HIT
MDC

第八章
图算法

张开旗
海量数据计算研究中心
计算学部

提要

8.1 网络流算法

8.2 匹配问题

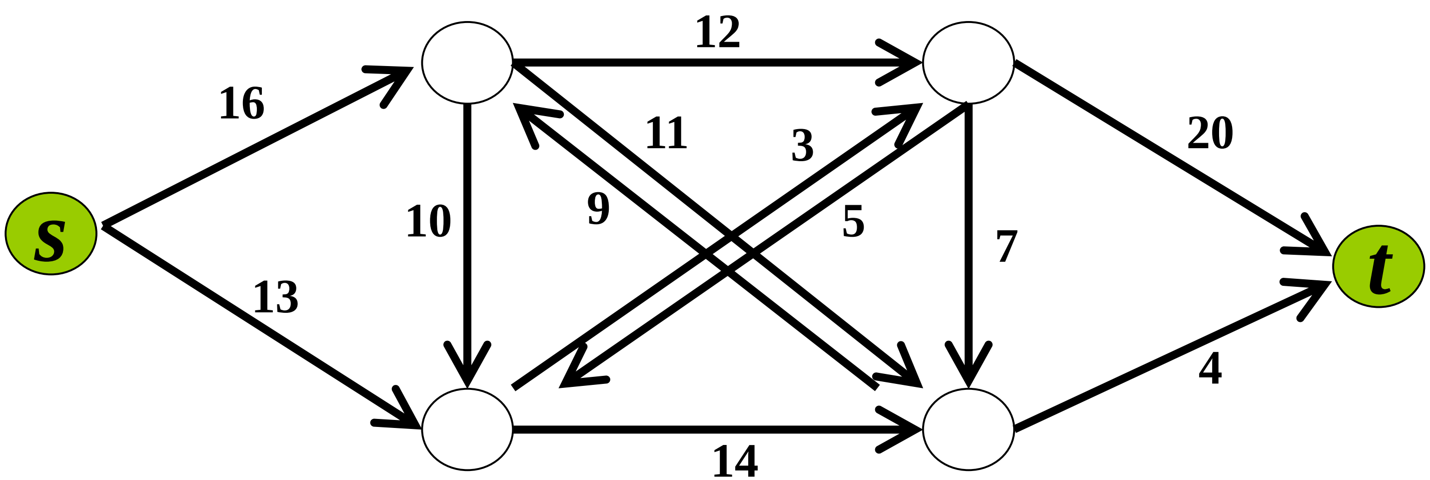
8.1 网络流算法

- 基本概念与问题定义
- Ford-Fulkerson方法
- 推送复标(Push-Relable)方法

8.1.1 基本概念与问题定义

应用背景

- 很多实际问题可以建模为流网络
  - 装配线上物件的流动
  - 电网中电流的流动
  - 通信网络中信息的流动
  - 道路交通网络中的运输活动
  - ……
- 一个源点s、一个汇点t，由源点流向汇点



- 流网络

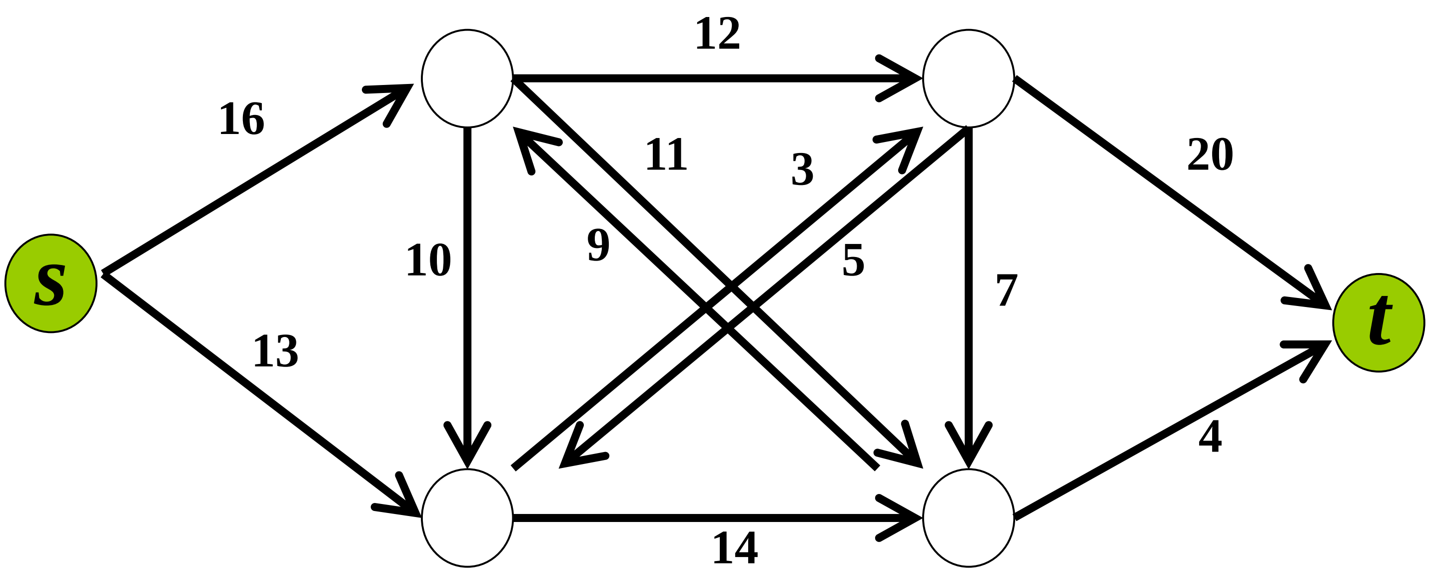
是一个无自环的有向图$G=(V, E)$,

(1) 图中的每条边$(u, v)\in E$有一个非负的容量值$c(u, v)\geq 0$。

if $(u, v)\notin E$ $c(u, v)=0$;

(2) 有两个特殊结点$s, t\in V$, s称为源结点(source), t称为汇点(sink): s没有入边, t没有出边。

(3) For $\forall v\in V$, 存在一个s到t经过v的路径$s\Rightarrow v\Rightarrow t$.



流网络是连通的

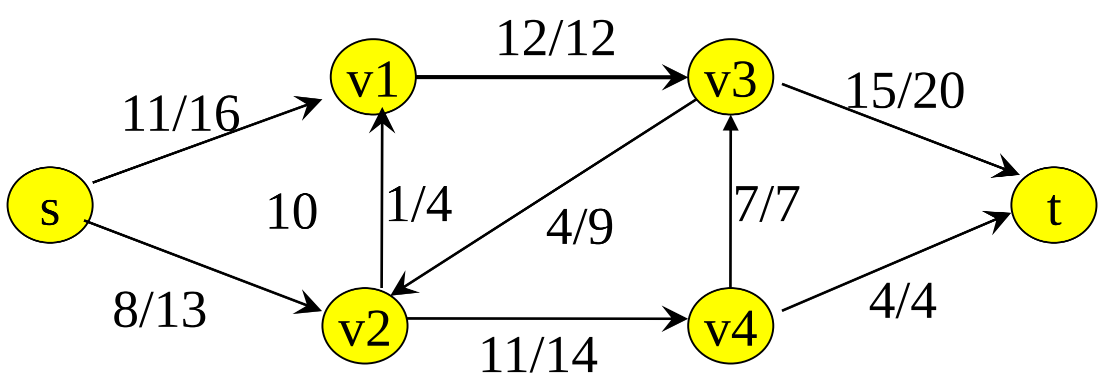
除源结点外, 每个结点都至少有一条进入的边, 所以$|E|\geq|V|-1$

- 流(Flow)

设$G(V, E)$是一个流网络，c是容量函数，s源结点，t是汇点。
G中的流是一个实值函数f：$V\times V\rightarrow R$，满足下列性质：
(1) 容量限制：$\forall u,v\in V,\ 0\le f(u, v)\le c(u, v)$；
(2) 流量守恒：$\forall u\in V-\{s,t\}$，有如下

$$\sum_{w\in V} f(w, u) = \sum_{v\in V} f(u, v)$$ 理解：流入的=流出的

称$f(u, v)$为从结点u到结点v的流
当$(u, v)\notin E$时，从结点u到v之间没有流，因此$f(u, v)=0$.

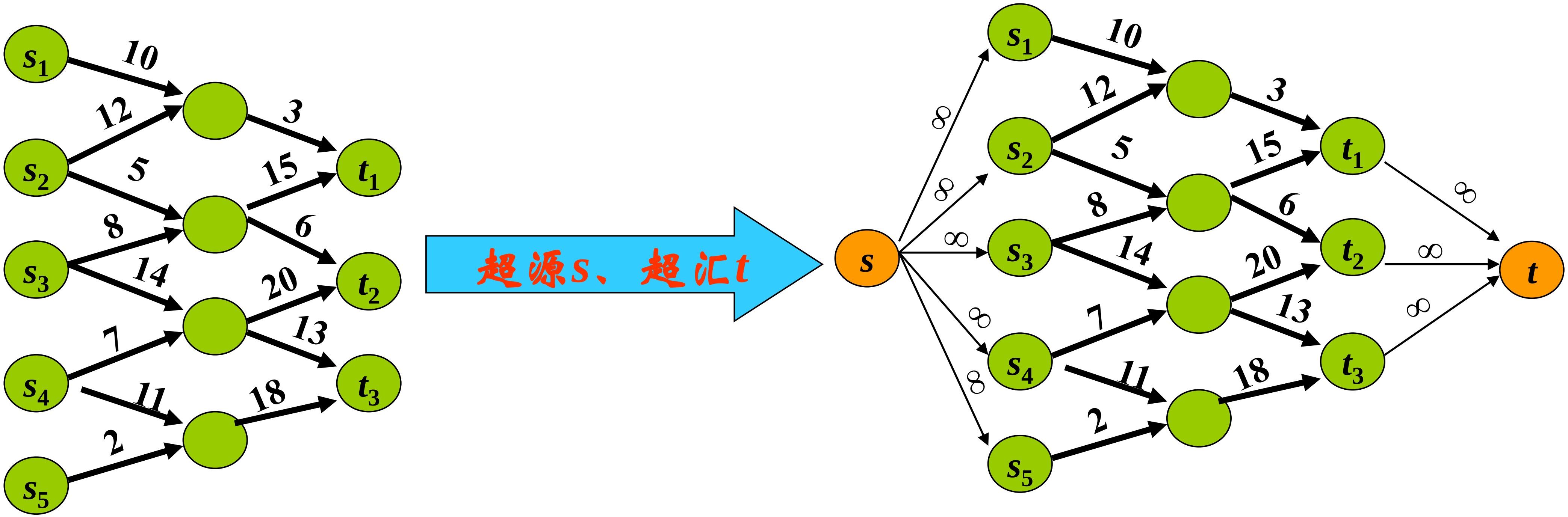


一个流f的值$|f|$定义为：

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

基本概念

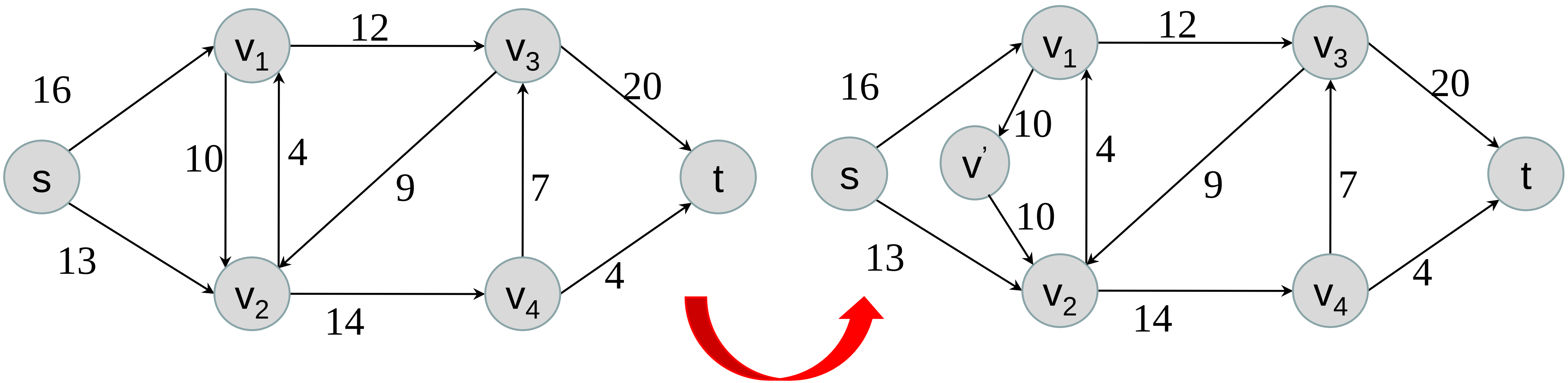
- 多源多汇的网络



只需讨论单源单汇的网络流

一些假设

- 单源点、单汇点流网络
- 假设：流网络中无反向边
  - 给定有向图$G=(V, E)$，如果边$(u, v)\in E$，则边$(v, u)\notin E$



最大流问题

- 问题定义
  - 输入: 流网络G=(V, E)
  - 输出: 具有最大流值的流f

直观想法

- 循环递进
  - 初始：网络上的流为0
  - 找出一条从s到t的路径p和正数a，使得p上的每一条边(u,v)的流量增加a之后仍能够满足容量约束：$f(u,v)+a \le c(u,v)$

    //将p上的每条边的流量增加a，得到一个更大的流
  - 重复执行第二步，直到找不到满足约束条件的路径.

关键在于：

1. 如何找路径p，以便得到更大的流?
2. 如何判断循环结束的条件?

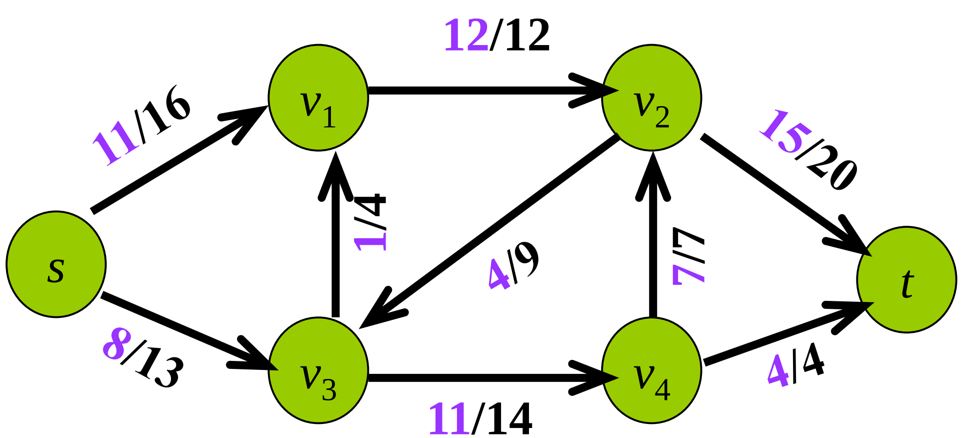
即：当循环结束时，所得到的流一定是最大流么？

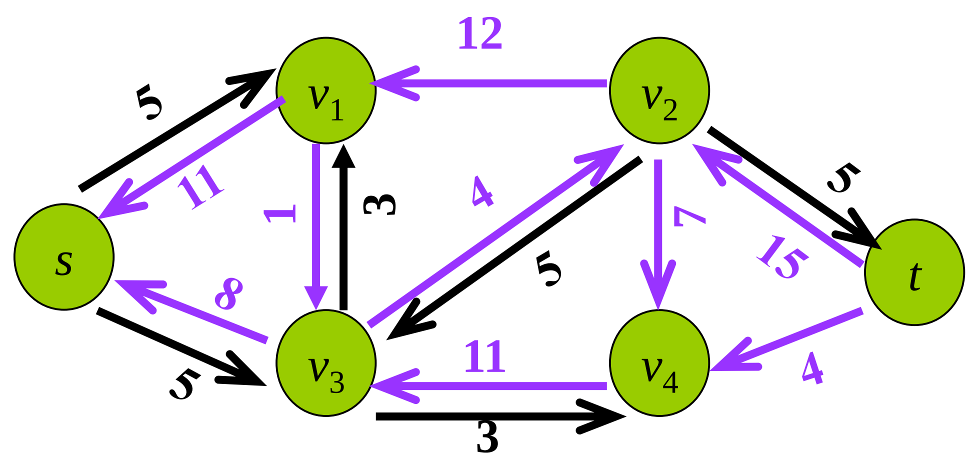
8.1.2 Ford-Fulkerson方法

- 如何找路径p，以便得到更大的流?
- 如何判断循环结束的条件？

相关概念

- 在一个关联的剩余网络(余图)中寻找一条增广路径
- 剩余网络(Residual network)
  - 给定流网络$G(V,E)$和一个流f，则由f诱导的G的剩余网络为$G_f = (V, E_f)$，其中E_f为
    - 对于G中每条边(u, v)，若$c(u,v)-f(u,v)>0$，则$(u, v) \in E_f$，且$c_f(u,v) = c(u,v)-f(u,v)$　(称$c_f(u,v)$为剩余容量)
    - 对于G中每条边(u, v)，在G_f中构造边(v, u)，且$c_f(v, u) = f(u,v)$

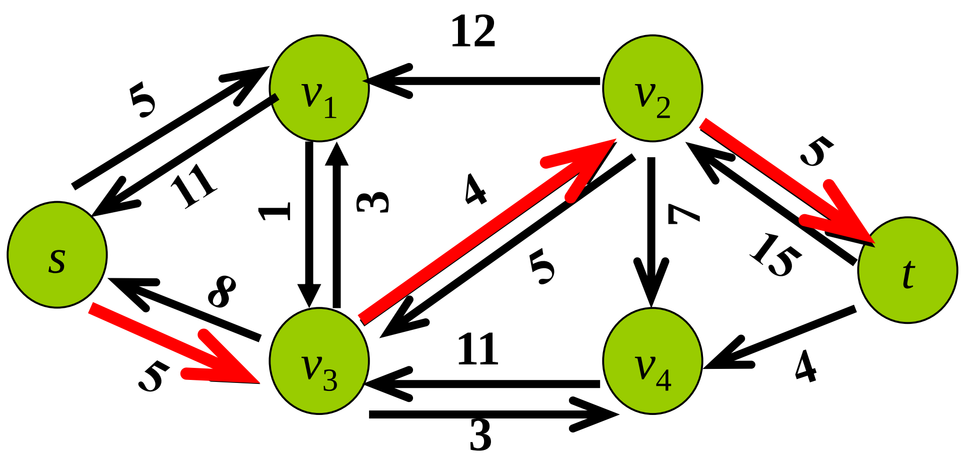




E_f中的边要么是E中原有的边，要么是其反向边，因此$|E_f| \le 2|E|$

相关概念

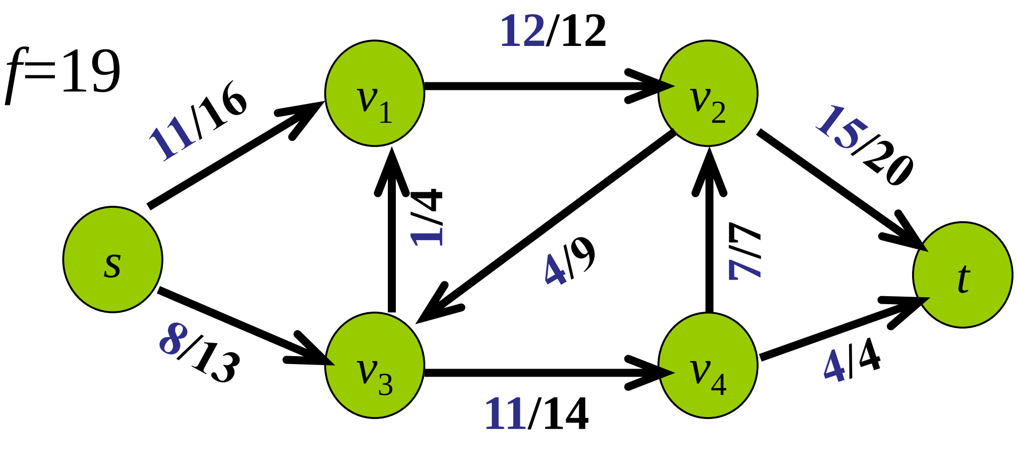
- 增广路径
  - 剩余网络中的一条由源结点s到汇点t的一条路径p
- 增广路径p的剩余容量
  - $c_f(p)=\min\{c_f(u, v) : (u, v)$属于路径$p\}$
    - 表示了该路径能够增加的流的最大值



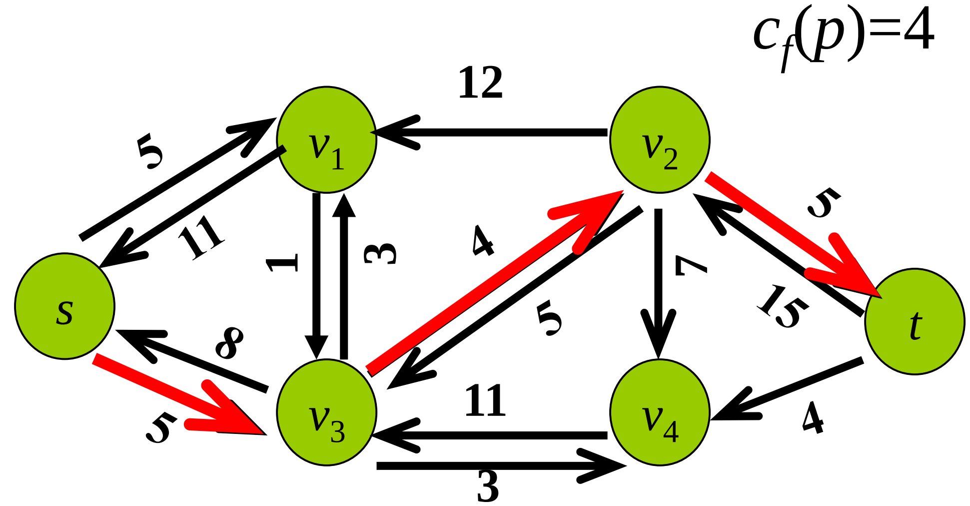
图中红色标注的路径为一条增广路径，其剩余容量为4

Ford-Fulkerson方法

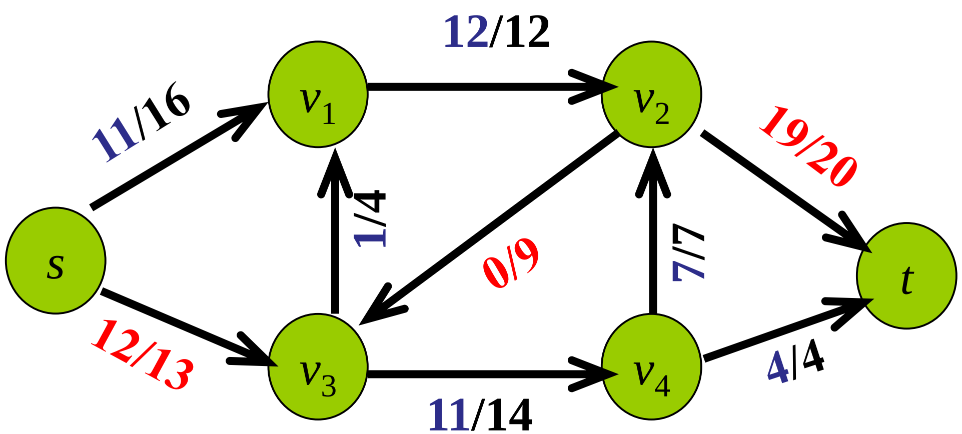
- 在剩余网络中寻找增广路径



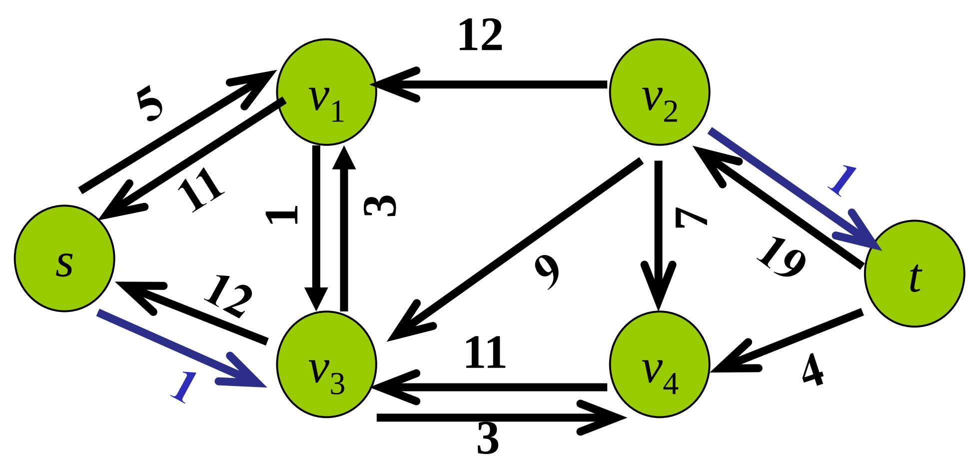
(a) 流网络G及流f



(b) 由(a)诱导的剩余网络



(c) 由增广路径得到的更大流



(d) 由(c)诱导的剩余网络

Ford-Fulkerson方法

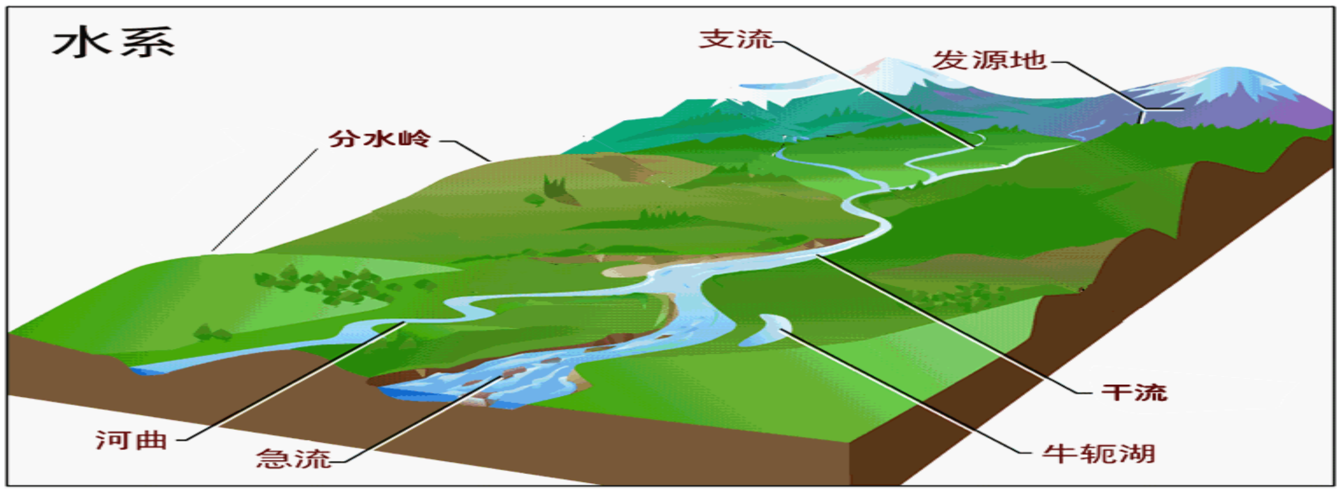
- FF算法的核心是：通过增广路径不断增加路径上的流量，直到找到一个最大流为止

问题：
如何判断算法结束时，确实找到了一个最大流？

如何判断是否已获得最大流?

河水的最大流量取决于

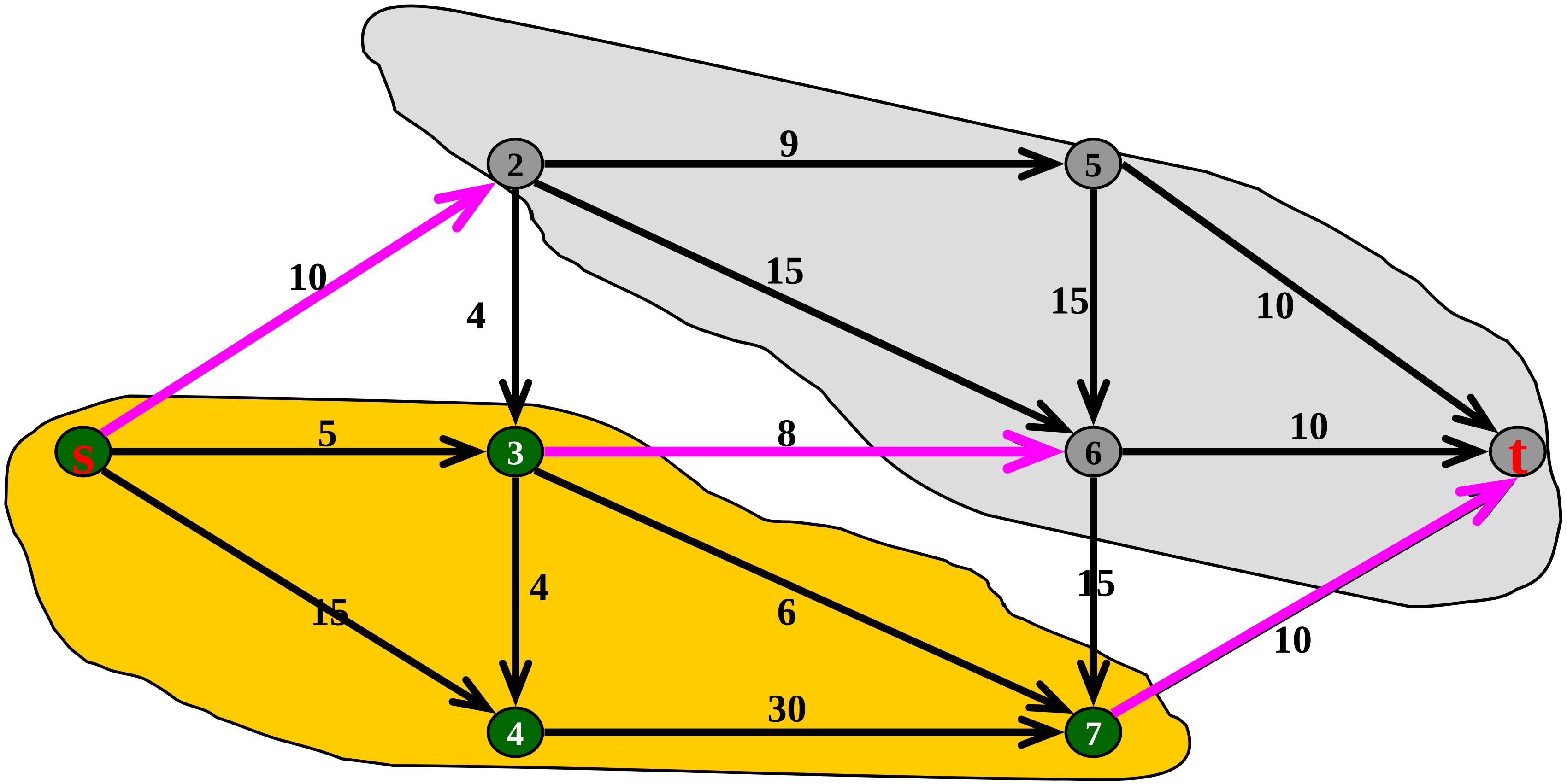
干流中河道狭窄处的通行能力



这种观察能否用于最大流问题呢?

如何判断是否已获得最大流？

从s流到t的最大流量不会超过10+8+10=28



流网络的割

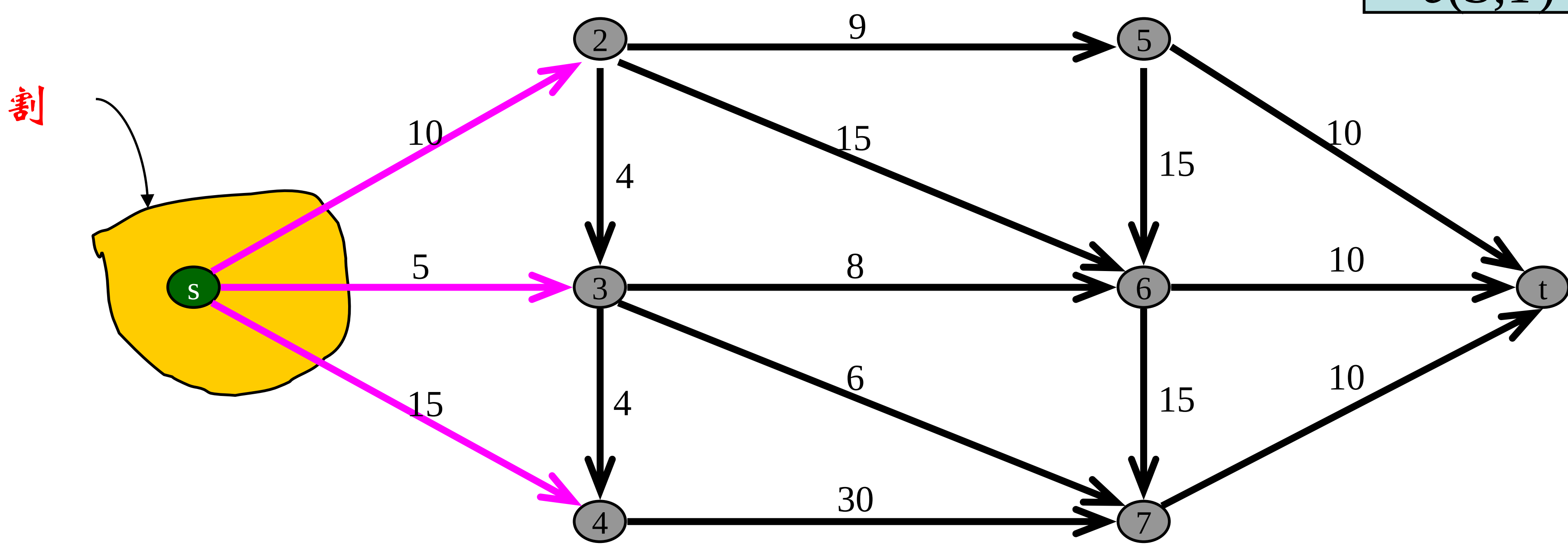
给定流网络 $G=(V,E)$,其源为点s，汇点为t,

G的一个割(S, T)是结点集合V的划分, T=V-S 且$s\in S$, $t\in T$。

割的容量定义为

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

$c(S,T) = 30$

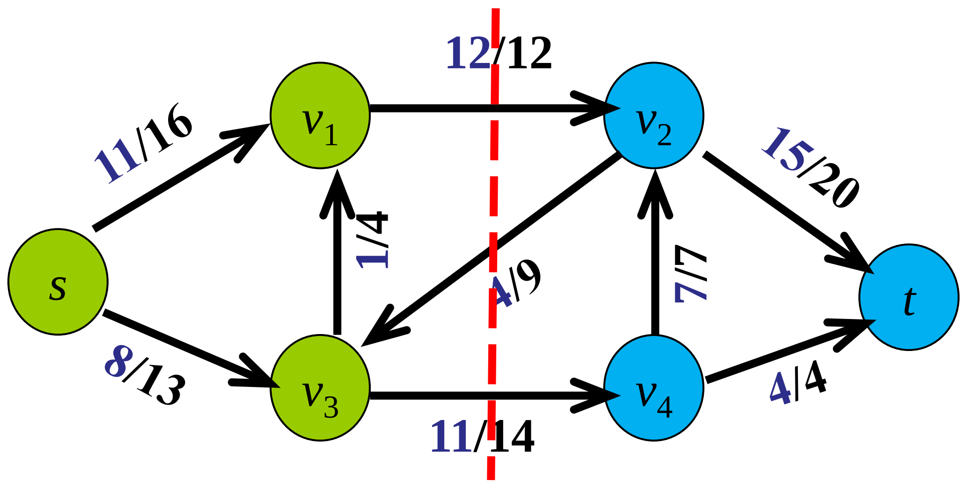


流网络的割

引理1. 设f为流网络G的一个流，该流网络的源点为s，汇点为t，设(S,T)为流网络G的任意一个割，则横跨割(S,T)的净流量为流值$|f|$.

横跨割(S,T)的净流量定义为：

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



流网络G及流f

$|f|=19$

推论1. 流网络G中任意流的值不能超过G的任意割的容量.

由引理知：

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

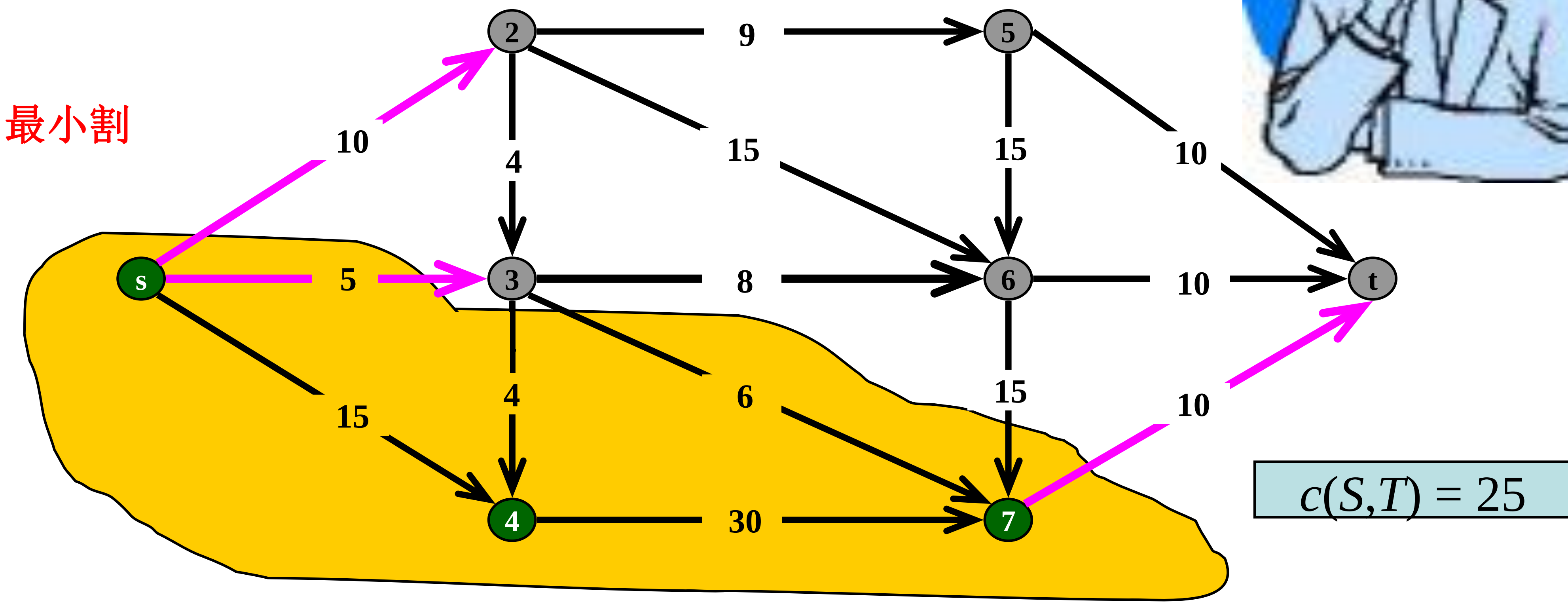
$$\le \sum_{u\in S}\sum_{v\in T} f(u,v)$$

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

最小割

- 一个流网络的最小割

  是指：整个流网络中容量最小的割



Max-Min关系

最大流最小割定理：

设f为流网络$G(V,E)$一个流，该流网络的源点为s，汇点为t，则下面命题等价：

1. f是G的最大流.
2. 剩余网络G_f不包含增广路径.
3. 对于G的某个划分(S, T), $|f|=c(S, T)$.

一个最大流的流值实际上等于一个最小割的容量

剩余网络G_f不再包含增广路径即可

Max-Min关系：对偶关系

最大流与最小割

最大匹配与最小覆盖

……

Max-Min关系

- 对同一问题从不同角度考虑，有两种对立的描述
  - 例如，平面中矩形面积与周长的关系

正方形： 周长一定，面积最大的矩形

面积一定，周长最小的矩形

Max问题

Min问题

利用Max-Min关系求解最大流问题

1. 初始化一个可行流f
   - 0-流：所有边的流量均等于0的流
2. 不断将f增大，直到f不能继续增大为止
3. （能够）找出一个割(S,T)使得$|f|=c(S,T)$
   - 由此断言f是最大流，而(S,T)是最小割

Max-Min关系提供了高效求解最大流-最小割问题的机制！

Ford-Fulkerson算法

算法Ford-Fulkerson(G,s,t)

Input　流网络G，源s，汇t

Output　G中从s到t的最大流

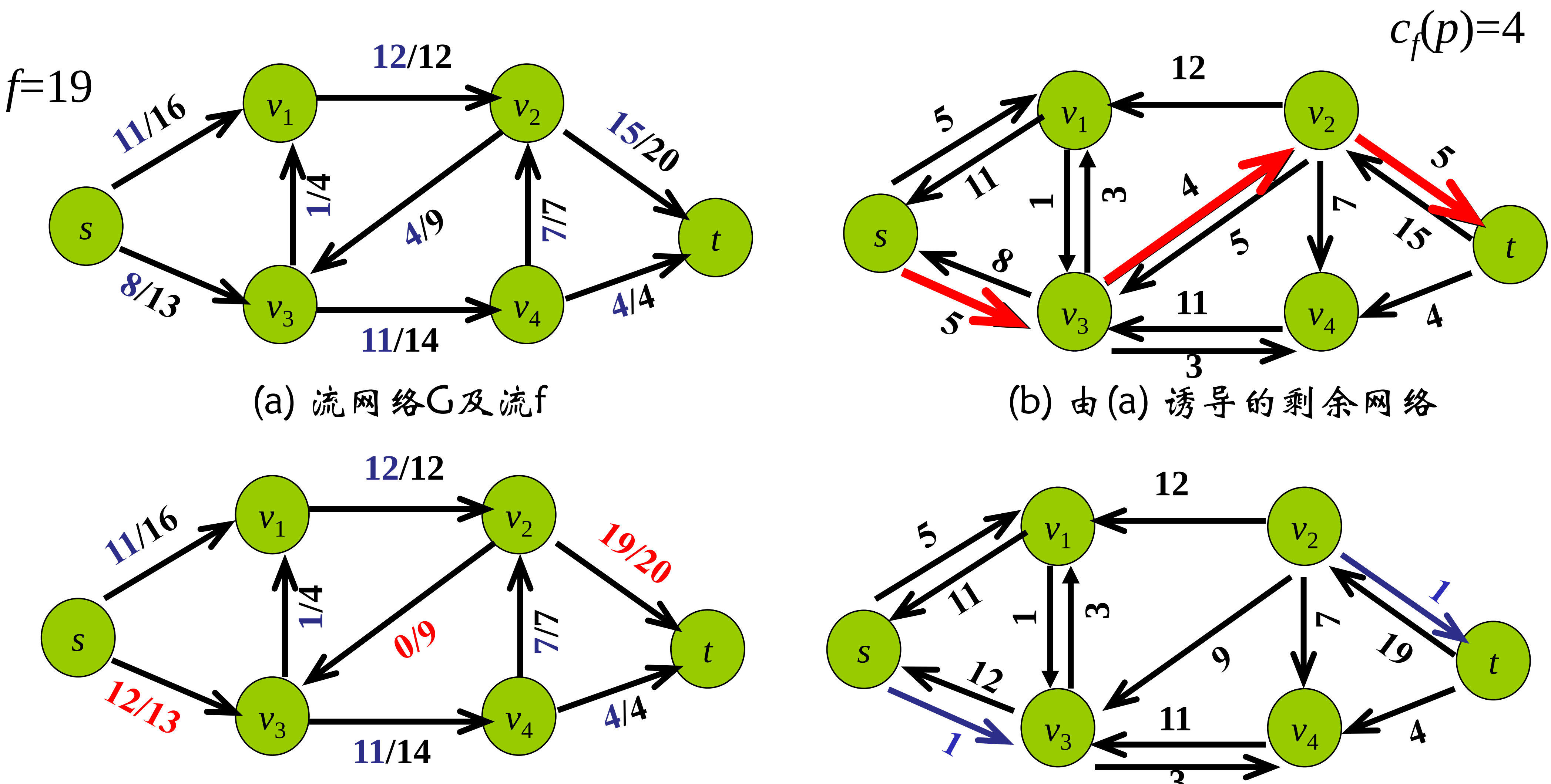
For $\forall(u,v)\in E[G]$ do
　　$f(u,v)\leftarrow 0$
　　$f(v,u)\leftarrow 0$
While G_f存在增广路径p do
　　$c_f(p)=\min\{c_f(u,v)\mid (u,v)$是$p$上的边$\}$
　　For p上的每条边(u,v) do
　　　　If (u,v)是流网络中的边 Then
　　　　　　$f(u,v)\leftarrow f(u,v)+c_f(p)$
　　　　Else
　　　　　　$f(v,u)\leftarrow f(v,u)-c_f(p)$

Ford-Fulkerson算法分析

- 正确性分析
  1. 算法输出的一定是最大流
     - 由最大流－最小割定理可得
  2. 算法可终止性
     - 假设整数容量，每次流量增加1

Ford-Fulkerson方法

- 在剩余网络中寻找增广路径



(a) 流网络G及流f

(b) 由(a)诱导的剩余网络

(c) 由增广路径得到的更大流

(d) 由(c)诱导的剩余网络

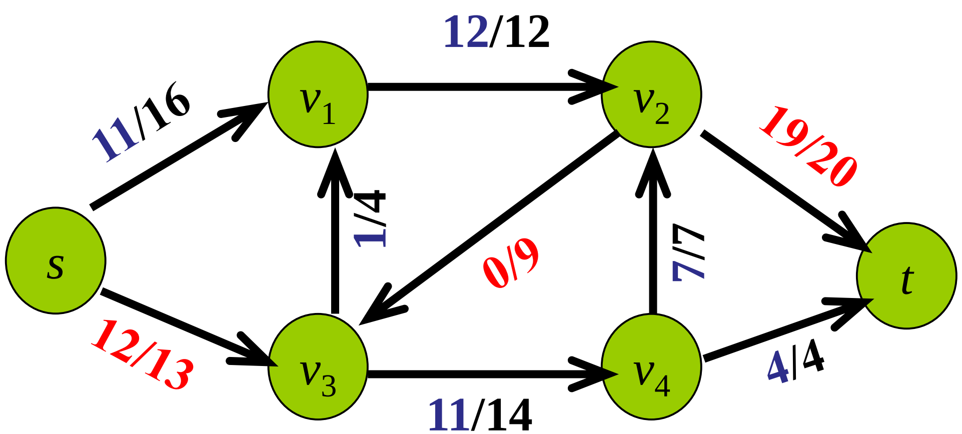
FF算法可以求得最大流

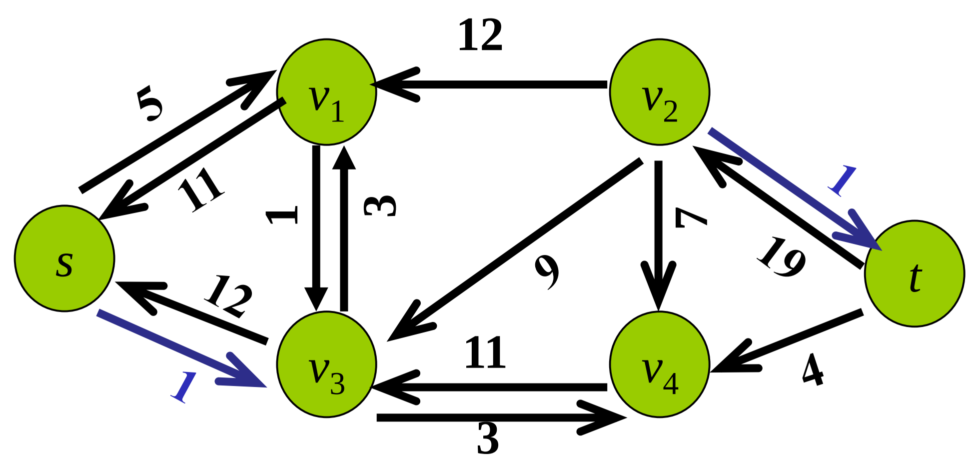
设FF算法在流 f 上停止：

- 在剩余网络中从 s 出发的DFS不包含 t
- 这意味着在剩余网络中DFS经过的结点和其余结点之间割的容量是0
- 因此这个割中的每条边都被增广流逆转过，这意味着

$$c(\text{DFS},\text{DFS}') = |f|$$

- 因此可知得到的 $|f|$ 最大





Ford-Fulkerson算法分析

算法Ford-Fulkerson(G,s,t)

Input 流网络G，源s，汇t

Output G中从s到t的最大流

1. For $\forall(u,v)\in E[G]$ do
2. $f(u,v)\leftarrow 0$
3. $f(v,u)\leftarrow 0$
4. While G_f存在增广路径p do
5. $c_f(p)=\min\{c_f(u,v) \mid (u,v)$是$p$上的边$\}$
6. For p上的每条边(u,v) do
7. If (u,v)是流网络中的边 Then
8. $f(u,v) \leftarrow f(u,v)+c_f(p)$
9. Else
10. $f(v,u) \leftarrow f(v,u)-c_f(p)$

1-3步：$O(E)$

4-8步：循环次数最多为$|f^*|$

f^*是最大流，每次循环流值至少增加一个单位

第4步在G_f中找路径(深度或广度优先)代价$O(E)$

总的复杂度$O(|f^*||E|)$

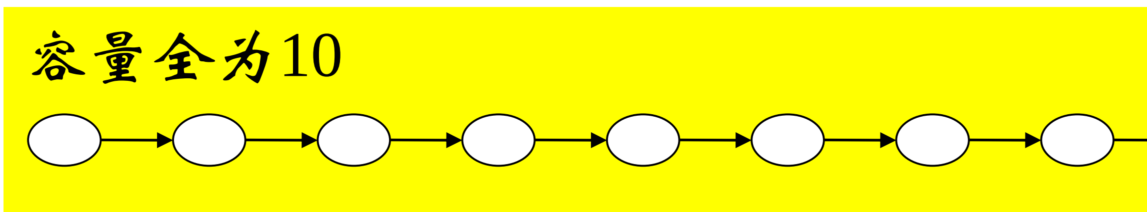
如何改进Ford-Fulkerson算法？

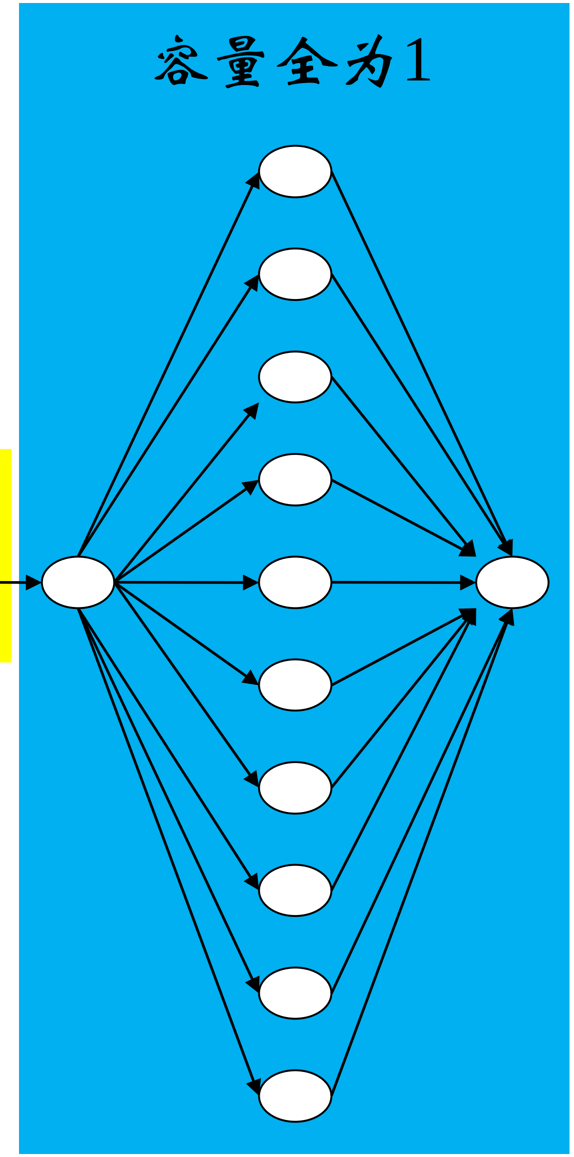
8.1.3 推送复标(Push-Relable)方法

HIT
MDC

基于增广路径算法的问题

- 基于增广路径算法的缺陷
  - 每次选一条增广路径
  - 每一次增广复杂度为O(|*E*|)





无论采用何种增广路径算法,都会找到10条增广路径,每条路径长为10,容量为1. 总代价为10*10.

能否直接将前8条边的流量增广10个单位,而只对后面长为2的不同的有向路单独操作呢?

预流推进 (preflow push)的思想

基于增广路径算法的问题

- 预流推进 (preflow push)与增广路径算法的区别
  - 增广路径算法
    - 从全局考虑，沿着一条从s到t的路径推送流量
    - 要求结点流量守恒：流入结点i的流量 = 流出结点i的流量
  - 预流推进算法
    - 仅考虑结点局部信息，每次只在一条边上最大可能地推送流量
    - 没有结点流量守恒约束

$$\text{流入节点 } i \text{ 的流量} \geq \text{流出节点 } i \text{ 的流量}(i \neq s).$$

预流(preflow)

- 在每个中间阶段，允许到达结点的流量比离开结点的流量多
- 预流：是一个函数f：$V\times V\rightarrow R$，满足
  - 容量约束性质
  - 弱化的流量守恒性质：$\forall u\in V\text{-}\{s\}$，

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

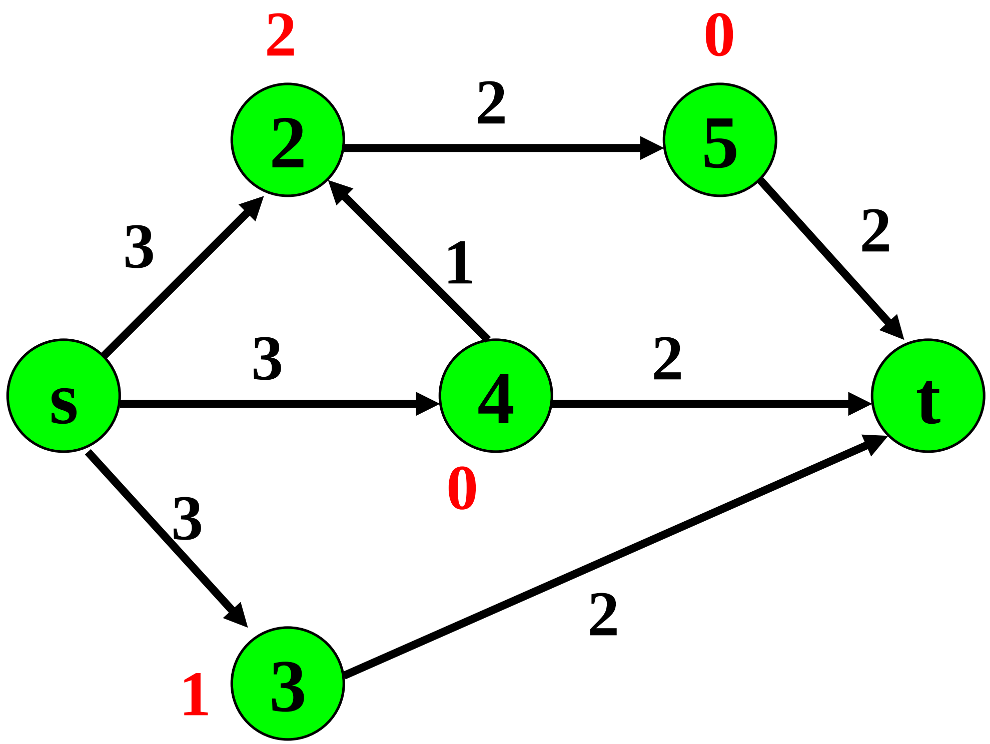
    - 即：进入一个结点的流可以超过流出该结点的流

$$e(u) = \sum_{v\in V} f(v,u) - \sum_{v\in V} f(u,v)$$ 为进入结点u的超额流

如果对于$\forall u\in V\text{-}\{s\}$，$e(u)>0$，则称结点$u$溢出

预流(preflow)

- 一个可行的预流

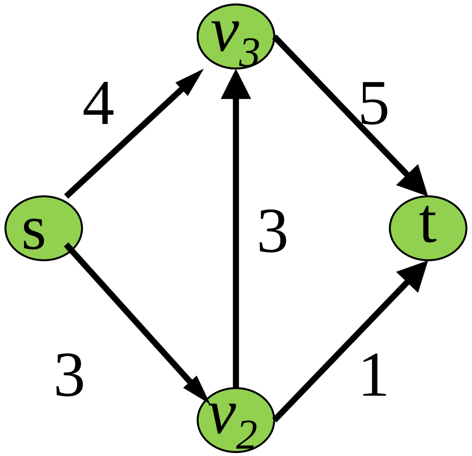


每个节点$u(j \neq s)$的超额流 $e(u)$=流入u的流量 − 流出u的流量

总超额流=流出s与流入t的流量之差。

预流(preflow)

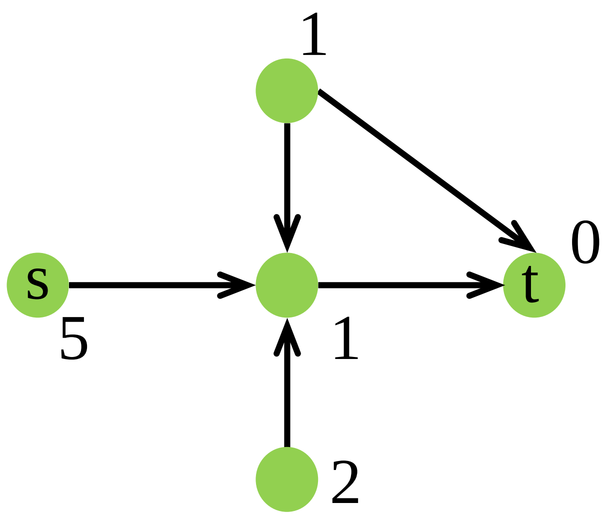
- 将流网络看成管道网络
  - 结点具有库存能力
  - 液体将怎么流动?
  - 引导机制：结点设置不同高度

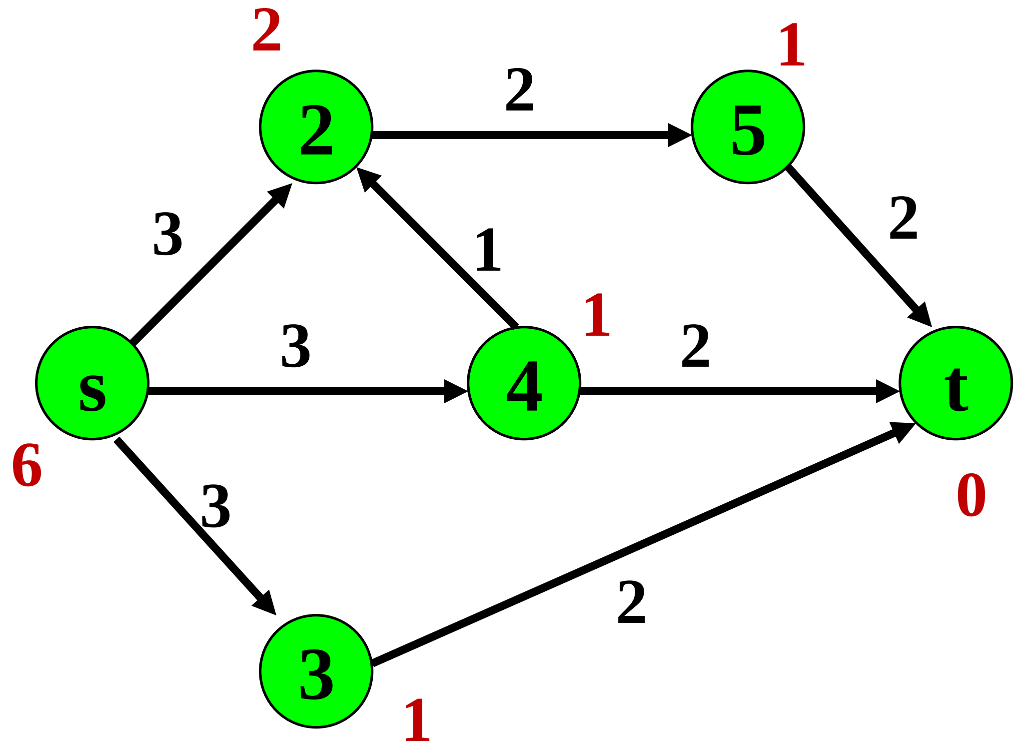


预流(preflow)

- 结点的高度
  - 设$G(V,E)$是一个源结点为s，汇点为t的流网络，f为G的一个预流。如果函数h：$V \rightarrow N$，满足下面条件，则h是一个高度函数。
    - $h(s)=|V|$, $h(t)=0$，并且
    - 对于所有的边$(u, v) \in E_f$，有$h(u) \leq h(v) + 1$

      结点u至多比v高一个高度
- 如果一个结点溢出了，那么它的超额流只能流向高度标号比自己低的结点(水往低处流)
- 对于任意结点u,通常用其在剩余网络中到汇点t的最短距离$d(u)$来表示其高度
  - 满足高度函数h的两个条件





引理3：设$G(V,E)$为一个流网络，f为一个预流，设h为V上的高度函数。对于任意两个结点$u,v\in V$，如果$h(u)>h(v)+1$，则(u,v)不是剩余网络中的一条边

由高度函数的定义可直接得到

两个基本操作

- 推送操作：push(u,v)
  - 将结点u的超额流推送到结点v
  - 操作执行的前提条件：存在可容纳的边(u,v)
    - 结点u是一个溢出结点($e(u)>0$)，
    - 在剩余网络中边(u,v)上的剩余容量$c_f(u,v)>0$，并且
    - $h(u)=h(v)+1$

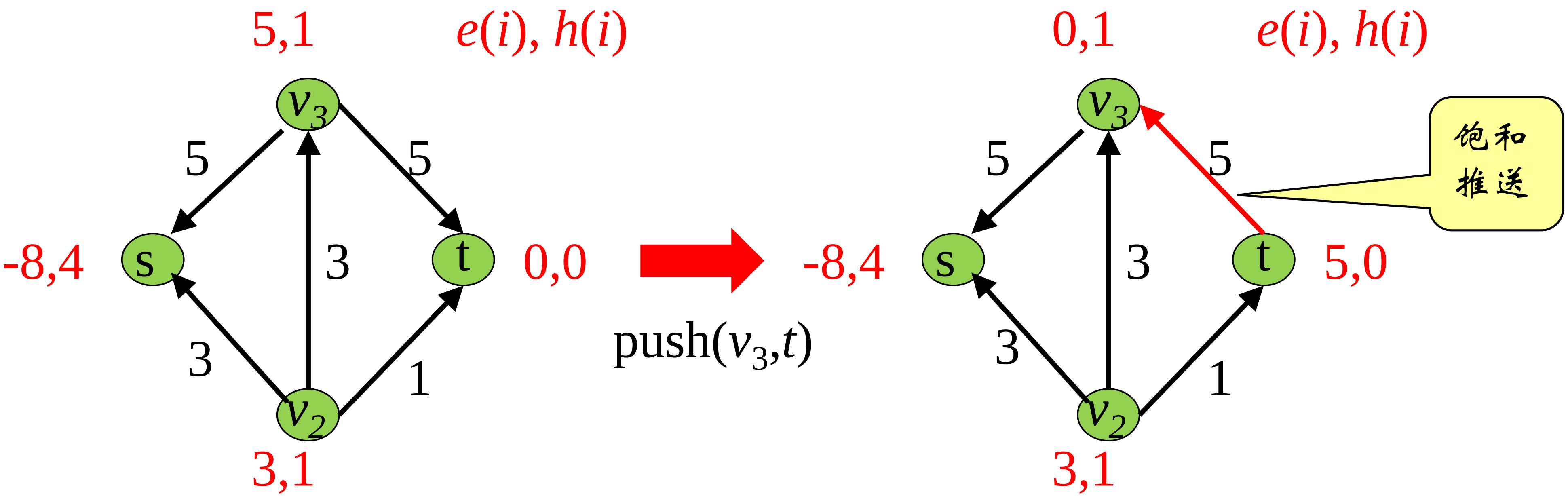
Push(u,v)
//对预流f和u，v两个结点的超额流进行更新
1. $\Delta_f(u,v)=\min(e(u), c_f(u,v))$
2. 从u到v推送$\Delta_f(u,v)$个单位的流

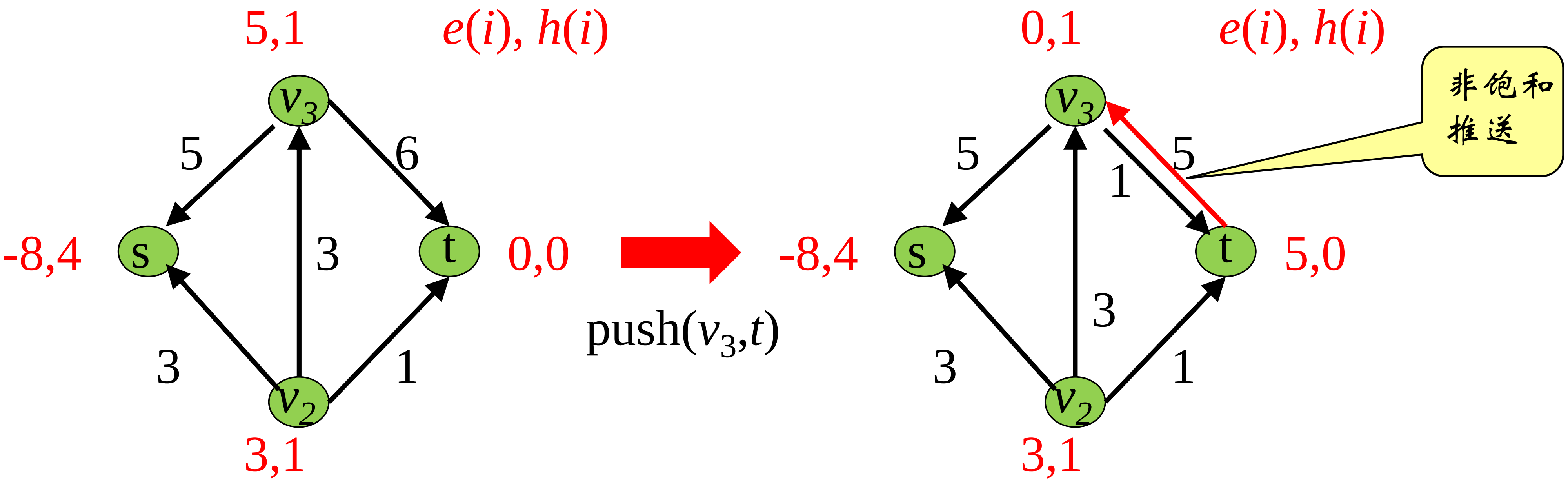
推送操作只将超额流向高度差为1的下层节点推送

因此，如果推送操作前f是一个预流，则push操作后f仍然是一个预流

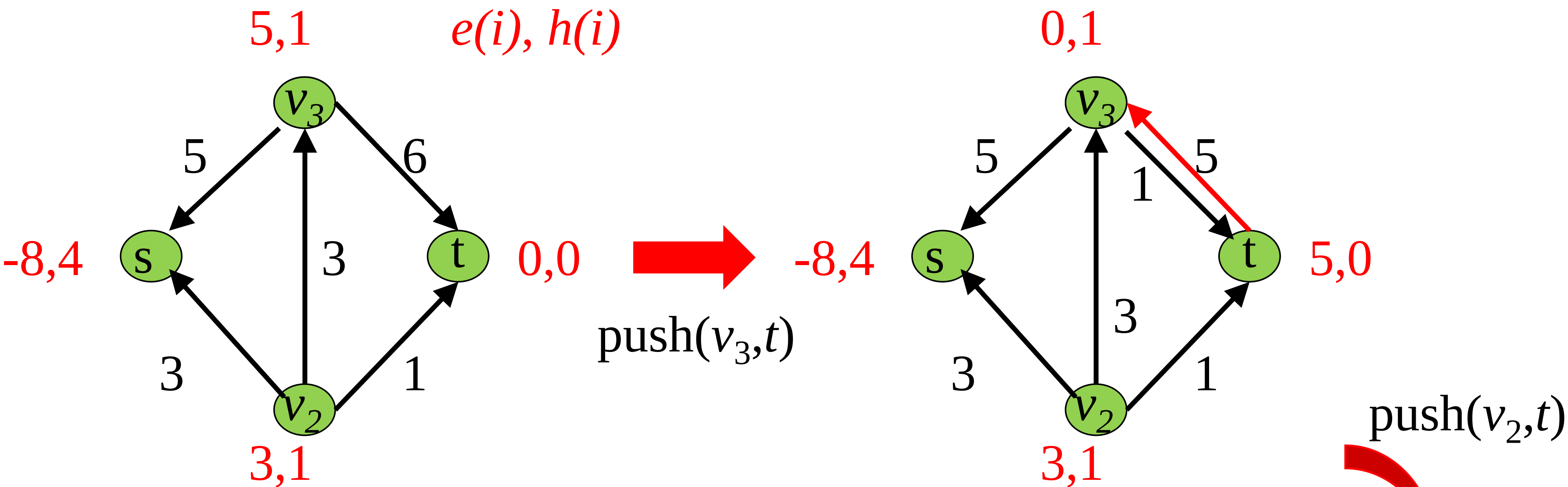
两个基本操作

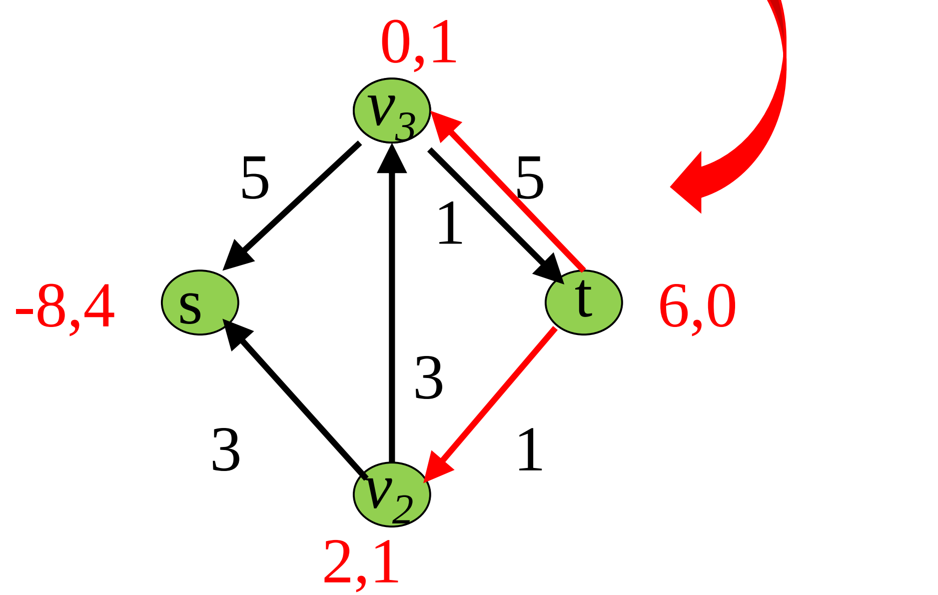
- 推送操作：push(u,v)
  - 饱和推送
    - 如果push(u,v)操作后， $c_f(u,v)=0$. 称边(u,v) 达到饱和状态
    - 如果一条边达到饱和状态，它将从剩余网络中消失
  - 非饱和推送





引理4：在从u到v的一个非饱和推送后，结点u将不再溢出



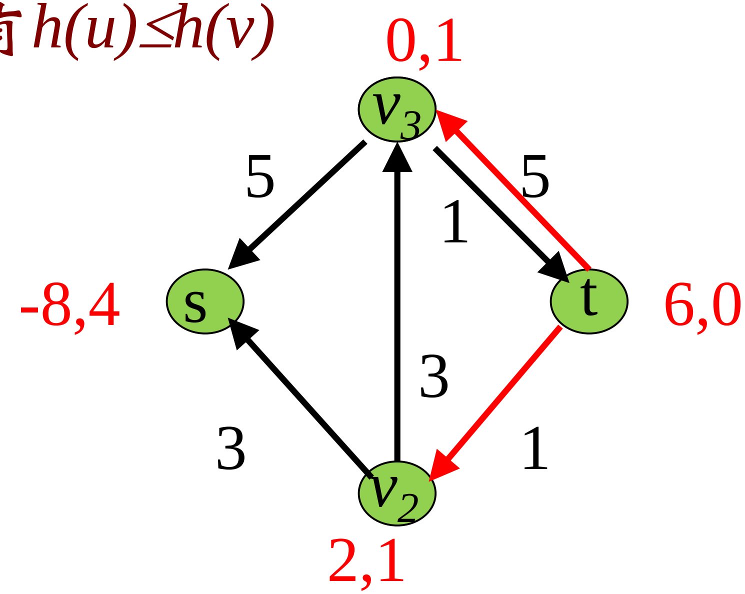


v_2虽然溢出，但无法继续推送

两个基本操作

- 重贴标号操作：Relable(u)
  - 对一个溢出结点进行重贴标号的操作：提高结点高度
  - 操作执行的前提条件：
    - 结点u是一个溢出结点$(e(u)>0)$，
    - 对于剩余网络中所有的边$(u,v)\in E_f$，有$h(u)\le h(v)$

Relable (u)
//提高结点u的高度
1. $h(u)=1+\min\{h(v): (u,v)\in E_f\}$



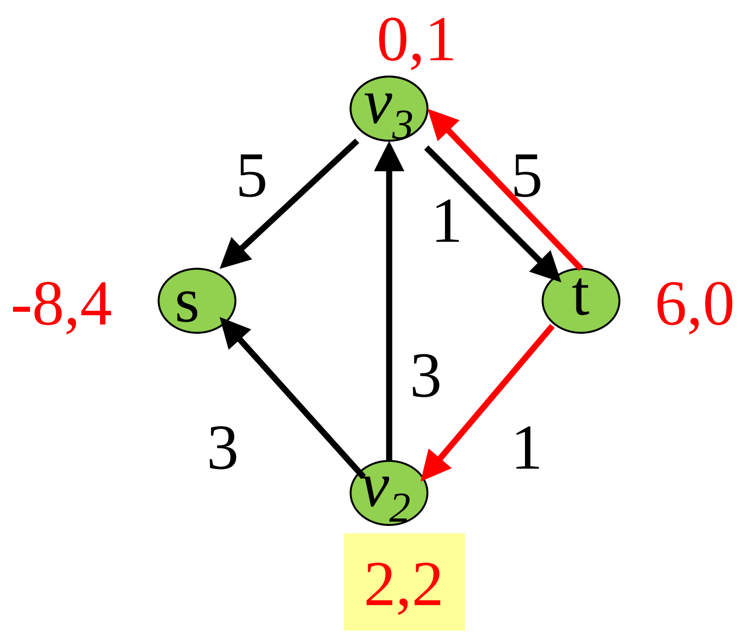
使用了relable操作后，至少存在一个(u,v)满足$h(u)=h(v)+1$

两个基本操作

- 重贴标号操作：Relable(u)
  - 对一个溢出结点进行重贴标号的操作：提高结点高度
  - 操作执行的前提条件：
    - 结点u是一个溢出结点($e(u)>0$)，
    - 对于剩余网络中所有的边$(u,v)\in E_f$，有$h(u)\le h(v)$

Relable (u)
//提高结点u的高度
1. $h(u)=1+\min\{h(v): (u,v)\in E_f\}$

例如：对结点v_2重贴标号



Push-relable算法

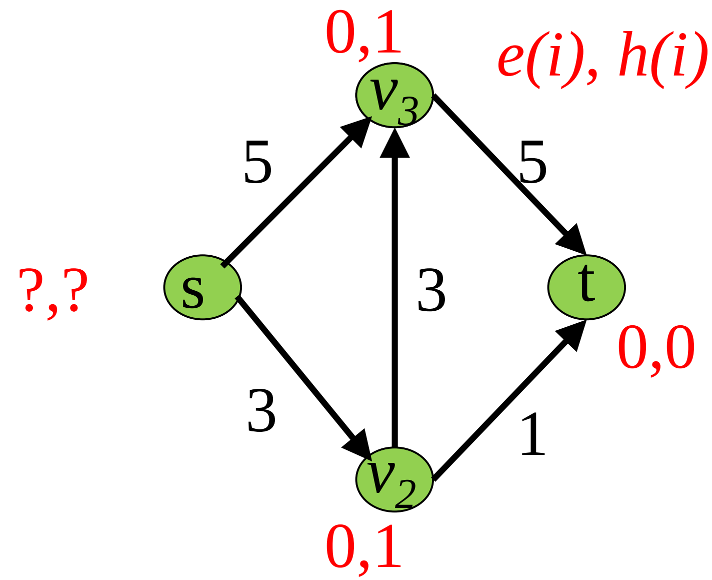
- 初始化预流：
  - $\forall u \in V-\{s\},\ h(u)=d(u),\ e(u)=0$

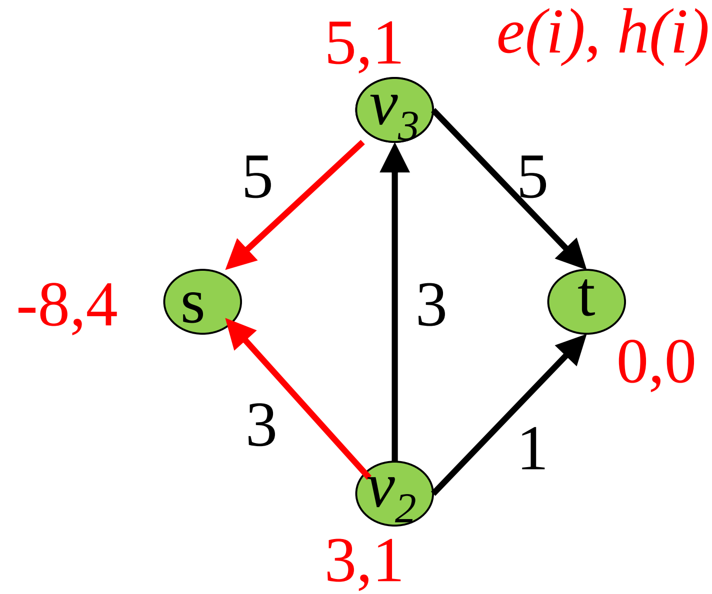
    $\forall (u,v) \in E, f(u,v)=0$
  - $h(s)=|V|$

    $\forall (s,v) \in E,$

    $f(s,v)=c(s,v),\ e(v)=c(s,v),$

$$e(s)=\sum_{(s,v)\in E} -c(s,v)$$





从s出发的所有边都充满流，且这些边已达到饱和状态而其它边上都没有流

Push-relable算法

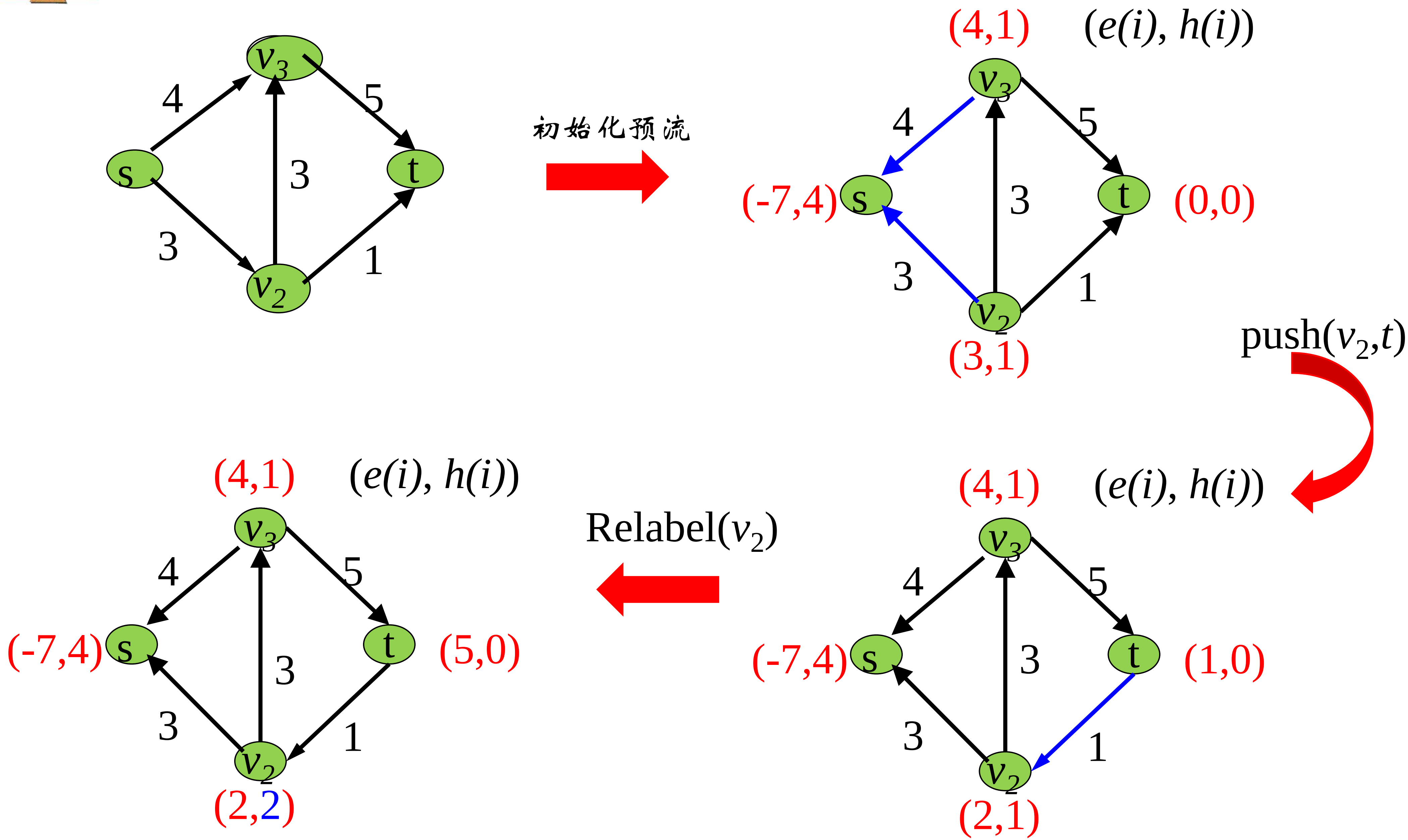
Generic push-relable(G)

1. 初始化预流;
2. While 若存在一个可行的push或relable操作
3. 选择一个push或relable操作，执行之

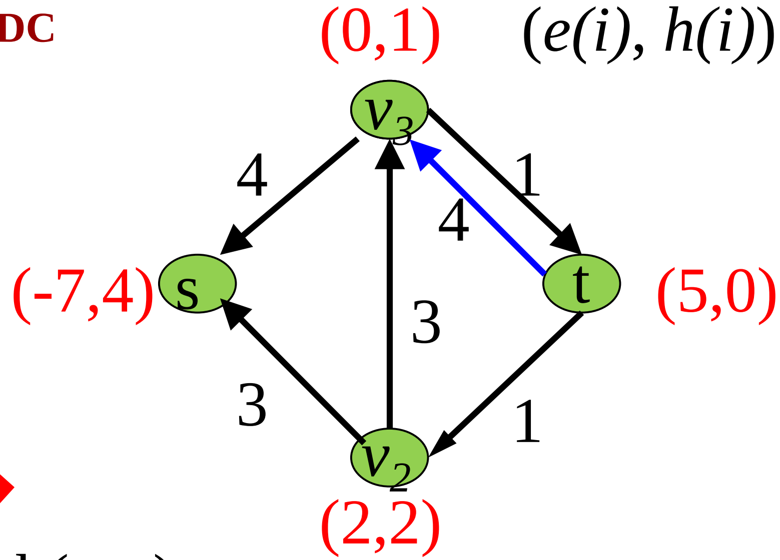
只要存在溢出结点，算法就会执行push或relable操作

当不存在溢出结点时，算法结束!

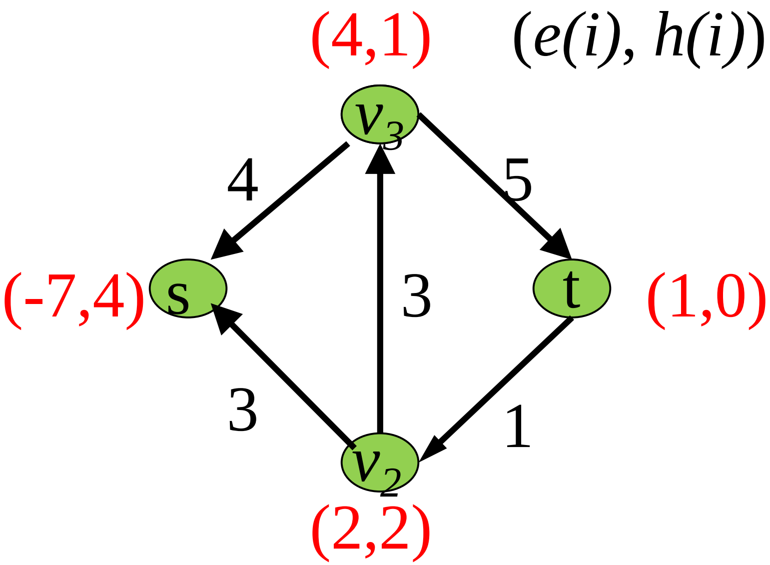
得到一个可行流,并且还是最大流!

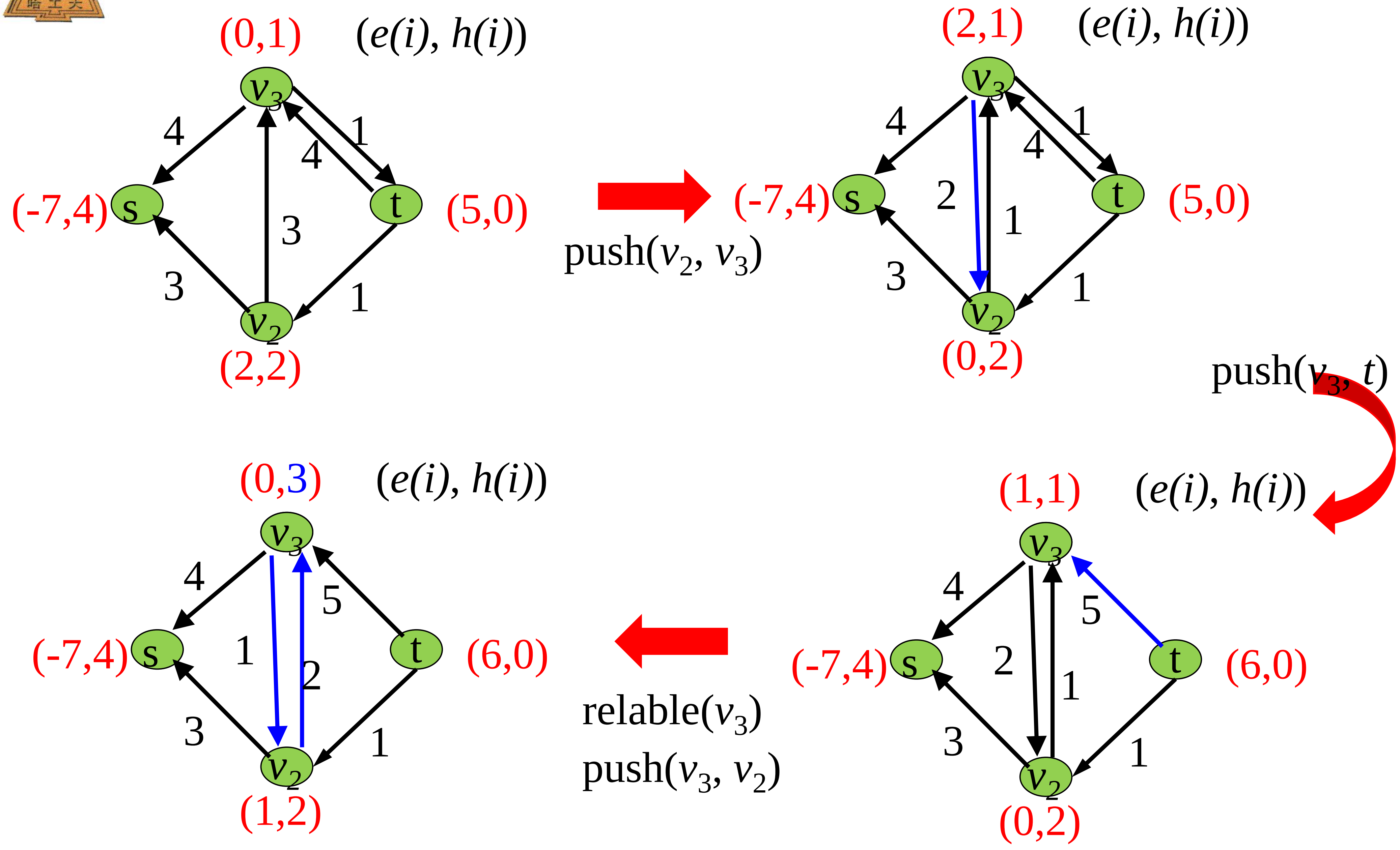

初始化预流
(4,1)
(e(i), h(i))
(-7,4)
(0,0)
(3,1)
push(v_2,t)
(4,1)
(e(i), h(i))
(-7,4)
(1,0)
(2,1)
Relabel(v_2)
(4,1)
(e(i), h(i))
(-7,4)
(5,0)
(2,2)

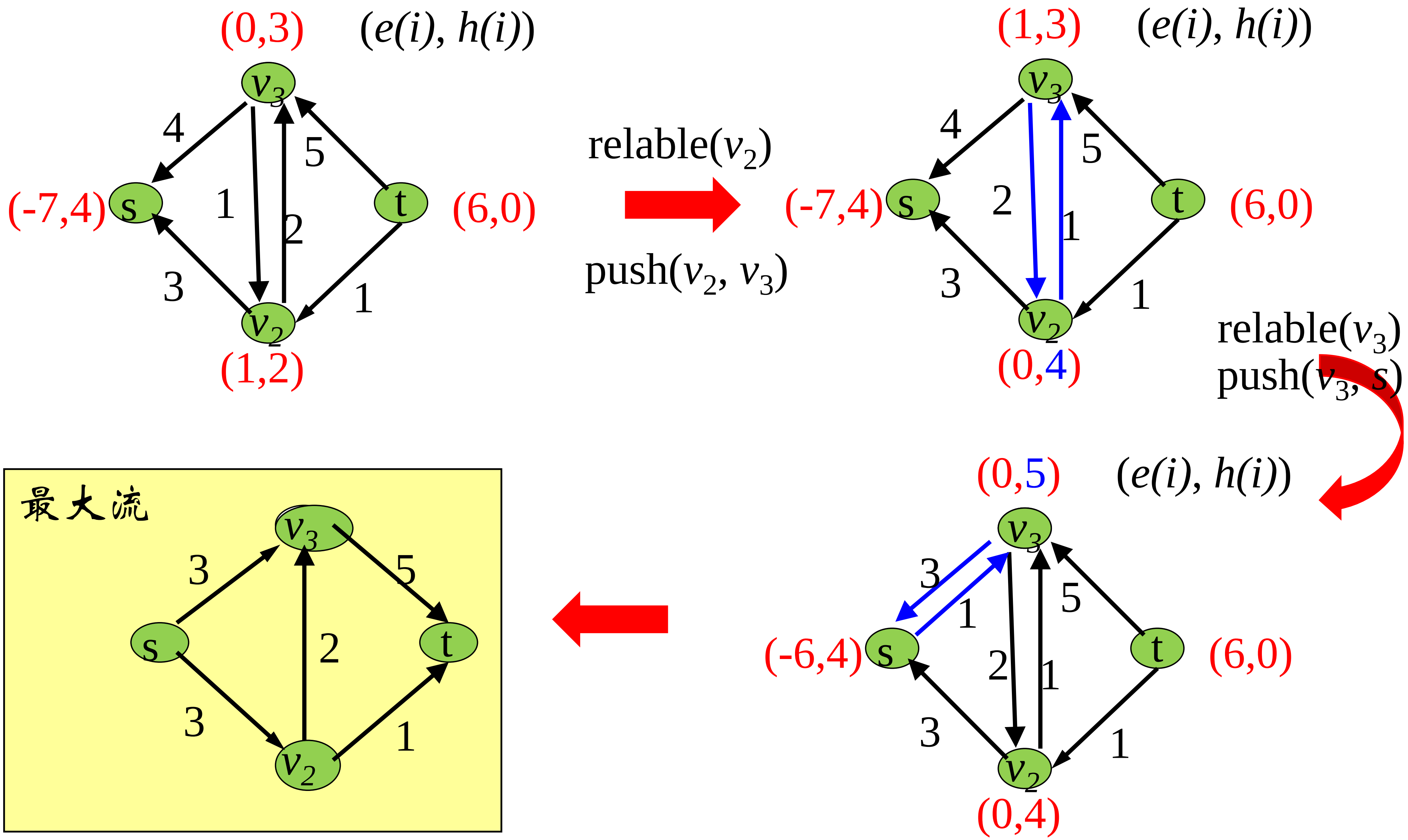
HIT
MDC



push(v_3,t)




(e(i), h(i))
(0,1)
(-7,4)
(5,0)
(2,2)
push(v2, v3)
(e(i), h(i))
(2,1)
(-7,4)
(5,0)
(0,2)
push(v3, t)
(e(i), h(i))
(1,1)
(-7,4)
(6,0)
(0,2)
relable(v3)
push(v3, v2)
(e(i), h(i))
(0,3)
(-7,4)
(6,0)
(1,2)

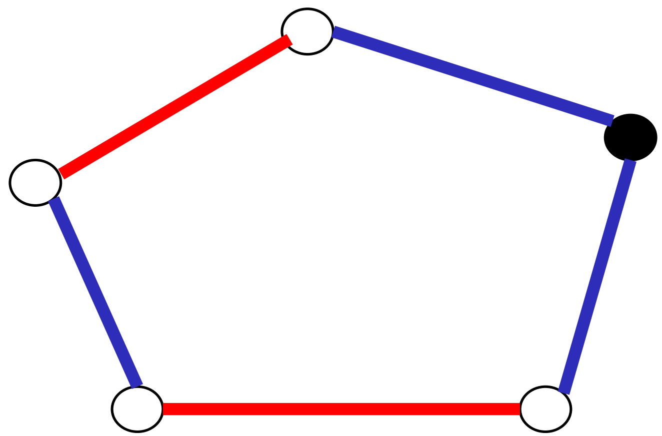

(0,3)
(e(i), h(i))
v3
4
5
1
2
(-7,4)
s
t
(6,0)
3
1
v2
(1,2)
relable(v2)
push(v2, v3)
(1,3)
(e(i), h(i))
v3
4
5
2
1
(-7,4)
s
t
(6,0)
3
1
v2
(0,4)
relable(v3)
push(v3, s)
(0,5)
(e(i), h(i))
v3
3
1
5
(-6,4)
s
2
1
t
(6,0)
3
1
v2
(0,4)
最大流
v3
3
5
s
2
t
3
1
v2

HIT
MDC

8.2 匹配问题

- 基本概念与应用
- 最大二分匹配

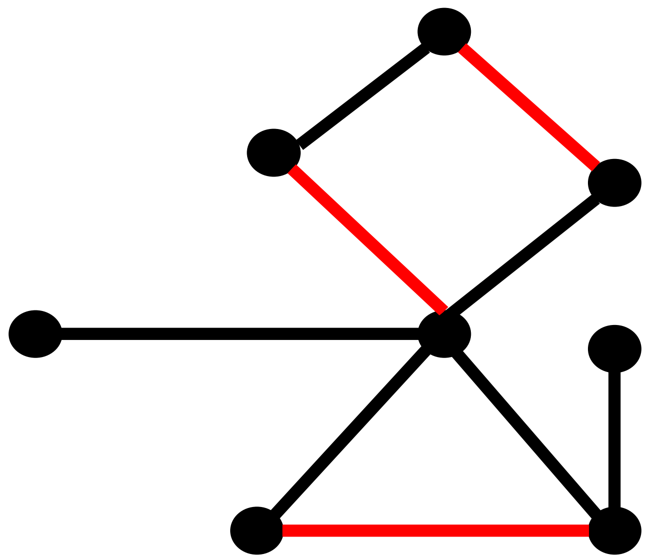
- 匹配(Matching)
  - 给定图$G(V,E)$，一个匹配M是由G的一组没有公共端点的不是圈的边构成的集合。
    - 对于所有的结点$v\in V$，子集M中最多有一条边与结点v相连
    - 与匹配M中边关联的那些结点是被M-浸润的(即饱和点，白色的点)，其余结点是M-未浸润的(即非饱和点，黑色的点)



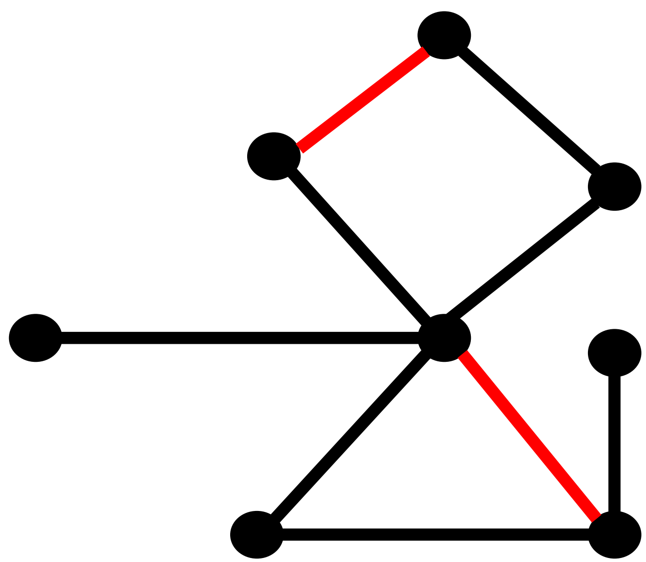
基本概念与应用

- 极大匹配：不能再通过添加边使其变大的匹配
  - 即：不存在 $e \in E$ 满足 $M \cup \{e\}$ 也是匹配
- 最大匹配：$|M|$最大的匹配
- 完美匹配：浸润了所有结点的匹配
  - 即 $|M|=n/2$

性质：完美匹配是最大匹配，反之不然

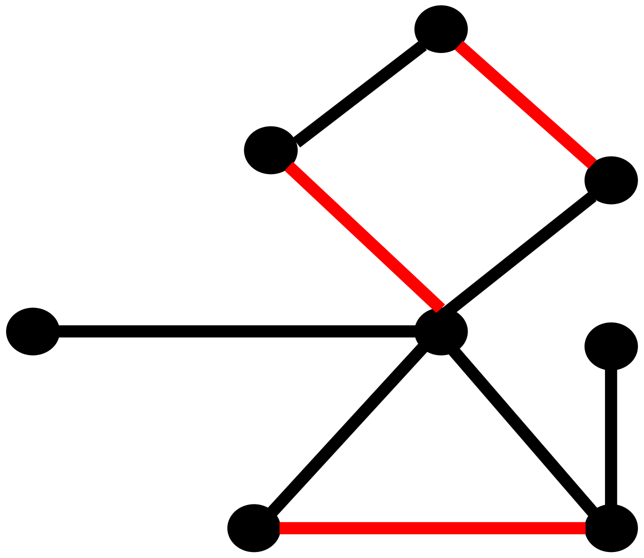


最大匹配

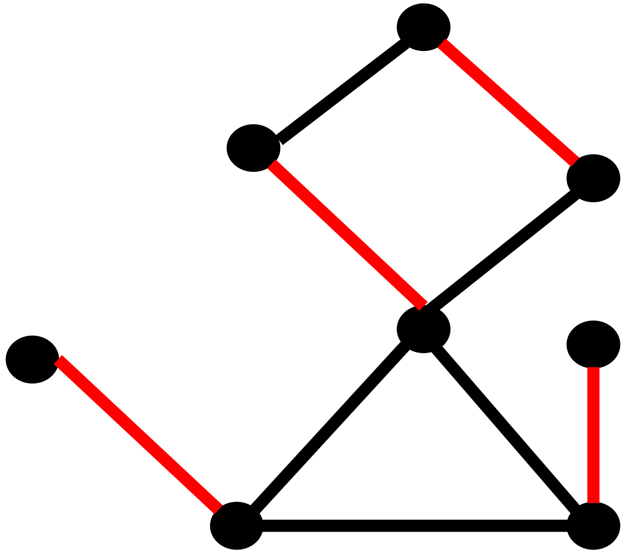


极大匹配

HIT
MDC



最大匹配
非完美匹配



最大匹配
完美匹配

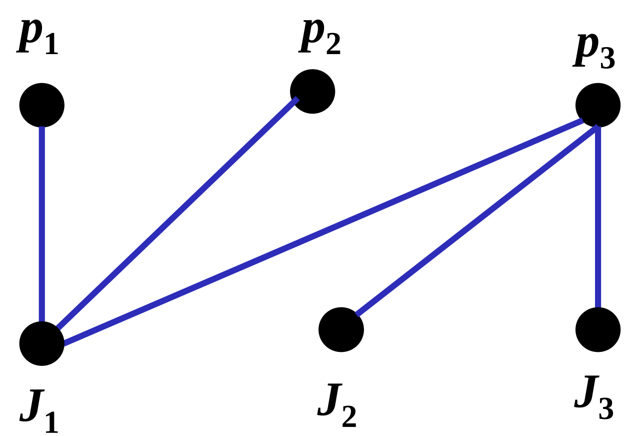
基本概念与应用

- 最大匹配问题
  - 输入：图$G(V,E)$
  - 输出：G的最大匹配M
- 应用
  - 人员指派
  - 教室指派
  - 任务安排
  - 赛程安排

工作分配

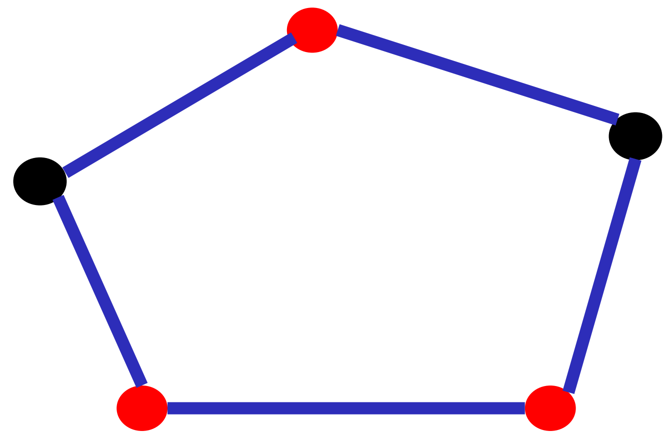
输入：n个人$p_1,...,p_n$, n项工作$J_1,...,J_n$,
第i个人胜任其中k项工作

输出：是否存在工作分配方案使得每个人完成1项自己胜任的工作



基本概念与应用

- 顶点覆盖
  - 图$G(V,E)$的一个顶点覆盖是指顶点集合$C \in V$，C包含每条边上的至少一个端点
    - C的所有顶点覆盖边集E
- 最小顶点覆盖
  - $|C|$最小顶点的覆盖
- 最小顶点覆盖问题
  - 输入：图$G(V,E)$
  - 输出：G的最小顶点的覆盖



目前，大规模图数据管理已经非常盛行
很多高效算法都以匹配算法或覆盖算法为基础！

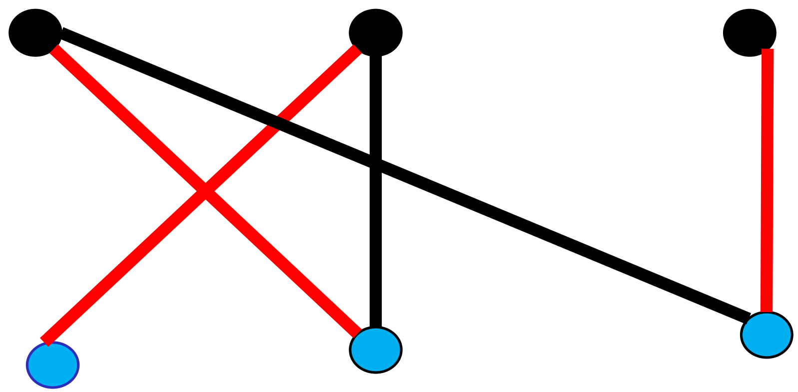
基本概念与应用

- 加权最大/小匹配问题
  - 每条边有一个代价，寻找具有最大/小代价的匹配

- 加权最小覆盖问题
  - 每个顶点有一个代价，寻找具有最小代价的覆盖

最大二分匹配问题

- 二分图(Bipartite Graph，又称二部图）
  - 图$G(V,E)$称为二分图，如果$V=L\cup R$，$L\cap R=\Phi$，E中所有边一定是有一个顶点属于集合L，另一个顶点属于集合R
- 最大二分匹配问题定义
  - 输入：二分图$G(V,E)$
  - 输出：G的最大匹配

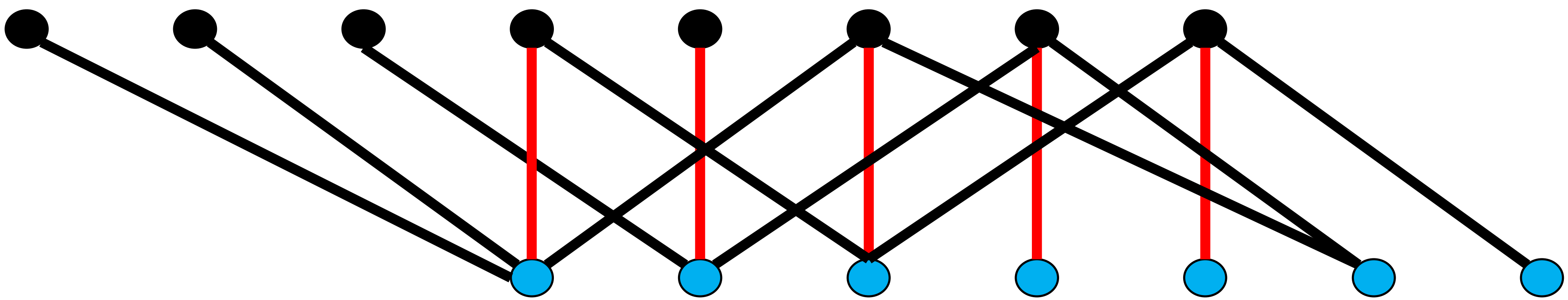


一个Min-Max关系

- 定理(Konig-Egervary)：如果G是一个二分图，则G中最大匹配的大小等于G的最小顶点覆盖的大小
- 这意味着二分图上的最大匹配可以这样求解
  - 初始化一个匹配M
  - 不断地增大M
  - M无法增大时，找出一个顶点覆盖C使得$|M|=|C|$
  - M是最大匹配，C是最小覆盖

求解最大二分匹配问题

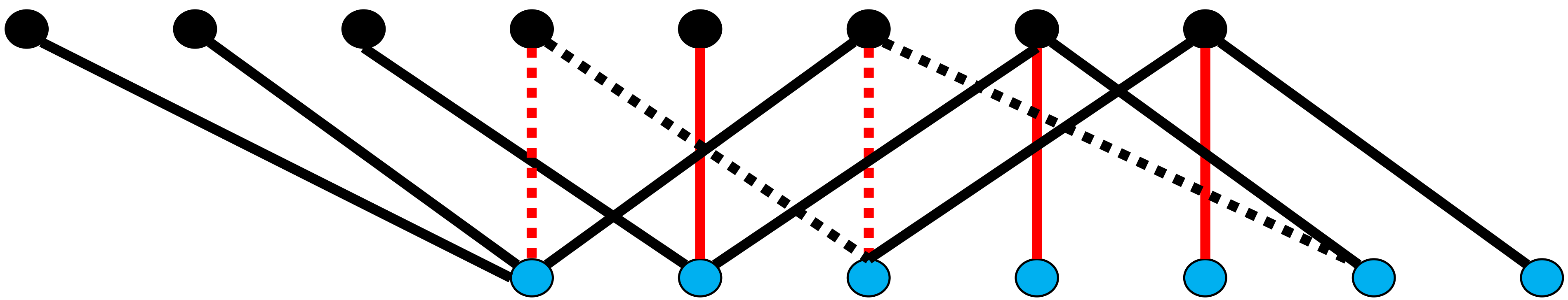
- M交错路径
  - 边交替出现在M和E-M的路径



M={图中红色边}

求解最大二分匹配问题

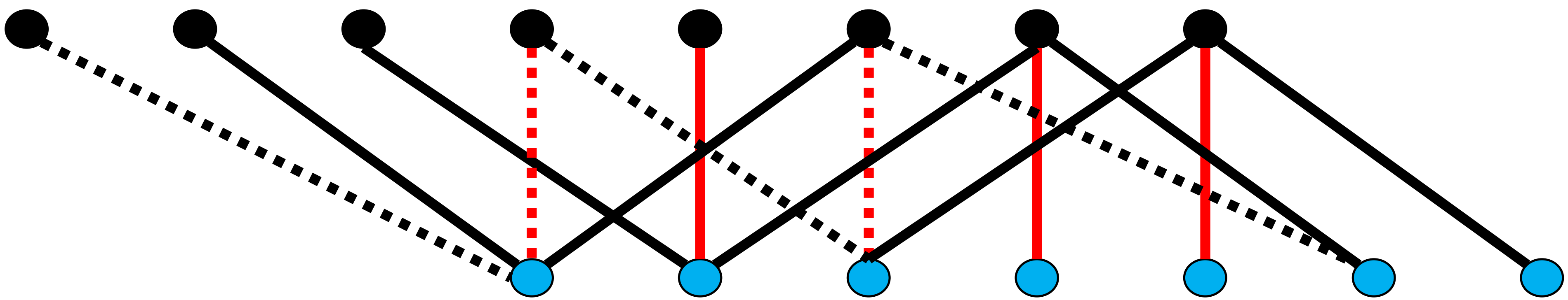
- M交错路径
  - 边交替出现在M和E-M的路径
- M增广路径：
  - 端点未被M浸润的交错路径



M={图中红色边}

求解最大二分匹配问题

- *M*交错路径
  - 边交替出现在*M*和*E-M*的路径
- *M*增广路径：
  - 端点未被*M*浸润的交错路径
  - *M*增广路径的长度必为奇数，第一条边和最后一条边不属于*M*



M={图中红色边}

求解最大二分匹配问题

1. $U \leftarrow V$中未被M浸润的所有顶点
2. While 存在从$x \in U$出发的M增广路径
   则增大M
   $U \leftarrow V$中未被M浸润的所有顶点
3. M是最大匹配

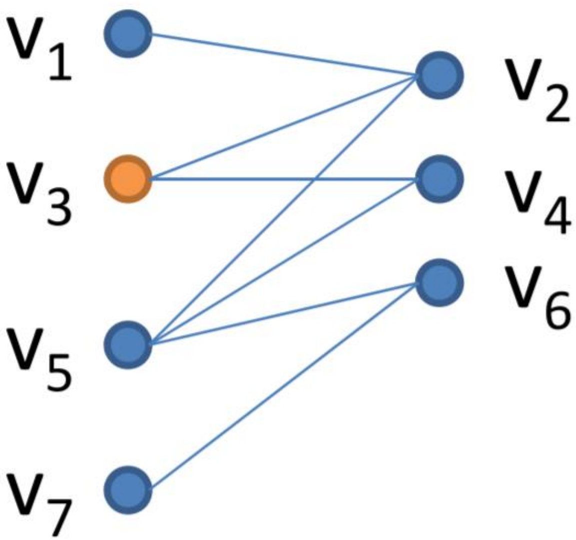
复杂度： $O(|V||E|)$

求解最大二分匹配问题

- 每轮搜索仅需从二部图特定的一侧顶点中开始(Why?)
  – 增广路的长度是奇数⇒ 起点和终点位于二部图的两侧 ⇒ 仅从任何一侧开始搜索都能确保找到
- 每轮搜索的起点?
  – 从一侧中(不妨左侧)每一个未被浸润的顶点开始
- 搜索哪些边?
  – 左侧到右侧：不在当前匹配中的边
  – 右侧到左侧：在当前匹配中的边
- 搜索的过程
  – 找到一个未被当前匹配浸润的顶点（起点除外）⇒ 找到一条增广路，替换得到更大的匹配 ⇒ 进入下一轮搜索
- 搜索的终止条件
  – 已搜索所有的点和边，仍未找到 ⇒ 无增广路 ⇒ 找到最大匹配

求解最大二分匹配问题

举例：



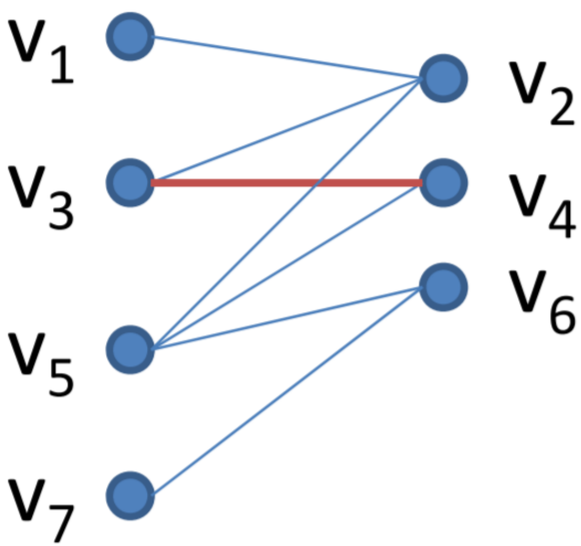
第一轮搜索开始

- 当前匹配：{}
- 未浸润的左侧顶点：$\{v_1, v_3, v_5, v_7\}$

从$\{v_1, v_3, v_5, v_7\}$中未搜索过的v_3开始搜索

求解最大二分匹配问题

举例：



沿不在当前匹配中且本轮未搜索过的(v_3, v_4)到达v_4 ⇒ v_4未浸润

找到增广路v_3v_4 ⇒ 替换进当前匹配中 ⇒ 本轮搜索结束

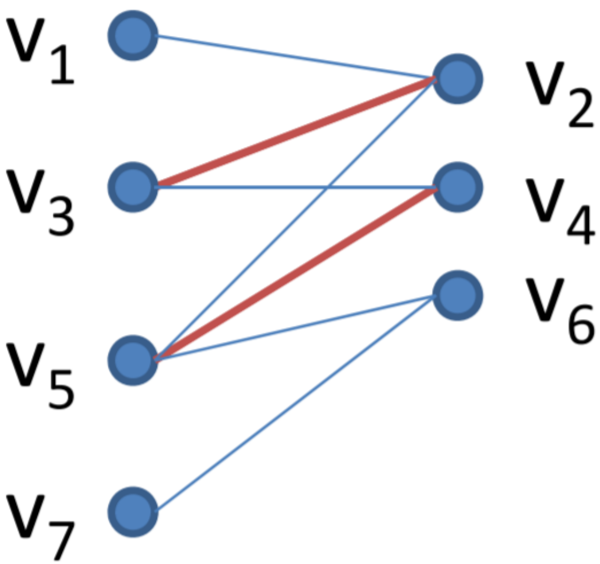
$M=\{(v_3,v_4)\}$

求解最大二分匹配问题

举例：

第二轮搜索开始

- 当前匹配：$M=\{(v_3,v_4)\}$
- 未浸润的左侧顶点：$\{v_1, v_5, v_7\}$



从$\{v_1, v_5, v_7\}$中未搜索过的v_5开始搜索

沿不在当前匹配中且本轮未搜索过的(v_5,v_4)，到达v_4(已被M浸润)

沿当前匹配中的(v_3, v_4)到达v_3

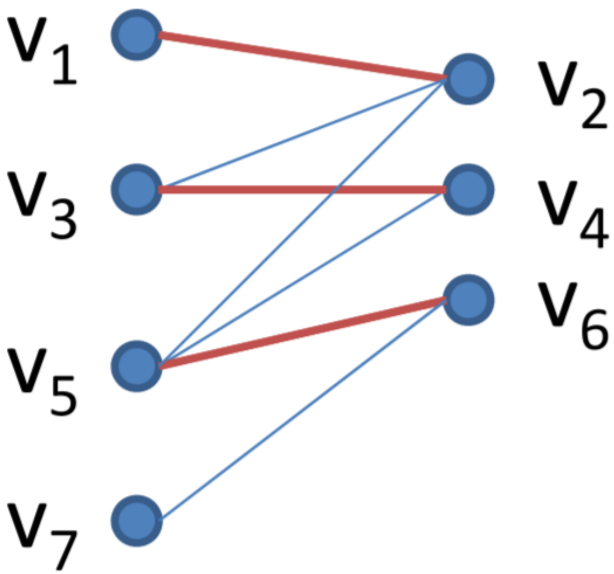
沿不在当前匹配中且本轮未搜索过的(v_3,v_2)，到达v_2(未被M浸润)

$M=\{(v_5,v_4), (v_3,v_2)\}$

找到增广路$v_5v_4v_3v_2$ ⇒ 替换进当前匹配中 ⇒ 本轮搜索结束

求解最大二分匹配问题

举例：



$M=\{(v_5,v_4), (v_3,v_2)\}$

第三轮搜索开始

- 当前匹配：$\{(v_3, v_2), (v_5, v_4)\}$
- 未浸润的左侧顶点：$\{v_1, v_7\}$

从$\{v_1, v_7\}$中未搜索过的v_1开始搜索

沿不在当前匹配中且本轮未搜索过的(v_1,v_2)，到达v_2(已被M浸润)

沿当前匹配中的(v_3, v_2)到达v_3

沿不在当前匹配中且本轮未搜索过的(v_3,v_4)，到达v_4(已被M浸润)

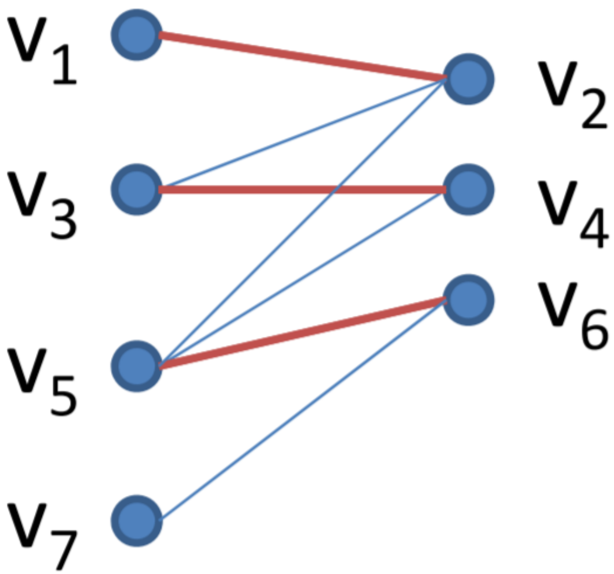
沿当前匹配中的(v_5, v_4)到达v_5

沿不在当前匹配中且本轮未搜索过的(v_5,v_6)，到达v_6(未被M浸润)

找到增广路$v_1v_2v_3v_4v_5v_6$ ⇒ 替换进当前匹配中 ⇒ 本轮搜索结束

求解最大二分匹配问题

举例：



第四轮搜索开始

- 当前匹配： $\{(v_1, v_2), (v_3, v_4), (v_5, v_6)\}$
- 未浸润的左侧顶点： $\{v_7\}$

……

$$M=\{(v_1,v_2), (v_3,v_4), (v_5,v_6)\}$$

求解最大二分匹配问题

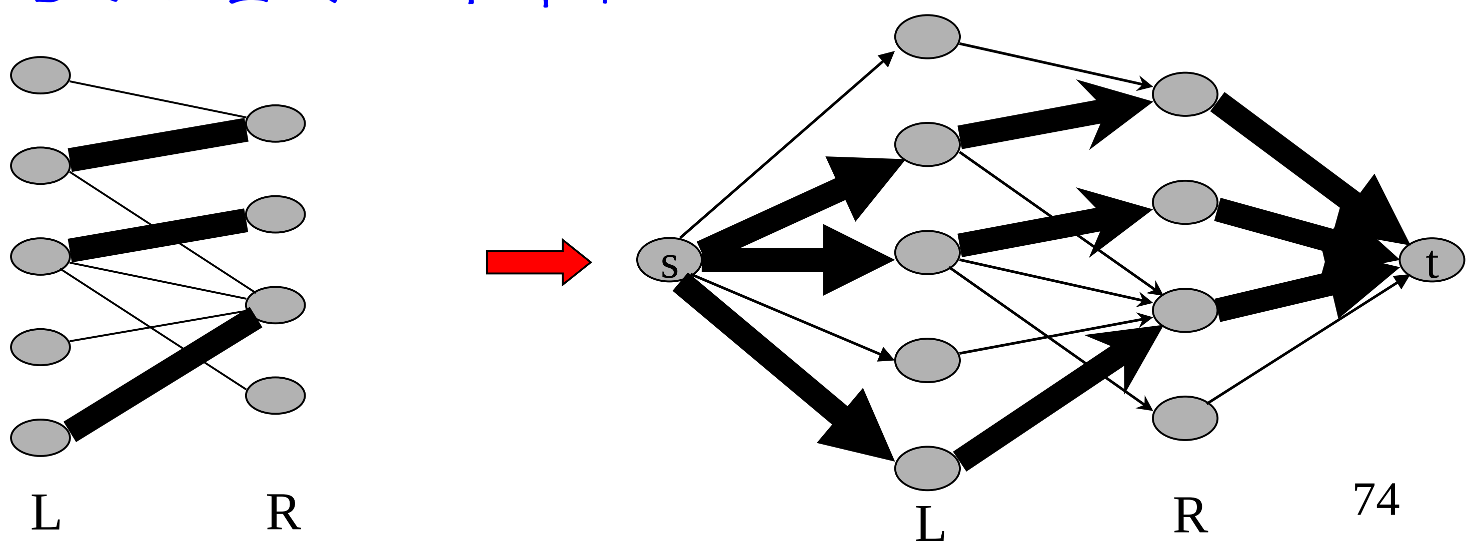
- 规约为最大流问题，利用最大流求解算法
- 二部图的对应流网络

给定二部图$G=(V, E)$, $V=L\cup R$, G对应流网络为$G'=(V', E')$

$V'=V\cup\{s, t\}$,

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

每条边的容量为一个单位



求解最大二分匹配问题

引理1. 设$G'=(V', E')$是$G=(V, E)$的对应流网络, 则$|E'|=\Theta(|E|)$.

证明.

由于V中每个节点v至少有一条连接v的边, $|E|\geq|V|/2$.

于是, $|E|\leq|E'|=|E|+|V|\leq 3|E|$, 即$|E'|=\Theta(|E|)$.

求解最大二分匹配问题

– 算法
1. 由G构造对应的G';
2. Ford-Fulkerson(G', s, t); //或任意其它最大流算法
3. 由G'的最大流构造G的最大匹配

$$M=\{ (u, v) \mid u\in L, v\in R, f(u, v)>0\}.$$

– 算法复杂性
- 第1步: $\Theta(|E|)$;
- 第2步: $O(|V||E|)$; //或其它
- 第3步: $O(|E|)$;
- 总时间: $O(|V||E|)$.

算法正确性证明

引理2. 设$G=(V, E)$是一个二部图, $V=L\cup R$, G'是G对应的流网络. M是G的一个匹配 iff G'中存在一个整数值流f, $|f|=|M|$.

证明.

$\Rightarrow$ 设M是G的匹配. 如下定义G'中的流f:

若$(u, v)\in M$, $f(s, u)=f(u, v)=f(v, t)=1$, $f(u, s)=f(v, u)=f(t, v)=-1$. 对于其他$(u, v)\in E'$, $f(u, v)=0$.

容易证明: f满足容量约束性、斜对称性、流守恒性.

往证$|f|=|M|$.

对于$\forall(u, v)\in M$, (u, v)对应一个1单位流$s\rightarrow u\rightarrow v\rightarrow t$.

由M的边导致的路径除s和t外节点不相交.

于是, 跨越划分$(L\cup\{s\}, R\cup\{t\})$的net flow等于$|M|$. 由前节的引理4, $|f|=|M|$.

$\Leftarrow$ 设f是G'的整数值流. 如下定义G中匹配M:

$$M=\{(u, v) \mid u\in L, v\in R, f(u, v)>0\}.$$

往证M是G中匹配.

对于$\forall u\in L$, u仅有一个容量为1的入边(s, u). 于是, 至多有一个单位正值流进入u, 且由流守恒性, 若有单位流进入u, 必有单位流离开u.

由于f的值是整数, 单位流只能经一条边进入或离开u. 于是, 单位流进入u iff 存在一个节点$v\in R$, 使$f(u, v)=1$.

同样的结论对$\forall v\in R$也成立.

于是, M是G的一个匹配.

往证$|f|=|M|$.

对于每个匹配节点$u\in L$, $f(s, u)=1$; 对于$\forall(u, v)\in E-M$, $f(u, v)=0$. $|M|=f(L, R)=f(L, V')-f(L, L)-f(L, s)-f(L, t)$.

由流守恒性, $f(L, V')=0$; 由斜对称性$-f(L, t)=f(s, L)$; 由于无L到t的边, $f(L, t)=0$; $f(L, L)=0$. 于是

$$|M|=f(s, L)=f(s, V')=|f|.$$

HIT
MDC

定理1. 若容量函数c仅取整数值, 则由Ford-Fulkerson方法产生的最大流f具有下列性质:

(1). | f |是整数;

(2). 对于所有节点u和v, f(u, v)是整数.

证明. 对于循环数做数学归纳证明.

推论1. 设M是G中最大匹配, f是G对应的G’的最大流, 则|M|=| f |.

证明. 设M是G的最大匹配, f不是G’的最大流.

必存在G’的最大流f’, | f’|>| f |.

由于G’的容量值是整数, 由定理1, 可以设f’是整数值流. 于是, f’对应于G的一个最大流M’, |M’|=| f’|>| f |=|M|, 与M是最大流矛盾.

类似可证, 若f是G’的最大流, 它对应的匹配M必为G的最大匹配.