算法设计与分析

Computer Algorithm Design & Analysis

吕志鹏

zhipeng.lv@hust.edu.cn

群名称：2023秋 算法设计与分析

群　号： 921525307

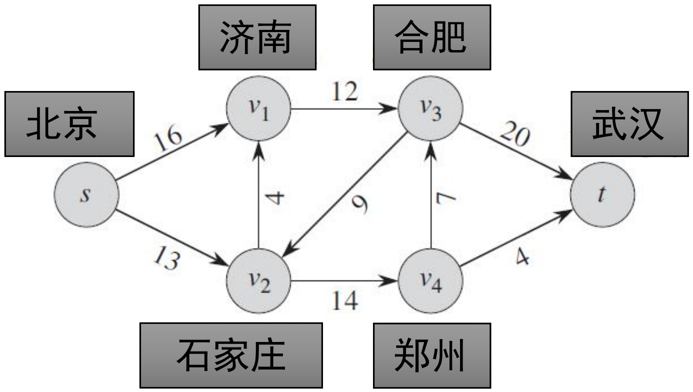
Chapter 26

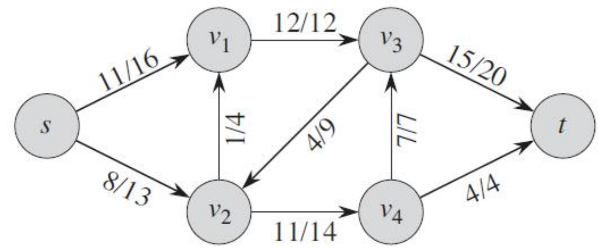
Maximum Flow

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**



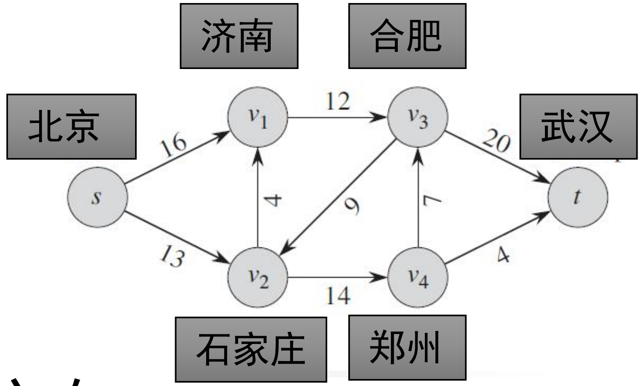


最优解

在物流网络中，从一个城市（称为**源结点**）发送一批货物到另一个城市（称为**汇点**）。假设源结点可以源源不断地提供货物，汇点可以来者不拒地接收货物；路径连接在任意两个城市之间，但路径上有**运输容量有限制**。货物从源结点到汇点可以选择不同的运输路径。

问：**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**。

用带权有向图来表示:



- **结点**表示城市
- 结点间的**有向边**表示运输路径和物流的方向
- 边上的**权重**表示运量限制

—— 这种用来表示“**流**”的图称为“**流网络**”。

重要的约定:

(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,不积累和聚集;

(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。

26.1 流网络

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

（1） 有一个源节点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)；

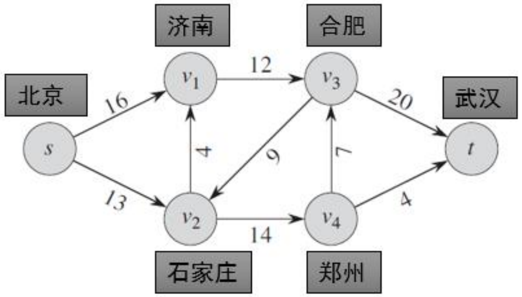
（5） 图中不允许有自循环；

（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） $|E|\geq|V|-1$



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，记为 $f:V\times V\rightarrow R$，并满足以下两条性质：

（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

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

这里，$f(u,v)$ 称为从结点u到结点v的流。

若 $(u,v)\notin E$，记 $f(u,v)=0$，表示从结点u到结点v没有流。

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，用|f |表示：

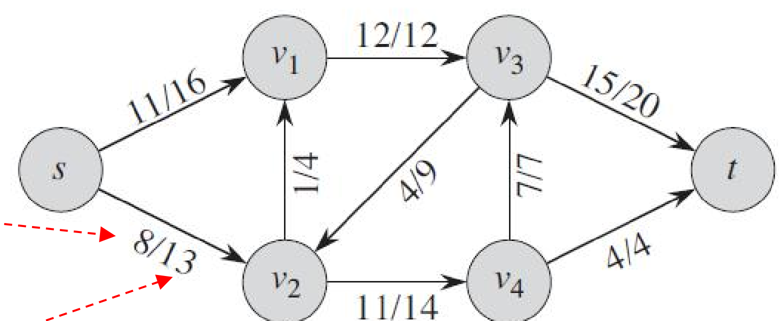
$$|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s)$$

但通常流网络中没有流入源结点的边，因此$\sum_{v \in V} f(v,s) = 0$

示例

边上的流量

边的容量



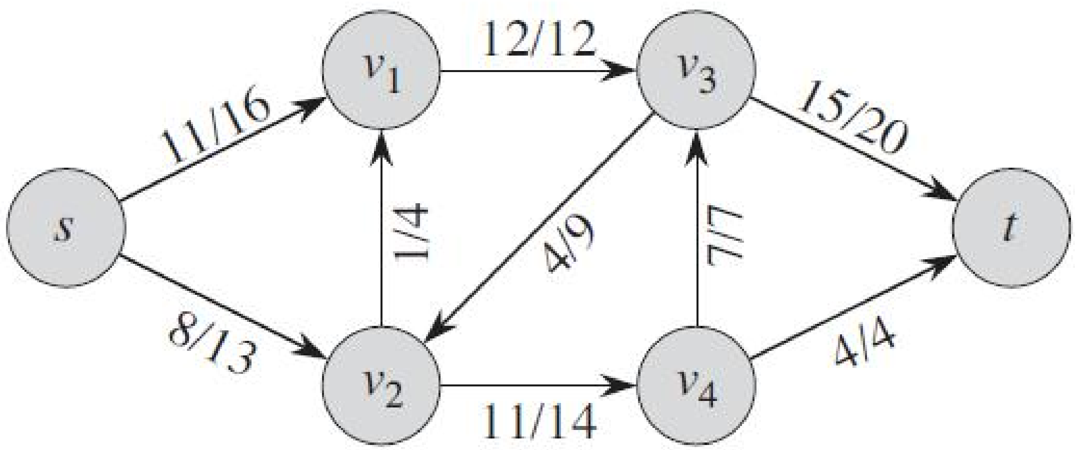
边上的流量

$f(s,v_1)=11$
$f(s,v_2)=8$
$f(v_1,v_3)=12$
$f(v_2,v_1)=1$
$f(v_2,v_4)=11$
$f(v_3,v_2)=4$
$f(v_4,v_3)=7$
$f(v_3,t)=15$
$f(v_4,t)=4$

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

最大流问题：就是在给定的流网络G选中找一个**流值最大的流**的问题

最大流示例



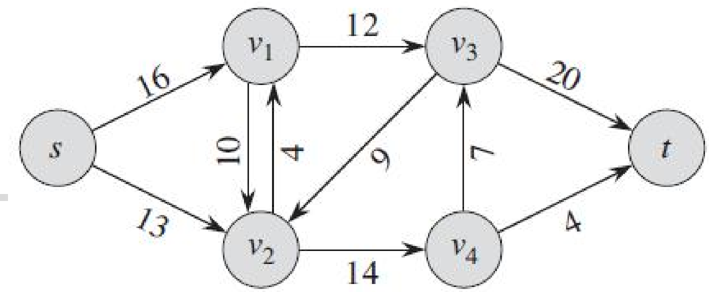
$$|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) = 19$$

上述定义的流网络有两个标准特性：

(1) **无反向边**，或者称为**反平行边**

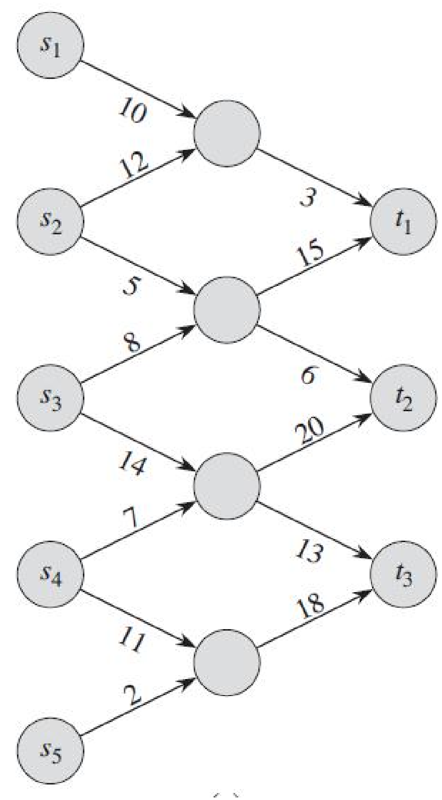
即，如果$(u,v)\in E$，则$(v,u)\notin E$。

这里,(v,u)、(u,v)互为反平行边。



反平行边代表逆向的输入/输出

(2) **只有单一的源结点和汇点**



一般流网络

不满足上述要求的流网络这里视为非标准的**一般流网络**。对于一般流网络需转化为标准流网络进行处理。

（1）具有反平行边的流网络

转化方法：对每一组反平行边 (u,v) 和 (v,u) ，选择其中的一条，

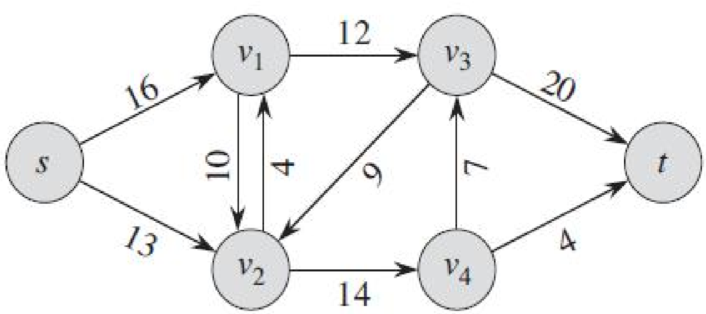
比如 (u,v) ，然后加入一个新的结点v’ ，将其分为

两条边 (u,v') 和 (v',v)，并将两条新加入的边的容量

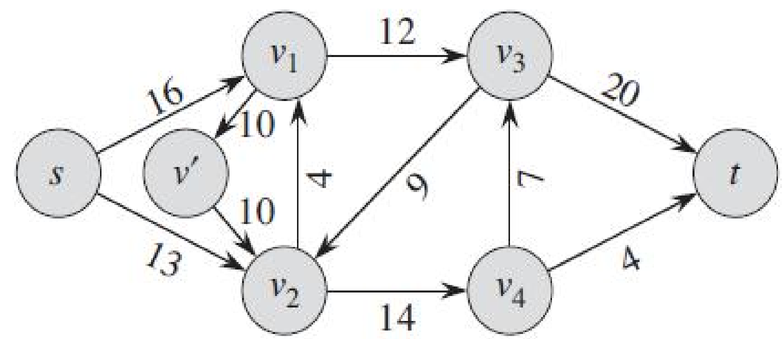
设为被替代掉的边 (u,v) 的容量，即

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

如：



具有反平行边 (v_1,v_2) (v_2,v_1)



替换 (v_1,v_2)

可以证明，**转换后的网络与原网络等价**。留作练习（26.1-1）

(2) 具有多个源结点和多个汇点的网络

◆ **如果流网络中有多个源结点** $\{s_1, s_2, \ldots, s_m\}$，

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 $c(s, s_i) = \infty$，$1 \le i \le m$；

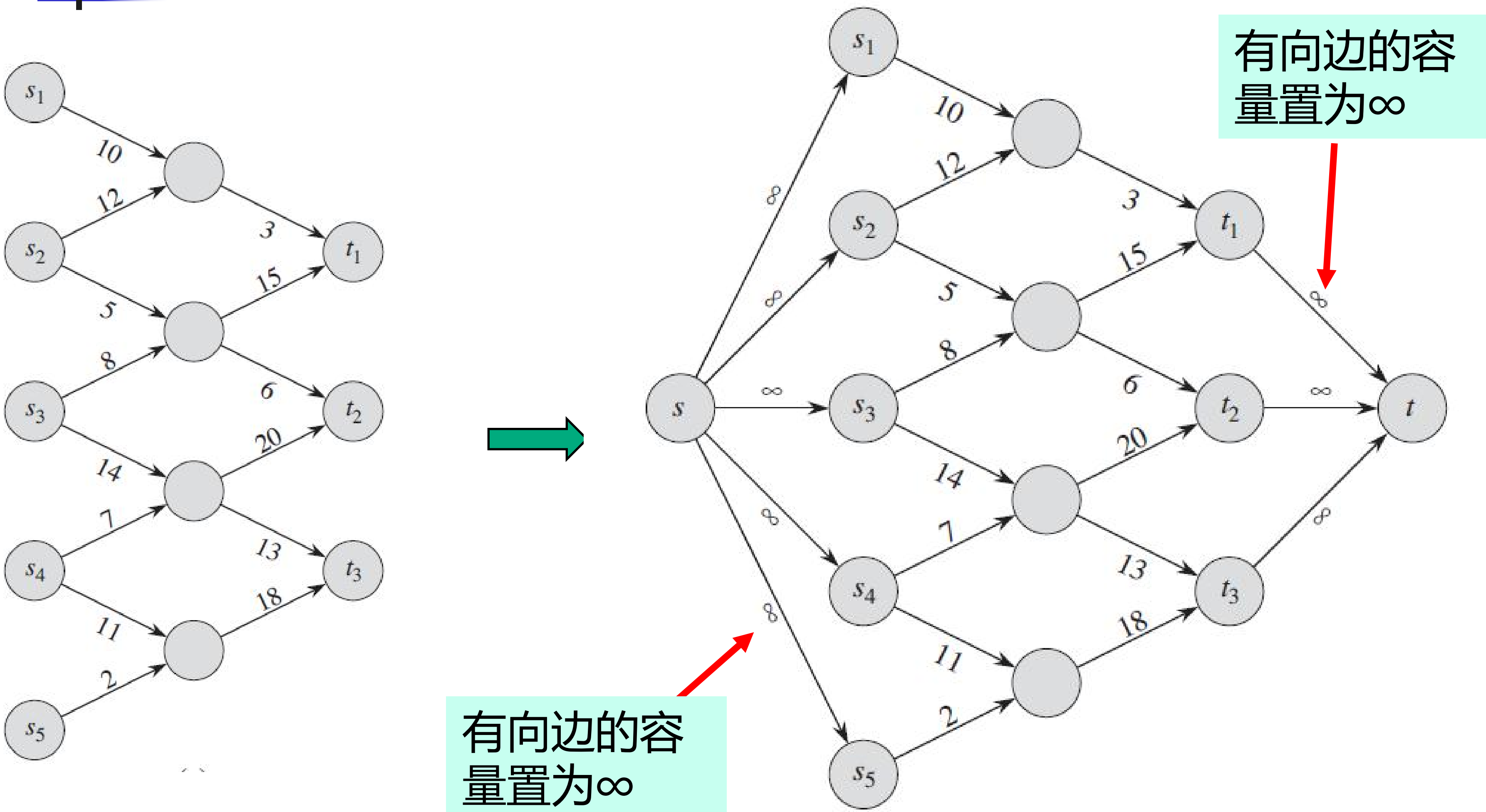
◆ **如有多个汇点** $\{t_1, t_2, \ldots, t_n\}$，

转化方法：加入一个**超级汇点t**，并加入有向边 (t_i, t),

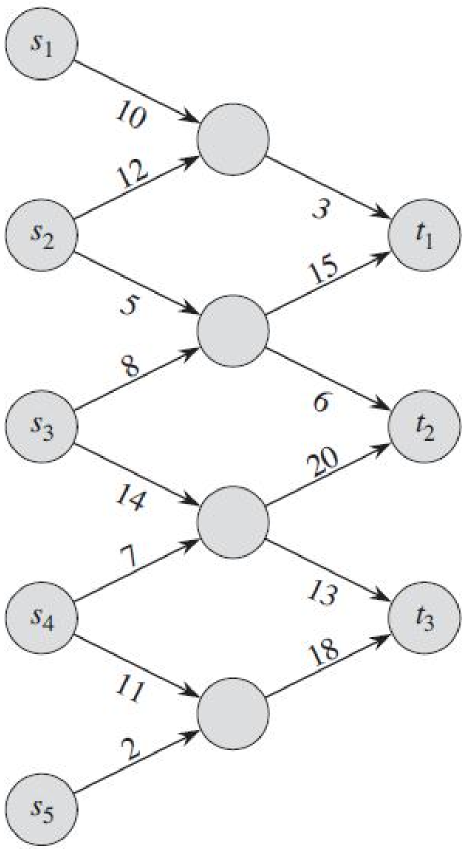
然后令 $c(t_i, t) = \infty$，$1 \le i \le n$。

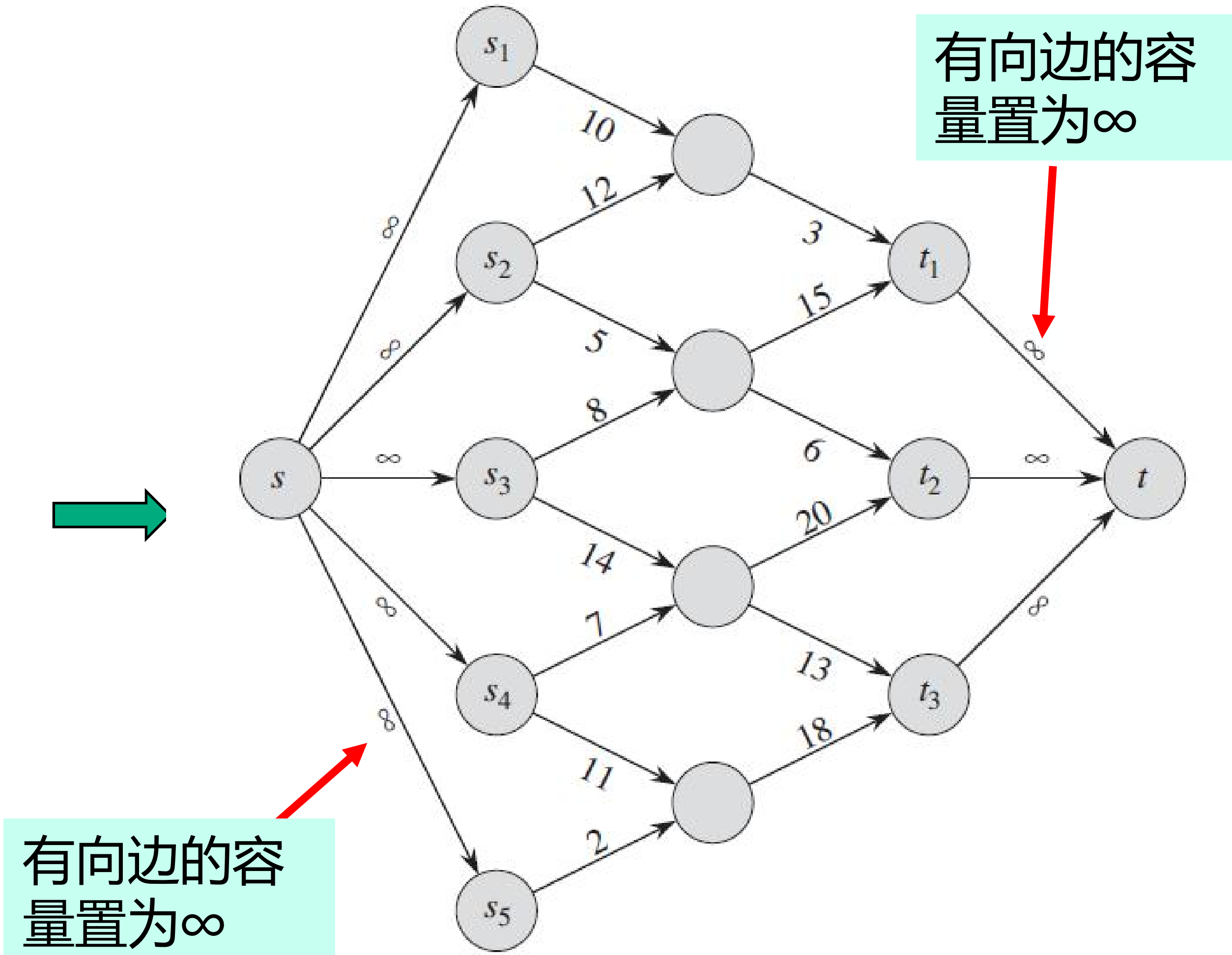
超级源结点s能够给原来的多个源结点s_i提供所需的流量，超级汇点t可以消费原来所有汇点t_i所消费的流量。**可以证明转化后的两个网络是等价的，具有相同的流**。证明留作练习26.1-2。

例：将一个具有5个源结点、3个汇点的一般流网络转换成为一个等价的单源结点单汇点的流网络



例：将一个具有5个源结点、3个汇点的一般流网络转换成为一个等价的单源结点单汇点的流网络





26.2 Ford-Fulkerson方法

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

1. Ford-Fulkerson方法的基本思想

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）确认无法再增加流值时，即得到最大流；

Ford-Fulkerson方法的要点：

（1）对流网络G，在其“**残存网络G_f**”(residual network)中寻找一条“**增广路径p**”(augmenting path)。

（2）如果存在增广路径，则对路径上边的流量进行修改，以增加流网络的流量。

（3）重复这一过程，直到不再存在增广路径为止。

判断是否得到最大流的理论基础是**最大流最小切割定理**，该定理将说明在**算法终止时将获得一个最大流**。

Ford-Fulkerson方法的过程描述

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

寻找增广路径

沿增广路径增加流值

2. 残存网络（Residual Network）

对给定流网络G和流量f，G的**残存网络** G_f 由G中的结点和以下的边组成：

对于G中任意边(u,v)，

(1) 若 $f(u,v) < c(u,v)$，则将**边(u,v)**和它的**反向边**(v,u)都加入G_f，并设其“**残存容量**”为：

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

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

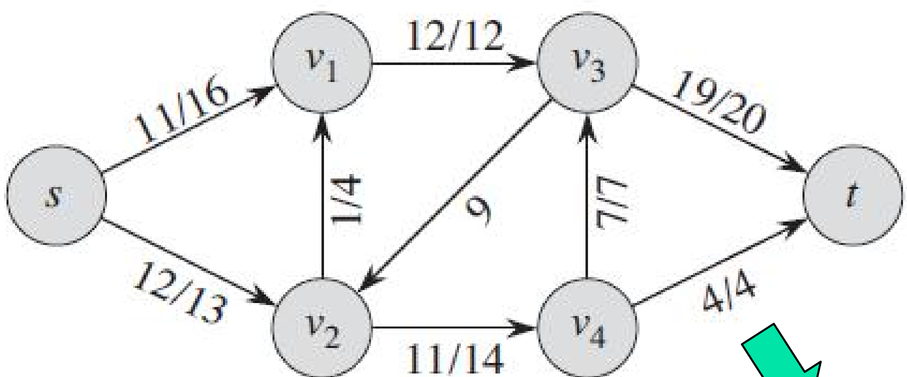
■ 残存容量$c_f(u,v)$反映了边上可以增加流量的空间。

(2) 如果边(u,v)的流量等于其容量，则 $c_f(u,v) = 0$，此时(u,v)不加入G_f，即**只有有多余流量的边才加入G_f。**

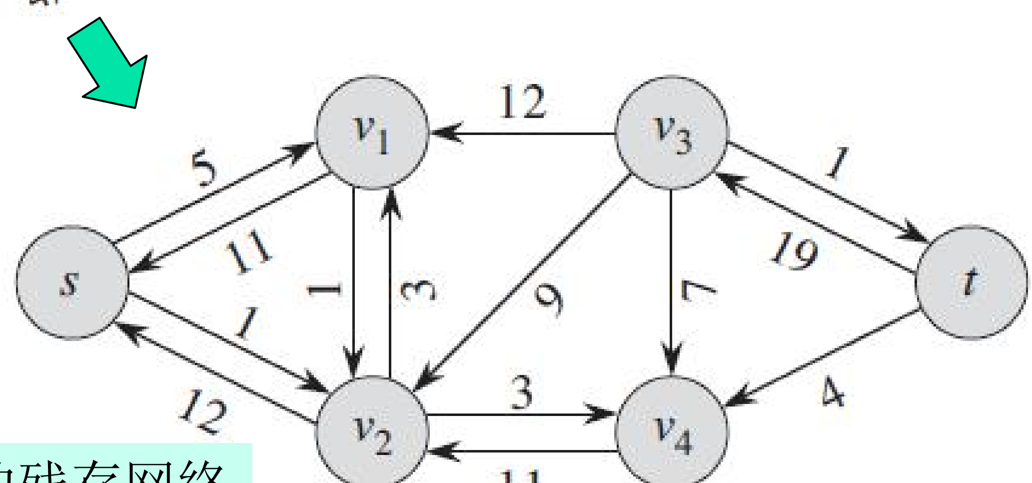
但仍将**反向边(v,u)** 加入G_f，并置 $c_f(v,u) = f(u,v)$。

- 目的也是为对一个正流量 $f(u,v)$ 进行缩减，且最多减少 $f(u,v)$。

例：



一个流网络G



G的残存网络

残存容量

设流网络 $G=(V,E)$，f是G中的一个流，$u,v\in V$

◆ 定义边(u,v)的**残存容量** $c_f(u,v)$ 为:

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

残存网络：由f所诱导的图G的残存网络记为 $G_f=(V,E_f)$，

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

即**G_f由残存流量大于0的边构成**，且$|E_f|\le 2\,|E|$。

残存容量

设流网络$G=(V,E)$，f是G中的一个流，$u,v\in V$

◆定义边(u,v)的**残存容量**$c_f(u,v)$为:

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

残存网络：由f所诱导的图G的残存网络记为$G_f=(V,E_f)$,

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

即**G_f由残存流量大于0的边构成**，且$|E_f|\le 2\,|E|$。

- **残存网络G_f类似于一个容量为c_f的流网络**，但**G_f不满足流网络的定义**：因为G_f可能包含边(u,v)和它的反向边(v,u)。

- **但除此之外，残存网络和流网络具有同样的性质**。

因此，也可以**在残存网络中定义一个流**：

设f是G的一个流，记f'是残存网络G_f中的一个流

- f'针对残存网络G_f中的容量c_f定义且满足流的性质。
  - 容量限制、流量守恒
- f'指出这样一个**路线图**：如何在原来的流网络中增加流。

定义$f \uparrow f'$为流f' **对流f的递增**（augmentation），是一个从$V \times V$到R的函数：

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

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

在残存网络中将流量发送到反向边上等同于在原来的网络中缩减流量。在残存网络中将流量推送回去称为抵消操作(cancellation)，其意义在于调整流网络中总流量的分布。

注：$f \uparrow f'$最终成为原网络中的一个流，是基于f' 对原网络中流f调整的结果。见下。

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：$|f\uparrow f'|=|f|+|f'|$

引理26.1说明：如果在G' 中能够找到一个合法的流f'，则可以用f'对G上原来的流f进行调整，调整的结果是：

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

如果$|f'|>0$，则可以达到对f递增的效果。

引理26.1的证明思路：

首先，证明$f \uparrow f'$是图G的一个流，即$f \uparrow f'$对E中的每条边满足容量限制，且对每个结点$V-\{s,t\}$满足流量守恒性质。

其次，证明流量值公式成立。

流必须非负且不能超过给定容量的限制

对于非源结点和汇点的结点，流入等于流出

证明过程如下：

首先，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 $f'(v,u) \le c_f(v,u) = f(u,v)$

因此，

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

即，$f \uparrow f'$ 是非负值。

此外，

$$
\begin{aligned}
(f \uparrow f')(u,v) & \\
&= f(u,v) + f'(u,v) - f'(v,u) && \text{（根据定义）} \\
&\le f(u,v) + f'(u,v) && \text{（根据流值非负）} \\
&\le f(u,v) + \underline{c_f(u,v)} && \text{（残存网络容量限制）} \\
&= f(u,v) + \underline{c(u,v) - f(u,v)} && \text{（根据残存容量的定义）} \\
&= c(u,v).
\end{aligned}
$$

即，递增后，流 $f \uparrow f'$ 也不超过容量的限制。

容量限制：对任一结点对u和v，满足 $0 \le f(u,v) \le c(u,v)$

(2) 流量守恒

因为f和f'均遵守流量守恒性质，所以对所有的$u \in V-\{s,t\}$结点，有：

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

流量守恒：对于 $u \in V-\{s,t\}$ ，有 $\sum_{v\in V} f(v,u) = \sum_{v\in V} f(u,v)$

其次证明：$|f \uparrow f'| = |f| + |f'|$

证明：

定义 $V_1 = \{v : (s,v) \in E\}$

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

V₁是所有从源结点有边可到达的结点的集合

V₂是所有有边通往源结点s的结点的集合

由于(s,v)和(v,s)不可能同时存在于流网络中，所以有：

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

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

注：若 $(w,x) \notin E$ ，则 $|f \uparrow f'(w,x)| = 0$ 。

根据 $f\uparrow f'$ 的定义，重组上式有，

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

注：$V_1 \cup V_2 \subseteq V$

证毕

$|f\uparrow f'| = |f| + |f'|$ 给出了对G的流增加的方法，如何找 f'？

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。

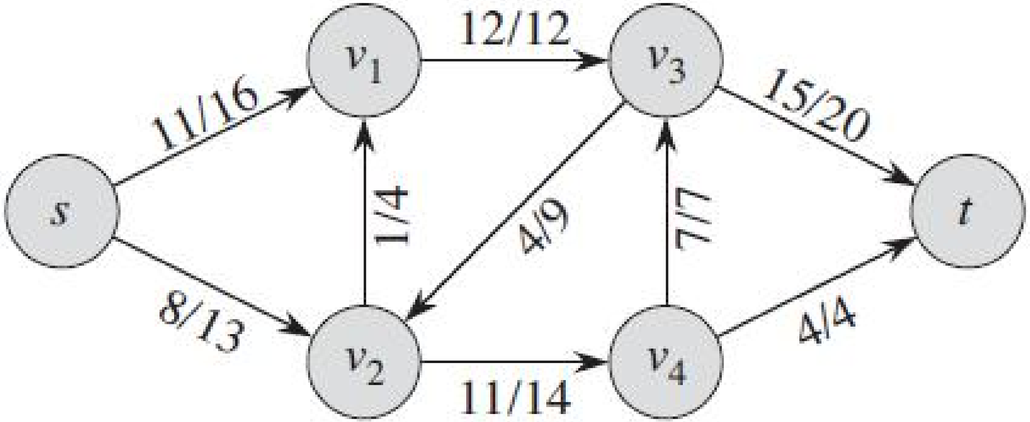
- 根据残存网络的定义，对于增广路径上的一条边(u,v)，其可增加的流值最大为该边的残存容量$c_f(u,v)$ 。
- **对于一条增广路径p，能增加的最大流值称为该路径的残存容量**，等于**p上所有边残存容量的最小值**，即：

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

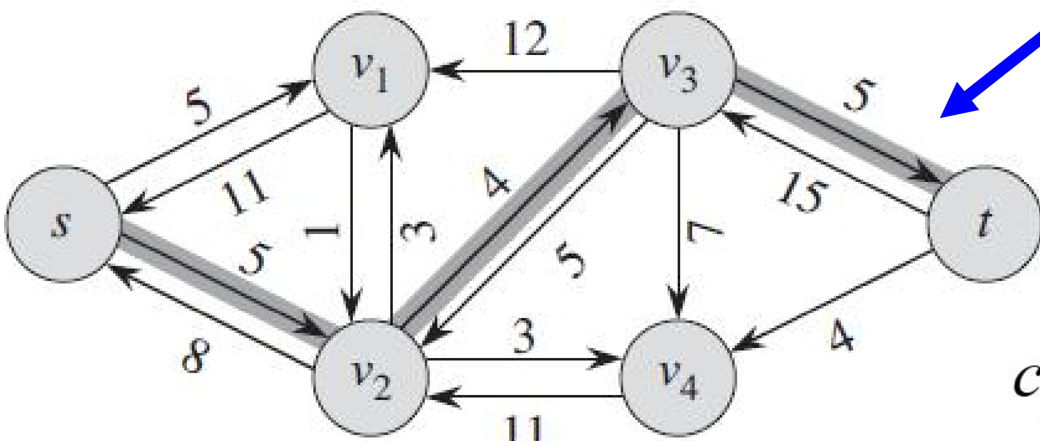
注： $c_f(p) > 0$

残存容量最小的边是瓶颈，增广路径的流量受其限制。

示例:



流网络



残存网络

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

边的残存容量

$c_f(s, v_2) = 5$

$c_f(v_2, v_3) = 4$

$c_f(v_3, t) = 5$

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

增广路径的残存容量

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 $f_p:V\times V\to R$ 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 $|f_p|=c_f(p)>0$。

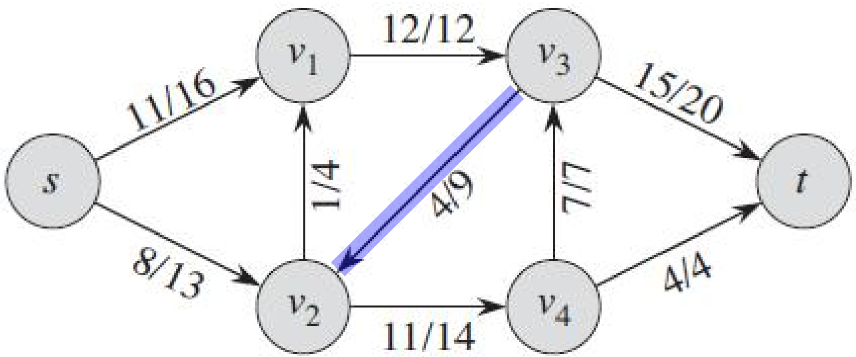
引理26.2告诉我们，G的残存网络G' 中，一个合法的流f'怎么去找：**f_p就是这样的一个合法的流。**

所以可以用f_p为f进行递增计算。

证明：略（参见26.2-7）。

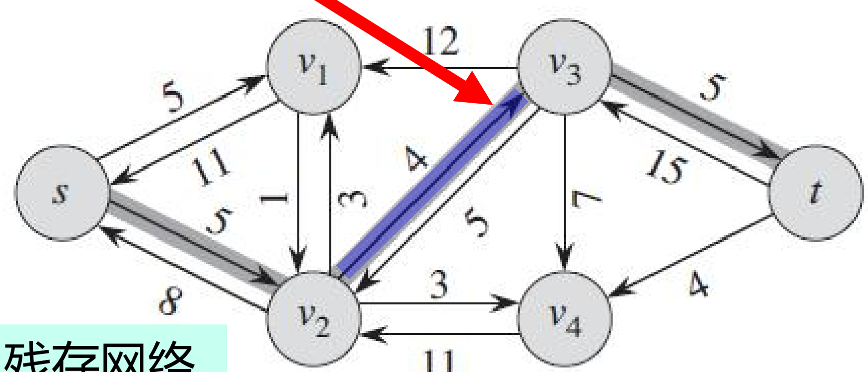
f_p **的作用**：将流 f 增加 f_p 的量，得到的仍是G的一个流，

且**该流的值更加接近最大值。**

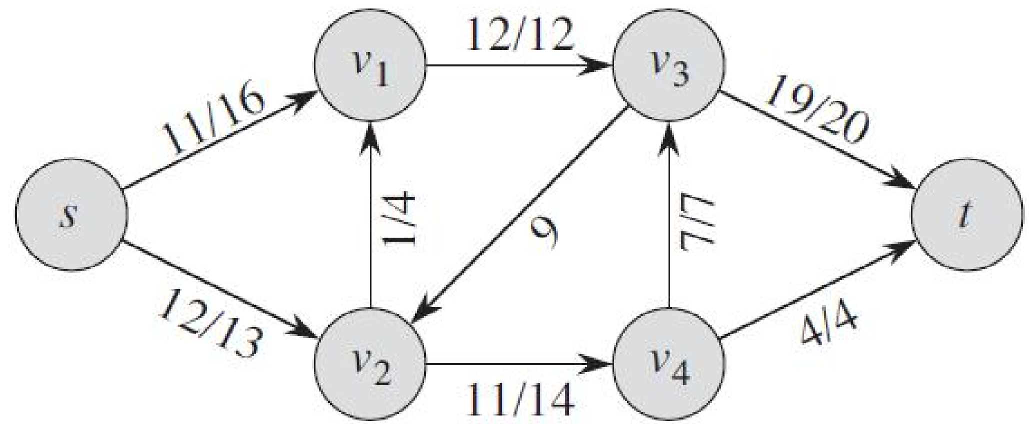


流网络

增广路径



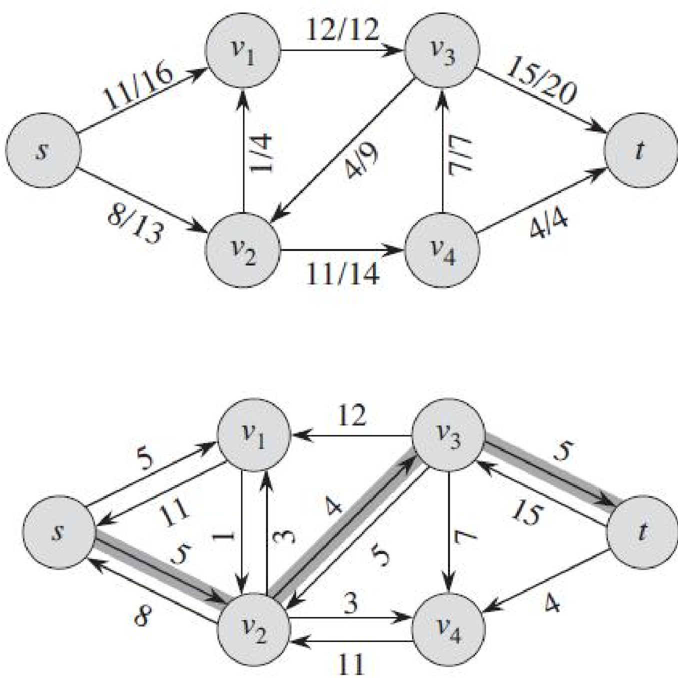
残存网络

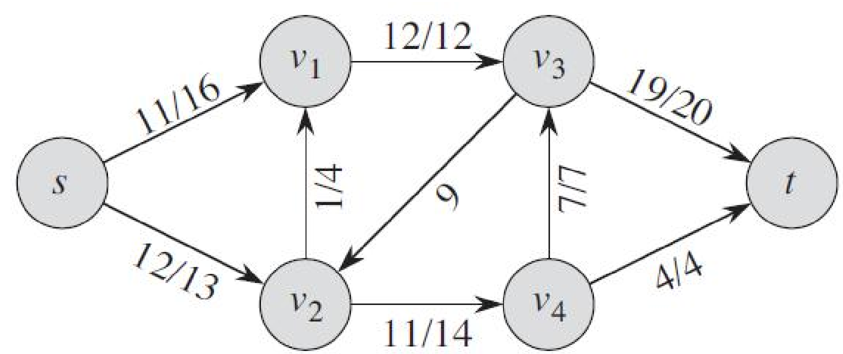


用 f_p 增加流量后的流网络

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

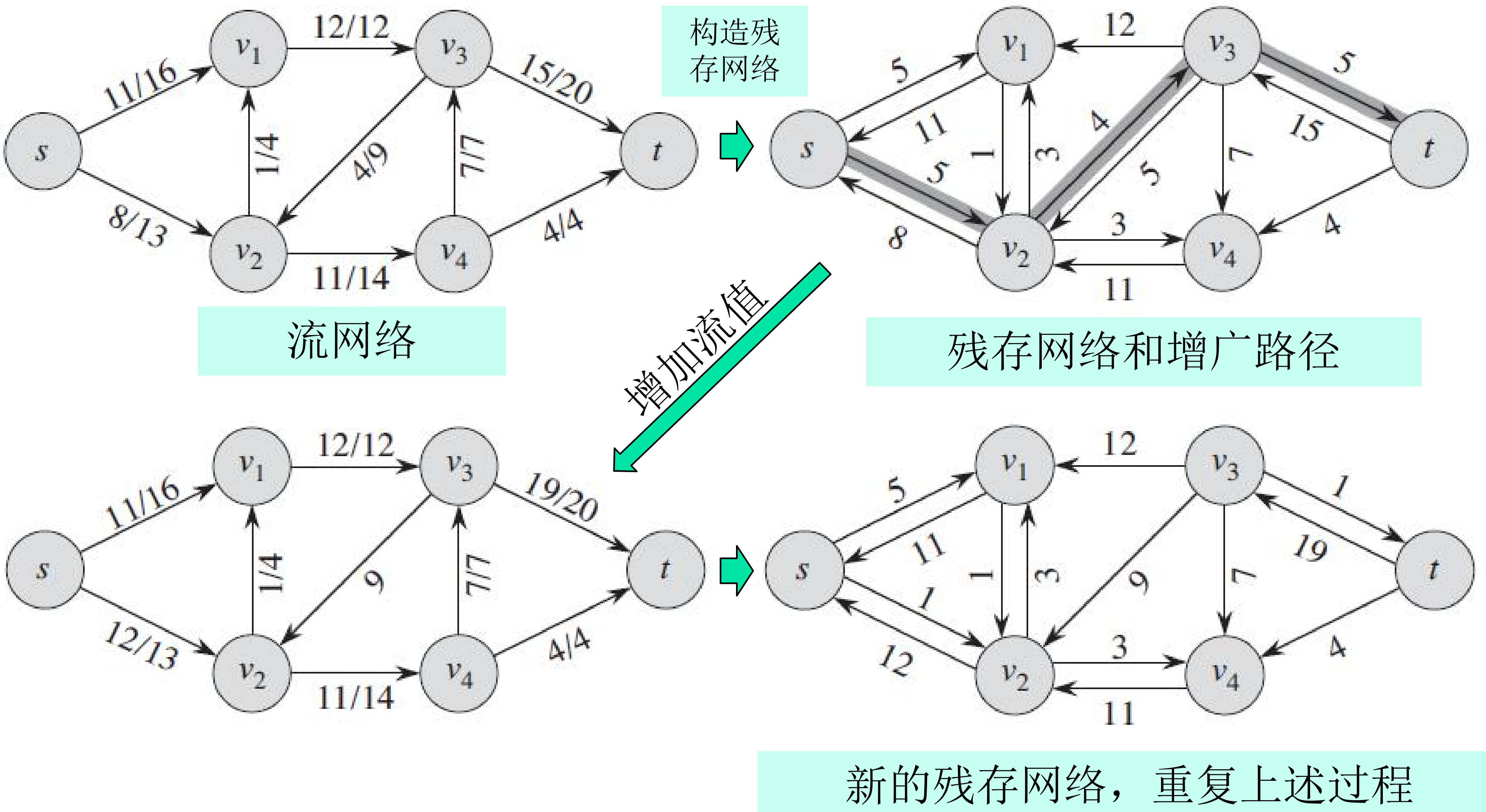
证明：根据引理26.1和引理26.2可得证。





利用残存网络和增广路径实现Ford-Fulkerson方法

示例：



残存网络和增广路径总结

- **已经解决的问题**：**如何增加流值——利用增广路径。**
- **未解决的问题**：如何判断算法终止时，确实**找到了最大流呢**？

——利用**最大流最小切割定理进行判定**

最大流最小切割定理

- **建立最大流和切割容量之间的关系**，从而建立最大流和残存网络增广路径上的残存容量之间的关系。

一个流是最大流当且仅当**其残存网络中不包含任何增广路径**。

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** $s \in S, t \in T$。

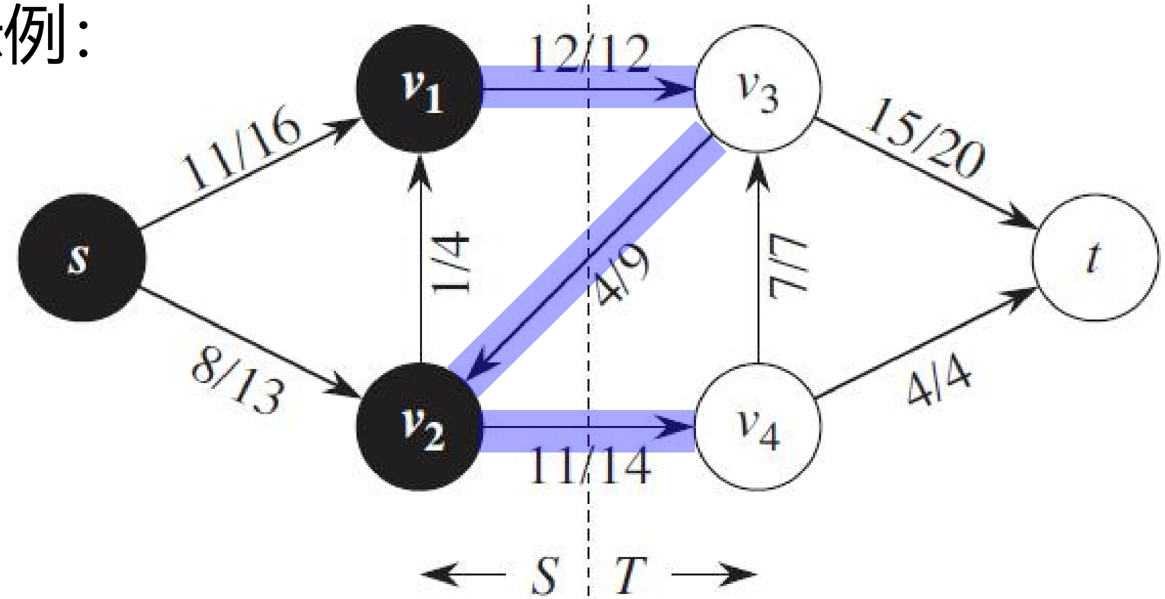
若f是G上的一个流，定义横跨切割(S,T)的净流量f(S,T)为：

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

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

□ **最小切割**：一个网络的最小切割是**网络中容量最小的切割**。

切割的示例：



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

横跨切割的净流量：$f(S,T)=f(v_1,v_3)+f(v_2,v_4)-f(v_3,v_2)=12+11-4=19$

净流量的计算是从S到T的流量减去从T到S的反方向的流量

该切割的容量：$c(S,T)=c(v_1,v_3)+c(v_2,v_4)=12+14=26$

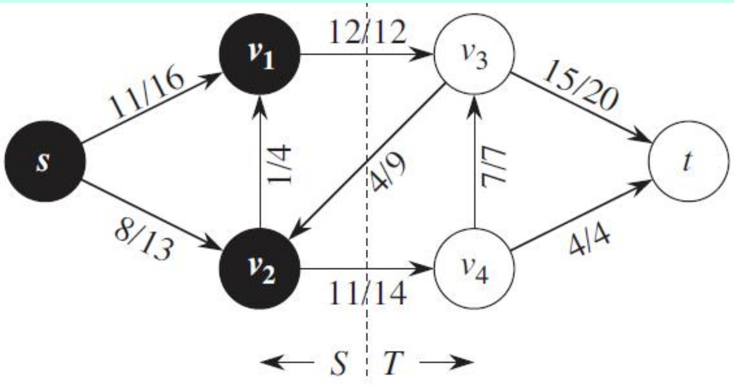
容量计算只考虑从集合S发出进入集合T的边的容量

流、切割净流量和切割容量之间的关系

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** $f(S,T)=|f|$。

引理26.4说明：对流网络的任意切割**(S,T)**，切割截面上的净流量就等于流网络的流量。

而且，横跨任何切割的净流量都相同，都等于流的值|f |。



◆ **引理26.4的证明如下**：

证明：对于任意 $u \in V-\{s,t\}$，根据结点的流量守恒性质有：

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

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

又根据流的定义：$$|f| = \sum_{v\in V} f(s,v) - \sum_{v\in V} f(v,s)$$

将两式相加

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

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

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

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

注意到 $V = S \cup T$ 并且 $S \cap T = \emptyset$，将上式针对S和T进行分解，得：

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

继续化简，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 $|f|$。

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

净流量和切割的容量之间的关系

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。

根据因引理26.4和容量限制，可以得到

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

推论26.5说明，任何流，包括最大流都不能超过最小切割的容量的限制。

容量限制：$0 \le f(u,v) \le c(u,v)$

问题：最大流不可能超过最小切割的容量，**最大流值是否等于最小切割的容量**?

答案是肯定的，两者是相等的。

- 由**最大流最小切割定理**进行了描述

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

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 在流的值等于任意切割的容量时达到，而此时在对应的残存网络 G_f 中不再有增广路径。所以可以用有无增广路径判断当前流 f 是否是最大流，如果是，则算法也可以终止了。

无增广路径的含义：

对当前流f，由f诱导的残存网络G_f中不再有从s到t的、路径残余容量大于0的简单路径p。即不再有路径p使得 $|f_p| = c_f(p) > 0$ 。

而G_f中没有残余容量等于0的边。所以**无增广路径事实上与G_f中没有从s到t的路径等价。**

最大流最小切割定理的证明

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**

(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26.3，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) 残存网络　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

(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

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

为证明定理，只需证明 $|f| = c(S,T)$。

根据引理26.4， $f(S,T) = |f|$**，故只需要证明** $f(S,T) = c(S,T)$

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

考虑任意一对结点 $u \in S, v \in T$，有三种情况：

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，可以得到

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

证毕

（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

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

因此, $|f| = c(S, T)$ 意味着f是一个最大流。

定理证毕

如何实施Ford-Fulkerson方法求解最大流：？

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

基本思想：通过不断增加**可行流值**的方式找到最大流

技术路线：

（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：**在残存网络中寻找增广路径，利用增广路径对流值进行增加。**

（3）确认无法增加流值，即得到最大流：**当残存网络中不再有增广路径为止，此时获得最大流。**

Ford-Fulkerson算法的细化

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

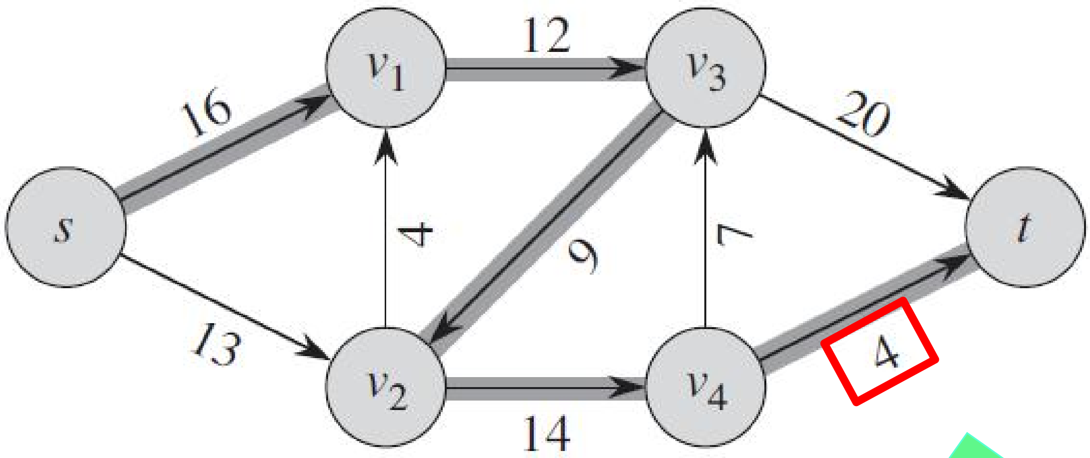
修正与增广路径对应的每条边的流值

如何寻找增广路径?

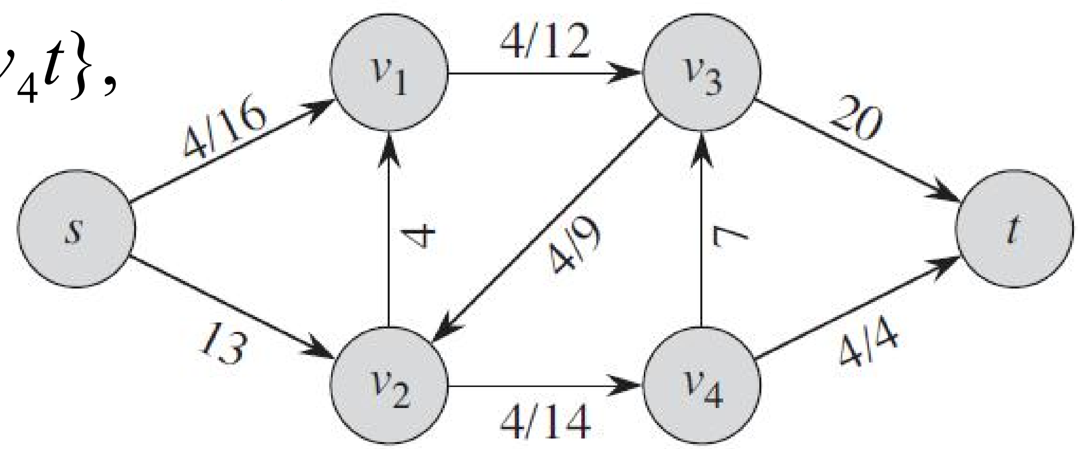
- 可通过**深度优先搜索**或者**广度优先搜索**得到

基本的Ford-Fulkerson算法

算法运行示例：Iteration 1



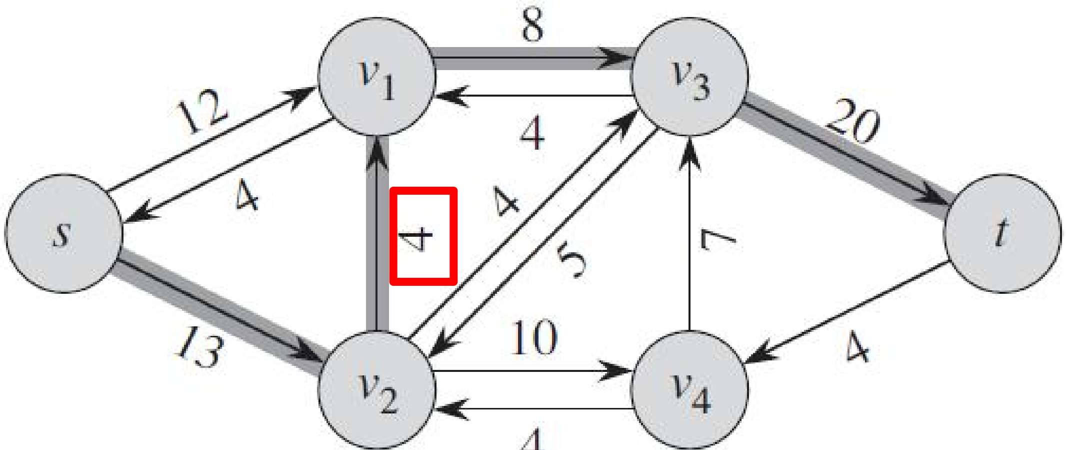
流网络和增广路径



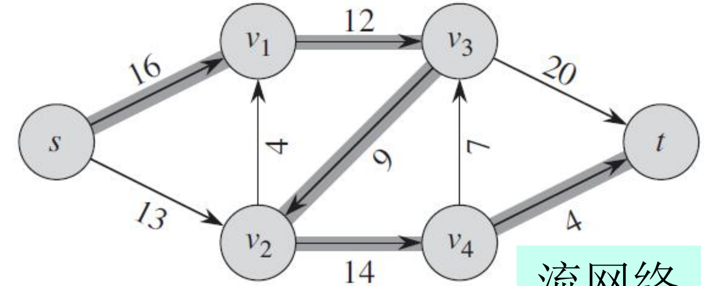
$$p = \{sv_1, v_1v_3, v_3v_2, v_2v_4, v_4t\},$$

$$c_f(p) = 4$$

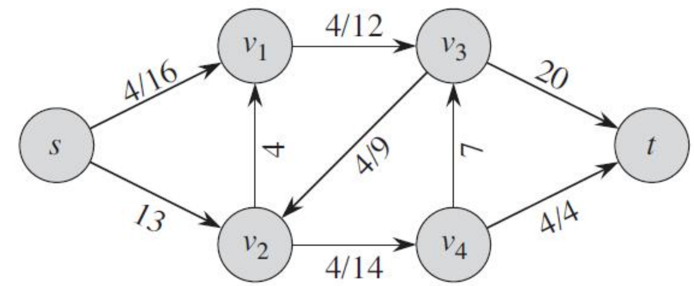
Iteration 2



残余网络和增广路径



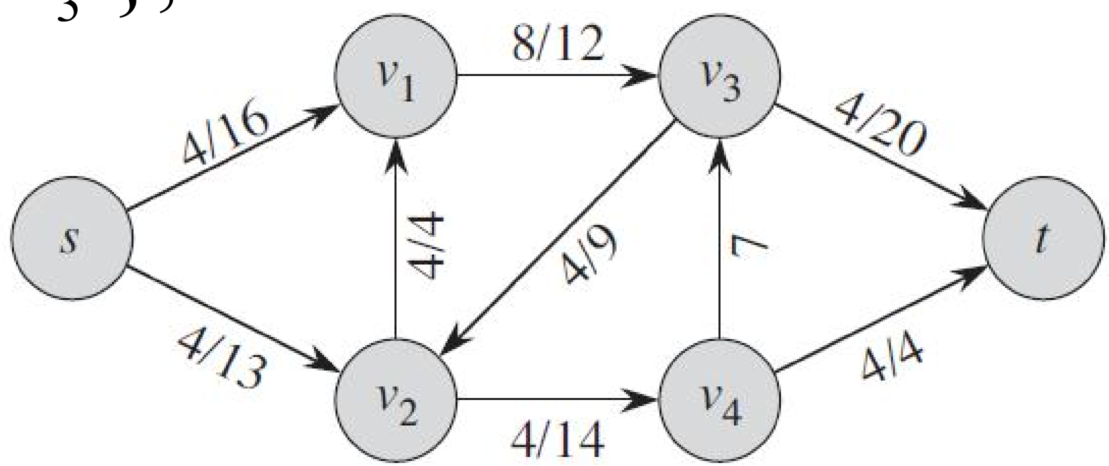
流网络



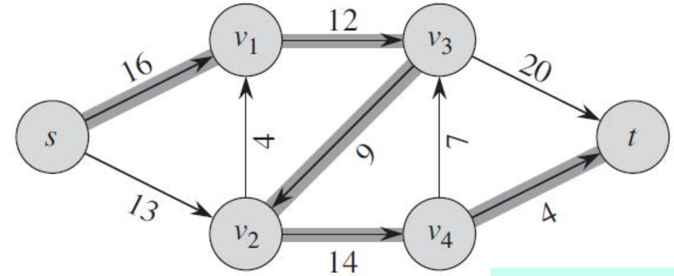
Iteration 1流网络

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

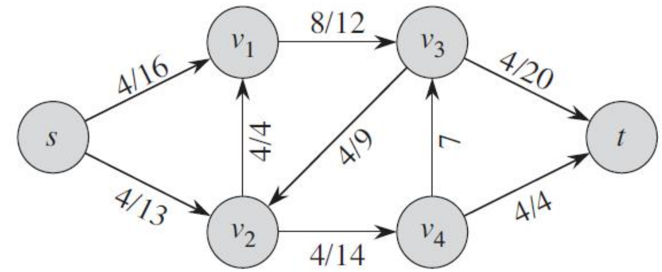
$$c_f(p) = 4$$



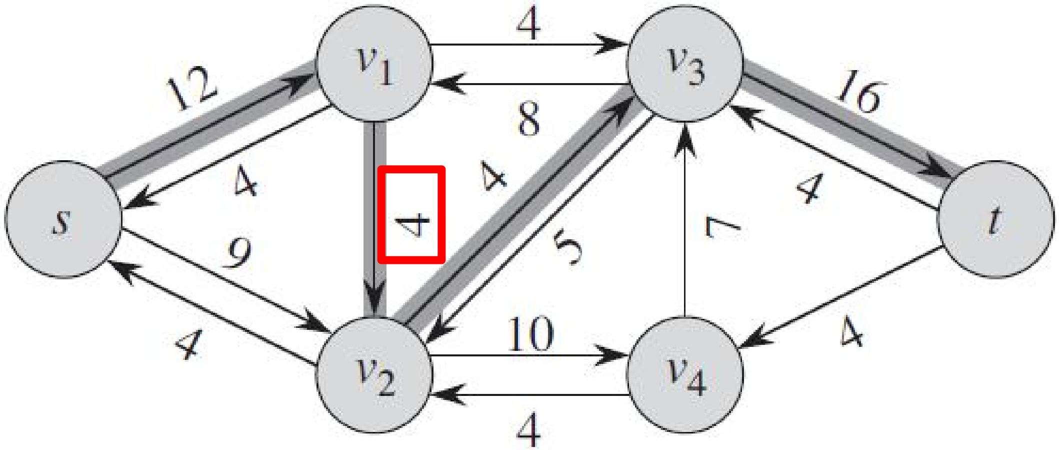
Iteration 3



流网络



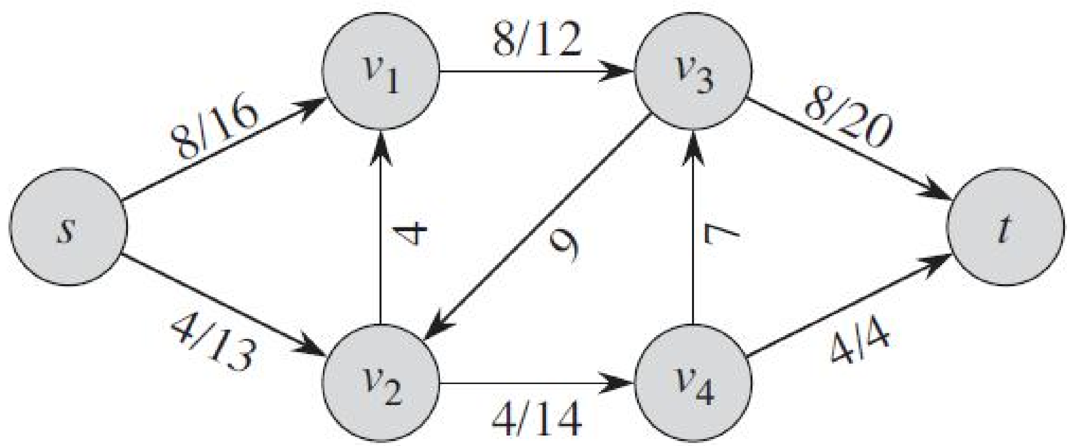
Iteration 2流网络



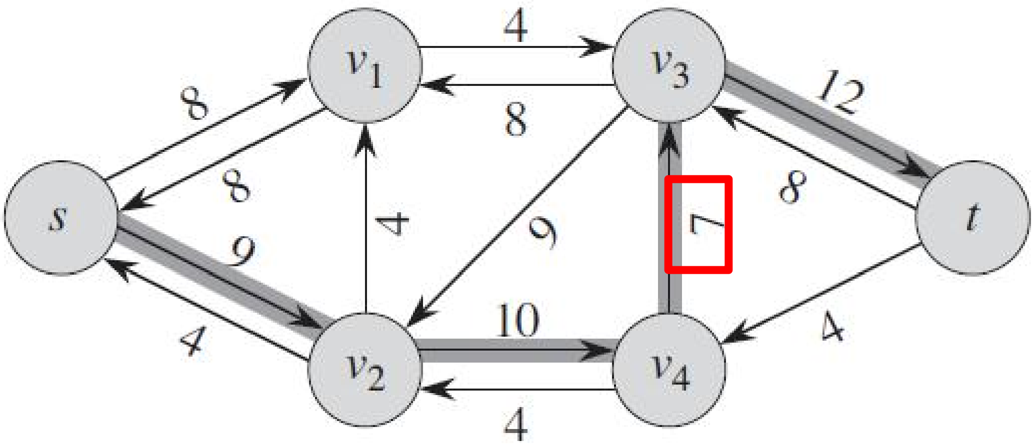
残余网络和增广路径

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

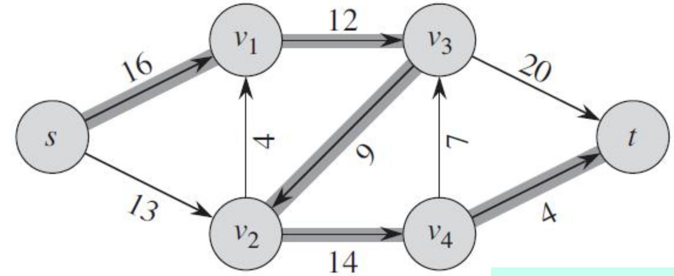
$$c_f(p) = 4$$



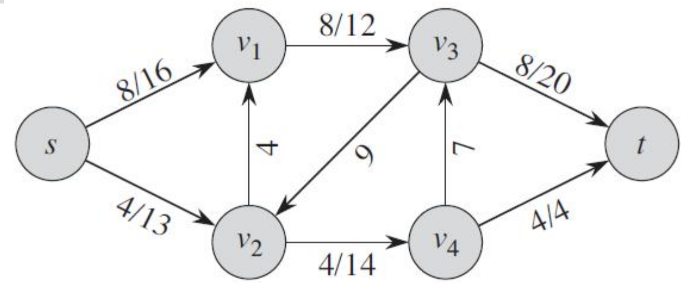
Iteration 4



残余网络和增广路径



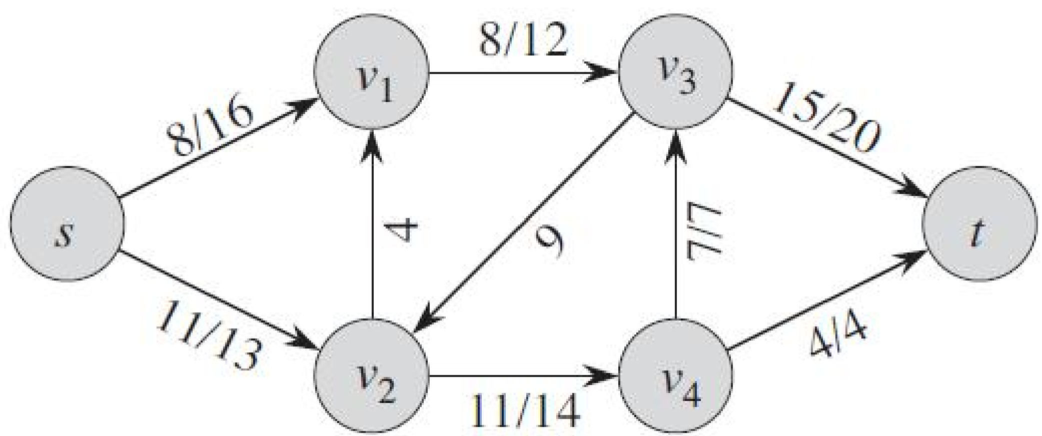
流网络



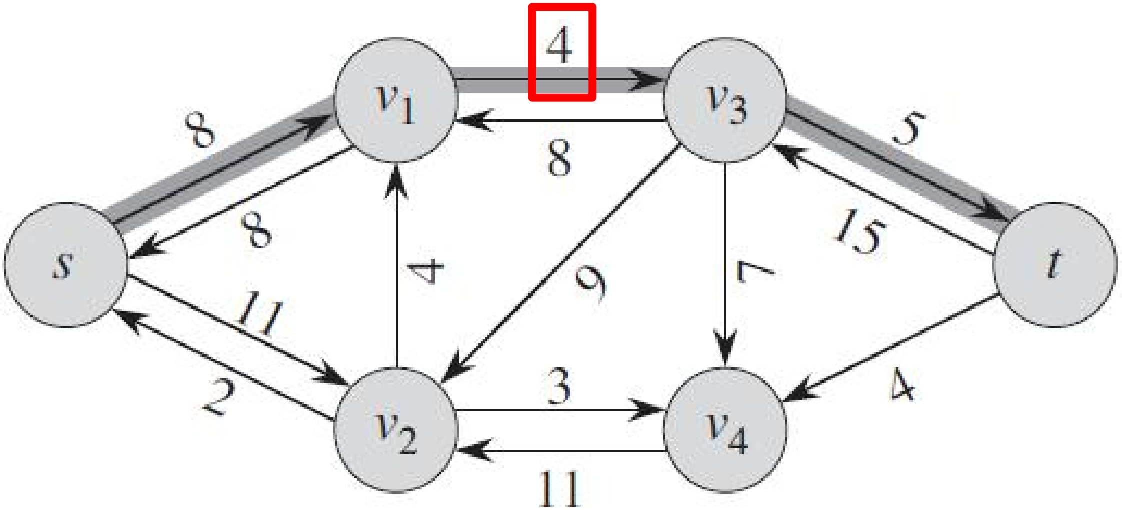
Iteration 3流网络

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

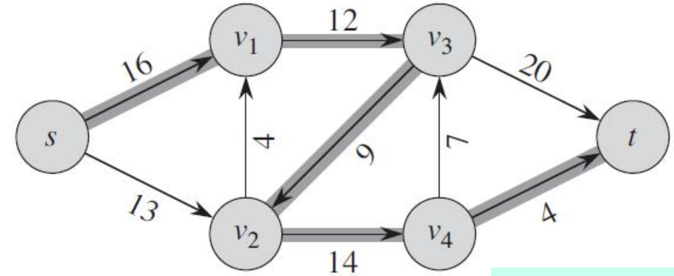
$$c_f(p) = 7$$



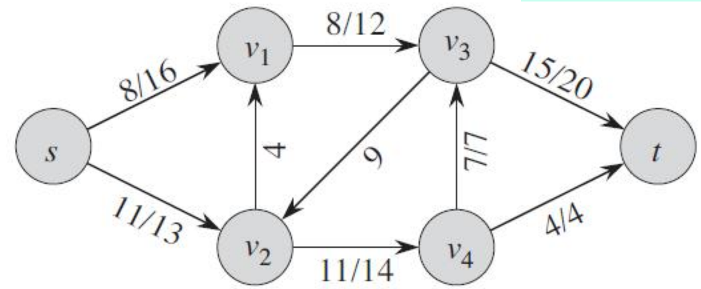
Iteration 5



残余网络和增广路径



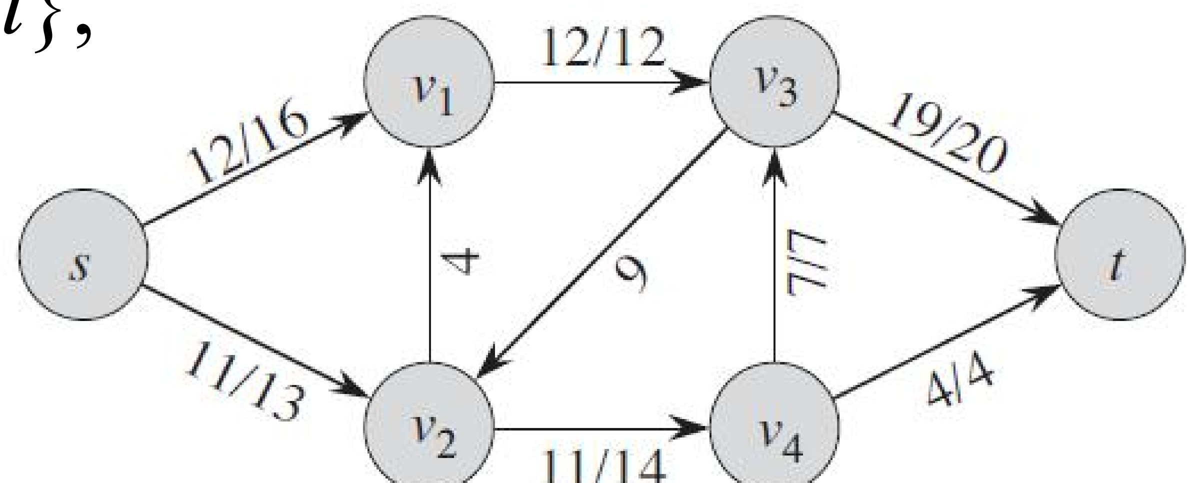
流网络

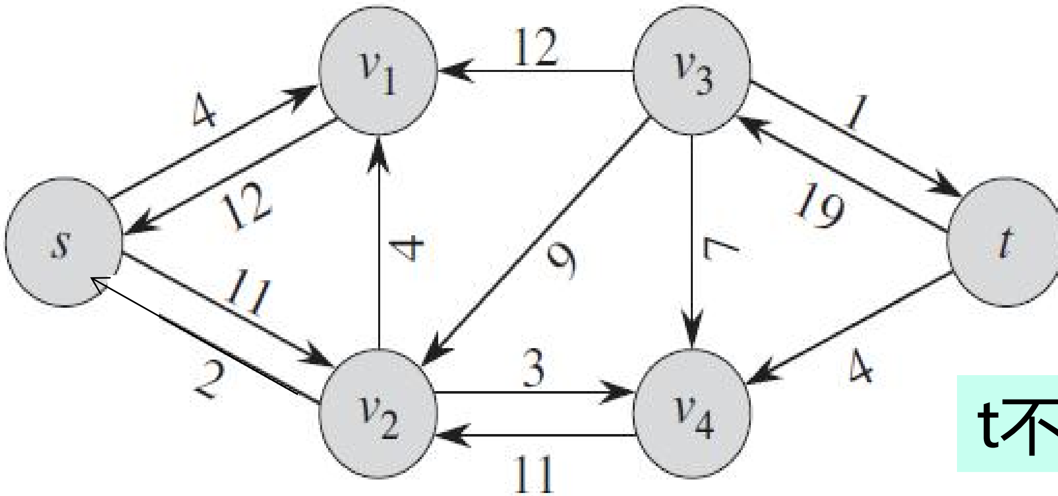


Iteration 4流网络

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

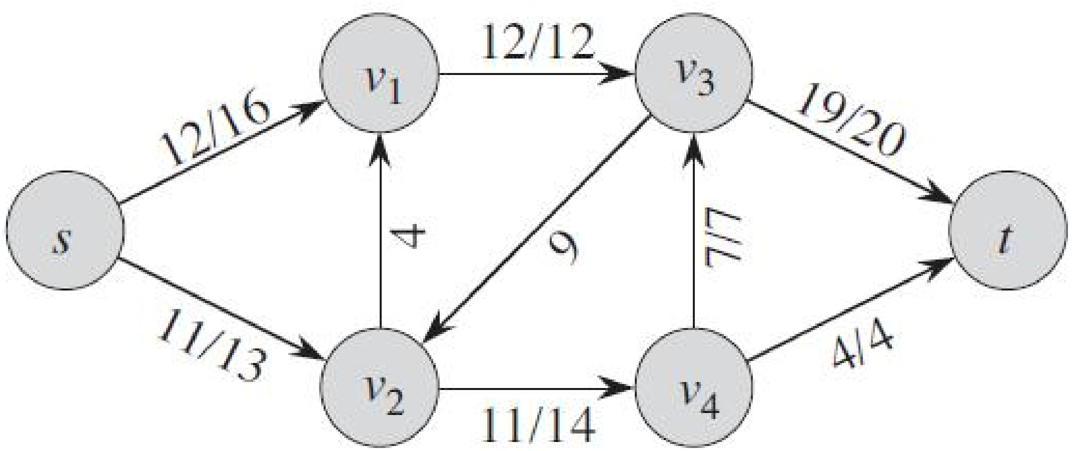
$$c_f(p) = 4$$





t不可达==无增广路径

无增广路径，得到最大流 $|f| = 23$



Ford-Fulkerson算法复杂性分析

假定：所有的容量均为整数。

时间复杂度：$O(|E| \times |f^*|)$，

其中E是流网络的边集，f^*是最大流值

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：包括两部分

（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$O(|f^*|)$循环

（2）每次循环所用时间

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**需要计算每条边的残存容量，运行时间为$O(|E|)$

◆ 利用深度优先搜索或者广度优先搜索，**计算增广路径**，运行时间为 $O(|V|+|E|) \in O(|E|)$

◆ **更新每条边的流值**的时间是: $O(|E|)$

综上，**总的计算时间是** $O(|E| \times |f^*|)$

算法存在的问题：**时间复杂性与最大流值有关，当最大流值非常大时，效率较低。**

Edmonds-Karp算法

Edmonds-Karp算法仍然是基于Ford-Fulkerson方法，不同的是**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（单位距离）**，从而得到不依赖于最大流值的运行时间上界

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算法使用广度优先搜索寻找最短路径作为增广路径

运行时间：O(VE²)

Edmonds-Karp算法运行时间分析

(1) 在残存网络中，**采用广度优先搜索找一条从s到t的最短路径（单位距离）的时间是O(E)。**

(2)可以证明，**随着算法的进行，流的值不断增加，同时增广路径的长度也逐步是递增的。**这使得在算法的整个执行过程中，总共**最多会处理O(VE)条关键边**。

所以Edmonds-Karp算法运行时间为：**$O(VE^2)$。**

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，这里每条边的权重为**单位距离**，路径长度等于路径上的边数。

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： 利用反证法证明。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 $\delta_{f'}(s,v)<\delta_f(s,v)$。

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

可以得到 $(u,v) \in E_{f'}$，并且 $\delta_{f'}(s,u) = \delta_{f'}(s,v) - 1$

根据v的选取，可以得到 $\delta_{f'}(s,u) \geq \delta_f(s,u)$

我们断言 $(u,v) \notin E_f$。否则就有

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

就与假设 $\delta_{f'}(s,v) < \delta_f(s,v)$ 矛盾。

v是最短路径变小的结点中最短路径最小的结点，所以s到u的最短路径不能变小，否则应选u而不是v了。

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，$f(u,v) = c(u,v)$，$c_f(u,v)=0$，

所以(u,v)不在残余网络 G_f 中；

(2) $(u,v)\in E_{f'}$ ：意味着相对流f'，$f'(u,v) < c(u,v)$，$c_{f'}(u,v)>0$。

也因此有：**由残存网络G_f中计算得到的增广路径上边(v,u)上有流量，导致增量计算以后，边(u,v)上的流量因边(v,u)上流量的抵消而减少**，从而有

$c_{f'}(u,v)= c(u,v)-f'(u,v)>0$。

这也意味着，**(v,u)是相对于流*f*的**残存网络G_f的**增广路径上的一条边，而Edmonds-Karp算法中**，增广路径是最短路径，所以这也意味着，**(v,u)存在于**残存网络G_f中**s到u的最短路径上，而且是**从源结点s到结点u的最短路径上的最后一条边。因此有

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

$\delta_{f'}(s,u) \ge \delta_f(s,u)$

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

所以s到v的最短路径并没有减小，与假设 $\delta_{f'}(s,v) < \delta_f(s,v)$ 相矛盾。**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的残存容量是该条路径上边（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。

- 任何一条增广路径上至少存在一条关键边；
- 增流后，当前增广路径上的关键边将从残存网络中消失。

◆ 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。**

◆ 下面利用这个性质证明：**每条边（u,v）成为关键边之后，再下一次成为关键边之时，s到u的最短距离会增加2。**

设u,v∈V，且(u,v)∈ 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：$\delta_f(s,v) = \delta_f(s,u) + 1$

而一旦对流进行增加后，(u,v)就从下一面的残存网络中消失，直到某一步时从u到v的流量减小了。此时，**(v,u)是增广路径上的边，并且有正流量。**记此时的流为f'，则有

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

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

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

通过上述分析得到：当(u,v)从成为关键边到再次成为关键边，

从s到u的距离至少增加2个单位。

s到u的距离最初至少为0，而(u,v)成为增广路径上的边时，这条路径上的中间结点不可能包含t（(u,v)能成为增广路径上的边，意味着u≠t，路径上的中间结点只可能是除u和t以外的其他结点）。因此，一直到u成为不可到达结点之前（最后成为不可达结点的时候意味着流网络中不再有增广路径），s到u的距离最大是|V|-2

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)能成为关键变的总次数最多为 $|V|/2$

注意到每次流值增加，都会至少有一条关键边，因此流值递增的操作次数至多为 O(|V||E|)

最后，由于一共有O(E)对结点可以在残存网络中有边彼此相连，**每条边都可能成为关键边。**

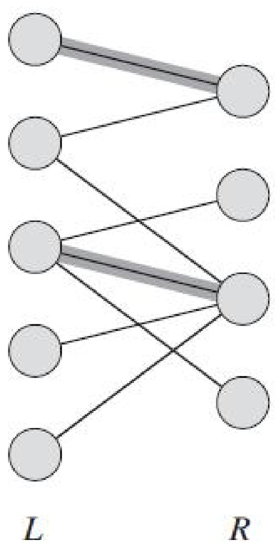
根据上面的分析，在Edmonds-Karp算法执行的整个过程中，每条边最多有|V|/2次机会成为关键边，所以在算法执行的整个过程中关键边的总数为O(VE)。

而每条增广路径至少有一条关键边，每条增广路径意味着要对流网络进行一次流量递增计算，所以Edmonds-Karp算法执行的流量递增操作的次数为**O(VE)。** **证毕。**

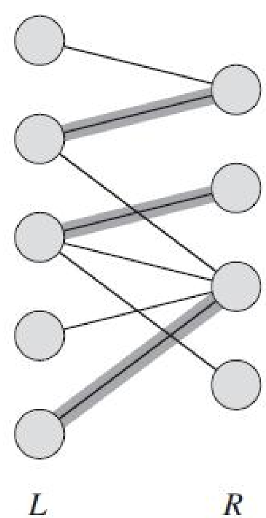
最大流算法应用：寻找最大二分匹配

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



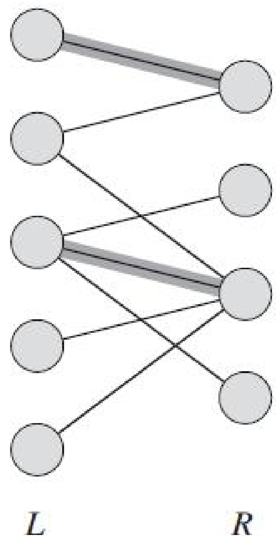
最大匹配

如果子集M中的某条边与结点v相连，则称结点v由M所**匹配**；否则，结点v就是**没有匹配**的。

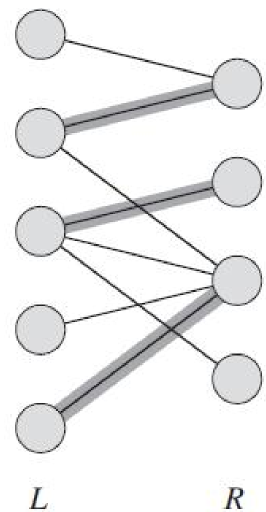
二分图：

结点集合V可以划分为两部分L和R，$L \cup R = V, L \cap R = \varnothing$ ，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

问题：设计算法在二分图中寻找最大匹配

寻找最大二分匹配

基本的思想：构建一个流网络，将寻找最大二分匹配问题转化为求流网络的最大流问题，**流对应于匹配**，然后用Ford-Fulkerson方法寻找最大二分匹配

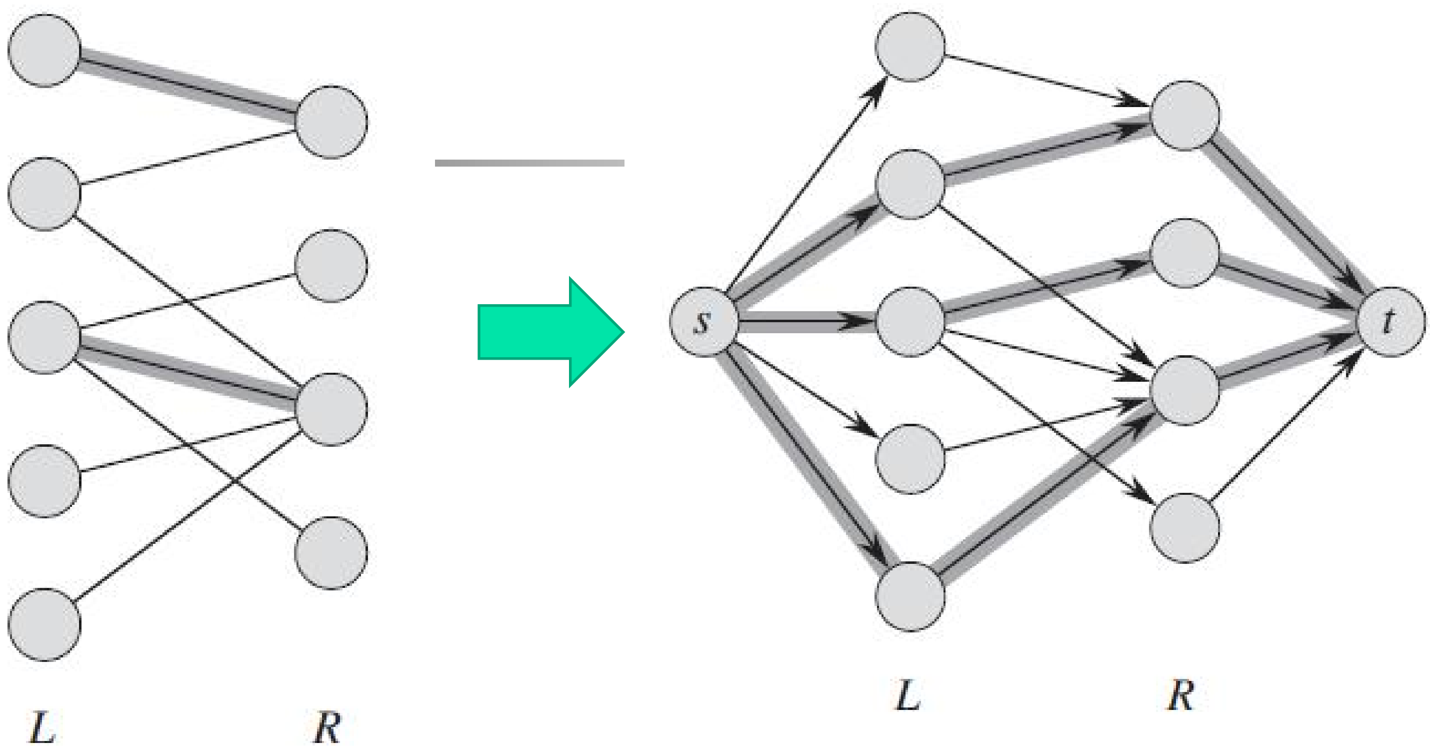
转化：将二分图G所对应的流网络 $G' = (V', E')$ 定义如下

- **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，令**

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

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

一个二分图转化为流网络的例子：



➢ 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|≥|V|/2，则有 $|E| \leq |E'| = |E| + |V| \leq 3|E|$

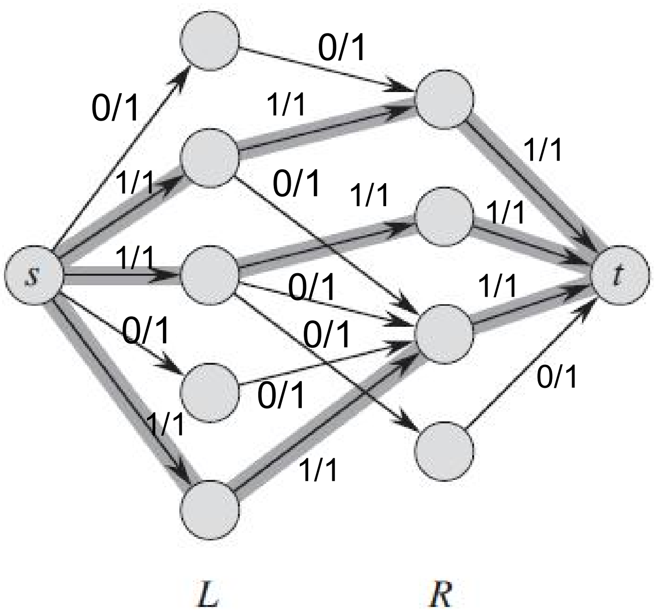
所以 $|E'| \in \Theta(|E|)$

寻找最大二分匹配算法

上述转化中，经过赋值， G' 中所有边的容量为整数值1，所以之后计算所得的流值也将是整数。

算法思路：

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

转化的正确性证明分三步：

(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。

(2) 证明在容量是整数前提限制下，Ford-Fulkerson方法产生的流是整值流，从而保证算法计算的流可以还原到原图的匹配。

(3) 证明最大流的流值等于最大匹配的基数。

(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）能否说明$|M|=|f|$?

$|M|=|f|$?

直观上理解：（L∪{s}，R∪{t}）是一个切割，M中的边恰好是横跨该切割的边，其上是一个单位的流量。所以该切割的净流值就是f的流值，而切割的净流值等于匹配的边数。

所以得到|f|=|M|。

如何证明呢?

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：u有一个单位的流入，根据能量守恒性质，就必有一个点位的流出。由于f是整数值的流， 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，即：

1单位的流进入结点u当且仅当恰好存在一个结点v∈R，使得f(u,v)=1，并且在离开u的边中最多有一条出边带有正值的流。

同样的讨论可应用于v∈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）。

从而可有$|M|=|f|$。

可证明如下：

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

因此，横跨切割(L∪{s}，R∪{t})的净流量f(L∪{s}, R∪{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，从而保证算法计算的流可以还原到原图的匹配。

- 注：如果最大流算法返回流网络G' 中的一个非整数流f(u,v) (即使流的值|f|本身是整数)，上述讨论将存在问题。但**定理26.10说明这种情况不会发生。**

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：用反证法证明

假定M是图G中的一个最大匹配，但其相应的流网络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。

至此，对给定的一个二分无向图G，可以通过创建对应的流网络G'，并在其上运行Ford-Fulkerson方法来找到G的一个最大匹配。

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