

第9讲 网络流问题 (中)

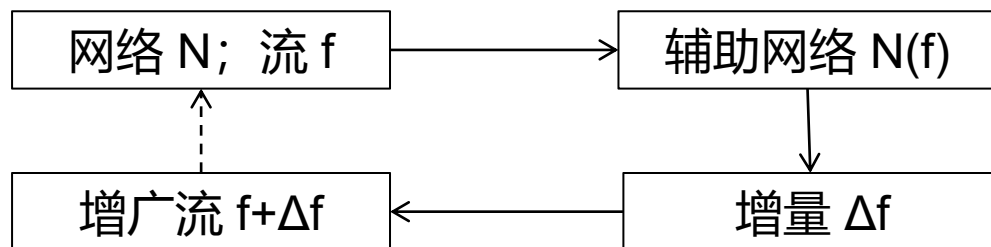
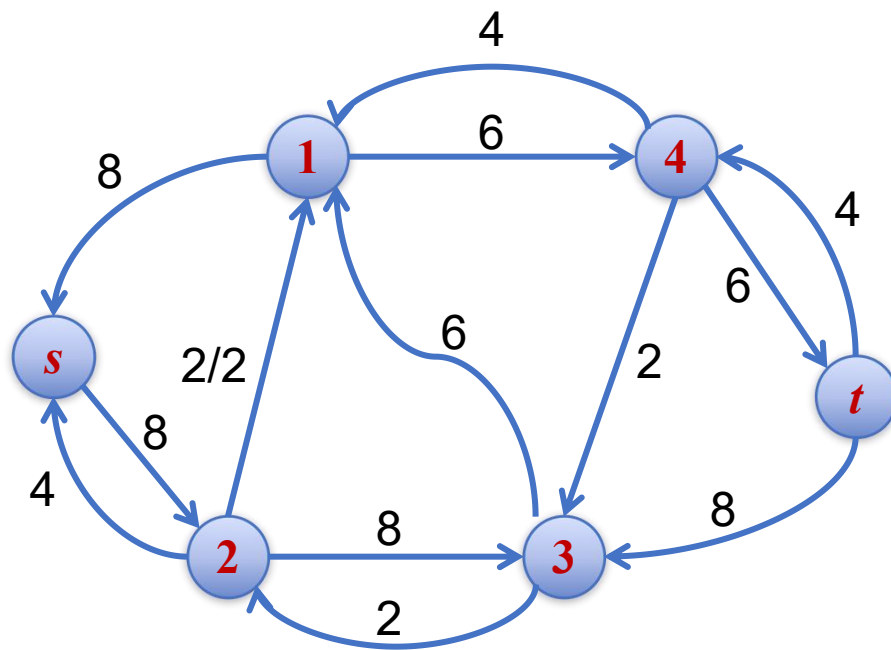
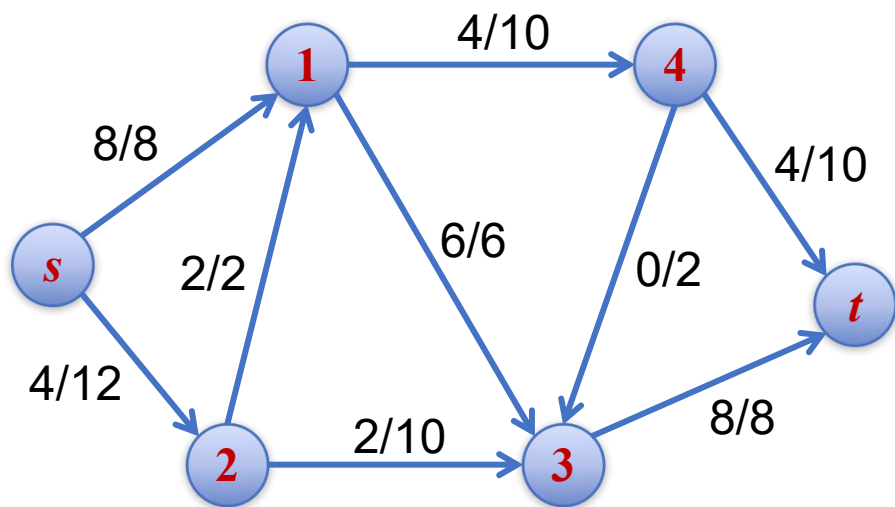
罗国杰

gluo@pku.edu.cn

2024年春季学期

Review: Max-flow Algorithms

► <https://visualgo.net/en/maxflow>



Ford-Fulkerson: $O(E |f^*|)$
 Edmonds-Karp: $O(VE^2)$
 Dinic: $O(V^3) \sim O(EV^2)$

最大流 预流推进算法

- ➡ 预流推进 (preflow-push) 算法
 - ▶ 预流的概念
 - ▶ 预流推进算法的思想
 - ▶ 算法正确性证明
 - ▶ 算法时间复杂性分析
- ➡ 预流推进算法的优化
 - ▶ 重标记前置 (relabel-front) 预流推进算法
 - ▶ 算法性质证明和时间复杂性分析

增广路径与预流推进的对比

➤ 基于增广路径的算法

- ▶ 从零流（可行解）出发
- ▶ 每步基于增广路径构造一个更大的流
- ▶ 流不能增大时为止

➤ 基于预流推进的算法

- ▶ 从源点出发
- ▶ 或者把预流（松弛解）尽可能地向前推（push）
- ▶ 或者重新标记中间结点的高度（relabel）
- ▶ 不能继续操作时为止

预流 (preflow)

► 预流 (preflow) 的性质:

► 预流函数 $f: V \times V \rightarrow \mathbb{R}$

► 容量限制

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

► 松弛的流量平衡——超额流 (excess flow) 非负

- $e(u) = \sum_v f(v, u) - \sum_v f(u, v) \geq 0$, for $\forall u \in V - \{s\}$

► 溢出 (overflow) 结点

► $e(u) > 0$

预流推进算法的直观思想

- ➡ 洪水流经河网和水库流向大海的过程
 - ▶ 最上游水库开闸，洪水迅速涌向下游水库
 - 此时下泄流量只与河道容量有关（割的容量）
 - ▶ 洪峰到达，下游水库开闸，洪水继续涌向下游 (push)
 - 流出量只和与下游水库间的河网总容量有关
 - ▶ 流出量小于流入量时，水库水位不断增高 (relabel)
 - 可能会出现新的泄洪渠道
 - 或减弱入库流量（洪水回涌）
 - ▶ 最后最上游水库出库流量与河网泄洪流量一致
 - 泄洪过程的动态平衡

预流推进的基本操作

► 基本操作

- 推进流: *push*
- 重标记: *relabel*

► 高度函数

- 流网络 G 上的预流 f 的高度函数 h
- $h[s] = |V|, h[t] = 0$
- $h[u] \leq h[v] + 1$, for $\forall (u, v) \in E_f$

► 高度函数与余量网络边的关系 (引理26.12)

- $h[u] > h[v] + 1 \Rightarrow (u, v) \notin E_f$

推进流 (PUSH)

PUSH(u, v)

- 1 ▷ Applies when: u is overflowing, $c_f(u, v) > 0$, and $h[u] = h[v] + 1$.
- 2 ▷ Action: Push $\Delta_f(u, v) = \min(e[u], c_f(u, v))$ units of flow from u to v .
- 3 $\Delta_f(u, v) \leftarrow \min(e[u], c_f(u, v))$
- 4 if $(u, v) \in E$ then $f[u, v] \leftarrow f[u, v] + \Delta_f(u, v)$
- 5 else $f[v, u] \leftarrow f[v, u] - \Delta_f(u, v)$
- 6 $e[u] \leftarrow e[u] - \Delta_f(u, v)$
- 7 $e[v] \leftarrow e[v] + \Delta_f(u, v)$

关于PUSH

► PUSH(u, v)

- 从 u 到 v 推进流（满足条件时才能推进）

► 饱和推进 (*saturating push*)

- 如果推进后边的流量饱和（剩余容量 $c_f(u, v) = 0$ ）

► 不饱和推进 (*saturating push*)

- 如果推进后边的流量不饱和（剩余容量 $c_f(u, v) > 0$ ）
- 不饱和推进把溢出结点转变为非溢出结点（引理26.13）

重标号 (RELABE)

RELABEL(u)

- 1 ▷ Applies when: u is **overflowing** and for all $v \in V$ such that $(u, v) \in E_f$, we have $h[u] \leq h[v]$.
- 2 ▷ Action: Increase the **height** of u .
- 3 $h[u] \leftarrow 1 + \min \{h[v] : (u, v) \in E_f\}$

通用的预流推进算法

INITIALIZE-PREFLOW(G, s)

1 for each vertex $u \in V[G]$

2 do $h[u] \leftarrow 0$

3 $e[u] \leftarrow 0$

4 for each edge $(u, v) \in E[G]$

5 do $f[u, v] \leftarrow 0$

6 $h[s] \leftarrow |V[G]|$

...

...

7 for each vertex $u \in Adj[s]$

8 do $f[s, u] \leftarrow c(s, u)$

9 $e[u] \leftarrow c(s, u)$

10 $e[s] \leftarrow e[s] - c(s, u)$

GENERIC-PUSH-RELABEL

GENERIC-PUSH-RELABEL(G)

1 INITIALIZE-PREFLOW(G, s)

2 **while** 如果能够应用PUSH或RELABEL操作

3 **do** 任意选取结点做PUSH或RELABEL操作

➤ 任何溢出结点 (引理26.14)

- ▶ 或者可以应用PUSH
- ▶ 或者可以应用RELABEL

➤ 证明：用高度函数的定义和不能应用PUSH蕴含着可以应用RELABEL的条件即可证明。

预流推进算法的正确性

► 证明思路

- 算法终止时，所得预流即最大流
- 算法一定会终止

引理26.15 结点高度不会降低

- 结点高度 $h[u]$ 不会减小,
并且重标号使结点高度 $h[u]$ 至少增加1
- 证明:
 - ▶ 因为 $h[u]$ 只在RELABEL过程中改变,
 - ▶ 而当结点 u 可以应用RELABEL时,
对于余量网络任意满足 $(u, v) \in E_f$ 的结点 v , $h[u] \leq h[v]$
 - ▶ 所以 $h[u] < 1 + \min \{h[v] : (u, v) \in E_f\}$, 即 $h[u]$ 必增大

引理26.16 始终满足高度函数性质

- 在通用预流推进算法过程中，结点高度始终保持遵从高度函数的性质（循环不变式）。
- 证明：（归纳法）
 - 初始时遵从高度函数性质
 - **RELABEL(u)**操作不改变高度函数性质
 - 任意出边 $(u, v) \in E_f$ ，重标号后 $h[u] \leq h[v] + 1$
 - 任意入边 $(w, u) \in E_f$ ，重标号前 $h[w] \leq h[u] + 1$ ，而重标号后 $h[u]$ 至少增加1，故 $h[w] < h[u] + 1$
 - **PUSH(u, v)**可能在 E_f 中增加边 (v, u) 或删除边 (u, v)
 - 增加时： $h[v] = h[u] - 1 < h[u] + 1$
 - 删除时： $h[v]$ 与 $h[u]$ 间无约束

引理26.17 G_f 中无从 s 到 t 的路径

- 流网络 $G=<V, E>$ 上预流 f 的余量网络 G_f 中无从 s 到 t 的路径
- 证明:
 - ▶ 反证法, 设 $p = <v_0, v_1, \dots, v_k>$ 是 G_f 中无从 s 到 t 的路径, 其中 $v_0=s, v_k=t$
 - ▶ 不失一般性, p 是一条简单路径, 故 $k < |V|$
 - ▶ 而对于 $(v_i, v_{i+1}) \in E_f, (i=1..k-1)$ 有 $h[v_i] \leq h[v_{i+1}]+1$
 - ▶ 于是 $|V| = h[s] \leq h[t]+k = k$
 - ▶ 矛盾。

定理26.18：通用预流推进算法是正确的

- 如果在流网络 $G=\langle V, E \rangle$ 上**GENERIC-PUSH-RELABEL**算法能结束，则得到的预流 f 就是 G 的最大流。
- 证明：循环不变量（每一步得到的 f 总是一个预流）
 - ▶ 初始：初始化后，得到的 f 显然是预流
 - ▶ 保持：循环中只设计**PUSH**和**RELABEL**操作。
 - **RELABEL**操作只改变高度，不改变流；
 - **PUSH**不会使任何结点的超额为负，同时保持 f 的斜对称性和容量限制。
 - ▶ 结束：不存在溢出结点，即对于 $\forall u \in V - \{s, t\}$ ， $e[u]=0$ 。而 f 始终是预流，故 f 是 G 上的流。又 G_f 中不存在从 s 到 t 的路径，故 f 是最大流。

预流推进算法的时间效率分析

- ➡ 方法：考察各种操作的执行次数上界
- ➡ 只有三种操作：
 - ▶ RELABEL：重标记
 - ▶ PUSH：推进流
 - 饱和推进 (*saturating push*)
 - 不饱和推进 (*saturating push*)
- ➡ 结论：
 - ▶ 通用预流推进算法的运行时间上界为 $O(V^2E)$

引理26.19 溢出结点 u 在 G_f 中可达 s

- 在余量网络 G_f 中存在一条从溢出结点 u 到源点 s 的简单路径
- 证明：反证法。
 - ▶ 设 $U = \{v \mid \text{在 } G_f \text{ 中 } u \text{ 可达 } v\}$ ，假设 $s \notin U$ ，设 $\bar{U} = V - U$ 。
 - ▶ 则 $\forall w \in \bar{U}, \forall v \in U, f(w, v) = 0$
 - 否则，如果 $f(w, v) > 0$ ，则
 $c_f(v, w) = f(w, v) > 0$ ，
 即 $(v, w) \in E_f$ ，这样 $w \in U$ ，矛盾。
 - ▶ 于是 $f(\bar{U}, U) = 0$ ，就有
 - $e[U] = f(V, U) = f(\bar{U}, U) + f(U, U) = f(\bar{U}, U) = 0$
 - ▶ 既有 $e[u] \neq 0$ ，矛盾。

引理26.20 结点高度 $h[u]$ 小于 $2|V|-1$

- 在通用预流推进算法过程中, 对 $\forall u \in V$ 有 $h[u] \leq 2|V| - 1$ 。
- 证明:
 - ▶ 根据定义, 源点 s 和汇点 t 满足引理。
 - ▶ 对 $\forall u \in V - \{s, t\}$, 初始时 $h[u] = 0 \leq 2|V| - 1$ 。
 - ▶ 在RELABEL(u)后, u 溢出, 根据引理26.19, 在 G_f 存在从 u 到 s 的简单路径 $p = \langle v_0, v_1, \dots, v_k \rangle$, 其中 $v_0 = u, v_k = s, k \leq |V| - 1$ 。
 - ▶ 而对于 $(v_i, v_{i+1}) \in E_f, (i=1..k-1)$ 有 $h[v_i] \leq h[v_{i+1}] + 1$
 - ▶ 于是 $h[u] \leq h[s] + k \leq 2|V| - 1$

各种操作次数的上界

► RELABEL: $2|V|^2$

- 每个结点最多重标记 $2|V|-1$ 次, 共 $|V|-2$ 个结点。

► 饱和PUSH: $2|V||E|$

- u 和 v 间饱和PUSH的次数为 $2|V|-1$
- 当 $\text{PUSH}(u,v)$ 时有 $h[v] = h[u] - 1$, 再次 $\text{PUSH}(u,v)$ 须发生在 $\text{PUSH}(v,u)$ 之后 ($h[v] = h[u] + 1$), $h[v]$ 值增加 2

► 不饱和PUSH: $4|V|^2(|V| + |E|)$

- 定义势函数 $\Phi = \sum_{v: e(v) > 0} h[v] \geq 0$ 。
- RELABEL 增加势: $\leq 2|V|$; 饱和PUSH 增加势: $\leq 2|V|$
- 不饱和PUSH 减少势至少 1。

通用预流推进算法的时间效率

- 通用预流推进算法一定能结束
 - ▶ 只需执行 $O(V^2E)$ 次操作，算法结束
- 算法运行时间上界
 - ▶ 如果RELABEL操作的开销为 $O(V)$ ； PUSH操作的开销为 $O(1)$ 。
 - ▶ 则运行时间上界为 $O(V^2E)$

重标号前置预流推进算法

► 通用预流推进算法

- 对溢出结点做流推进或重标号操作
- 操作的顺序是任意的
- 时间界为 $O(V^2E)$

► 重标号前置预流推进算法（略）

- 对溢出结点做流推进或重标号操作
- 通过对重标号的结点前置规范结点被处理的顺序
- 时间界为 $O(V^3)$
- 借助于“允许边”的概念证明和分析

最小费用流

- 在容量网络 $N = \langle V, E, c, s, t \rangle$ 添加费用函数 $w: E \rightarrow R^*$

称作容量-费用网络，记作 $N = \langle V, E, c, w, s, t \rangle$

- 设 f 是 N 上的可行流， f 是的费用为

$$w(f) = \sum_{\langle i, j \rangle \in E} w(i, j) f(i, j)$$

- 最小费用流:

流量为 v 的 N 上所有可行流中费用最小者

最小费用流算法

- 负回路算法：从可行流开始，迭代直至费用最低
- 最短路径算法：从零流开始，迭代直至满足流量约束

容量-费用网络的辅助网络

► 设容量-费用网络 $N = \langle V, E, c, w, s, t \rangle$,
 f 是 N 上的可行流,

► 其辅助网络的定义为

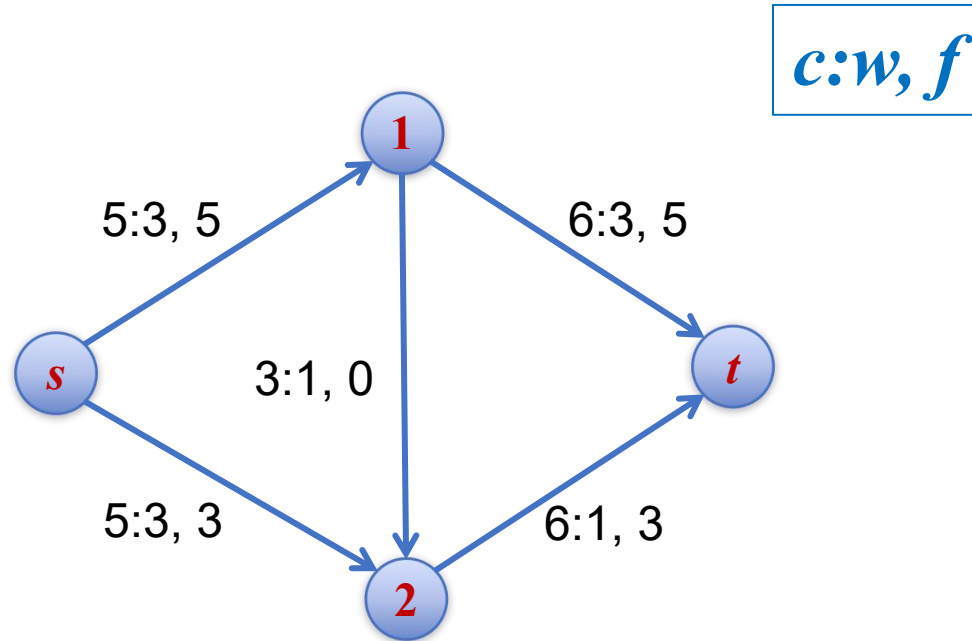
$$N(f) = \langle V, E(f), ac, aw, s, t \rangle$$

其中 $E(f)$ 和 ac 的定义与最大流辅助网络一致

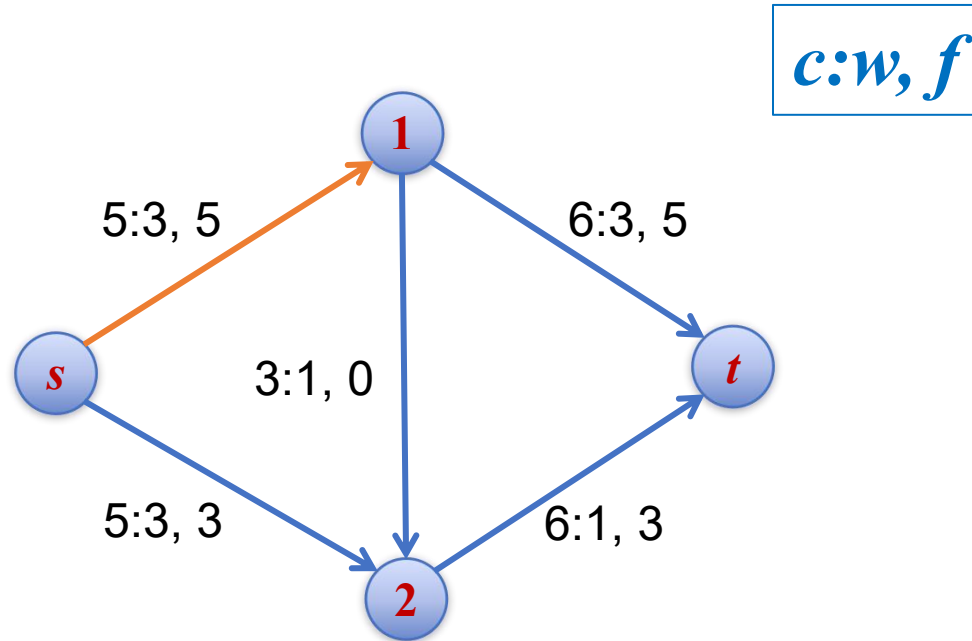
辅助费用 aw 为

$$aw(i, j) = \begin{cases} w(i, j) & \langle i, j \rangle \in E^+(f) \\ -w(j, i) & \langle i, j \rangle \in E^-(f) \end{cases}$$

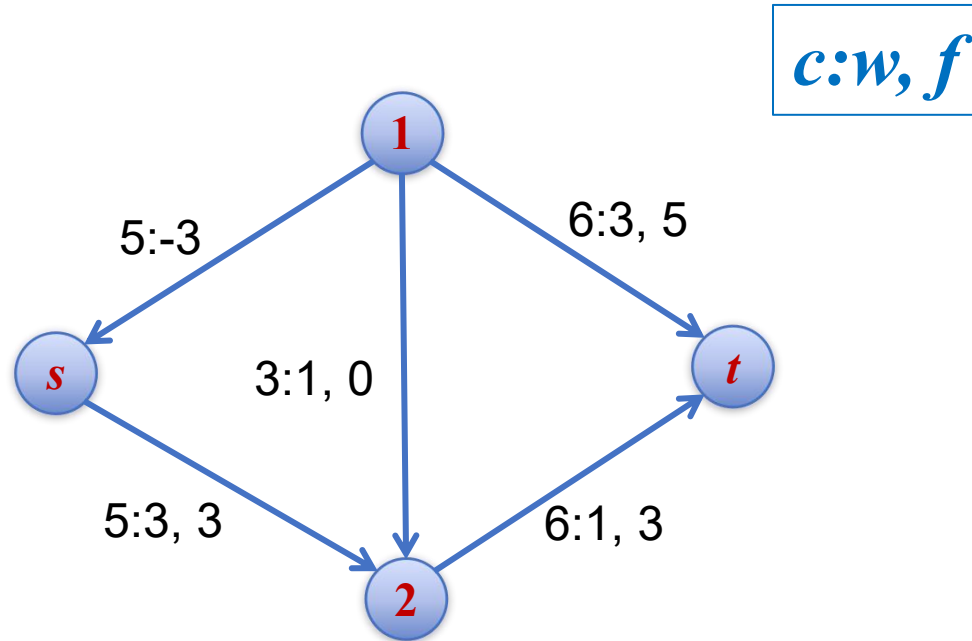
容量-费用网络与其辅助网络



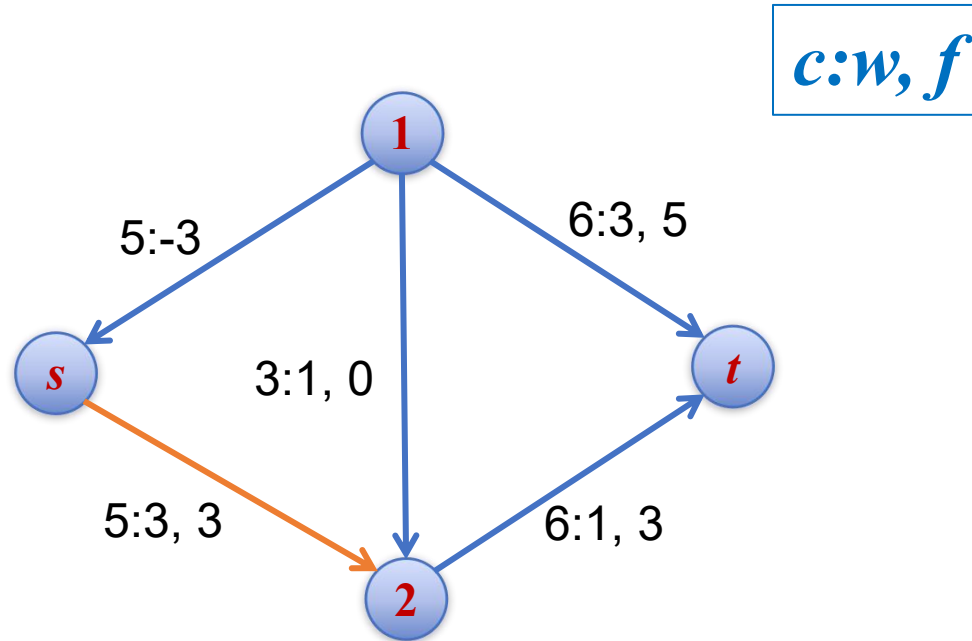
容量-费用网络与其辅助网络



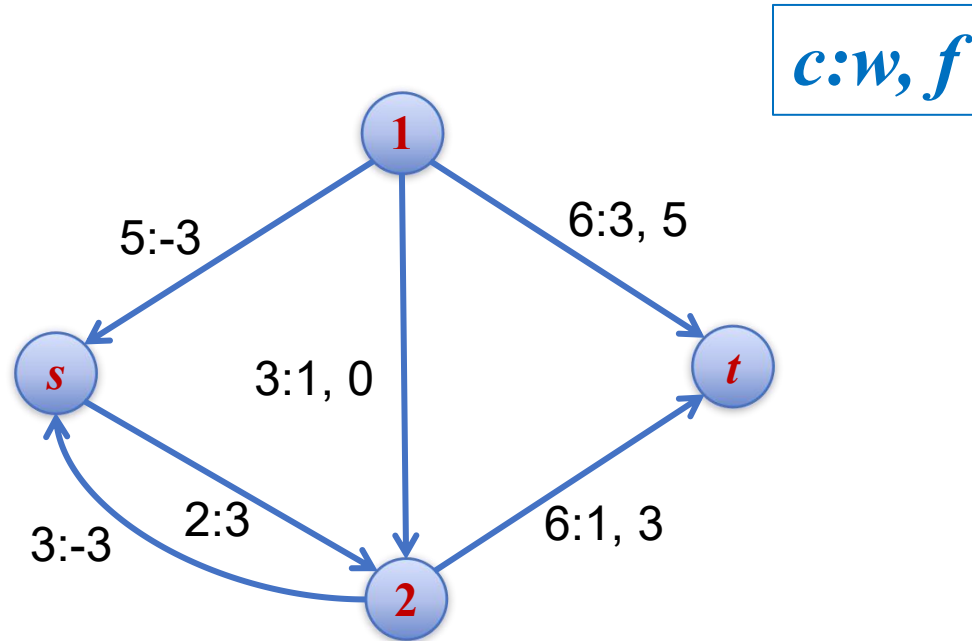
容量-费用网络与其辅助网络



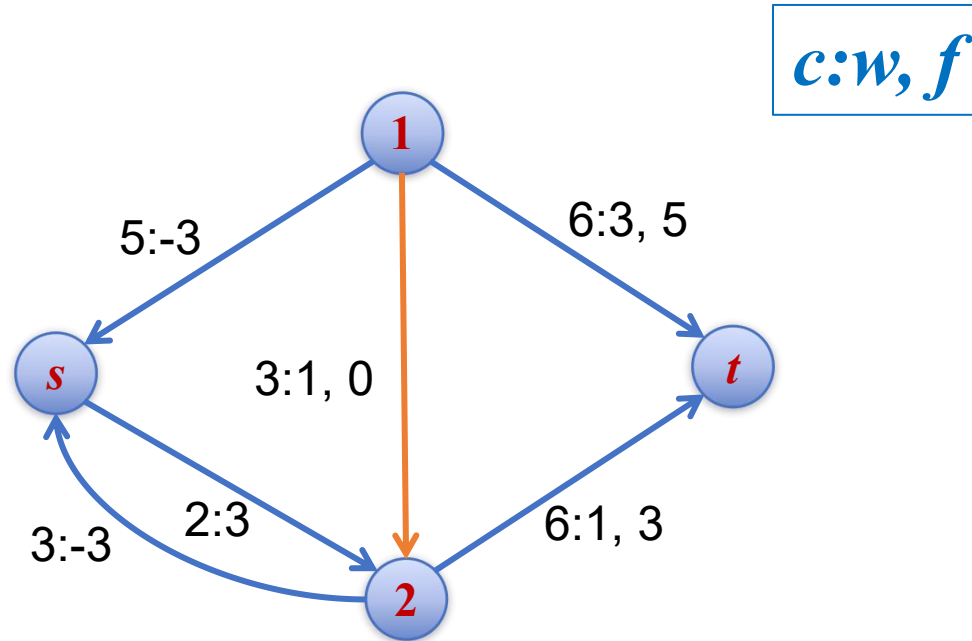
容量-费用网络与其辅助网络



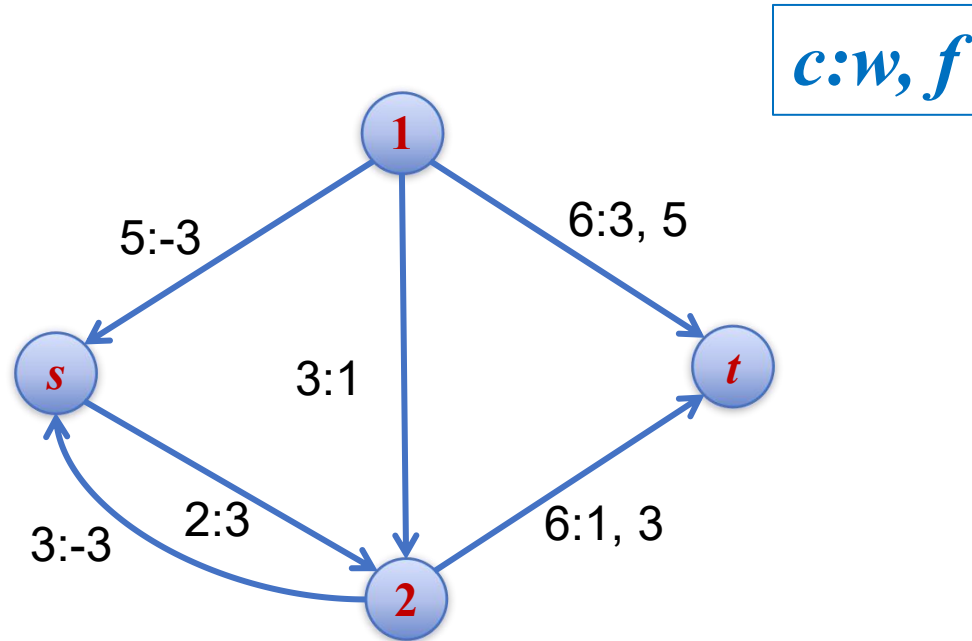
容量-费用网络与其辅助网络



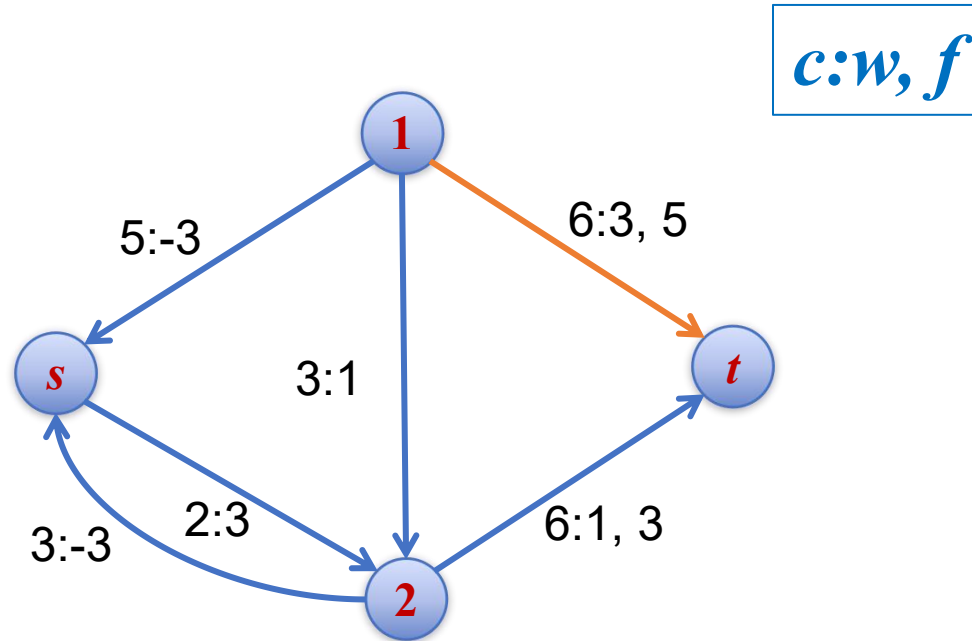
容量-费用网络与其辅助网络



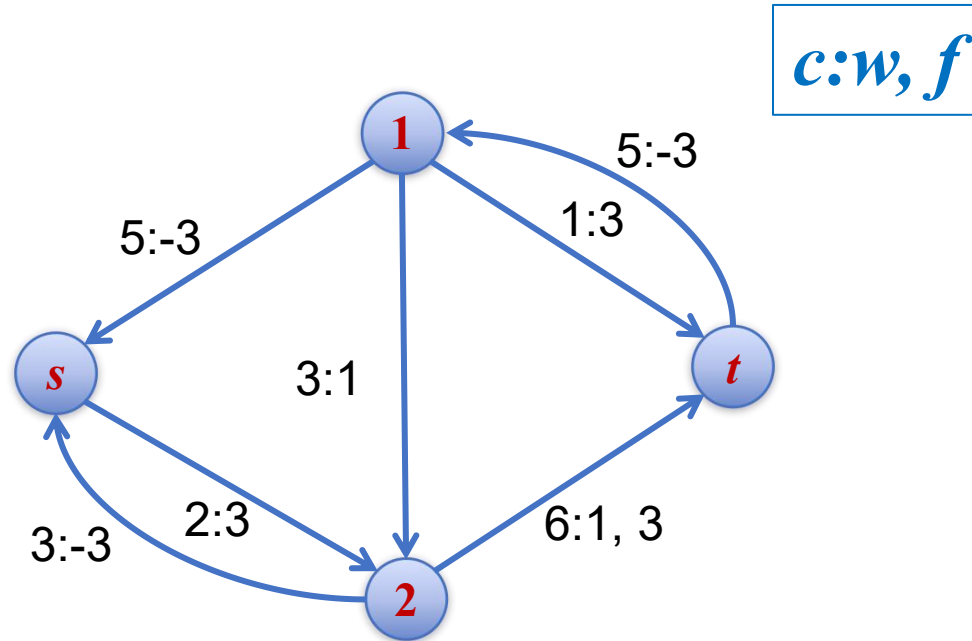
容量-费用网络与其辅助网络



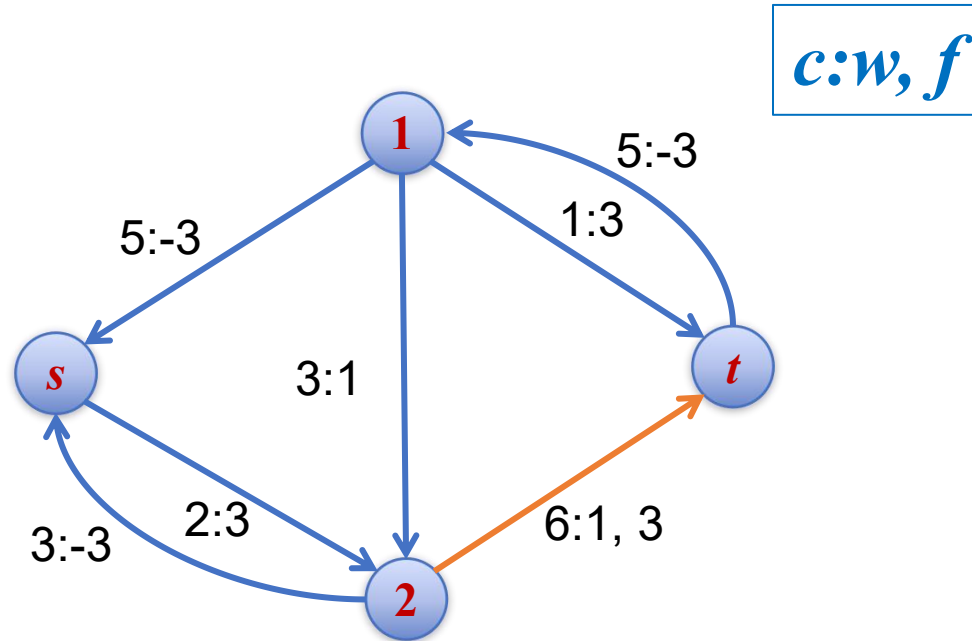
容量-费用网络与其辅助网络



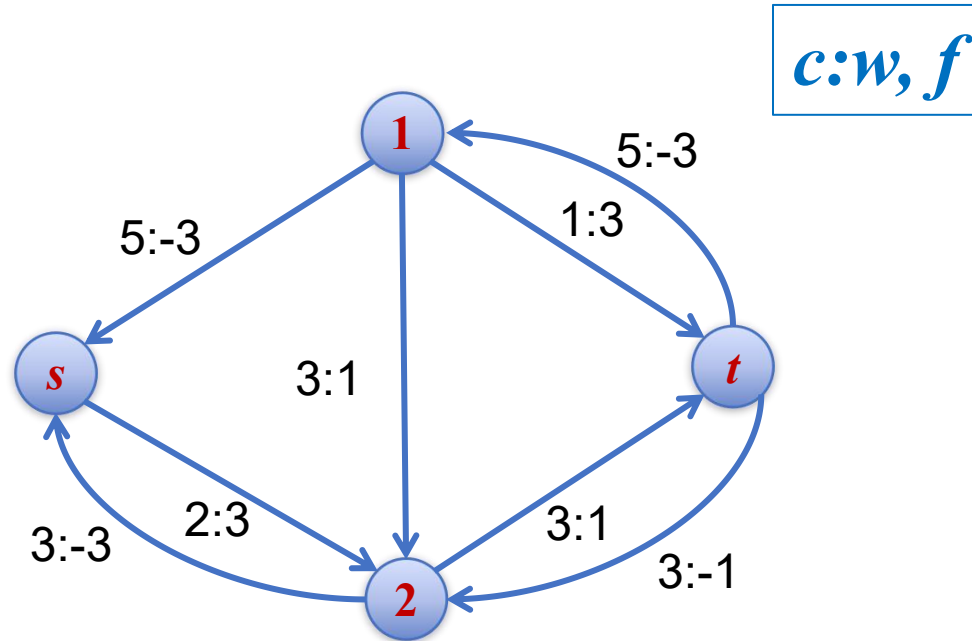
容量-费用网络与其辅助网络



容量-费用网络与其辅助网络



容量-费用网络与其辅助网络



费用流的可叠加性

► 引理7.9

f 是容量-费用网络 N 上的可行流,

g 是辅助网络 $N(f)$ 上的可行流

$f' = f + g$, 则

$$w(f') = w(f) + \alpha w(g)$$

证明: 略, 类似引理7.5的证明。

圈流与环流量

► 设容量-费用网络 $N = \langle V, E, c, w, s, t \rangle$,
 C 是 N 中一条边不重复的回路,

► C 上的 圈流 h^C 定义为

$$\forall \langle i, j \rangle \in E(C), \quad h^C(i, j) = \delta;$$

$$\forall \langle i, j \rangle \in E - E(C), \quad h^C(i, j) = 0;$$

其中 h^C 环流量 $\delta = \min_{\langle i, j \rangle \in E(C)} \{c(i, j)\} > 0$

► h^C 是一个可行流, 流量为零但通常费用非零

$$v(h^C) = 0, \quad w(h^C) = \delta \cdot w(C)$$

其中 $w(C) = \sum_{\langle i, j \rangle \in E(C)} w(i, j)$

圈流与环流量

f 是容量-费用网络 N 上的可行流,

h^C 是辅助网络 $N(f)$ 上的圈流,

$$f' = f + h^C$$

f' 是 N 上的可行流, 且

$$v(f') = v(f)$$

$$w(f') = w(f) + \delta \cdot aw(C)$$

如果 C 是 $N(f)$ 上关于 aw 的负回路, 则

$$aw(C) < 0 \Rightarrow w(f') < w(f)$$

最小费用流负回路算法

1. 求得 N 上一个流量 v_0 为的可行流 f
2. 构造辅助网络 $N(f)$
3. 利用 Bellman-Ford 算法检查 $N(f)$ 中是否存在负回路
4. 如果 $N(f)$ 存在关于 aw 的负回路 C
5. 令 $f' = f + h^C$, $f \leftarrow f'$, 重复步骤2
6. 否则, $N(f)$ 不存负回路, 算法结束

带负权的最短路径问题

➡ 最短路存在性的充要条件：无负回路

▶ 必要性

- 如果有负回路，每绕一圈路径权值和必减少

▶ 充分性

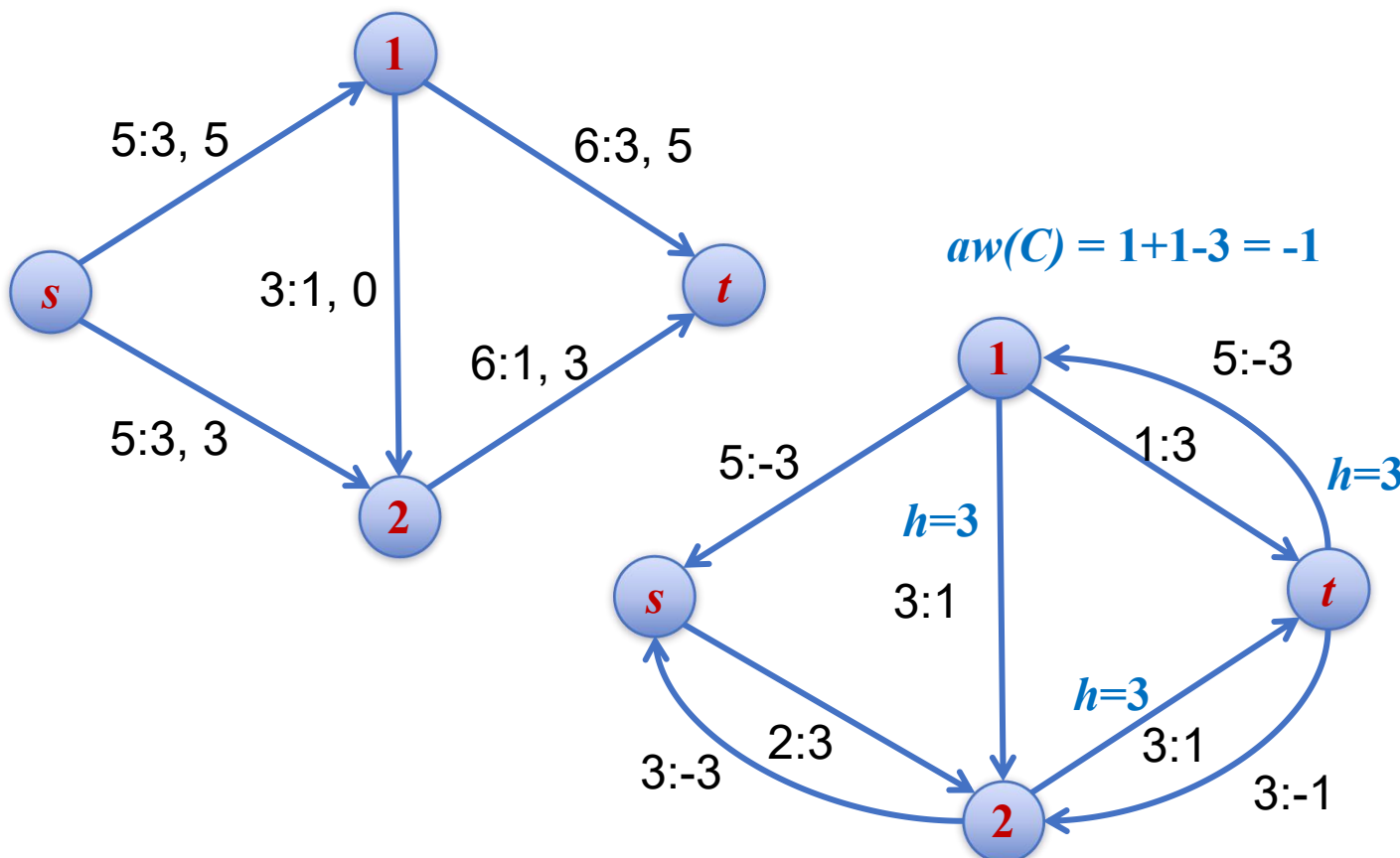
- 最短路必是简单路径（无重复顶点）
- 简单路径的数量是有限的，必有最短路

➡ 检测负回路的 Bellman-Ford 算法

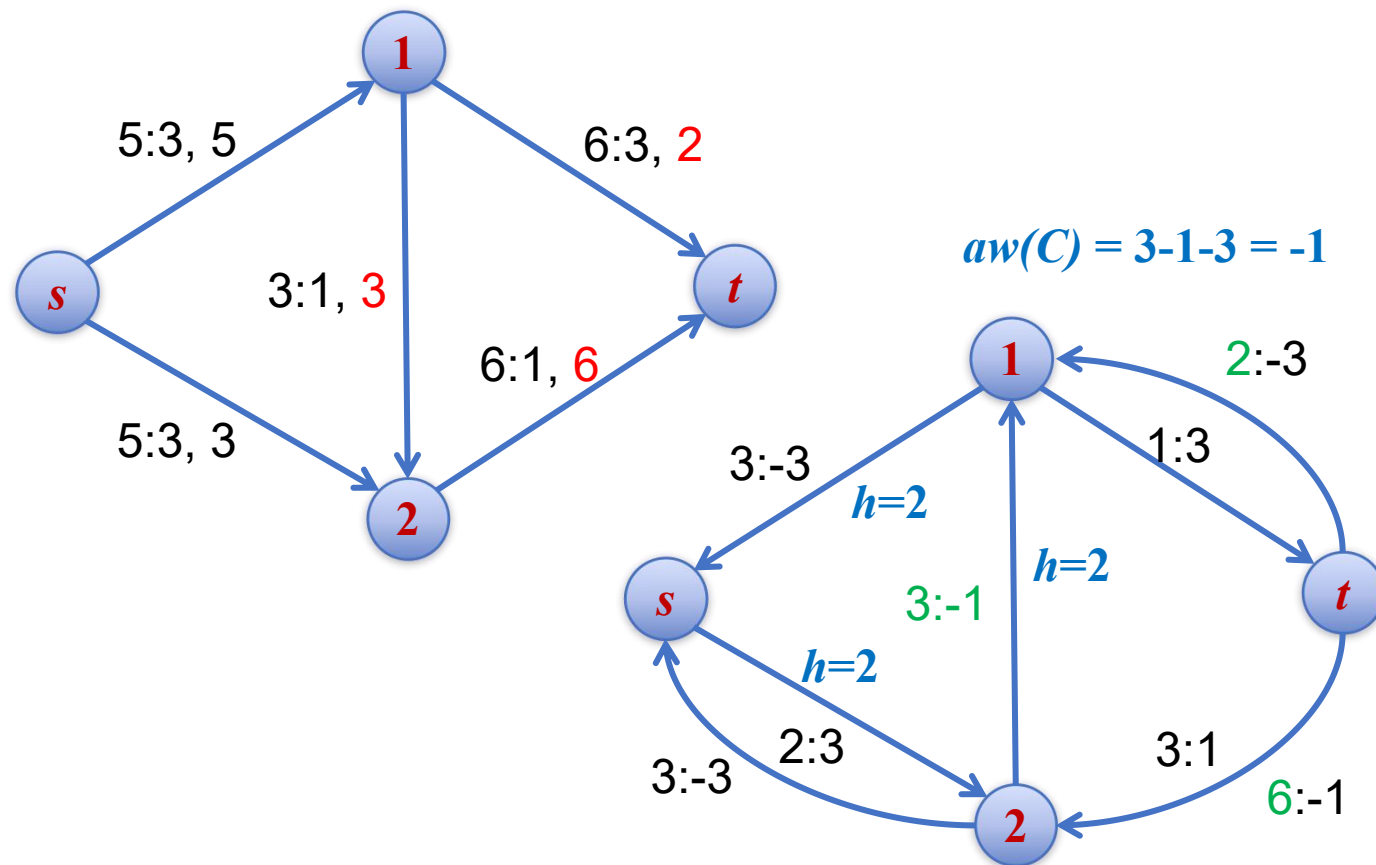
- ▶ 迭代 $|V|$ 次后，第 $|V| + 1$ 次仍能降低最短路长度 \Rightarrow 存在负回路

- ▶ 复杂度 $O(|V||E|)$

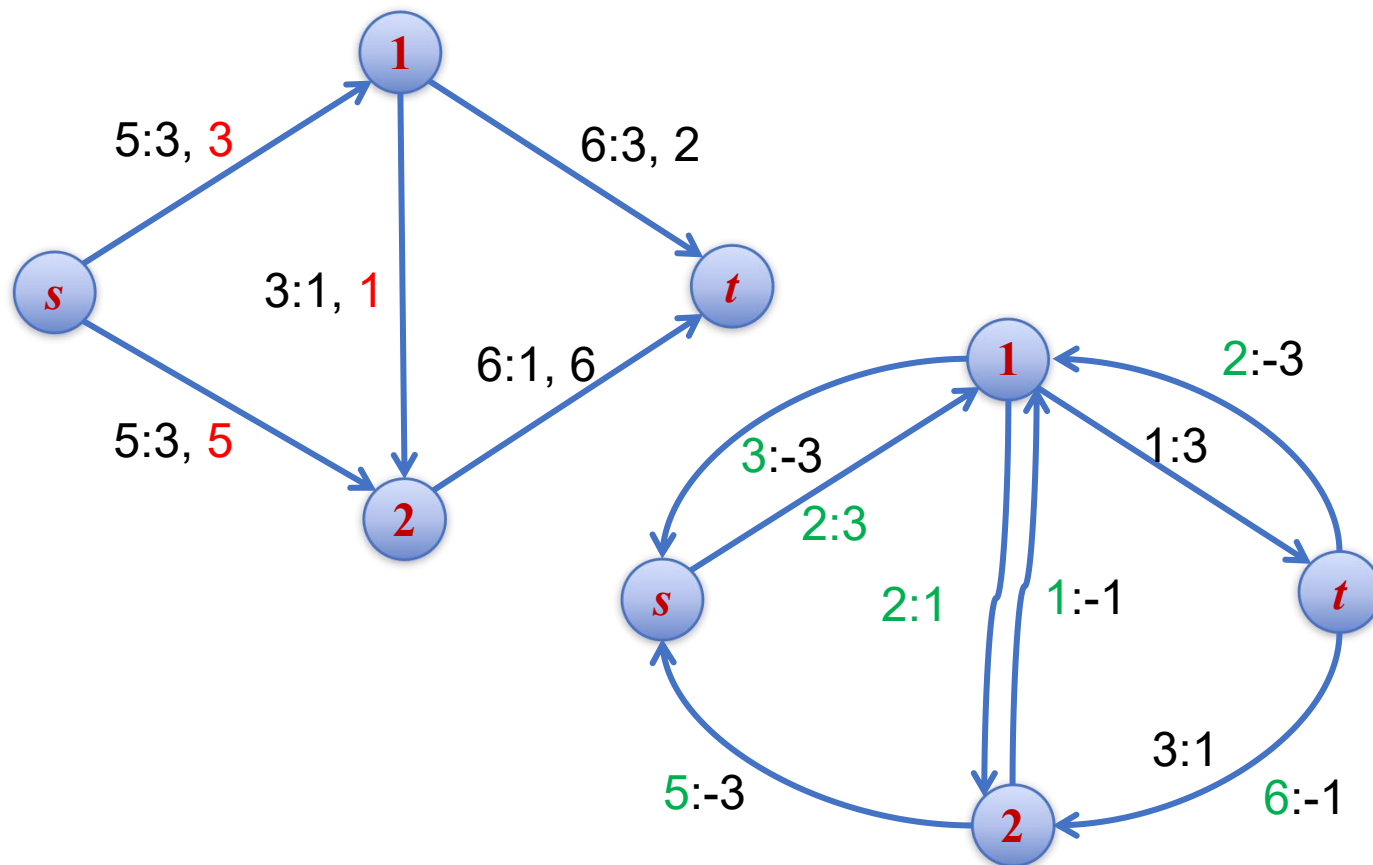
示例：负回路算法（1）



示例：负回路算法（2）



示例：负回路算法（3）



负回路算法的正确性

- 定理7.5 设 f 是容量-费用网络 N 上流量为 v_0 的可行流，则 f 是最小费用流 当且仅当 $N(f)$ 中不存在以辅助费 aw 为权的负回路

证明:

必要性显然：基于负回路可构造费用更小的流

充分性在于：假设存在费用更小的可行流 f'

则 $g = f' - f$ 将是 $N(f)$ 中流量为零的可行流

g 只能是一些环流的并，不存在负回路则 $w(g) \geq 0$

于是 $w(f') = w(f) + w(g) \geq w(f)$ ，矛盾

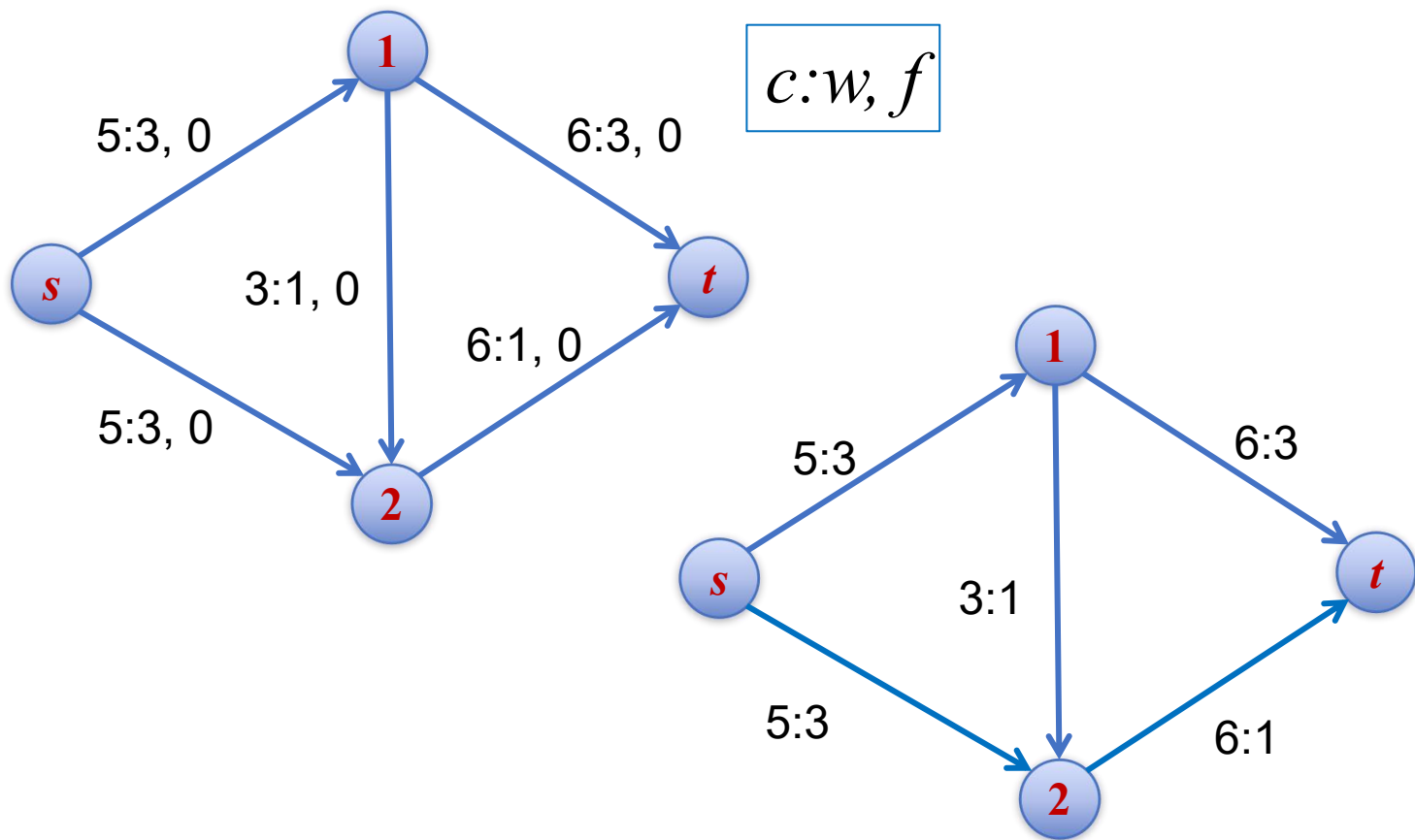
负回路算法的有限终止性

- 假设边的容量都是整数
- 计算过程中的可行流都是整数值、圈流的环流量也是整数值
- $w(f)$ 的值是有限的整数每次消减负回路至少把 $w(f)$ 的值减少 1
- 计算流量为 v_0 的可行流与循环消减负回路求得最小费用流均可在有限步内终止
- 时间复杂度 $O(|V||E|W_{init})$ 或 $O(|V||E|^2 c_{max} w_{max})$
 - ▶ 单次负回路检测算法耗时 $O(|V||E|)$, 最多检测 W_{init} 次
 - ▶ 其中 $W_{init} = w(f_{init})$ 是初始可行流 f_{init} 的费用
 - ▶ 费用 W_{init} 上界是 $|E|c_{max}w_{max}$, 其中 c_{max} 是最大边容量、 w_{max} 是最大边费用绝对值

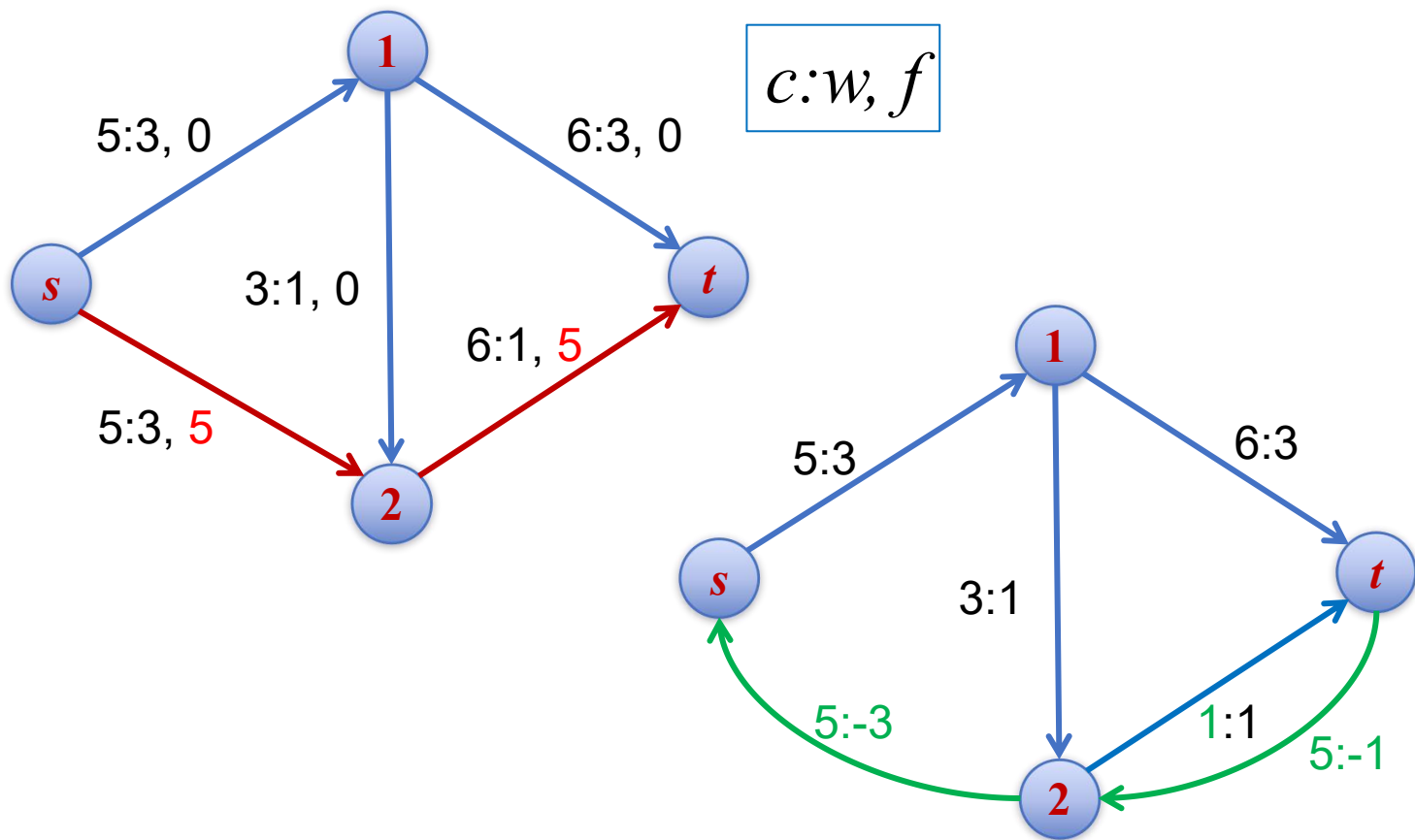
最小费用流的最短路径算法

- 从初始最小费用流（零流）开始
- 通过费用最小的s-t增广链扩张流 f
- 直到 f 的流量值等于 v_0 为止
- 保证该算法正确，只需证明步骤2得到的始终是当前流量下的最小费用流
- 时间复杂度分析
 - ▶ 最多 v_0 次迭代，每次迭代求一次最短路
 - ▶ 如果用堆优化的 Dijkstra 算法，总复杂度是 $O(v_0(|V| + |E|\log |V|))$

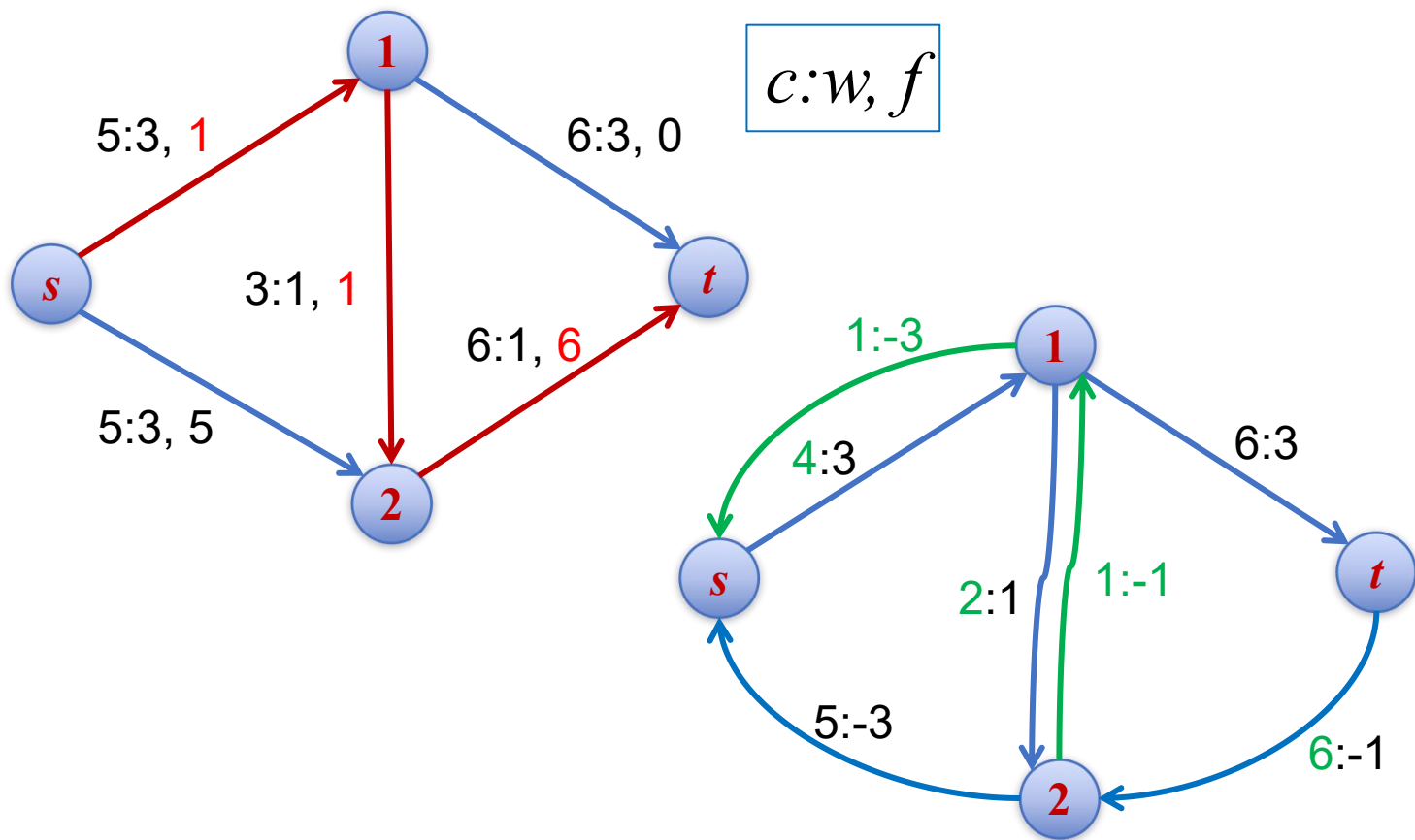
示例：求最小费用最大流 $v(f)=10$



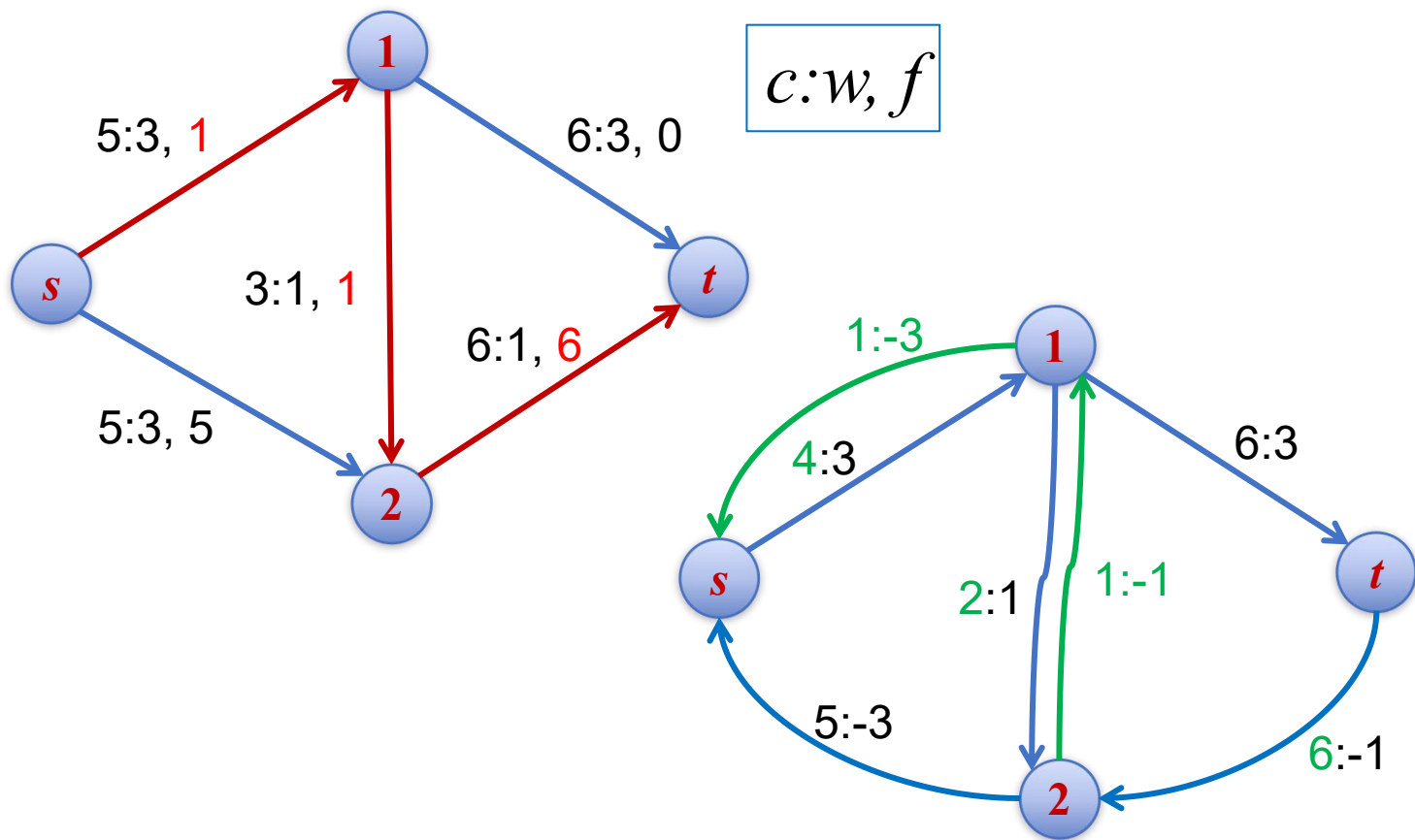
延最短路增加流再求辅助网络



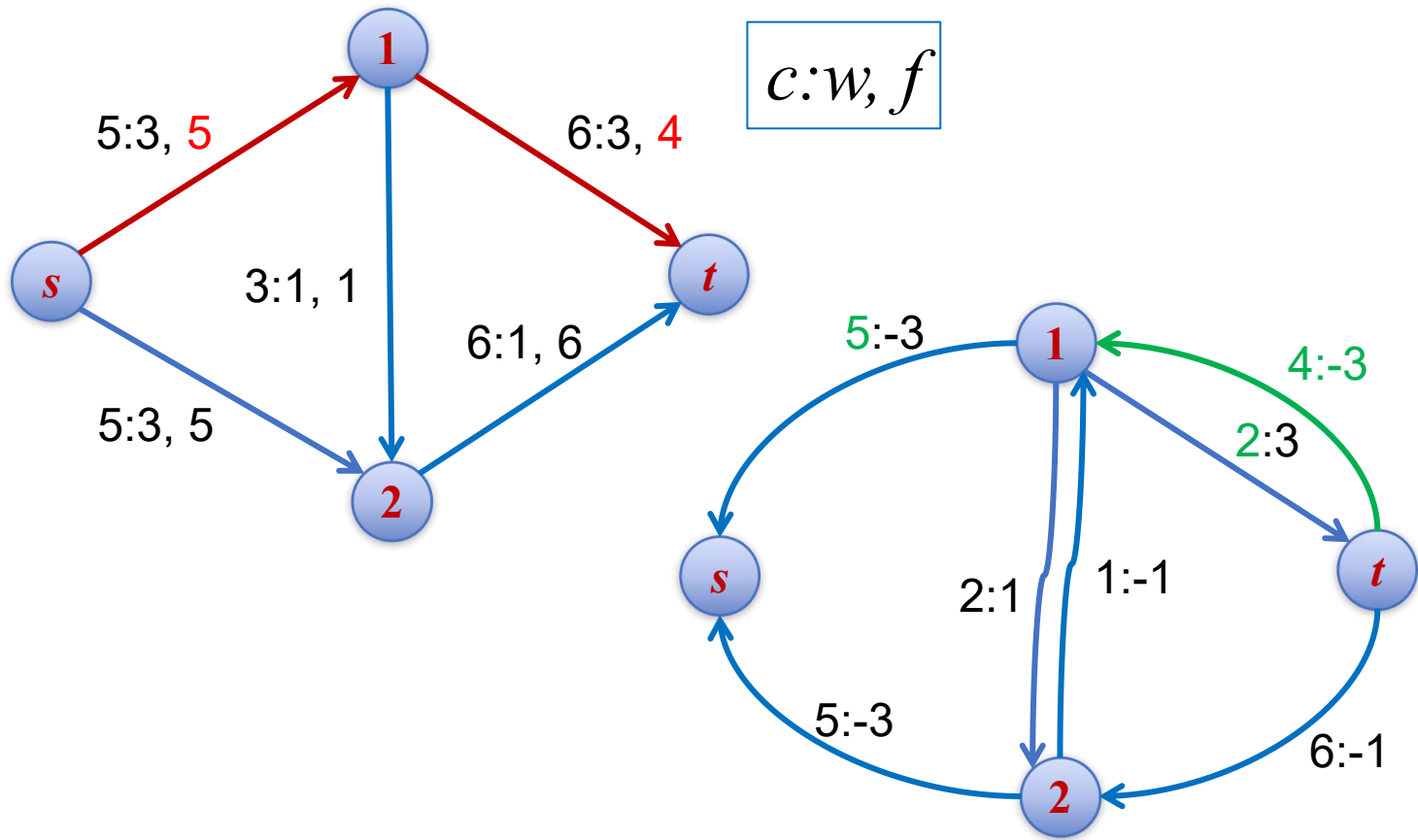
再延最短路增加流并求辅助网络



再延最短路增加流并求辅助网络



最后得到最小费用最大流

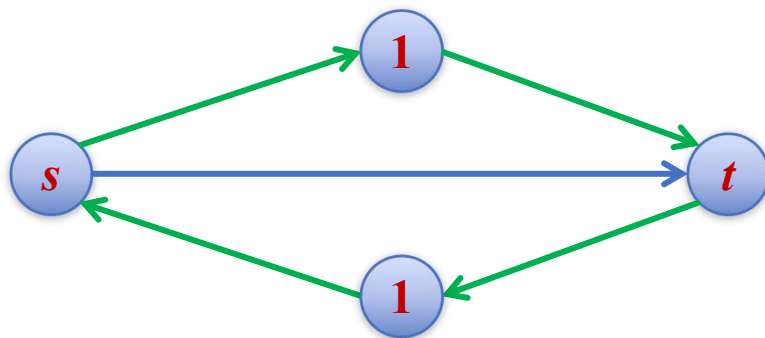


算法的正确性

思路：假设存在负回路则导出矛盾即可

引理 7.12

设有向图 $D = \langle V, E \rangle$ 没有孤立点，顶点 s 的出度比入度大 1， t 的入度比出度大 1，其余顶点的出度等于入度，则 D 可表示成一条 s - t 路径与若干条回路的并。



沿最短路扩张一定得到最小费用流

定理 7.6

设 f 是 N 上流量为 v_0 的最小费用流,

P 是 $N(f)$ 中权 aw 的 s - t 最短路径,

g 是 P 上流量为 θ 的可行流,

则 $f' = f + g$ 是流量为 $v_0 + \theta$ 的最小费用流。

其中对 $\forall \langle i, j \rangle \in E(f)$

$$g(i, j) = \begin{cases} \theta & \langle i, j \rangle \in E(P) \\ 0 & otherwise \end{cases}$$

$$0 < \theta \leq \min \{ac(i, j) | \langle i, j \rangle \in E(P)\}$$

定理 7.6 证明

反正法：假设 f' 不是最小费用流，
则 $N(f')$ 中存在权 aw 的负回路 C 。

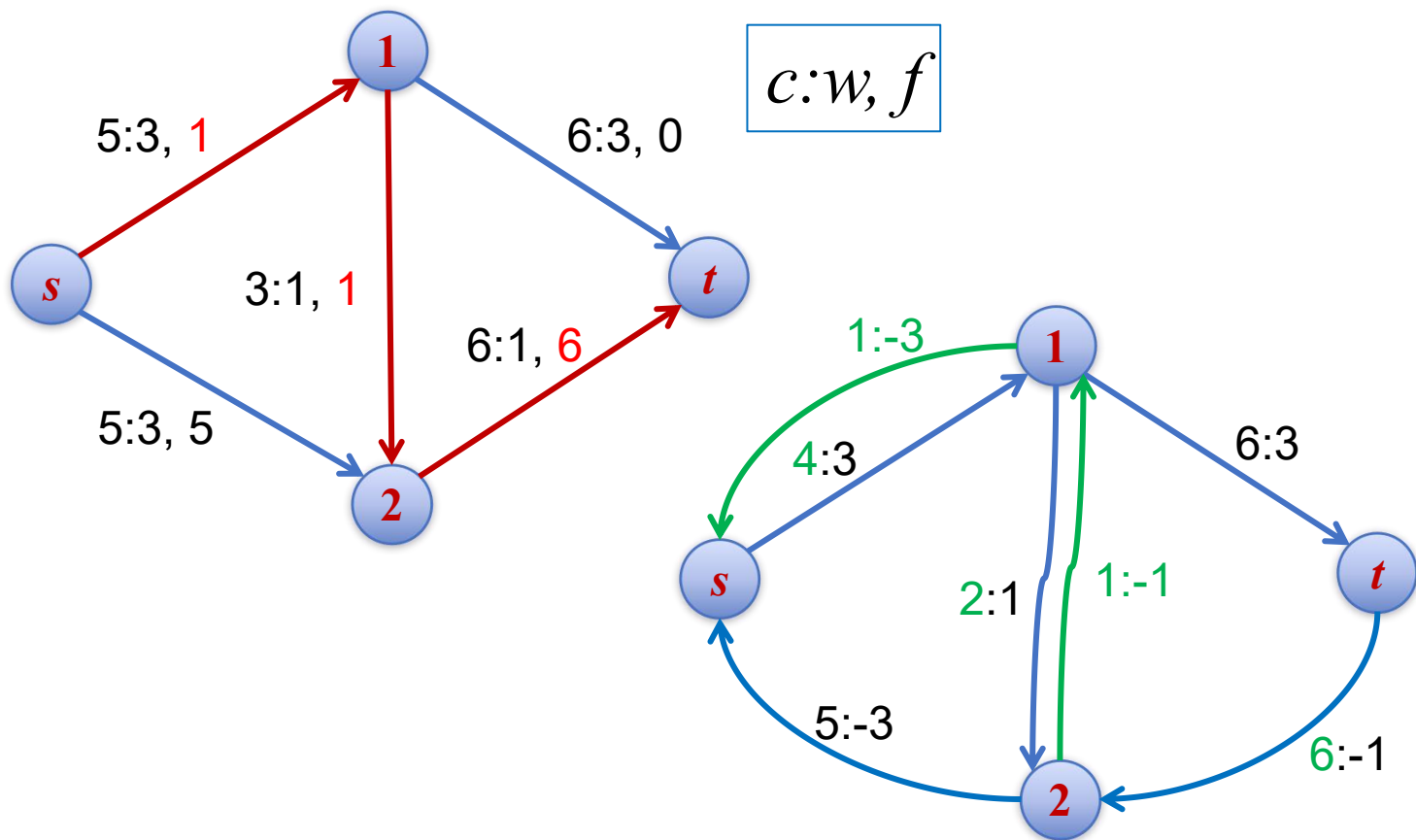
考察 $N(f)$ 与 $N(f')$ 间的差异

f 与 f' 仅在对 P 的增广链上不同

$E(f') - E(f)$ 定与 P 有关，即有

$$\langle i, j \rangle \in E(f') - E(f) \Rightarrow \langle j, i \rangle \in E(P)$$

回顾刚才的例子



定理 7.6 证明（续）

负回路 C 中必有 $E(f') - E(f)$ 中的边

因为 f 是最小费用流， $N(f)$ 中无负回路
设这些边为

$$\langle i_1, j_1 \rangle, \langle i_2, j_2 \rangle, \dots, \langle i_r, j_r \rangle$$

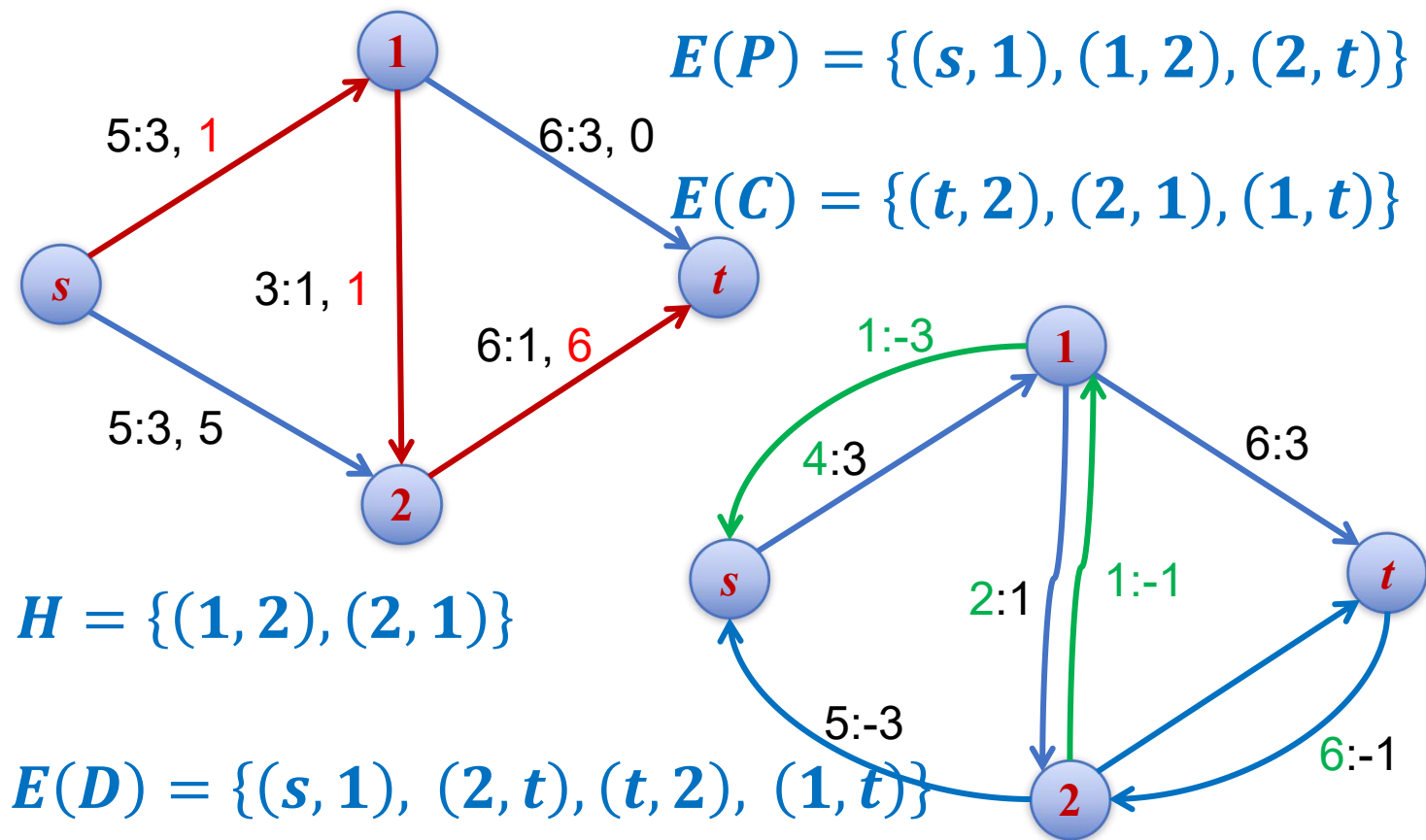
则有

$$\langle j_1, i_1 \rangle, \langle j_2, i_2 \rangle, \dots, \langle j_r, i_r \rangle \in E(P)$$

记这 $2r$ 条的边集为 H

从 P 和 C 构成的子图中删除 H （以及孤立点）记为 D

再回顾刚才的例子



定理 7.6 证明 (续)

子图 D 满足引理 7.12

它由一条 s - t 路径 P'

与若干条回路 C_1, C_2, \dots, C_l 组成

于是

$$aw(D) = aw(P') + \sum_{i=1}^l aw(C_i)$$

而 $aw(H) = 0$, 于是

$$aw(P) + aw(C) = aw(D) + aw(H)$$

定理 7.6 证明 (续)

$$aw(P') = aw(P) + aw(C) - \sum_{i=1}^l aw(C_i)$$

假设条件

$$aw(C) < 0$$

而

$$aw(C_i) \geq 0, i \in \{1, 2, \dots, l\}$$

于是

$$aw(P') < aw(P)$$

与 P 是 $N(f)$ 中权 aw 的 s - t 最短路径矛盾 ■

小结

➡ 最大流

- ▶ 复习：Ford-Fulkerson类算法
- ▶ 基于“预流”推进的算法

➡ 最小费用流

- ▶ 负回路算法：从可行流开始，迭代直至费用最低
- ▶ 最短路径算法：从零流开始，迭代直至满足流量约束