图与网络优化

本章主要内容：

- 图论的发展历史
- 图的基本概念与模型
- 树与图的最小树
- 最短路问题
- 网络的最大流
- 最小费用最大流问题

图论的发展历史

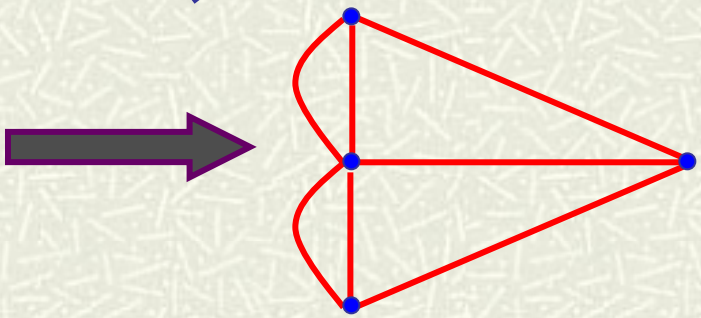
图论是专门研究图的理论的一门数学分支，属离散数学范畴，与运筹学有交叉，近 300 年历史，发展历史大体分为三个阶段：

1. 十八世纪中叶到十九世纪中叶：多数问题产生于游戏，最具代表性的是所谓的 Euler 七桥一笔画问题 (1736年)。
2. 十九世纪中叶到二十世纪中叶：图论问题大量出现，如 Hamilton 问题，地图染色的四色问题、可平面性问题等。期间也出现用图解决实际问题，如 Cayley 把树应用于化学领域，Kirchhoff 用树去研究电网络等.
3. 二十世纪中叶以后：由生产管理、军事、交通、运输、计算机网络等方面提出实际问题，以及大型计算机使大规模问题的求解成为可能，特别是以 Ford 和 Fulkerson 建立的网络流理论，与线性规划、动态规划等优化理论和方法相互渗透，促进了图论对实际问题的应用。

哥尼斯堡七桥问题

德国哥尼斯堡（现名加里宁格勒）城中有一条河叫普雷格尔河，它将城分成两部分，河中有两个小岛. 十八世纪时，河两边及小岛之间共有七座桥，当时人们提出这样的问题：有没有办法从某处出发，经过各桥一次且仅一次最后回到原地?

Königsberg桥对应的图

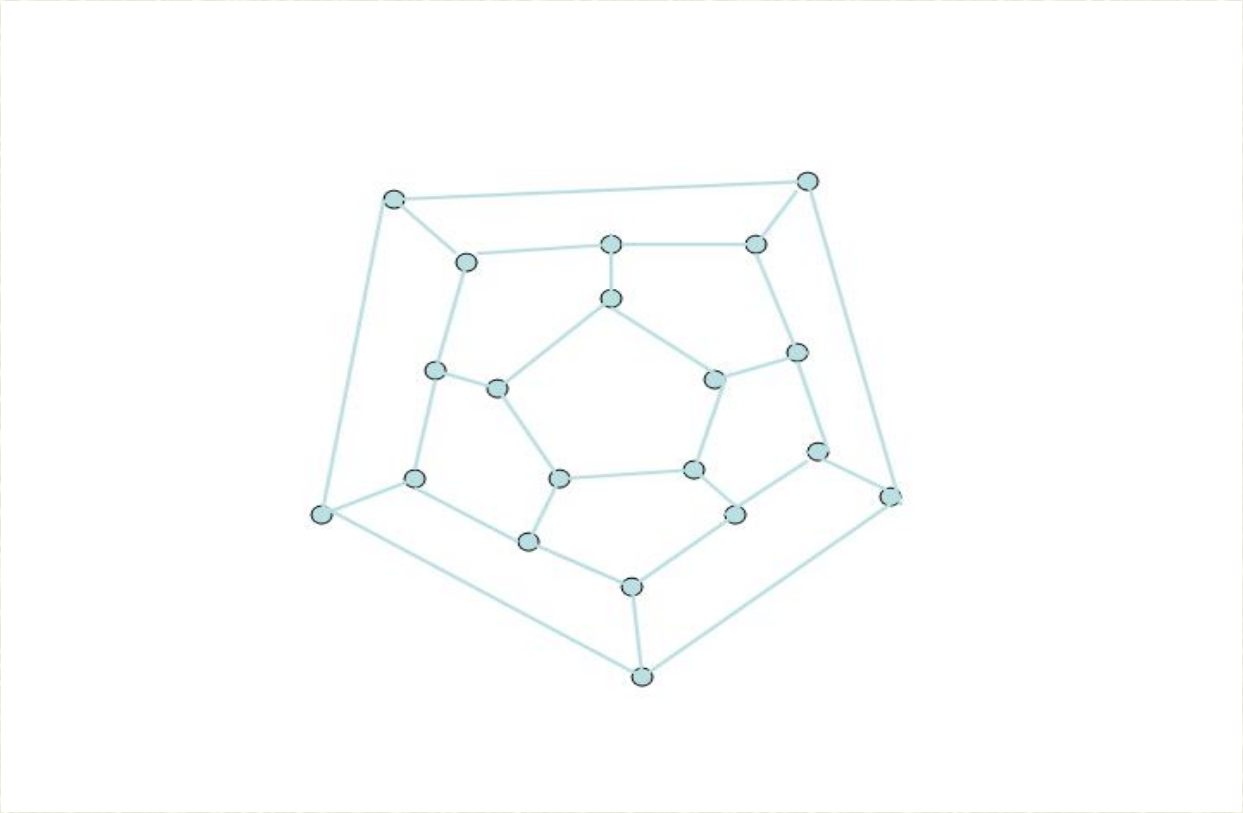


哥尼斯堡七桥问题

1736 年瑞士数学家 Euler 发表 “根据几何位置的解题方法” 一文，有效地解决了此类问题，是有记载的第一篇图论论文。此类问题也称 Euler 回路问题（经过每条边一次且仅一次的回路），Euler被认为是图论的创始人

Hamilton 回路问题

哈密尔顿 (Hamilton) 回路是 1857 年英国数学家哈密尔顿提出：给出一个正 12 面体图形，共有 20 个顶点表示 20 个城市，要求从某个城市出发沿着棱线寻找一条经过每个城市一次而且仅一次，最后回到原处的周游世界线路（并不要求经过每条边）。



图论

在实际的生产和生活中，人们为了反映事物之间的关系，常常在纸上用点和线来画出各式各样的示意图。

1. 如城市间的交通干线图、体育比赛中参赛各方的的胜负图等。
2. 图论中常用点和点之间的线所构成的图，反映实际生产和生活中的某些特定对象之间的特定关系。一般来说，通常用点表示研究对象、用点与点之间的线表示研究对象之间的特定关系。
3. 在一般情况下，图中的相对位置如何，点与点之间线的长短曲直，对于反映研究对象之间的关系，显的并不重要。　因此，图论中的图与几何图，工程图等本质上是不同的。

图的基本概念与模型

图论中图是由点和边构成，可以反映一些对象之间的关系。一般情况下图中点的相对位置如何、点与点之间联线的长短曲直，对于反映对象之间的关系并不是重要的。

图的定义:

若用点表示研究的对象，用边表示这些对象之间的联系，则图G可以定义为点和边的集合，记作:

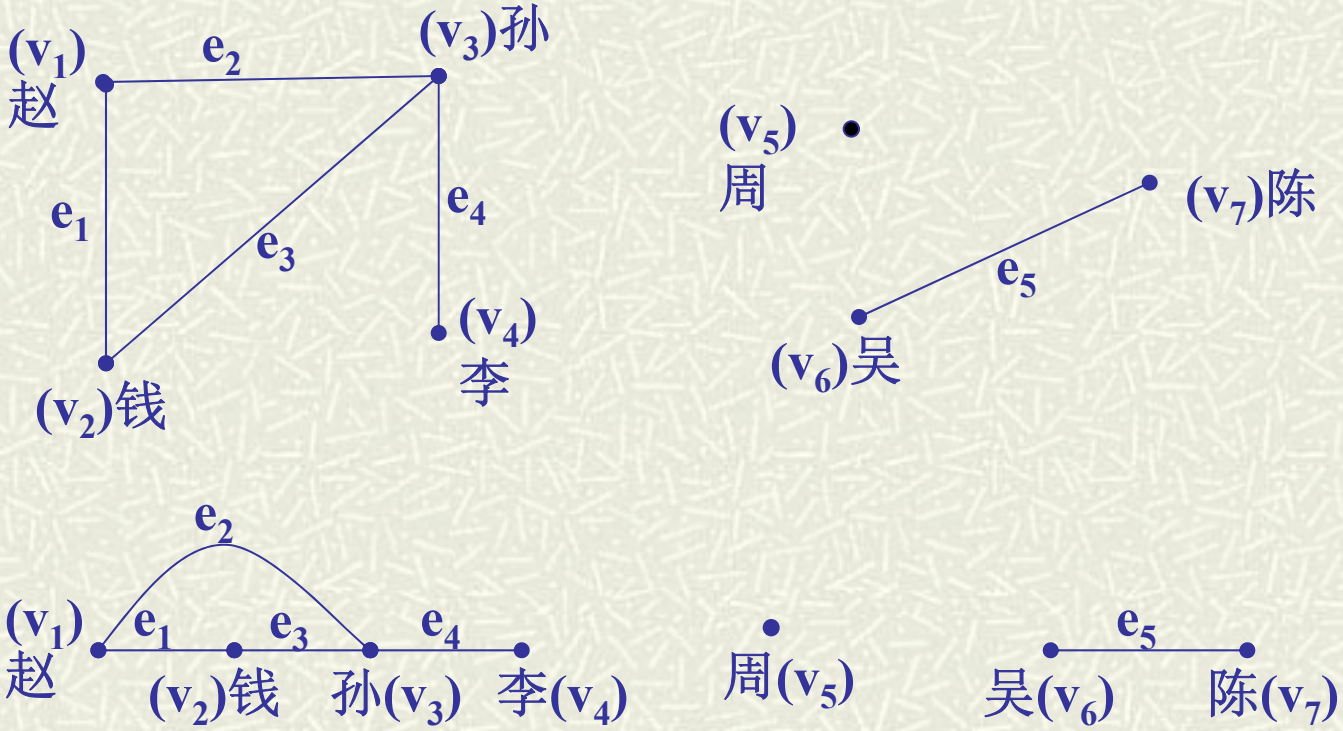
$$G = \{V, E\}$$

其中: V——点集　E——边集

※ 图G区别于几何学中的图。这里只关心图中有多少个点以及哪些点之间有连线。

图的基本概念与模型

例如：在一个人群中，对相互认识这个关系我们可以用图来表示。



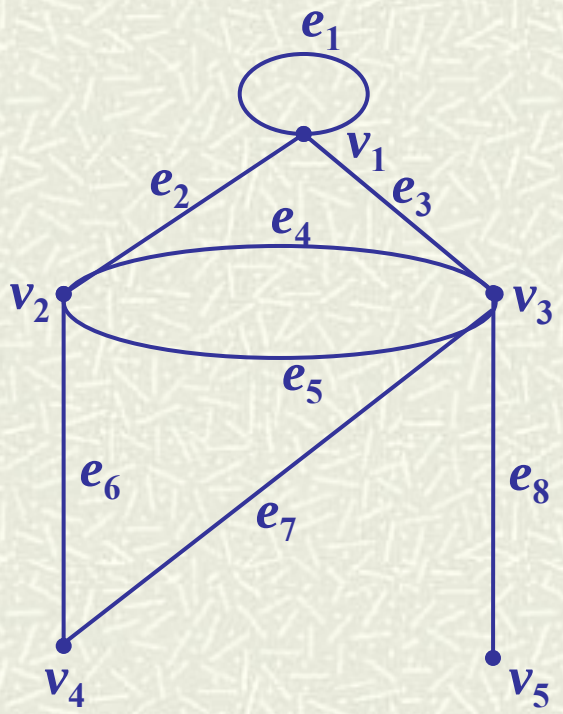
可见图论中的图与几何图、工程图是不一样的。

图的基本概念与模型

定义: 图中的点用v表示，边用e表示。对每条边可用它所连接的点表示，记作：$e_1=[v_1,v_1]$; $e_2=[v_1,v_2]$;

- 端点, 关联边, 相邻

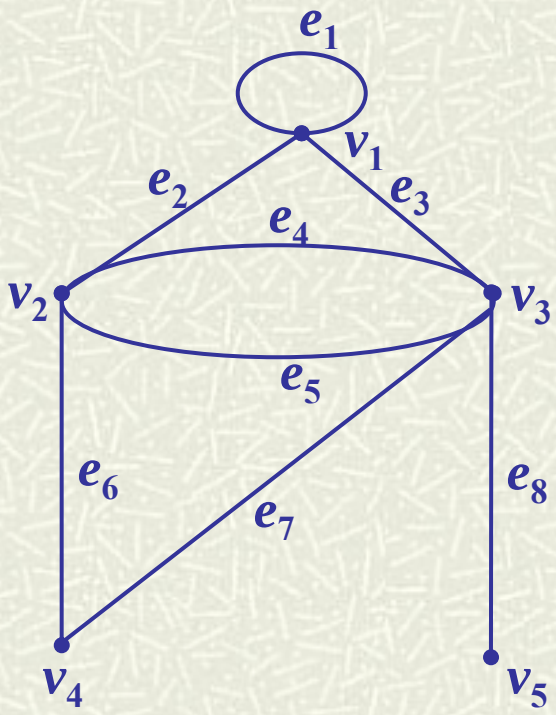
若有边e可表示为$e=[v_i,v_j]$，称v_i和v_j是边e的端点，反之称边e为点v_i或v_j的关联边。若点v_i与 v_j与同一条边关联，称点v_i和v_j相邻；若边e_i和e_j具有公共的端点，称边e_i和e_j相邻。



图的基本概念与模型

环, 多重边, 简单图

如果边e的两个端点相重，称该边为环。如右图中边e_1为环。如果两个点之间多于一条，称为多重边，如右图中的e_4和e_5，对无环、无多重边的图称作简单图。



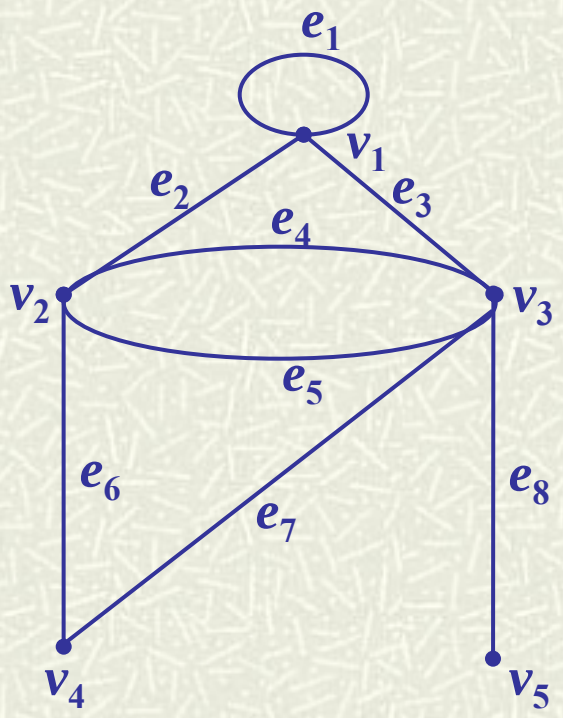
图的基本概念与模型

次，奇点，偶点，孤立点

与某一个点v_i相关联的边的数目称为点v_i的次（也叫做度），记作$d(v_i)$。

右图中$d(v_1) = 4$, $d(v_3)=5$, $d(v_5)=1$。

次为奇数的点称作奇点，次为偶数的点称作偶点，次为1的点称为悬挂点，次为0的点称作孤立点。



图的次: 一个图的次等于各点的次之和。

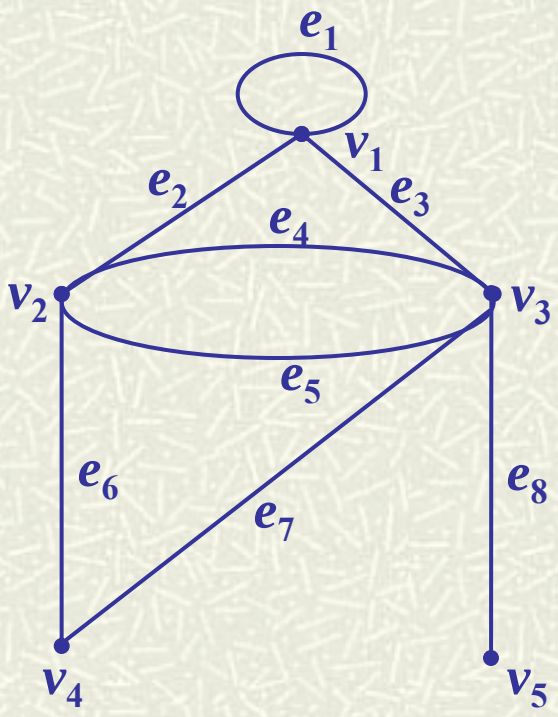
图的基本概念与模型

链，圈，连通图

图中某些点和边的交替序列，若其中各边互不相同，且对任意v_{t-1}和v_t均相邻称为链。用μ表示：

$$\mu = \{v_0, e_1, v_1, \Lambda\ , e_k, v_k\}$$

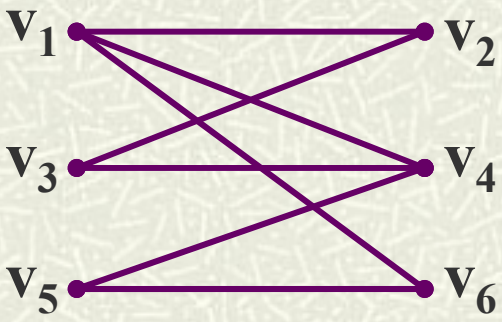
起点与终点重合的链称作圈。如果每一对顶点之间至少存在一条链，称这样的图为连通图，否则称图不连通。



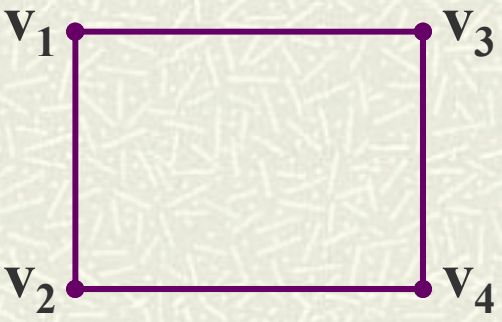
图的基本概念与模型

二部图（偶图）

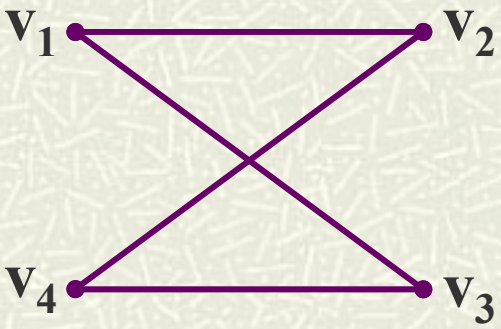
图 G = (V,E) 的点集V可以分为两各非空子集X，Y，集X∪Y=V，X∩Y=Ø, 使得同一集合中任意两个顶点均不相邻，称这样的图为偶图。



(a)



(b)



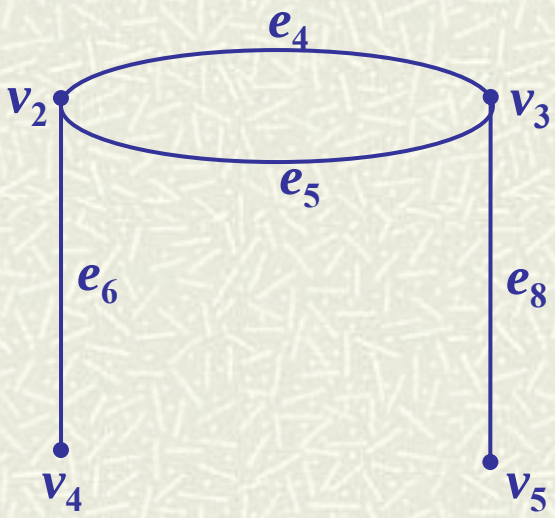
(c)

(a)明显为二部图，(b)也是二部图，但不明显，改画为(c)时可以清楚看出。

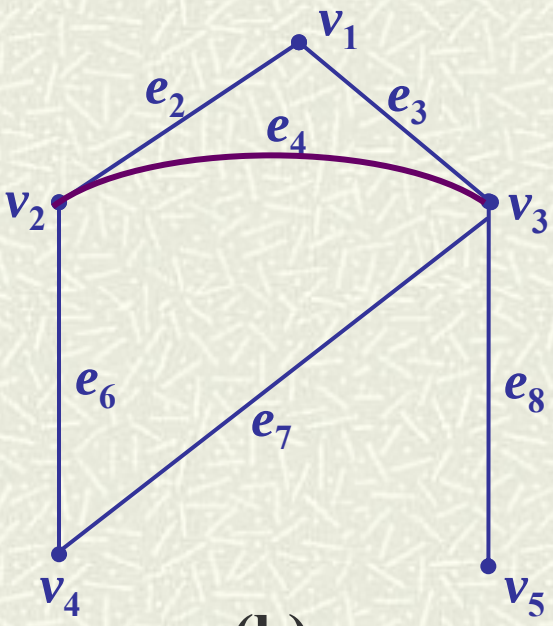
图的基本概念与模型

子图，部分图(支撑子图)

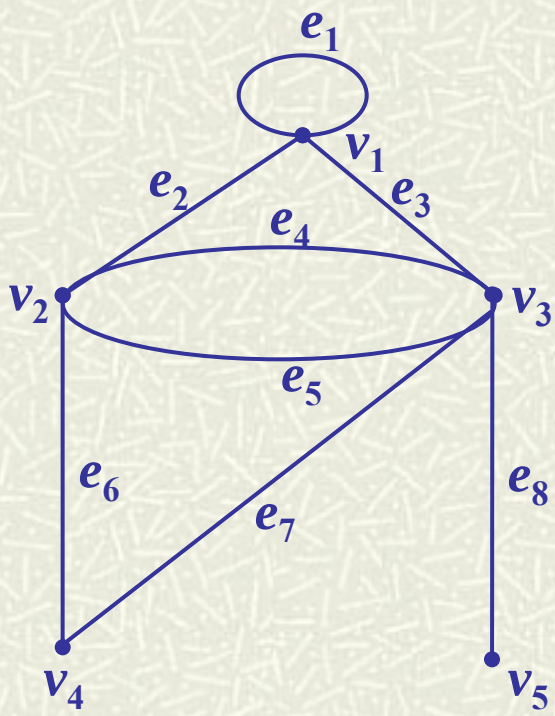
图$G_1=(V_1, E_1)$和图$G_2=(V_2, E_2)$. 如果有$V_1 \subseteq V_2$和$E_1 \subseteq E_2$称G_1是G_2的一个子图。若有$V_1=V_2$，$E_1 \subseteq E_2$，则称G_1是G_2的一个部分图(支撑子图)。



(a)



(b)



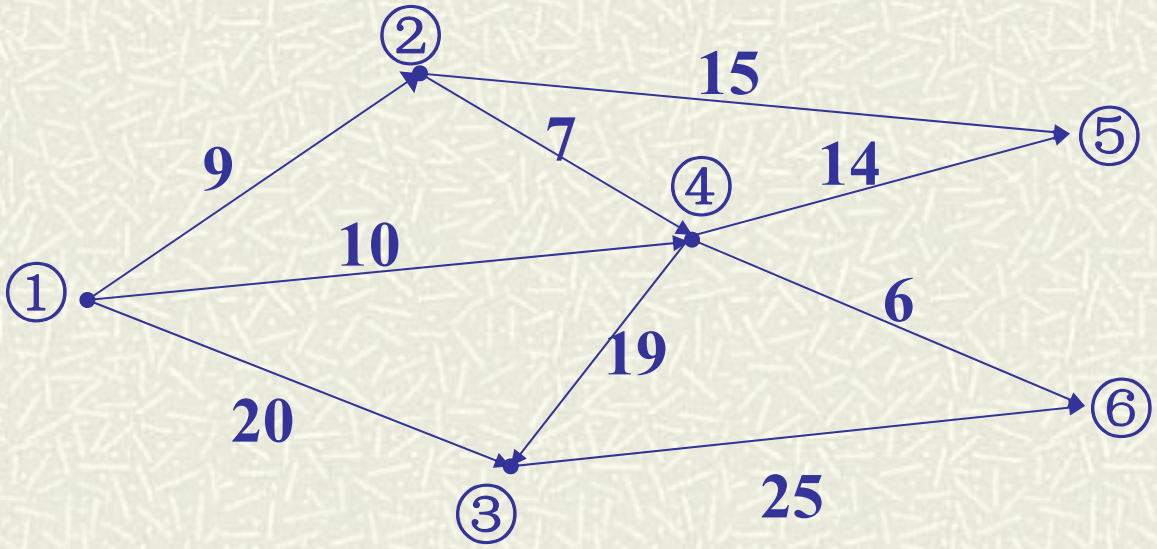
(G图)

图的基本概念与模型

网络（赋权图）

设图G＝（V，E），对G的每一条边(v_i, v_j)相应赋予数量指标w_{ij}，w_{ij}称为边(v_i, v_j)的权，赋予权的图G称为网络(或赋权图）。

权可以代表距离、费用、通过能力(容量)等等。端点无序的赋权图称为无向网络，端点有序的赋权图称为有向网络。



图的基本概念与模型

出次与入次 (出度、入度)

有向图中，以v_i为始点的边数称为点v_i的出次，用$d^+(v_i)$表示；以v_i为终点的边数称为点v_i 的入次，用表示$d^-(v_i)$ ； v_i 点的出次和入次之和就是该点的次。

※ 有向图中，所有顶点的入次之和等于所有顶点的出次之和。

图的基本概念与模型

图的模型应用

例6.1 有甲, 乙, 丙, 丁, 戊, 己6名运动员报名参加A,B,C,D,E,F 6个项目的比赛。下表中打√的是各运动员报名参加的比赛项目。问6个项目的比赛顺序应如何安排，做到每名运动员都不连续地参加两项比赛。

	A	B	C	D	E	F
甲	√			√		
乙	√	√		√		
丙			√		√	
丁	√				√	
戊	√	√			√	
己			√	√		√

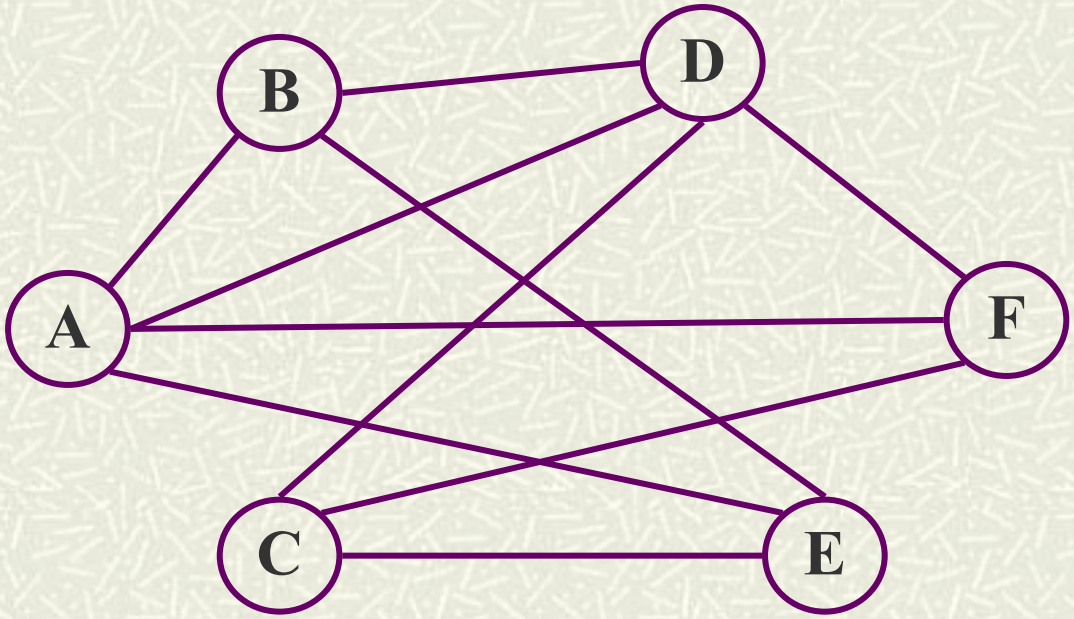
图的基本概念与模型

解：用图来建模。把比赛项目作为研究对象，用点表示。如果2个项目有同一名运动员参加，在代表这两个项目的点之间连一条线，可得下图。

在图中找到一个点序列，使得依次排列的两点不相邻，即能满足要求。如：

1) A,C,B,F,E,D

2) D,E,F,B,C,A



此图不对，AF间无边

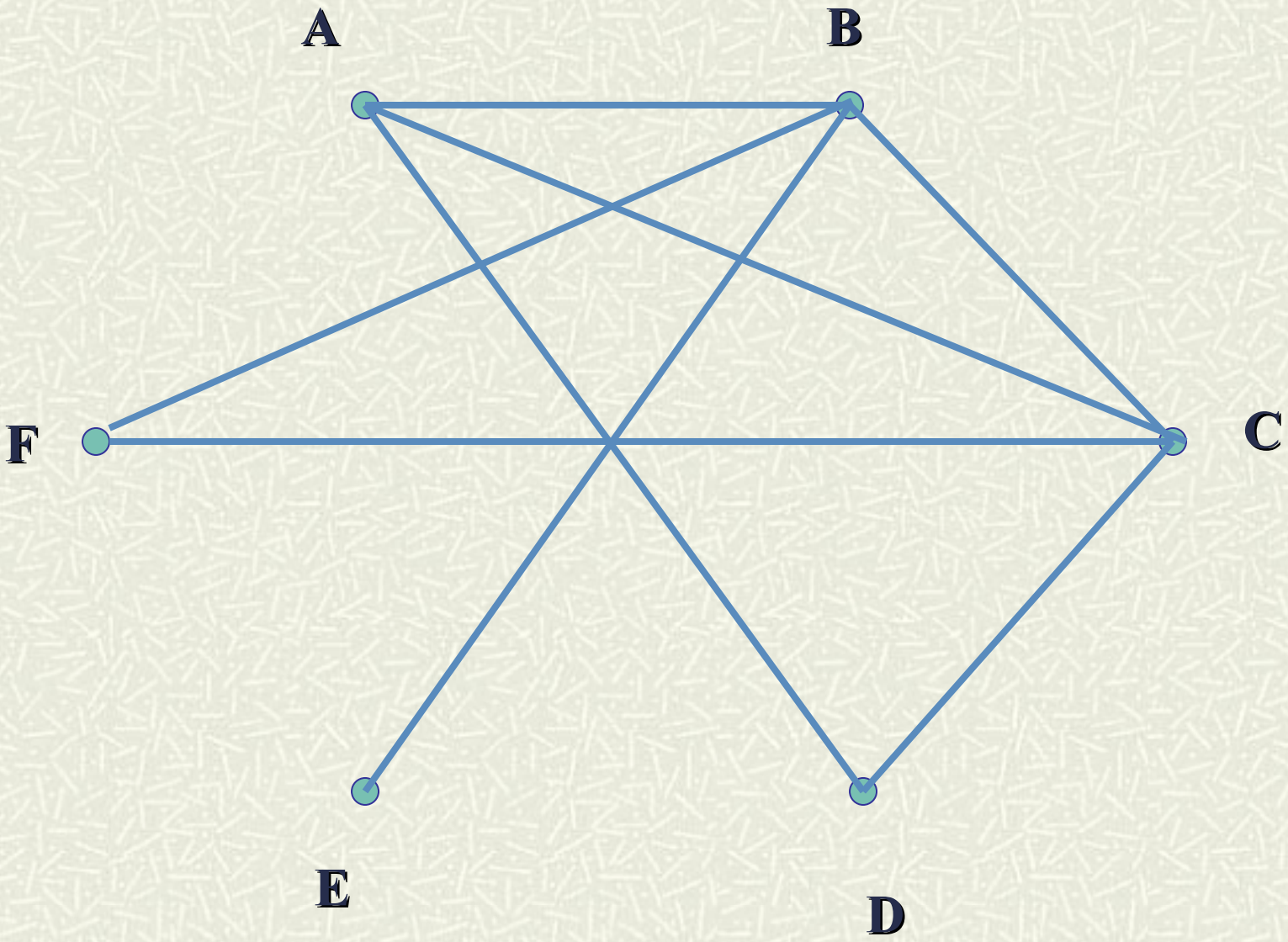
思考题

一个班级的学生共计选修A、B、C、D、E、F六门课程，其中一部分人同时选修D、C、A，一部分人同时选修B、C、F，一部分人同时选修B、E，还有一部分人同时选修A、B，期终考试要求每天考一门课，六天内考完，为了减轻学生负担，要求每人都不会连续两天参加考试，试设计一个考试日程表。

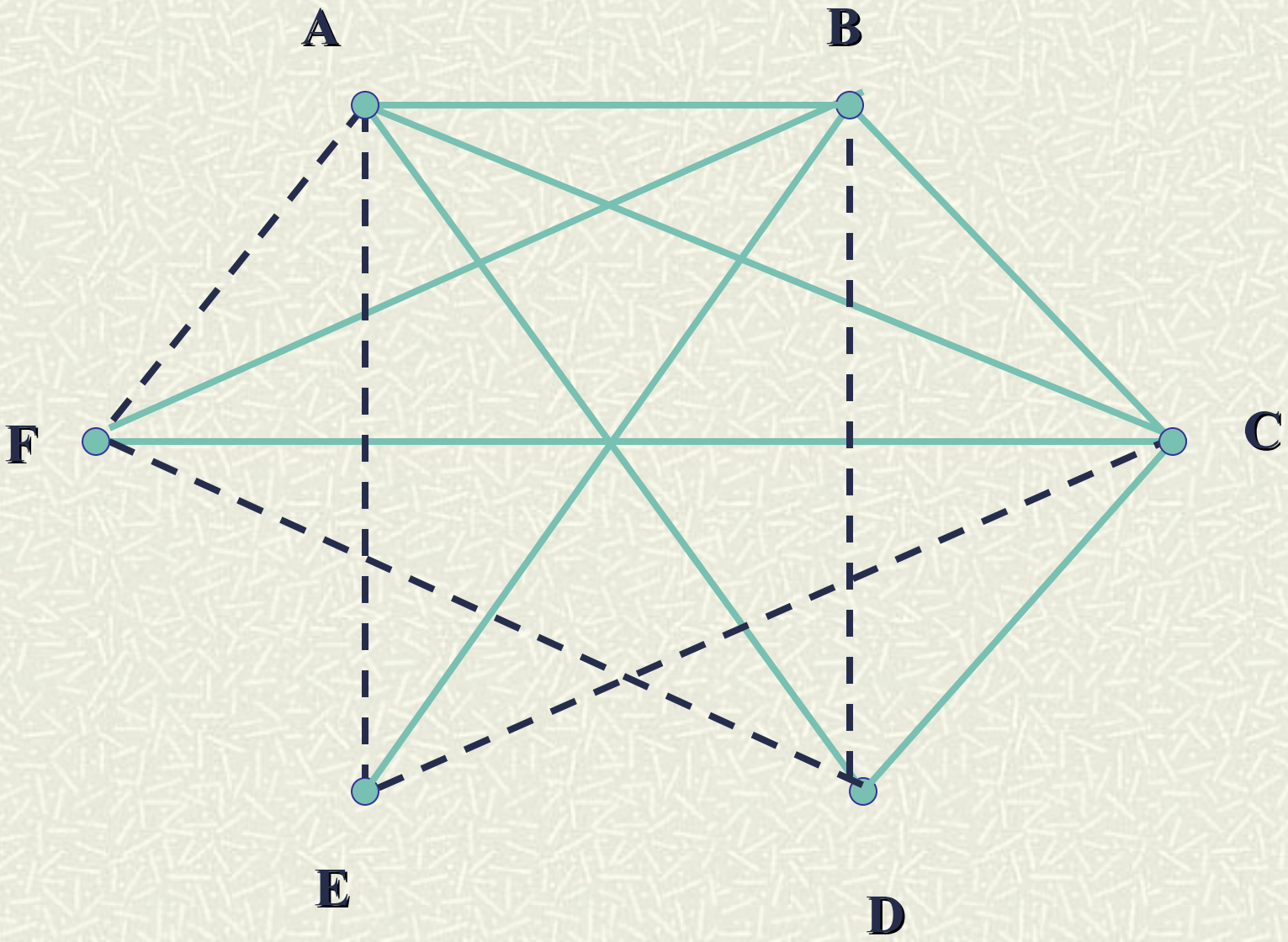
思考题解答:

以每门课程为一个顶点，共同被选修的课程之间用边相连，得图，按题意，相邻顶点对应课程不能连续考试，不相邻顶点对应课程允许连续考试，因此，作图的补图，问题是在图中寻找一条哈密顿道路，如**C—E—A—F—D—B**，就是一个符合要求的考试课程表。

图的基本概念与模型



图的基本概念与模型



图的基本性质：

定理1 任何图中，顶点次数之和等于所有边数的2倍。

证明：由于每条边必与两个顶点关联，在计算点的次时，每条边均被计算了两次，所以顶点次数的总和等于边数的2倍。

定理2 任何图中，次为奇数的顶点必为偶数个。

证明：设V1和V2分别为图G中奇点与偶点的集合。由定理1可得：

$$\sum_{v\in V_1} d(v) + \sum_{v\in V_2} d(v) = \sum_{v\in V} d(v) = 2m$$

2m为偶数，且偶点的次之和$\sum_{v\in V_2} d(v)$也为偶数，所以$\sum_{v\in V_1} d(v)$必为偶数，即奇数点的个数必为偶数。

基本概念

1. 图是由非空点集 V= {vi: i = 1,2,⋯,m} 和 V 中元素构成的点对的集合 E={ej: j = 1,2,⋯,n} 所构成的二元组，记为G=(V, E);
2. 若 E 中点对无序，则称 G 为无向图；否则，称为有向图. 有向图也记为 D=(V, A);
3. V 中的元素叫做顶点；对无向图，E 中的元素叫做边；对有向图，E 中的元素叫做弧;
4. 图 G 的顶点数与边数分别记为 p(G) 与 q(G).
5. 连接顶点 u 与v 的边记为 [u, v] 或 [v, u]. 顶点 u 到顶点v 的弧记为 (u, v), u 为起点，v 为终点;
6. 一条边连接两个顶点，称此两顶点相邻，并称其为该边的两个端点;
7. 若一条边连接的两个顶点是相同的，则称该边为环;
8. 若两顶点间多于一条边，称它们为多重边 (有向图中两顶点间有不同方向的两条边，不是多重边).

基本概念

1. 简单图: 不含环和多重边的图;
2. 多重图: 不含环，含多重边的图;
3. 以 v 为端点的边数叫做点 v 的 度，记作 d(v);
4. 度为 1 的点称为悬挂点，连接悬挂点的边称为悬挂边;
5. 度为零的点称为孤立点，度为奇数的点为奇点，度为偶数的点为偶点.

基本概念

1. 链: 无向图的点边交错序列 (v1,e1,v2,e2,...,v{k-1},e{k-1}, vk) 满足 ei = [vi, v{i+1}], i=1,2,...,k-1, 则称其为联结起点 v1 和终点 vk 的一条 链;
2. 简单链: 链中所含的边均不相同 (点可能相同);
3. 初等链: 链中所含的点均不相同 (边一定不相同，因此初等链一定是简单链);
4. 圈: 若 v1 = vk, 则称该链为圈;
5. 简单圈: 圈中所含的边均不相同 (点可能相同);
6. 初等圈: 圈中所含的点均不相同 (边一定不相同，因此初等圈一定是简单圈);

如无说明，仅考虑初等链与初等圈.

基本概念

连通图：图中任意两点之间均至少有一条链; 否则称其为不连通图；若 G 不连通，G 的每一个连通部分称为 G 的连通分图;

设有图G=(V, E) 与G'=(V', E').

子图：若 V'\subseteq V, E'\subseteq E, 且E' 中的边仅与 V' 中的顶点相关联，则称G' 是 G 的子图;

支撑子图： 若 V'=V, 则 G' 是 G 的生成子图 或 支撑子图;

导出子图：若 V' \subseteq V, E'= { [u,v]\in E: u,v \in V'}, 称 G' 是 G 中由 V' 导出的子图.

基本概念

基础图：有向图 D = (V,A) 中去除所有边的方向得到的图 G = (V,E) 称为其基础图;

路：设 u,v\in V, G 为D 的基础图。若G 中有连接u, v 的链，且该链中的每一条边，作为 D 中的弧，方向均一致，并且为 从 u 到 v, 则称该链为 从 u 到 v 的路;

回路：起点终点相同 的路;

初等路：各顶点都不相同的路;

初等回路：起点终点相同 的初等路;

连通图：基础图是连通图;

强连通图：任两点间有路.

图的基本概念与模型

图的矩阵描述:

如何在计算机中存储一个图呢？现在已有很多存储的方法，但最基本的方法就是采用矩阵来表示一个图，图的矩阵表示也根据所关心的问题不同而有:

邻接矩阵、关联矩阵、权矩阵等。

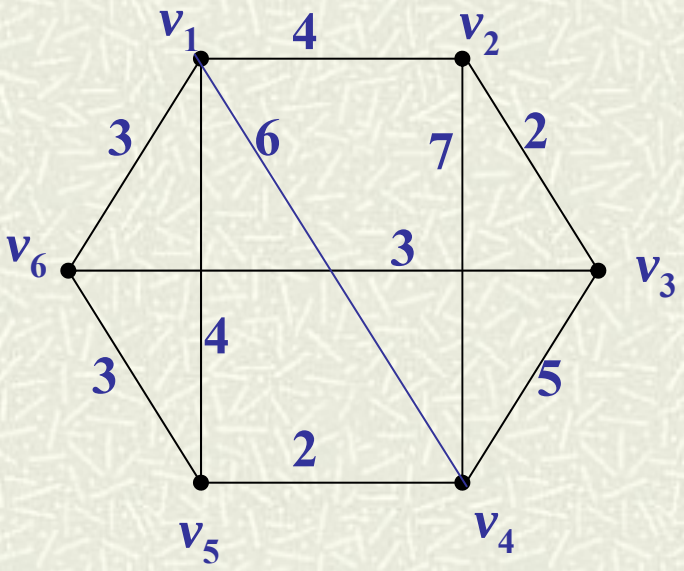
1. 邻接矩阵

对于图G=（V，E），|V|=n，|E|=m，有n×n阶方矩阵 $A=(a_{ij})_{n\times n}$，其中

$$a_{ij}=\begin{cases}1 & \text{当且仅当}v_i\text{与}v_j\text{之间有关联边时}\\ 0 & \text{其它}\end{cases}$$

图的基本概念与模型

例6.2 下图所表示的图可以构造邻接矩阵A如下



$$A_{6\times 6}=\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array}\begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

图的基本概念与模型

2. 关联矩阵

对于图G=(V,E), | V |=n, | E |=m, 有m×n阶矩阵$M=(m_{ij})_{n\times m}$,

其中:

$$m_{ij}=\begin{cases}2 & \text{当且仅当}v_i\text{是边}e_j\text{的两个端点}\\ 1 & \text{当且仅当}v_i\text{是边}e_j\text{的一个端点}\\ 0 & \text{其他}\end{cases}$$

3. 权矩阵

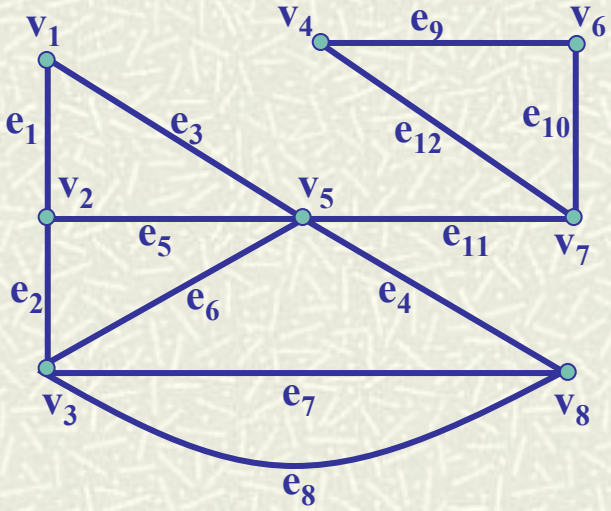
对于赋权图G=(V,E), 其中边 (v_i, v_j)有权 w_{ij}, 构造矩阵$B=(b_{ij})_{n\times n}$

其中:

$$b_{ij}=\begin{cases}w_{ij} & (v_i, v_j)\in E\\ 0 & (v_i, v_j)\notin E\end{cases}$$

图的基本概念与模型

例6.3 下图所表示的图可以构造关联矩阵M如下:

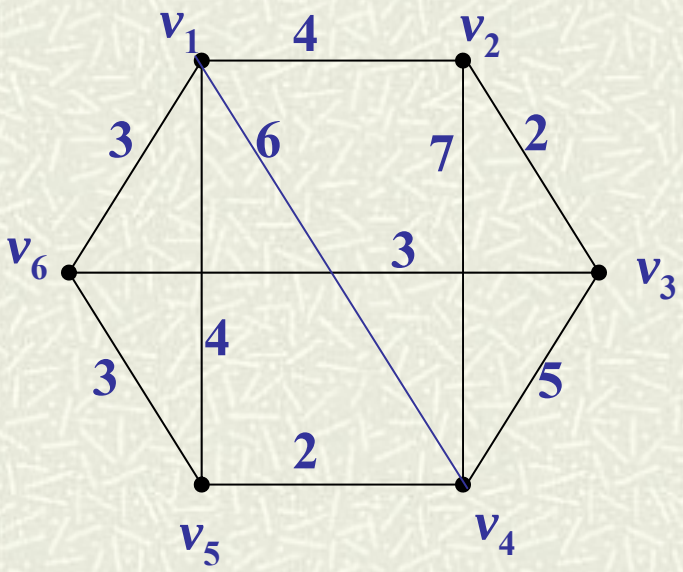


$M=(m_{ij})=$

	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12
v1	1	0	1	0	0	0	0	0	0	0	0	0
v2	1	1	0	0	1	0	0	0	0	0	0	0
v3	0	1	0	0	0	1	1	1	0	0	0	0
v4	0	0	0	0	0	0	0	0	1	0	0	1
v5	0	0	1	1	1	1	0	0	0	0	0	0
v6	0	0	0	0	0	0	0	0	1	1	0	0
v7	0	0	0	0	0	0	0	0	0	1	1	1
v8	0	0	0	1	0	0	1	1	0	0	0	0

图的基本概念与模型

例6.4 下图所表示的图可以构造权矩阵B如下:

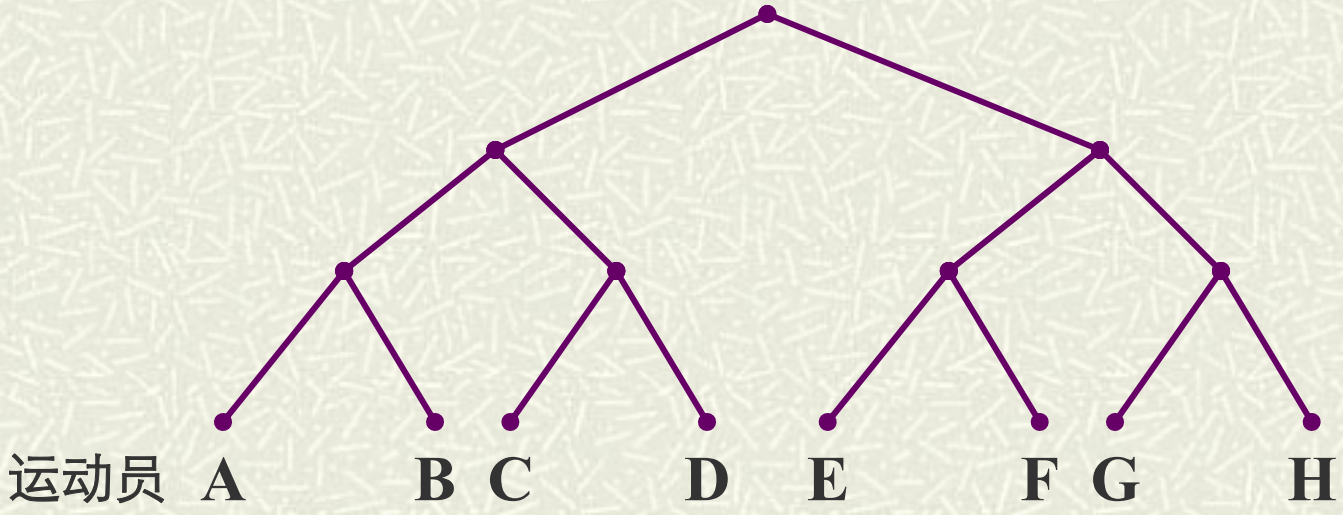


$$B=\begin{array}{c}\\ v_1\\ v_2\\ v_3\\ v_4\\ v_5\\ v_6\end{array}\begin{bmatrix}0&4&0&6&4&3\\4&0&2&7&0&0\\0&2&0&5&0&3\\6&7&5&0&2&0\\4&0&0&2&0&3\\3&0&3&0&3&0\end{bmatrix}\begin{array}{c}\\ v_1\ v_2\ v_3\ v_4\ v_5\ v_6\end{array}$$

树与图的最小树

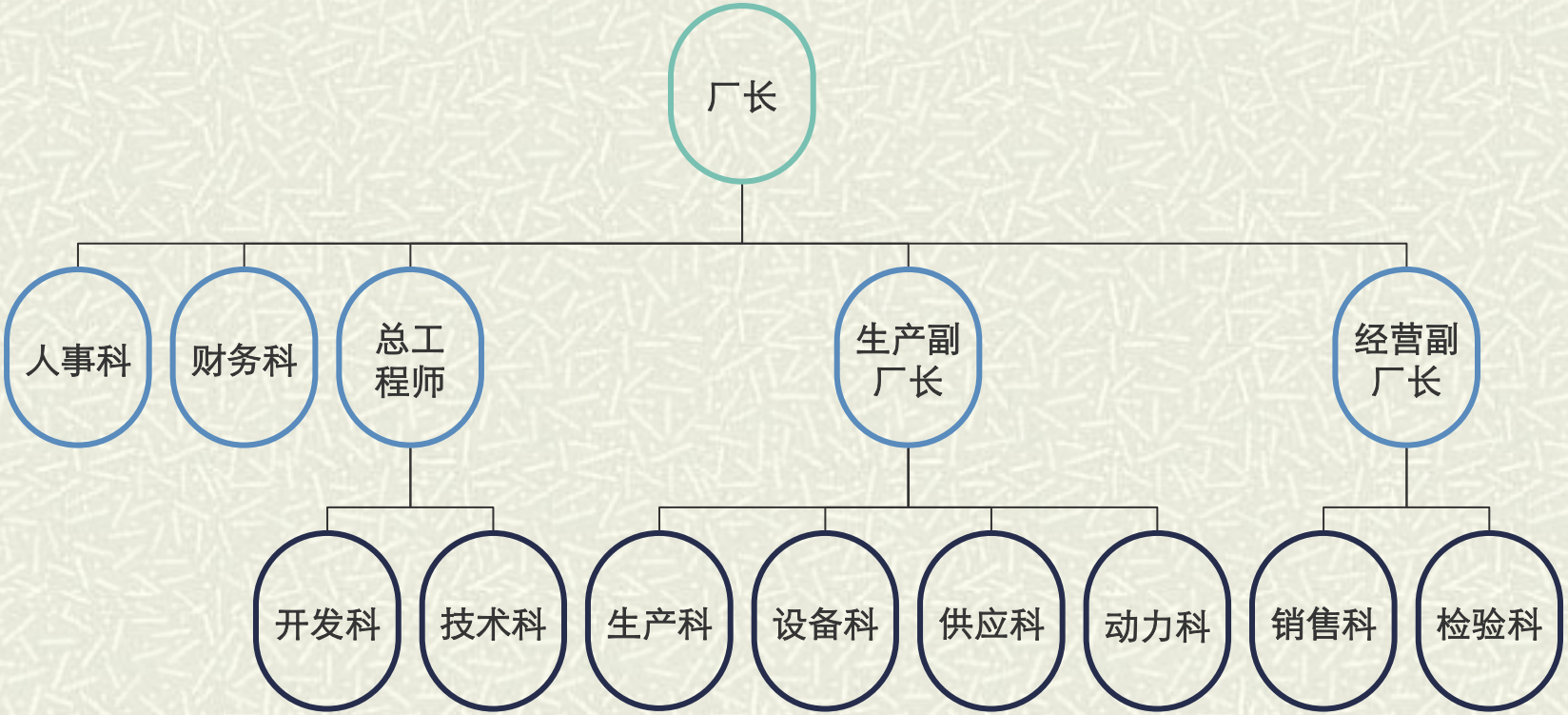
树是图论中结构最简单但又十分重要的图。在自然和社会领域应用极为广泛。

例6.2 乒乓求单打比赛抽签后，可用图来表示相遇情况，如下图所示。



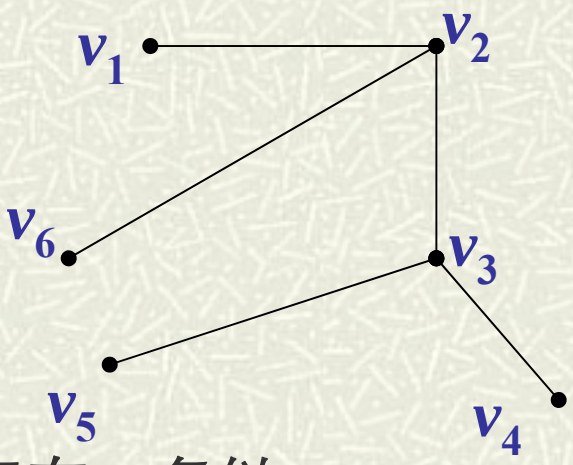
树与图的最小树

例6.3 某企业的组织机构图也可用树图表示。



树与图的最小树

- 树：无圈的连通图即为树



性质1：任何树中必存在次为1的点。

性质2：n 个顶点的树必有n-1 条边。

性质3：树中任意两个顶点之间，恰有且仅有一条链。

性质4：树连通，但去掉任一条边，必变为不连通。

性质5：树无回圈，但不相邻的两个点之间加一条边，恰得到一个圈。

树的性质

1. 设图 G=(V, E) 是一个树, 顶点数 p(G)\geq 2, 那么图G 中至少有两个悬挂点。
2. 图G=(V, E) 是一个树的充要条件是G 不含圈，并且有且仅有 p(G)-1 条边。
3. 图G=(V, E) 是一个树的充要条件是G 是连通图，并且有且仅有 p(G)-1 条边。
4. 图G 是一个树的充分必要条件是任意两个顶点之间有且仅有一条链。

上述定理表明（1）从一个树中任意去掉一条边，那么剩下的图不是连通图，亦即，在点集合相同的图中，树是含边数最少的连通图；（2）在树中不相邻的两个点之间加上一条边， 恰好得到一个圈。

支撑树

1. 设图 K=(V,E') 是图 G=(V,E) 的一支撑子图. 如果图 K 是一个树, 那么称 K 是 G 的一个支撑树.
2. 显然, 如果图 K=(V,E') 是图 G=(V,E) 的一支撑树, 那么 K 的边数是 p(G)-1;
3. G 中不属于支撑树K 的边构成 K 的余树，其边数是 q(G)-p(G)+1.
4. 定理：一个图 G 有支撑树的充要条件是 G 是连通图.

上述定理充分性的证明，提供了一个寻找连通图支撑树的方法，叫做“破圈法”：从图中任取一个圈，去掉一条边, 再对剩下的图重复以上步骤，直到不含圈时为止，这样就得到一个支撑树。

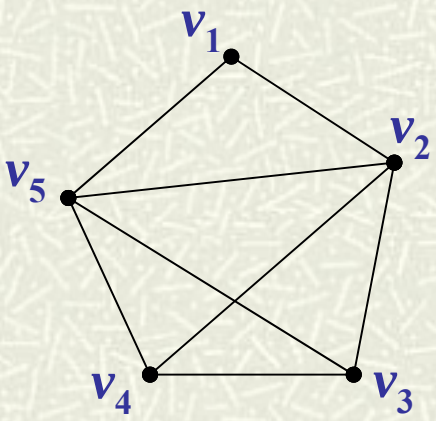
最小支撑树

1. 如果对于 G=(V,E) 中的每一条 [vi,vj], 相应地有一个数 w{ij}, 那么称这样的图 G 为赋权图, wij 称为边[vi,vj] 的权 (这里所指的权，是广义意义下的数量值，根据实际研究问题的不同，可以具有不同的含义。例如长度，费用、流量等等)
2. 如果图T =(V,E') 是图 G 的一个支撑树，那么称 E' 上所有边的权之和为支撑树 T 的权，记作S(T).
3. 如果图G 的支撑树 T* 的权 S(T*) 在 G 的所有支撑树 T 中的权最小，即 S(T*) = min S(T), 那么称 T* 是G 的最小支撑树 (最小支撑树同样可由破圈法求的: 每次找到一个圈，去掉权最大的边即可)

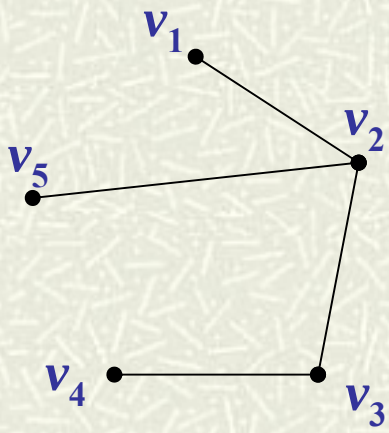
树与图的最小树

图的最小部分树(支撑树)

如果G2是G1的部分图，又是树图，则称G2是G1的部分树（或支撑树）。树图的各条边称为树枝，一般图G1含有多个部分树，其中树枝总长最小的部分树，称为该图的最小部分树（或最小支撑树）。

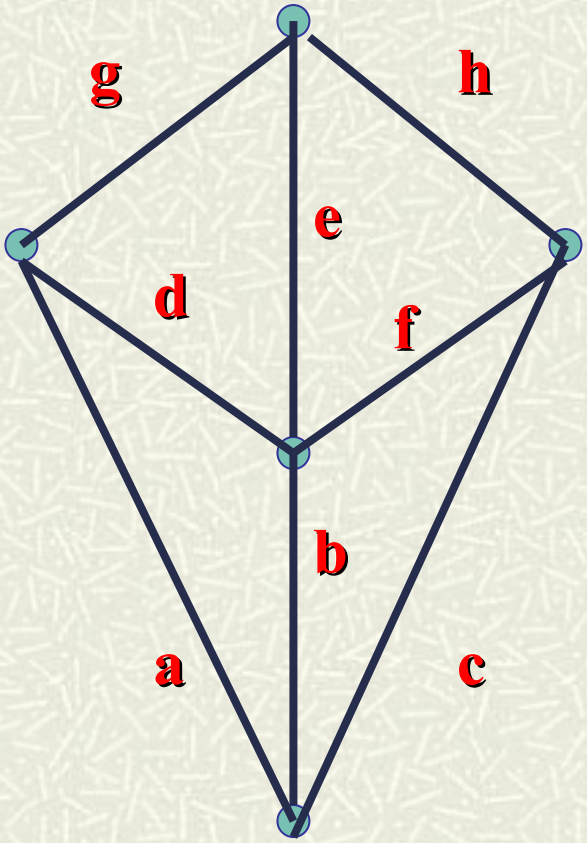


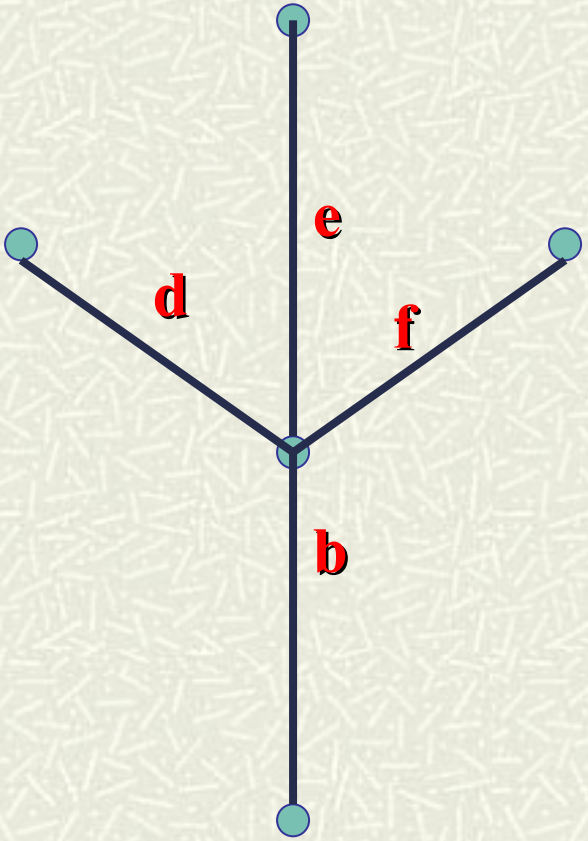
G1



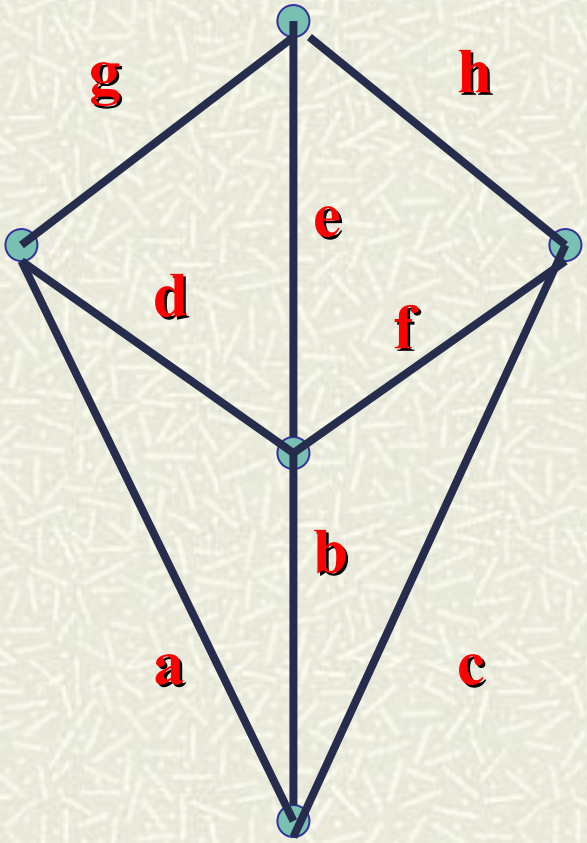
G2

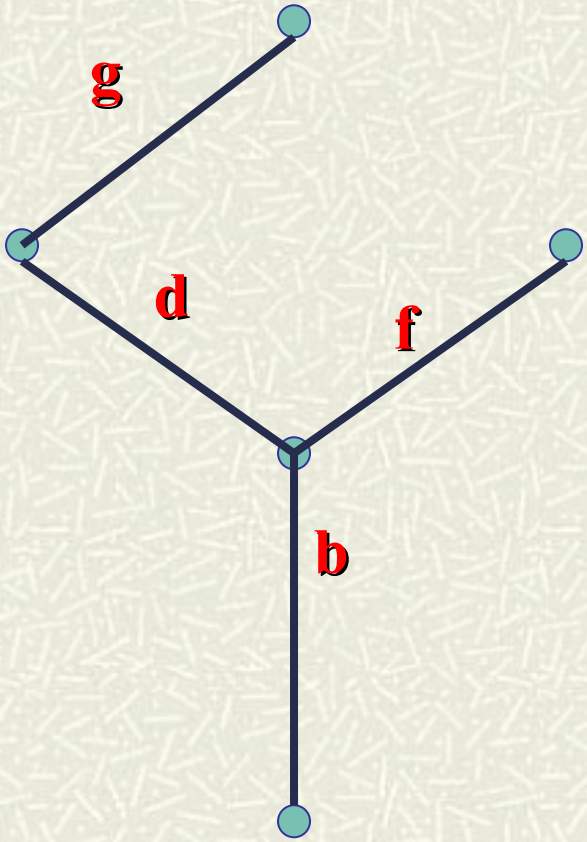
树与图的最小树



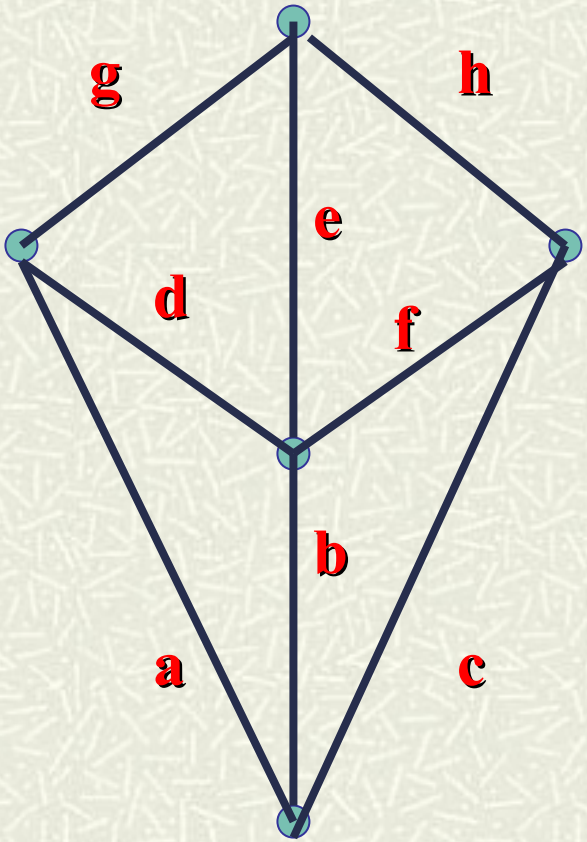


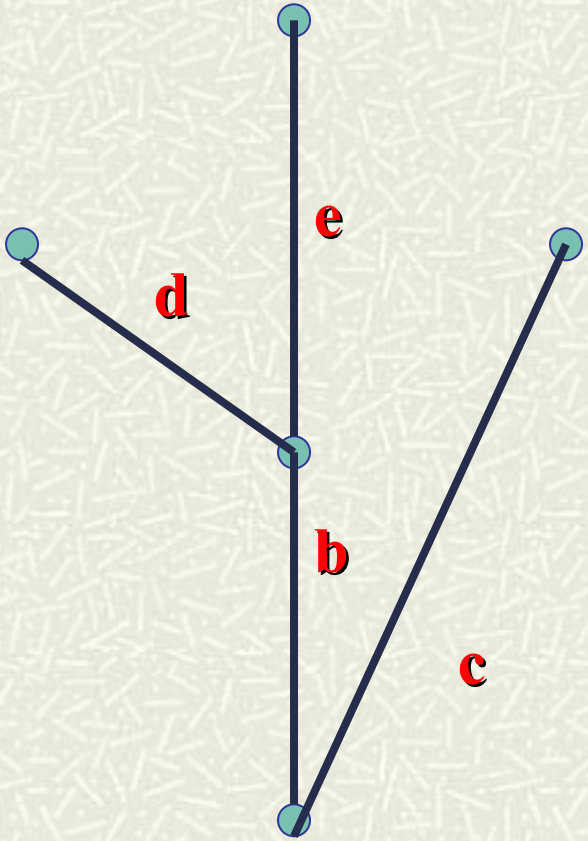
树与图的最小树



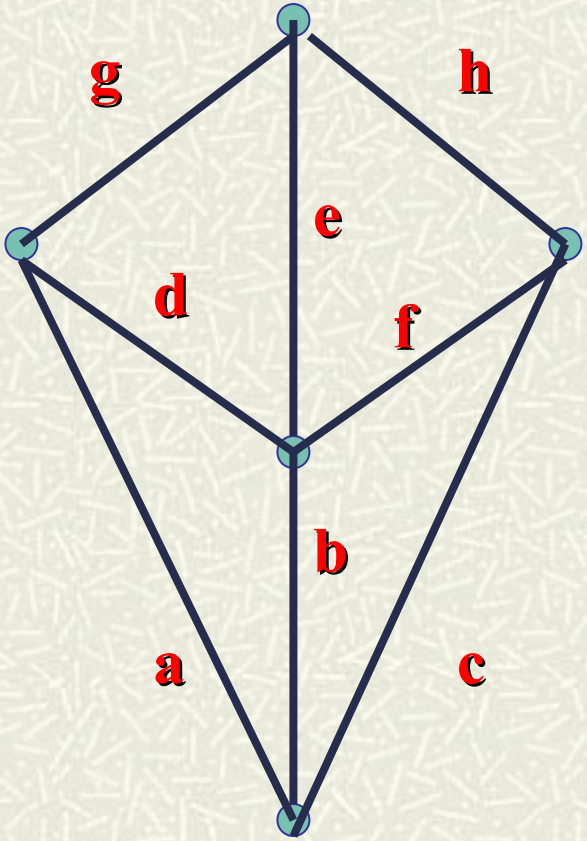


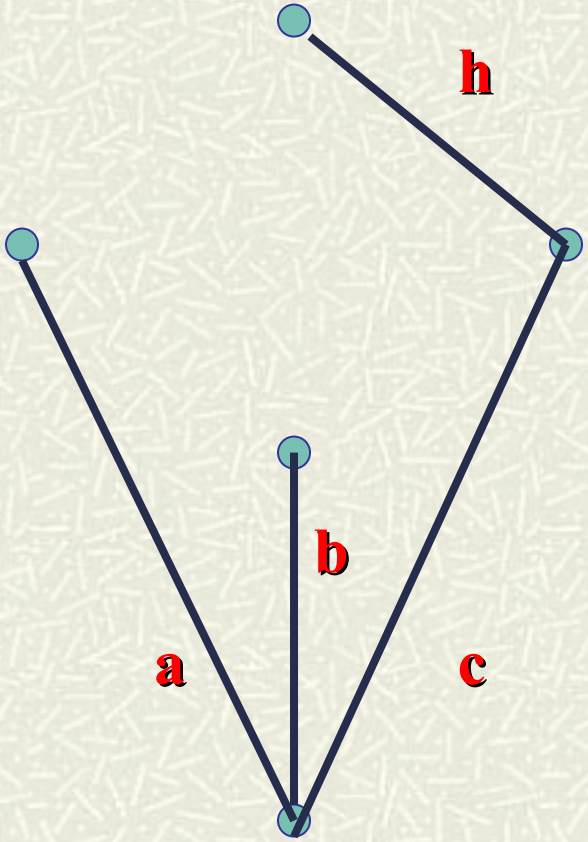
树与图的最小树



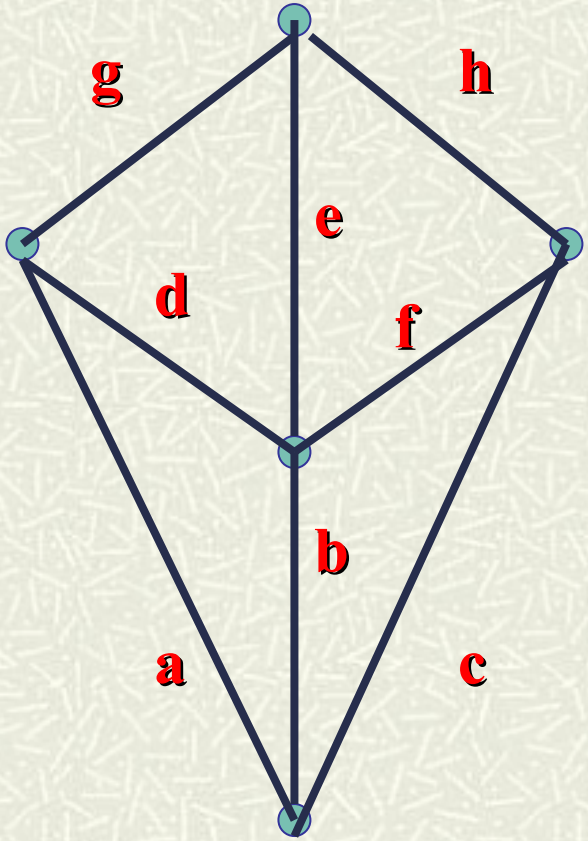


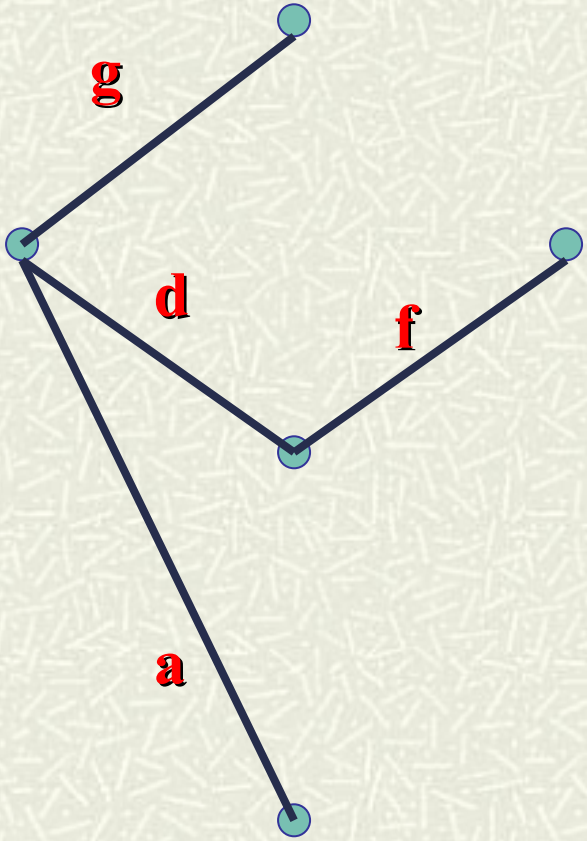
树与图的最小树





树与图的最小树

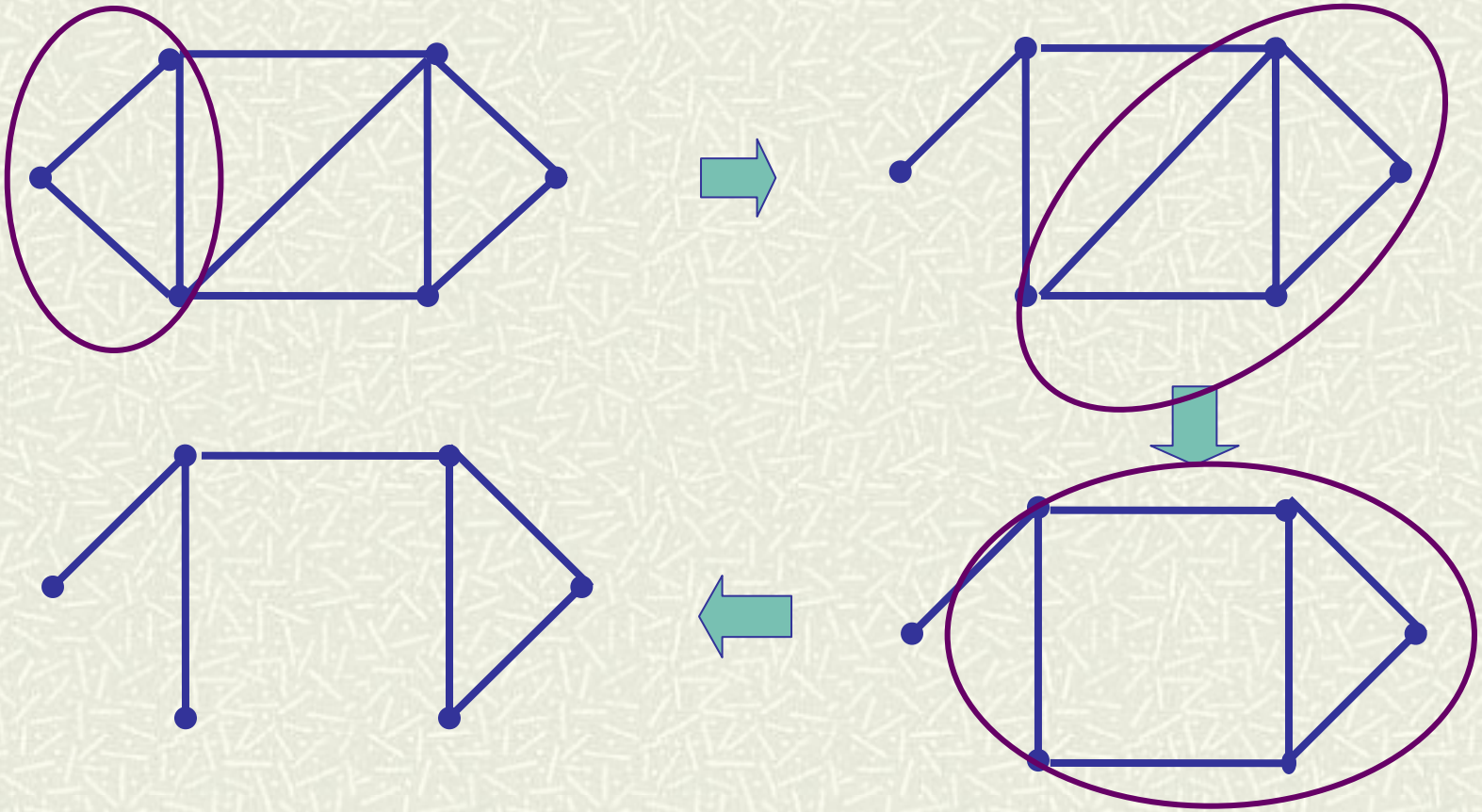




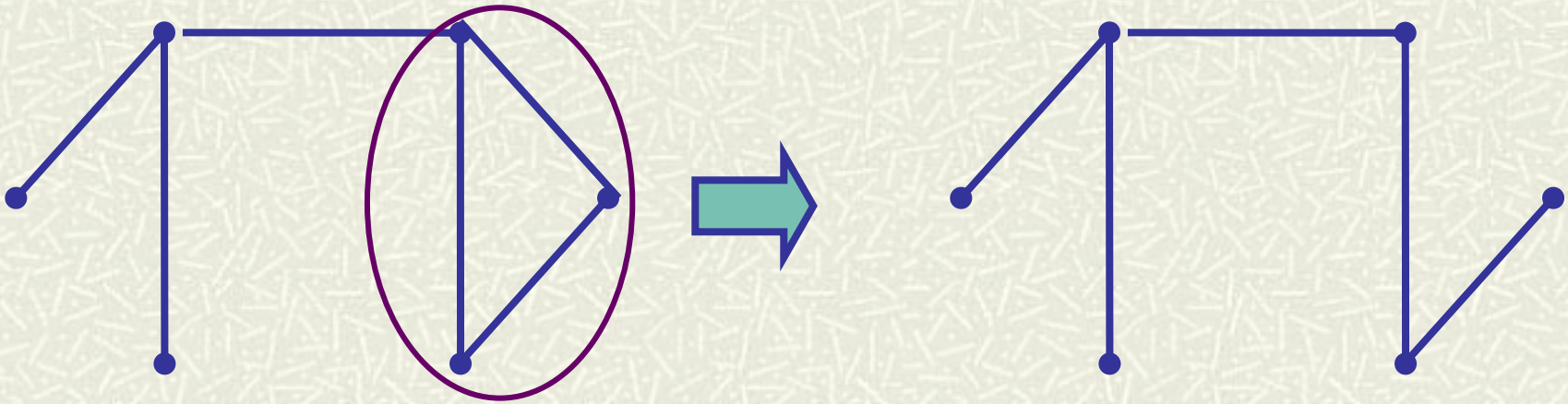
树与图的最小树

求树的方法：破圈法和避圈法

破圈法



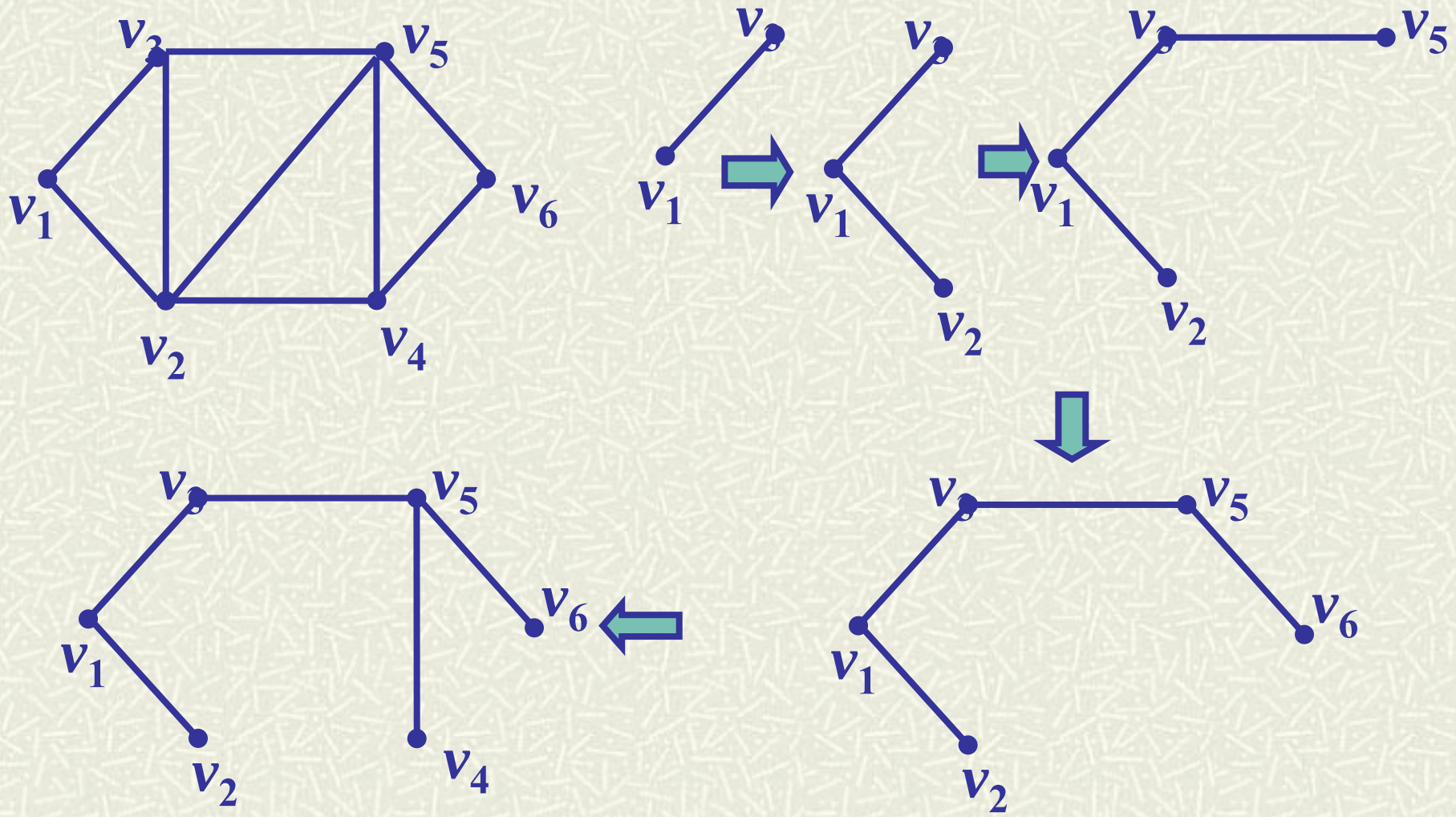
树与图的最小树



部分树

树与图的最小树

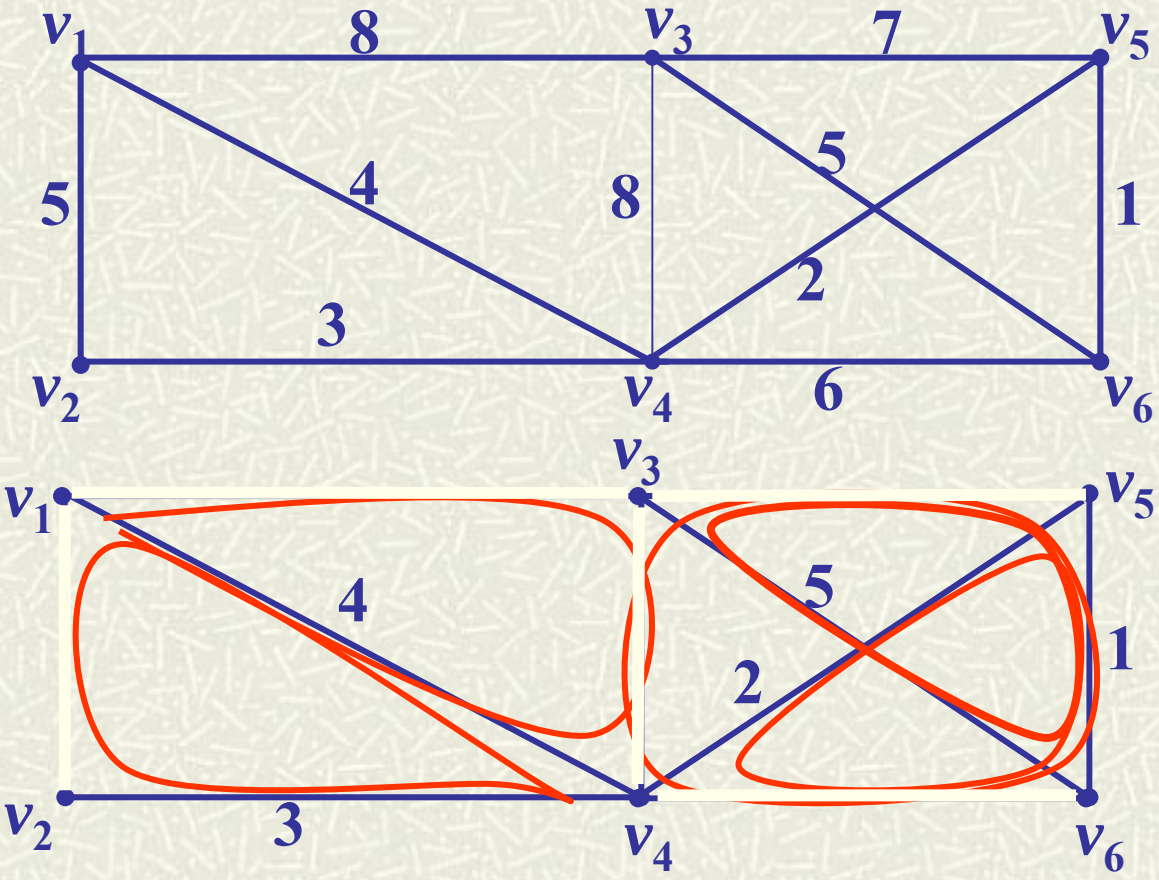
避圈法



树与图的最小树

赋权图中求最小树的方法：破圈法和避圈法

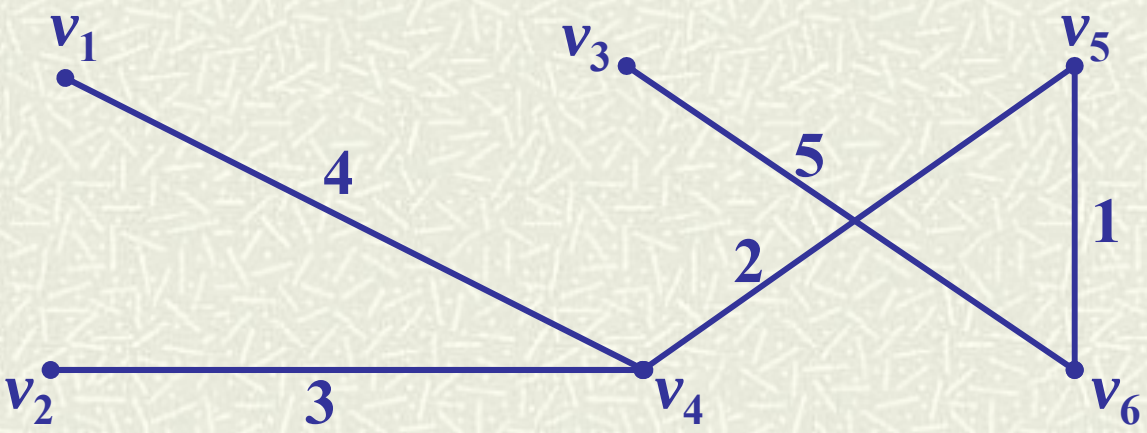
破圈法：任取一圈，去掉圈中最长边，直到无圈。



边数 = n-1=5

树与图的最小树

得到最小树:



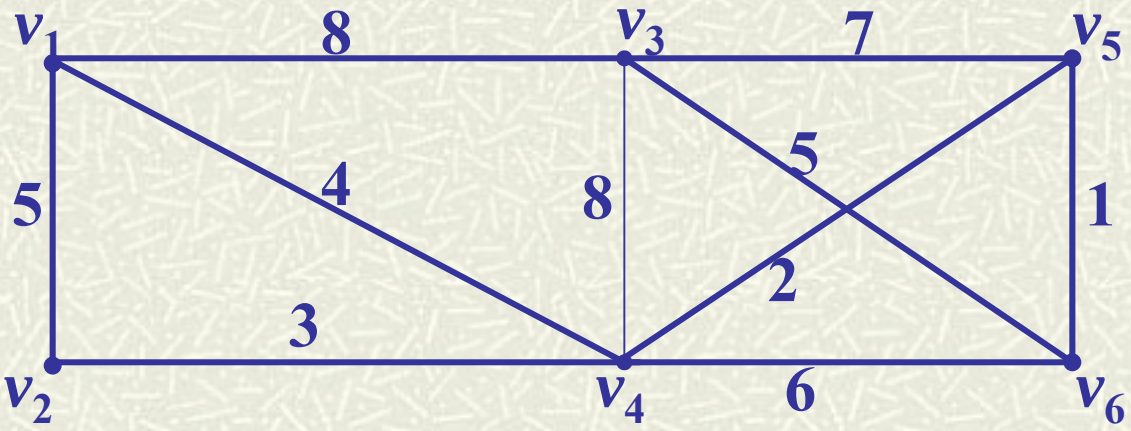
Min C(T)=15

树与图的最小树

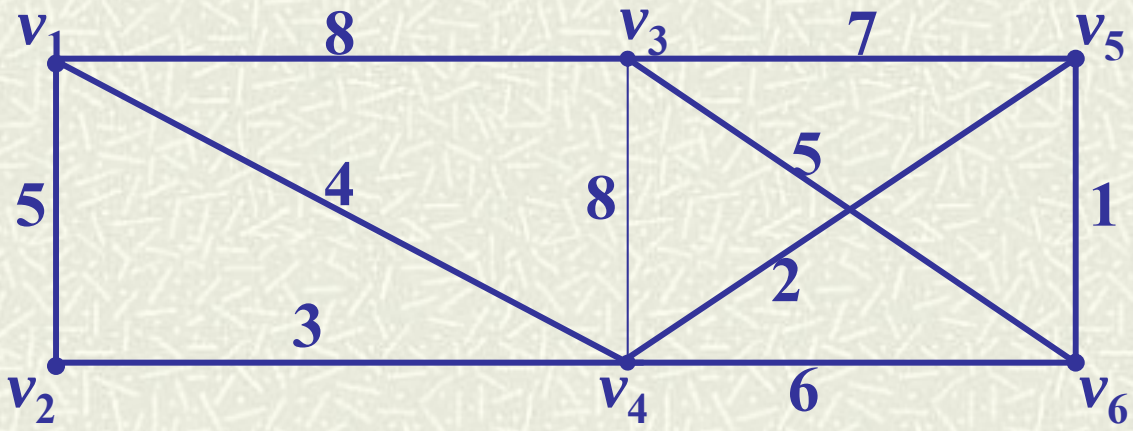
避圈法:

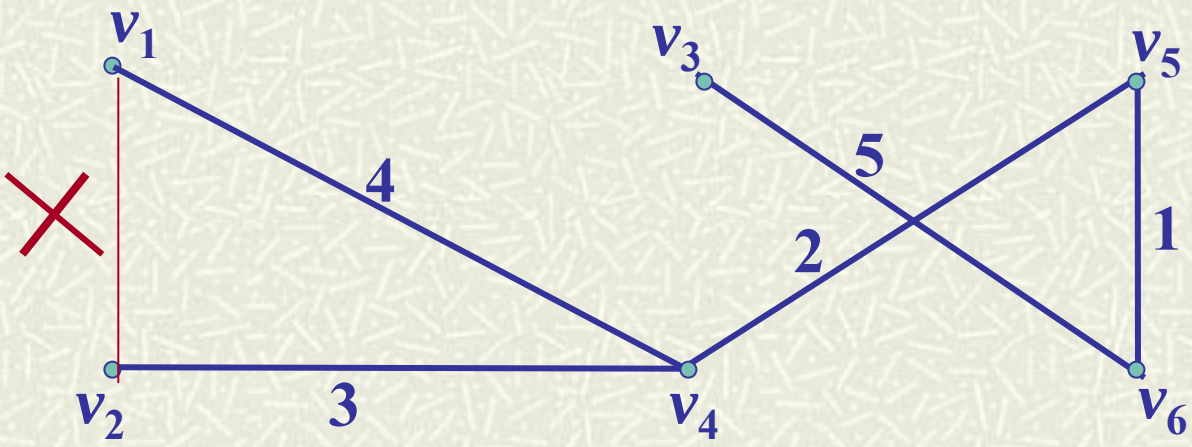
去掉G中所有边，得到n个孤立点；然后加边。

加边的原则为：从最短边开始添加，加边的过程中不能形成圈，直到点点连通(即:n-1条边)。



树与图的最小树

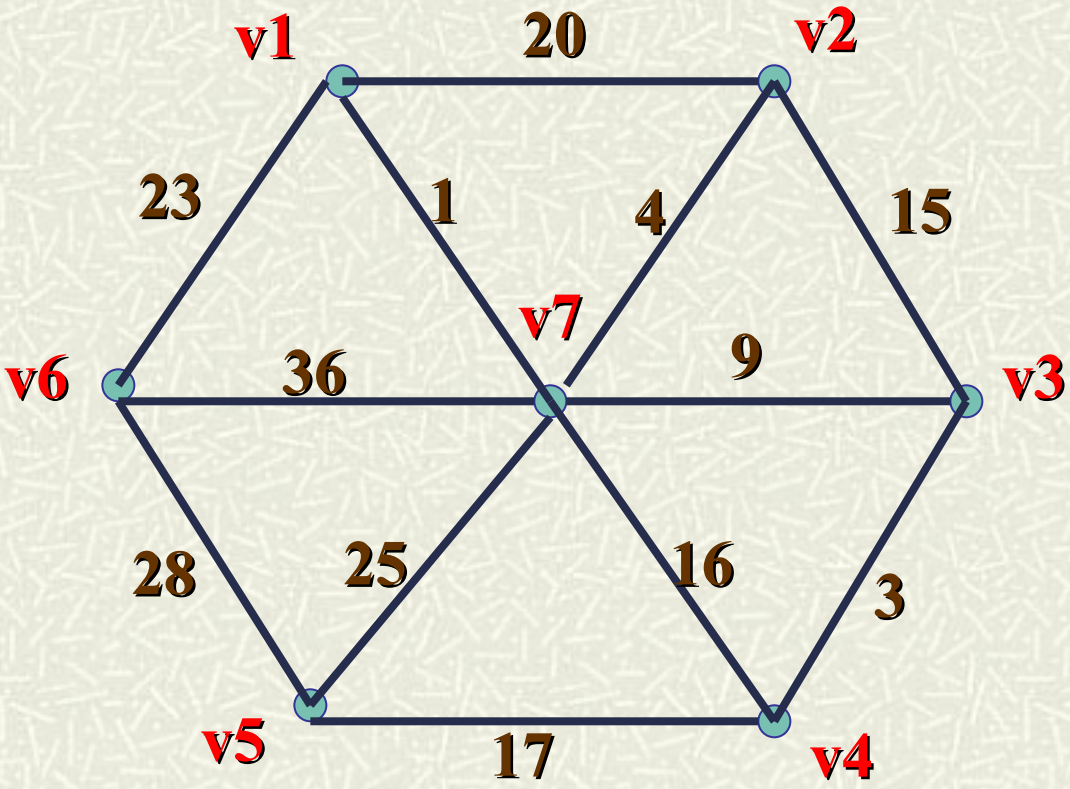




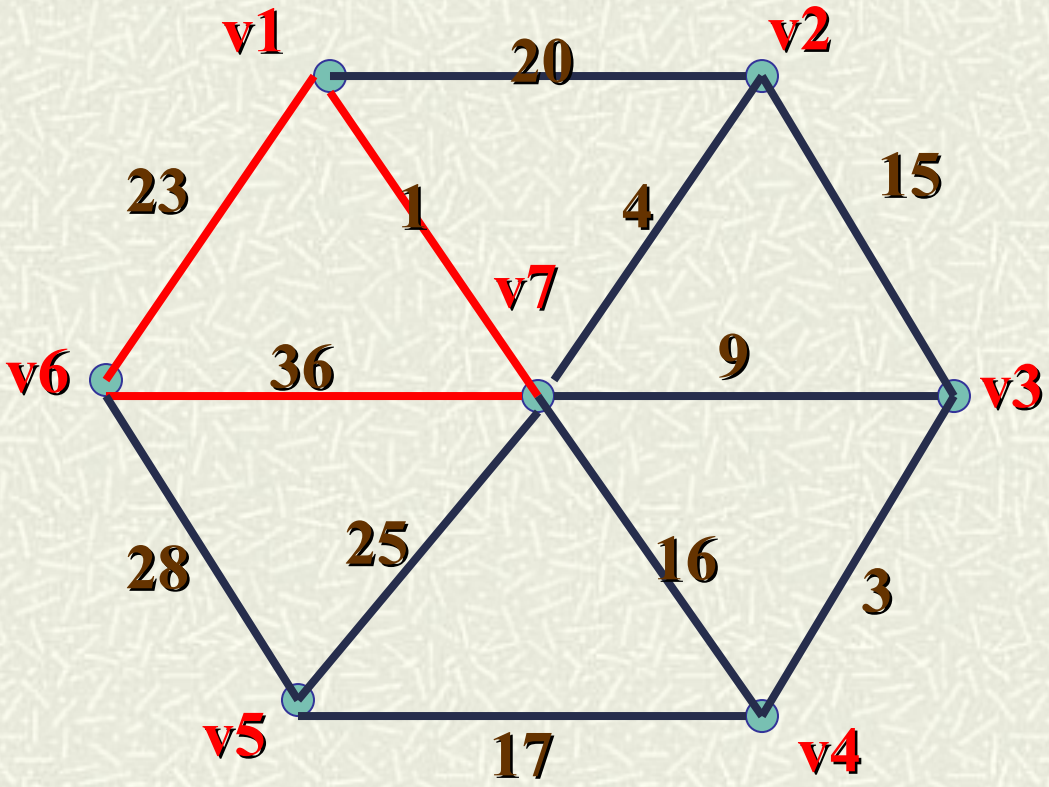
Min C(T)=15

树与图的最小树

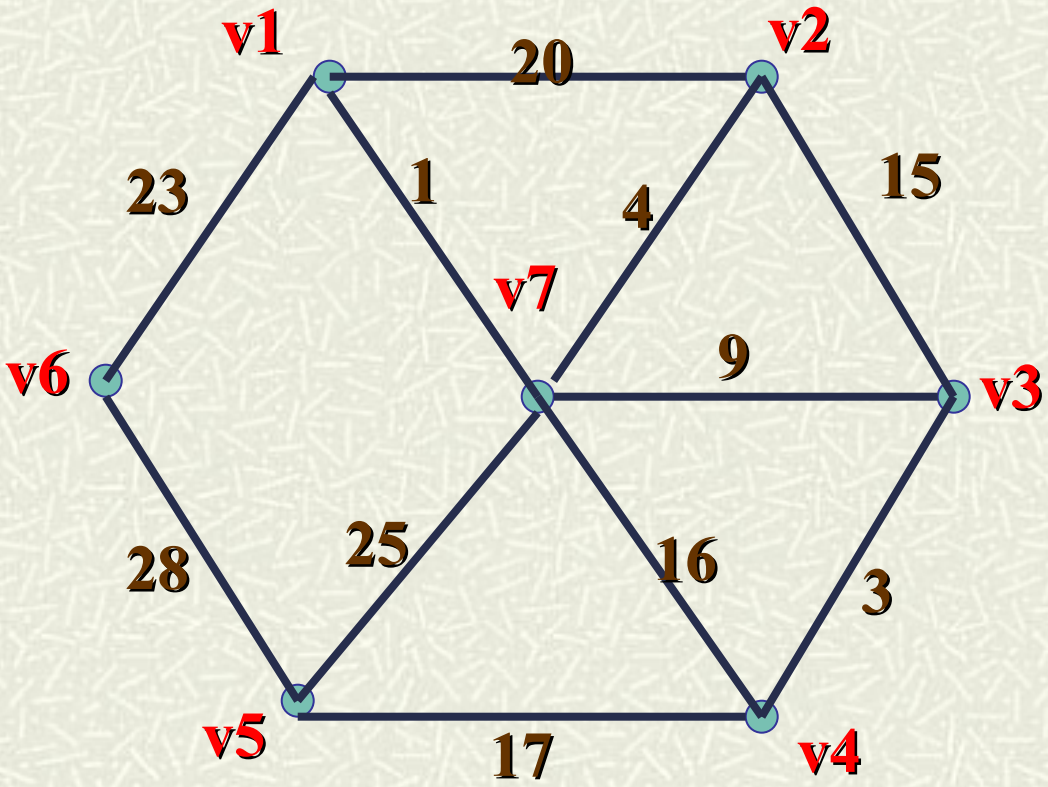
练习：应用破圈法求最小树



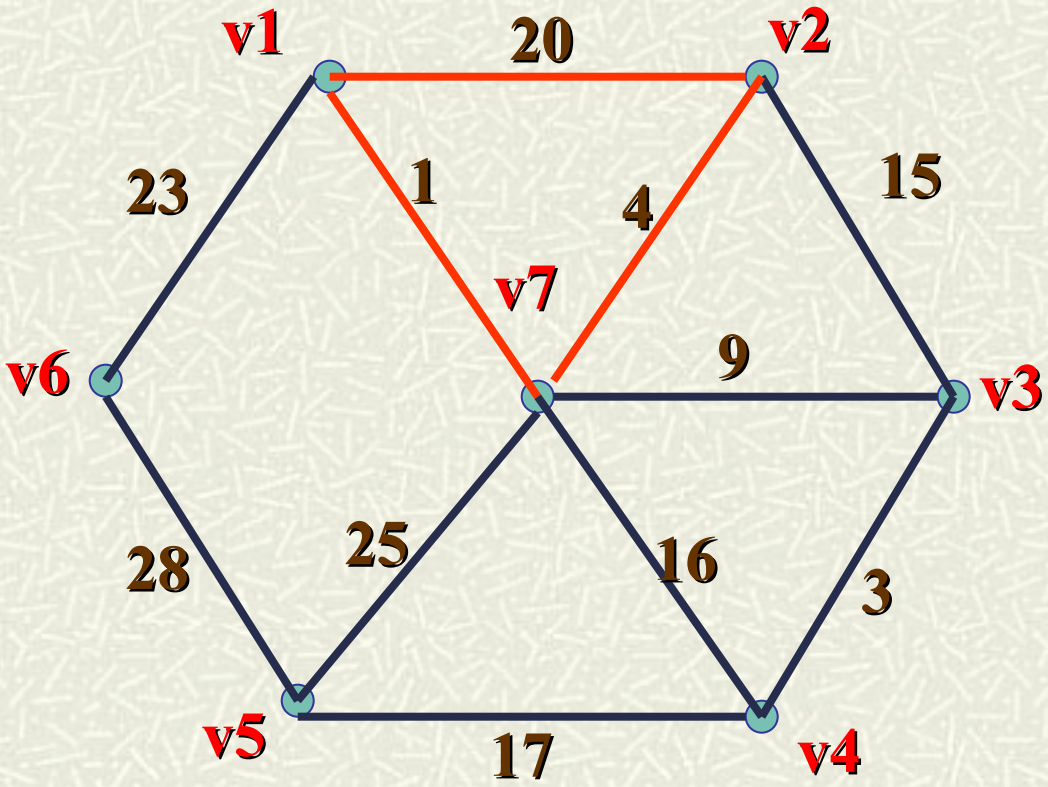
树与图的最小树



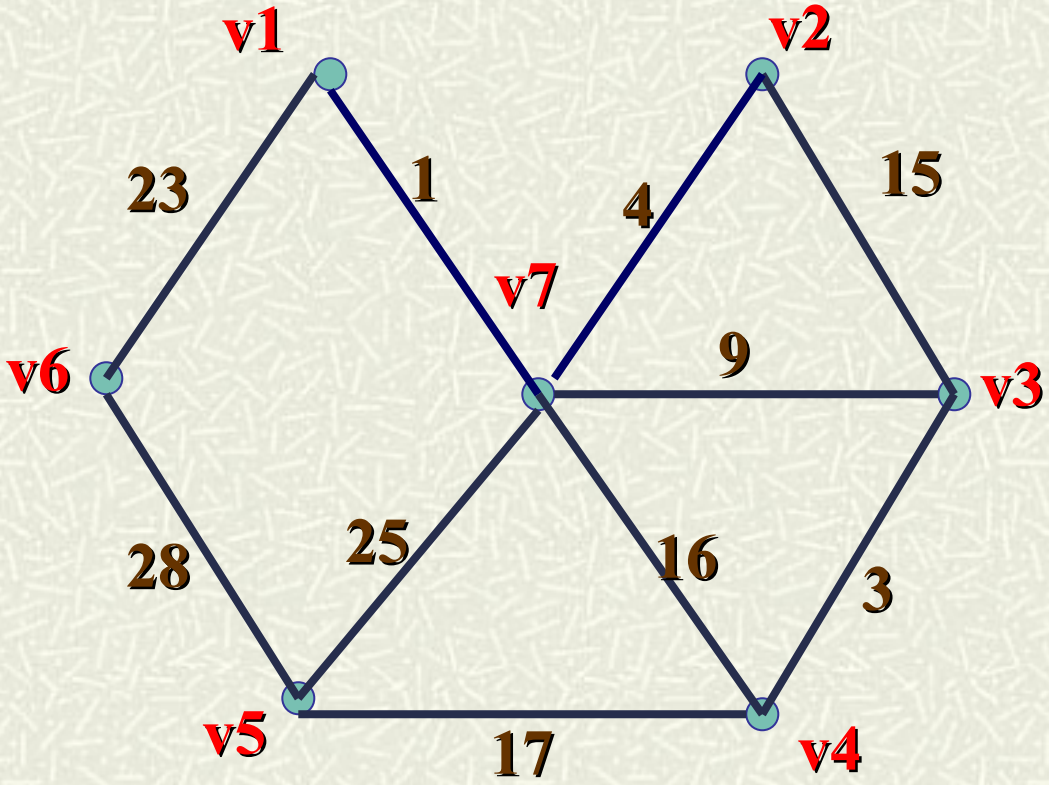
树与图的最小树



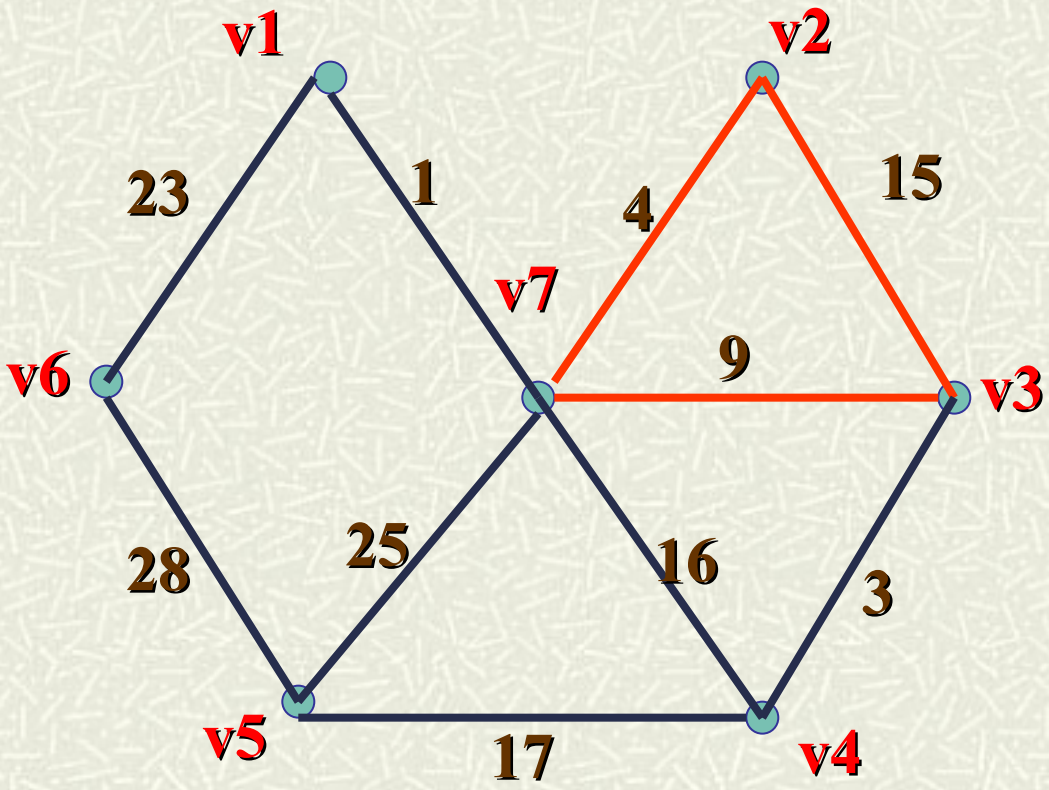
树与图的最小树



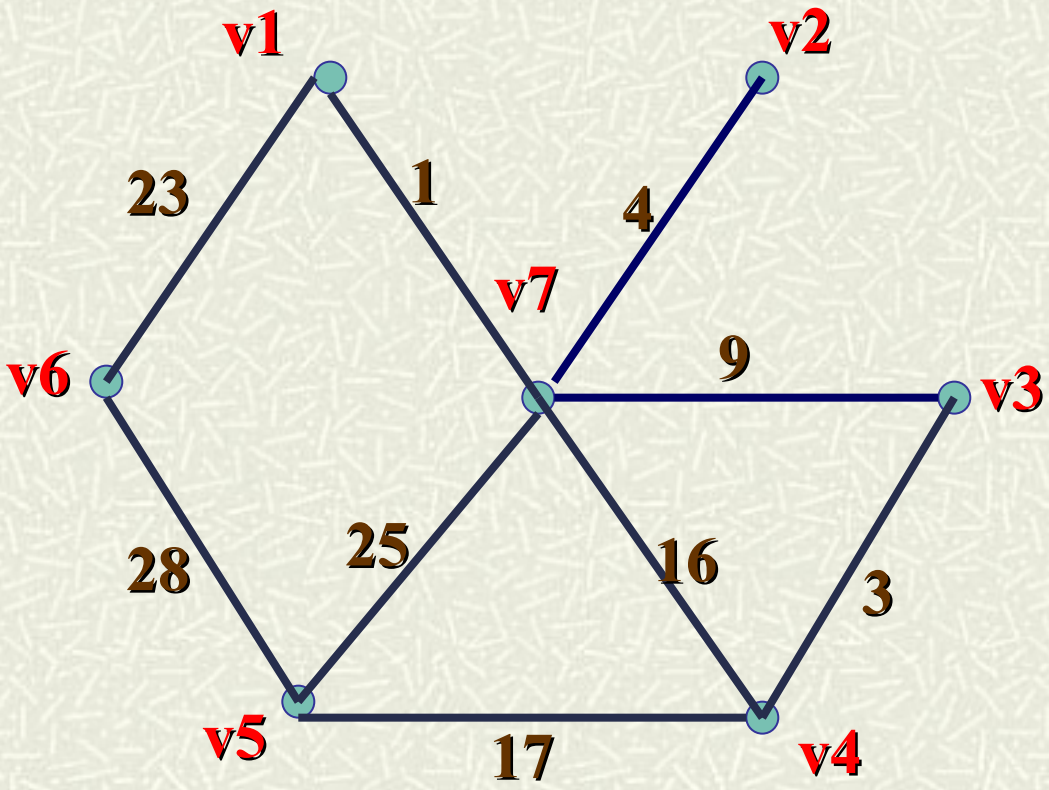
树与图的最小树



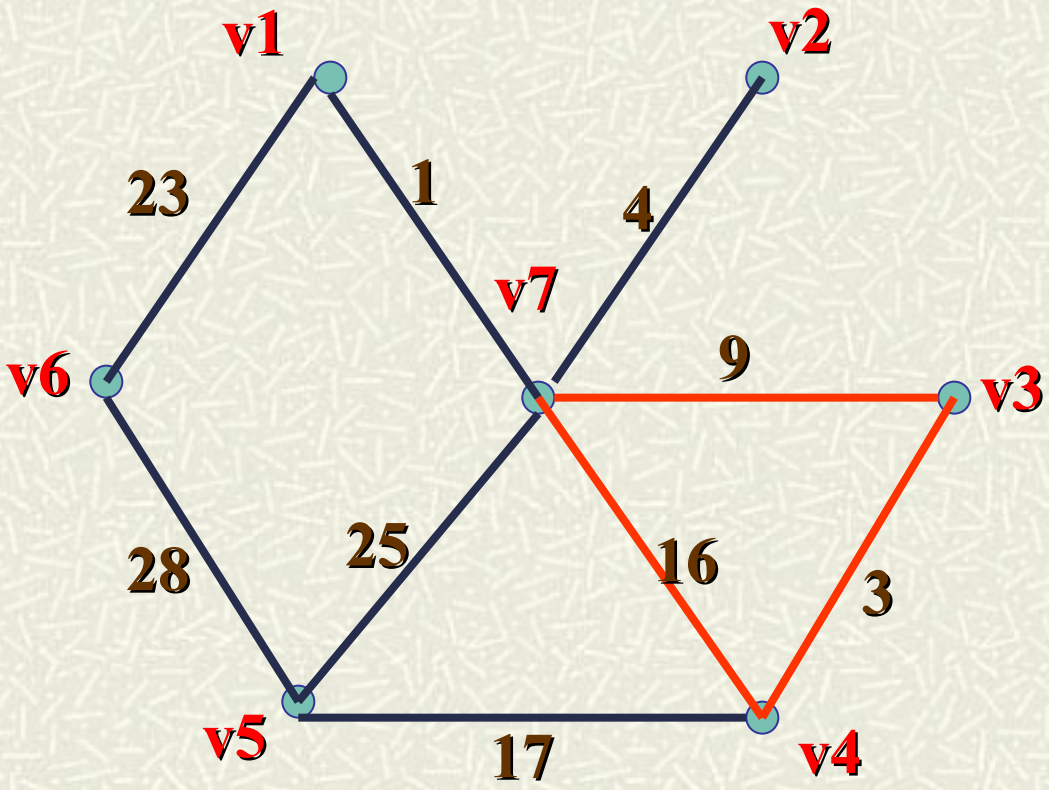
树与图的最小树



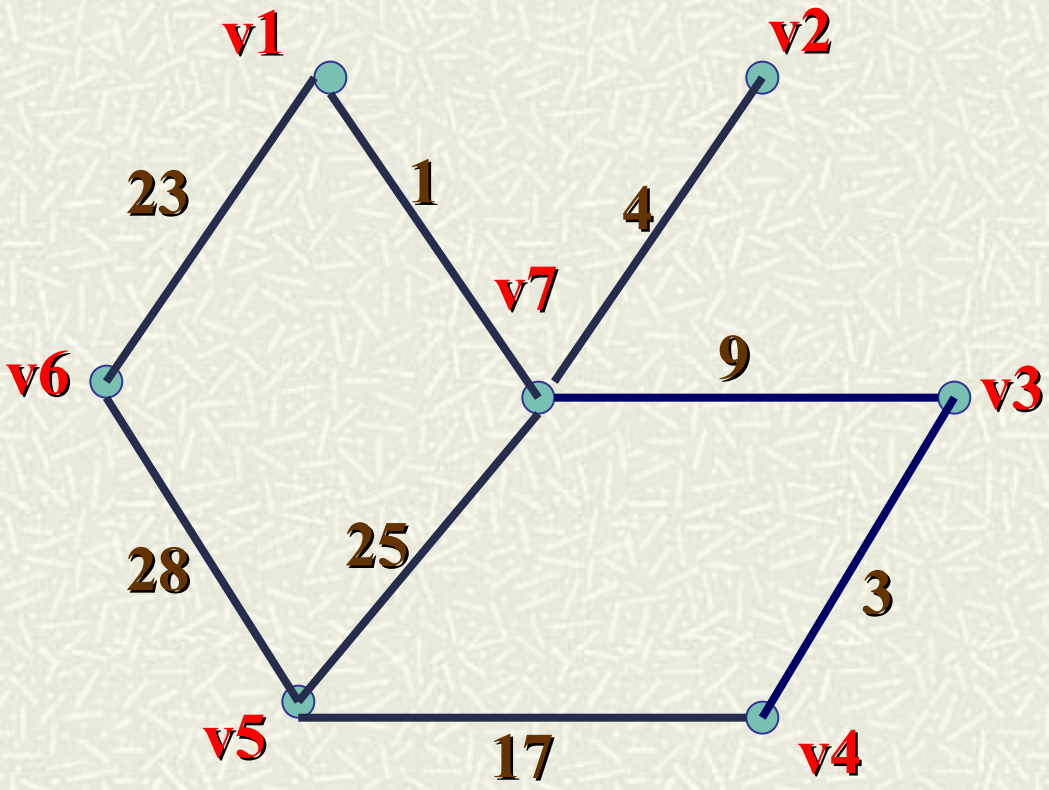
树与图的最小树



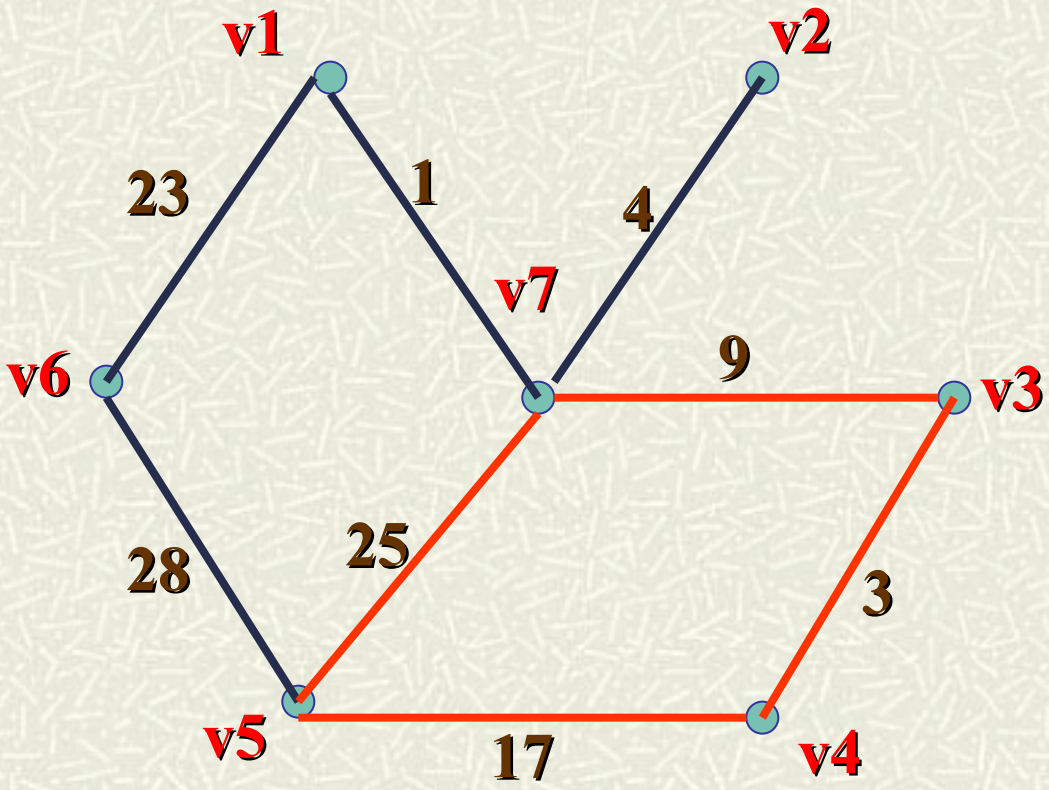
树与图的最小树



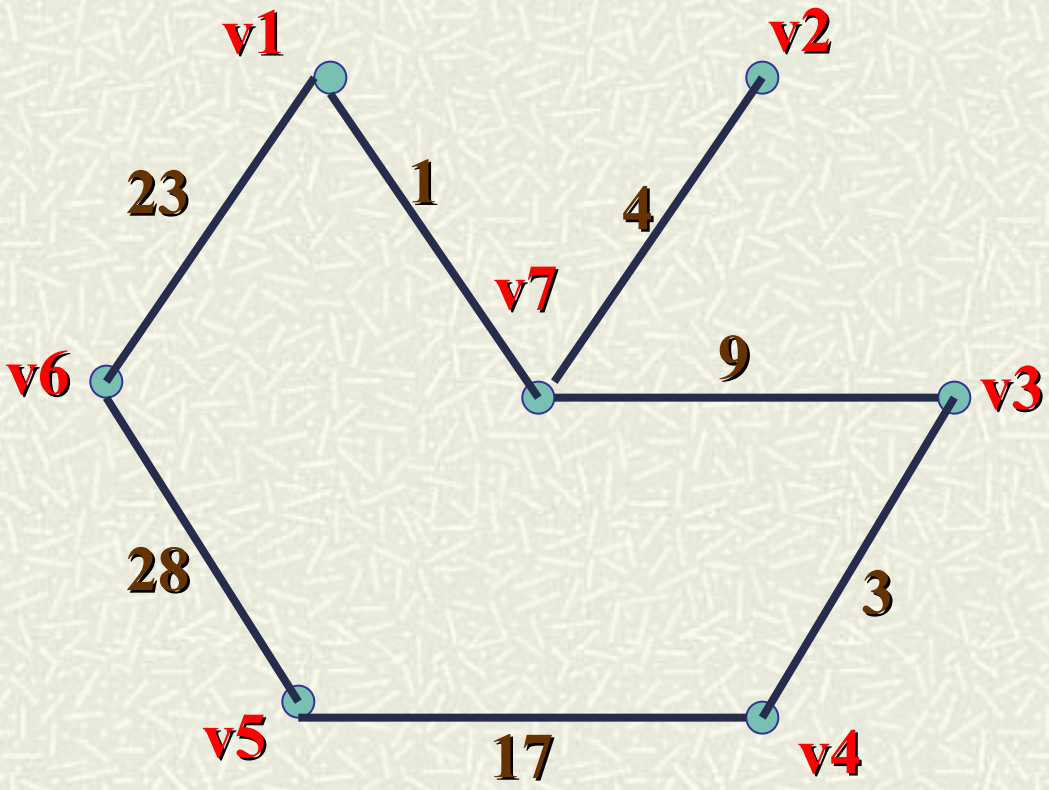
树与图的最小树



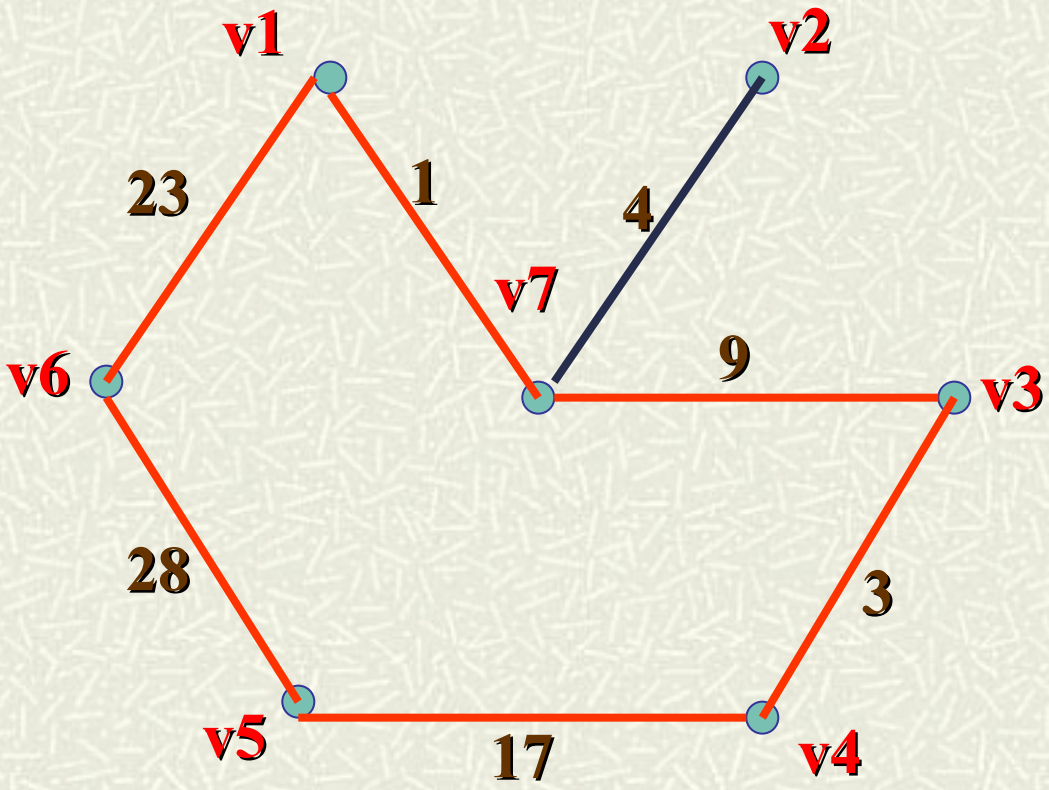
树与图的最小树



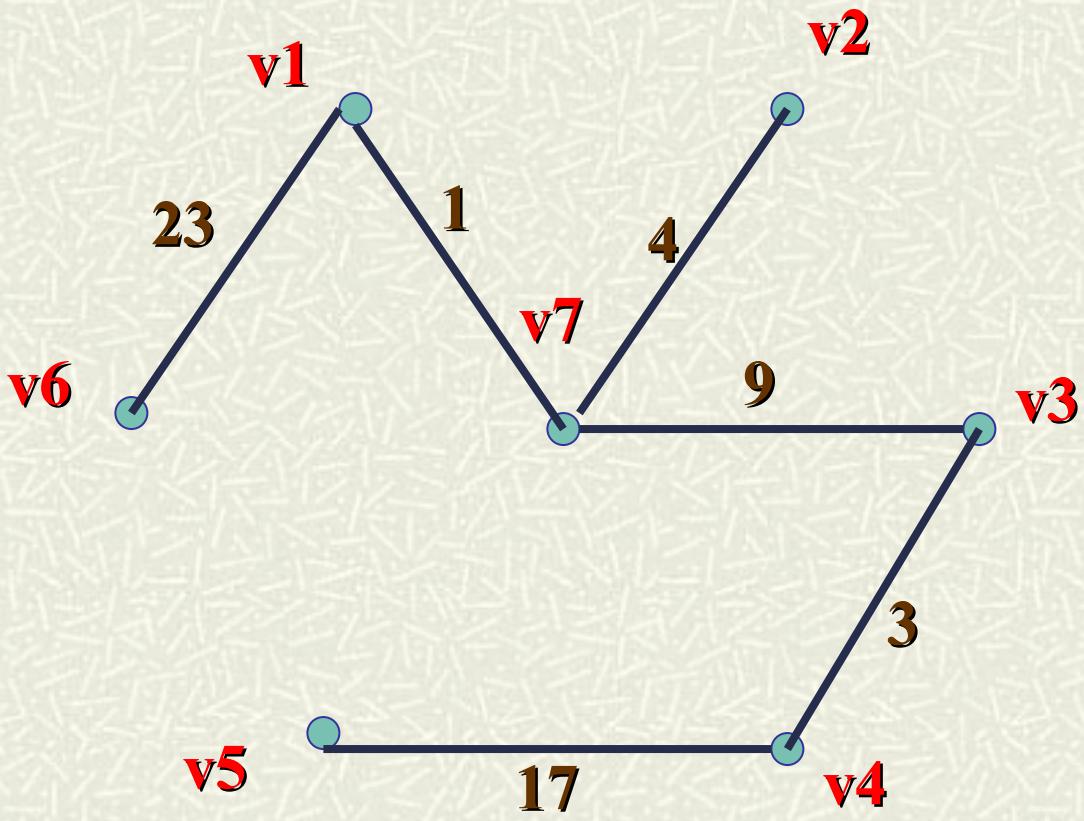
树与图的最小树



树与图的最小树

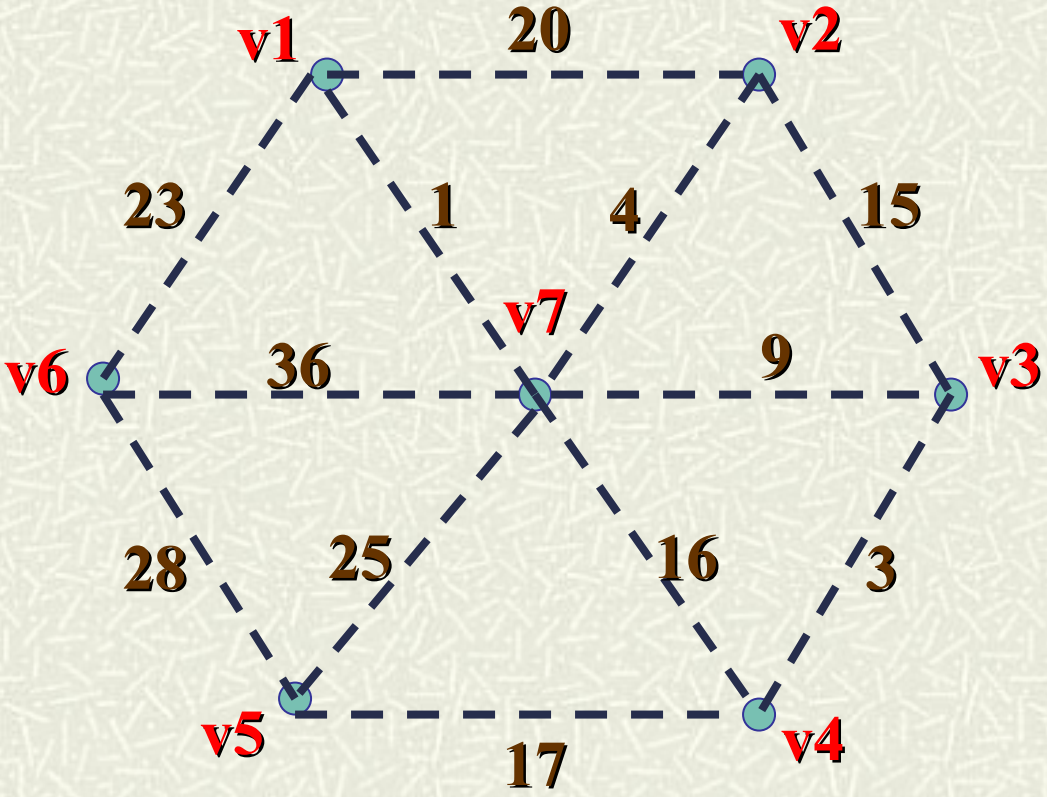


树与图的最小树

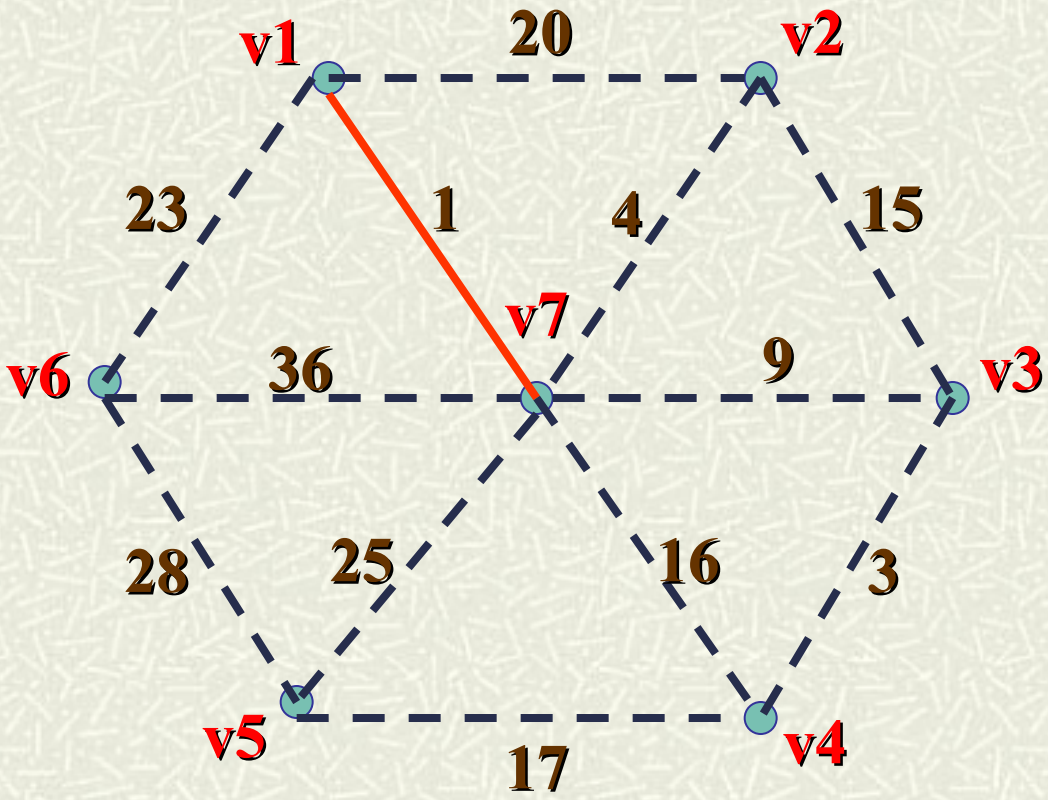


min=1+4+9+3+17+23=57

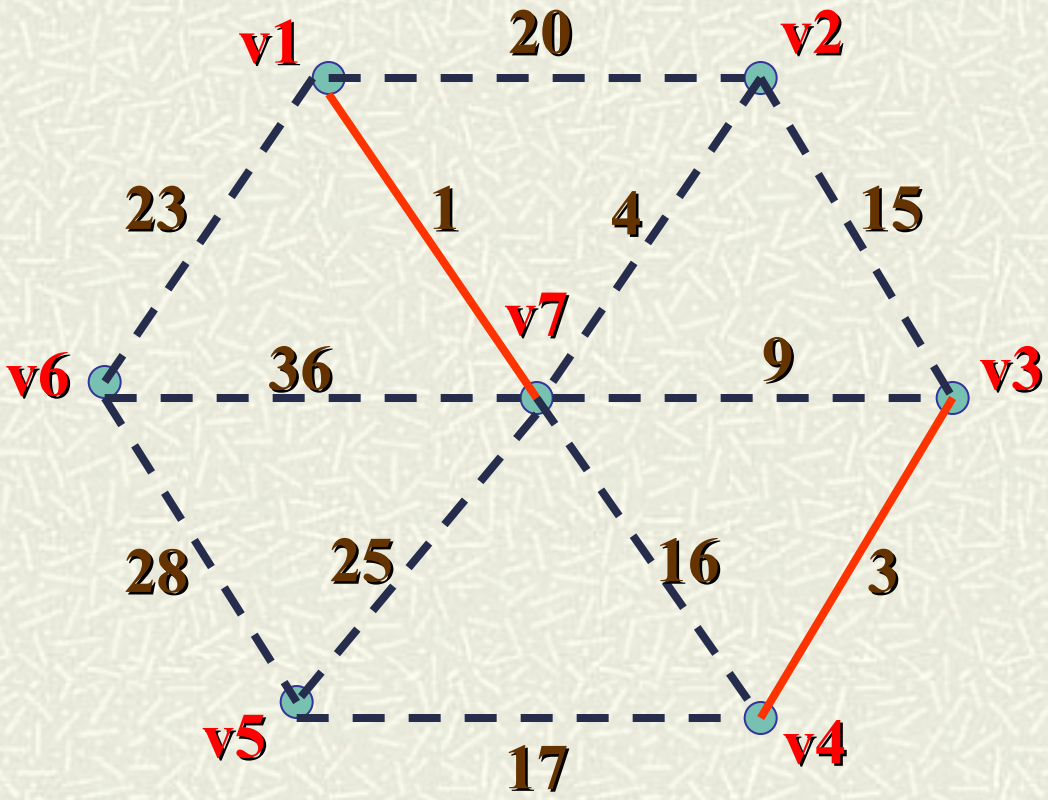
练习：应用避圈法求最小树



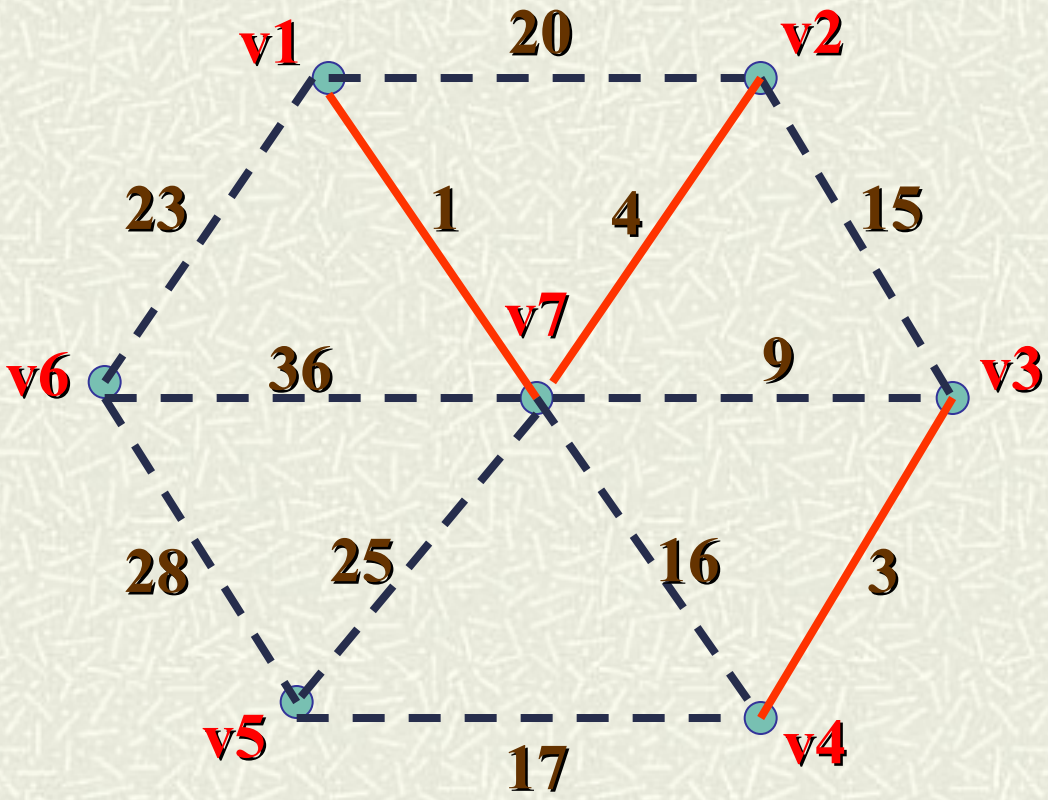
树与图的最小树



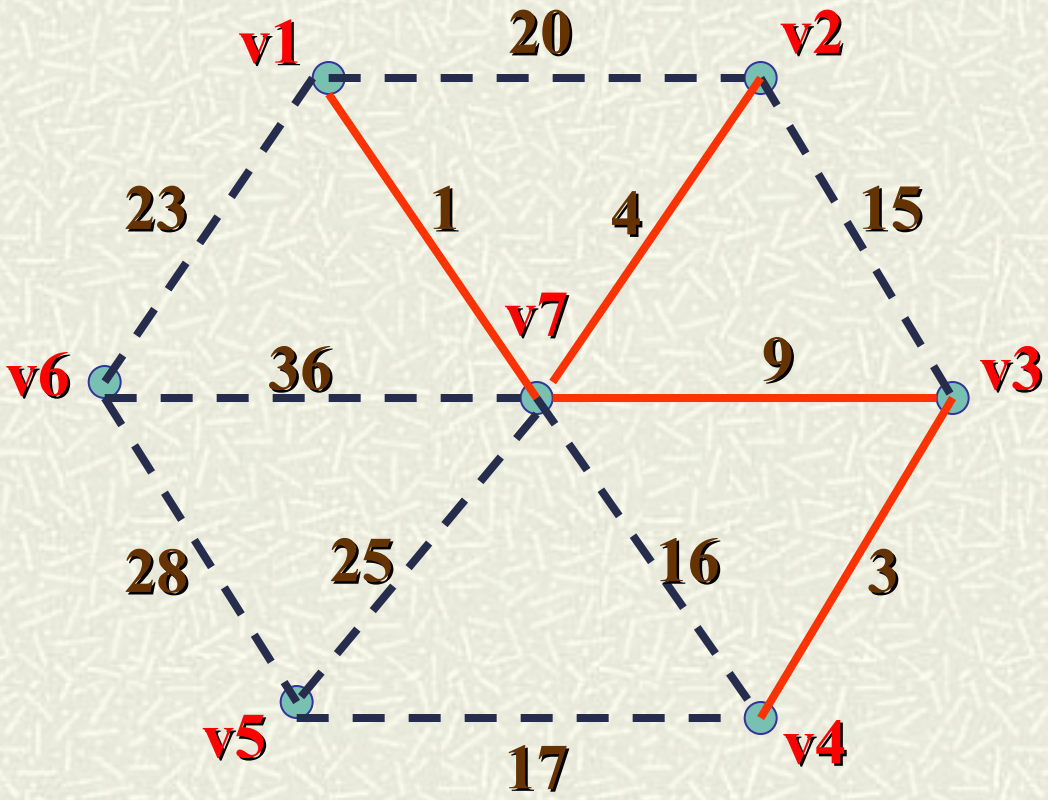
树与图的最小树



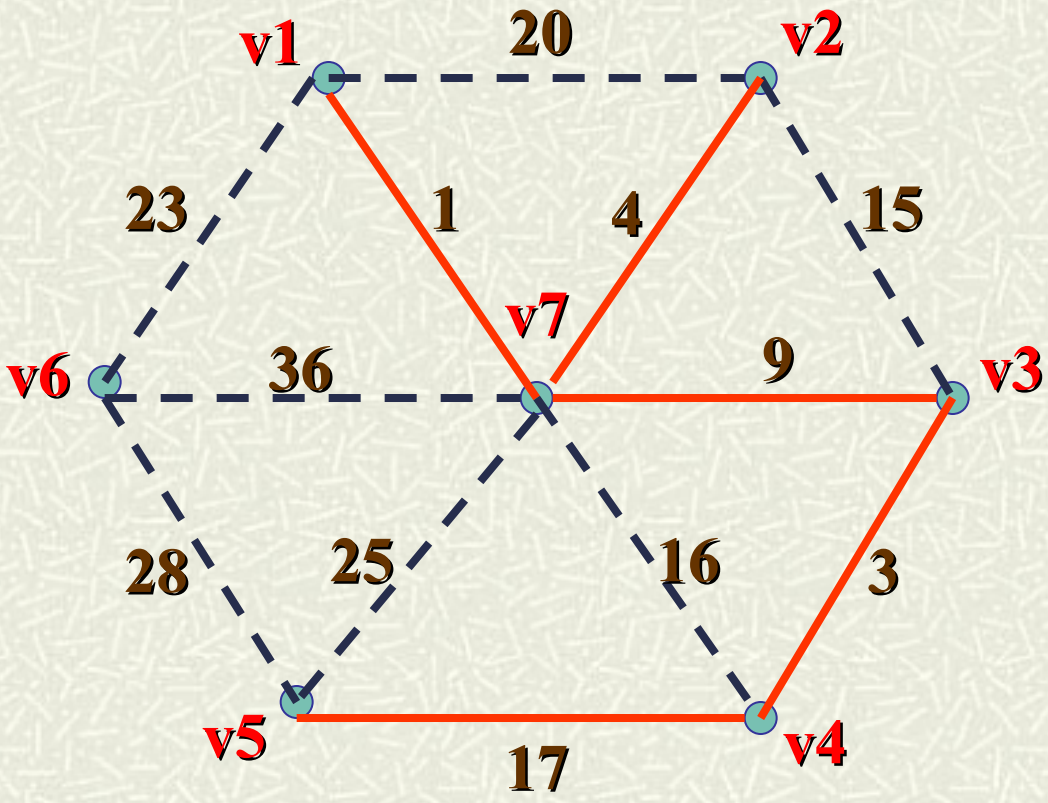
树与图的最小树



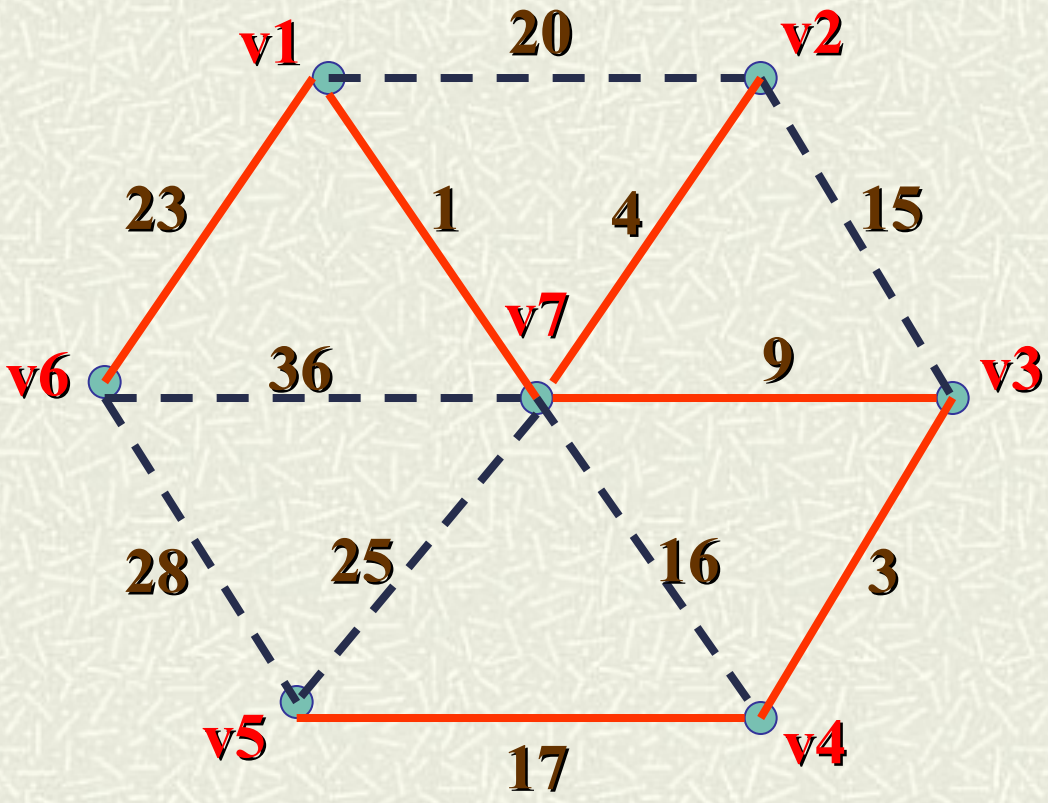
树与图的最小树



树与图的最小树



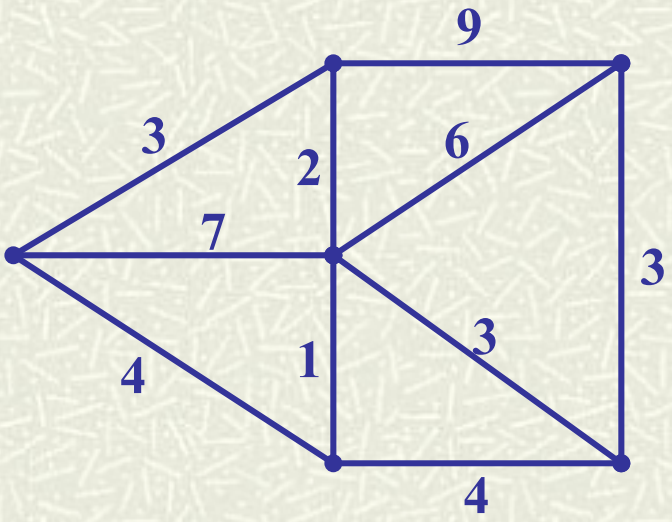
树与图的最小树

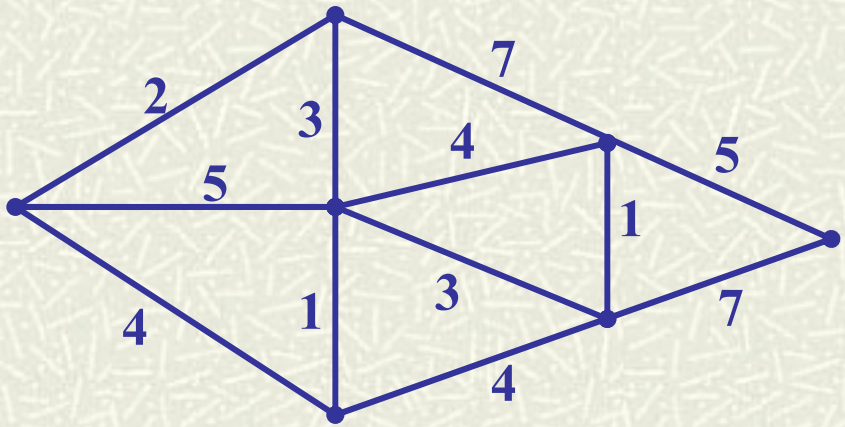


min=1+4+9+3+17+23=57

树与图的最小树

课堂练习：



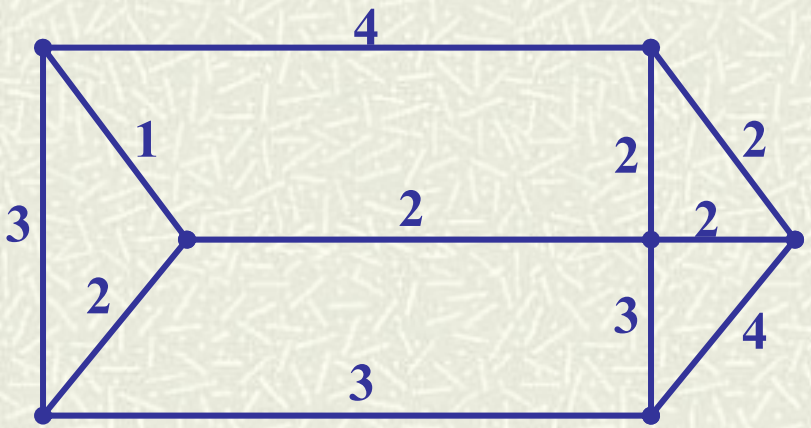


答案：

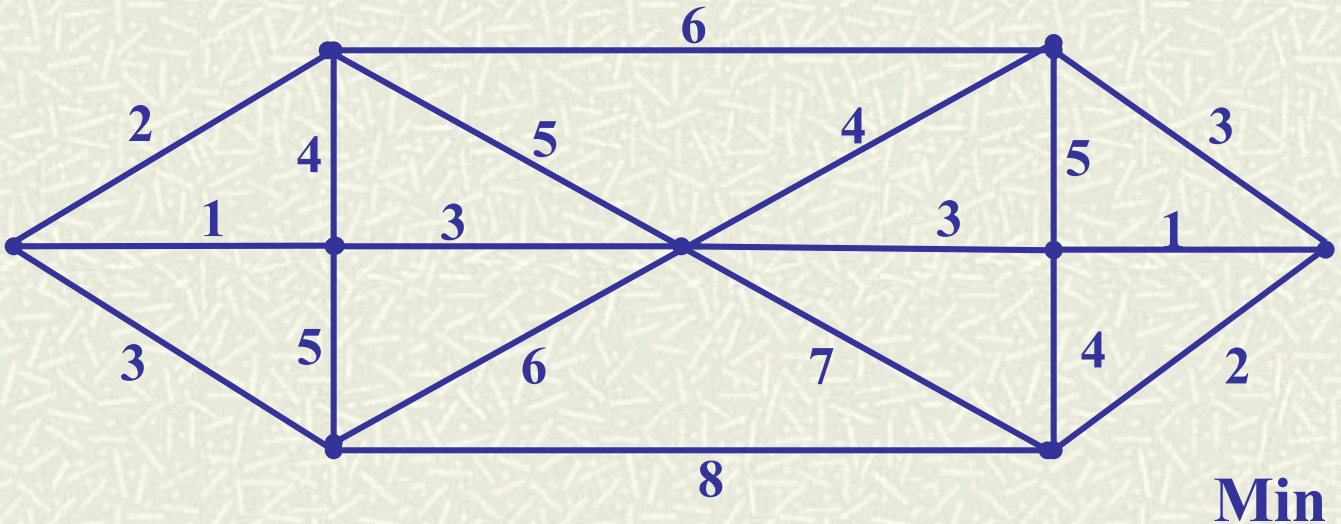
Min C(T)=12

Min C(T)=15

树与图的最小树



Min C(T)=12



Min C(T)=18

最短路问题

问题描述:

就是从给定的网络图中找出一点到各点或任意两点之间距离最短的一条路.

有些问题，如选址、管道铺设时的选线、设备更新、投资、某些整数规划和动态规划的问题，也可以归结为求最短路的问题。因此这类问题在生产实际中得到广泛应用。

最短路问题是图论中十分重要的最优化问题之一，它作为一个经常被用到的基本工具，可以解决生产实际中的许多问题，比如城市中的管道铺设，线路安排，工厂布局，设备更新等等; 也可以用于解决其它的最优化问题。

最短路问题

1. 设赋权有向图 D =(V,A), 对每个弧 a =(vi,vj), 相应地有权 w{ij}. vs, vt 是 D 中的两个顶点，P 是 D 中从 vs 到 vt 的任意一条路，定义 路的权 是 P 中所有弧权的和，记作 S(P).
2. 最短路问题就是找一条从 vs 到 vt 的路 P0, 使得

   $$S(P0) = \min\_P\ S(P).$$

   P0 叫做从 vs 到 vt 的最短路。P0 的权 S(P0)叫做从 vs 到 vt 的距离，记作d(vs,vt).
3. 由于 D 是有向图，d(vs,vt) 与 d(vt,vs) 一般不相等.

例6.4 渡河游戏

一老汉带了一只狼、一只羊、一棵白菜想要从南岸过河到北岸，河上只有一条独木舟，每次除了人以外，只能带一样东西；另外，如果人不在，狼就要吃羊，羊就要吃白菜，问应该怎样安排渡河，才能做到既把所有东西都运过河去，并且在河上来回次数最少？这个问题就可以用求最短路方法解决。

最短路问题

定义：

1） 点—— v_i 表示河岸的状态

2） 边—— e_k 表示由状态 v_i 经一次渡河到状态 v_j

3） 权——边 e_k 上的权定为 1

我们可以得到下面的加权有向图

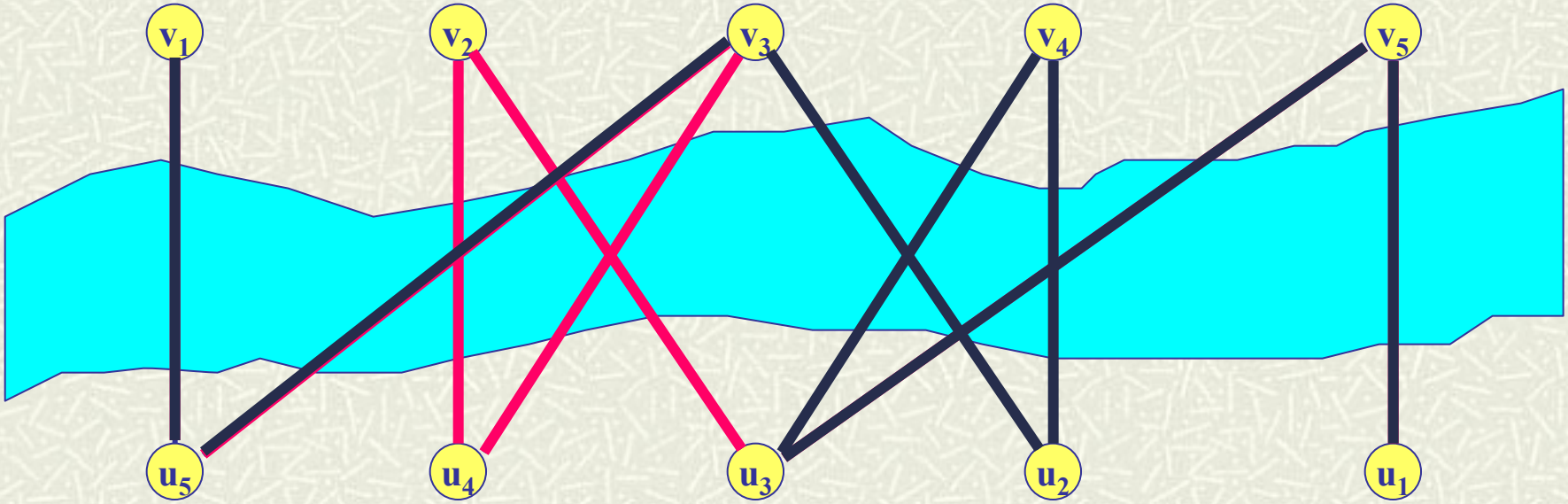
最短路问题

状态说明:

v_1, u_1 = (人,狼,羊,草); v_2, u_2 = (人,狼,羊); v_3, u_3 = (人,狼,草);

v_4, u_4= (人,羊,草); v_5, u_5 = (人,羊)

此游戏转化为在下面的二部图中求从 v_1 到 u_1 的最短路问题。



最短路问题的Dijkstra标号算法

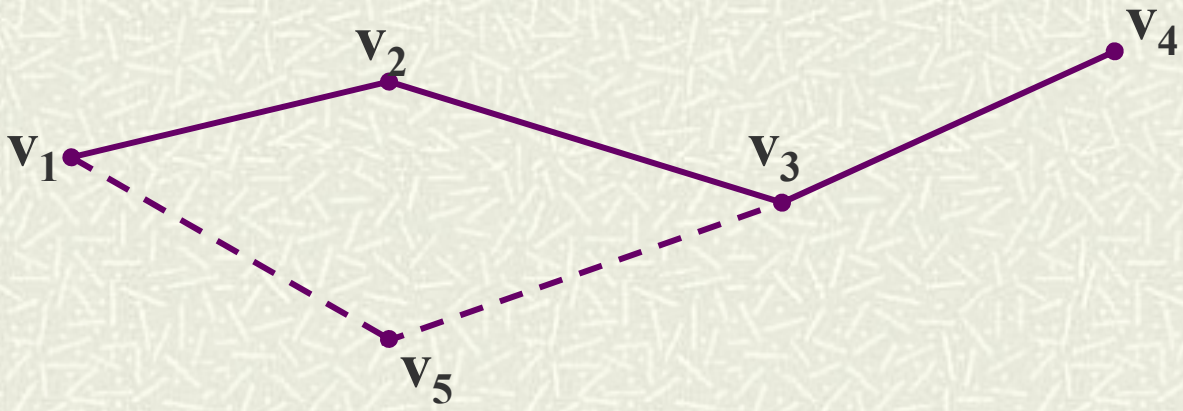
Edsger Wybe Dijkstra (1930-2002, 荷兰人) 计算机科学家、1972年获图灵奖，被誉为结构程序设计之父，与Donald Knuth并称为当代最伟大的计算机科学家。

他于1956年设计的最短路标号算法是迄今为止最有效地算法（非负权）。

最短路问题

若序列{ $v_1 \ldots v_{n-1}, v_n$ }是从v_1到v_n间的最短路，则序列{$v_1 \ldots v_{n-1}$ } 必为从v_1 到v_{n-1}的最短路。

假定$v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4$是$v_1 \rightarrow v_4$的最短路，则$v_1 \rightarrow v_2 \rightarrow v_3$一定是$v_1 \rightarrow v_3$的最短路，$v_2 \rightarrow v_3 \rightarrow v_4$也一定是$v_2 \rightarrow v_4$的最短路。



Dijkstra标号算法的基本思想

从vs出发，逐步地向外探寻最短路。执行过程中，与每个顶点对应，记录下一个数（称为这个点的标号），它或者表示从vs到该点的最短路的长度（称P标号，permanent），或者是从vs到该点的最短路长度的上界（称T标号，tentative, temporary）。方法的每一步是去修改T标号，并且把某一个T标号点改变为P标号点，从而使图中P标号点的个数增加一个，这样，至多经过p(G)-1步，就可以求出从vs到各点的最短路。

最短路问题的Dijkstra标号算法

求网络图的最短路，设图的起点是v_s, 终点是v_t ,以v_i为起点v_j为终点的弧记为 (i, j) , 距离记为d_{ij}

P标号(点标号): $b(i)$ 起点v_s到点vi的最短路的长度;

T标号(边标号): $k(i,j) = b(i) + d_{ij}$,

步骤:

1. 令起点的标号 $b(s) = 0$。

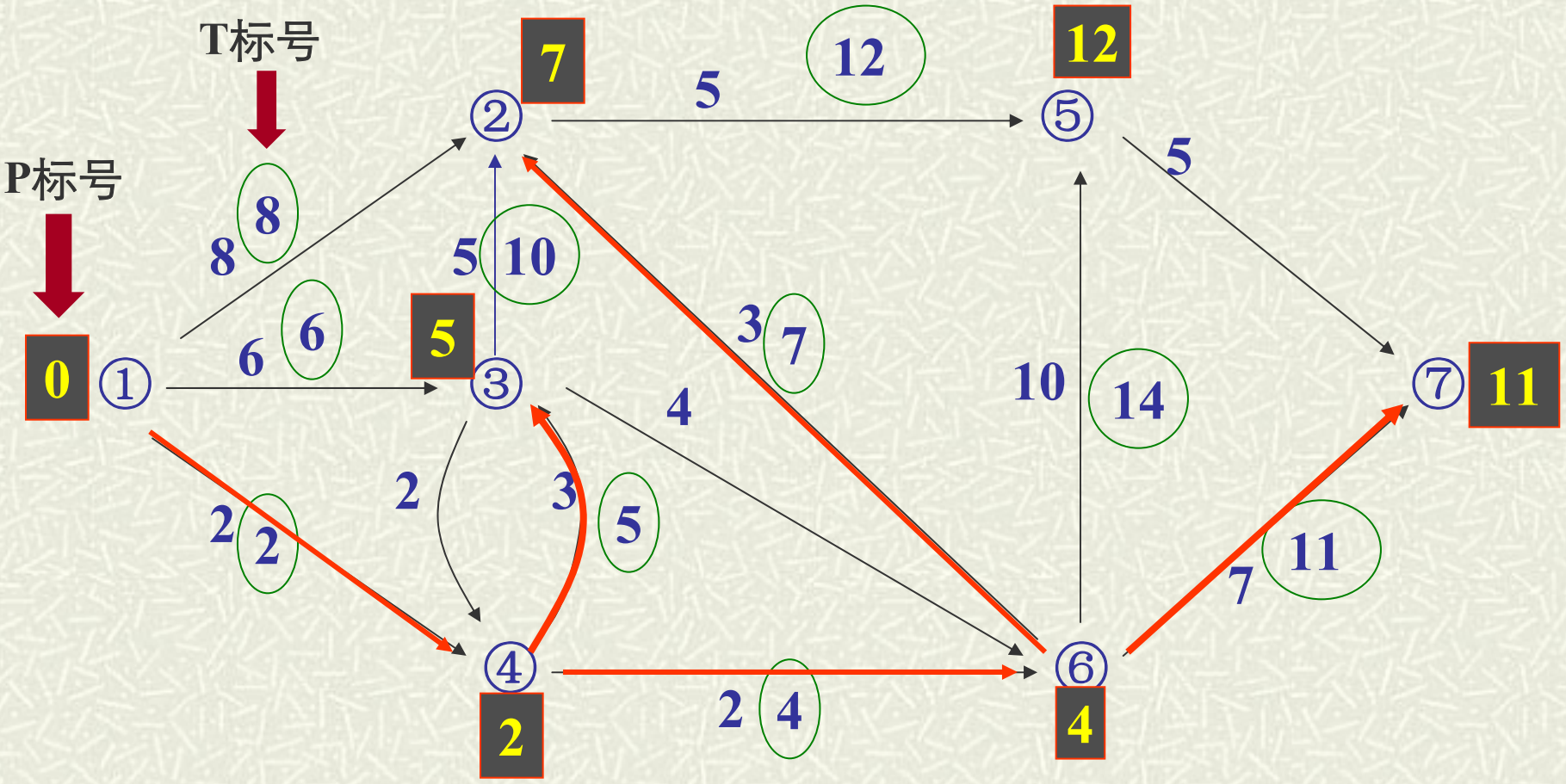
2. 找出所有v_i已标号v_j未标号的弧集合 $B=\{(i, j)\}$ 如果这样的弧不存在或v_t已标号则计算结束;

3. 计算集合B中弧 $k(i, j)=b(i)+d_{ij}$ 的标号

4. 选一个点标号 $b(l) = \min_j \{k(i, j) \mid (i, j) \in B\}$, 在终点$v_l$处标号b(l), 返回到第2步。

最短路问题

例6.5 求下图$v1$到$v7$的最短路长及最短路线



v_7已标号，计算结束。从$v1$到$v7$的最短路长是 11,

最短路线： $v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7$

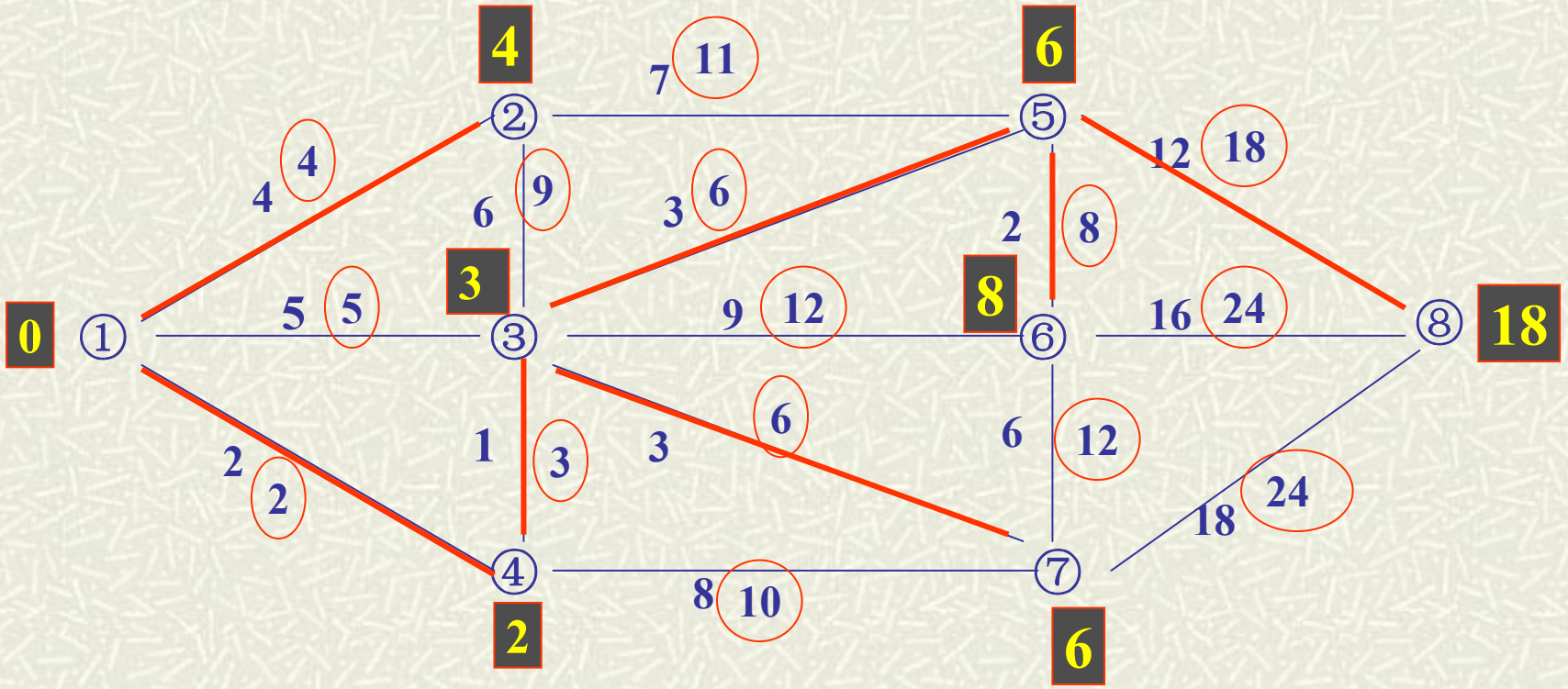
最短路问题

从上例知，只要某点已标号，说明已找到起点v_s到该点的最短路线及最短距离，因此可以将每个点标号，求出v_s到任意点的最短路线，如果某个点v_j不能标号，说明v_s不可达v_j。

注：无向图最短路的求法只需将上述步骤2中的弧改成边即可。

最短路问题

例6.6 求下图v_1到各点的最短距离及最短路线。

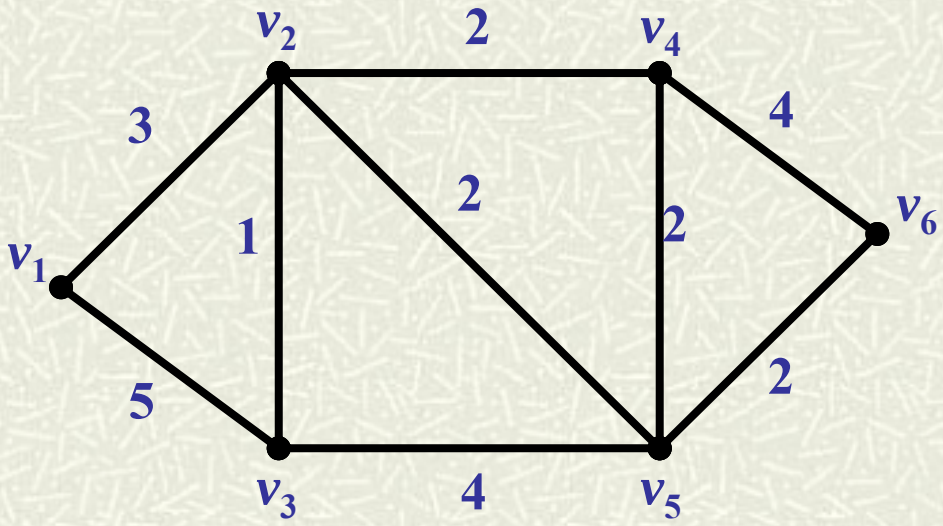


所有点都已标号，点上的标号就是v1到该点的最短距离，最短路线就是红色的链。

最短路问题

课堂练习：

1. 用Dijkstra算法求下图从v_1到v_6的最短距离及路线。

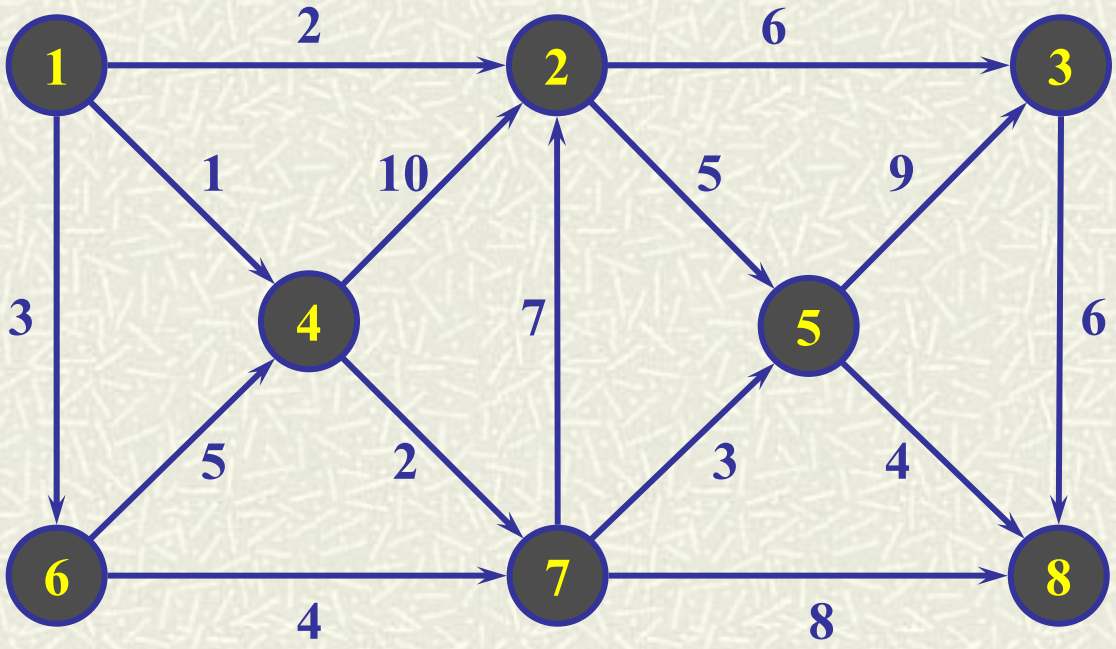


v_1到v_6的最短路为：

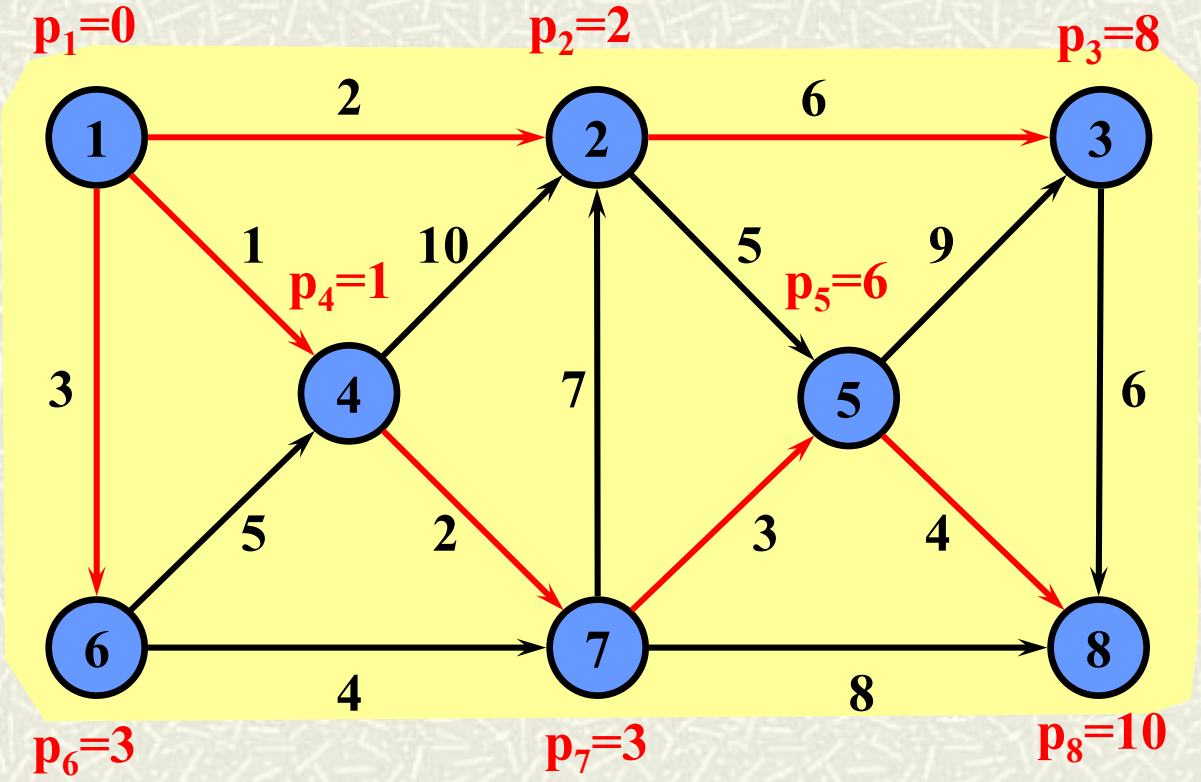
$$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6$$

最短路问题

2. 求从v1到v8的最短路径



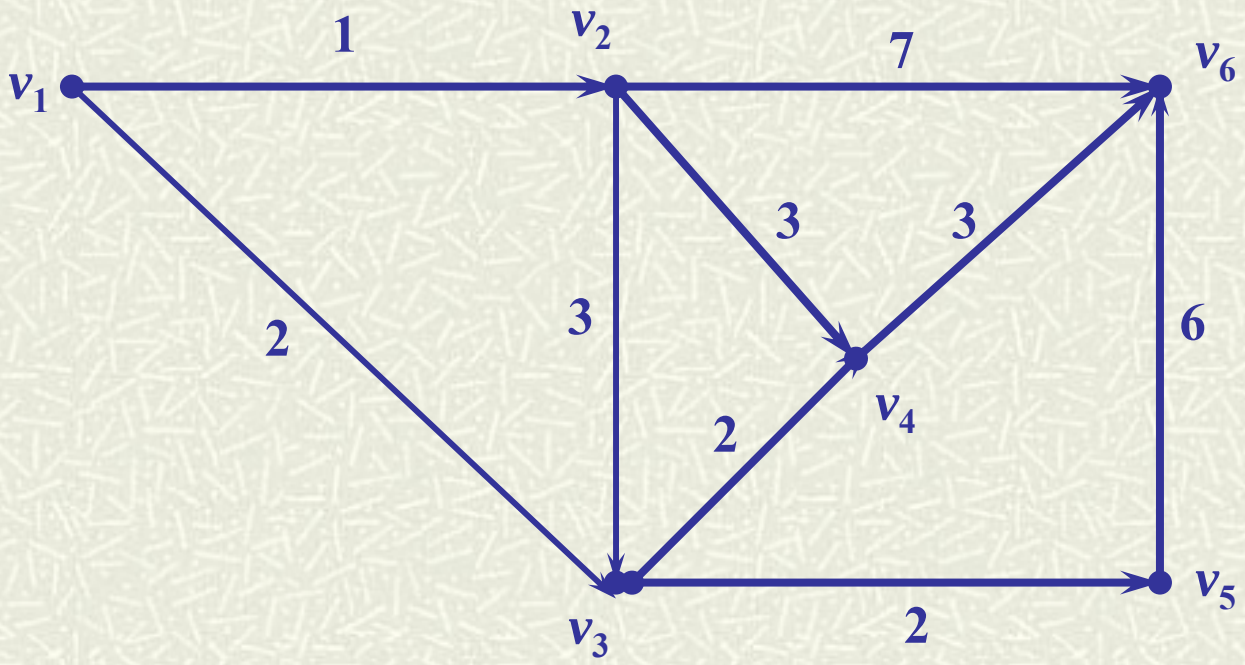
最短路问题



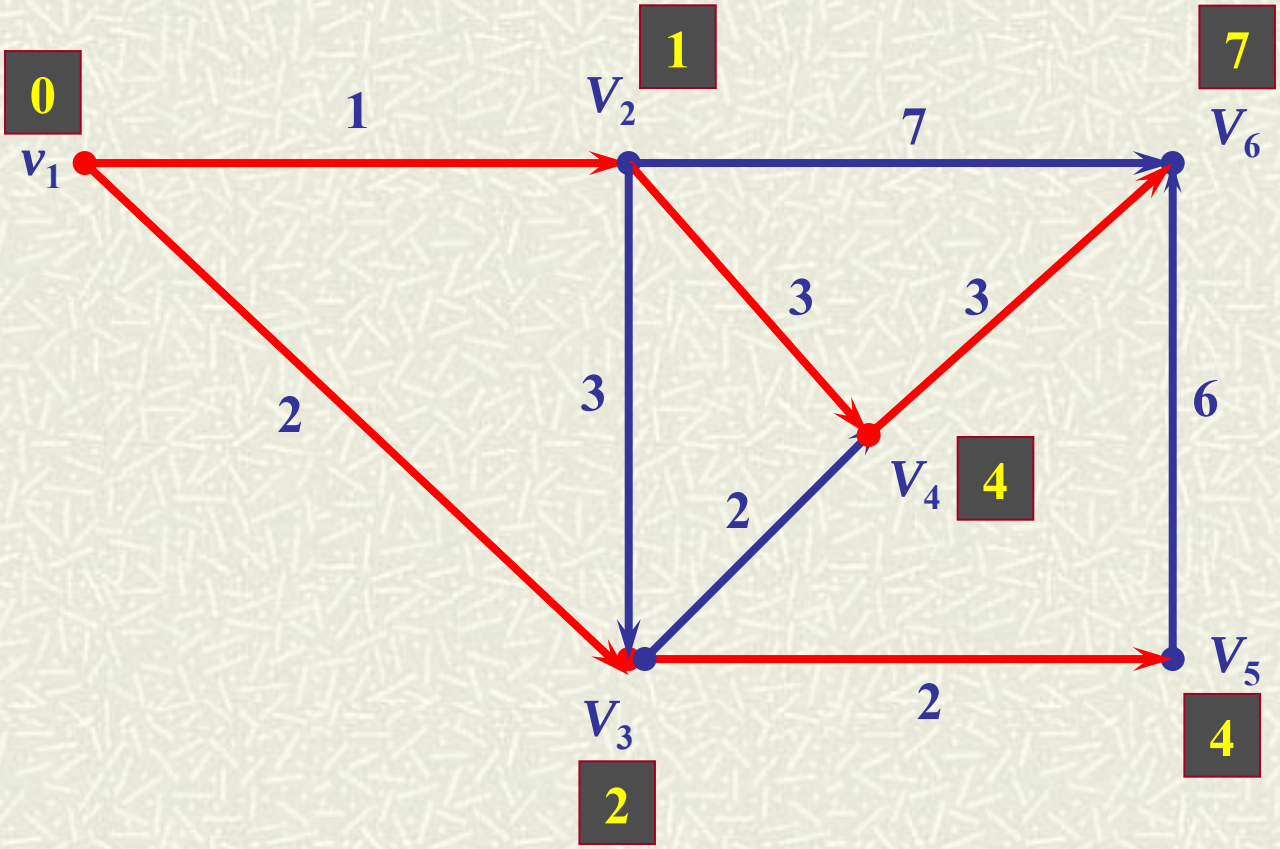
v_1到v_8的最短路径为$v_1 \rightarrow v_4 \rightarrow v_7 \rightarrow v_5 \rightarrow v_8$，最短距离为10

最短路问题

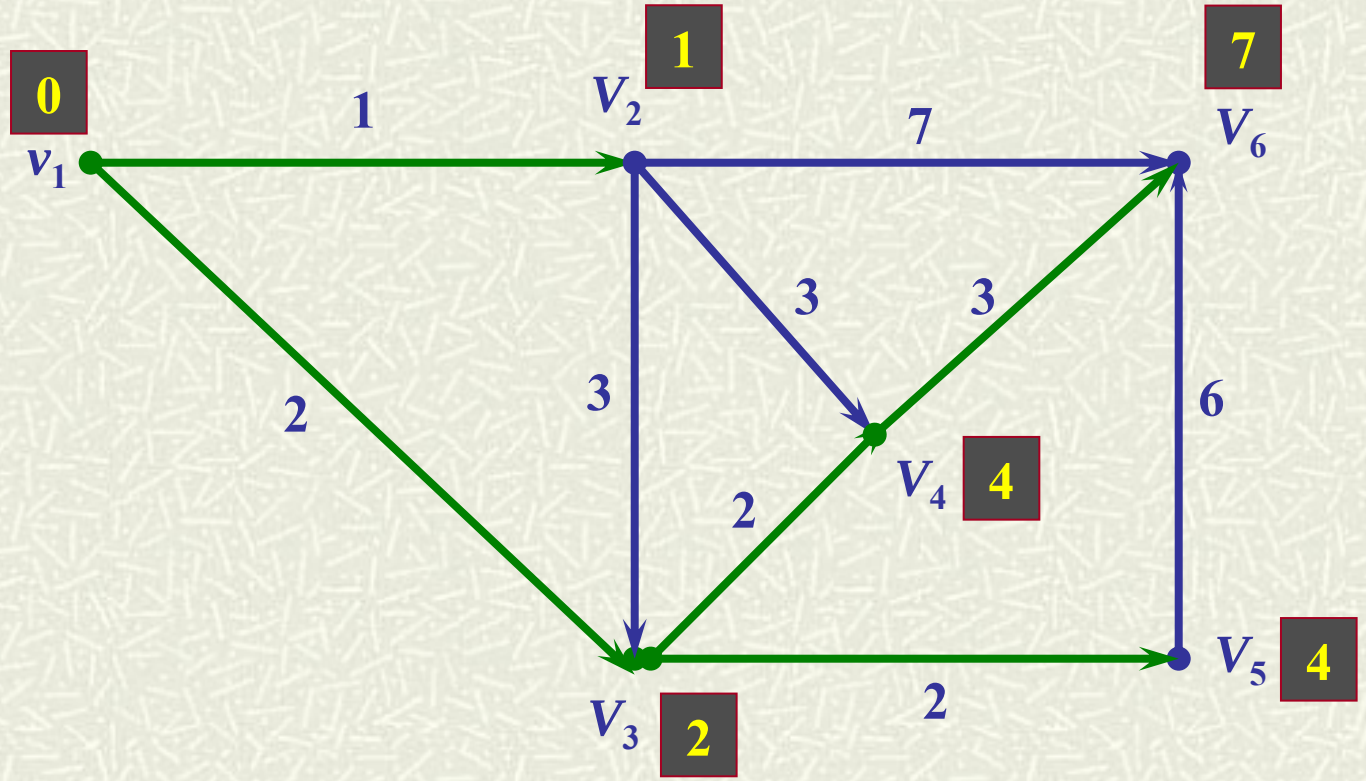
3. 求下图中v1点到另外任意一点的最短路径



最短路问题



最短路问题

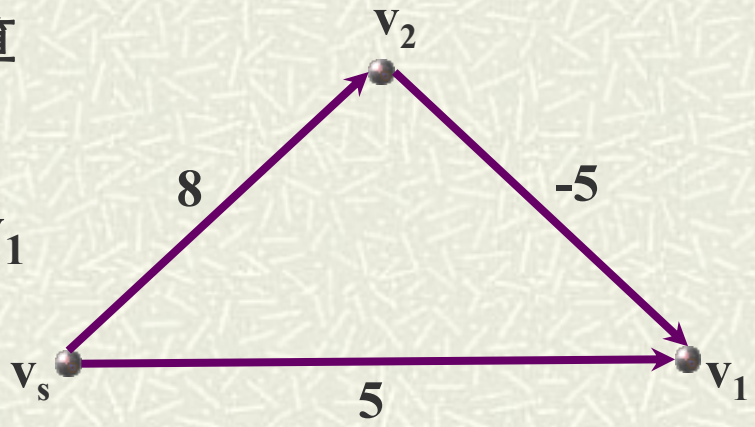


最短路问题

算法适用条件:

Dijkstra算法只适用于全部权为非负情况，如果某边上权为负的，算法失效。此时可采用逐次逼近算法 (参见清华大学运筹学教材，迭代算法）。

例6.7 如右图所示中按Dijkstra算法可得$P(v_1)=5$为从$v_s \rightarrow v_1$的最短路长显然是错误的，从$v_s \rightarrow v_2 \rightarrow v_1$路长只有3。



求最短路的逐次逼近法

首先，若$(v_i, v_j) \notin A$，则令$w_{ij} = +\infty$.

v_s到v_j的最短路必满足方程：$d(v_s, v_j) = \min_i \{d(v_s, v_i) + w_{ij}\}$.

为求得$d(v_s, v_1)$，$d(v_s, v_2)$，. . . , $d(v_s, v_p)$,采用如下递推公式：

$$d^{(1)}(v_s, v_j) = w_{sj},\ j = 1, 2, ..., p.$$

对$t = 2, 3, ...$

$$d^{(t)}(v_s, v_j) = \min_i \{d^{(t-1)}(v_s, v_i) + w_{ij}\},\ j = 1, 2, ..., p.$$

若进行到某一步时，比如第k步，有

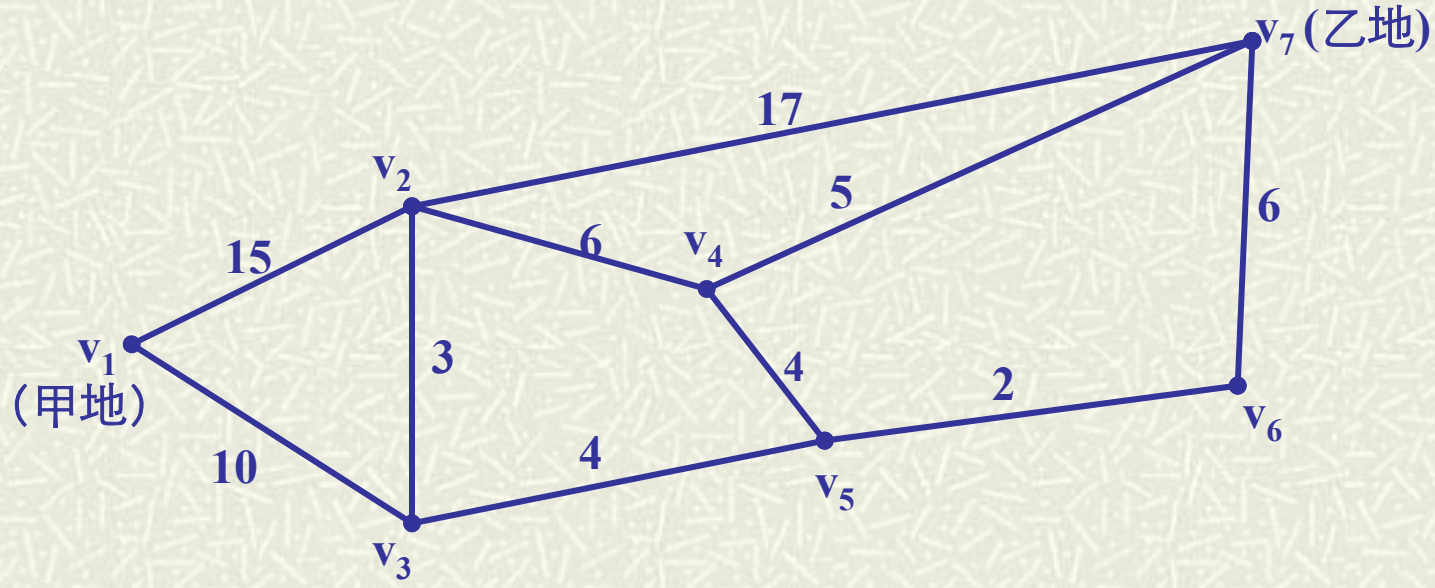
$$d^{(k)}(v_s, v_j) = d^{(k-1)}(v_s, v_j),\ j = 1, 2, ..., p,$$

则$\{d^{(k)}(v_s, v_j),\ j = 1, 2, ..., p\}$即为$v_s$到各点的最短路的长度。

最短路问题

最短路问题的应用：

例6.7 电信公司准备准备在甲、乙两地沿路架设一条光缆线，问如何架设使其光缆线路最短？下图给出了甲乙两地间的交通图。权数表示两地间公路的长度（单位：公里）。



最短路问题

解：这是一个求无向图的最短路的问题。

最短路问题

例6.8 设备更新问题。某公司使用一台设备，在每年年初，公司就要决定是购买新的设备还是继续使用旧设备。如果购置新设备，就要支付一定的购置费，当然新设备的维修费用就低。如果继续使用旧设备，可以省去购置费，但维修费用就高了。请设计一个五年之内的更新设备的计划，使得五年内购置费用和维修费用总的支付费用最小。已知：

设备每年年初的价格表

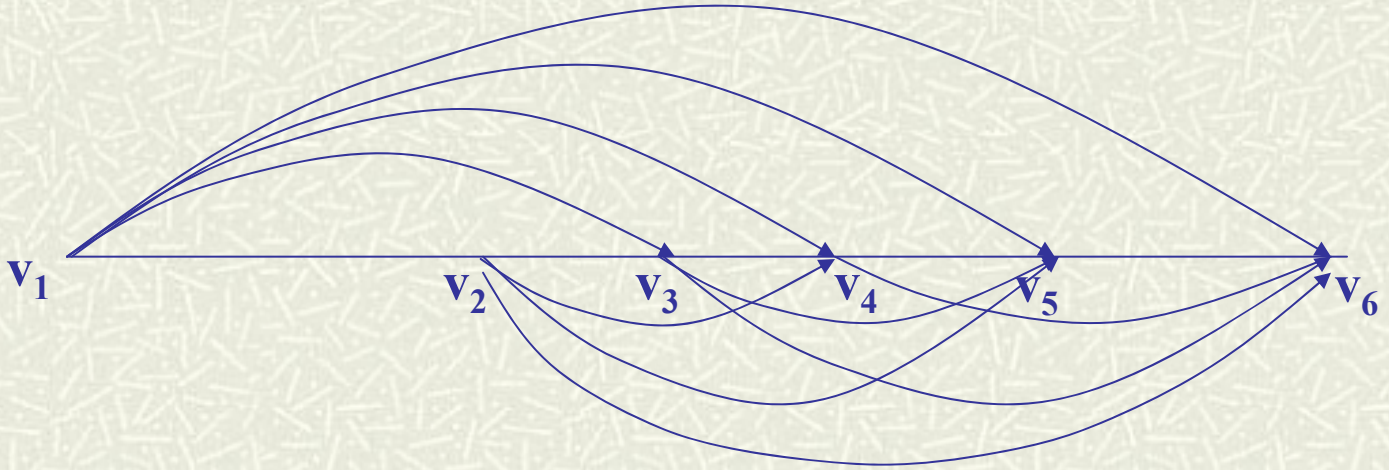
年份	1	2	3	4	5
年初价格	11	11	12	12	13

最短路问题

设备维修费如下表

使用年数	0-1	1-2	2-3	3-4	4-5
每年维修费用	5	6	8	11	18

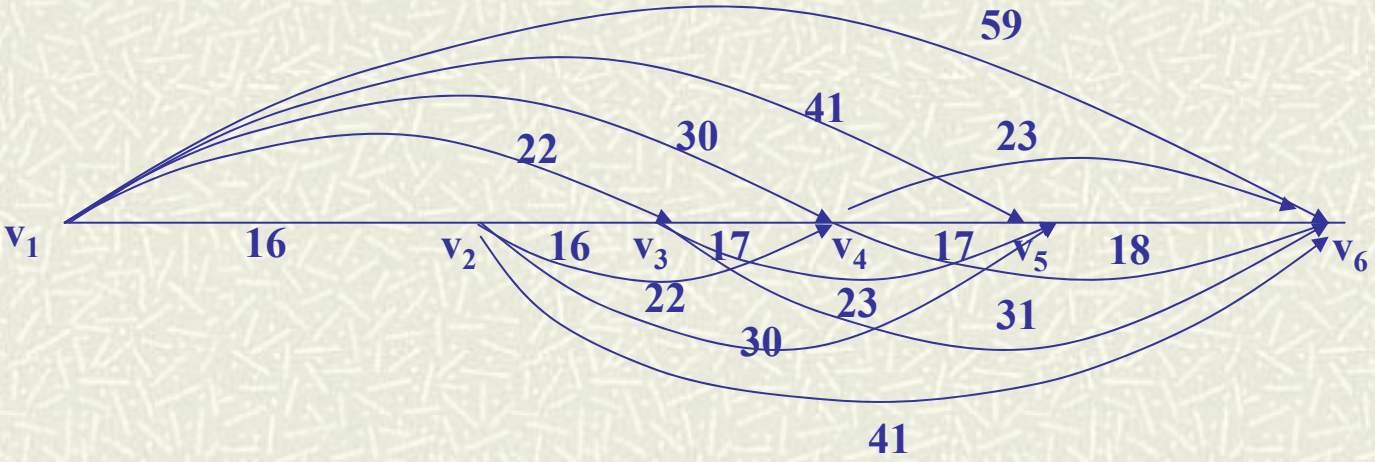
解：将问题转化为最短路问题，如下图：用v_i表示“第i年年初购进一台新设备”，弧（vi,vj）表示第i年年初购进的设备一直使用到第j年年初。



最短路问题

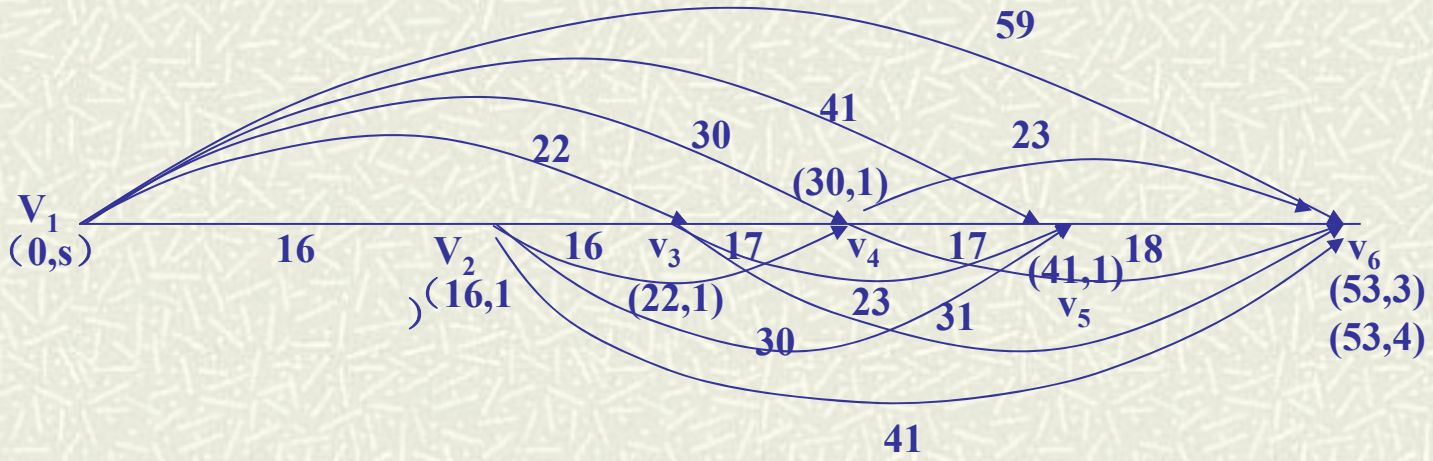
把所有弧的权数计算如下表，把权数赋到图中，再用Dijkstra算法求最短路。

	1	2	3	4	5	6
1		16	22	30	41	59
2			16	22	30	41
3				17	23	31
4					17	23
5						18
6						



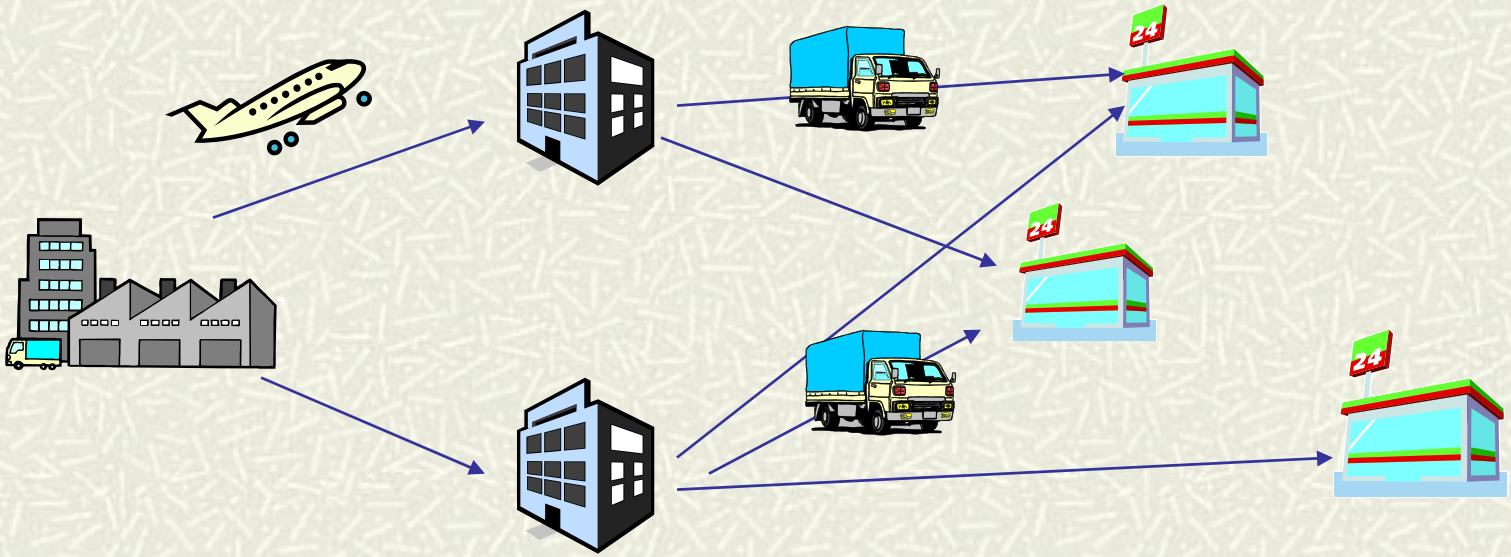
最短路问题

最终得到下图，可知，v_1到v_6的距离是53，最短路径有两条： $v_1 \rightarrow v_3 \rightarrow v_6$和 $v_1 \rightarrow v_4 \rightarrow v_6$



网络的最大流

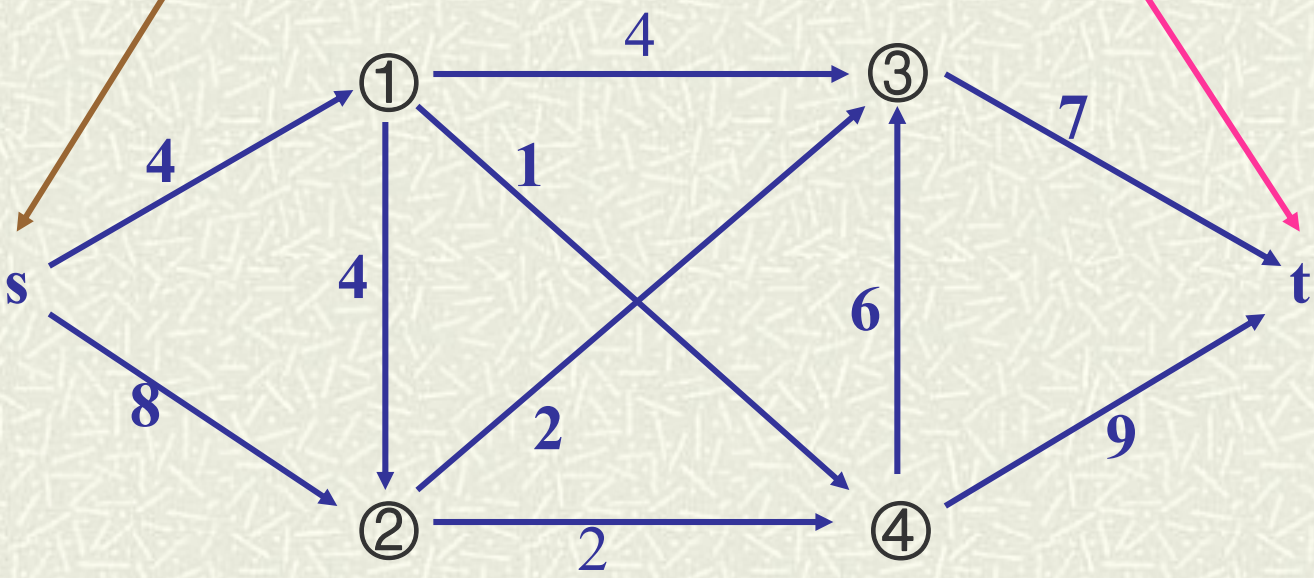
如何制定一个运输计划使生产地到销售地的产品输送量最大。这就是一个网络最大流问题。



网络的最大流

基本概念：

1. 容量网络： 网络上的每条弧(v_i, v_j)都给出一个最大的通过能力，称为该弧的**容量**，简记为c_{ij}。容量网络中通常规定一个**发点**(也称源点，记为s)和一个**收点**(也称汇点，记为t)，网络中其他点称为**中间点**。



网络的最大流

2. 网络的最大流

是指网络（带流量限制或赋权的有向图）中从发点到收点之间允许通过的最大流量。

3. 流与可行流

流是指加在网络各条弧上的实际流量，对加在弧(v_i,v_j)上的负载量记为f_{ij}。若$f_{ij}=0$，称为零流。

满足以下条件的一组流称为可行流。

- 容量限制条件。容量网络上所有的弧满足：$0 \leqslant f_{ij} \leqslant c_{ij}$
- 中间点平衡条件。

$$\sum_{j:(i,j)\in A} f(v_i,v_j) - \sum_{k:(k,i)\in A} f(v_k,v_i) = 0 \quad (i \neq s,t)$$

- 若以v(f)表示网络中从s→t的流量，则有：

$$v(f) = \sum_{j:(s,j)\in A} f(v_s,v_j) - \sum_{k:(k,t)\in A} f(v_k,v_t) = 0$$

网络的最大流

任何网络上一定存在可行流。显然，零流即是可行流。

网络最大流问题：使 v(f) 值达到最大的可行流.

网络最大流问题是线性规划的特殊情形:

Max v(f), subject to {f_{ij}} 满足可行流条件。

网络的最大流

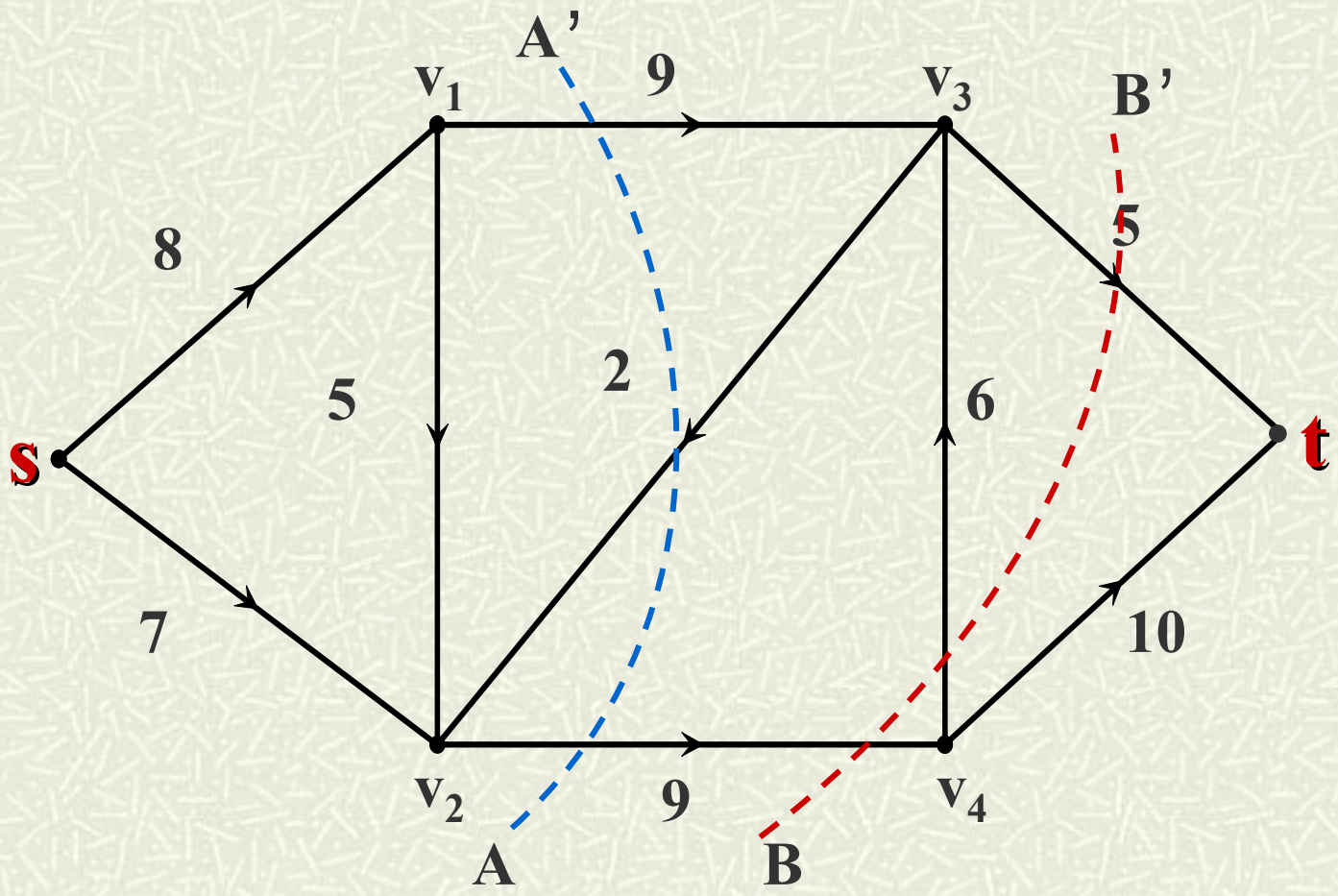
- 割与割集

割（截）是指容量网络中将发点和收点分割开，并使s→t的流中断的一组弧的集合。割容量（截量）是组成割集合中的各条弧的容量之和，用 $c(V,\overline{V})$ 表示。

$$c(V,\overline{V}) = \sum_{(i,j)\in(V,\overline{V})} c(v_i,v_j)$$

如下图中，AA′ 将网络上的点分割成 $V,\overline{V}$ 两个集合。并有 $s\in V, t\in\overline{V}$ ，称弧的集合$\{(v_1,v_3),(v_2,v_4)\}$是一个割，且 $V\to\overline{V}$ 的流量为18。

网络的最大流



网络的最大流

定理1 设网络N中一个从 s 到 t 的可行流 f 的流量为$v(f)$，(V, V')为N的任意一个割集，则

$$v(f) = f(V, V') - f(V', V)$$

推论1 网络 N中任意可行流 f 的流量 $v(f)$ 和割集 (V, V')，有

$$v(f) \le c(V, V')$$

[证明] $v(f) = f(V, V') - f(V', V) \le f(V, V') \le c(V, V')$

推论2 最大流量$v^*(f)$不大于最小割集的容量，即：

$$v^*(f) \le \min\{c(V, V')\}$$

定理2 （最大流最小割定理） 在网络中s→t的最大流量等于它的最小割集的容量，即：$v^*(f) = c^*(V, V')$

网络的最大流

定义（增广链）

设f是一个可行流，μ是从发点vs到收点vt的一条链。若μ满足下列条件，则称之为（关于可行流f的）增广链：

（1） 在该链上所有指向为s→t的弧 （称为前向弧，顺向弧，记作μ^+）*都非饱和，即*

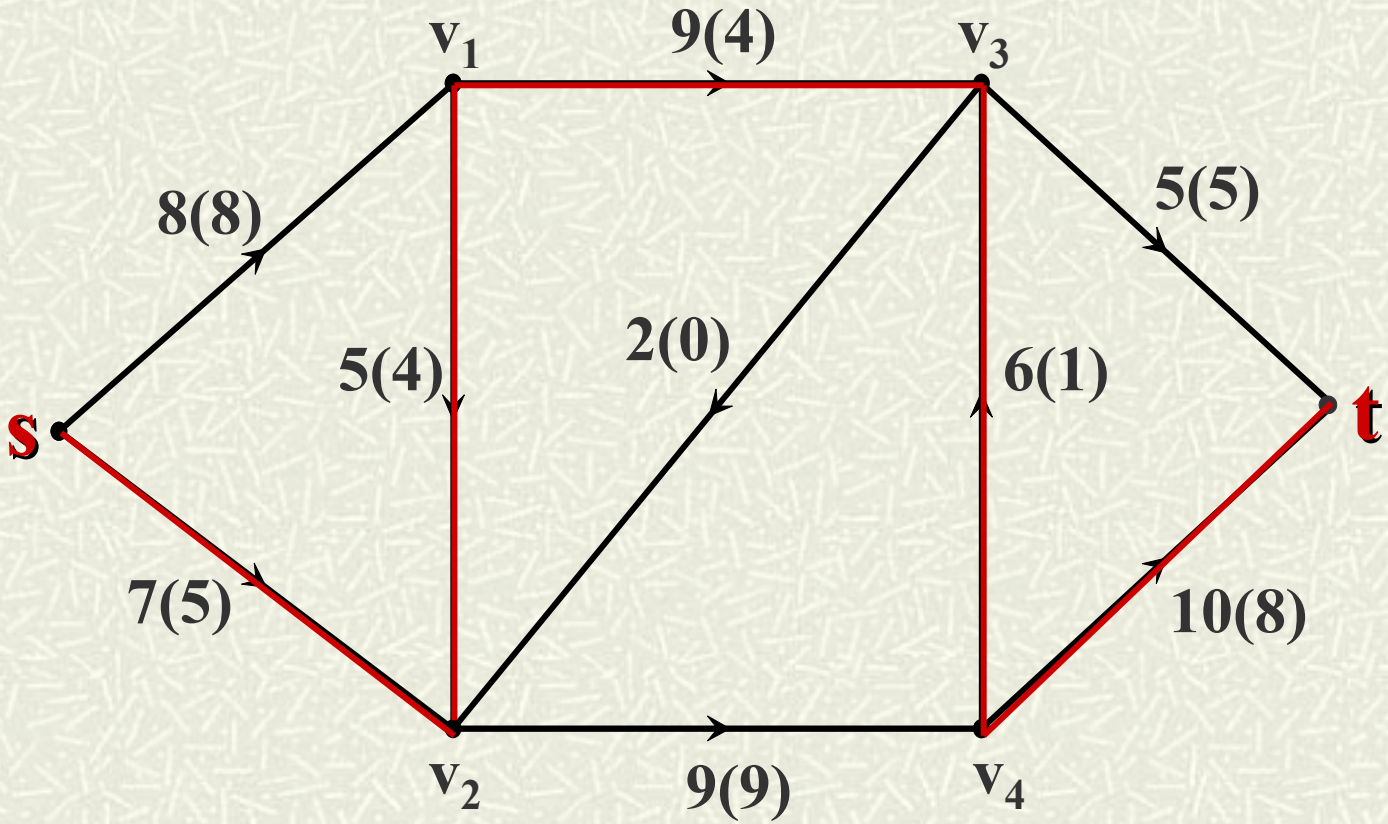
$$(v_i, v_j) \in \mu^+ \Rightarrow 0 \le f_{ij} < c_{ij}.$$

(2) 在该链上所有指向为t→s的弧 （称为后向弧，逆向弧，记作μ^-）*流量均大于零（非零流），即*

$$(v_i, v_j) \in \mu^- \Rightarrow 0 < f_{ij} \le c_{ij}.$$

定理3 网络N中的可行流 *f* 是最大流当且仅当N中不包含 *f* 的增广链。

网络的最大流



括号中的数字给出一个可行流，S→v2→v1→v3→v4→t为该可行流的一个增广链（增流链）。

网络的最大流

求网络最大流的 Ford-Fulkerson 标号（原始对偶）算法:

[基本思想]

由一个流开始，系统地搜寻增广链，然后在此链上增流，继续这个增流过程，直至不存在增广链。

[基本方法] (Ford-Fulkerson, 1962)

(1) 找出第一个可行流，（例如所有弧的流量f_{ij}=0。）

(2) 用标号的方法找一条增广链

- 首先给发点 s 标号(∞), 标号中的数字表示允许的最大调整量。
- 选择一个点 *vi* (已标号) 并且另一端未标号的弧沿着某条链向收点 t 检查:

网络的最大流

- 如果弧的起点为v_i，并且有$f_{ij}<C_{ij}$，则给v_j标号为$(C_{ij}-f_{ij})$
- 如果弧的方向指向v_i，并且有$f_{ji}>0$，则v_j标号(f_{ji})

(3) 重复第(2)步，可能出现两种结局：

- 标号过程中断，t无法标号，说明网络中不存在增广链，目前流量为最大流。同时可以确定最小割集，记已标号的点集为V, 未标号的点集合为V′，则 (V,V′)为网络的最小割。
- t得到标号，反向追踪在网络中找到一条从s到t的由标号点及相应的弧连接而成的增广链。继续第(4)步

(4) 修改流量。设原图可行流为f，令

$$f'=\begin{cases} f+\varepsilon & \text{对增广链上所有前向弧} \\ f-\varepsilon & \text{对增广链上所有后向弧} \\ f & \text{所有非增广链上的弧} \end{cases}$$

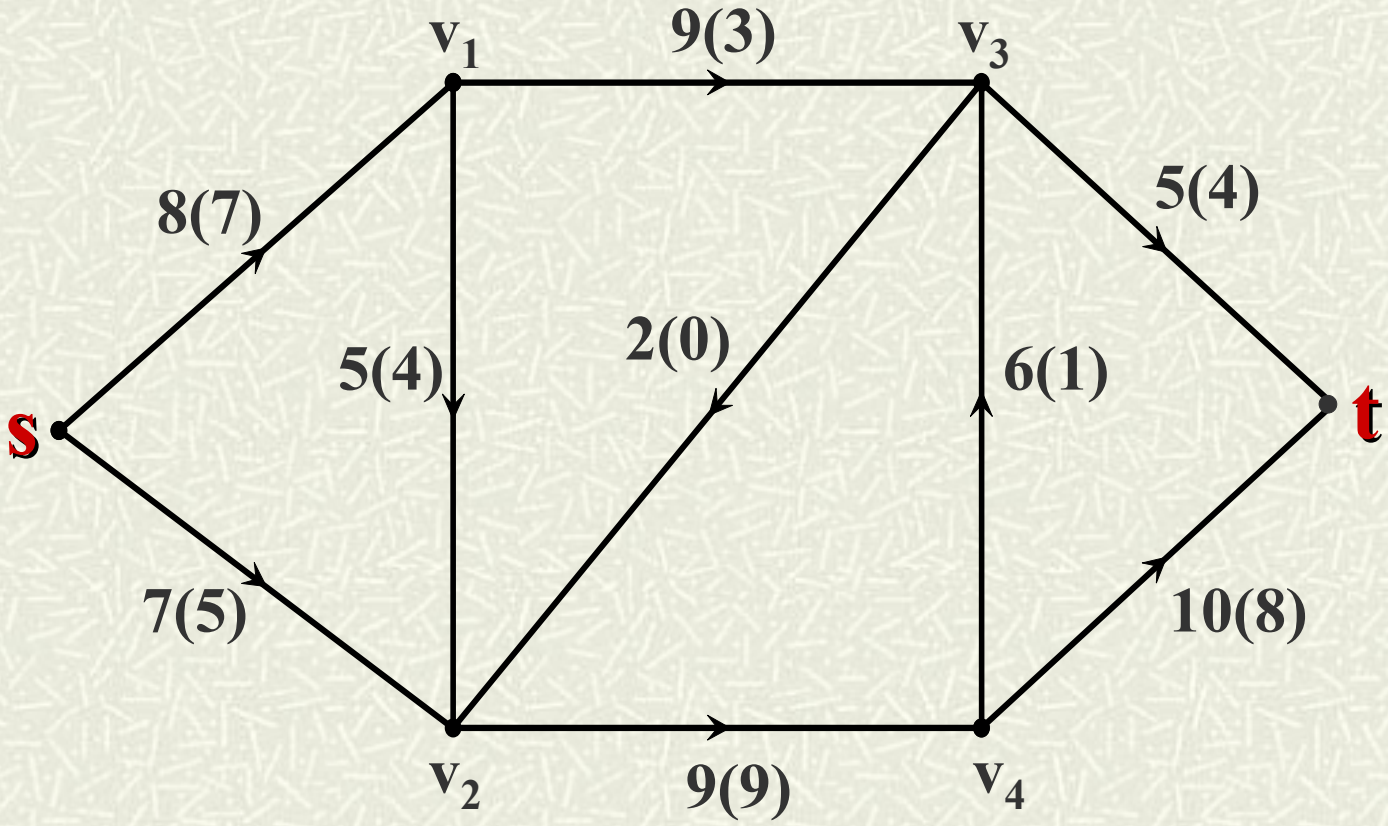
得到网络上一个新的可行流f’.其中

$$\varepsilon=\min\{\min_{\mu^+}\{c_{ij}-f_{ij}\},\min_{\mu^-}\{f_{ij}\}\}.$$

(5) 擦除图上所有标号，重复(1)-(4)步，直到图中找不到任何增广链，计算结束。

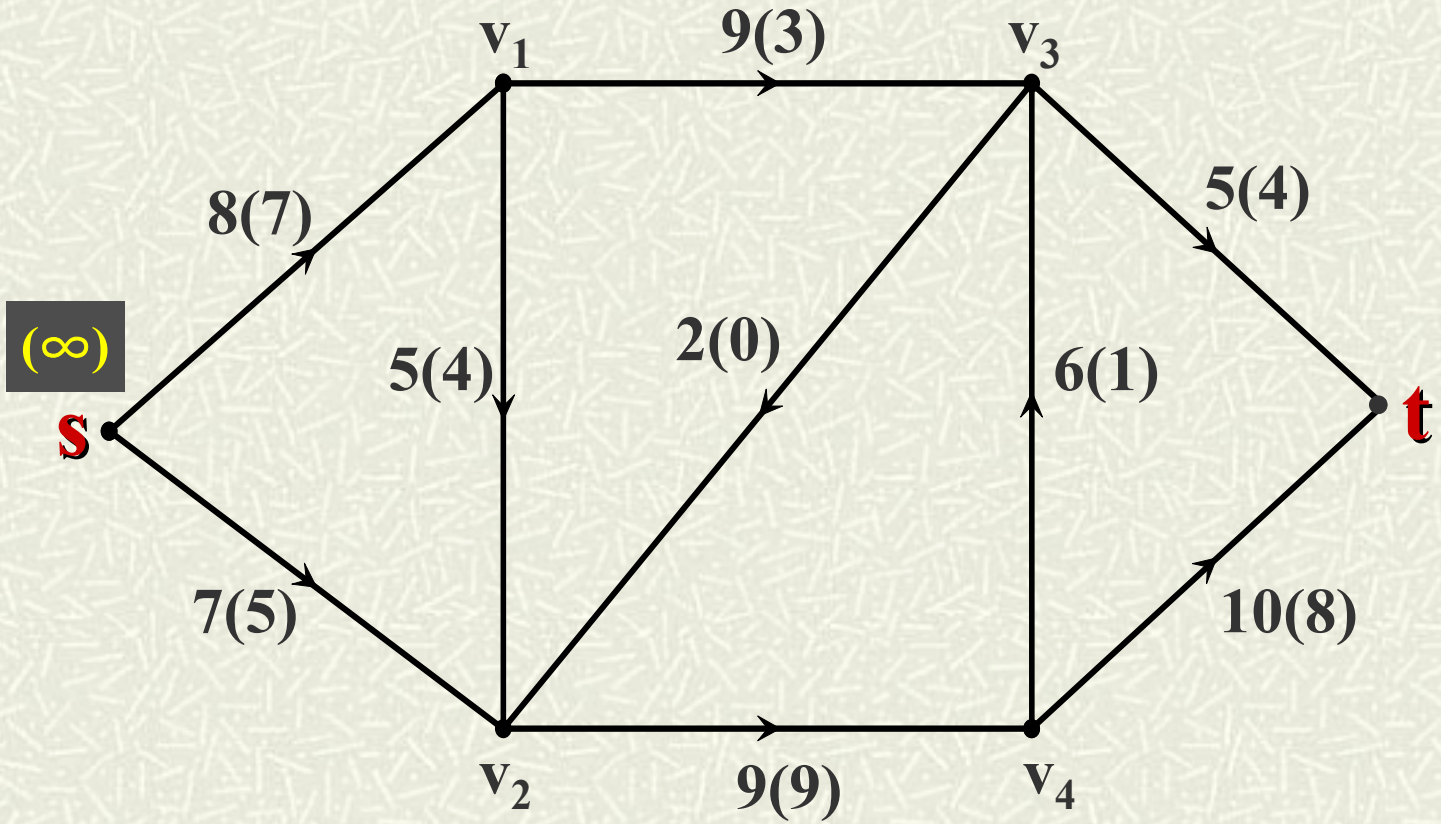
网络的最大流

例6.10 用标号算法求下图中s→t的最大流量，并找出最小割。



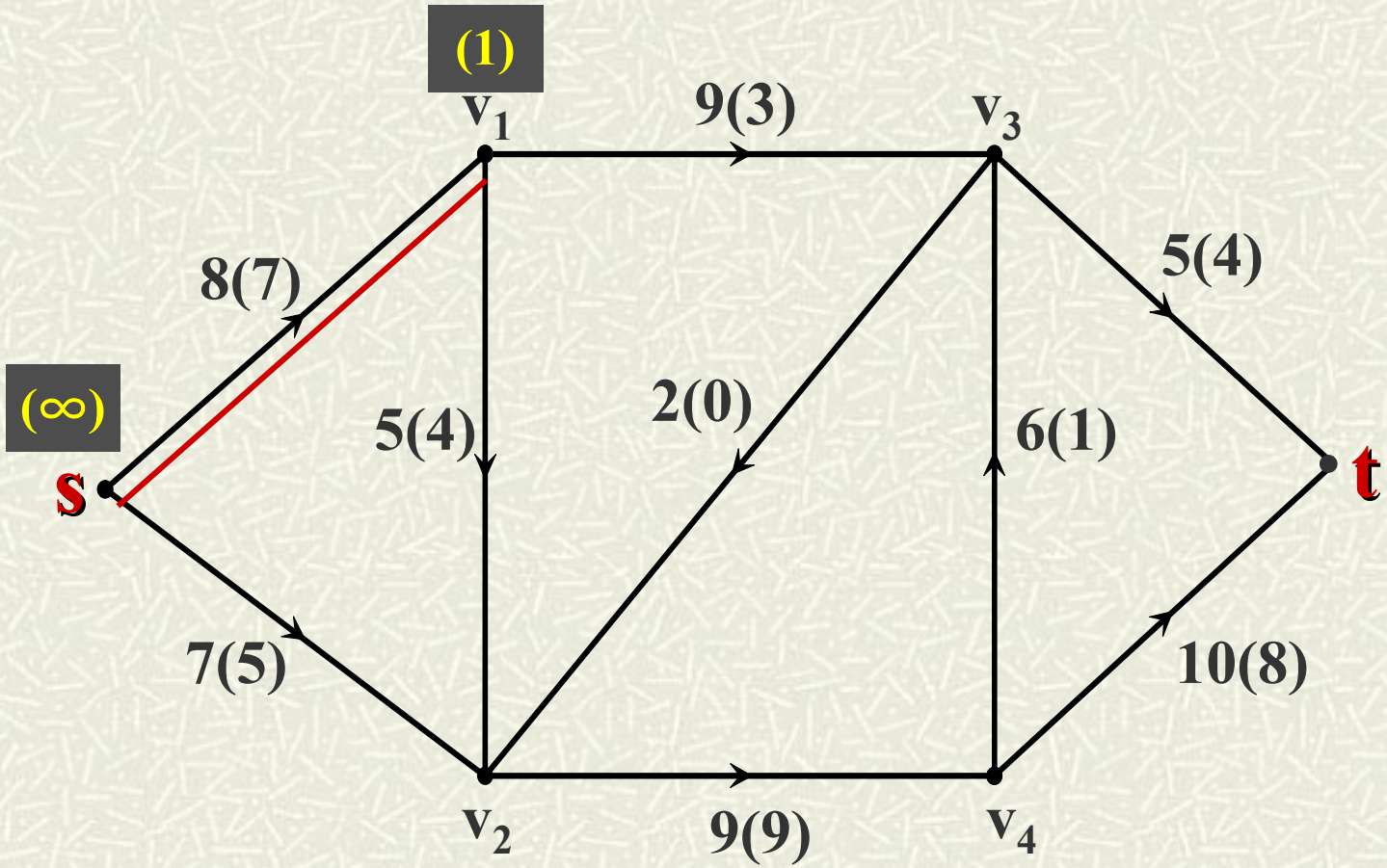
网络的最大流

解：(1) 先给s标号(∞)



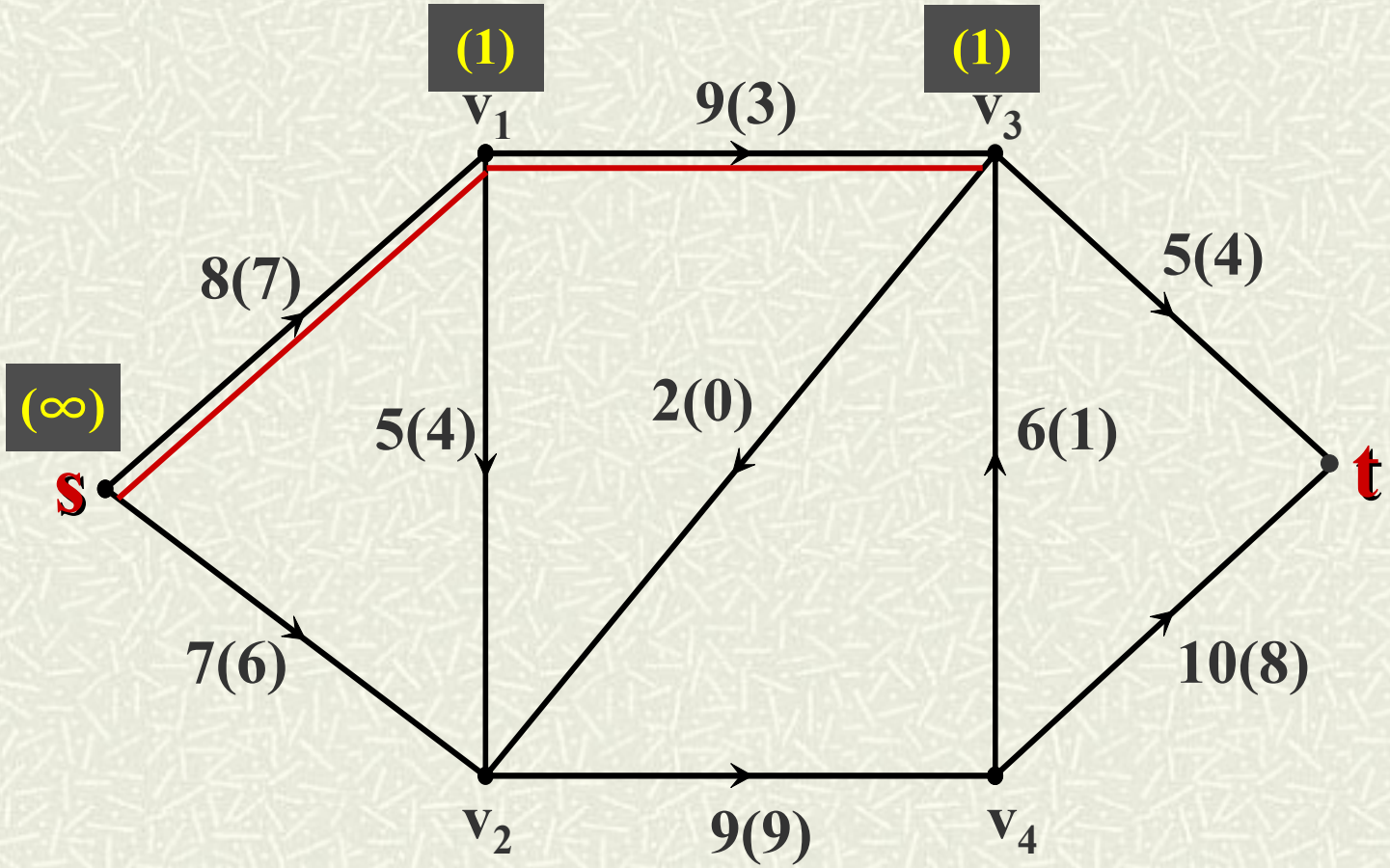
网络的最大流

(2) 检查与s点相邻的未标号的点，因$f_{s1}<c_{s1}$，故对v_1标号 $\varepsilon(1)=\min\{\infty, c_{s1}-f_{s1}\}=1$,



网络的最大流

(2) 检查与v_1点相邻的未标号的点，因$f_{13}<c_{13}$，故对v_3标号$\varepsilon(3)=\min\{1, c_{13}-f_{13}\}= \min\{1, 6\}= 1$

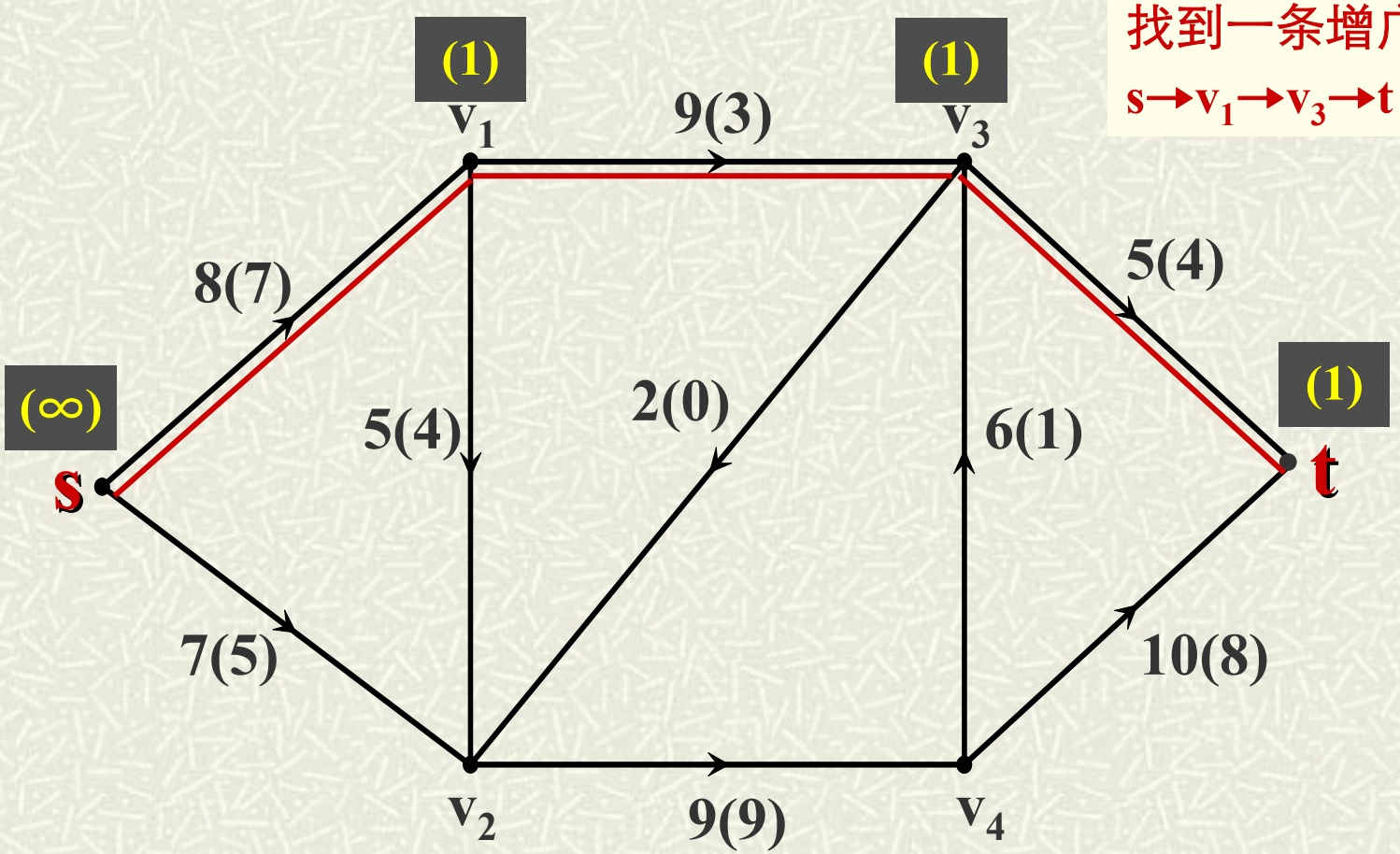


网络的最大流

(3) 检查与v_3点相邻的未标号的点，因$f_{3t}<c_{3t}$，故对v_t标号 $\varepsilon(t)$ $=\min\{1, c_{3t}-f_{3t}\}= \min\{1, 1\}= 1$

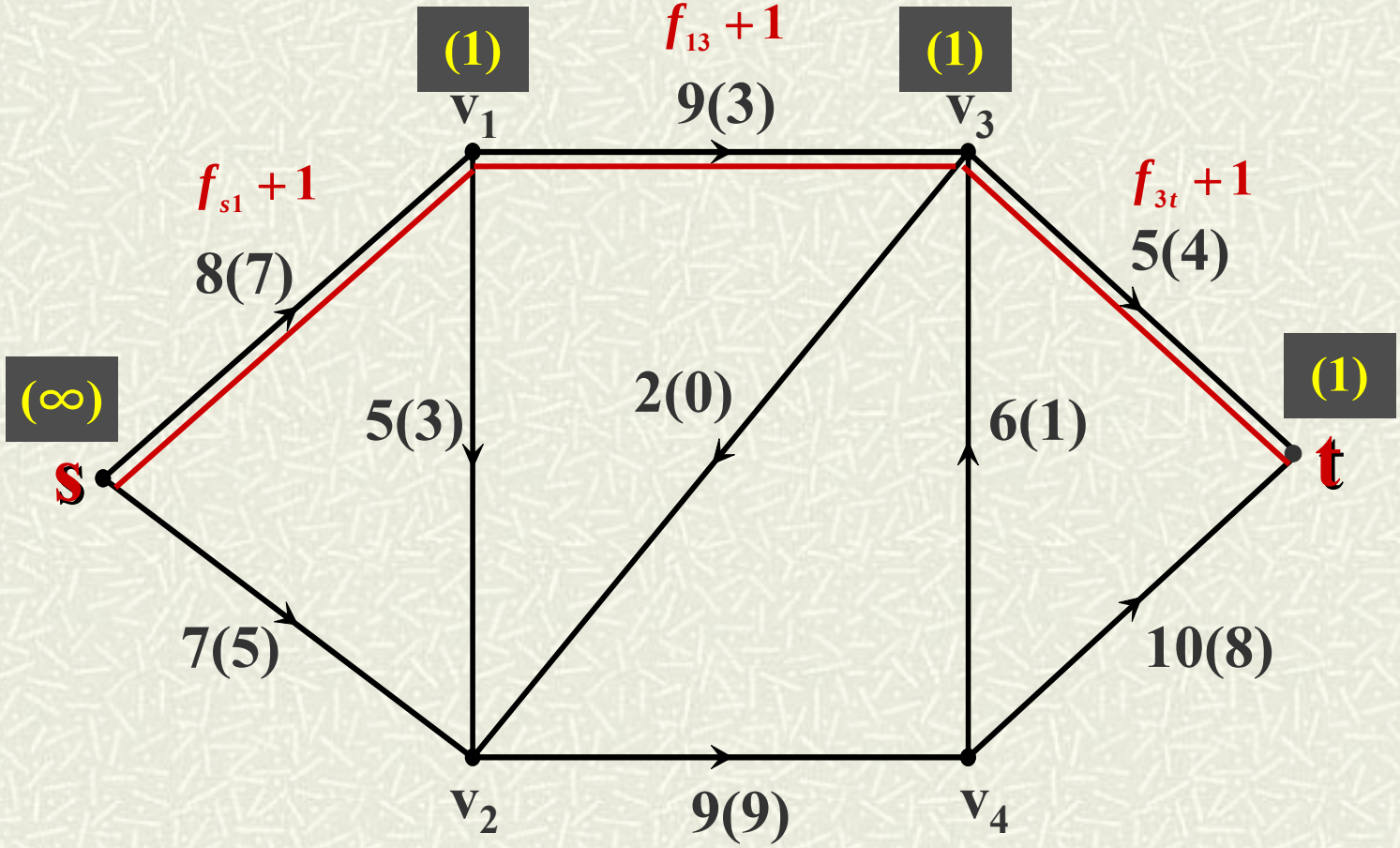
找到一条增广链

$s \rightarrow v_1 \rightarrow v_3 \rightarrow t$



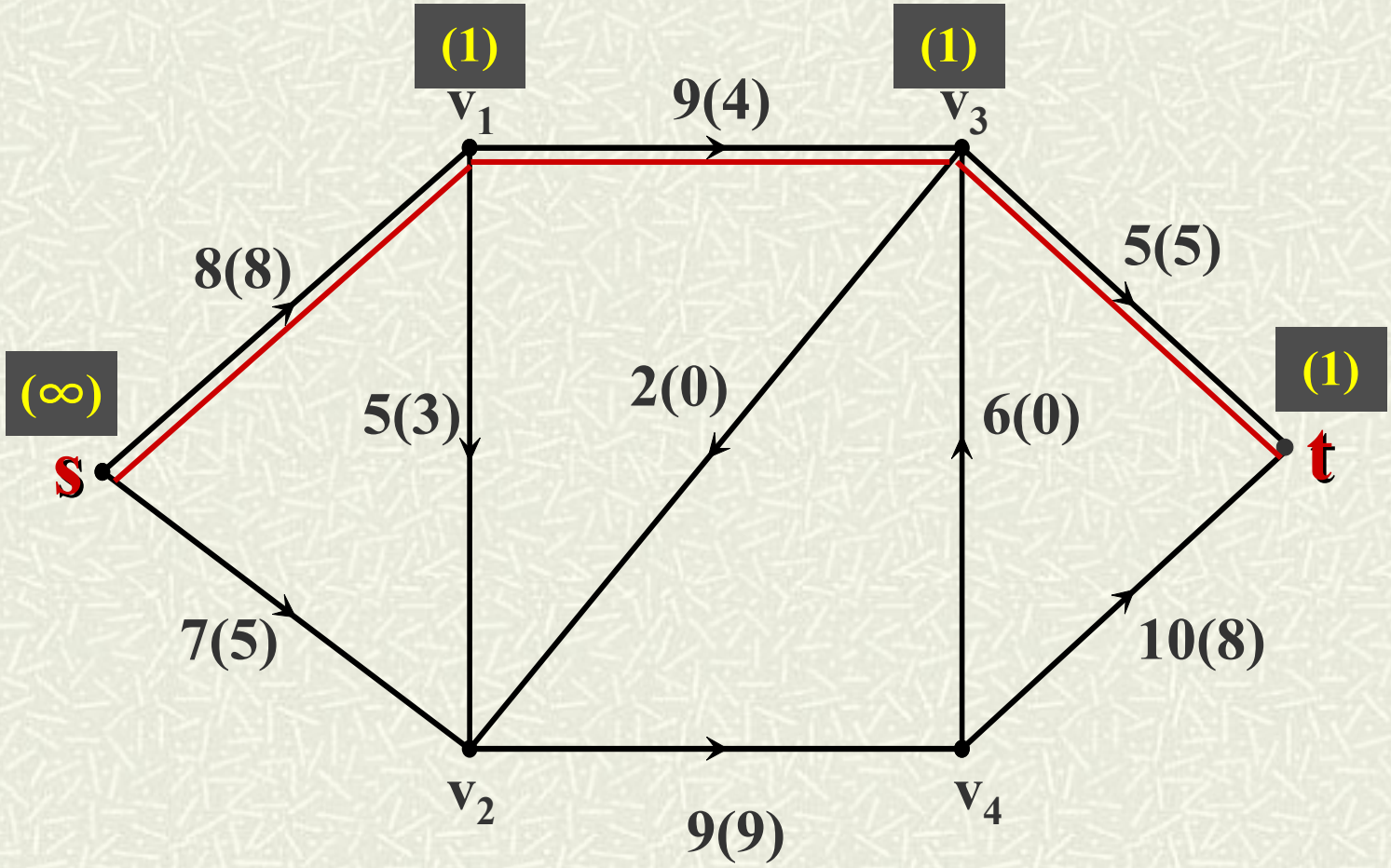
网络的最大流

(4) 修改增广链上的流量，非增广链上的流量不变，得到新的可行流。



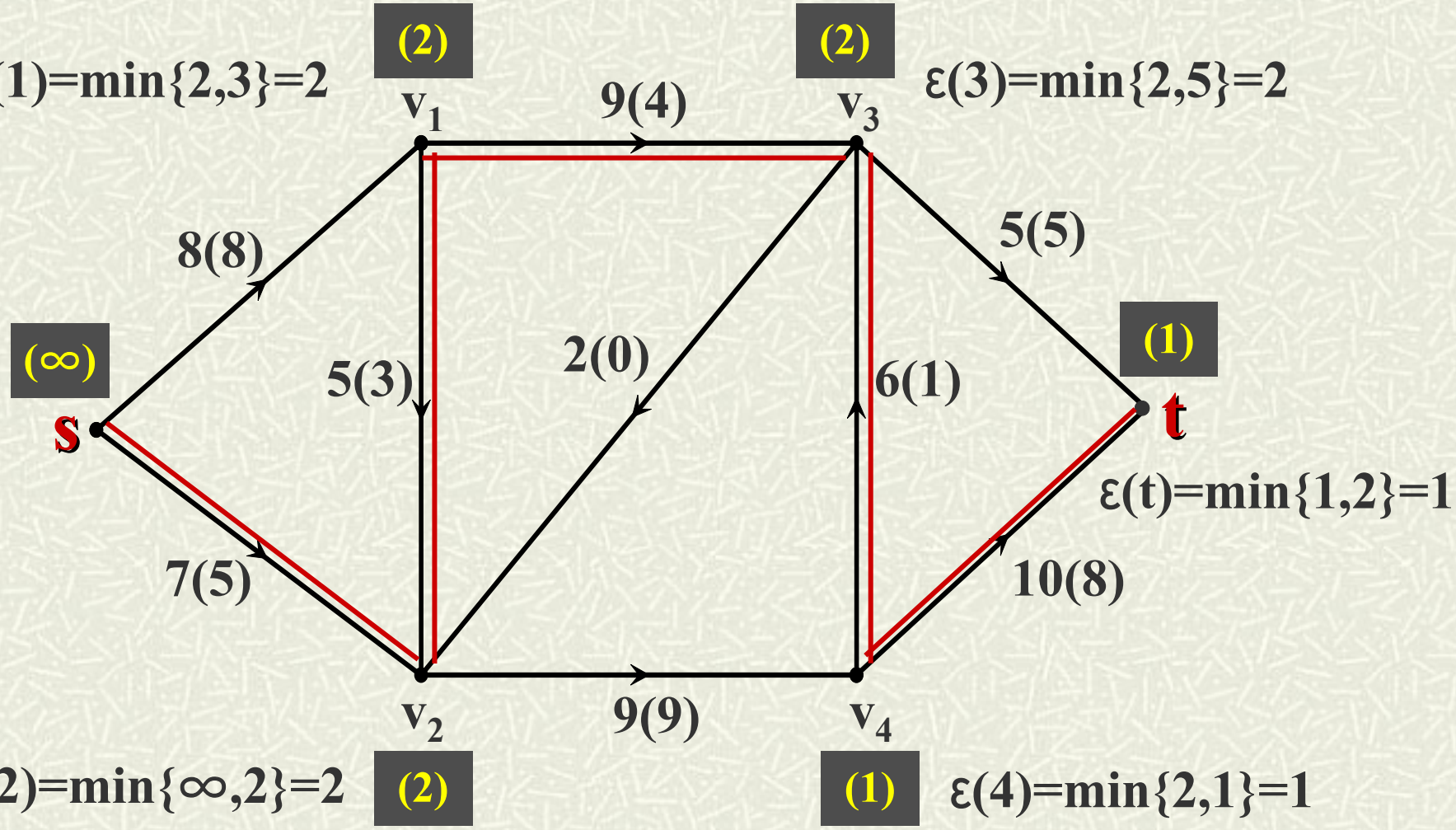
网络的最大流

(5) 擦除所有标号，重复上述标号过程，寻找另外的增广链。



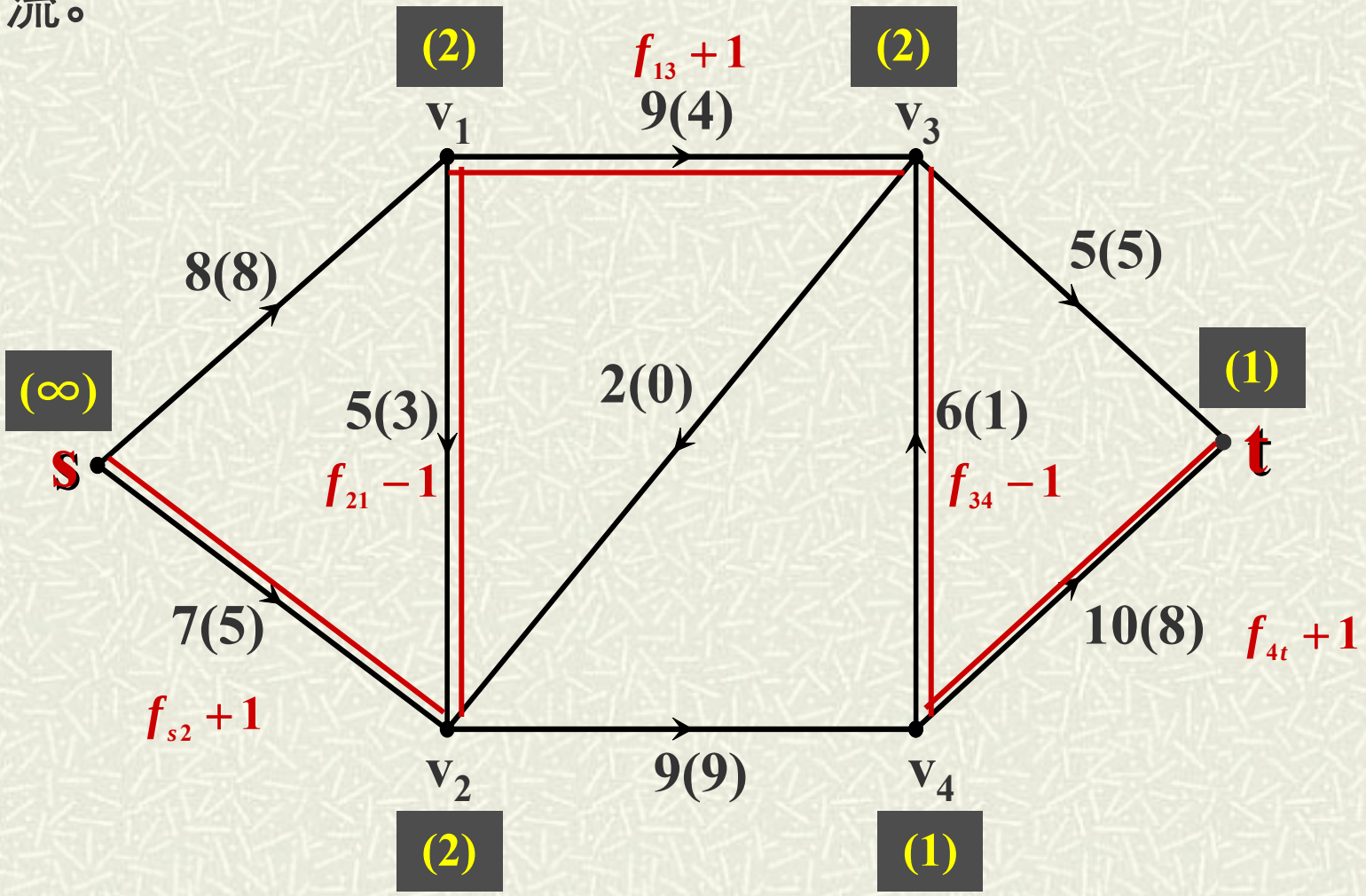
网络的最大流

(5) 擦除所有标号，重复上述标号过程，寻找另外的增广链。



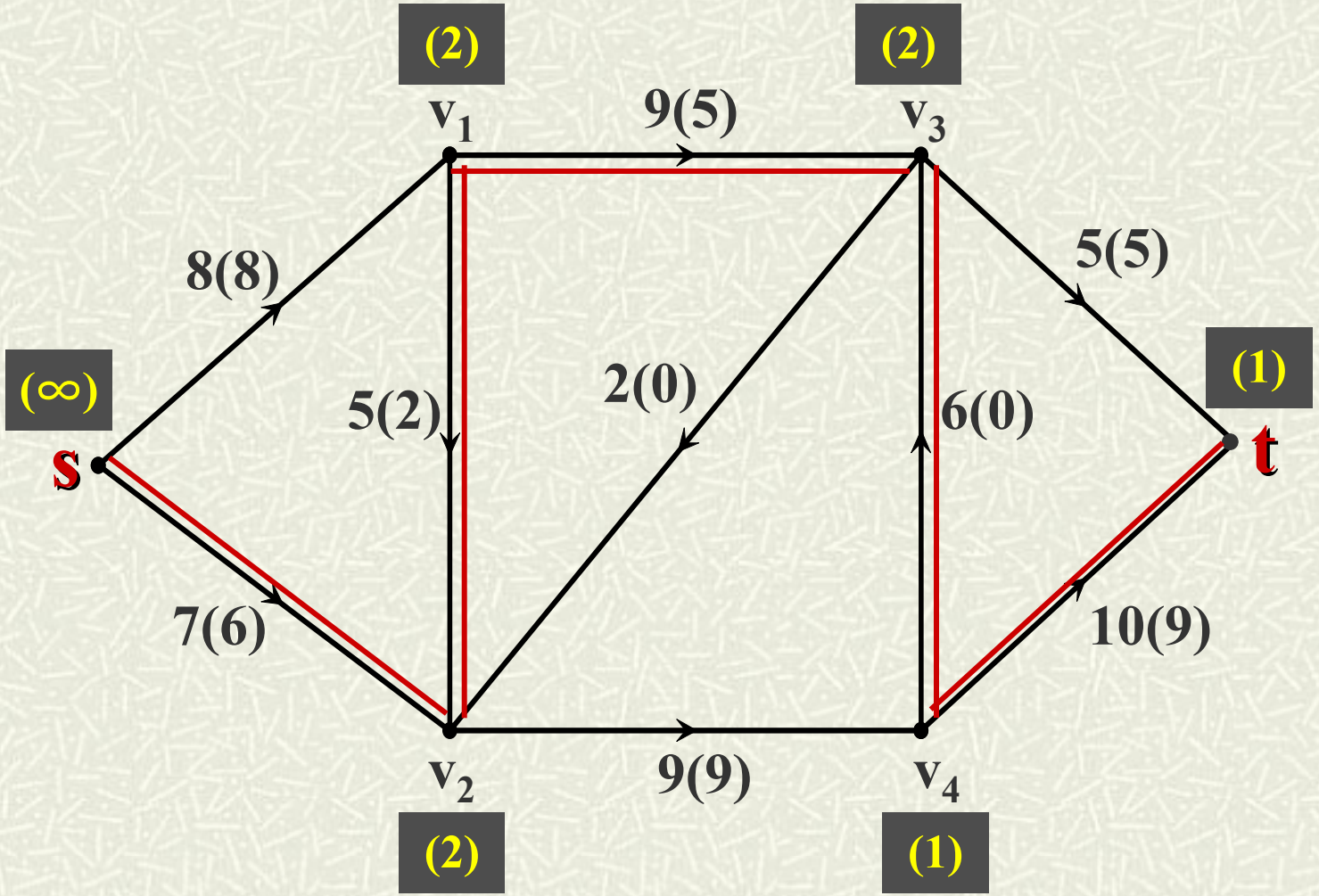
网络的最大流

(6) 修改增广链上的流量，非增广链上的流量不变，得到新的可行流。



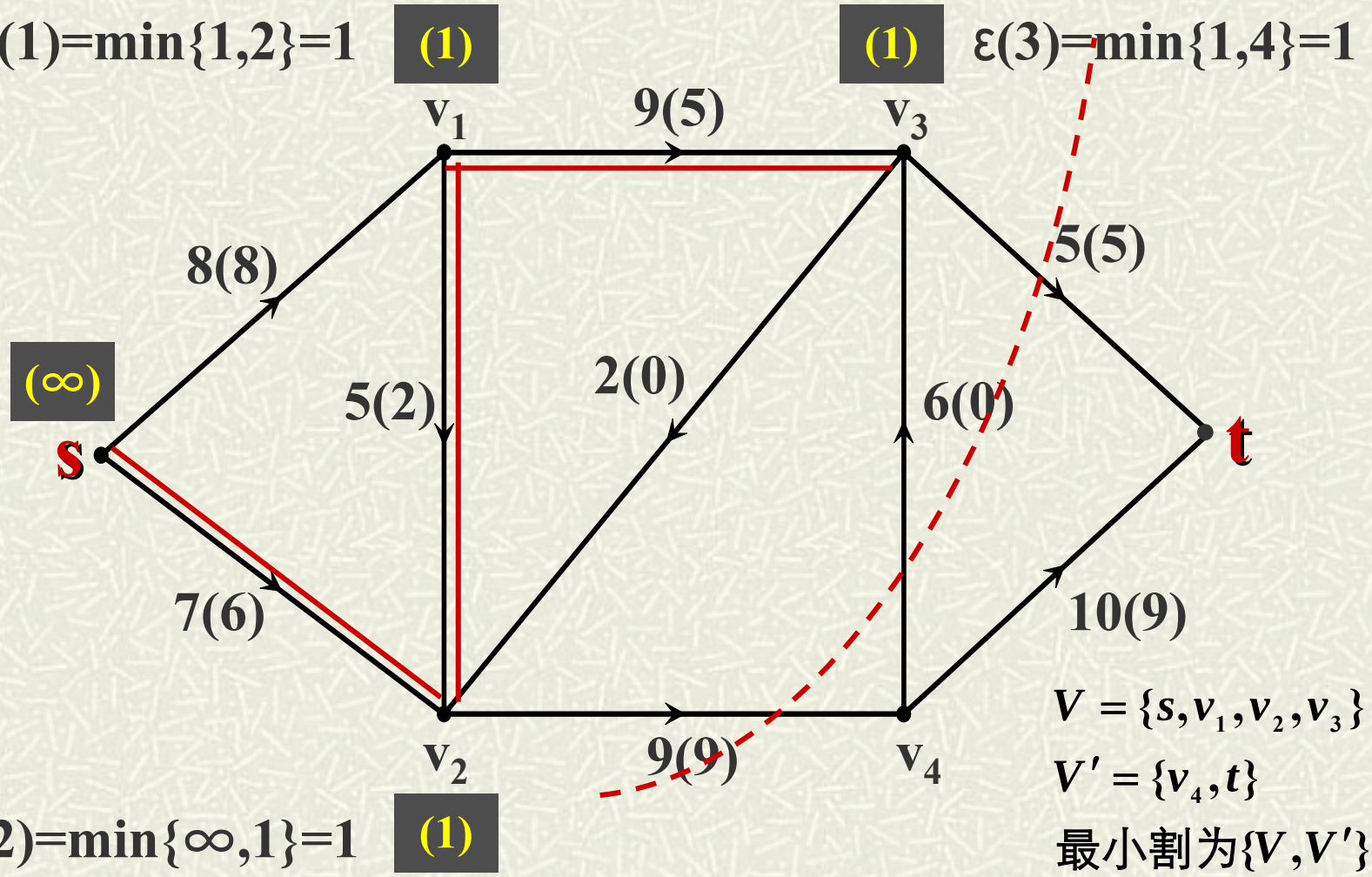
网络的最大流

(7) 擦除所有标号，重复上述标号过程，寻找另外的增广链。



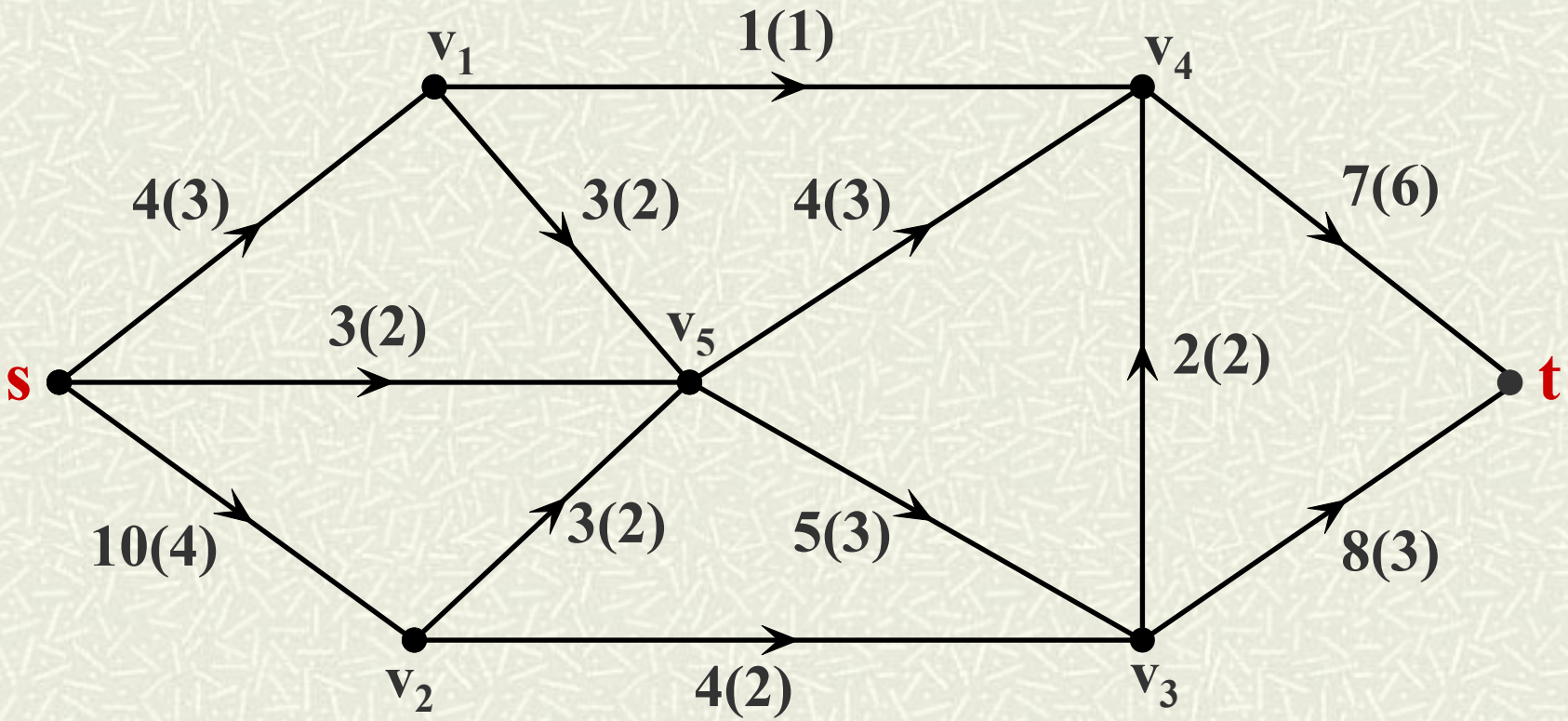
网络的最大流

(7) 擦除所有标号，重复上述标号过程，寻找另外的增广链。



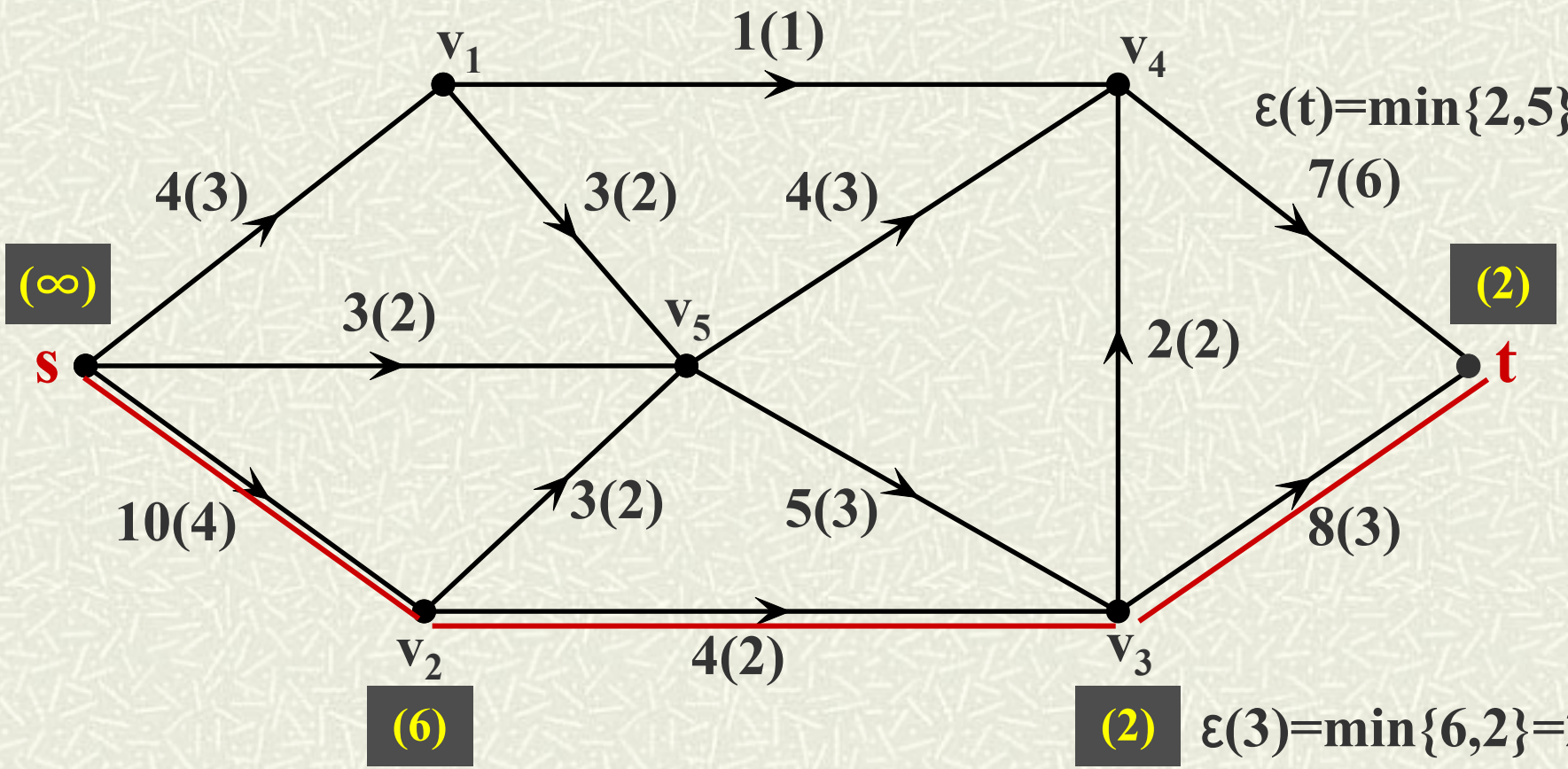
网络的最大流

例6.9 求下图$s \to t$的最大流，并找出最小割



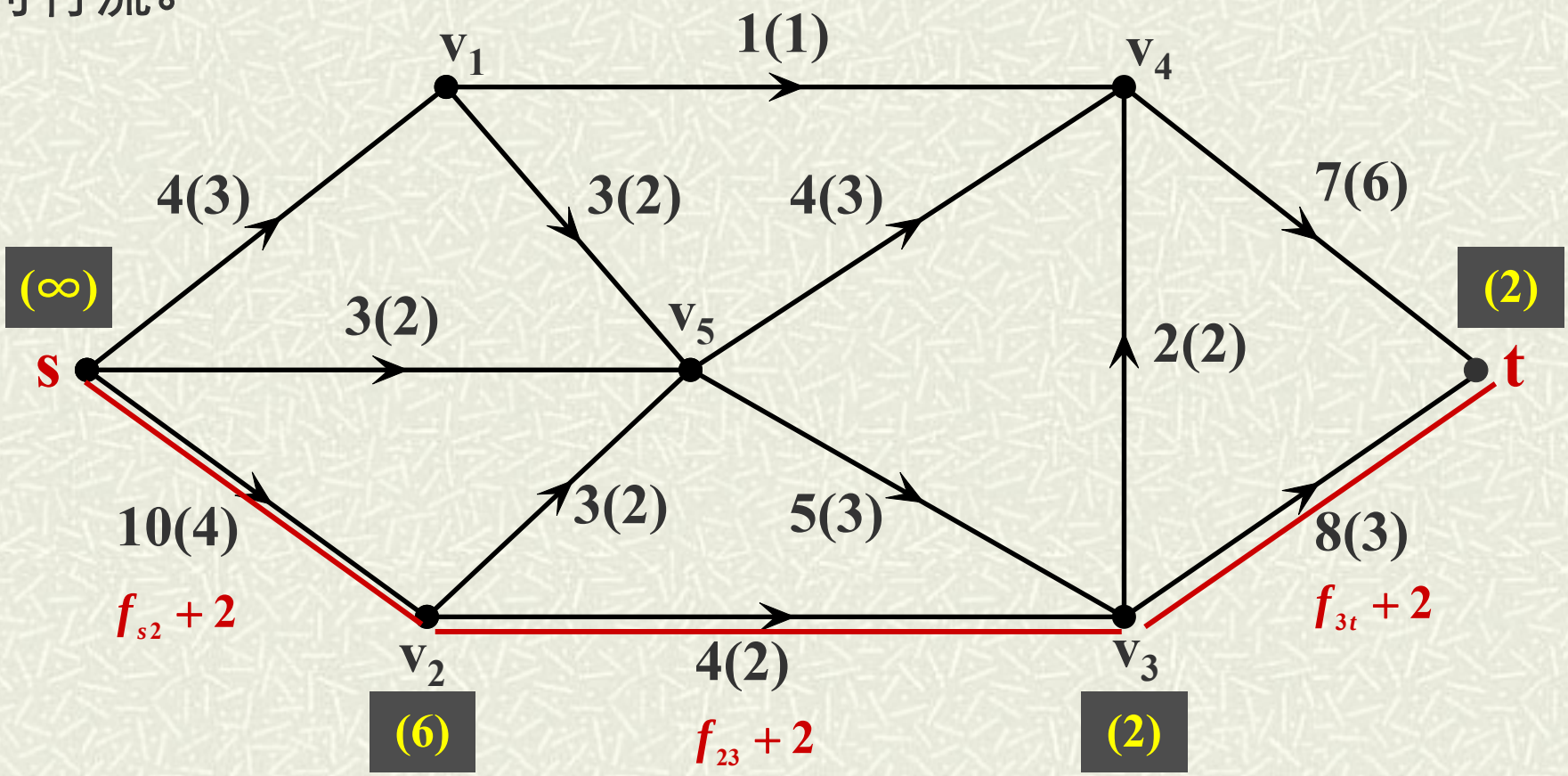
网络的最大流

解: (1) 在已知可行流的基础上，通过标号寻找增广链。



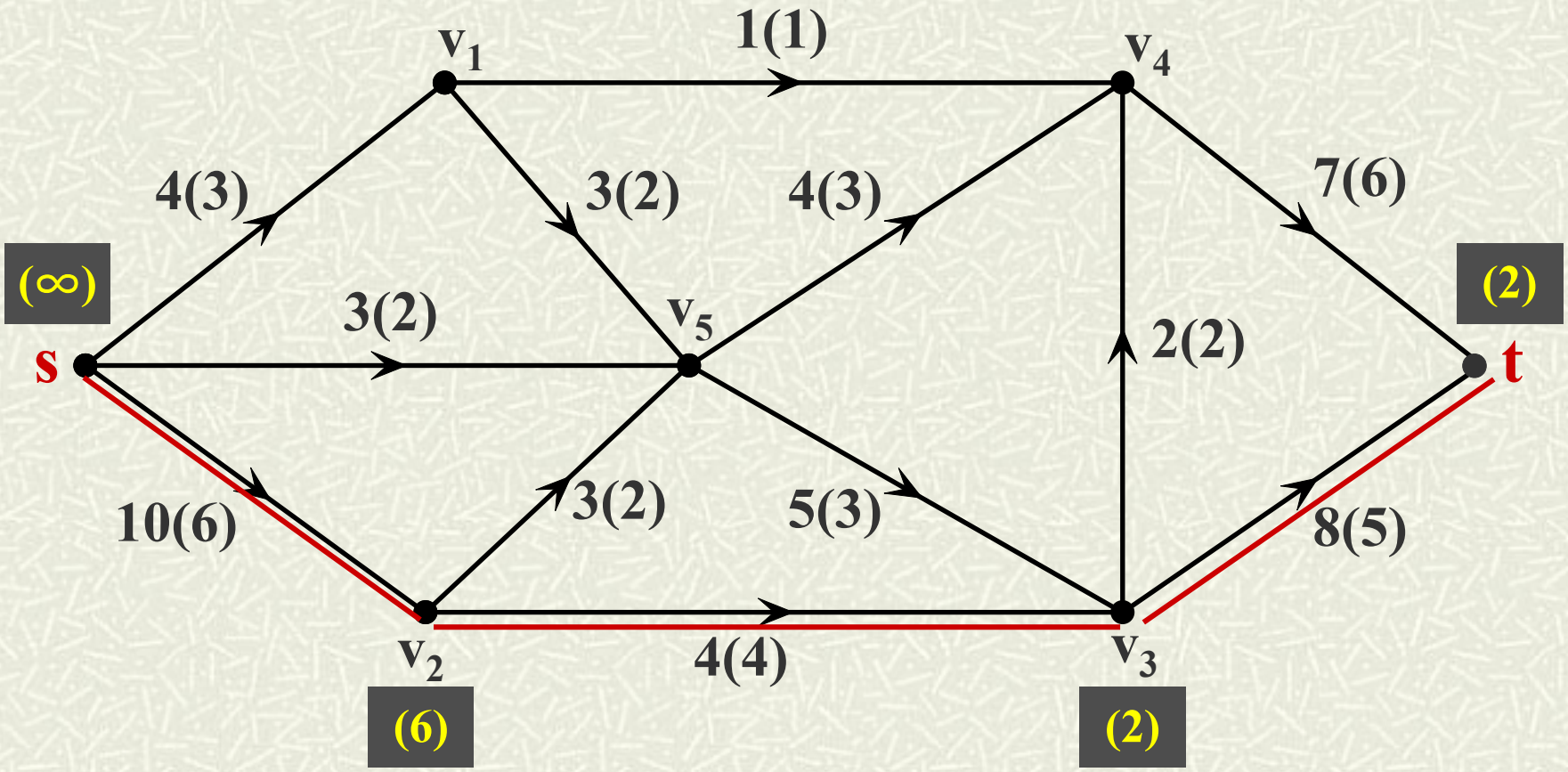
网络的最大流

(2) 修改增广链上的流量，非增广链上的流量不变，得到新的可行流。



网络的最大流

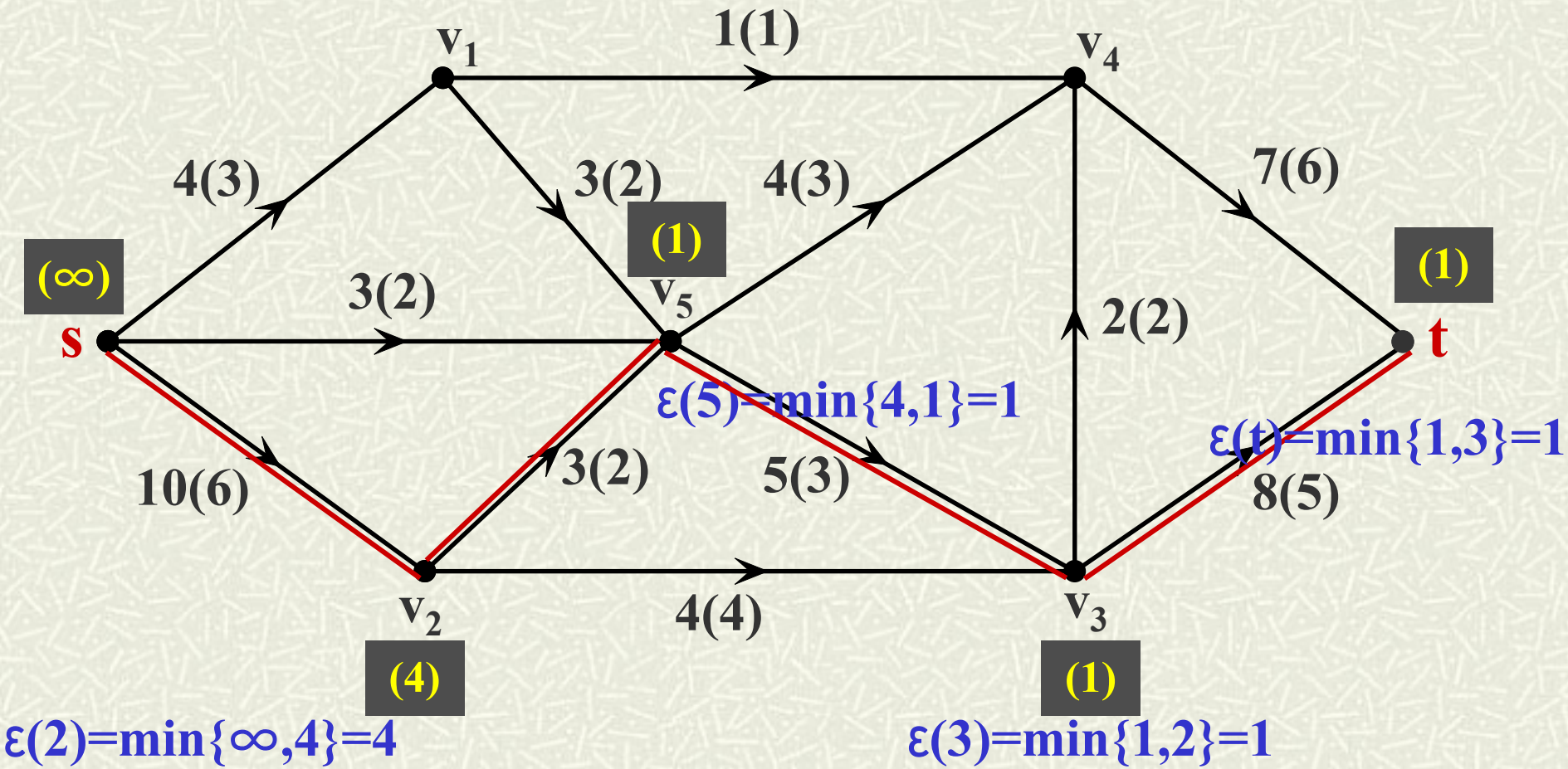
(3) 擦除原标号，重新搜寻增广链。



网络的最大流

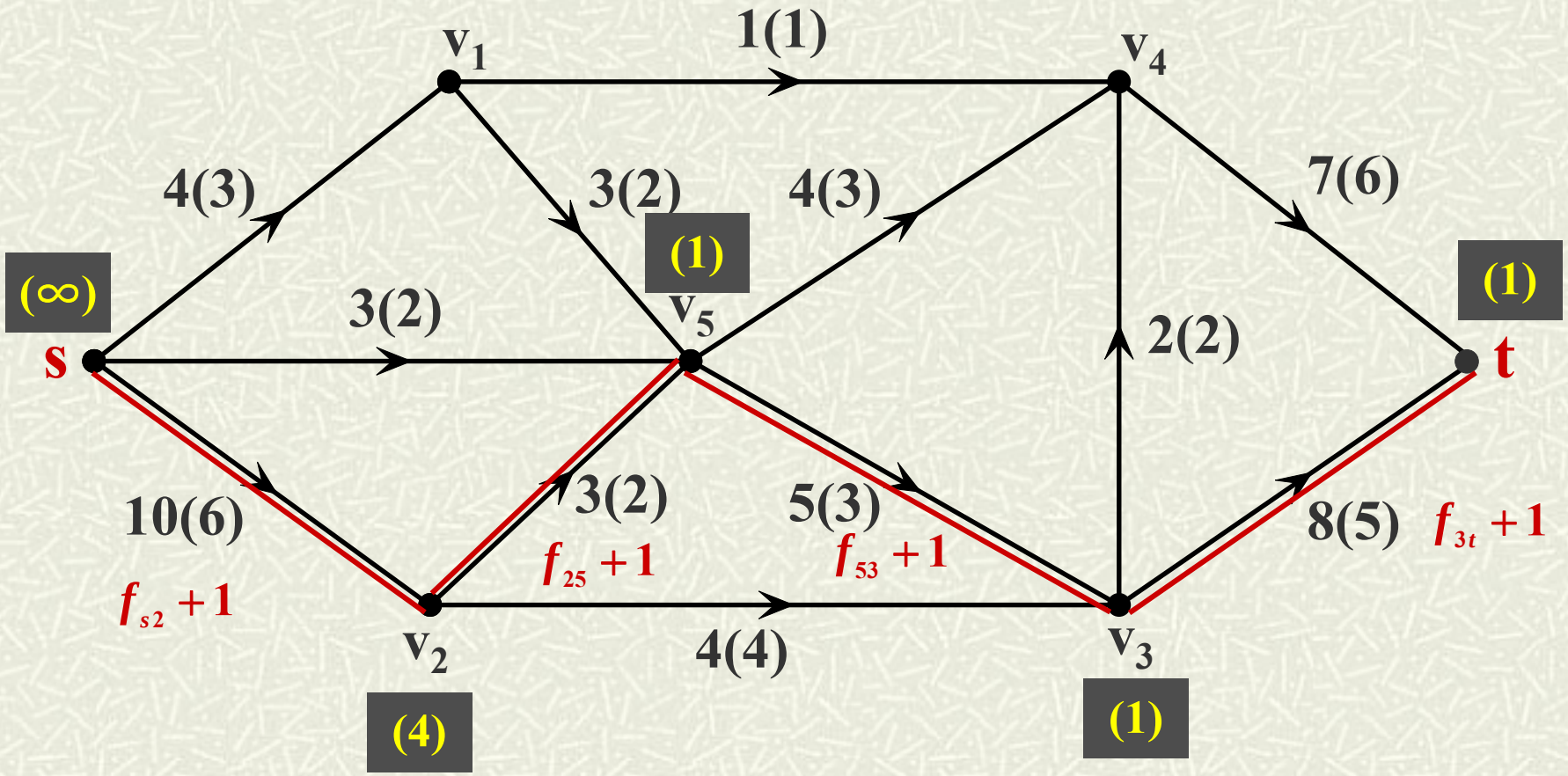
(4) 重新搜寻增广链。

存在增广链：$s \to v_2 \to v_5 \to v_3 \to t$



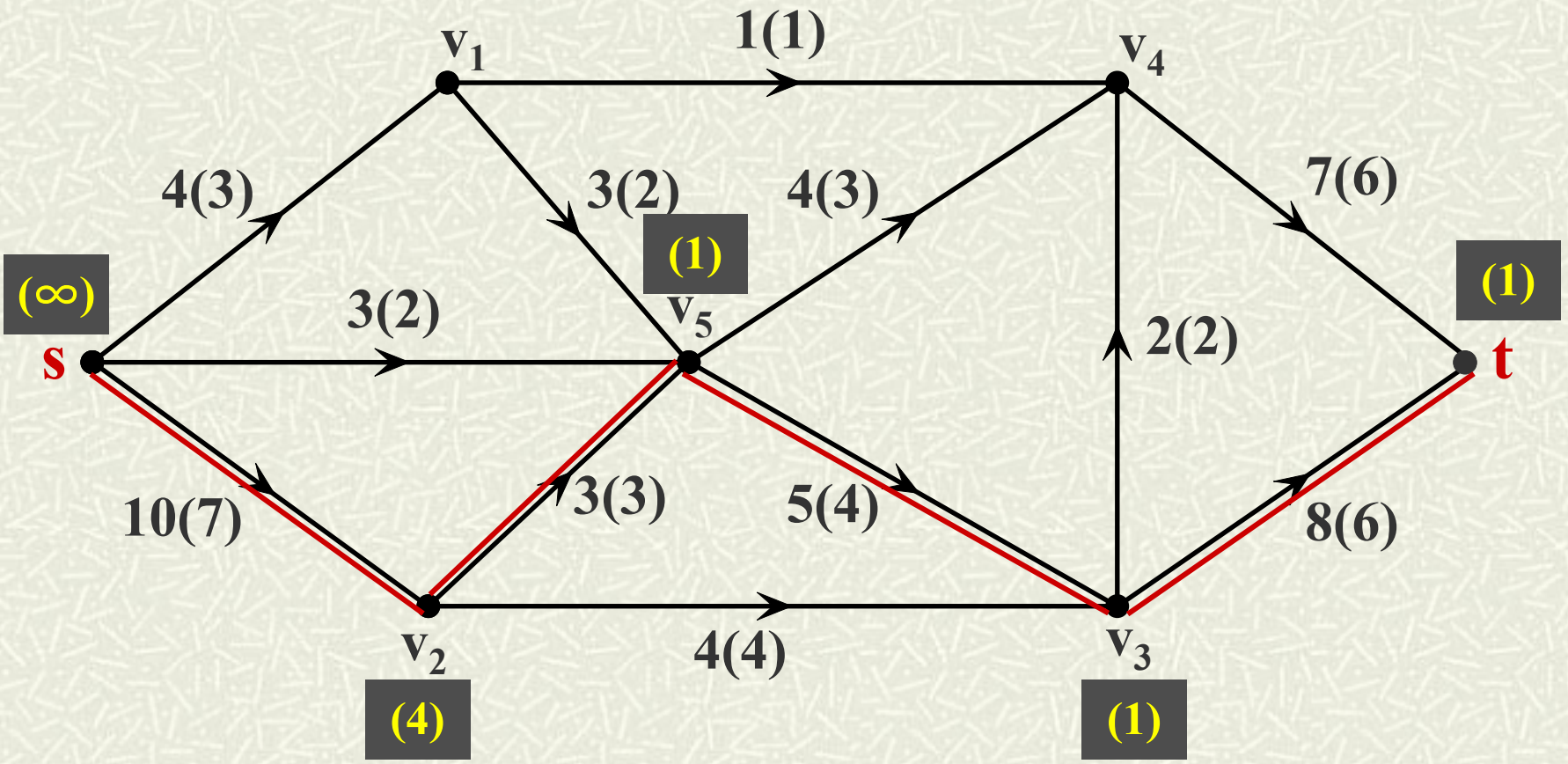
网络的最大流

(5) 修改增广链上的流量，非增广链上的流量不变，得到新的可行流。



网络的最大流

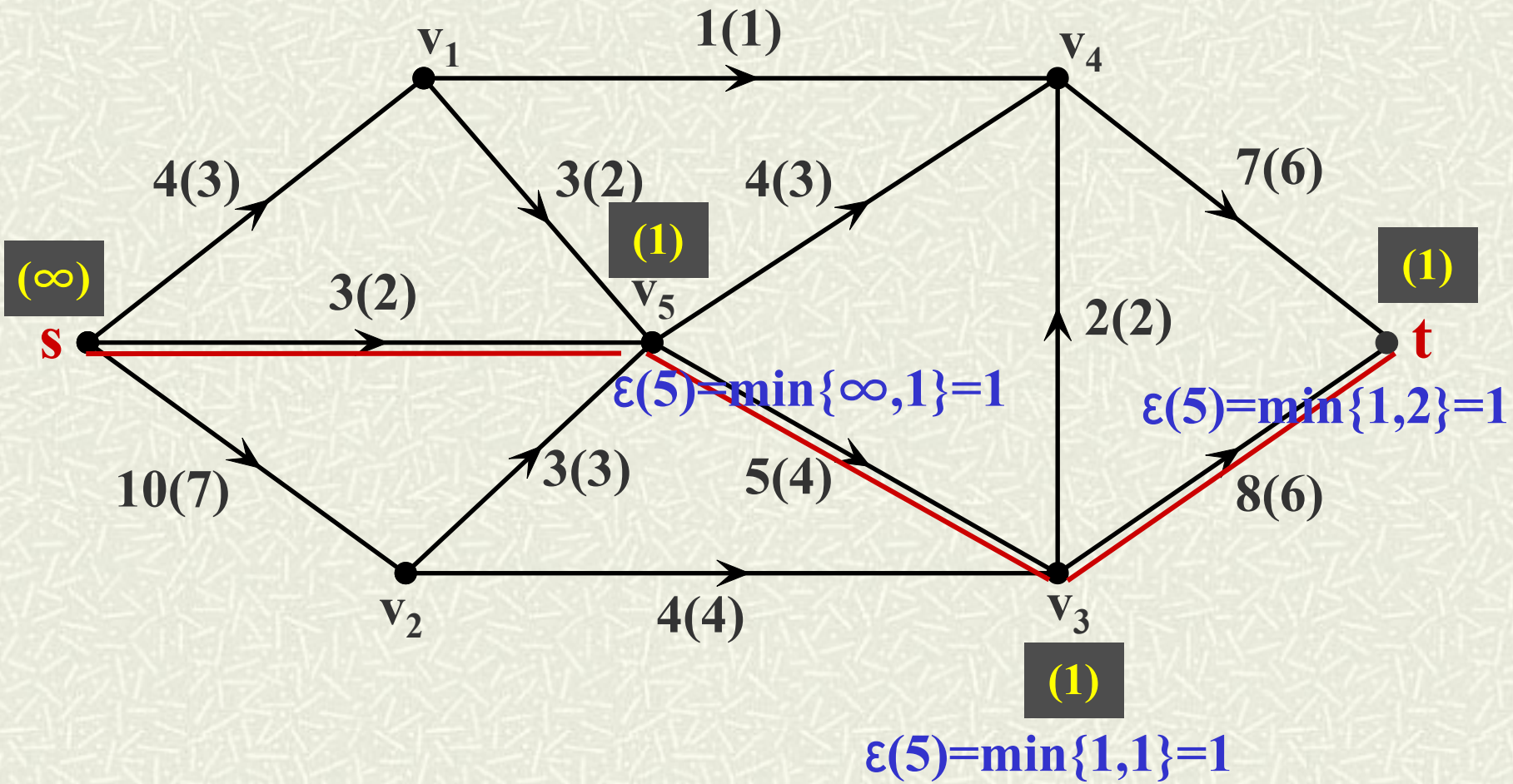
(6) 擦除原标号



网络的最大流

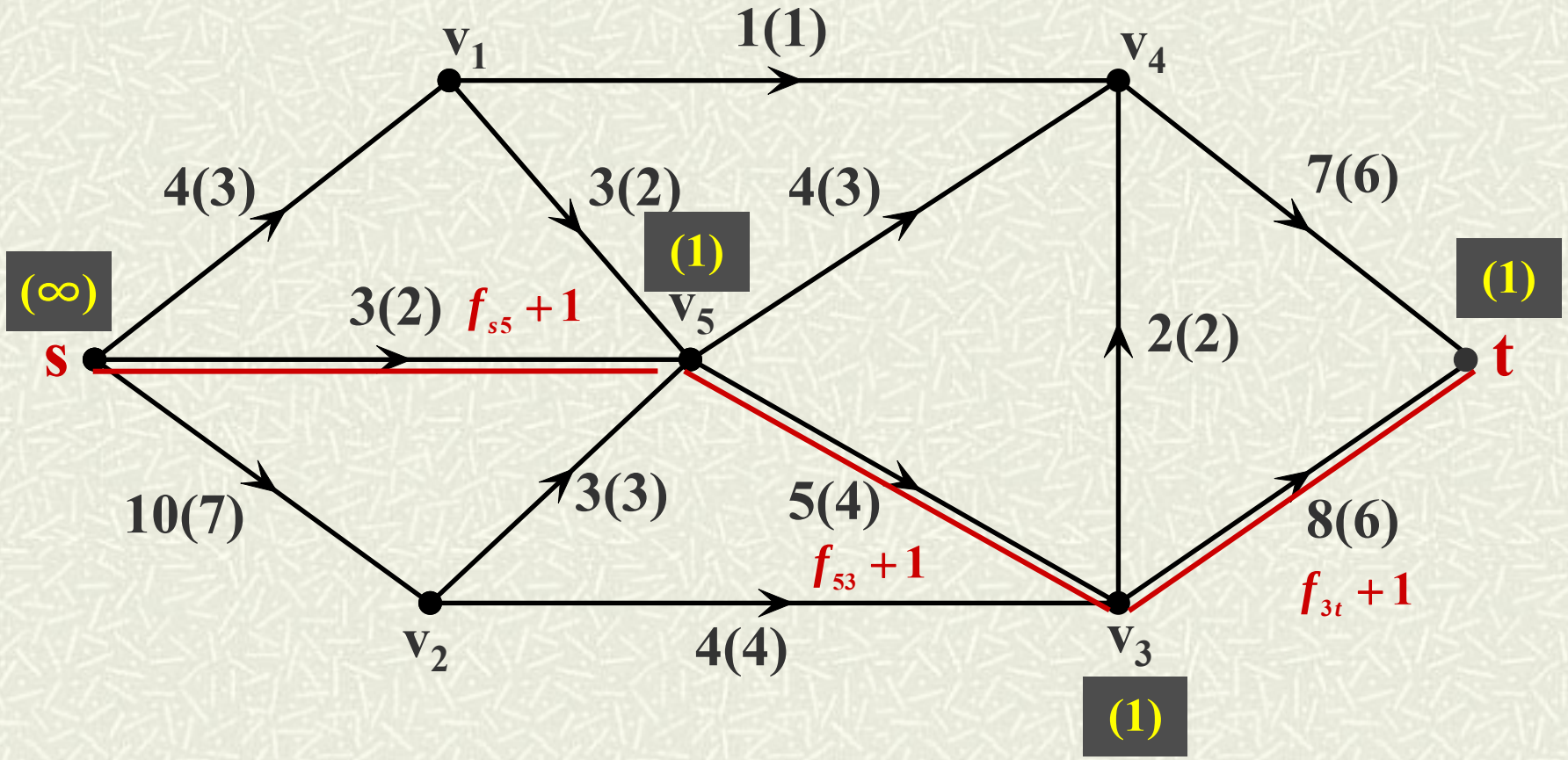
(7) 重新搜寻增广链。

存在增广链：$s \to v_5 \to v_3 \to t$



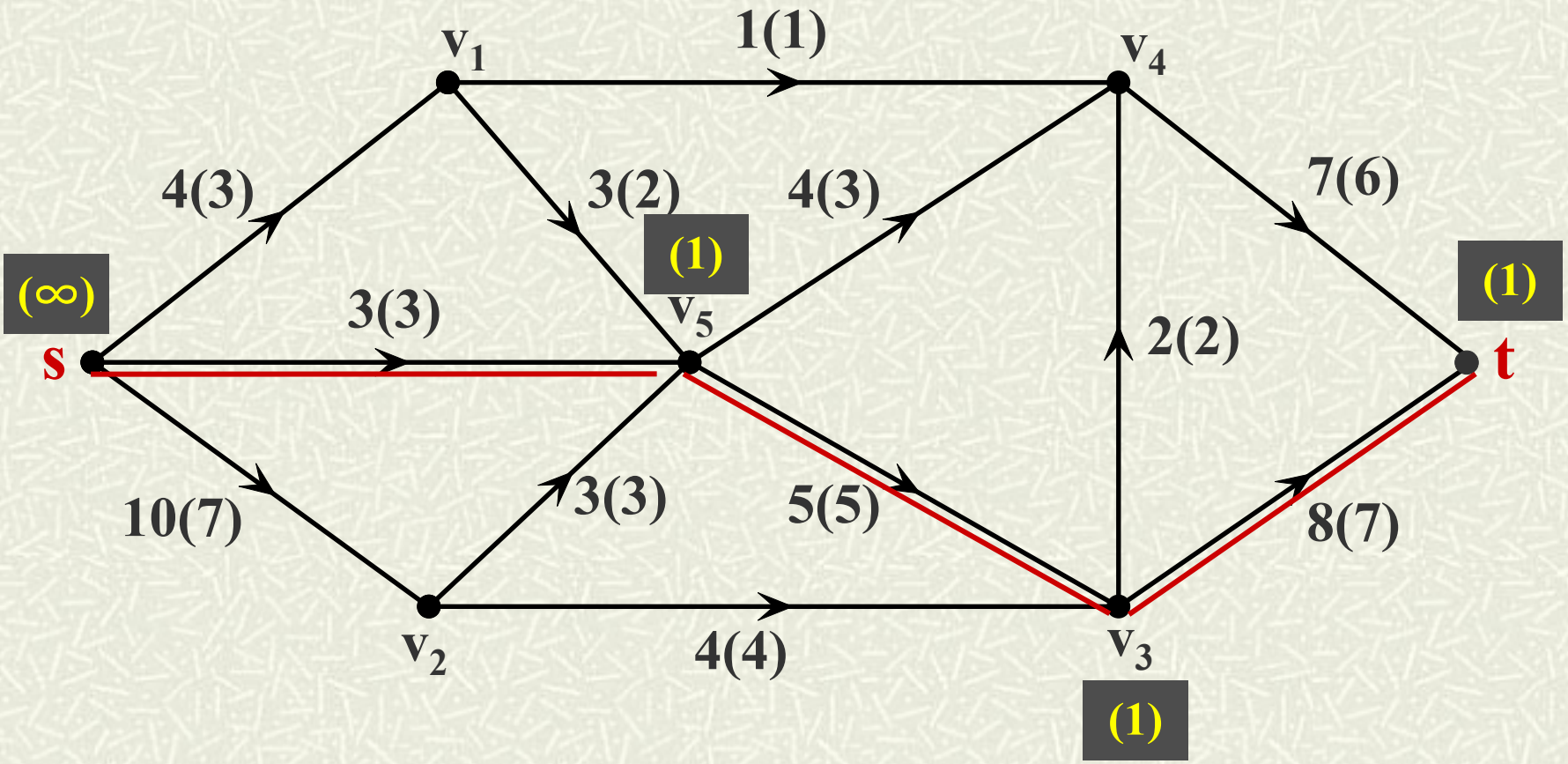
网络的最大流

(8) 调整增广链上的流量，非增广链流量不变，得到新的可行流



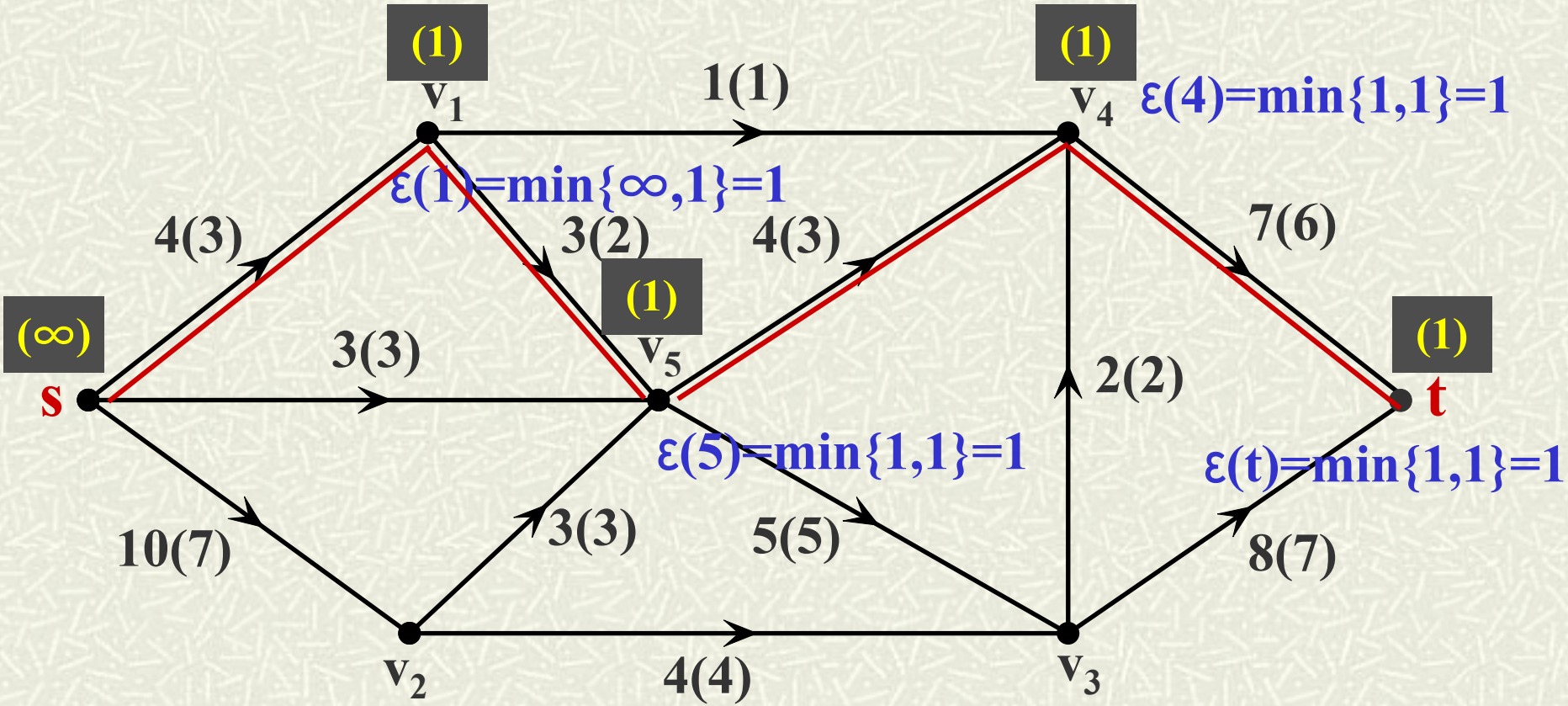
网络的最大流

(9) 擦除原标号



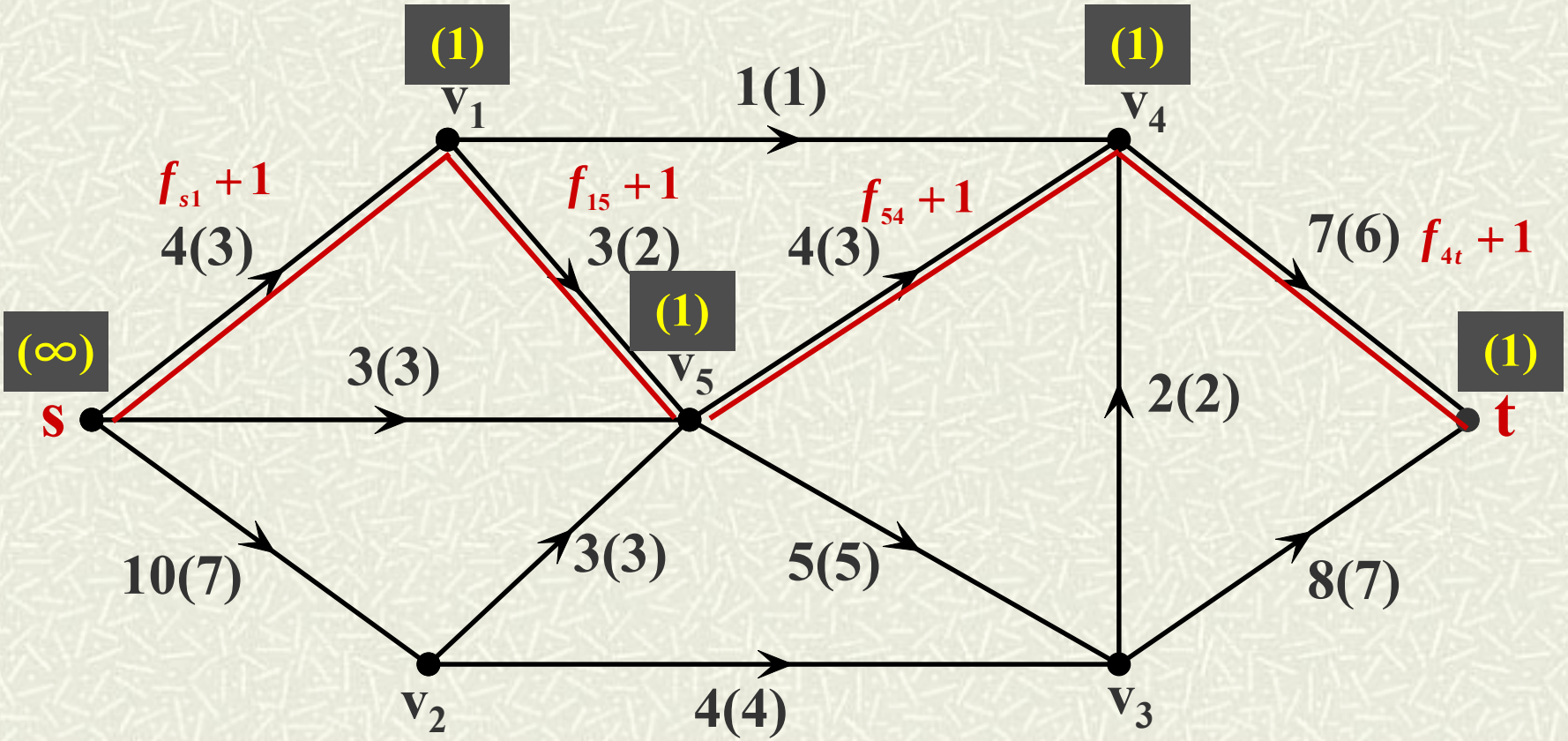
网络的最大流

(10) 重新标号，搜索增广链　　存在增广链：$s \to v_1 \to v_5 \to v_4 \to t$



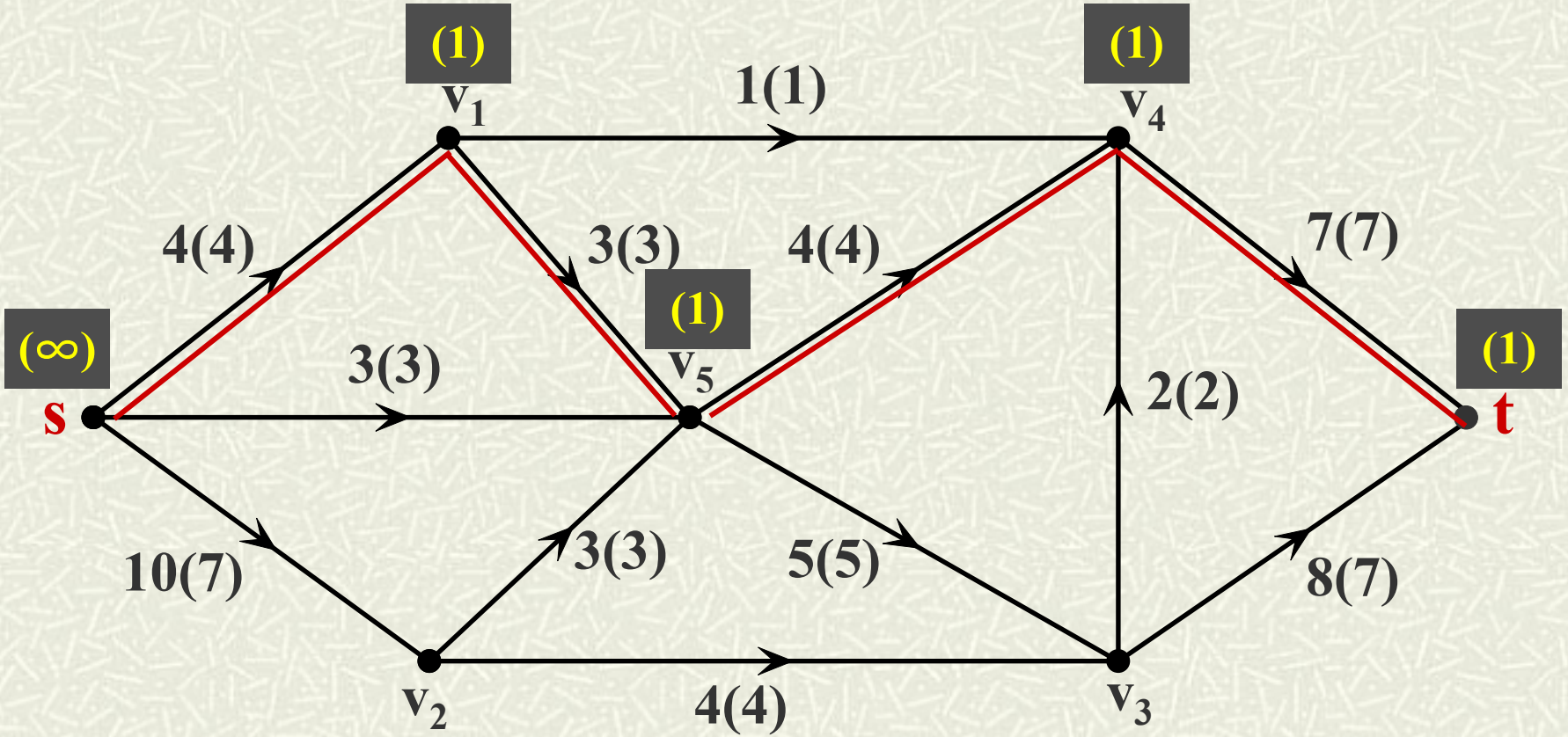
网络的最大流

(11) 调整增广链上的流量，非增广链流量不变，得到新的可行流



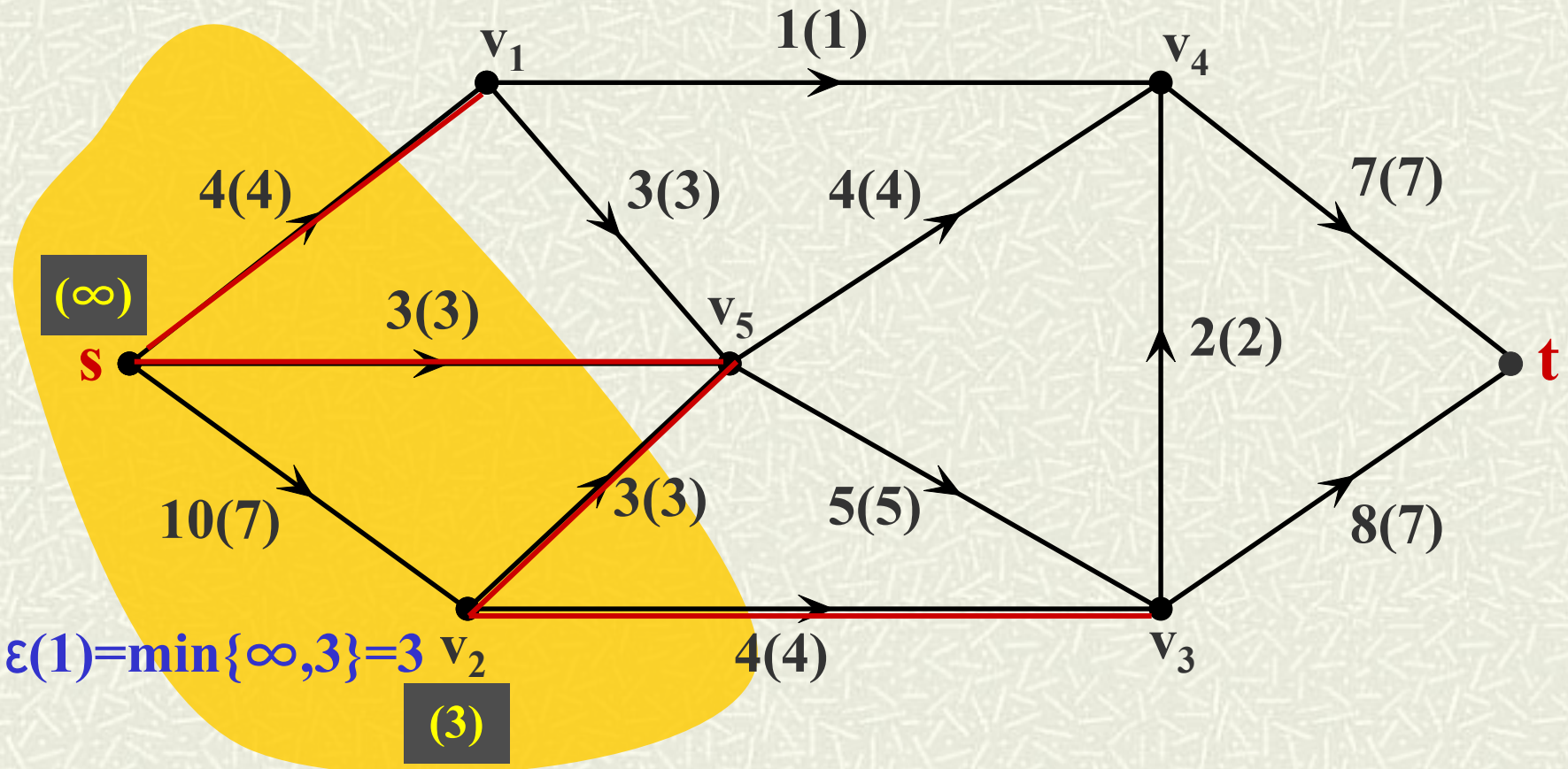
网络的最大流

(11) 擦除标号，在新的可行流上重新标号。



网络的最大流

(11) 擦除标号，在新的可行流上重新标号。



无法标号，不存在增广链，此可行流已为最大流。最大流量为14。

$V = \{s, v_2\}, V' = \{v_1, v_3, v_4, v_5, t\}$, 最小割为 $\{V, V'\}$

最小费用最大流

网络 D = (V, A, C) 中除了容量限制外，每条弧上的单位流均产生一个费用 b(ij) >= 0. 最小费用最大流问题是指：求一个最大流f, 使得其费用最小，即

Min b(f) = sum_{(vi,vj) \in A} {bij * fij}

s.t. f is max flow.

其中b(f)为f的费用。

增广链的费用

设当前的可行流为f, 已找到f的一条增广链mu。 沿着增广链对f进行调整得到可行流 f’. 假设调整量为1. b(f’) 与 b(f) 相比增加多少?

b(f’) − b(f) = sum_{顺向弧} {bij} - sum_{逆向弧} {bij}.

将sum_{顺向弧} {bij} - sum_{逆向弧} {bij} 称为该增广链的费用。

最小费用最大流

一个基本事实:

如果可行流f是所有流量为 v(f)的可行流中费用最小者，并且mu是关于f的所有增广链中费用最小的，那么沿着该增广链对f进行调整得到的可行流 f’也是所有流量为 v(f’)的可行流中费用最小者.

求最小费用增广链的方法

基于当前可行流为f 构造赋权有向图 W(f):

W(f) 的顶点包含原网络D中的所有顶点;
D中的每一条弧（vi,vj) 都改为两条弧： (vi,vj), (vj,vi);
赋权（可能带负权）:

w_ij = bij (如果 fij < cij)
w_ij = +infinity (如果 fij = cij); (增加单位流量产生的费用）
w_ji = -bij (如果 fij > 0)
w_ji = +infinity (如果 fij = 0); (减少单位流量减少的费用）

求最小费用增广链即为求 W(f)中 vs 到 vt 的最短路问题。

最小费用最大流算法

1. 以零流开始， f=0;
2. 构造 W(f), 求 vs到vt的最短路，得到vs到vt的最小费用增广链，则转3； 若在W(f)中找不到vs到vt的最短路，则当前可行流即为最小费用最大流。
3. 沿着增广链调整f得到新的可行流;
4. 重复 2， 3， 。。。;