暑期课“应用算法”

- 主讲老师Tami Tamir
- 以色列交叉学科中心教授
- 主页：http://www.faculty.idc.ac.il/tami/

课程主要内容

Tentative list of topics:

- Introduction and Review: Graphs, Complexity classes, Game Theory basics
- Scheduling Theory
- Facility Location
- Packing Problems
- Algorithmic Game Theory

Grading: 3 homework assignments: 30%, final exam: 70%

上课时间：7月1日至14日，每周一到每周五上午9-12点，7月14日上午考试，英文授课

2017年暑期第一次开课，选课人数23人，优秀率70%

7.1 最大流算法

网络流及其性质

容量网络 $N=<V,E,c,s,t>$,其中$N=<V,E>$是有向连通图, 记 $n=|V|$, $m=|E|$, $c:E\to R^*$是边的容量, s 和 t 是两个特殊的顶点, s 称作发点(源), t 称作收点(汇), 其余顶点称作中间点.

设 $f:E\to R^*$满足下述条件:

(1) 容量限制 $\forall<i,j>\in E,\ f(i,j)\le c(i,j)$,

(2) 平衡条件 $\forall i\in V-\{s,t\}$,

$$\sum_{<j,i>\in E} f(j,i) = \sum_{<i,j>\in E} f(i,j)$$

称 f 是N上的一个可行流, 称 s 的净流出量 $v(f)$为 f 的流量,即

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

流量最大的可行流称作最大流.

i-j 增广链

定义 设容量网络 $N=<V,E,c,s,t>$, f 是N上的一个可行流.

(1) N中流量等于容量的边称作**饱和边**, 流量小于容量的边称作**非饱和边**.

(2) 流量等于0的边称作**零流边**, 流量大于0的边称作**非零流边**.

(3) 不考虑边的方向, N中从顶点 i 到 j 的一条边不重复的路径称作 ***i-j* 链**. *i-j* 链的方向是从 i 到 j. 链中与链的方向一致的边称作**前向边**, 与链的方向相反的边称作**后向边**.

(4) 如果链中所有前向边都是非饱和的, 所有后向边都是非零流的, 则称这条链为 ***i-j* 增广链**.

Ford-Fulkeson算法

设计思想

从给定初始可行流 (通常取零流) 开始

取当前可行流的 s-t 增广链 P, 修改P上流量得到新的可行流.

重复进行, 直到不存在 s-t 增广链为止.

标号法

从 s 开始给顶点作标号, 直到 t 得到标号为止.

顶点 j 得到标号表示已找到从 s 到 j 的增广链.

标号为 (l_j, δ_j)

δ_j 等于 s-j 链上可增加流量的最大值

$l_j = +i$ 表示链是从 i 到 j 的且$<i, j>$是前向边

$l_j = -i$ 表示链是从 i 到 j 的且$<j, i>$是后向边

顶点状态　已标号已检查的, 已标号未检查的, 未标号的.

辅助网络

定义 设容量网络 $N=<V, E, c, s, t>$, f 是N上的一个可行流. 关于 f 的辅助网络 $N(f) = <V, E(f), ac, s, t>$, 其中

$E^+(f) = \{<i, j> \mid <i, j>\in E$ 且 $f(i, j) < c(i, j) \}$

$E^-(f) = \{<j, i> \mid <i, j>\in E$ 且 $f(i, j) > 0 \}$

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

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

ac 称作辅助容量. $N(f)$也是容量网络.

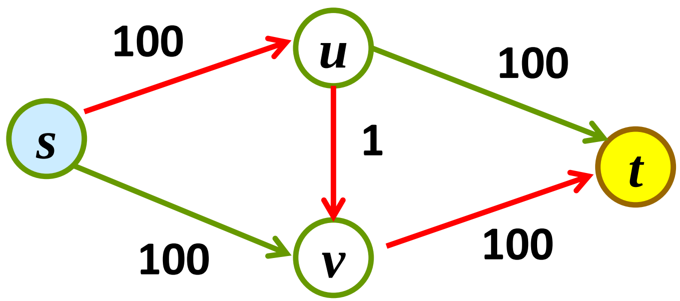
算法实现

算法：

- 给出初始流
- 通过辅助网络选择增广路径以增加流值

问题：在存在多条增广路径情况下，如何选择 g？

一个坏的例子：



可能执行200次增广操作

解决方案：通过分层辅助网络，每次增广都选辅助网络中的极大流

分层辅助网络

分层辅助网络

按广度优先搜索$N(f)$中 s-t 最短路，按到s距离d 对顶点分层

$AN(f) = \langle V(f), AE(f), ac, s, t \rangle$，其中

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

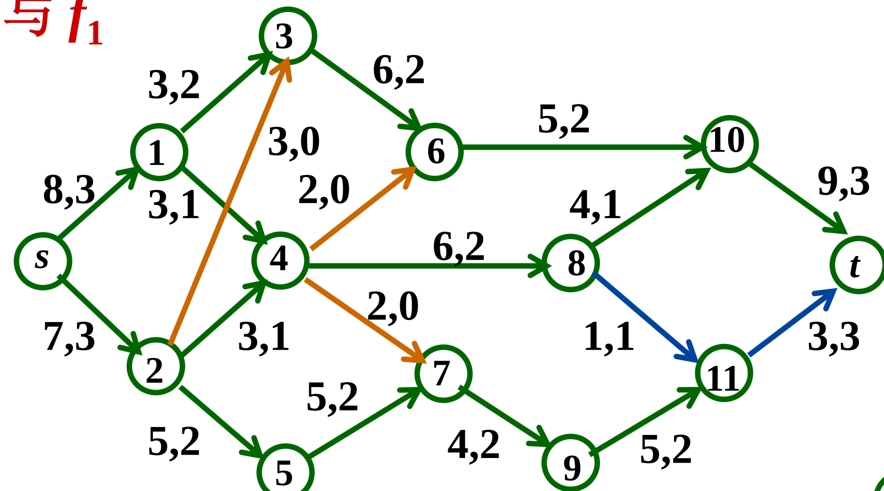
$$AE(f) = \bigcup_{k=0}^{d-1} \{\langle i, j \rangle \mid \langle i, j \rangle \in E(f) \wedge i \in V_k(f), j \in V_{k+1}(f)\}$$

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

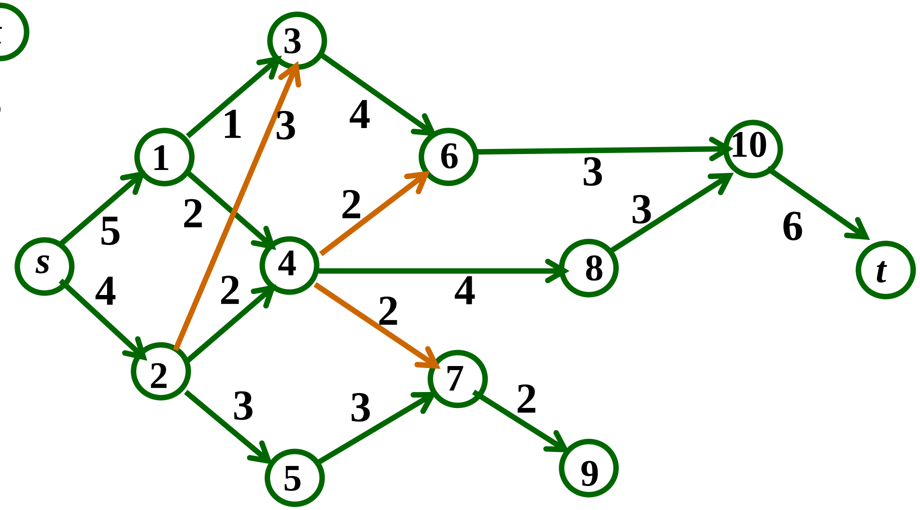
$V_d(f) = \{ t \}$

例2

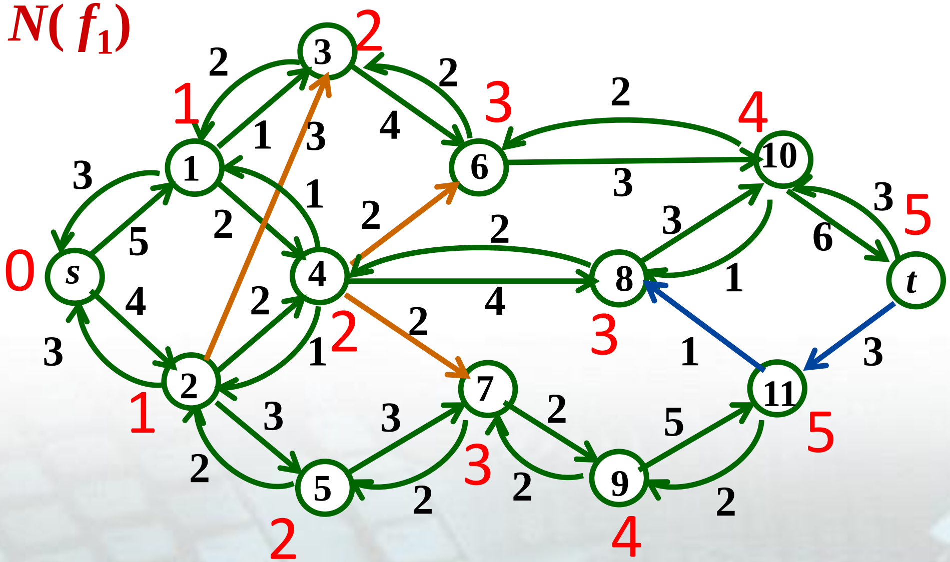
***N*与 f_1**



$AN(f_1)$

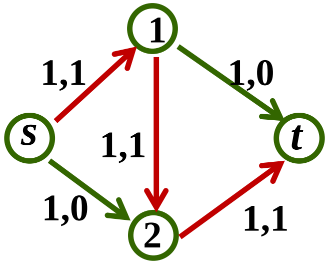


$N(f_1)$



删除$N(f_1)$中s-t最短路径外的顶点和边

Dinic算法



有关概念:

(1) **前向增广链**: 关于 f 的不含后向边的增广链.

(2) **极大流**: 不存在前向增广链的可行流.

最大流必是极大流, 但极大流不一定是最大流.

(3) **中间点 i 的流通量 $th(i)$**: 以 i 为终点的边的容量之和与以 i 为始点的边的容量之和中的小者.

Dinic算法 （关键：求 $AN(f)$中的极大流）

1. 在 $AN(f)$中找尽可能多的从 s 到 t 的最短路径并给出这些路径上的流量.

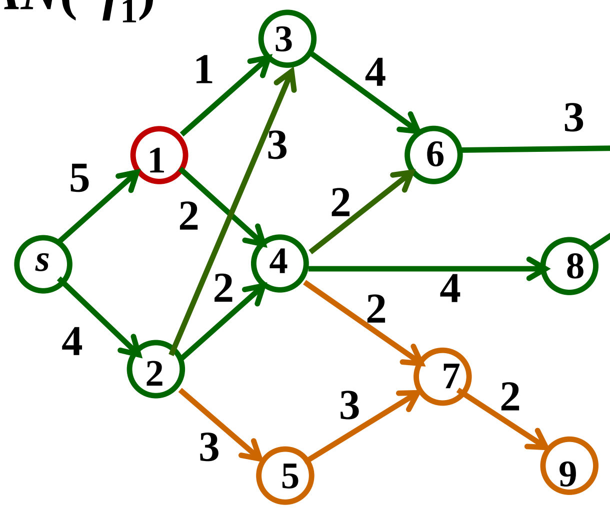
2. 删去流通量为0的顶点及关联的边. 找到流通量最小的顶点 k, 从k开始向后和向前安排流量 $th(k)$, 直到 t 和 s 为止. 修改相关边的容量.

3. 重复进行, 直到 s 和 t 不连通为止. 得到$AN(f)$的极大流 g.

令$f'=f+g$.

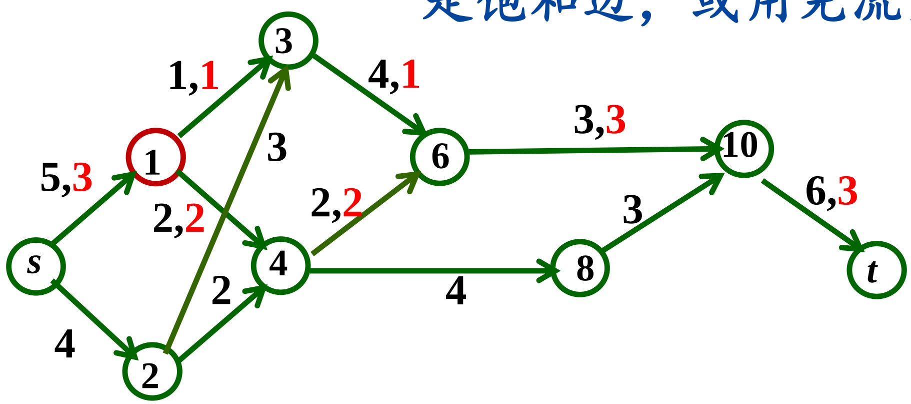
例2(续)

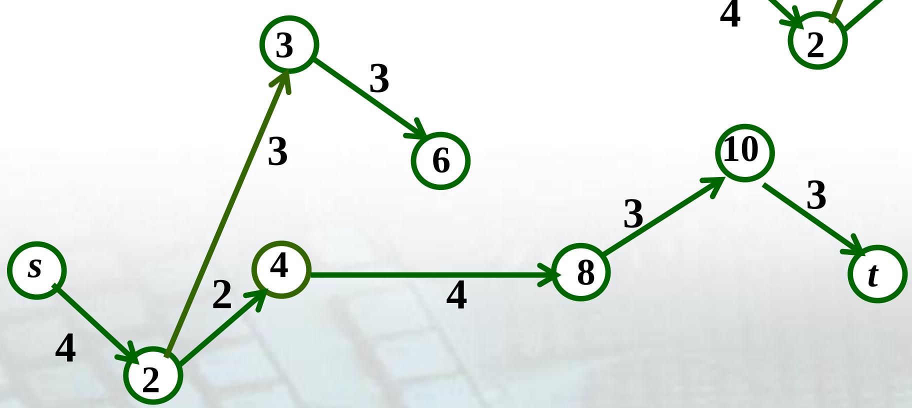
$AN(f_1)$



去掉0流通量顶点及边
找最小流通量顶点1

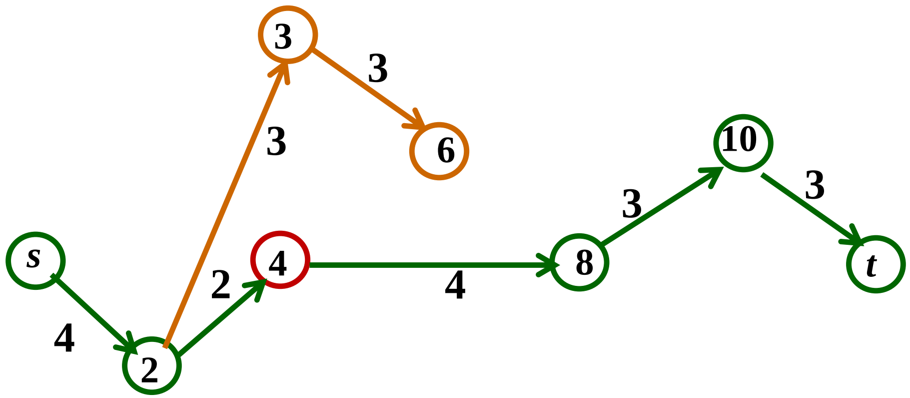
从1向前后推送3单位流
安排原则：得到流的边或是饱和边，或用完流量



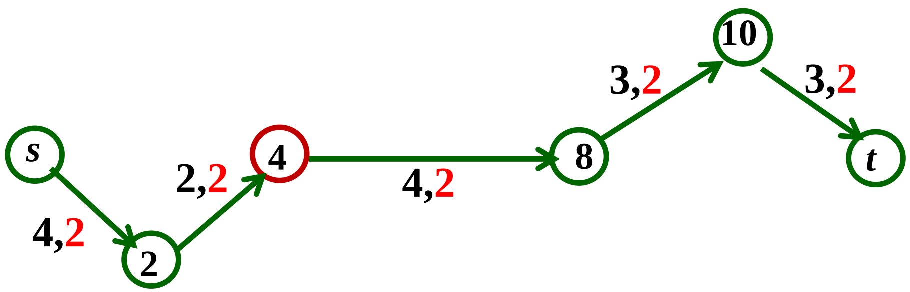


删流通量最小顶点1及关联边
删饱和边
修改剩余容量

例2(续)



去掉0流通量顶点及边
找最小流通量顶点4

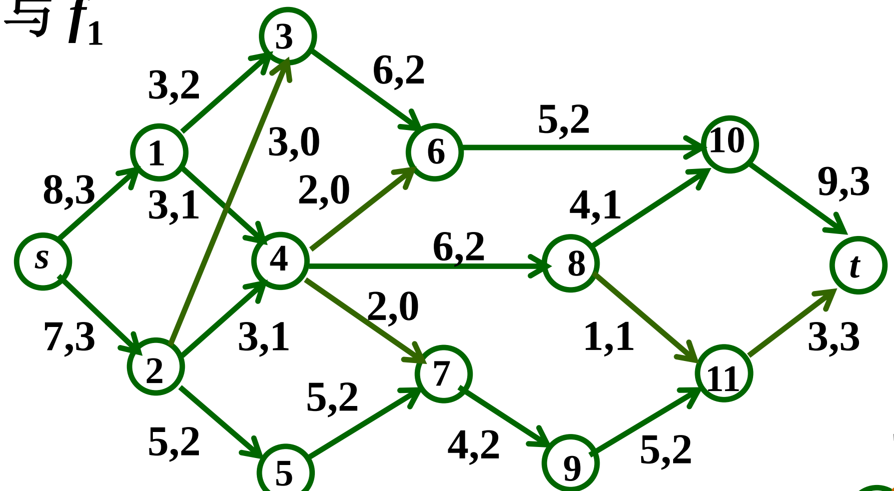


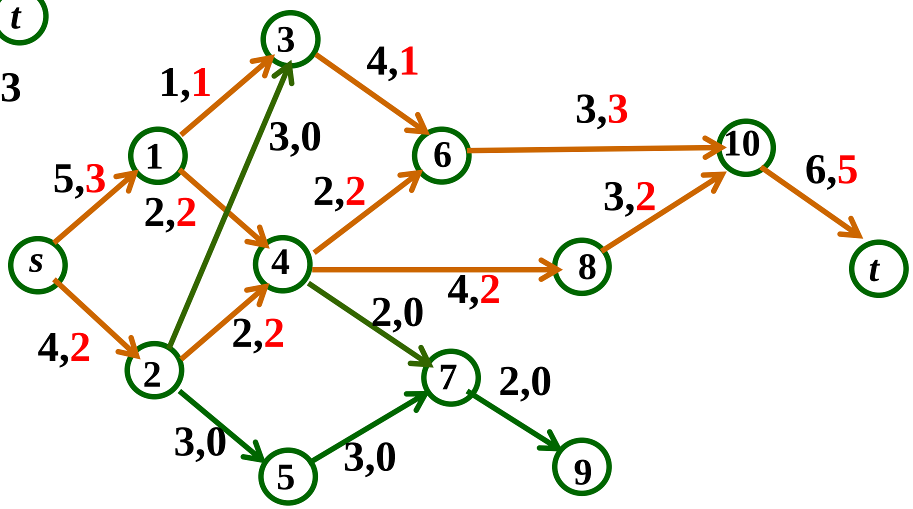
从4向前后推送2单位流

删流通量为0顶点
得到极大流 g

例2(续)

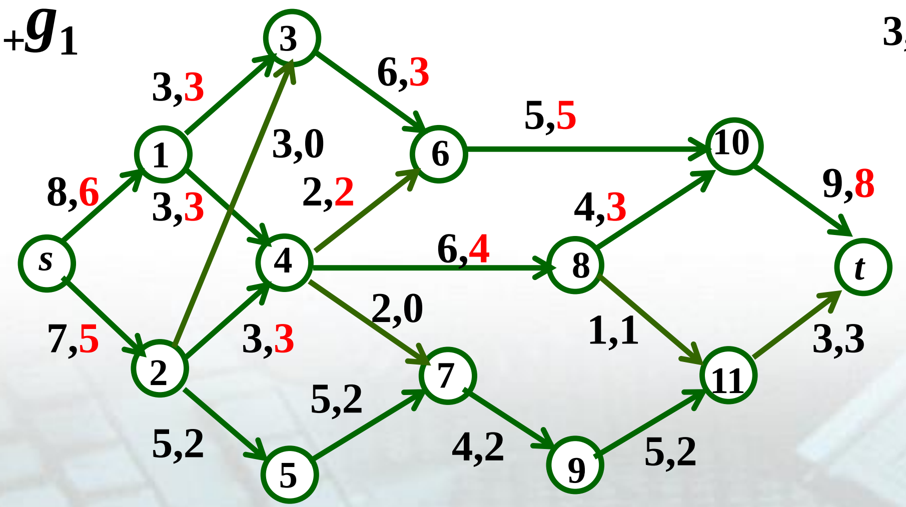
N与 f_1





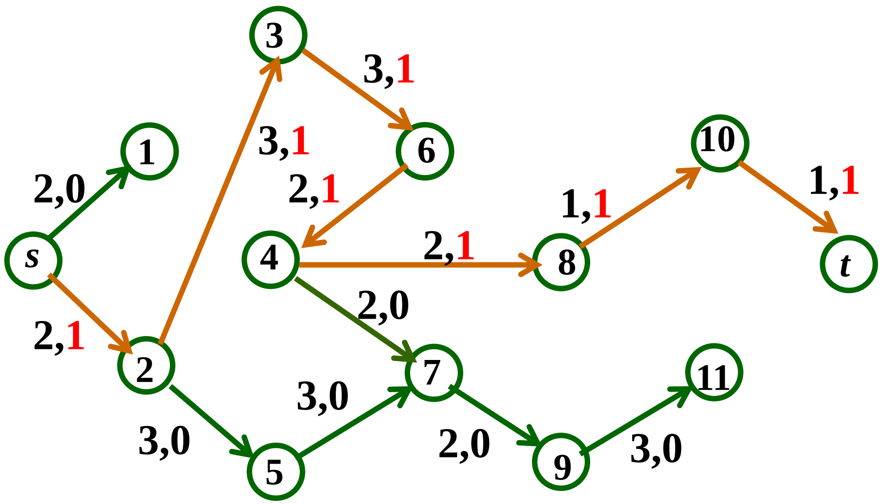
$AN(f_1)$与极大流 g_1

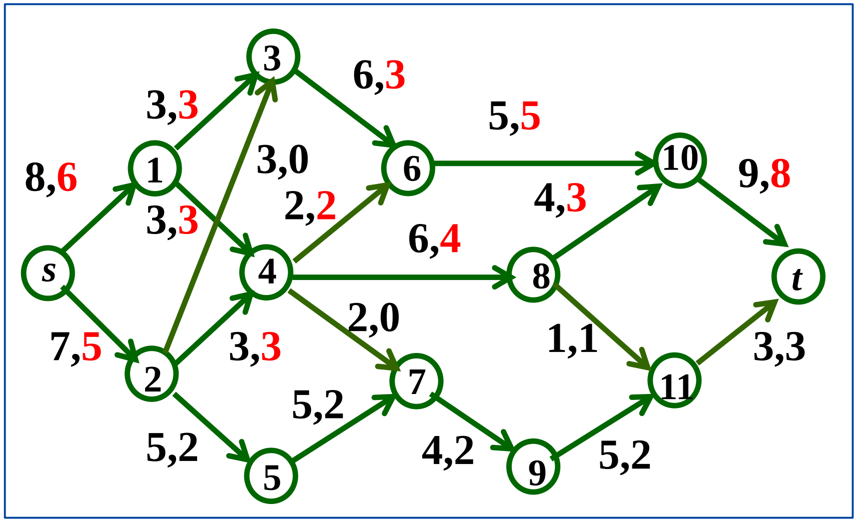
$f_2 = f_1 + g_1$



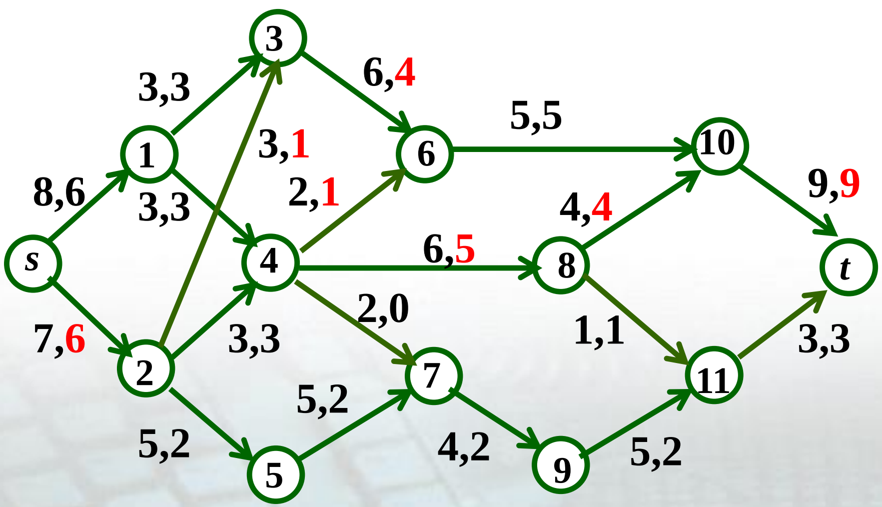
例2(续)

$AN(f_2)$与极大流 g_2





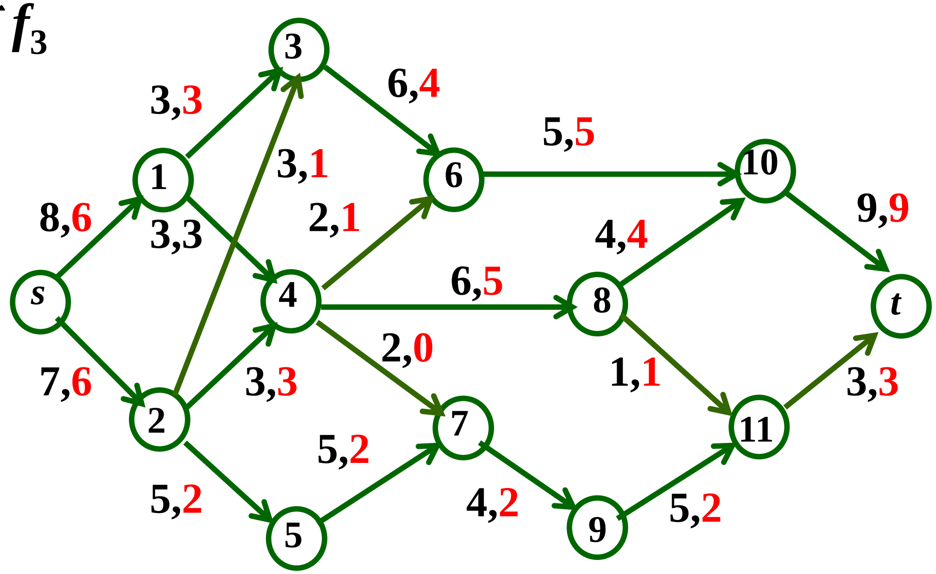
N与 f_2



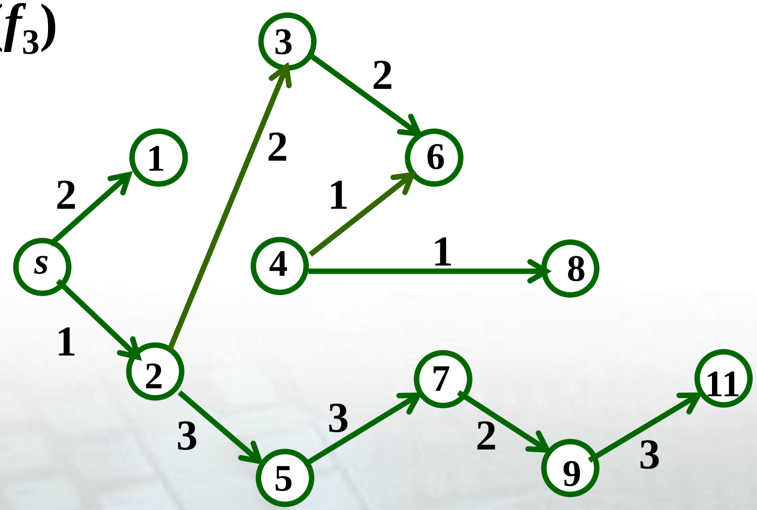
N与f_3

例2(续)

N与f_3



$AN(f_3)$



$AN(f_3)$中没有 s-t 路径

算法结束，输出：

$f=f_3$, $v(f_3)=12$

Dinic算法

Dinic算法

1. $f \leftarrow 0$ //取零流作初始可行流
2. 构造$AN(f)$
3. if $AN(f)$不存在从s到t的路径
   then return f //计算结束
4. $g \leftarrow 0$
5. if $AN(f)$中存在 $th(i)=0$ then
6.   if $i=s \vee i=t$ then goto 15
7.   else 删去 i 及其关联的边
8. 找到流通量最小顶点k, 从k
   将$th(k)$个单位流向前送到 t,
   向后推到 s, 并加到g上. //$th(k)>0$
9. 删去k及其关联的边
10. for $<i,j> \in AE(f)$ do
11.   if $g(i,j)= ac(i,j)$ then
12.   删去$<i,j>$
13.   else $ac(i,j) \leftarrow ac(i,j)-g(i,j)$
14. goto 5
15. $f \leftarrow f+g$
16. goto 2

Dinic算法的正确性

引理6 在每个阶段结束时，g是$AN(f)$中的极大流.

引理7 在每个阶段, $AN(f+g)$中s-t 距离大于前一个阶段$AN(f)$中 s-t 距离. 约定, 当不存在 s-t 最短路径时, 规定其距离为$+\infty$.

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

证 f 是最大流.

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,它的最大流为零流, 最大流量为 0.

根据引理4, $v^*-v(f)=0$, 其中v^*是最大流量. $v(f)=v^*$, f 是最大流.

算法时间$O(n^3)$

阶段数$\leq n$：由引理7, $AN(f)$中s-t 距离每阶段至少加1，s-t 距离$\leq n$-1, 至多n个阶段.

每阶段工作量$O(n^2)$

- 构造$AN(f)$可在$O(m)$步内完成
- 计算顶点的流通量需$O(m)$步
- 在生成 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$O(n^2)$步.
- 边操作1：修改容量后删去饱和边，至多$O(m)$次.
- 边操作2：修改容量后不删. 从同一顶点出发边中至多保留1条非饱和边. 从流通量最小顶点安排流，至多保留n条边, 上述过程至多n次, 共有$O(n^2)$次.

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

简单容量网络 边容量为1, 中间点入度为1或出度为 1

Dinic算法时间为$O(n^{1/2}m)$

7.2 最小费用流

定义

(1) 在容量网络$N=<V,E,c,s,t>$中添加单位费用 $w:E\to R^*$, 称作容量-费用网络, 记作$N=<V,E,c,w,s,t>$.

(2) 设 f 是N上的一个可行流, 称

$$w(f)=\sum_{<i,j>\in E} w(i,j)f(i,j)$$

为 f 的费用.

(3) 所有流量为 v_0的可行流中费用最小的称作流量v_0的最小费用流.

最小费用流问题

给定容量-费用网络 N 和流量 v_0, 求流量 v_0 的最小费用流.

北京大学

容量-费用网络的辅助网络

设容量-费用网络 $N=<V, E,c,w,s,t>$, 关于可行流 f 的**辅助网络**
$N(f)= <V, E(f), ac, aw, s, t >$, 其中辅助费用

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

把引理5 推广到容量-费用网络

引理9 设 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

$$w(f') = w(f) + aw(g)$$

圈流及其性质

定义 设C是N中边不重复的回路, C上的**圈流** h^C定义如下:

(1) $\forall <i, j> \in E(C)$, $h^C(i, j)=\delta$;

(2) $\forall <i, j> \in E-E(C)$, $h^C(i, j) = 0$,

其中$\delta > 0$是 h^C 的 **环流量**.

性质

(1) h^C为可行流，且 $v(h^C) = 0$, $w(h^C) = \delta \cdot w(C)$

(2) 设 f 是N上的一个可行流, h^C是$N(f)$上的一个圈流,

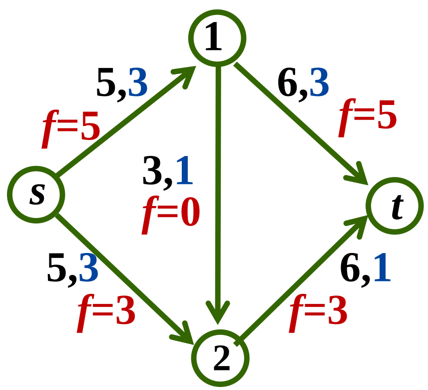
$f' = f + h^C$ 是 N上的可行流, 且

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

如果 $aw(C) < 0$, 则 $w(f') < w(f)$.

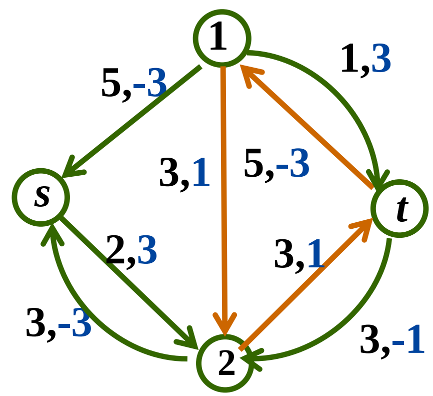
实例

例4 容量费用网络N，可行流 f，容量、费用、流值如下：



N与 f

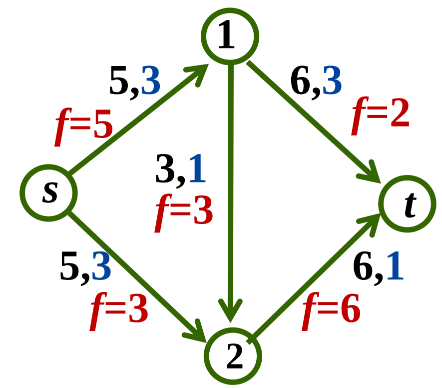
$v_0=8,\ w(f)=42.$



$N(f)$与C

$aw(C)=-1, \delta=3$

$aw(h^C)=-3$



$f_1=f+h^C$

$w(f_1)=42-3=39$

最小费用流的负回路算法

问题　容量-费用网络 $N=<V, E,c,w,s,t>$, 求N中流量为 v_0的最小费用流.

负回路算法的设计思想

1. 首先求一个流量 v_0 的可行流 f
2. 如果 $N(f)$ 中存在权 aw 的负回路 C, 令 $f' \leftarrow f + h^C$
3. 重复进行, 直至辅助网络中不存在权 aw 的负回路为止.

可证明：当$N(f)$中不存在权 aw 的负回路时, f 是最小费用流.

最短路径与负回路

求带负权的最短路径和检测负回路的算法

负回路　赋权有向图 $D=<V,E,w>$ 中权为负数的回路

命题1　赋权有向图 D中任意两点之间都有最短路径或不存在路径当且仅当 D 中不含负回路.

证　假设D中有负回路C, i 是C上的一个顶点, 那么从 i 到 j 的路径可以先重复走C若干次再到 j, 因而 i 到 j 的路径的权可以任意小.

假设D中不存在负回路C. 对任意两点 i 和 j, 如果从 i 到 j 的路径中有两个相同顶点, 删去它们之间的路径(回路)仍是从 i 到 j 的路径, 其权不增加. 只需考虑从 i 到 j 顶点都不相同的路径. 而从 i 到 j 顶点都不相同的路径只有有限条, 故一定存在最短路径.

最短路径的Floyd算法

动态规划方法.

$d^{(k)}(i, j)$: 从i到j经过号码不大于k的最短路径的长度.

递推方程

$$d^{(0)}(i, j) = w(i, j), \quad 1 \le i, j \le n$$

$$d^{(k)}(i, j) = \min\{d^{(k-1)}(i, j), d^{(k-1)}(i, k)+d^{(k-1)}(k, j)\},$$

$$1 \le i, j \le n \text{ 且 } i, j \ne k, 1 \le k \le n$$

规定: $\forall i \in V, w(i,i)=0$; $\forall \langle i, j\rangle \notin E, w(i, j)=+\infty$.

当D中不存在负回路时, i到j的距离 $d(i, j) = d^{(n)}(i, j)$.

当D中存在负回路时, 设负回路C经过i, 除i外顶点的最大号码是k, 则必有 $d^{(k)}(i, i) < 0$.

Floyd算法

$h^{(k)}(i,j)$: 从 i 到 j 经过号码不大于k 的最短路径中 i 的下一顶点

递推关系

$$h^{(0)}(i,j)=\begin{cases} j, & 若 <i,j>\in E \\ 0, & 否则 \end{cases},\qquad 1\le i,j\le n$$

$$h^{(k)}(i,j)=\begin{cases} h^{(k-1)}(i,j), 若 d^{(k-1)}(i,j)\le d^{(k-1)}(i,k)+d^{(k-1)}(k,j), \\ h^{(k-1)}(i,k), 否则 \end{cases}$$

$1\le i,j\le n$ 且 $i,j\ne k$, $1\le k\le n$

算法：利用递推公式迭代计算 $d^{(k)}(i,j)$ 和 $h^{(k)}(i,j)$,

$i,j,k=1,2,\ldots,n$

$d^{(n)}(i,j)$ 是 i 到 j 的最短路长度

$h^{(n)}(i,j)$ 用于追踪从 i 到 j 的路径

算法伪码

算法4 最小费用流的负回路算法

1. 调用最大流算法, 若求得流量v_0的可行流 f, 则转2; 若最大流量小于v_0, 则输出“无流量v_0的可行流”, 计算结束.
2. 构造 $N(f)$.
3. 用Floyd算法检测$N(f)$中是否存在权aw的负回路. 若存在负回路C, 则转4; 若不存在负回路, 则输出 f, 计算结束.
4. 令$\delta \leftarrow \min\{ ac(i, j) \mid \langle i, j\rangle \in E(C) \}$　　// h^C的环流量为δ
5. $\forall \langle i, j\rangle \in E(C)$, 若$\langle i, j\rangle \in E$, 令 $f(i, j) \leftarrow f(i, j)+\delta$;
   若$\langle j, i\rangle \in E$, 令 $f(j, i) \leftarrow f(j, i)-\delta$.　　// $f \leftarrow f + h^C$
6. 转2

其他算法

最短路径算法

设计思想：

1. 从一个初始的最小费用流 f (如零流)开始,
2. 如果 $v(f)<v_0$, 找一条费用最少的 s-t 增广链 P, 修改 P上的流量, 得到新的可行流 f'.
3. 重复进行, 直至流量等于v_0为止.

若存在负回路，用 Floyd 算法计算最短路径;

若不存在负回路，可以用 Ford算法.