

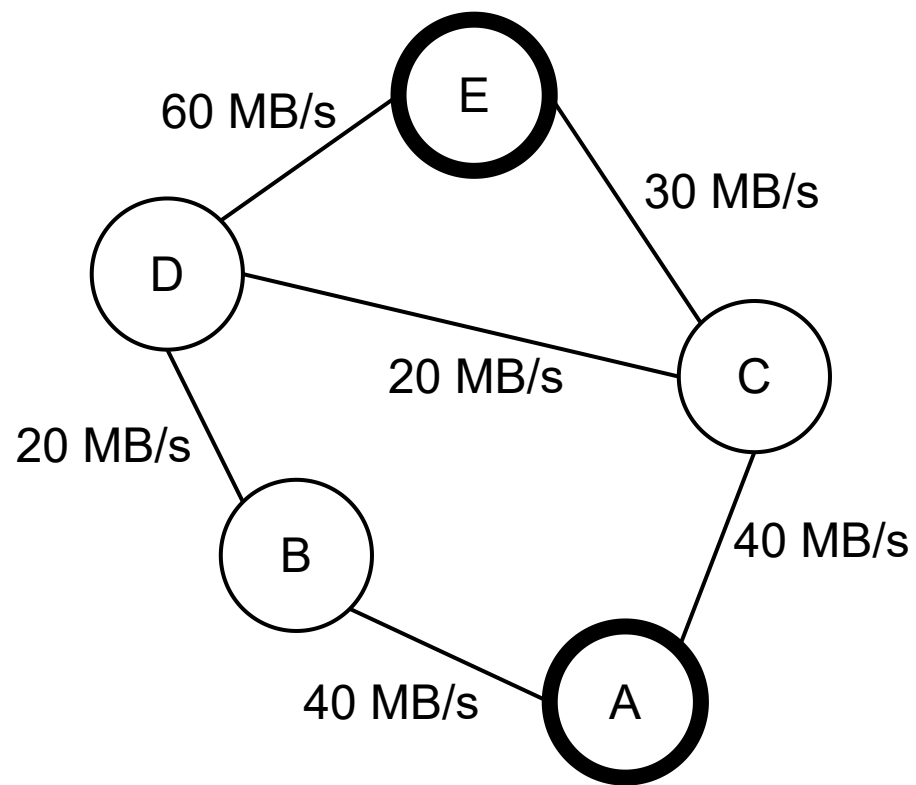
第9讲 网络流问题 (上)

罗国杰

gluo@pku.edu.cn

2023年春季学期

网络 & 流量

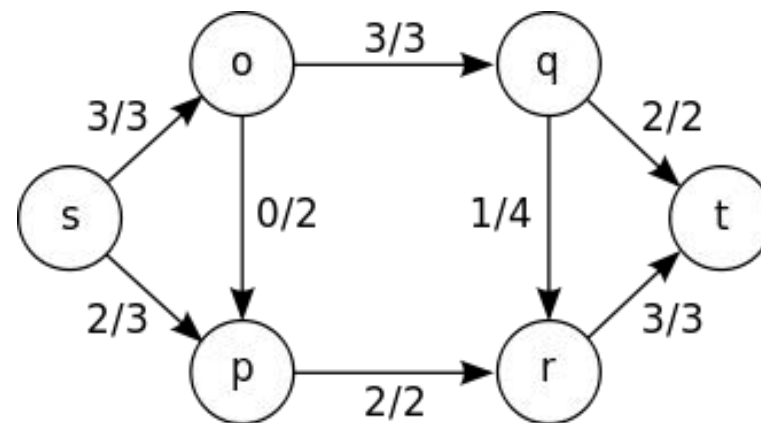


- ➡ A 能否以 60 MB/s 流量向 E 传数据?
- ➡ A 能否以 70 MB/s 流量向 E 传数据?

最大流问题

- 给定：有向连通网络、容量
 - 右图边标记：流量/容量
- 求解：最大流
 - 流量定义：流出s流量 - 流入s流量
- 约束：容量限制、平衡条件
- 线性规划表示（右图例子）

$$\begin{aligned}
 \max \quad & f_{s,o} + f_{s,p} \\
 \text{s.t.} \quad & f_{s,o} = f_{o,p} + f_{o,q} \\
 & f_{s,p} + f_{o,p} = f_{p,r} \\
 & f_{o,q} = f_{q,r} + f_{q,t} \\
 & f_{p,r} + f_{q,r} = f_{r,t} \\
 & f_{u,v} \leq c_{u,v} \quad (u,v) \in E
 \end{aligned}$$



source: wikipedia.org

最小费用流

► 给定：有向连通网络、容量、费用

► 右图边标记：(容量, 费用)

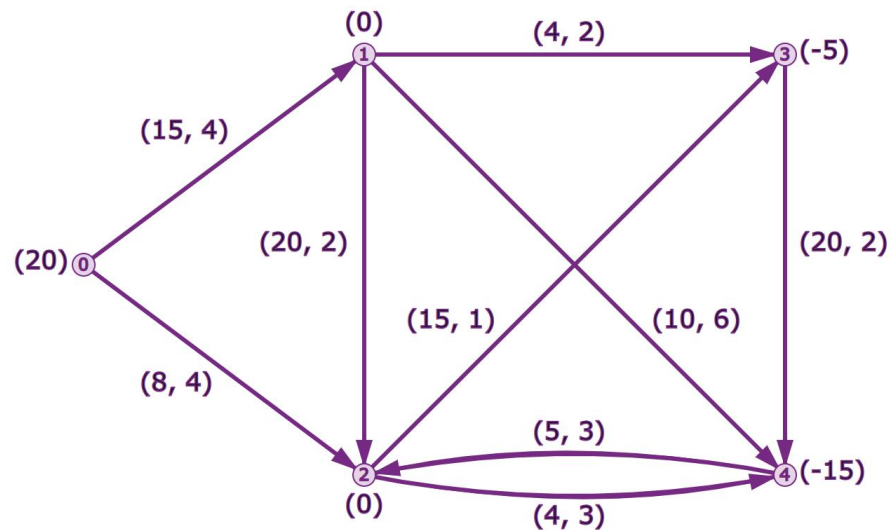
► 右图点标记：流量供应

► 求解：最小费用流

► 总费用： $\sum \text{边费用} \times \text{边流量}$

► 约束：容量限制、平衡条件

► 线性规划表示（略）





- LEMON stands for Library for Efficient Modeling and Optimization in Networks.

- ▶ <https://lemon.cs.elte.hu/trac/lemon>

- 输入：有向连通网络、容量、费用

- ▶ `Graph::addNode(Graph::Node);`
- ▶ `Graph::addArc(Graph::Arc);`
- ▶ `Solver::lowrMap(Graph::ArcMap);`
- ▶ `Solver::upperMap(Graph::ArcMap);`
- ▶ `Solver::costMap(Graph::ArcMap);`

- 输出：流量

- ▶ `Solver::run();`
- ▶ `Solver::flowMap(Graph::ArcMap);`

- 最大流算法

- ▶ Preflow push-relabel (with various heuristics)

- 可行流通算法

- ▶ Generalized version (push-relabel algorithm)

- 最小费用流算法

- ▶ Cycle-Canceling algorithms
 - two of which are strongly polynomial.
- ▶ Capacity Scaling algorithm
 - based on the successive shortest path method.
- ▶ Cost Scaling algorithm
 - based on push/augment and relabel operations.
- ▶ Primal Network Simplex algorithm
 - with various pivot strategies.

本节内容

➤ 网络流

- ▶ 容量网络、容量、源点、汇点
- ▶ 可行流、流量、最大流
- ▶ 割集、最小割集
- ▶ 最大流最小割集定理

➤ Ford-Fulkerson 算法

- ▶ f 是最大流 \Leftrightarrow 不存在关于 f 的 s - t 增广链
- ▶ 时间复杂度分析 $O(E |f^*|)$

➤ 例：Baseball Elimination Problem

➤ Ford-Fulkerson 方法的优化

- ▶ Edmonds-Karp 算法 $O(VE^2)$ (略)
- ▶ Dinic 算法 (优化版) $O(V^3)$
 - 容量网络关于流量 f 的余量网络 $N(f)$
 - 分层辅助网络

最大流/最小割模型有广泛应用

- Open-pit mining.
- Baseball elimination.
- Bipartite matching.
- Egalitarian stable matching.
- Network reliability.
- Network connectivity.
- Distributed computing.
- Image segmentation.
- Multi-camera scene reconstruction.
- Markov random fields.
- Data mining.
- Network intrusion detection.
- Security of statistical data.
- Sensor placement for homeland security.
-

容量网络的定义

- 设有向连通图 $N = \langle V, E \rangle$
记 $n = |V|$, $m = |E|$
- 容量: 每条边 $\langle i, j \rangle$ 均对应一个非负实数 $c(i, j)$
- N 有两个特殊的顶点: s 和 t ,
 - s 称作发点 (或源)
 - t 称作收点 (或汇)
 - 其余的顶点称作中间节点
- 称 N 为容量网络, 记作 $N = \langle V, E, c, s, t \rangle$

容量网络上的可行流与最大流

设 $f: E \rightarrow R^*$, 其中 R^* 为非负实数集, 满足:

① 容量限制: $\forall \langle i, j \rangle \in E, f(i, j) \leq c(i, j)$

② 平衡条件: $\forall i \in V - \{s, t\}$

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

称 f 是 N 上的一个可行流

称发点 s 的净流出量为 f 的流量, 记作 $v(f)$, 即

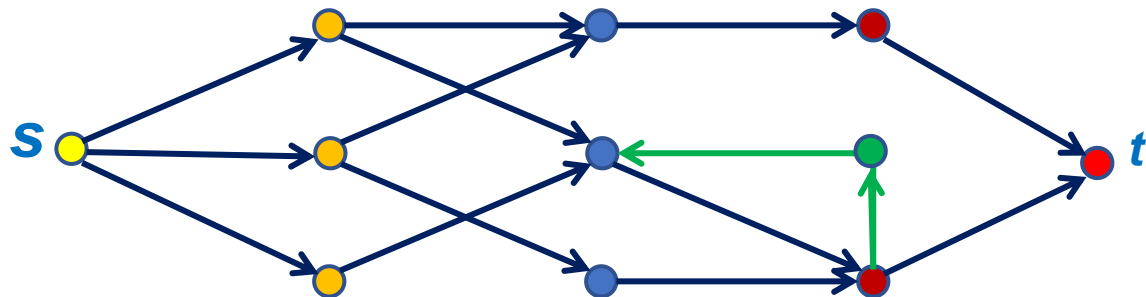
$$v(f) = \sum_{\langle s, j \rangle \in E} f(s, j) - \sum_{\langle j, s \rangle \in E} f(j, s)$$

最大流: 流量最大的可行流称作最大流

最大流问题: 求给定容量网络 N 上的最大流 f^*

关于网络流建模

- ➡ 多个发点（源）与多个收点（汇）的问题
- ➡ 正向边与反向边共存的问题



最大流问题的线性规划表述

$$\max v(f)$$

s.t.

$$f(i, j) \leq c(i, j), \quad \langle i, j \rangle \in E$$

$$\sum_{\langle i, j \rangle \in E} f(i, j) - \sum_{\langle j, i \rangle \in E} f(j, i) = 0, \quad i \in V - \{s, t\}$$

$$v(f) - \sum_{\langle s, j \rangle \in E} f(s, j) + \sum_{\langle j, s \rangle \in E} f(j, s) = 0,$$

$$f(i, j) \geq 0, \quad \langle i, j \rangle \in E$$

$$v(f) \geq 0$$

能求解线性规划的算法都能求解最大流问题

最大流问题有更有有效的求解算法

割集、割集的容量、最小割集

设容量网络 $N = \langle V, E, c, s, t \rangle$,

$A \subset V$ 且 $s \in A, t \in \bar{A}$, 称

$$(A, \bar{A}) = \{ \langle i, j \rangle | \langle i, j \rangle \in E \text{ 且 } i \in A, j \in \bar{A} \}$$

为 N 的割集; 称

$$c(A, \bar{A}) = \sum_{\langle i, j \rangle \in (A, \bar{A})} c(i, j)$$

为割集 (A, \bar{A}) 的容量。

容量最小的割集称作最小割集。

可行流的流量与割集上的流量

引理7.1 设容量网络 $N = \langle V, E, c, s, t \rangle$,

f 是 N 上的任一可行流,

$A \subset V$ 且 $s \in A$, $t \in \bar{A}$, 则

$$v(f) = \sum_{\langle i, j \rangle \in (A, \bar{A})} f(i, j) - \sum_{\langle j, i \rangle \in (\bar{A}, A)} f(j, i)$$

证明:

$$v(f) = \sum_{\langle s, j \rangle \in E} f(s, j) - \sum_{\langle j, s \rangle \in E} f(j, s)$$

引理7.1的证明 (续1)

$$\begin{aligned}v(f) &= \sum_{\langle s,j \rangle \in E} f(s,j) - \sum_{\langle j,s \rangle \in E} f(j,s) + \\&\quad \sum_{i \in A - \{s\}} \left\{ \sum_{\langle i,j \rangle \in E} f(i,j) - \sum_{\langle j,i \rangle \in E} f(j,i) \right\} \\&= \sum_{i \in A} \left\{ \sum_{\langle i,j \rangle \in E} f(i,j) - \sum_{\langle j,i \rangle \in E} f(j,i) \right\} \\&= \sum_{i \in A} \sum_{\langle i,j \rangle \in E} f(i,j) - \sum_{i \in A} \sum_{\langle j,i \rangle \in E} f(j,i)\end{aligned}$$

引理7.1的证明 (续2)

$$v(f) =$$

$$\begin{aligned} & \sum_{i \in A, j \in A} \sum_{\langle i, j \rangle \in E} f(i, j) + \sum_{i \in A, j \in \bar{A}} \sum_{\langle i, j \rangle \in E} f(i, j) - \sum_{i \in A, j \in A} \sum_{\langle j, i \rangle \in E} f(j, i) \\ & - \sum_{i \in A, j \in \bar{A}} \sum_{\langle j, i \rangle \in E} f(j, i) \end{aligned}$$

$$v(f) = \sum_{\langle i, j \rangle \in (A, \bar{A})} f(i, j) - \sum_{\langle j, i \rangle \in (\bar{A}, A)} f(j, i)$$



可行流流量与割集容量的关系

引理7.2 设容量网络 $N = \langle V, E, c, s, t \rangle$,

f 是 N 上的任一可行流, (A, \bar{A}) 是任一割集, 则

$$v(f) \leq c(A, \bar{A})$$

引理7.3 设容量网络 $N = \langle V, E, c, s, t \rangle$,

f 是 N 上的任一可行流, (A, \bar{A}) 是任一割集,

如果 $v(f) = c(A, \bar{A})$, 则

f 是最大流, (A, \bar{A}) 是最小割集

最大流最小割集定理

引理7.3

$$v(f) = c(A, \bar{A}) \Rightarrow$$

f 是最大流, (A, \bar{A}) 是最小割集

- 容量网络最大流的流量等于最小割集的容量
- 如何证明?
 - 增广链
 - Ford-Fulkerson算法的基础

增广链的相关概念

容量网络 $N = \langle V, E, c, s, t \rangle$, f 是 N 上任一可行流

- **饱和边**: 流量等于容量的边
- **非饱和边**: 流量小于容量的边
- **零流边**: 流量等于0的边
- **非零流边**: 流量大于0的边

增广链的相关概念（续）

- i - j 链： N 中从顶点 i 到 j 的一条边不重复的路径
（不考虑边的方向情况下的路径）
 - i - j 链的方向是从 i 到 j
 - 前向边：链中与链的方向一致的边
 - 后向边：链中与链的方向相反的边
- i - j 增广链：
 - 一条 i - j 链
 - 前向边都是非饱和边
 - 后向边都是非零流边

通过 s - t 增广链增加流量

- 设 P 是一条关于可行流 f 的 s - t 增广链

$$\delta = \min \left\{ \begin{array}{l} \min \{P \text{ 上前向边容量与流量之差}\} \\ \min \{P \text{ 上后向边流量}\} \end{array} \right\} > 0$$

- 基于可行流 f 构造新的可行流 f'

$$f'(i, j) = \begin{cases} f(i, j) + \delta & \langle i, j \rangle \text{ 是 } P \text{ 的前向边} \\ f(i, j) - \delta & \langle i, j \rangle \text{ 是 } P \text{ 的后向边} \\ f(i, j) & \text{否则} \end{cases}$$

- f' 满足容量限制, 平衡条件, 非负, 且

$$v(f') = v(f) + \delta$$

最大流最小割集定理证明

- 最大流上一定不存在 s - t 增广链
- 是否不存在 s - t 增广链的可行流就是最大流？

证明：

- 假设不存在关于可行流 f 的 s - t 增广链，设 $A = \{j \in V \mid \text{存在关于可行流 } f \text{ 的 } s-j \text{ 增广链}\}$
 $s \in A; \quad t \notin A \Rightarrow t \in \bar{A}$
- (A, \bar{A}) 是割集，且
$$f(i, j) = c(i, j), \langle i, j \rangle \in (A, \bar{A})$$
$$f(j, i) = 0, \langle j, i \rangle \in (\bar{A}, A)$$

最大流最小割集定理证明（续）

- 证明 $f(i, j) = c(i, j)$, $\langle i, j \rangle \in (A, \bar{A})$

否则存在 $\langle i, j \rangle \in (A, \bar{A})$ 使得 $f(i, j) < c(i, j)$

则 $s-i$ 增广链延 $\langle i, j \rangle$ 延伸得到 $s-j$ 链也是增广链

则根据 A 的定义有 $j \in A$, 与 $j \in \bar{A}$ 矛盾

- 证明 $f(j, i) = 0$, $\langle j, i \rangle \in (\bar{A}, A)$

否则存在 $\langle j, i \rangle \in (\bar{A}, A)$ 使得 $f(j, i) > 0$

则 $s-i$ 增广链延 $\langle j, i \rangle$ 延伸得到 $s-j$ 链也是增广链

则根据 A 的定义有 $j \in A$, 与 $j \in \bar{A}$ 矛盾

最大流最小割集定理证明（续）

引理7.1

$$\begin{aligned} v(f) &= \sum_{\langle i,j \rangle \in (A, \bar{A})} f(i,j) - \sum_{\langle j,i \rangle \in (\bar{A}, A)} f(j,i) \\ &= \sum_{\langle i,j \rangle \in (A, \bar{A})} f(i,j) - 0 = \sum_{\langle i,j \rangle \in (A, \bar{A})} c(i,j) = c(A, \bar{A}) \end{aligned}$$

引理7.3

$$v(f) = c(A, \bar{A}) \Rightarrow$$

f 是最大流, (A, \bar{A}) 是最小割集



Ford-Fulkerson最大流算法思想

1. 从给定的初始可行流 f 开始
 - 通常取 0 流，即每条边上流量均为 0 的可行流
2. 每次找一条关于当前可行流的 $s-t$ 增广链
3. 用增广链扩展当前可行流得到新的可行流
4. 重复步骤2和3
 - 直到不存在关于当前可行流的 $s-t$ 增广链为止

标号法求 s - t 增广链

标号方法：对顶点 j 标号 (l_j, δ_j)

初始所有结点均未标号，从源点 s 开始标号 $(\Delta, +\infty)$

找已标号未处理顶点 i ，对关联且未标号顶点 j

- $\langle i, j \rangle$ 非饱和边，对顶点 j 标号为

$$(l_j, \delta_j) = (i, \min \{\delta_i, c(i, j) - f(i, j)\})$$

- $\langle j, i \rangle$ 是非零流边，对顶点 j 标号为

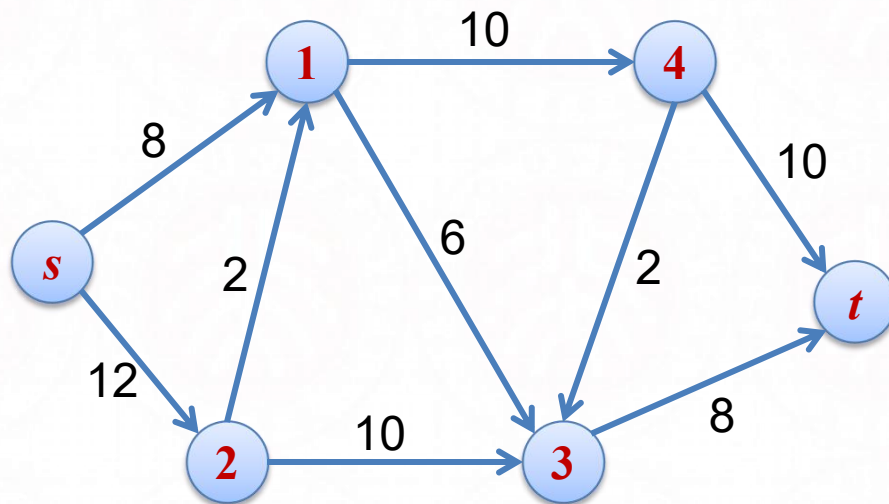
$$(l_j, \delta_j) = (-i, \min \{\delta_i, f(j, i)\})$$

直到汇点 t 被标号，找到 s - t 增广链扩展流

或已无未处理的标号顶点，FF最大流算法结束

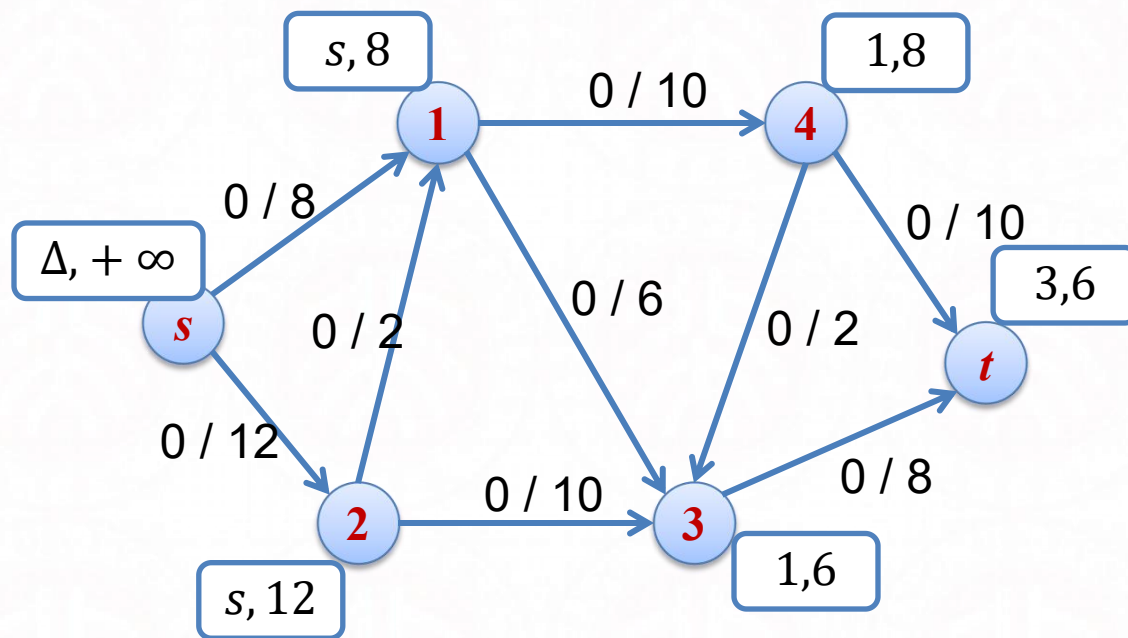
Ford-Fulkerson最大流算法示例

- 容量网络



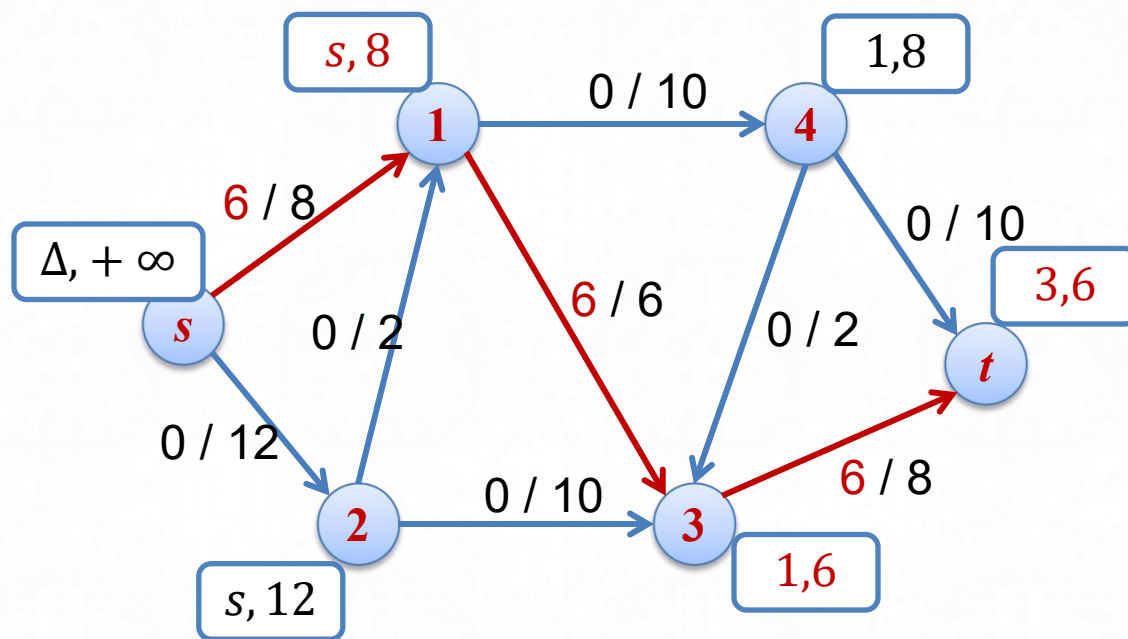
Ford-Fulkerson最大流算法示例

- 点标号: (l, δ) ; 边标号: f/c



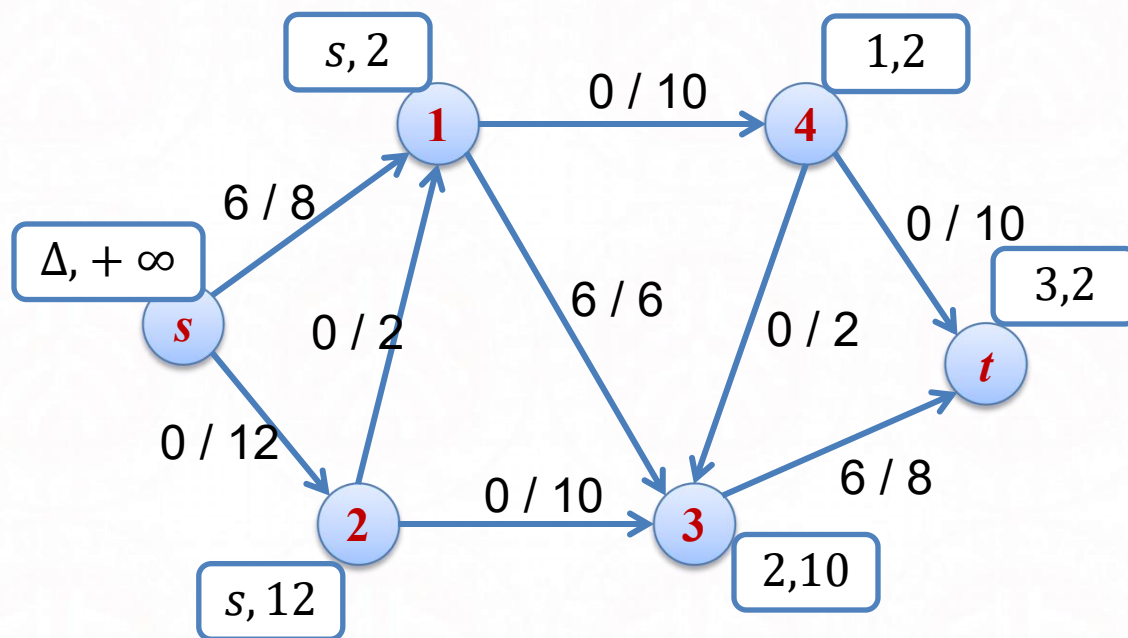
Ford-Fulkerson最大流算法示例

- 点标号: (l, δ) ; 边标号: f/c



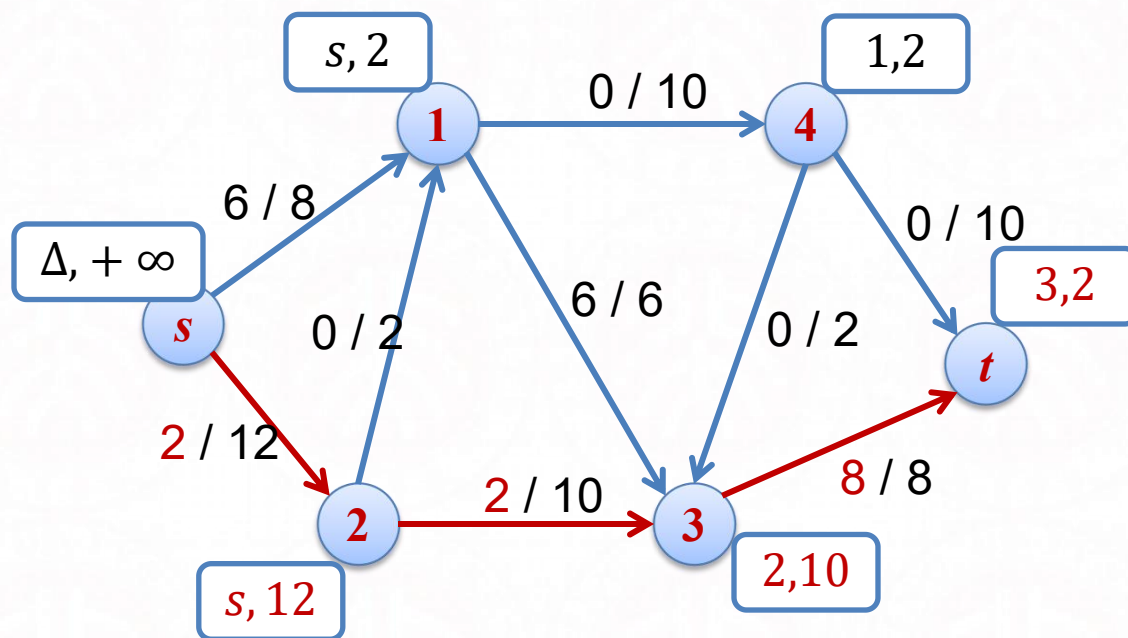
Ford-Fulkerson最大流算法示例

- 点标号: (l, δ) ; 边标号: f/c



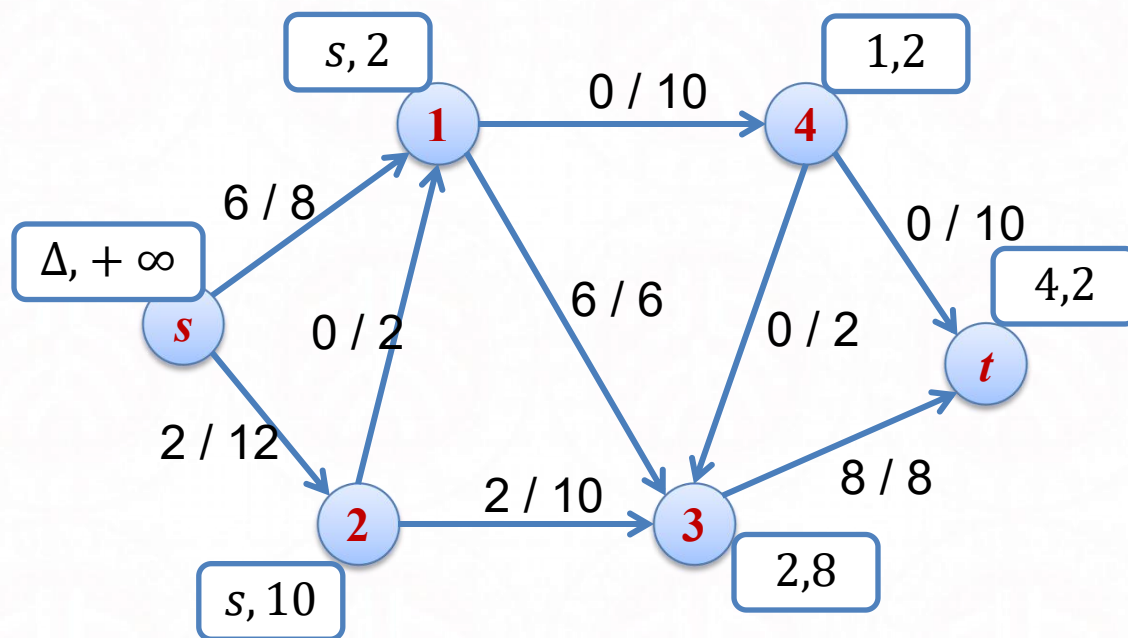
Ford-Fulkerson最大流算法示例

- 点标号: (l, δ) ; 边标号: f/c



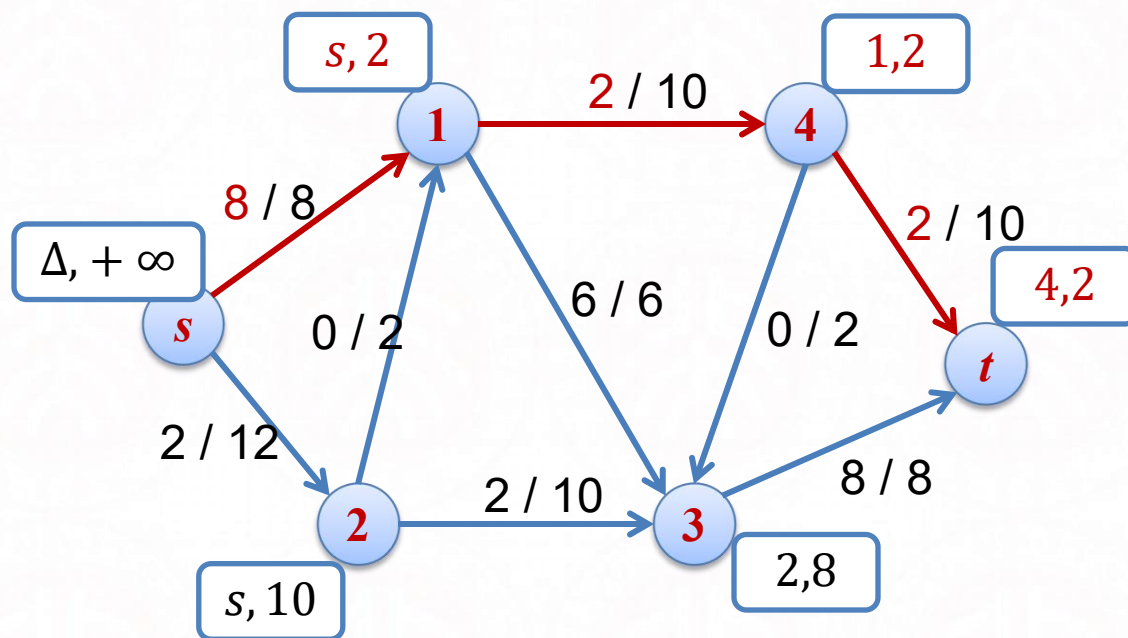
Ford-Fulkerson最大流算法示例

- 点标号: (l, δ) ; 边标号: f/c



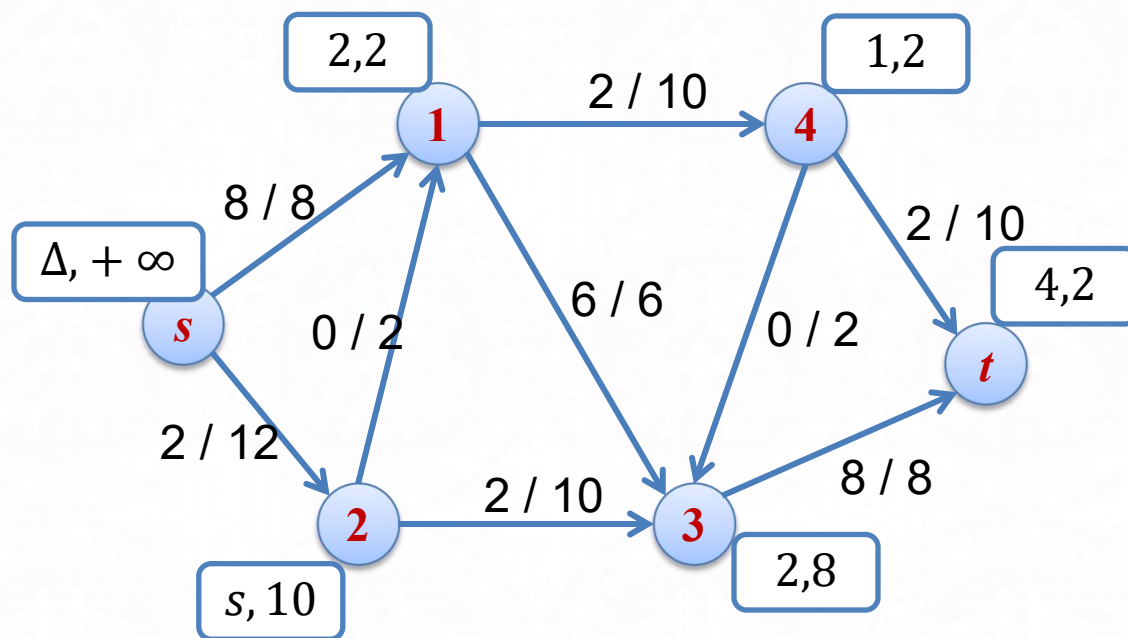
Ford-Fulkerson最大流算法示例

- 点标号: (l, δ) ; 边标号: f/c



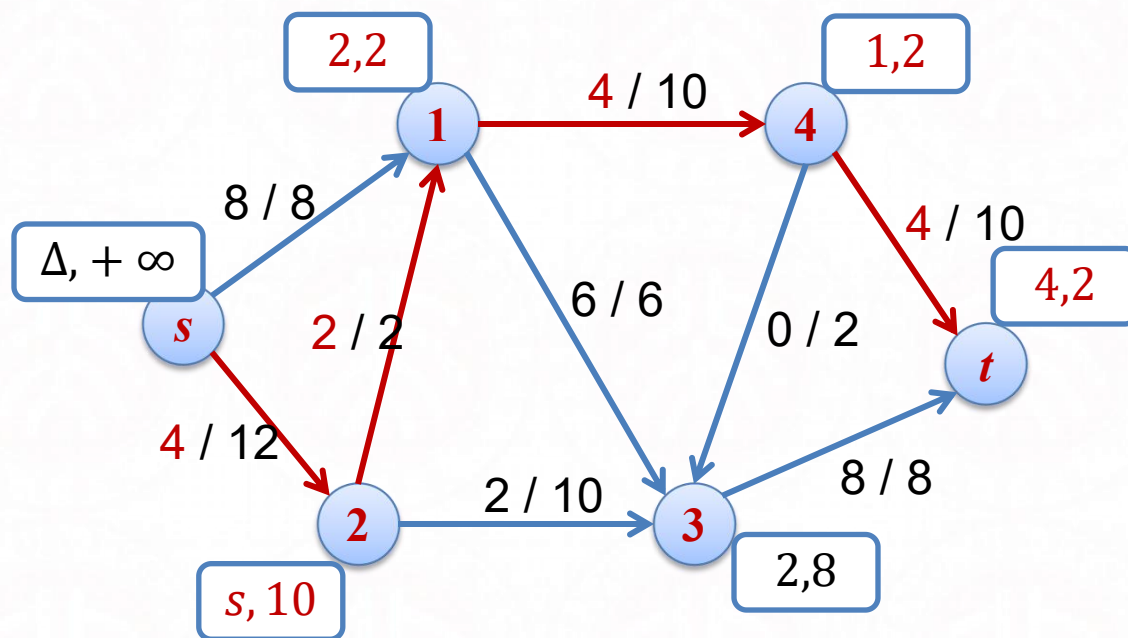
Ford-Fulkerson最大流算法示例

- 点标号: (l, δ) ; 边标号: f/c



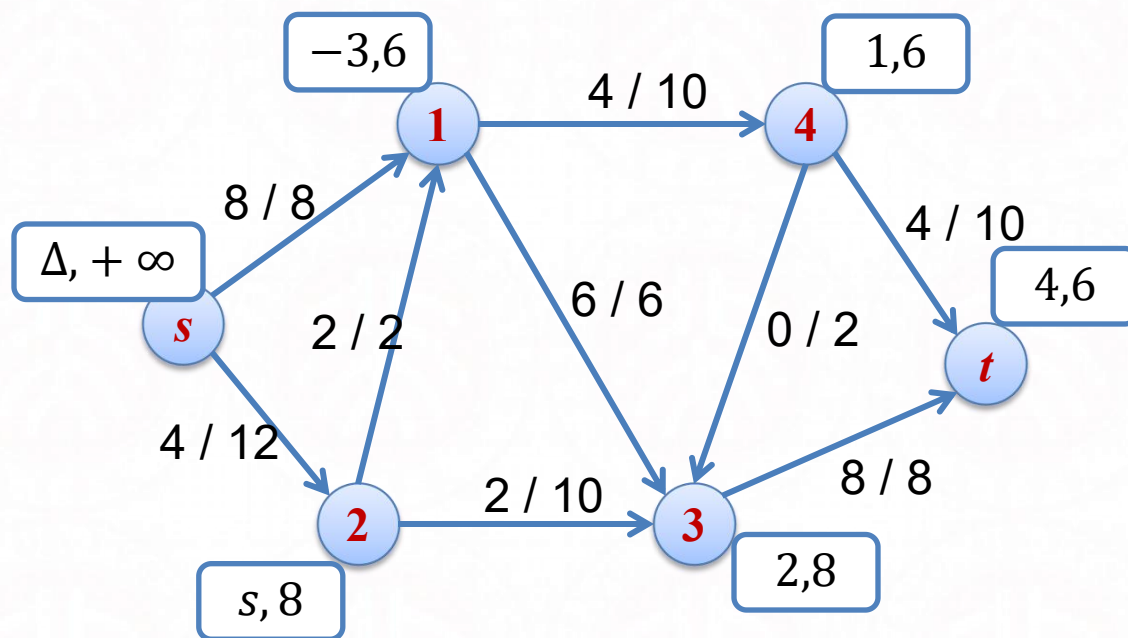
Ford-Fulkerson最大流算法示例

- 点标号: (l, δ) ; 边标号: f/c



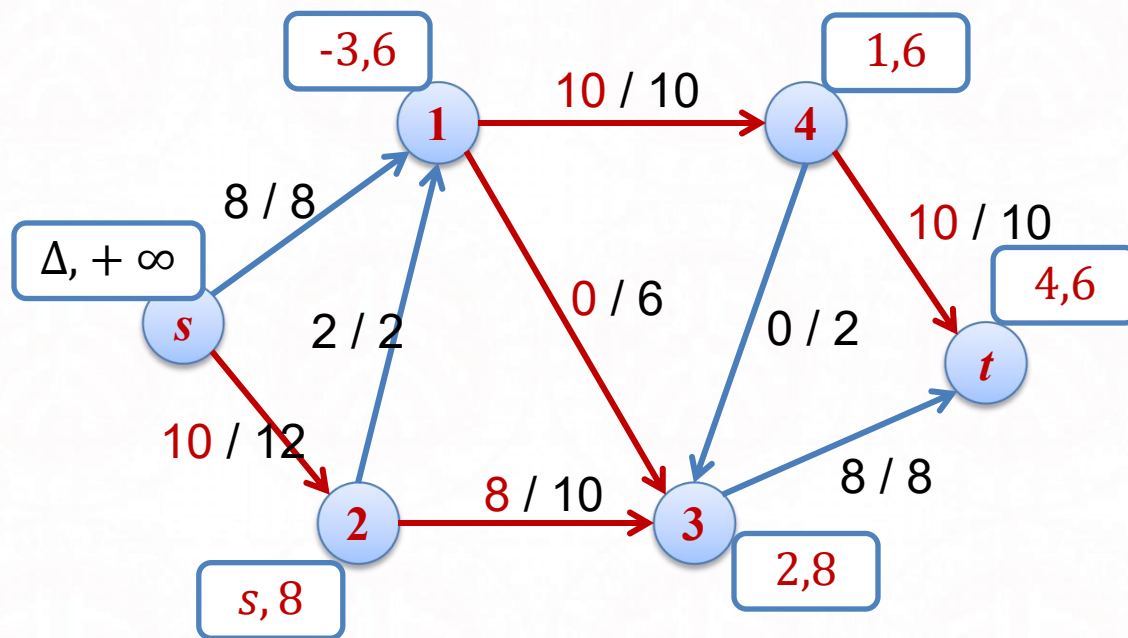
Ford-Fulkerson最大流算法示例

- 点标号: (l, δ) ; 边标号: f/c

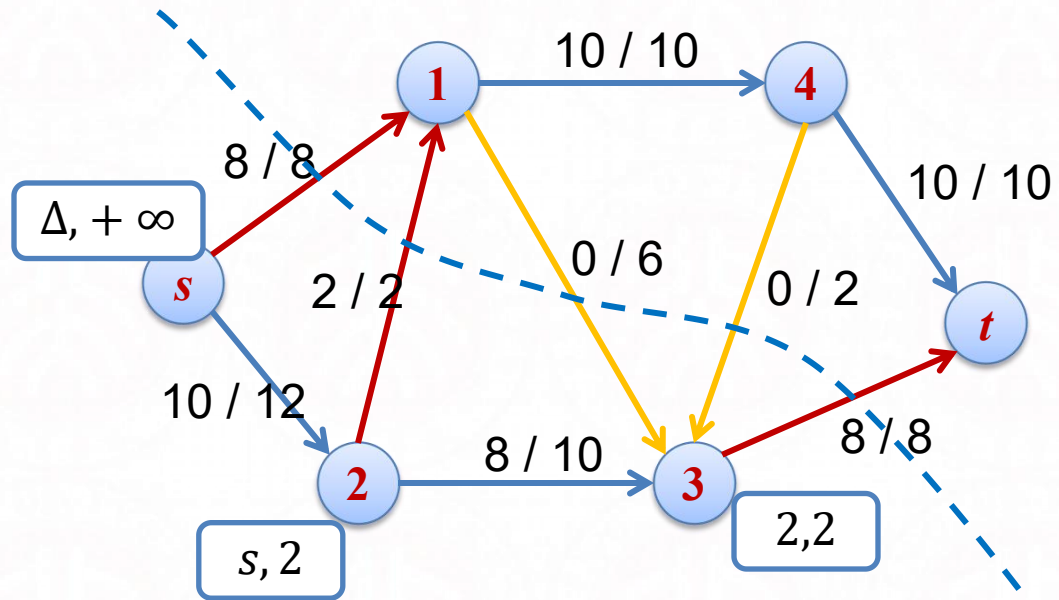


Ford-Fulkerson最大流算法示例

- 点标号: (l, δ) ; 边标号: f/c



Ford-Fulkerson最大流算法示例



Ford-Fulkerson最大流算法运行时间分析

- 影响FF算法的因素
 - 找 s - t 增广链的平均运行时间
 - 最大流流量 v^* 与增广链平均 δ 值之比
- 算法的有限终止性
 - 如果容量值都是正整数，则 δ 也是正整数
 - 显然 $C = \sum_{\langle i,j \rangle \in E} c(i,j)$ 有限，并且 $v^* \leq C$
 - 上述意义下，算法必在有限步内终止

Baseball Elimination Problem

- Suppose you are a reporter for the *Algorithmic Sporting News*
- Can Boston finish with at least as many wins as every other team in the division?

id	team	wins	to play	New York	Baltimore	Toronto	Boston
0	New York	92	2	-	1	1	0
1	Baltimore	91	3	1	-	1	1
2	Toronto	91	3	1	1	-	1
3	Boston	90	2	0	1	1	-

Baseball Elimination Problem

- Suppose you are a reporter for the *Algorithmic Sporting News*
- Can Boston finish with at least as many wins as every other team in the division?

id	team	wins	to play	New York	Baltimore	Toronto	Boston
0	New York	90	11	-	1	6	4
1	Baltimore	88	6	1	-	1	4
2	Toronto	87	11	6	1	-	4
3	Boston	79	12	4	4	4	-

- Observation. Answer depends not only on how many games already won and left to play, but on **whom** they're against.

Baseball Elimination Problem

- Current standings.
 - ▶ Set of teams S .
 - ▶ Distinguished team $z \in S$.
 - ▶ Team x has won w_x games already.
 - ▶ Teams x and y play each other r_{xy} additional times.
- Baseball elimination problem. Given the current standings, is there any outcome of the remaining games in which team z finishes with the most (or tied for the most) wins?

SIAM REVIEW
Vol. 8, No. 3, July, 1966

POSSIBLE WINNERS IN PARTIALLY COMPLETED TOURNAMENTS*

BENJAMIN L. SCHWARTZ†

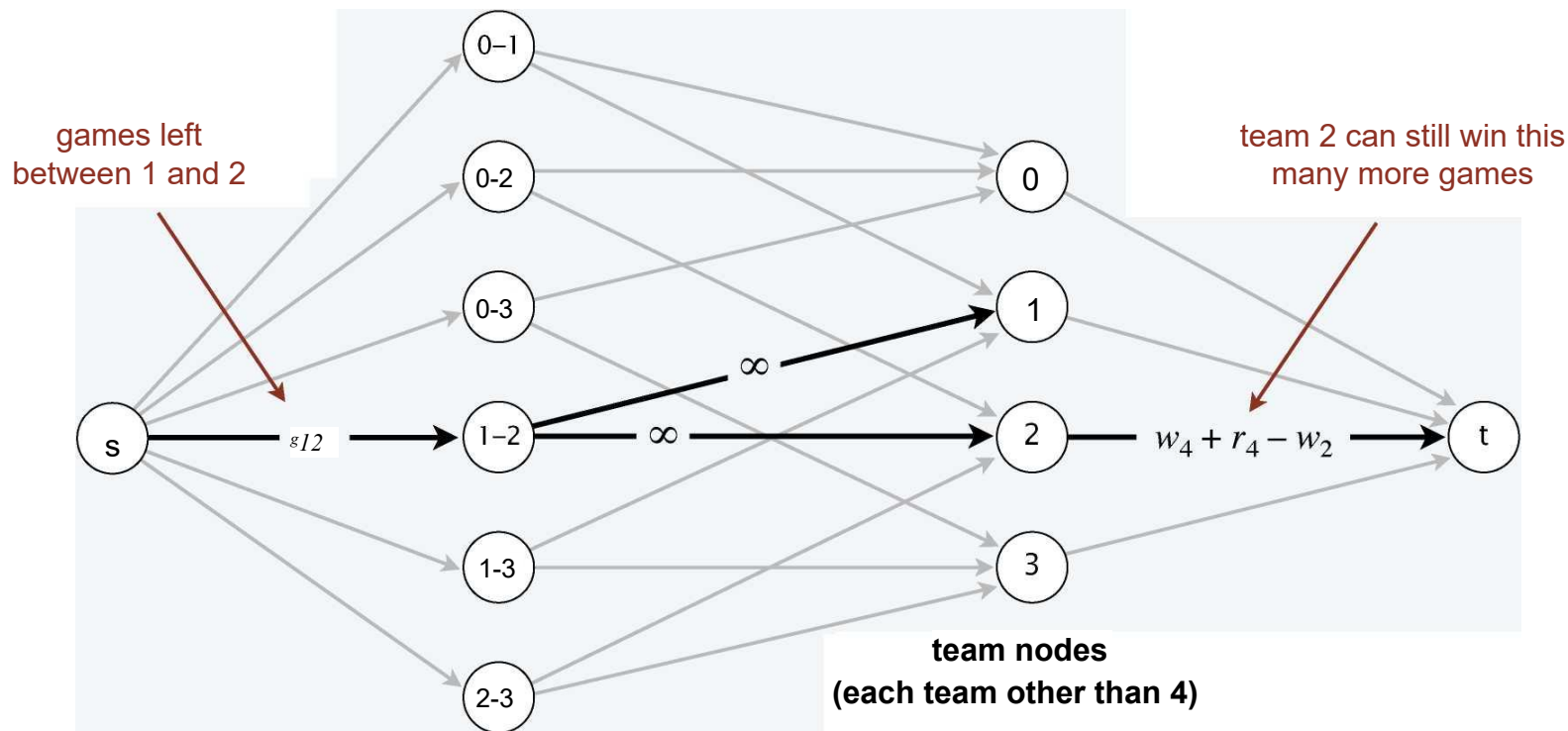
1. Introduction. In this paper, we shall investigate certain questions in tournament scheduling. For definiteness, we shall use the terminology of baseball. We shall be concerned with the categorization of teams into three classes during the closing days of the season. A team may be definitely eliminated from pennant possibility; it may be in contention, or it may have clinched the championship. It will be our convention that a team that can possibly tie for the pennant is considered still in contention. In this paper necessary and sufficient conditions are developed to classify any team properly into the appropriate category.

Baseball Elimination Problem: Explanation

- Suppose that team z has indeed been eliminated.
- Then there exists a certificate of this fact of the following form:
 - ▶ z can finish with at most m wins.
 - ▶ There is a set of teams $T \subseteq S$ so that $\sum_{x \in T} w_x + \sum_{x, y \in T} g_{xy} > m|T|$
 - ▶ (And hence one of the teams in T must end with strictly more than m wins.)

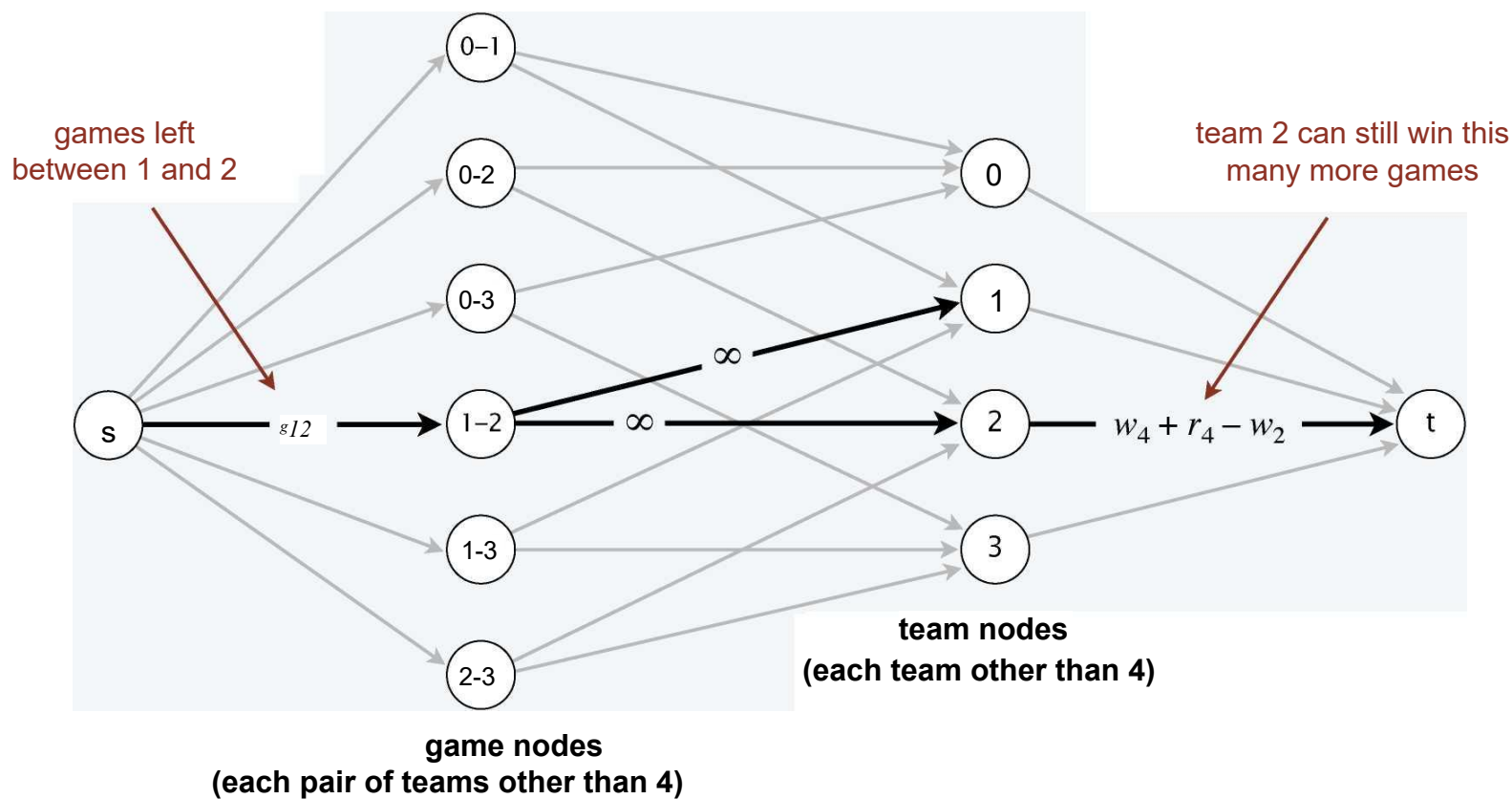
Baseball Elimination Problem: Max-Flow

- Can team 4 finish with most wins?
 - Assume team 4 wins all remaining games $\Rightarrow w_4 + r_4$ wins.
 - Allocate remaining games so that all teams have $\leq w_4 + r_4$ wins.



Baseball Elimination Problem: Max-Flow

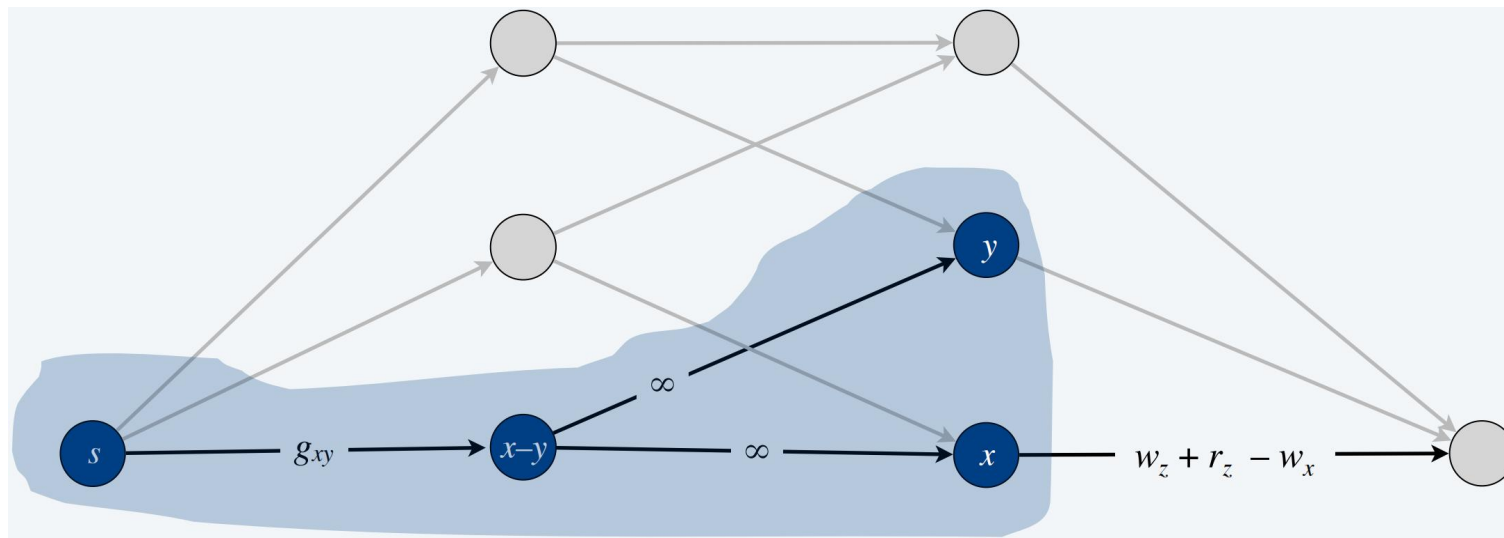
- Theorem. Team 4 not eliminated iff max flow saturates all edges leaving s .



Baseball Elimination Problem: Explanation






- [Hoffman-Rivlin 1967] Team z is eliminated iff there exists a subset T such that

$$\frac{w(T) + g(T)}{|T|} > w_z + g_z$$



Baseball Elimination Problem: Exercise

- Exercise: prove that Detroit was mathematically eliminated

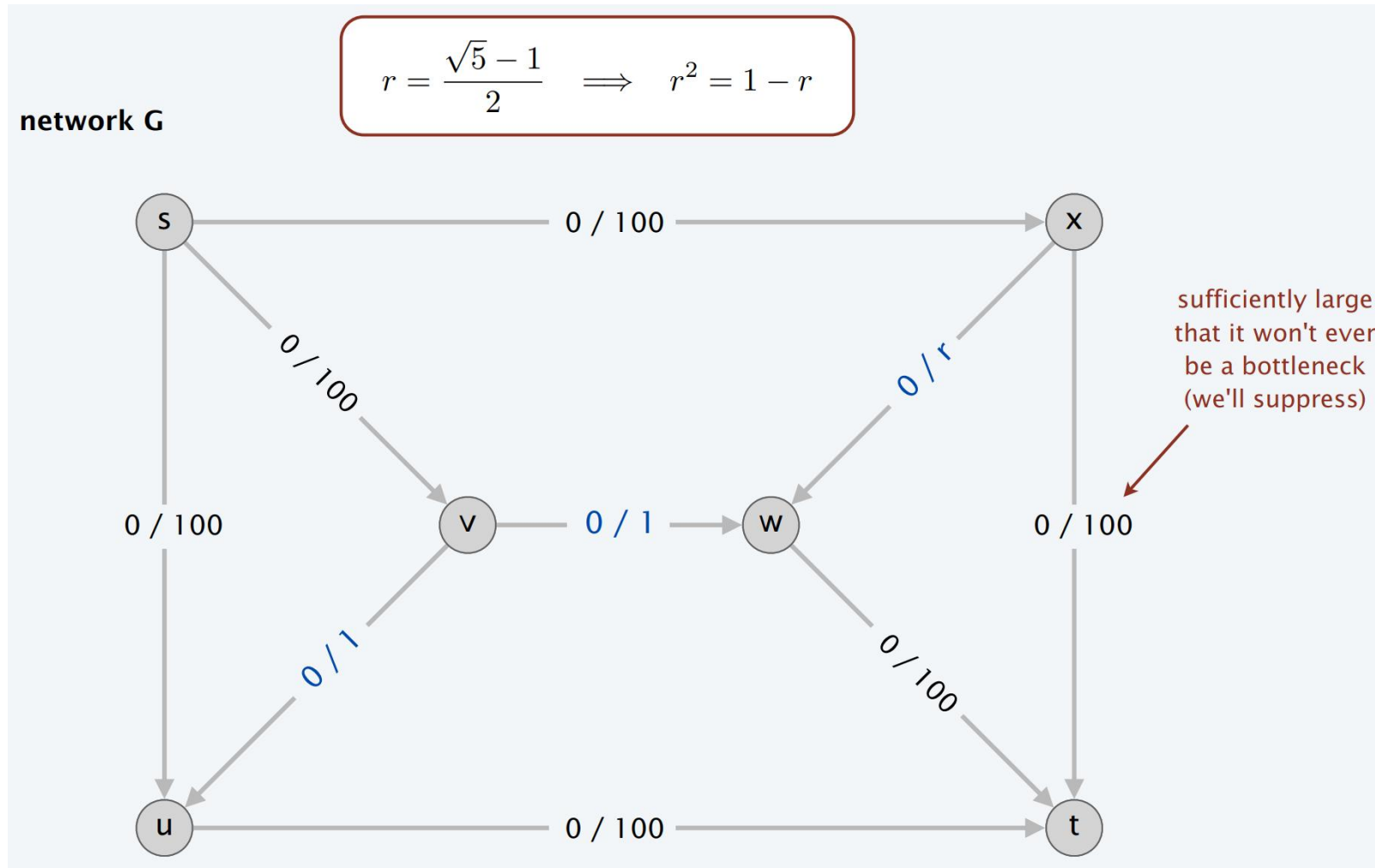
i		team	wins	losses	to play	NYN	BAL	BOS	TOR	DET
0		New York	75	59	28	–	3	8	7	3
1		Baltimore	71	63	28	3	–	2	7	4
2		Boston	69	66	27	8	2	–	0	0
3		Toronto	63	72	27	7	7	0	–	0
4		Detroit	49	86	27	3	4	0	0	–

AL East (August 30, 1996)

Ford-Fulkerson最大流算法运行时间分析

- 影响FF算法的因素
 - 找 s - t 增广链的平均运行时间
 - 最大流流量 v^* 与增广链平均 δ 值之比
- 算法的有限终止性
 - 如果容量值都是正整数，则 δ 也是正整数
 - 显然 $C = \sum_{(i,j) \in E} c(i,j)$ 有限，并且 $v^* \leq C$
 - 上述意义下，算法必在有限步内终止；时间复杂度 $O(E |f^*|)$
- 存在无法有限步终止的可能
 - 容量值为无理数， δ 可能越来越小并趋向于0

Ford-Fulkerson Pathological Example



Ford-Fulkerson最大流算法的改进途径

- Ford-Fulkerson算法没有规定如何求 $s-t$ 增广链
- 改进的策略
 - 保证所求增广链具有一定性质
 - 如最短: Edmonds-Karp in $O(VE^2)$
 - 并且能一次能求得多个增广链
- 余量网络 + 分层辅助网络
 - 帮助我们实现上述两个改进

余量网络的定义

容量网络 $N = \langle V, E, c, s, t \rangle$, f 是 N 上可行流, 定义关于 f 的余量网络 $N(f) = \langle V, E(f), ac, s, t \rangle$ 如下

$$E^+(f) = \{ \langle i, j \rangle \mid \langle i, j \rangle \in E \wedge f(i, j) < c(i, j) \}$$

$$E^-(f) = \{ \langle j, i \rangle \mid \langle i, j \rangle \in E \wedge f(i, j) > 0 \}$$

$$E(f) = E^+(f) \cup E^-(f)$$

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

ac 称作剩余容量, $N(f)$ 也是容量网络

N 中 s - t 增广链 $\leftrightarrow N(f)$ 中的 s - t 有向路径

余量网络的性质

N 的最大流与 $N(f)$ 的最大流有什么关系？

引理7.4 设 N 的最大流量为 v^* ， f 是 N 上可行流，
则 $N(f)$ 的最大流量为 $v^* - v(f)$ 。

证： N 中的割集 (A, \bar{A}) ，在 $N(f)$ 中 (A, \bar{A}) 也是割集

$$\begin{aligned} ac(A, \bar{A}) &= \sum_{\langle i, j \rangle \in (A, \bar{A})} (c(i, j) - f(i, j)) + \sum_{\langle j, i \rangle \in (\bar{A}, A)} f(j, i) \\ &= \sum_{\langle i, j \rangle \in (A, \bar{A})} c(i, j) - \left\{ \sum_{\langle i, j \rangle \in (A, \bar{A})} f(i, j) - \sum_{\langle j, i \rangle \in (\bar{A}, A)} f(j, i) \right\} \end{aligned}$$

$= c(A, \bar{A}) - v(f) \Rightarrow v^* - v(f)$ 是 $N(f)$ 的最大流量 ■

N 的可行流 f 与 $N(f)$ 的可行流叠加

设 f 是 N 上的一个可行流， g 是 $N(f)$ 上的一个可行流，定义 $f' = f + g$ 如下：

$$f'(i, j) = f(i, j) + g(i, j) - g(j, i), \quad \forall \langle i, j \rangle \in E$$

其中规定，当 $\langle i, j \rangle \notin E(f)$ 时， $g(i, j) = 0$

引理7.5 $f + g$ 是 N 上的可行流，且

$$v(f + g) = v(f) + v(g)$$

证明步骤：

容量限制、平衡条件、流量相加

N的可行流f与N(f)的可行流叠加：容量限制

$$\forall \langle i, j \rangle \in E$$

$$0 \leq g(i, j) \leq c(i, j) - f(i, j)$$

$$0 \leq g(j, i) \leq f(i, j)$$

相减

$$-f(i, j) \leq g(i, j) - g(j, i) \leq c(i, j) - f(i, j)$$

同时加 $f(i, j)$

$$0 \leq f(i, j) + g(i, j) - g(j, i) \leq c(i, j)$$

$$0 \leq f'(i, j) \leq c(i, j)$$

满足非负函数与容量限制

N的可行流f与N(f)的可行流叠加：平衡条件

$$\forall i \in E - \{s, t\}$$

$$\sum_{\langle j, i \rangle \in E} f'(j, i) - \sum_{\langle i, j \rangle \in E} f'(i, j)$$

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

$$= \sum_{\langle j, i \rangle \in E} (g(j, i) - g(i, j)) - \sum_{\langle i, j \rangle \in E} (g(i, j) - g(j, i))$$

N 的可行流 f 与 $N(f)$ 的可行流叠加：平衡条件（续）

$$\begin{aligned} &= \sum_{\langle j,i \rangle \in E(f) \wedge \langle j,i \rangle \in E} g(j,i) - \sum_{\langle i,j \rangle \in E(f) \wedge \langle j,i \rangle \in E} g(i,j) \\ &\quad - \sum_{\langle i,j \rangle \in E(f) \wedge \langle i,j \rangle \in E} g(i,j) + \sum_{\langle j,i \rangle \in E(f) \wedge \langle i,j \rangle \in E} g(j,i) \\ &= \sum_{\langle j,i \rangle \in E(f)} g(j,i) - \sum_{\langle i,j \rangle \in E(f)} g(i,j) \\ &= 0 \end{aligned}$$

也满足平衡条件，故 $f' = f + g$ 是 N 上的可行流。

N的可行流f与N(f)的可行流叠加：流量相加

$$\begin{aligned}v(f') &= \sum_{\langle s,j \rangle \in E} f'(s,j) - \sum_{\langle j,s \rangle \in E} f'(j,s) \\&= \sum_{\langle s,j \rangle \in E} (f(s,j) + g(s,j) - g(j,s)) - \sum_{\langle j,s \rangle \in E} (f(j,s) + g(j,s) - g(s,j)) \\&= v(f) + \sum_{\langle s,j \rangle \in E} (g(s,j) - g(j,s)) - \sum_{\langle j,s \rangle \in E} (g(j,s) - g(s,j))\end{aligned}$$

N的可行流f与N(f)的可行流叠加：流量相加（续）

$$\begin{aligned} &= v(f) + \sum_{\langle s,j \rangle \in E(f) \wedge \langle s,j \rangle \in E} g(s,j) - \sum_{\langle j,s \rangle \in E(f) \wedge \langle s,j \rangle \in E} g(j,s) \\ &\quad - \sum_{\langle j,s \rangle \in E(f) \wedge \langle j,s \rangle \in E} g(j,s) + \sum_{\langle s,j \rangle \in E(f) \wedge \langle j,s \rangle \in E} g(s,j) \\ &= v(f) + \sum_{\langle s,j \rangle \in E(f)} g(s,j) - \sum_{\langle j,s \rangle \in E(f)} g(j,s) \\ &= v(f) + v(g) \end{aligned}$$

如何利用余量网络的性质？

- 设 f 是容量网络 N 上的可行流， N 的最大流量为 v^* ，
则残量网络 $N(f)$ 的最大流量为 $v^* - v(f)$ 。
- 设 f 是容量网络 N 上的可行流， g 是余量网络 $N(f)$ 上的可行流，
则 $f + g$ 是 N 上的可行流，且 $v(f + g) = v(f) + v(g)$ 。

分层辅助网络

- 在 $N(f)$ 中, 设 $d(i)$ 表示 s 到顶点 i 的广度优先搜索的距离, 设 $d = d(t)$ 。
- 定义**分层辅助网络**: $AN(f) = \{V(f), AE(f), ac, s, t\}$

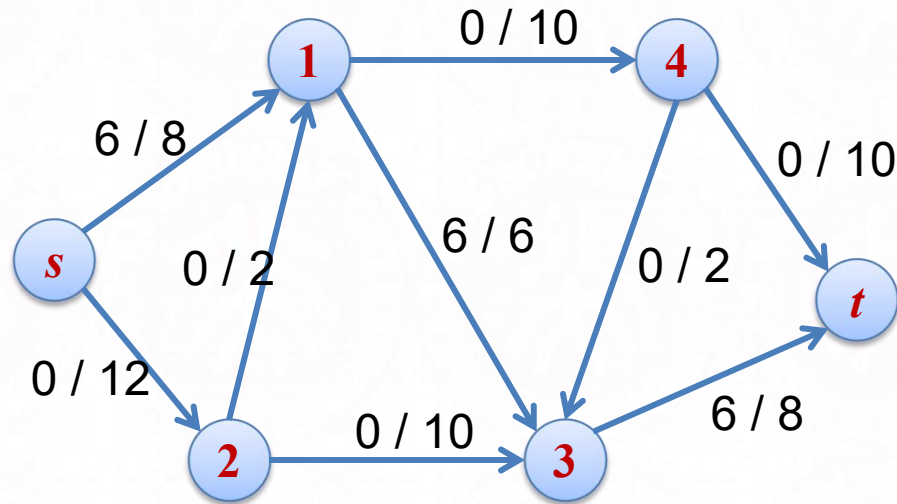
$$V_k(f) = \{i \in V \mid d(i) = k\}, \quad 0 \leq k \leq d - 1$$

$$V_d(f) = \{t\}$$

$$V(f) = \bigcup_{k=0}^d V_k(f)$$

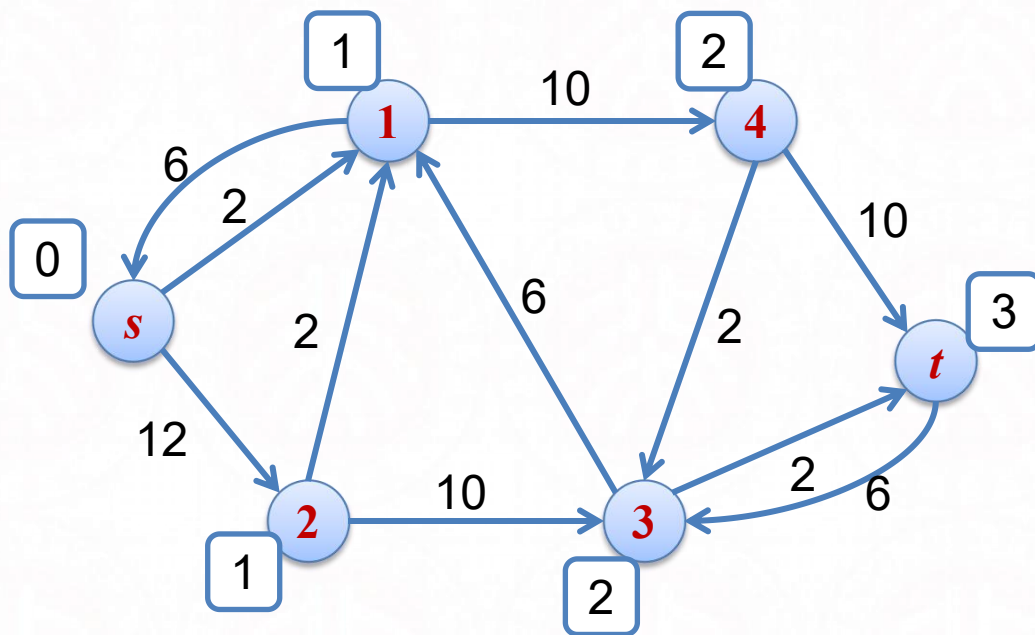
$$AE(f) = \bigcup_{k=0}^{d-1} \{\langle i, j \rangle \mid \langle i, j \rangle \in E(f) \wedge i \in V_k(f), j \in V_{k+1}(f)\}$$

分层辅助网络示例 N 与 f_1

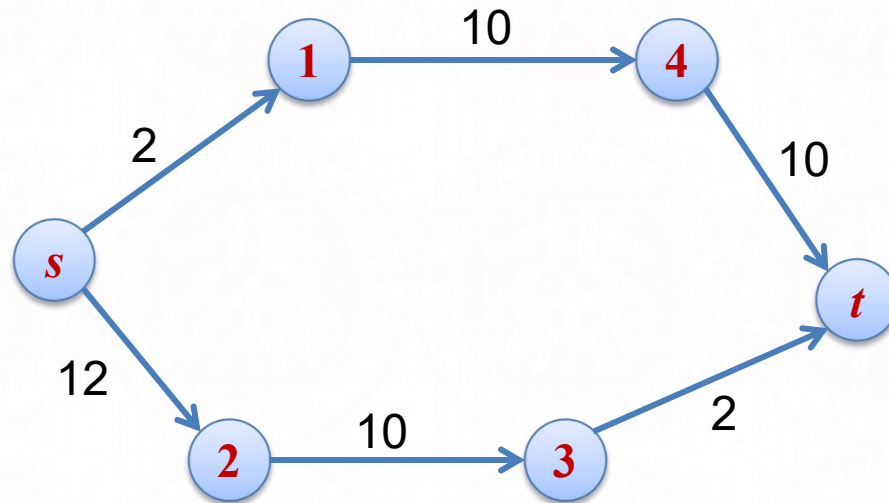


分层辅助网络示例 $N(f_1)$

- N : 宽度优先遍历距离



分层辅助网络示例 $AN(f_1)$

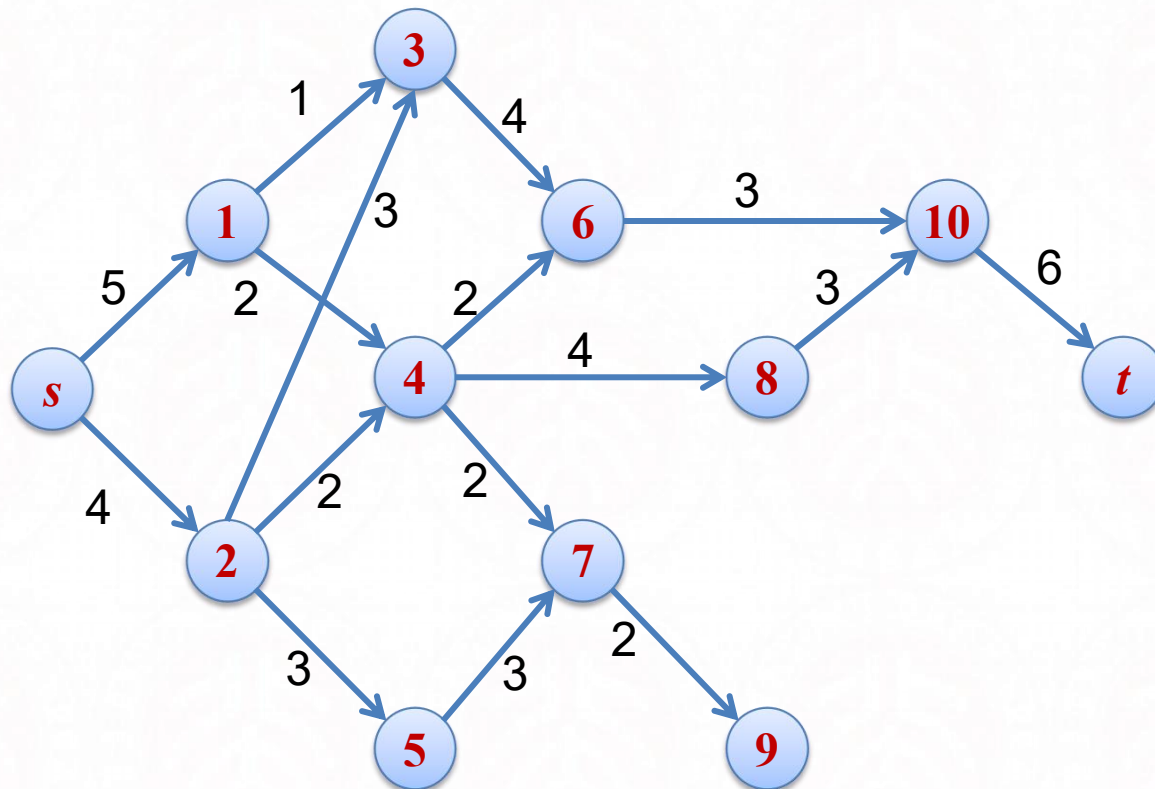


Dinic最大流算法思想

- 通过求 $AN(f)$ 的极大流，得到 $AN(f)$ 中尽可能多 $s-t$ 路径，即 N 中多条增广链
- 什么是极大流？
 - 如果不存在只包含前向边的关于 f 的 $s-t$ 增广链，则 f 是 N 中的极大流
- 怎么求 $AN(f)$ 的极大流？
 - 顶点流通量： $\min\{\text{入边容量和}, \text{出边容量和}\}$
 - 每次用最小流通量更新 $AN(f)$ 及其可行流 g
 - 直到 $AN(f)$ 中不再有 $s-t$ 路径为止

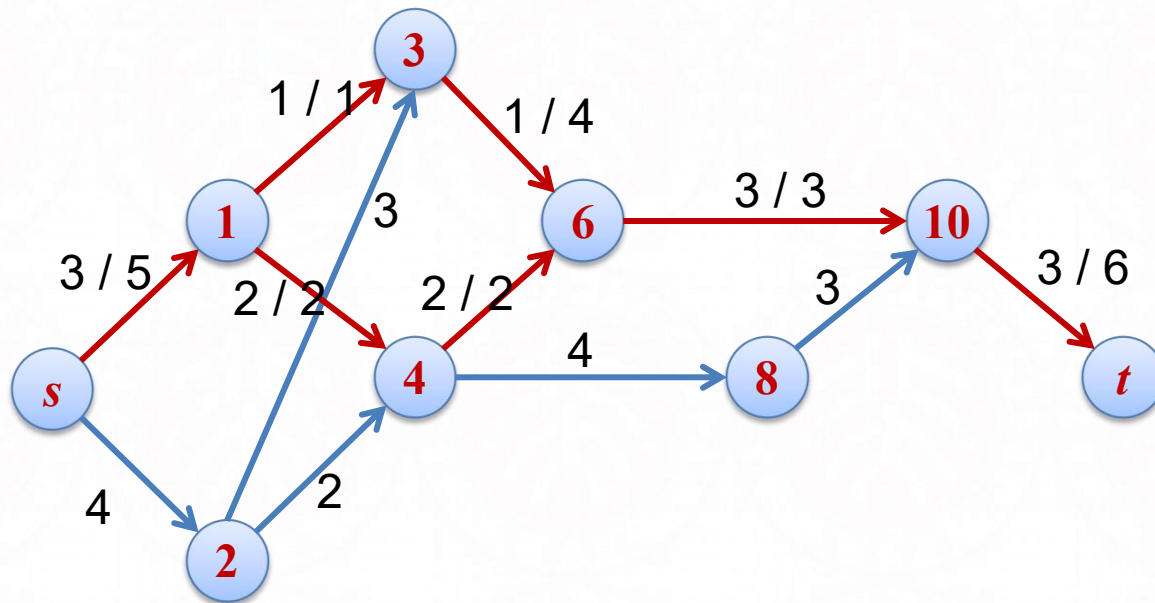
求 $AN(f)$ 的极大流示例

- 边标号: $C \sim$ 容量; $f / C \sim$ 流量 / 容量



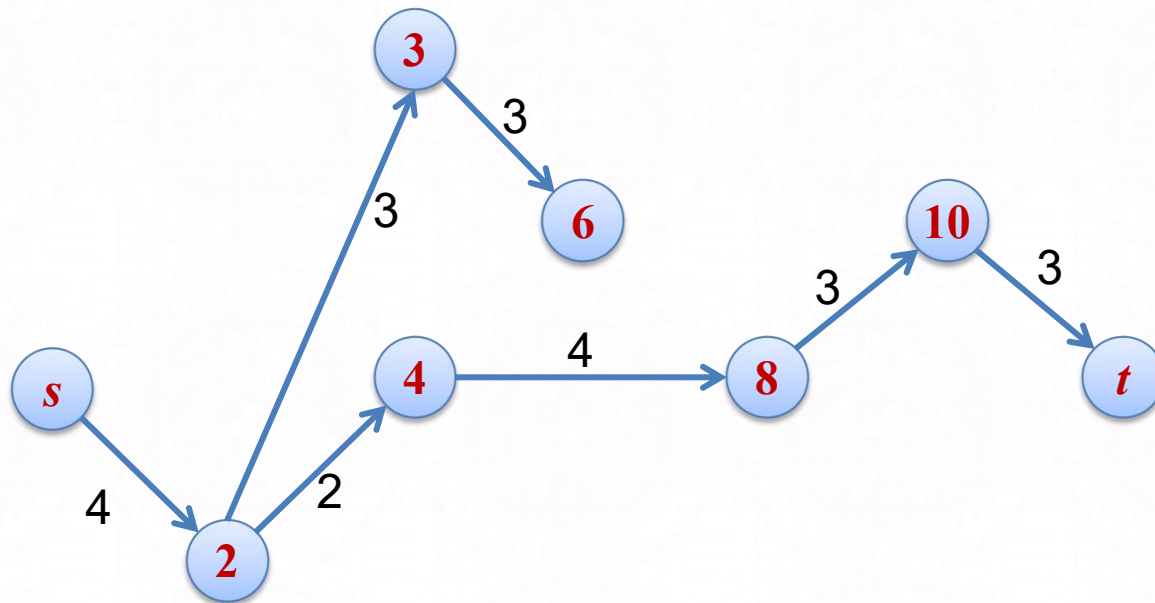
求 $AN(f)$ 的极大流示例

- 边标号: $C \sim$ 容量; $f / C \sim$ 流量 / 容量



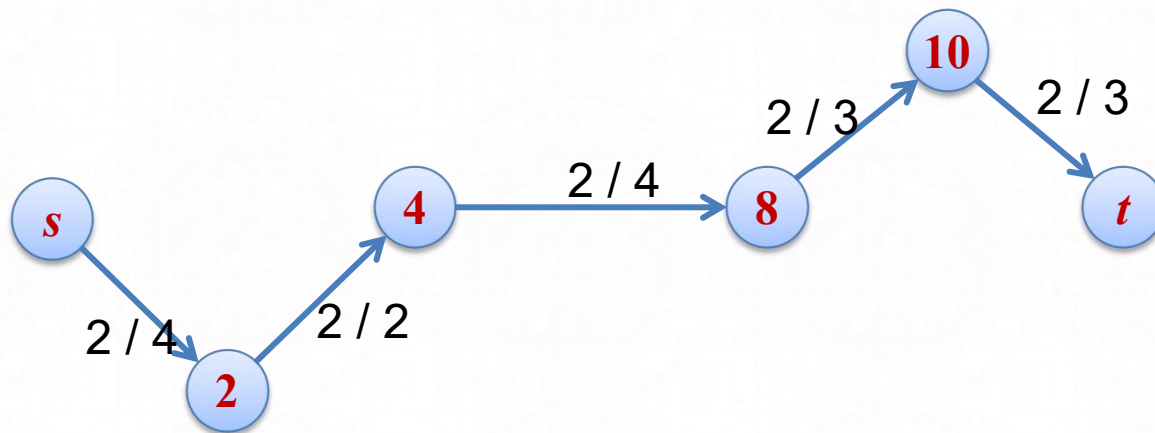
求 $AN(f)$ 的极大流示例

- 边标号: $C \sim$ 容量; $f / C \sim$ 流量 / 容量



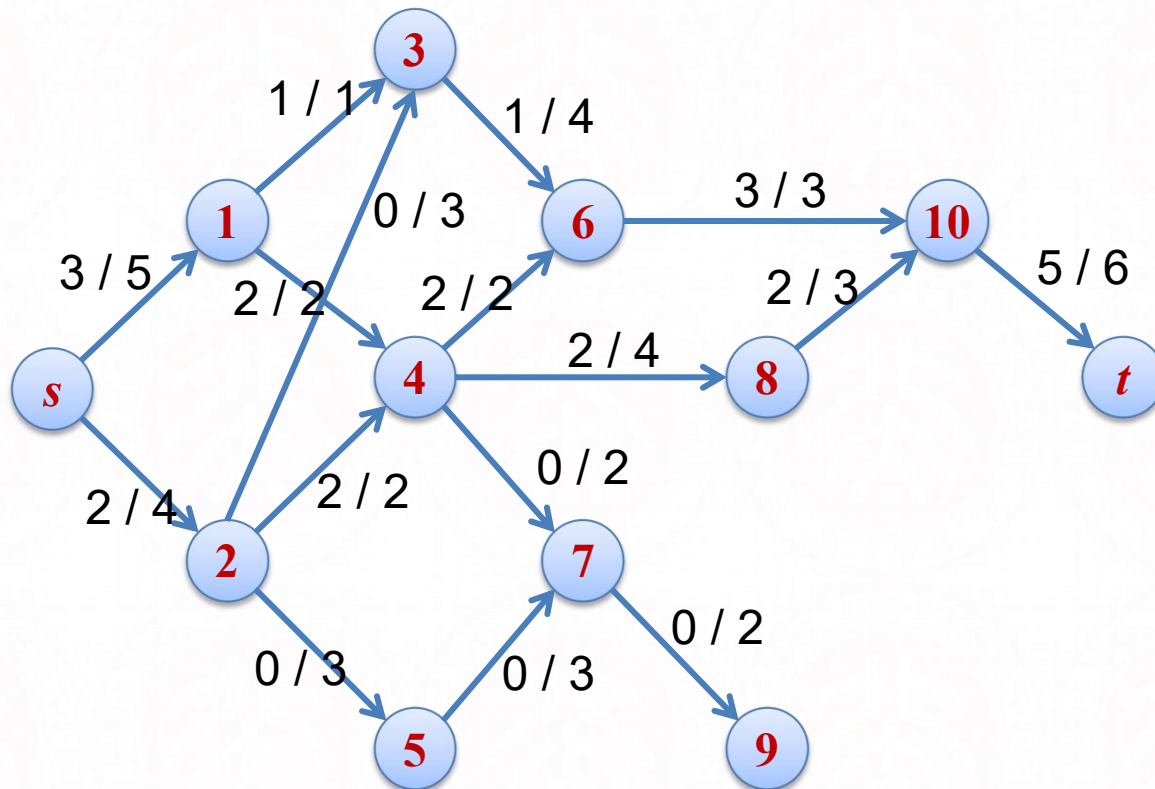
求 $AN(f)$ 的极大流示例

- 边标号: $C \sim$ 容量; $f / C \sim$ 流量 / 容量



求 $AN(f)$ 的极大流示例

- 边标号: $C \sim$ 容量; $f / C \sim$ 流量 / 容量



Dinic最大流算法

1. 从0流 f 开始
2. 构造分层辅助网络 $AN(f)$
3. 如果 $AN(f)$ 存在 $s-t$ 最短路径
 - ① 求得 $AN(f)$ 上的极大流 g
 - ② 叠加流 $f = f + g$; GOTO 2
4. 否则, f 为所求最大流

分层辅助网络层数严格递增

- 考虑 $AN(f)$ 和 $AN(f + g)$ 的 s - t 最短路所包含的边 (严格证明略)

有如下相关性质:

- 同时属于 $AN(f)$ 和 $N(f + g)$ 的边
 - 是关于 g 的非饱和边
- $N(f + g)$ 有但不属于 $AN(f)$ 的边
 - 不可能从 $AN(f)$ 的低 (浅) 层指向高 (深) 层
- 考虑 $AN(f + g)$ 中的 s - t 最短路径
 - 边不可能都是 $AN(f)$ 中的边
 - 否则都关于 g 非饱, 与 g 是极大流矛盾
- 包括非 $AN(f)$ 中的边必然导致 s - t 最短路径的长度大于 $AN(f)$ 的 s - t 距离

Dinic最大流算法正确性与复杂度

定理**7.3** Dinic算法得到的 f 是最大流，且算法在 $O(n^3)$ 步内终止。

证：当算法终止时， $AN(f)$ 中不存在从 s 到 t 的路径，因此 $N(f)$ 中也不存在从 s 到 t 的路径， $N(f)$ 的最大流量为 0 。

根据引理**7.4**， $v^* - v(f) = 0$ ，其中 v^* 是最大流量，推出 $v(f) = v^*$ ，得证 f 是最大流。

由引理**7.7**， $AN(f)$ 中 $s-t$ 距离每个阶段至少增加 1 ，而 $s-t$ 距离不会超过 $n - 1$ ，故至多有 n 个阶段。

Dinic最大流算法正确性与复杂度

在每个阶段：

- 构造 $AN(f)$ 可在 $O(m)$ 步内完成
- 计算顶点的流通量需要 $O(m)$ 步
- 生成 g 的过程中
 - 找一个流通量最小的顶点需 $O(n)$ 步，至多找 n 次，至多需要 $O(n^2)$ 步
 - 对边的操作如果修改容量后删除边（已饱和），这类操作至多有 $O(m)$ 次

Dinic最大流算法正确性与复杂度

▶ 否则对边操作为修改容量不删除边，一个顶点至多有一条这样的边（是非饱和边而未删除），每次推送流量最多涉及 n 条这样的边，最多 n 次推送，共有 $O(n^2)$ 次这类操作

► 每阶段的工作量为

$$O(m) + O(m) + O(n^2) + O(m) + O(n^2) = O(n^2)$$

► 总工作量为 $O(n^3)$



本节内容

➤ 网络流

- ▶ 容量网络、容量、源点、汇点
- ▶ 可行流、流量、最大流
- ▶ 割集、最小割集
- ▶ 最大流最小割集定理

➤ Ford-Fulkerson 算法

- ▶ f 是最大流 \Leftrightarrow 不存在关于 f 的 s - t 增广链
- ▶ 时间复杂度分 $O(E |f^*|)$

➤ 例：Baseball Elimination Problem

➤ Ford-Fulkerson 方法的优化

- ▶ Edmonds-Karp 算法 $O(VE^2)$ (略)
- ▶ Dinic 算法 (优化版) $O(V^3)$
 - 容量网络关于流量 f 的余量网络 $N(f)$
 - 分层辅助网络