |M|$, 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。 ■

A decorative graphic consisting of overlapping yellow, red, and blue squares with a black vertical line is positioned on the left side of the slide.

最大匹配 M 可以直接从找到的整数最大流 f 获得，这一过程的时间复杂度是 $O(VE)$ 。

最大流总结

1. 基本概念：**流网络**，**流**，**最大流**等；
2. 解决最大流问题的**Ford-Fulkerson方法**，相关概念有**残存网络**、**增广路径**以及理论基础**最大流最小切割定理**；
3. **Ford-Fulkerson算法**和**Edmonds-Karp算法**，相关性质和证明；
4. 最大流算法的应用：解决**最大二分匹配问题**。

作业：26.1-1, 26.2-3, 26.3-1