Part VI

图算法 (III)

songyou@buaa.edu.cn

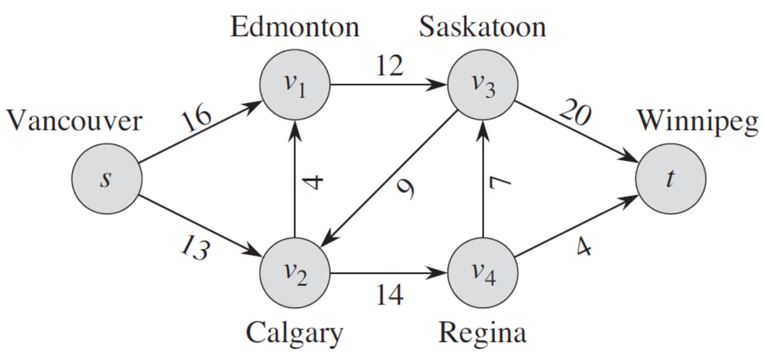
图算法

- 基本图算法
  - 图的表示
  - **BFS，DFS**
  - 拓扑排序
- 单源最短路径
  - 寻找从一个给定的源顶点到所有其他顶点的最短路径。
- 全对最短路径
- 最大流

26 最大流

26 最大流

想象一种物质从唯一的生产源头开始穿过一个系统，流到汇聚的地点。源以某种稳定的速度生产物质，而汇以同样的速度消耗物质



(a) (b)

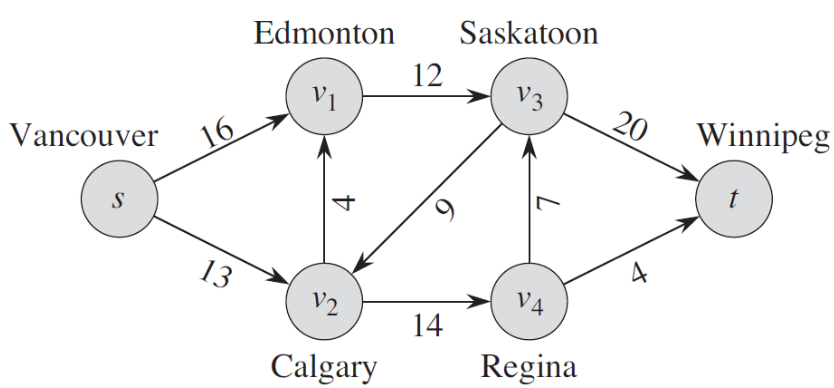
capacity network

最大流：又称为流网络的最大容量问题，或最小分割问题。

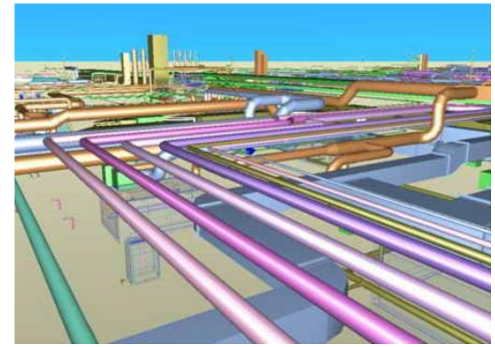
26 最大流

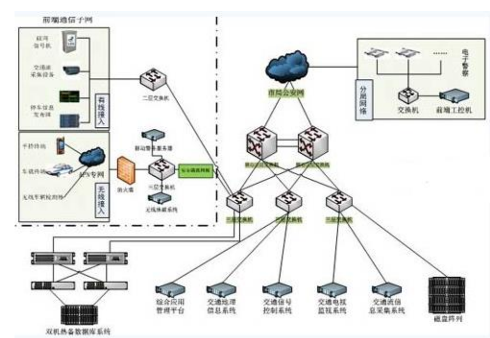
流网络

- 流经管道的液体
- 通过装配线的零件
- 通过电网的电流
- 通过通信网络的信息
- 公路交通网络上的汽车



capacity network





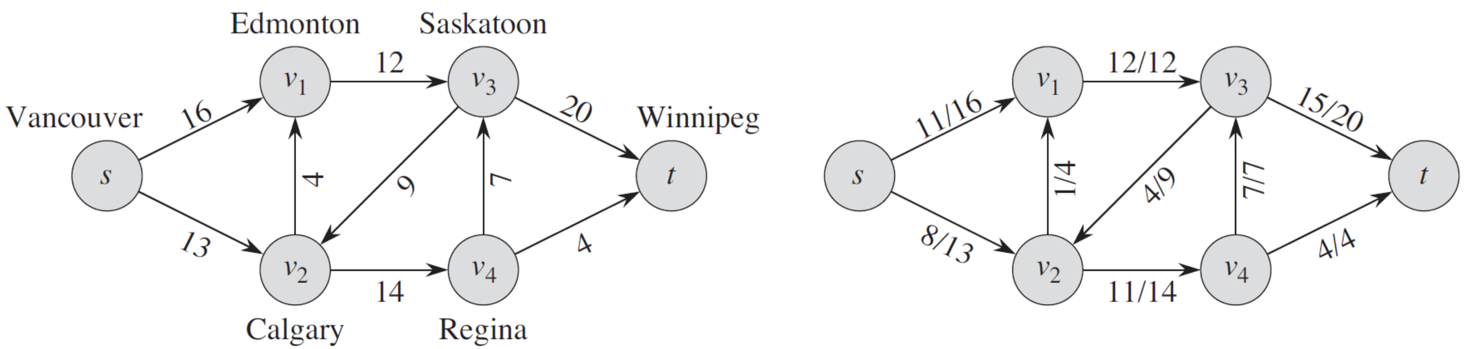
26 最大流

基本定义

1. Flow networks G
2. Flow f
3. Maximum-flow $f_{\max}$
4. Residual networks G_f (残留网络)
5. Residual capacity $c_f(u, v)$ of G_f (顶点间的残留容量)
6. Augmenting path p (增广路径)
7. Residual capacity $c_f(p)$ of p (路径上的残留容量)
8. Cut (S, T) and its net flow $f(S, T)$ and capacity $c(S, T)$
9. Max-flow min-cut (最大流最小割)

26 最大流

- **容量**: 物料流过管道的最大速率。
- **流量守恒**: 材料进入节点的速率与离开的速率相等。
- **最大流问题**: 我们希望计算在不违背容量限制的情况下，流量从源头运到汇点的最大速率。



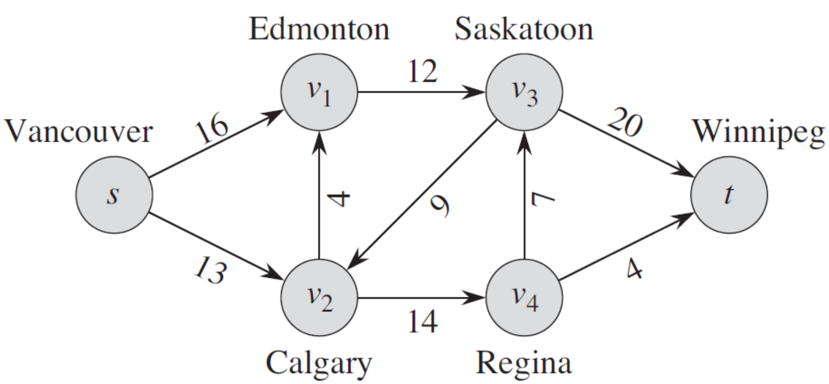
(a) (b)

capacity network

26.1 流网络

流网络和流

- 流网络 $G = (V, E)$是一个有向图，每条边 $(u, v) \in E$ 有非负的容量 $c(u, v) \geq 0$. If $(u, v) \,!\!\in E$, $c(u, v) = 0$.
- 每个顶点位于从源到汇的某条路径上
- ***源 s***
- ***汇 t***
- G 中的一条流是：满足以下两个属性的一个实值函数 $f : V \times V \rightarrow R$:



capacity network

26.1 流网络

流网络和流

- A ***flow*** $f : V \times V \rightarrow R$ 满足以下两个属性:

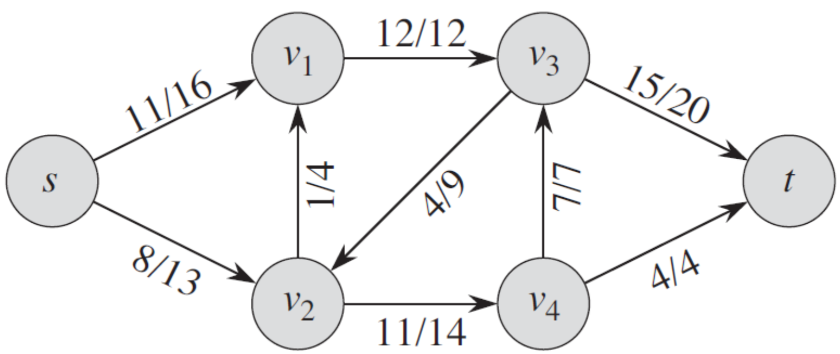
(1) 容量约束: 所有 $u, v \in V$, 要求

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

(2) 流量守恒: 所有 $u \in V - \{s, t\}$, 要求

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

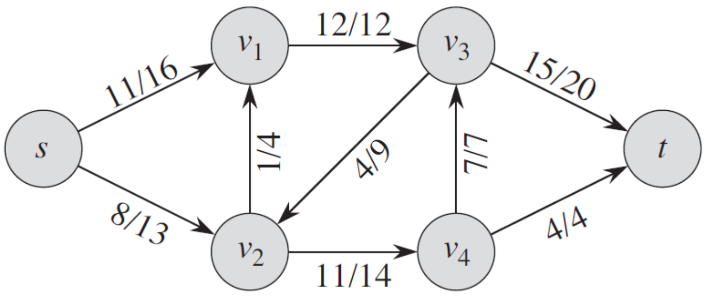
" flow in equals flow out. "



当 $(u, v) \,!\!\in E$, 没有从 u 到 v 的流,且 $f(u, v) = 0$.

26.1 流网络

流网络和流



- $f(u, v)$: 从顶点 u 到 v 的流。

- $|f|$ 的值定义为

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

流出源的总量减去流入源的总量

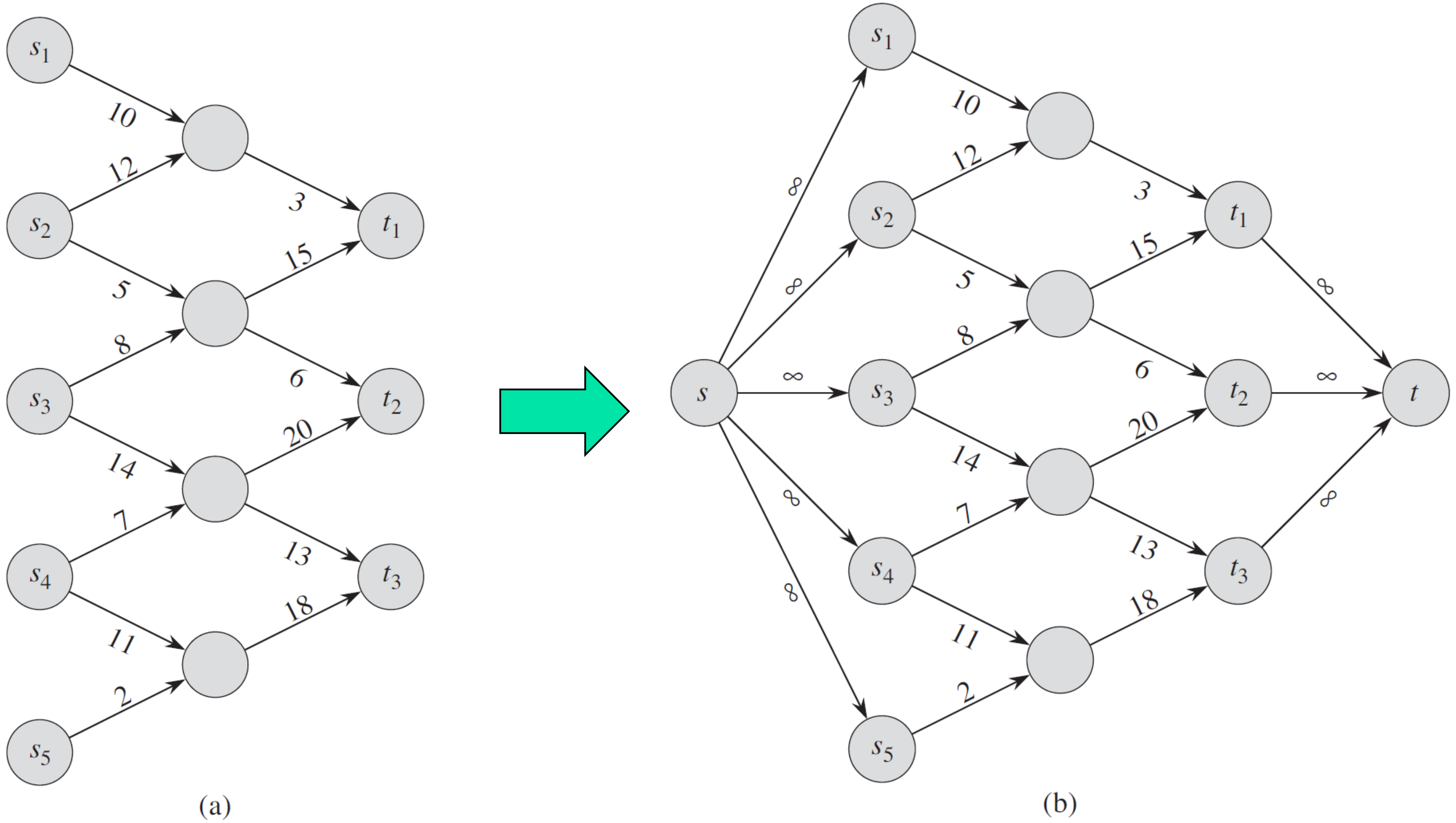
(符号 $|\cdot|$ 表示流量值而不是绝对值)

- **最大流问题**: 给定流网络 G， 源 s， 汇 t, 希望求出流量的最大值。

最大流：又称为流网络的最大容量问题，或最小分割问题。

26.1 流网络

具有多个源和汇的网络

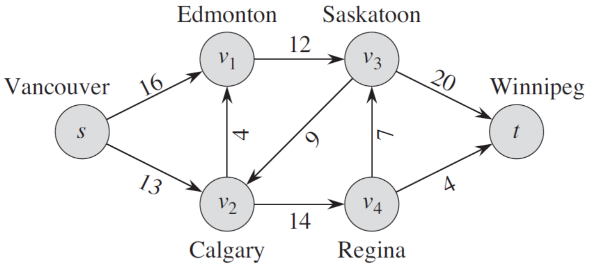


26.2 The Ford-Fulkerson method

一种 “方法” 而不是 “算法”.

Ford-Fulkerson 方法基于三个重要思想:

- 剩余网络（残留网络），核心思想：存在一些边，在其上还能增加额外流，这些边就构成了残留网络的增广路径）
- 增广路径
- 割



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
```

26.2 The Ford-Fulkerson method

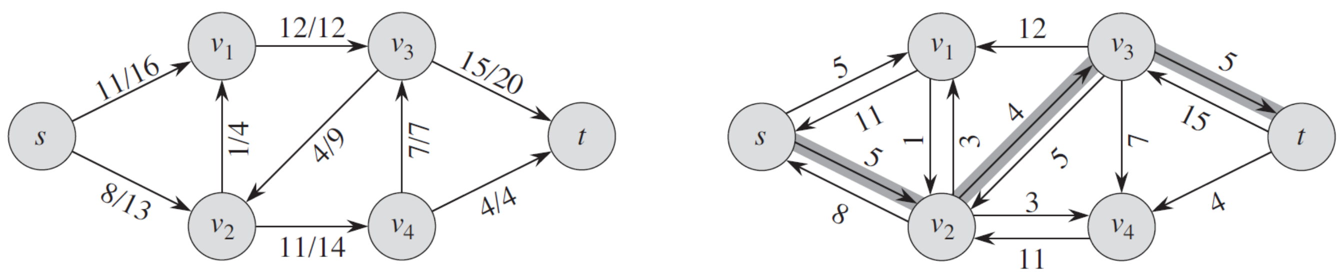
剩余网络

剩余容量

边上还能增加的额外流

反向流最多抵消正向流

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



(a) (b)

Example: 令$u \leftarrow s$, $v \leftarrow v_1$, 若 $c(u, v) = 16$, $f(u, v) = 11$, 在我们超出边(u, v)的容量限制前,我们可以将 $f(u, v)$ 增加到 $c_f(u, v) = 5$ 个单位 。我们还希望允许算法从 v 到 u返回最多11个单位的流量， 因此 $c_f(v, u) = 11$。

26.2 The Ford-Fulkerson method

剩余网络

- **剩余容量**

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

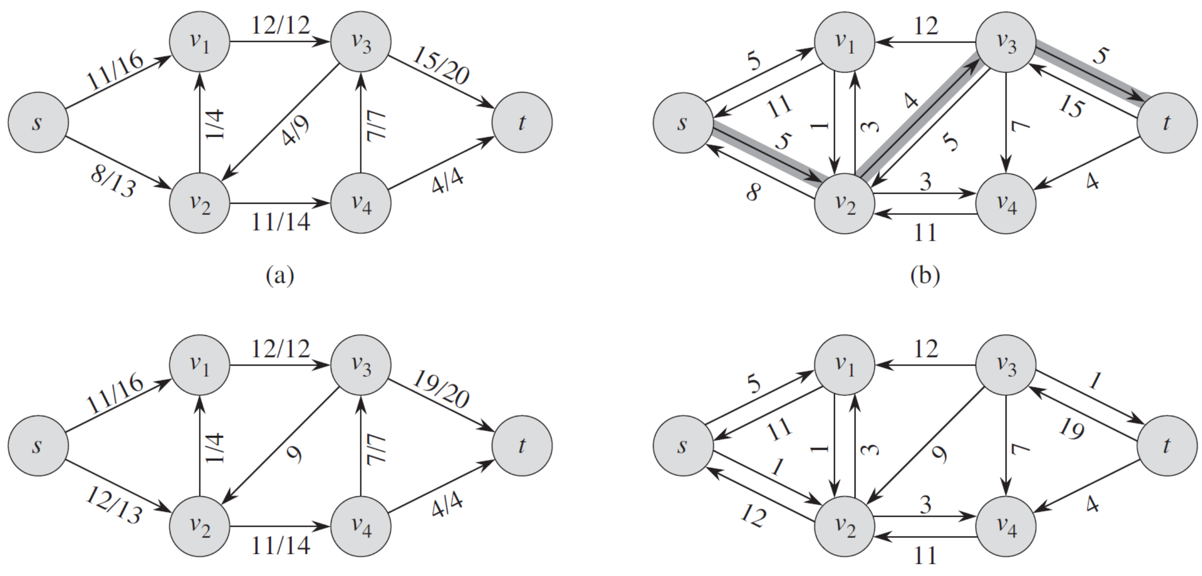
- **residual network:** $G_f = (V, E_f)$, where

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

残留网络：顶点跟流网络一样，边（残留边）的权值为流网络的残留容量

流网络

在(a)图中，沿着其残留网络的增广路径上增加流以后新的流网络



(a) (b) (c) (d)

(a)的残留网络（粗线是增广路径）

(c)的残留网络

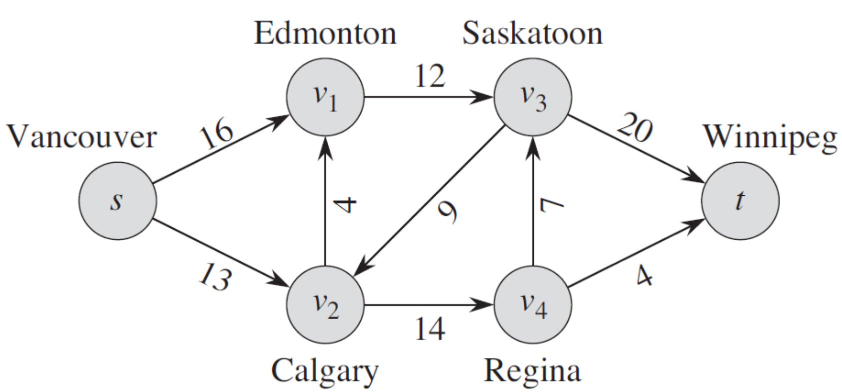
26.2 The Ford-Fulkerson method

残留网络

- 残留容量

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

- 只有容量（流为零）的网络，其残留网络就是其自身。如下图既是流网络原图（流为零），也是残留网络。



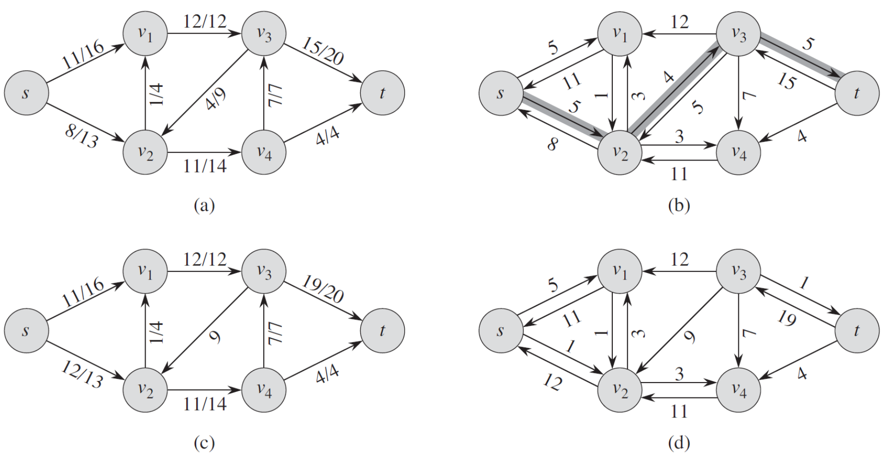
26.2 The Ford-Fulkerson method

残留网络

Lemma 26.1 $G = (V, E)$ 是一个流网络，源为 s， 汇为 t, 令 f 为 G 中的流. G_f 为 G 由 f 引入的残留网络, 令 f' 为 G_f 中的流。 由等式 (26.4) 定义的流的和 $f + f'$ 是 G 中的流，值为

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

$$(f_1 + f_2)(u, v) = f_1(u, v) + f_2(u, v) \quad \ldots\ldots (26.4)$$



Proof 证明

容量约束,

流量守恒

没有被违反… …

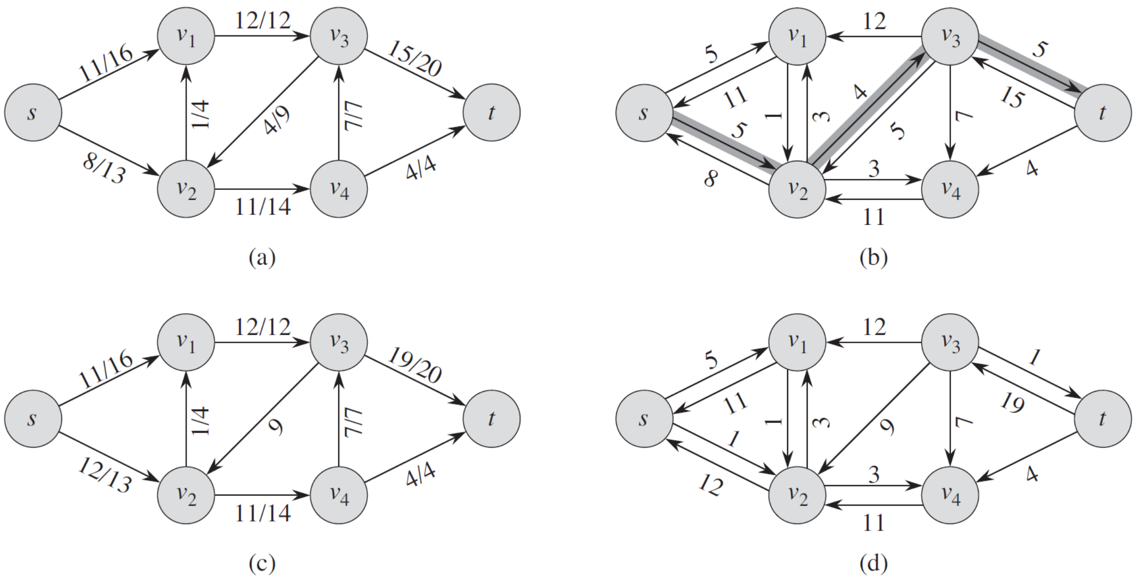
Capacity constraint: For all u, v, we have $f(u, v) \leq c(u, v)$

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

26.2 The Ford-Fulkerson method

残留网络

如何在G_f中找到流 f'?



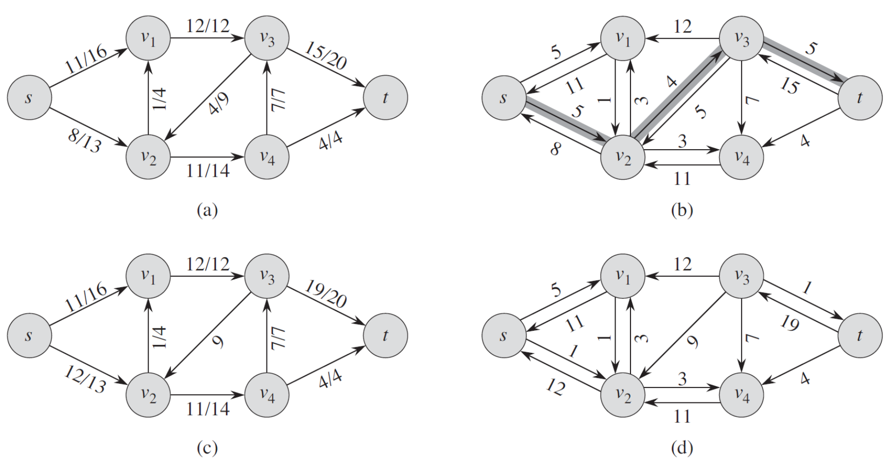
26.2 The Ford-Fulkerson method

Augmenting paths（增广路径）

- **增广路径 *p*** 是残留网络 G_f 中从 *s* 到 *t* 的一条简单路径。
- p 的**残留容量**：在增广路径 *p* 中每条边上可以增加流量的最大值。

  $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is on } p\}$.（容量最小的那条边 (u, v)，也称为关键边）
- **Lemma 26.2** 令$G = (V, E)$ 为流网络，f 为 G 中的流，令 p 是 G_f 中的增广路径。定义函数 $f_p : V \times V \to \mathbf{R}$

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



则 f_p 是 G_f 中的流，

值 $|f_p| = c_f(p) > 0$.

残留网络上的流

Proof： 验证流的两条属性…

26.2 The Ford-Fulkerson method

增广路径

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

Corollary 26.3 令$G = (V, E)$ 为流网络, f 为 G 中的流, 令 p 是 G_f 中的增广路径。由 $f' = f + f_p$定义函数 f': $V \times V \rightarrow \mathbf{R}$。那么 f' 为 G 中的流，值$|f'| = |f| + |f_p| > |f|$.

Proof **:**

Immediately,

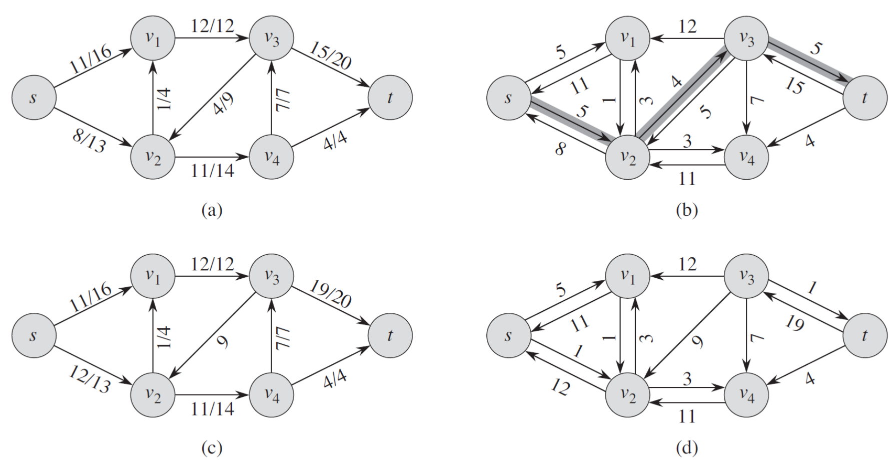
from Lemmas 26.2 and 26.1,

{

Lemmas 26.2: f_p is a flow in G_f.

Lemmas 26.1: $f + f_p$ is a flow in G.

}

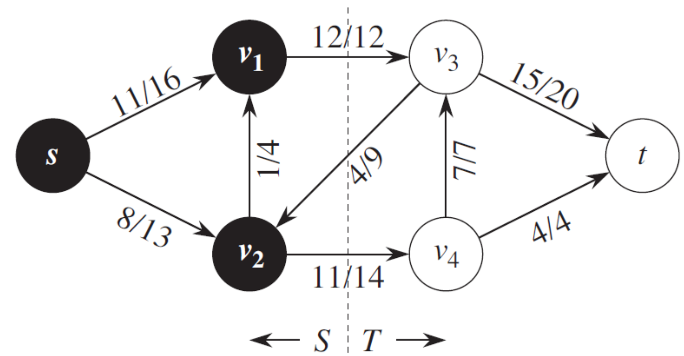


26.2 The Ford-Fulkerson method

流网络的割

- 流网络 $G = (V, E)$ 的**割** (S, T) 是将 V 划分为 S 和 $T = V - S$， 从而让 $s \in S$, $t \in T$. (源在 S, 汇在 T.)
- 若 f 为流, 那么通过割 (S, T) 的**净流量** 定义为 $f(S, T)$.

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



- 割(S, T)的***容量*** 是 $c(S, T)$.

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

注意：不含反向容量

- 一个网络的最小割是网络中具有最小容量的割

$f(S, T) = 19$

$c(S, T) = 26$

26.2 The Ford-Fulkerson method

流网络的割

Lemma 26.4 令 f 为流网络 G 中的流，(S, T) 为 G 的任意割。那么通过 (S, T) 的净流量 $f(S, T) = |f|$.

(任意割的容量都相等)

Proof …略（根据流的定义与流守恒性质来证明，证明略）

流的定义

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

⟺

切割的净流定义

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

流守恒性质

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

26.2 The Ford-Fulkerson method

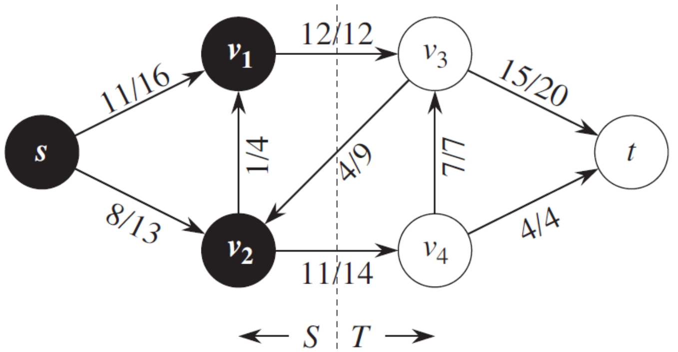
流网络的割

- **Lemma 26.4** 令 f 为流网络 G 中的流，(S, T) 为 G 的任意割。那么通过 (S, T) 的净流量 f(S, T) = | f |.
- **Corollary 26.5** 流网络 *G* 中任意流 *f* 受限于 *G* 中任意割的容量。

  (任意割的容量都是流的上界)

  Proof 很显然。根据切割的净流与容量的定义来证明。

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



流网络的割

Theorem 26.6: (Max-flow min-cut theorem)

若 f 为流, 以下条件等价:

1. f 是 G 中最大流。
2. 残留网络 G_f 不包含增广路径.
3. $|f| = c(S, T)$ 对于G中某些割 (S, T) 。

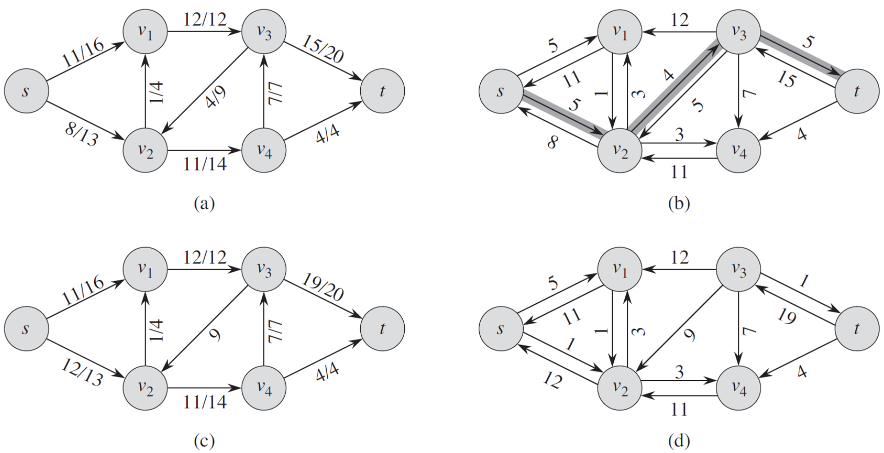
Proof …

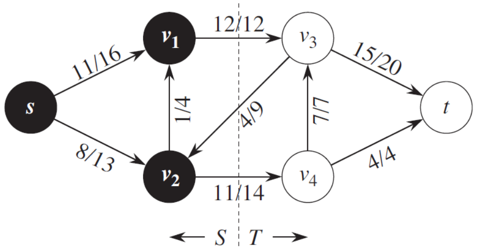
证明思路:

1=>2

2=>3

3=>1





26.2 The Ford-Fulkerson method

流网络的割

Theorem 26.6: (Max-flow min-cut theorem)

若 f 为流, 以下条件等价:

1. f 是 G 中最大流。
2. 残留网络 G_f 不包含增广路径.
3. $|f| = c(S, T)$ 对于G中某些割 (S, T) 。

最大流求解算法:

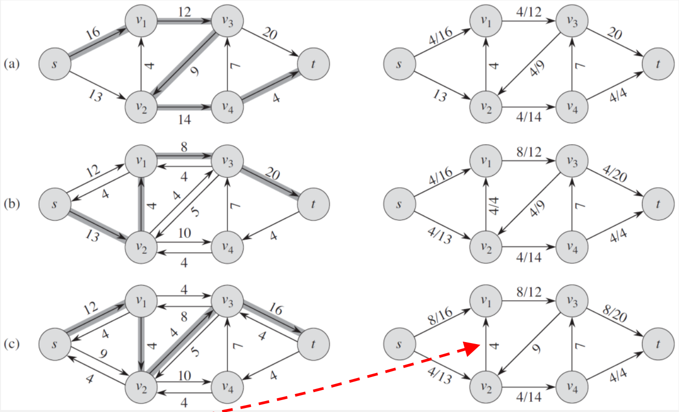
① 流网络比较小时：穷举出所有切割(cut)，求出最小cut。

② 流网络比较大时：求残留网络，找增广路径（求路径上的残留容量），在流网络中沿增广路径压入残留（剩余）容量。

26.2 The Ford-Fulkerson method

The basic Ford-Fulkerson algorithm

```
FORD-FULKERSON(G, s, t)
for each edge (u, v) ∈ E
    f[u, v] ← 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
            f[u, v] ← f[u, v] + c_f(p)
        else f[v, u] ← f[v, u] - c_f(p)
```



算法：求残留网络G_f，找增广路径p，求路径上的残留容量$c_f(p)$，在流网络中沿增广路径在每条边上压入残留（剩余）容量。

Ford-Fulkerson Algorithm

虚线框里的是residual network

1. 初始，图a-L，流f为0，增广路径的残留容量为4；

2. 图a-R，沿增广路径可压入流4，图中的流为4；

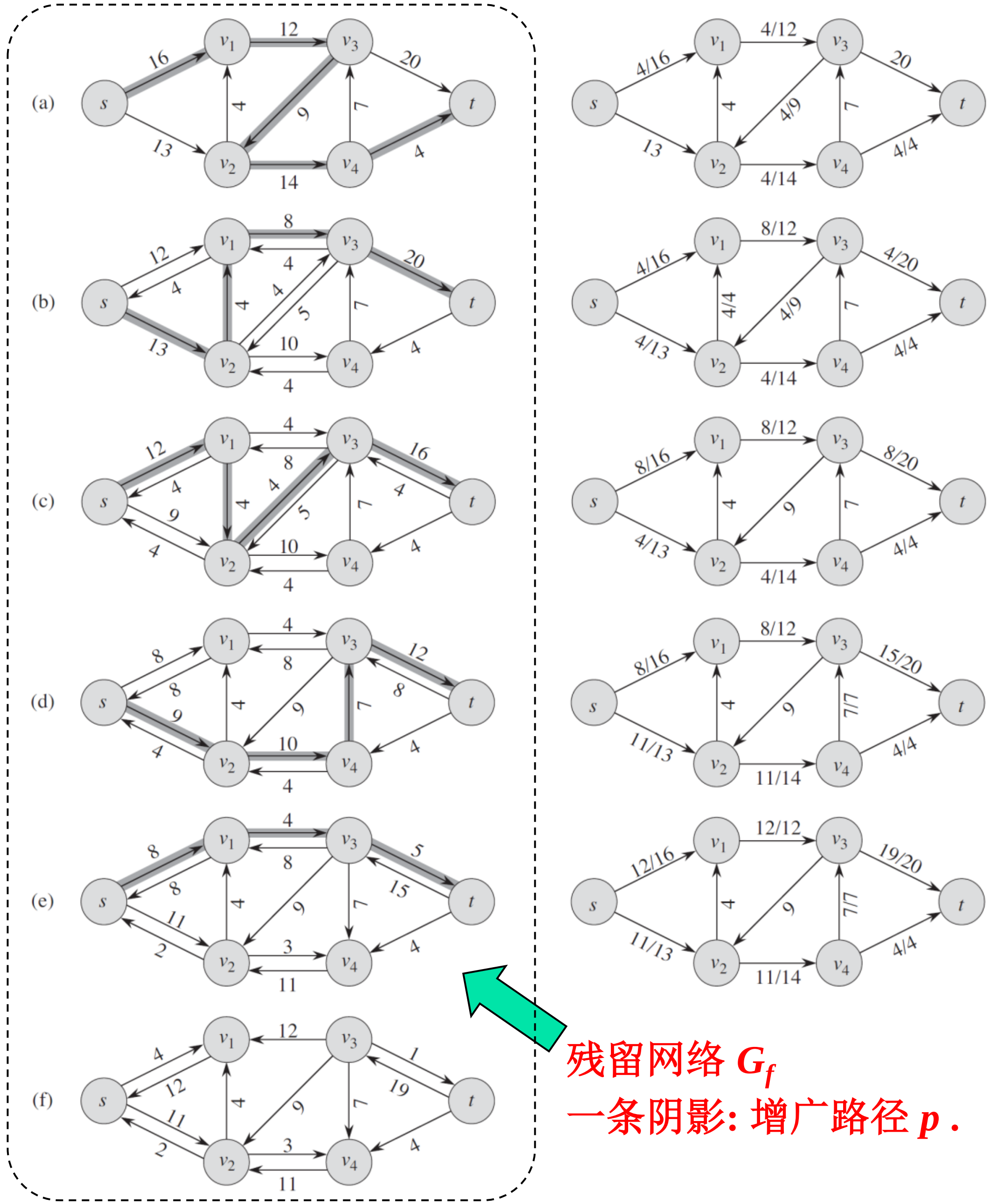
3. 图b-L是图a-R的残留网络，图b-L的一个增广路径的残留容量是4；

4. 在流网络图a-R的基础上，沿图b-L的增广路径可压入流4，得到图b-R；

……

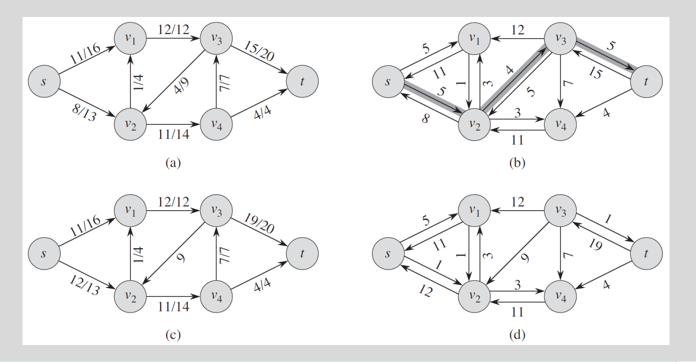
图f中不存在增广路径（不能再增加流，因此图e-2中的流就是最大流。

增广路径选取方法不同，计算效率不同。



26.2 The Ford-Fulkerson method

Analysis of Ford-Fulkerson



```
FORD-FULKERSON(G, s, t)
for each edge (u, v) ∈ E
    f[u, v] ← 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
            f[u, v] ← f[u, v] + c_f(p)
        else f[v, u] ← f[v, u] - c_f(p)
```

$O(\ E\cdot f^*\ )$

设最大流为 f^*：

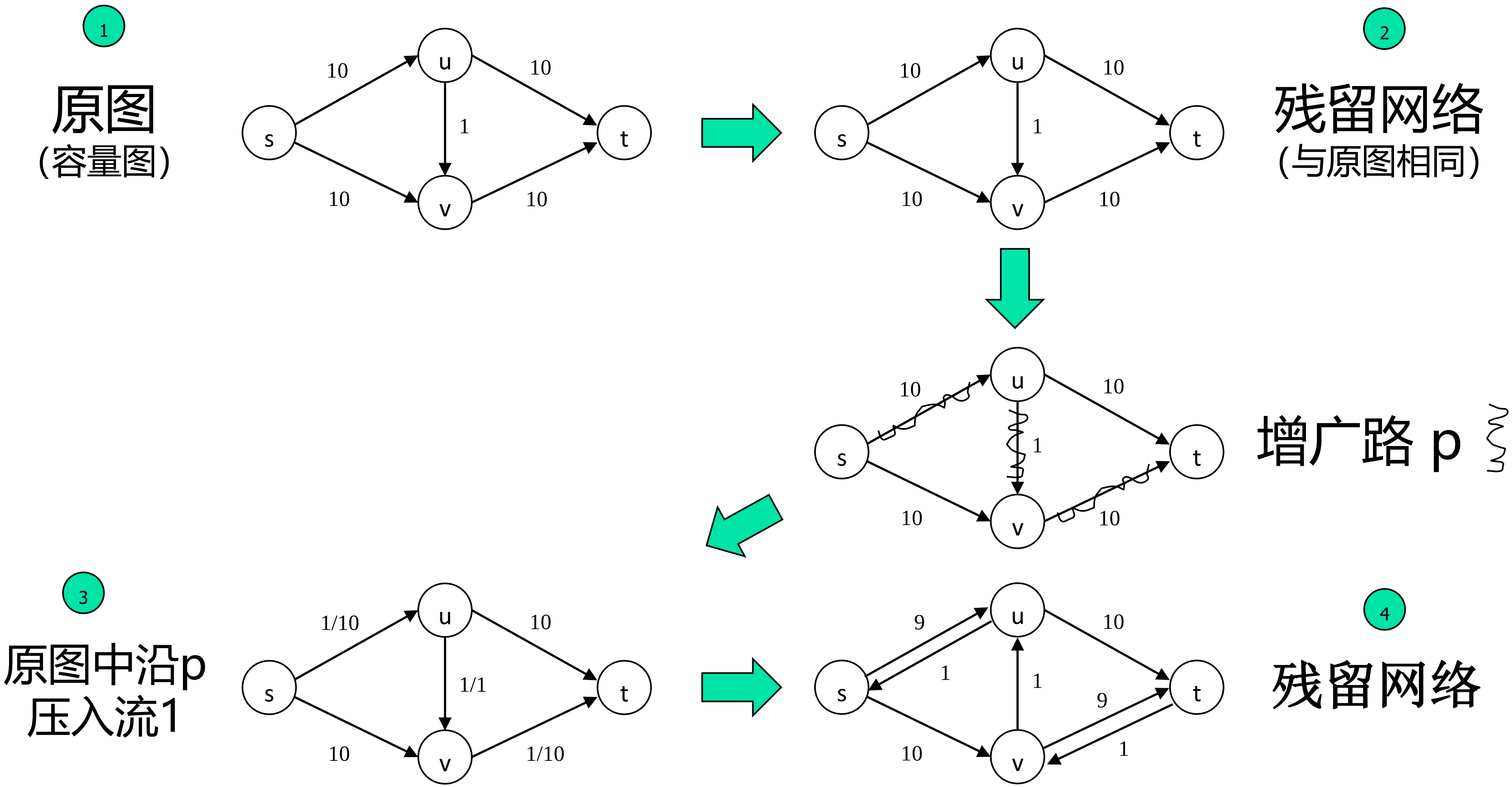
每次找到增广路径，流至少增加1，流从0增加到f^*，时间为$O(f^*)$；

每次找增广路径和给边增加流的操作，时间为$O(E)$；

总的时间， $O(E\cdot f^*)$

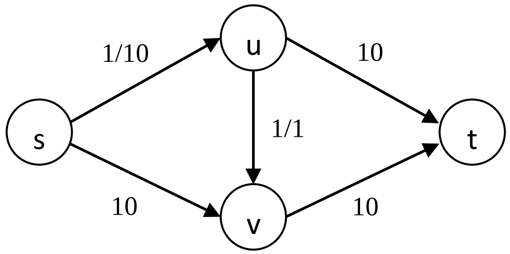
但容量为整数且最优流量 $|f^*|$ 较小时，**Ford-Fulkerson** 算法的运行时间较好。

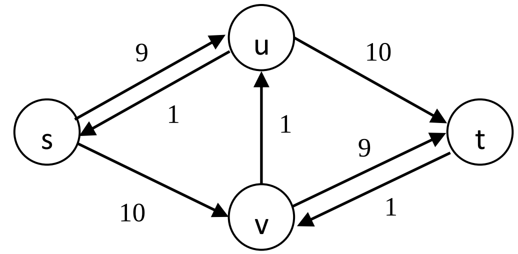
If *f** is large? an example...



3

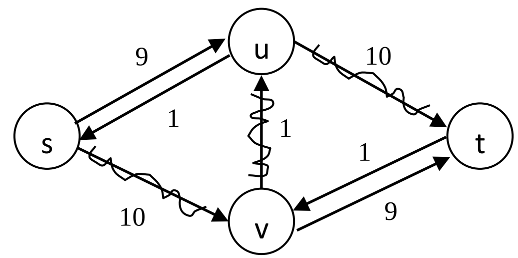
原图中沿p压入流1





4

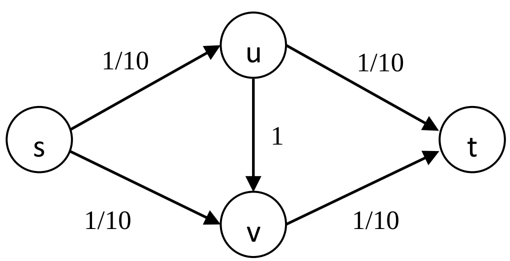
残留网络

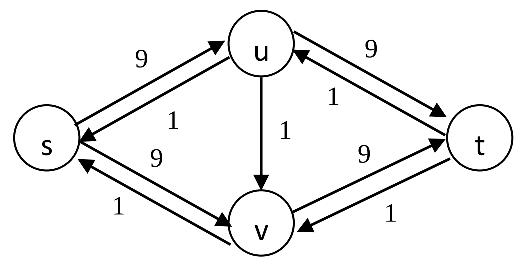


增广路 p

5

上一流图中沿p压入流1



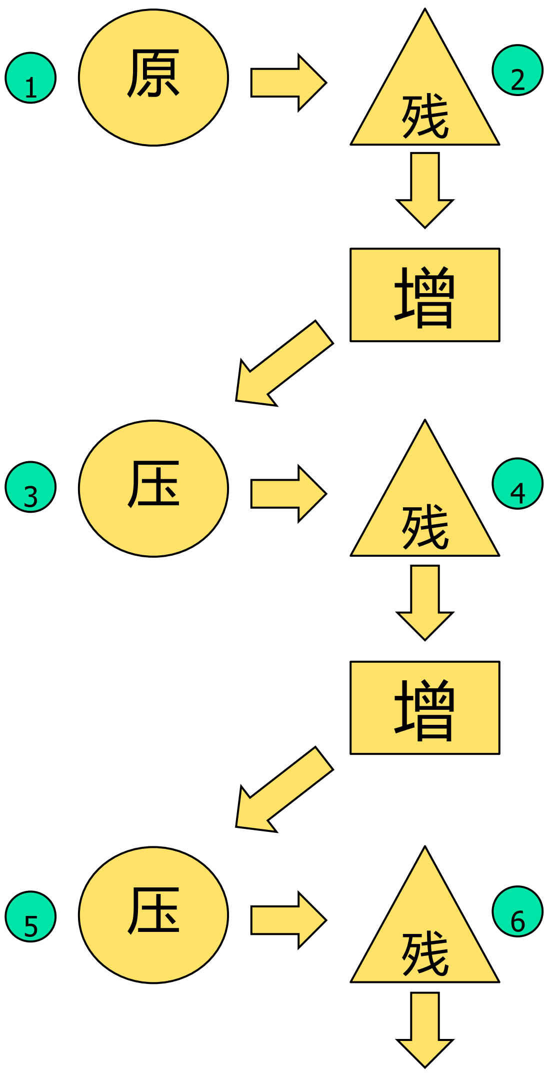


6

残留网络

If *f is large?**
an example...

若最优流量值 |*f**| 较大，F-F 算法表现不太好。



26.2 The Ford-Fulkerson method

```
FORD-FULKERSON(G, s, t)
1 for each edge (u, v) ∈ E
2     f[u, v] ← 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) is in p}
5     for each edge (u, v) in p
6         if (u, v) ∈ E
7             f[u, v] ← f[u, v] + c_f(p)
8         else f[v, u] ← f[v, u] - c_f(p)
```

$O(E \cdot f^*)$

The Edmonds-Karp algorithm

通过宽度优先搜索在第3行中找到增广路径 ***p***，我们可以改进**F-F**的边界。即我们选择 ***p*** 作为残留网络中从 ***s*** 到 ***t*** 的最短路径，其中每条边都有单位距离(权值)。我们称之为**F-F**方法，因此实现了**Edmonds-Karp**算法。**E-K**算法运行时间为$O(VE^2)$。证明……?

26.2 The Ford-Fulkerson method

EDMONDS-KARP(G, s, t)

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

    $f[u, v] \leftarrow 0$

while there exists a path p from s to t in the residual network G_f **(using BFS)**

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

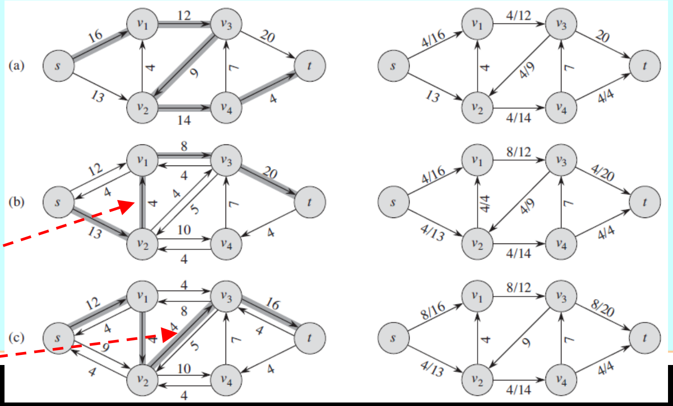
    for each edge (u, v) in p

        if $(u, v) \in E$

            $f[u, v] \leftarrow f[u, v] + c_f(p)$

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

$O(V \cdot E^2)$



证明思想：关键边（增广路径p上的最小容量边）。

沿着p增加流一次，关键边消失；边(u, v)最多$O(V)$次作为关键边；共E条边；E-K算法执行中的关键边数量$O(V \cdot E)$ **（关键边全部消失，不再有增广路径，最大流找到）**。每次找增广路径和给边增加流的操作，时间为$O(E)$。总的时间， $O(V \cdot E^2)$

26.2 The Ford-Fulkerson method

Idea:

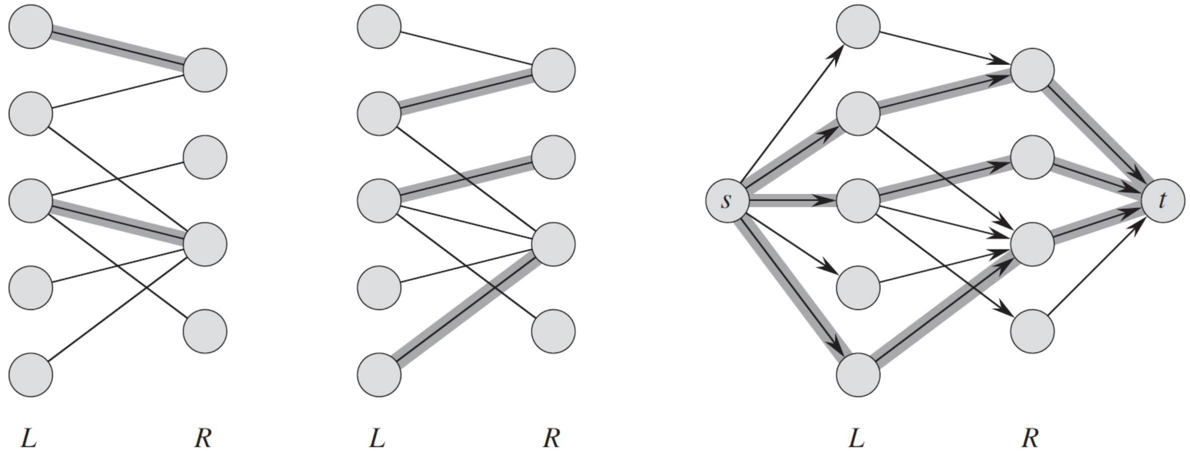
- 残留网络
- 增广路径
- 割

Method: The Ford-Fulkerson method

Algorithm: EK

Code:

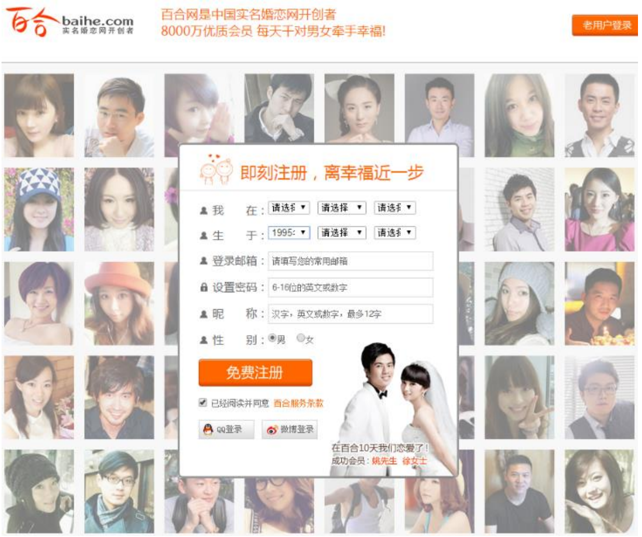
26.3 最大二分匹配



(a) (b) (c)

实际应用

- L : machines ; R : tasks
- L : students ; R : scholarships
- L : students ; R : mentors
- L : gentlemen ; R : ladies
- …



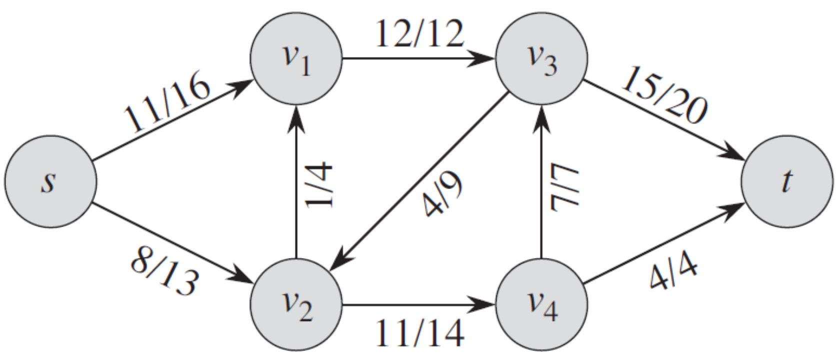
26.4 Push-relabel algorithms *

26.5 The relabel-to-front algorithm *

chapter 29 Linear Programming *

Exercise

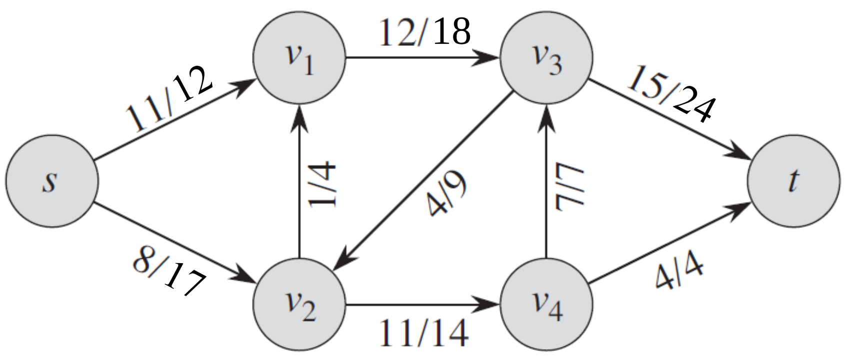
- 图 26.1(b), 通过割 $(S, T) = (\{s, v_2, v_4\}, \{v_1, v_3, t\})$的流量是多少?
  这个割的容量是多少?
- 图像的最小割?
  最大流?

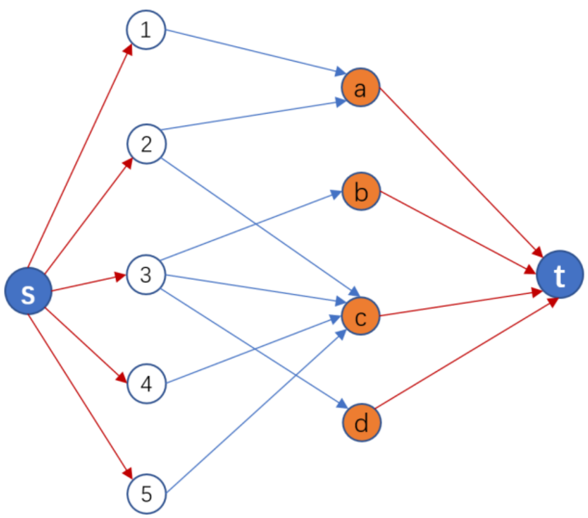


Exercise

下面的图，对于每张图，

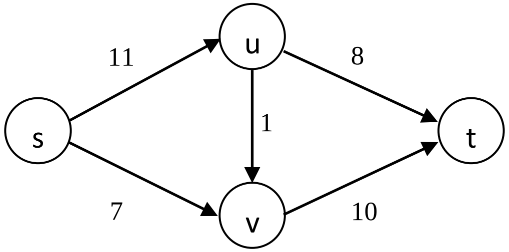
- 最小割是多少?
- 最大流是多少?

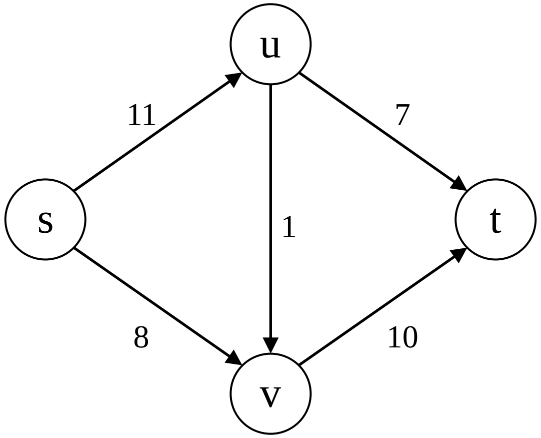


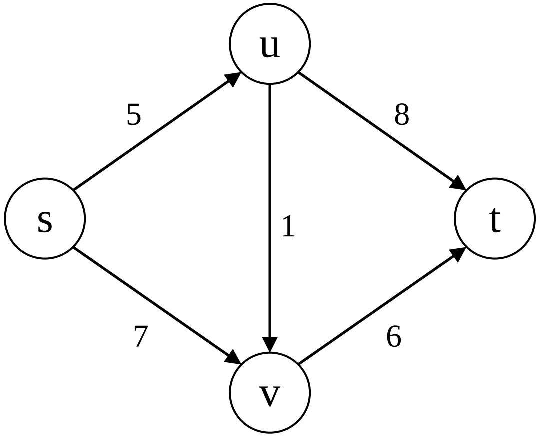


Exercise

最大流是多少?

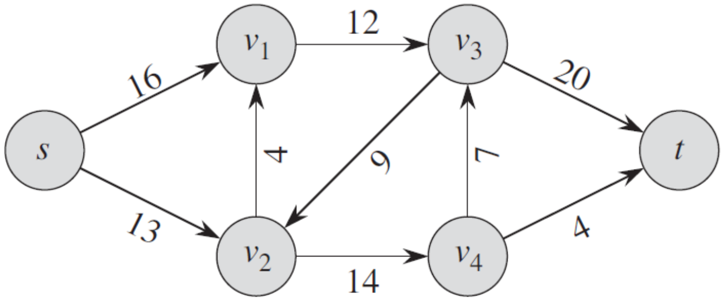






Exercise

采用E-K算法，画出左图的最大流求解过程。



EDMONDS-KARP(G, s, t)

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

  $f[u, v] \leftarrow 0$

while there exists a path p from s to t in the residual network G_f **(using BFS)**

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

  for each edge (u, v) in p

    if $(u, v) \in E$

      $f[u, v] \leftarrow f[u, v] + c_f(p)$

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