第12章　图（一）

计算机学院

主要内容

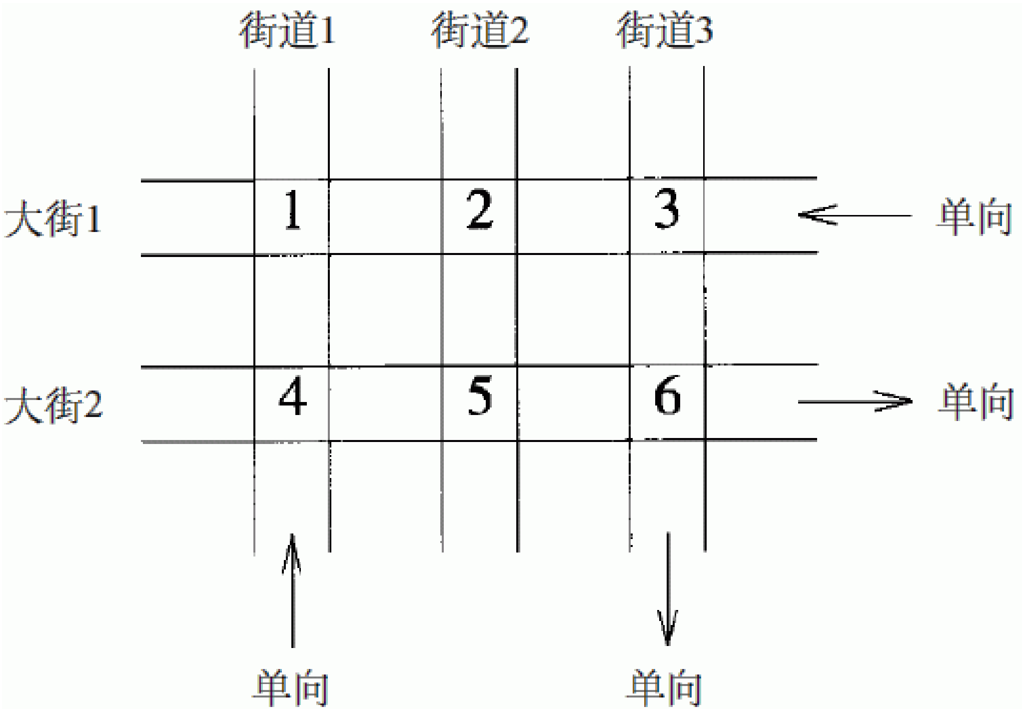
- 图的基本概念
- 图的存储
- 图的遍历
- 最小生成树

图的定义

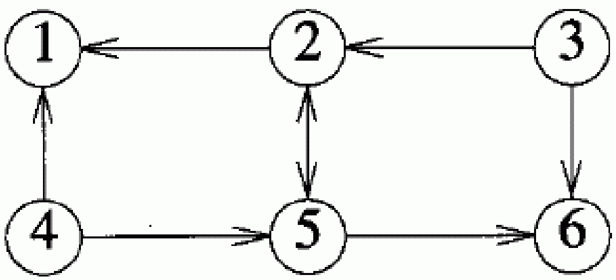
- **图**（graph）：用线（**边**）连接起来的顶点（**节点**）的集合
- 图G=(V, E)
  - V：顶点有限集合
  - E：边有限集合，（i，j），i、j∈V

南开大学校园平安地图


N
西门
高峰期车辆
拥堵，过马
路注意
红灯
西门主
要公交车
638，640
687，874
849
（门禁）
（向西单向行驶）
体育场
宿舍区
三食
校医院
急诊电话：23502400
老图书馆
办公楼
北村
东门主
要公交车
643，8，
832，859
952
数学院
七教
二食
一食
伯苓楼
图书馆
化学楼
小引河：小心溺水
七宿
二主楼
新开湖
小心溺水
马蹄湖
大中路:汽车禁行，路口多，自行车多，转弯注意
（门禁）
东门
高峰期车辆
拥堵，自行
车注意减速
慢行
西南村
此处可
步行通过
保卫处
电话:23508538
门禁
生科楼
主楼
东艺楼
幼儿园
综合实验楼
东村
派出所
（高峰拥堵）
（门禁）
西南门：车辆较多，
注意左转交通灯
西南门：47
698，831，48
57，95
南门：95
831，902，831
175，845
南门：禁止翻越
栏杆，车速较快，过
马路需绕行
图例
高峰时段保安巡逻处
交通拥堵区
非机动车行驶区
公交车提醒
水域



街道1
街道2
街道3
大街1
1
2
3
单向
大街2
4
5
6
单向
单向
单向

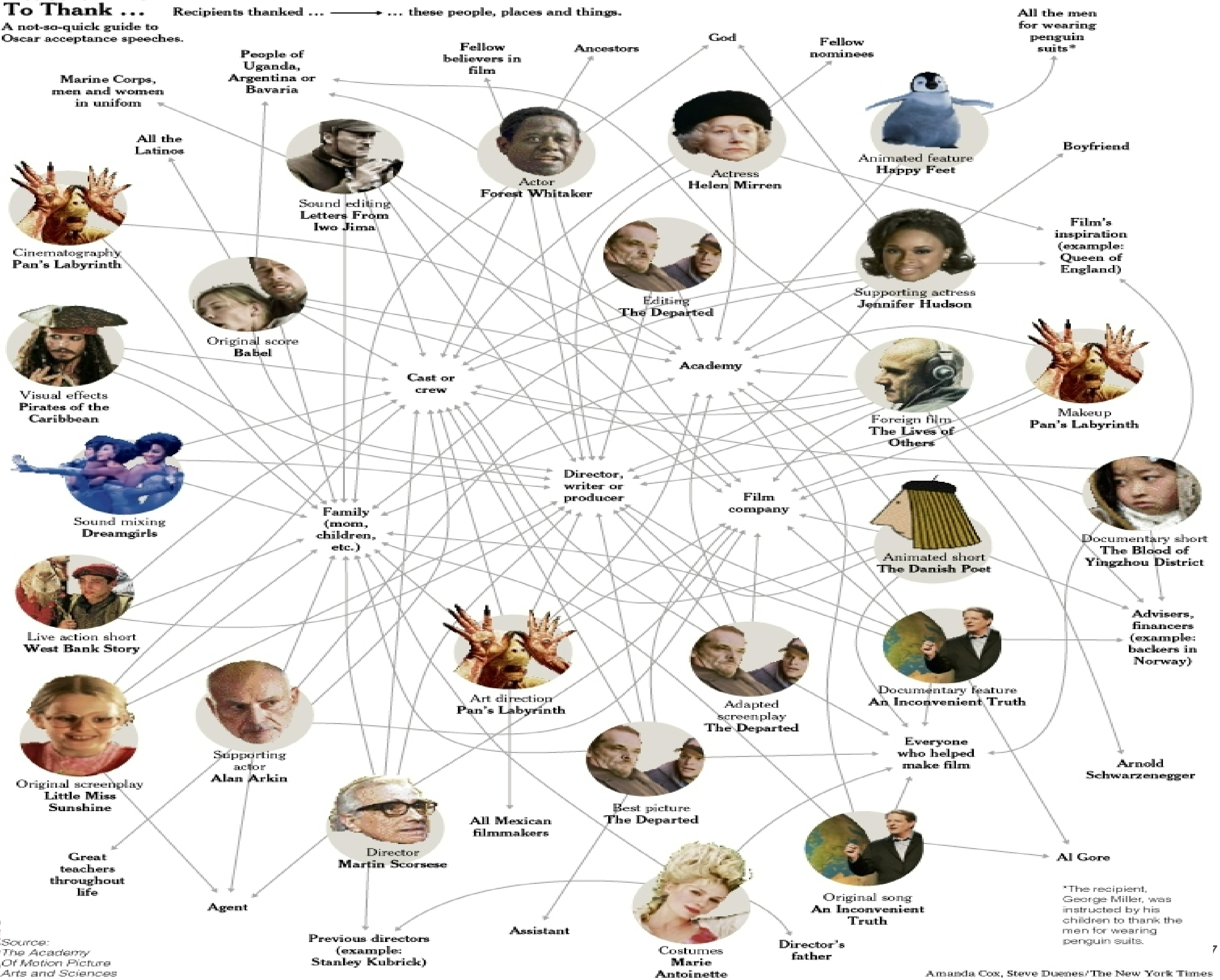


1
2
3
4
5
6


贾樟柯
分享到微博
张一一
查看详细
范晓萱
陈丹青
韩雪
张悦然
金莎
郭敬明
王朔
周云蓬
韩寒
李戡
金丽华
罗永浩
路金波
方舟子
韩仁均
马日拉
赵丽华
帮助

I Would Like To Thank ...

A not-so-quick guide to Oscar acceptance speeches.



Source: The Academy Of Motion Picture Arts and Sciences

Amanda Cox, Steve Duenes/The New York Times

基本概念

(1)顶点　(2)边　(3)无向边　(4)有向边

(5)关联于　(6)关联至　(7)邻接于　(8)邻接至

(9)无向图　(10)有向图　(11)完全图　(12)稀疏图　(13)稠密图

(14)带权图　(15)子图

(16)顶点的度　(17)入度　(18)出度

(19)路径　(20)路径长度　(21)简单路径　(22)回路

(23)连通图　(24)连通分量　(25)强连通图　(26)强连通分量

(27)生成树　(28)生成森林

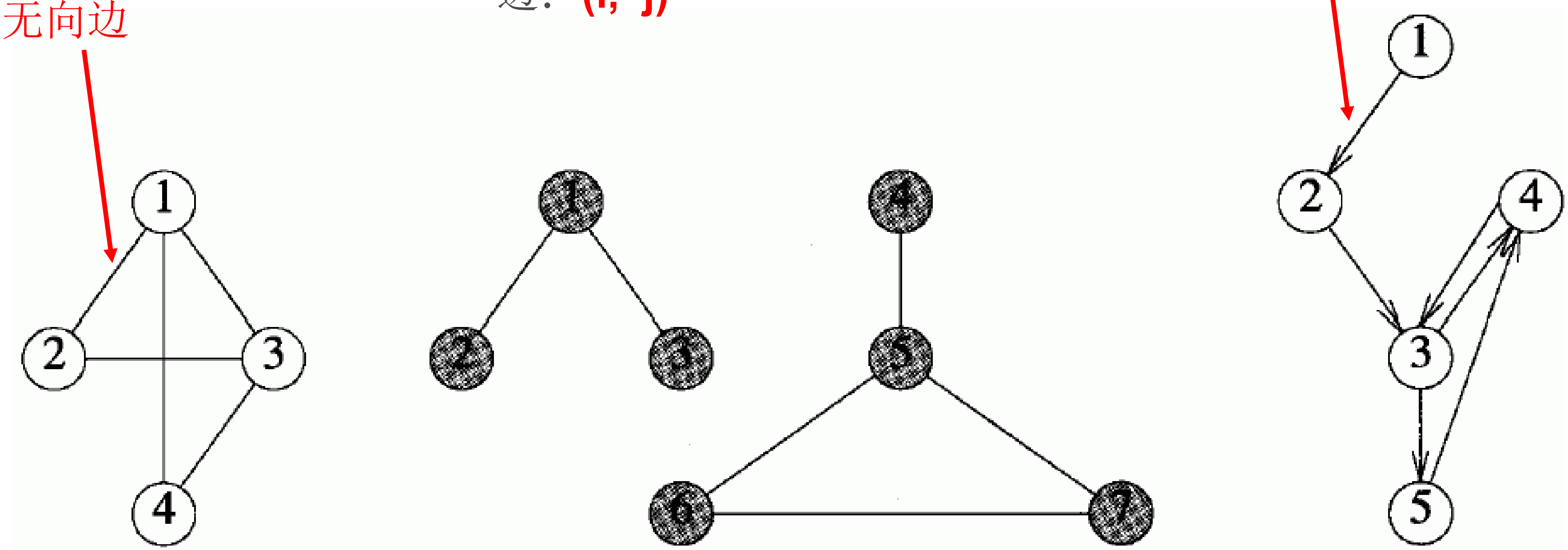
基本概念1-8

(1)顶点　(2)边　(3)无向边　(4)有向边
(5)关联于　(6)关联至　(7)邻接于　(8)邻接至

有向边

边：**(i, j)**

无向边



E={(1,2),(1,3),(1,4),(2,3),(3,4)}　　**E={(1,2),(2,3),(3,4),(4,3),(3,5),(5,4)}**

基本概念9-13

(9)无向图：所有边都是无向边的图

(10)有向图：所有边都是有向边的图

(11)完全图：边数达到最大的图　n(n-1)/2

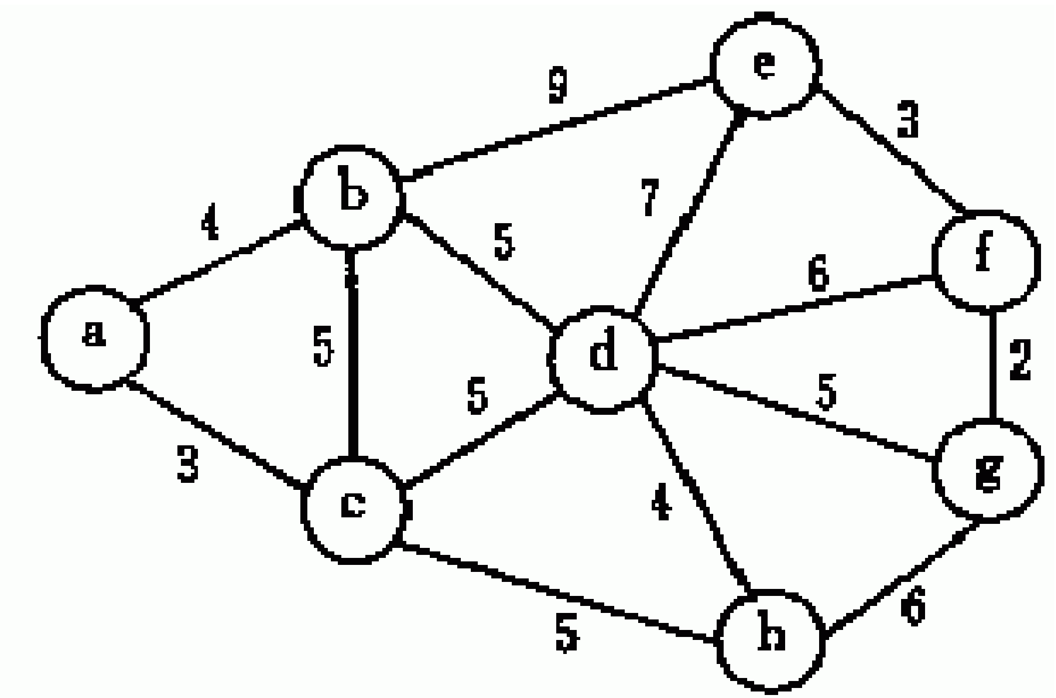
(12)稀疏图：有很少边的图

(13)稠密图：有较多边的图

关于边的基本认识：
1.两点之间至多有一条边
2.不存在自连边

基本概念14

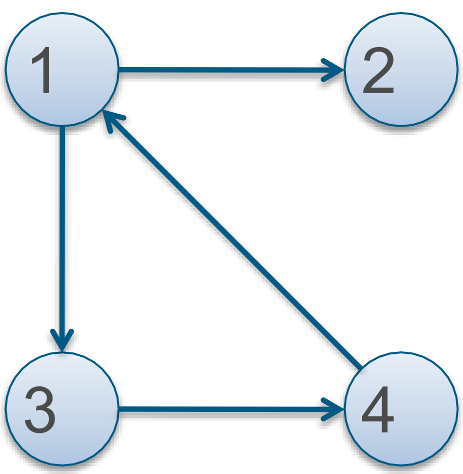
- **加权有向图、加权无向图：**
  每条边赋予一个权重
  ——(14) **网络**（network）

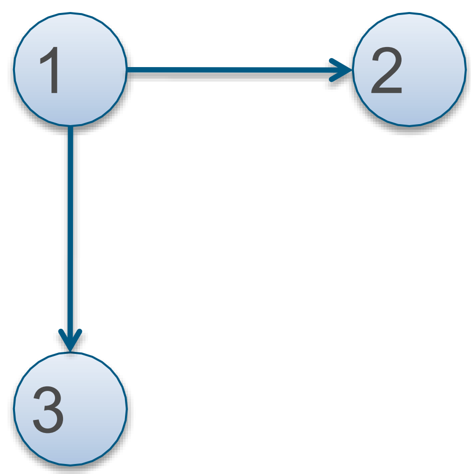


基本概念15

(15)子图：设有两个图G=(V, E)和G'=(V', E'),

- 如果V'是V的子集
- 而且E'是E的子集
- 则称G'是G的子图



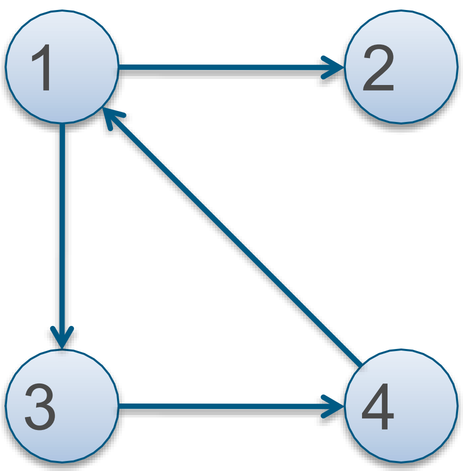


基本概念16-18

(16) 顶点v的**度**：与v关联的边的数目

(17) 有向图中顶点v的**入度**：关联至v的边的数目

(18) 有向图中顶点v的**出度**：关联于v的边的数目



入度：1,1,1,1
出度：2,0,1,1

基本概念19-22

(19) 路径：

当且仅当对于每一个j（$1\leqslant j\leqslant k$），

边（i_j，i_{j+1}）都在E中时，

顶点序列$P=i_1$，i_2，...，i_k是图或有向图$G=(V,E)$中一条从i_1到i_k的路径

(20) 路径长度：路径上所有边的长度之和

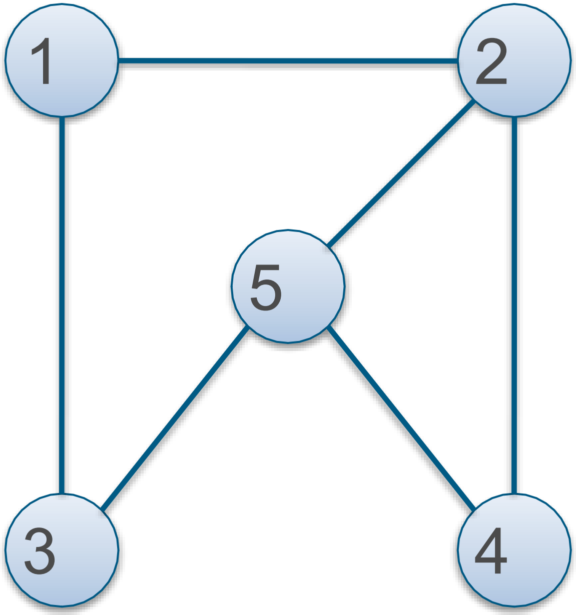
(21) 简单路径：序列中顶点不重复出现的路径

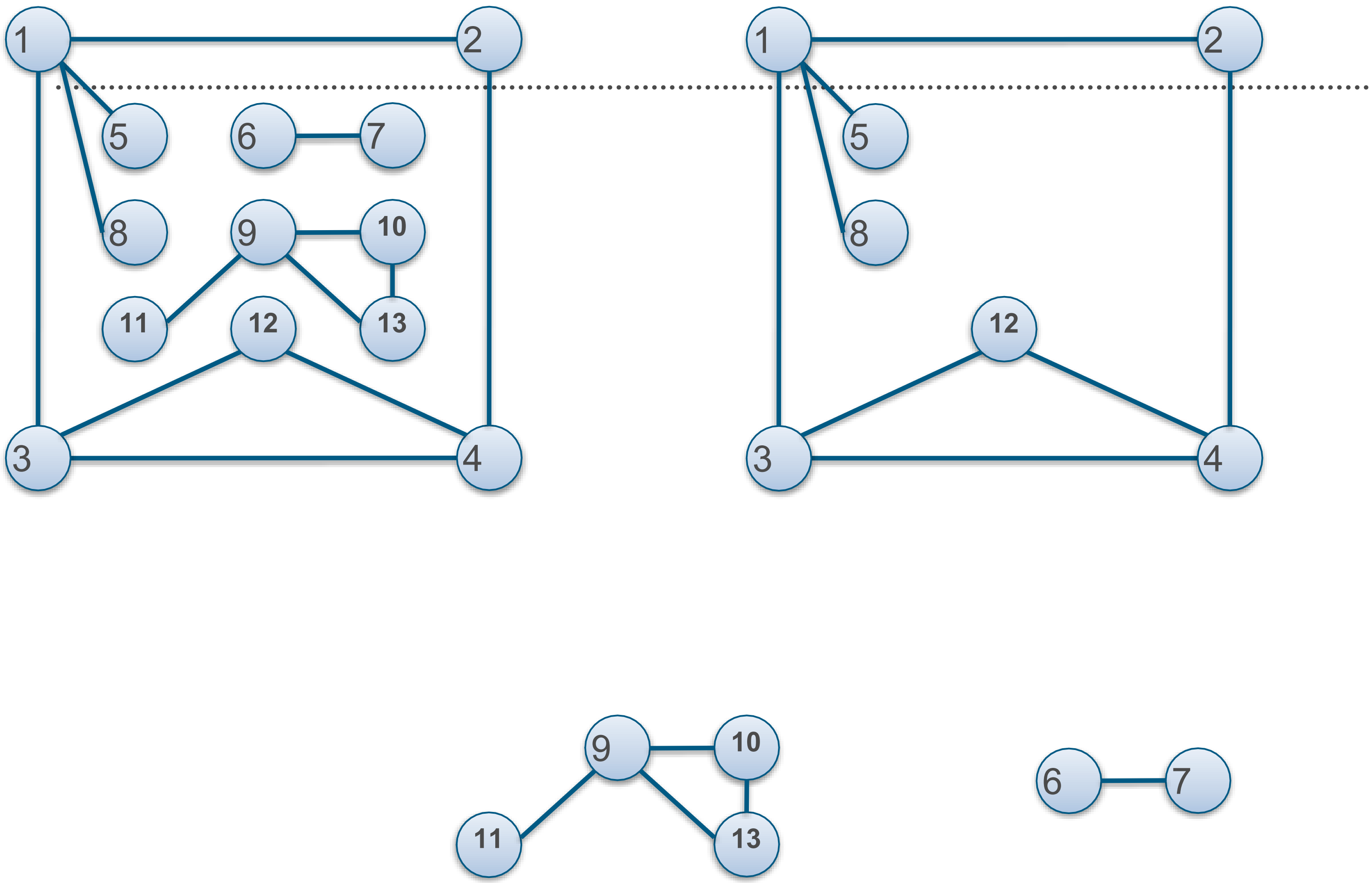
(22) 回路：第一个顶点和最后一个顶点相同的路径

基本概念23-24

(23) 如果图中任意两个顶点v_i和v_j都是连通的，则图G是**连通图**

(24) 无向图中的**极大**连通子图称为**连通分量**

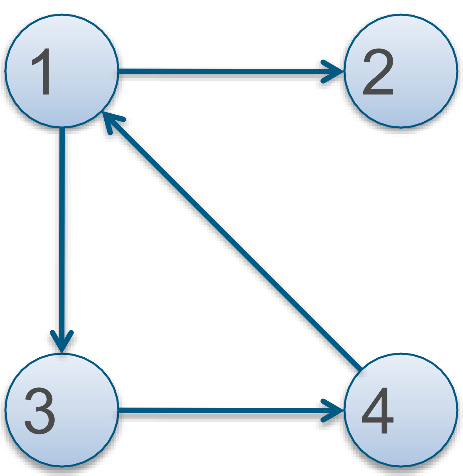


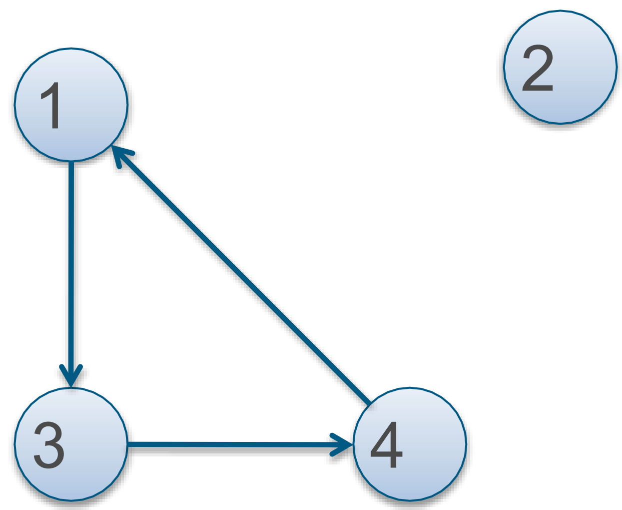

1 2 3 4 5 6 7 8 9 10 11 12 13
1 2 3 4 5 8 12
9 10 11 13
6 7

基本概念25-26

(25) 对于有向图中任意两个顶点v_i和v_j，如果从v_i到v_j和从v_j到v_i都有路径，则图G是**强连通图**

(26) 有向图中的极大连通子图称为**强连通分量**





思考：如何将左图修改为强连通图？

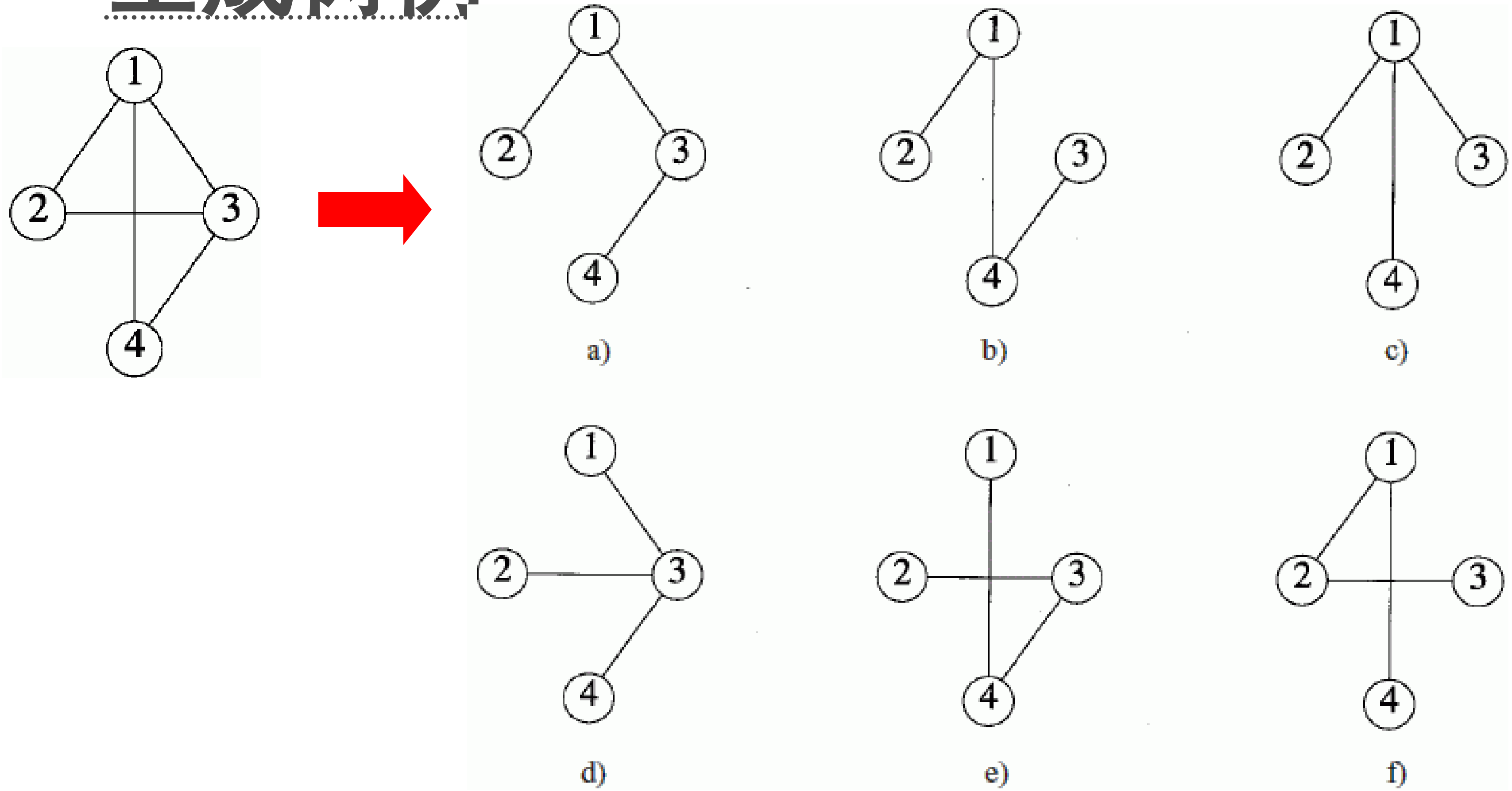
基本概念27-28

(27)一个连通图的**生成树**含有图中全部顶点，但只有足以构成一棵树的n-1条边

- 无环的无向连通图——树
- **生成树**（spanning tree）：包含G中所有顶点且是G的子图的树

(28)生成森林：略

生成树例



图的特性

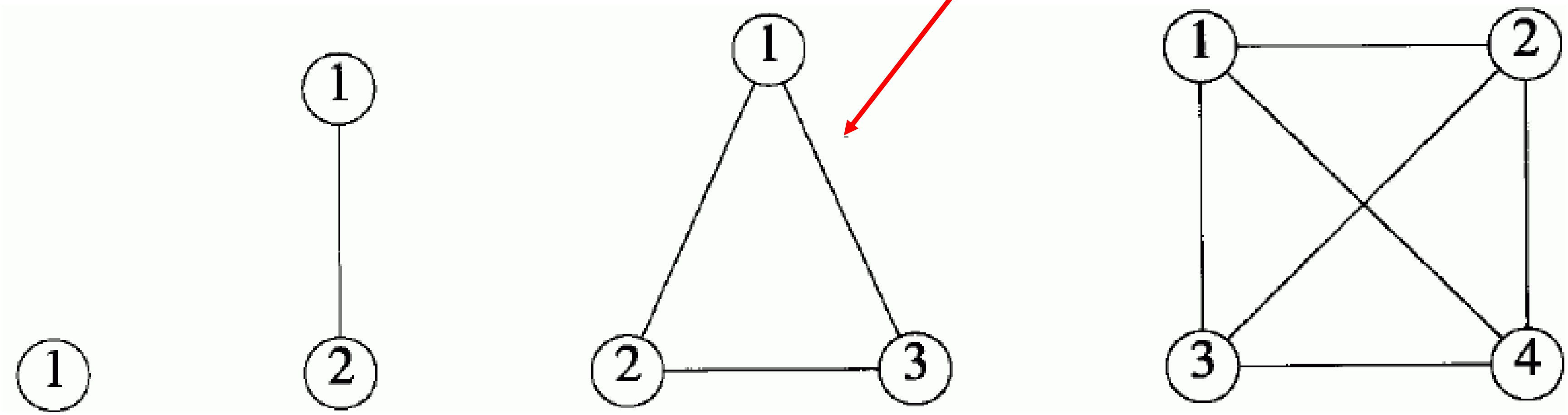
- G为无向图，顶点i的**度**（degree）：与顶点i相连的边的数目
- **特性1** 设G=(V, E)为无向图，|V|=n，|E|=e，d_i为顶点i的度，则有

1） $$\sum_{i=1}^{n} d_i = 2e$$

2） $0 \leqslant e \leqslant n(n-1)/2$

特性1示例

e=n(n-1)/2——完全图



特性2

- G为有向图
  入度（in-degree）：关联至顶点i的边的数目，d_i^{in}
  出度（out-degree）：关联于顶点i的边的数目，d_i^{out}

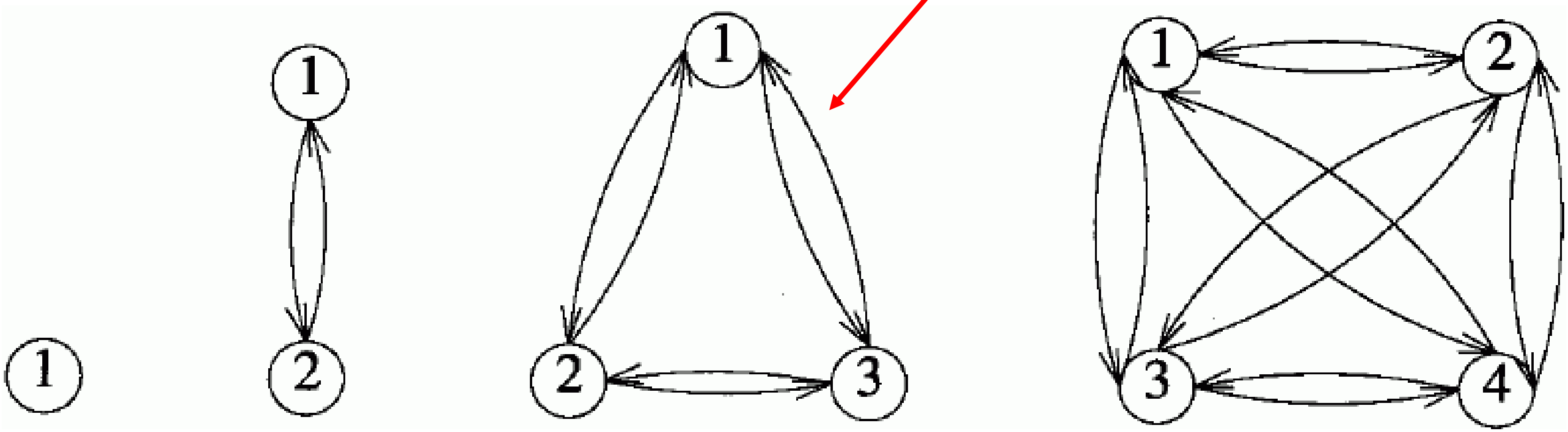
- 特性2　G=(V, E)为有向图，有：

1） $0 \leqslant e \leqslant n(n-1)$

2） $$\sum_{i=1}^{n} d_i^{in} = \sum_{i=1}^{n} d_i^{out} = e$$

特性2示例

e=n(n-1)——完全有向图



抽象数据类型

抽象数据类型*Graph*{

实例

顶点集合*V*和边集合*E*

操作

Create(*n*)：创建一个具有*n*个顶点、没有边的无向图

Exist(*i*, *j*)：如果存在边（*i*, *j*）则返回true，否则返回false

Edges()：返回图中边的数目

Vertices()：返回图中顶点的数目

Add(*i*, *j*)：向图中添加边（*i*, *j*）

Delete(*i*, *j*)：删除边（*i*, *j*）

Degree(*i*)：返回顶点*i*的度

}

有向图抽象数据类型

抽象数据类型*DiGraph*{

实例

顶点集合*V*和边集合*E*

操作

Create(*n*)：创建一个具有*n*个顶点、没有边的有向图

Exist(*i*, *j*)：如果存在边（*i*, *j*）则返回true，否则返回false

Edges()：返回图中边的数目

Vertices()：返回图中顶点的数目

Add(*i*, *j*)：向图中添加边（*i*, *j*）

Delete(*i*, *j*)：删除边（*i*, *j*）

InDegree(*i*)：返回顶点*i*的入度

OutDegree(*i*)：返回顶点*i*的出度 }

小结

- 线性表：单前驱、单后继
- 树：单前驱、多后继
- 图：节点之间的关系是任意的，图中任意两个数据元素之间都可能是相关的

- 图是真实世界中最一般的情况
- 曾经最热的研究：社会计算！

小结（续）

- 线性表可以是空表、树可以是空树，但图不能是空图，即**图的顶点集合必非空**
- 无向图考虑连通性，有向图考虑强连通性
- 对非强连通的有向图和非连通的无向图遍历将得到**生成森林**

特别说明

- 顶点对的细致表示方式
  - 有向图中：<i, j>
  - 无向图中：（i, j）

- 顶点对的通用表示方式
  - 有向图和无向图均为（i, j）

- 无特殊要求时，上述方式都可

主要内容

- 图的基本概念
- 图的存储及基本操作
- 图的遍历
- 最小生成树

三种存储方式

- 邻接矩阵
- 邻接压缩表

顺序存储

- 邻接链表
- 十字链表

链表存储

存储方式1：邻接矩阵

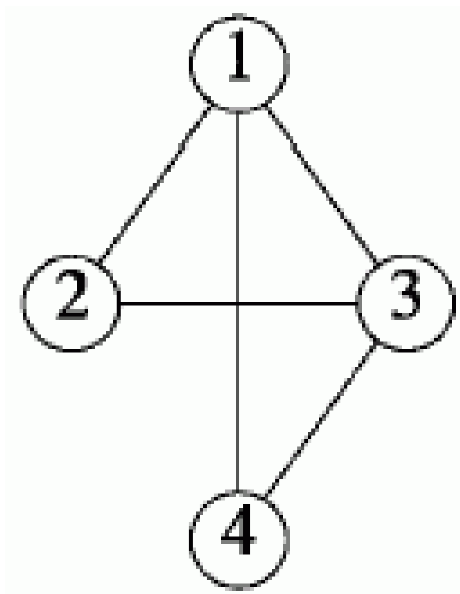
- 邻接矩阵（adjacency matrix）
  - 图G=(V, E)有n个顶点，V={1, 2, ..., n}
  - n×n的矩阵A
  - G为无向图

$$A(i,j)=\begin{cases}1,(i,j)\in E\text{或}(j,i)\in E\\0,\text{其他}\end{cases}$$

  - G为有向图

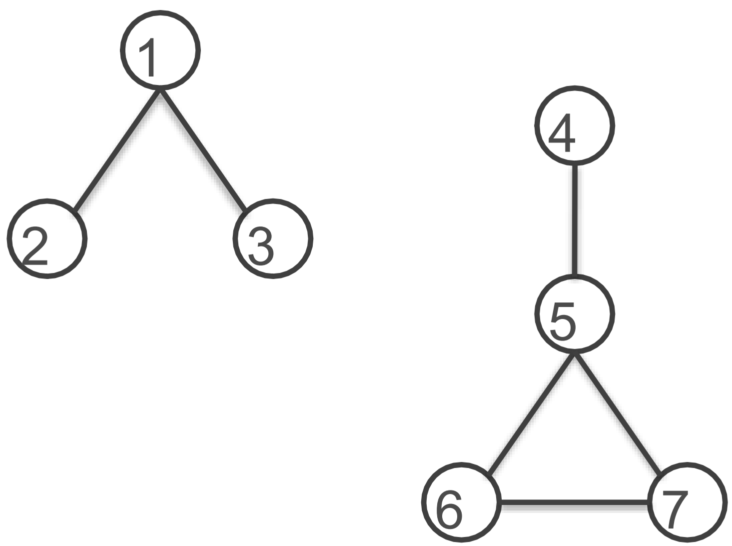
$$A(i,j)=\begin{cases}1,(i,j)\in E\\0,\text{其他}\end{cases}$$

邻接矩阵例



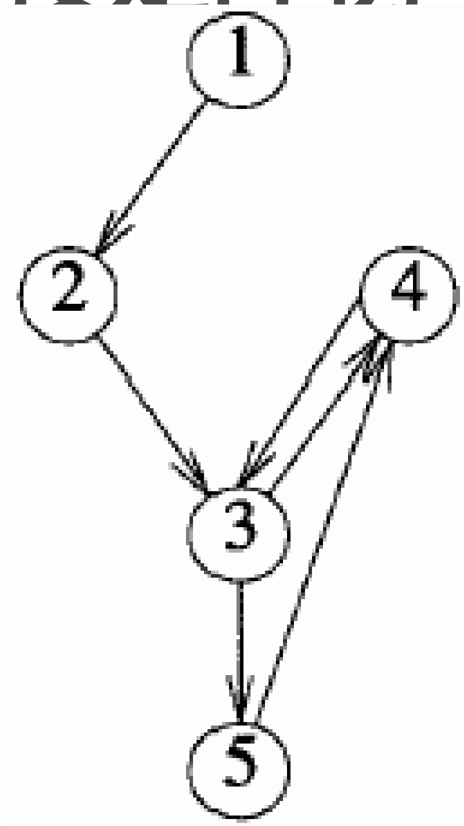
$$
\begin{array}{c c}
 & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}
\end{array}
$$

邻接矩阵例（续）



$$
\begin{array}{c|ccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
6 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
7 & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}
$$

邻接矩阵例（续）



$$
\begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 0 & 0 & 1 & 1 \\
4 & 0 & 0 & 1 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
$$

邻接矩阵特性

1） 对n顶点的无向图，A(i， i)=0，1≤i≤n

2） 无向图的邻接矩阵是**对称**的，
A(i， j)=A(j， i)，1≤i≤n

3） 对n顶点的无向图，$\sum_{j=1}^{n} A(i,j) = \sum_{j=1}^{n} A(j,i) = d_i$

行、列之和均等于顶点的度

邻接矩阵特性

4) 对n顶点的有向图

$$\sum_{j=1}^{n} A(i,j) = d_i^{out}$$

行之和等于顶点的出度

列之和等于顶点的入度

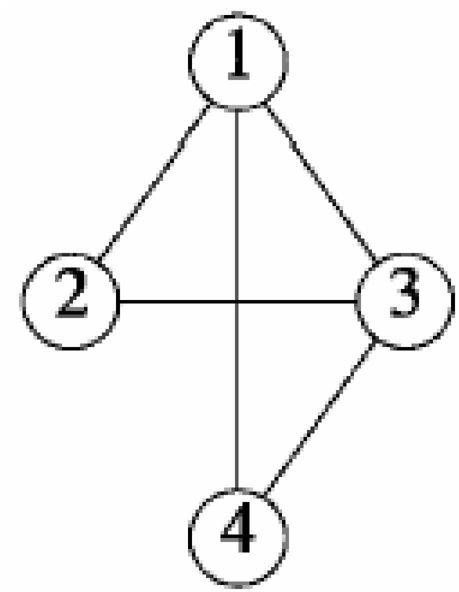
$$\sum_{j=1}^{n} A(j,i) = d_i^{in}$$

sizeof(int)=2

使用数组实现邻接矩阵

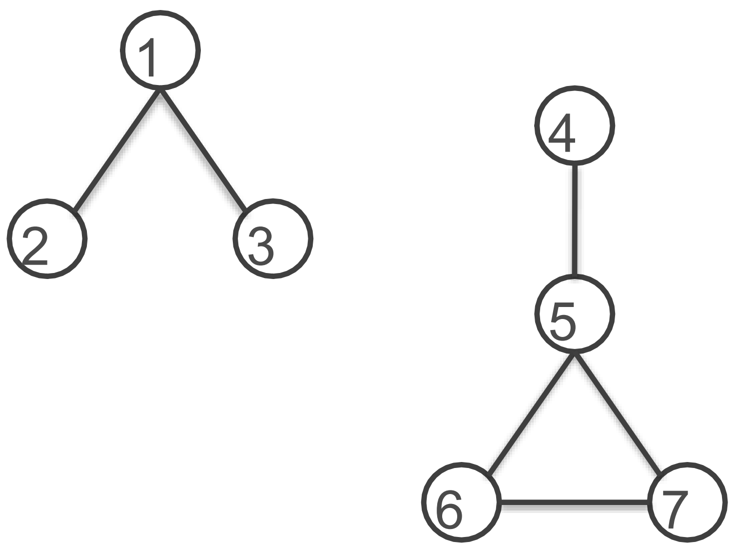
- 二维数组
  - a[n+1][n+1]，a[i][j]←A(i, j)，$2(n+1)^2$字节
  - a[n][n]，a[i-1][j-1]←A(i, j)，$2n^2$字节
- 优化策略
  - 对角线无需存储，节省2n字节，形成一个新的(n-1)×n的矩阵

数组实现示例



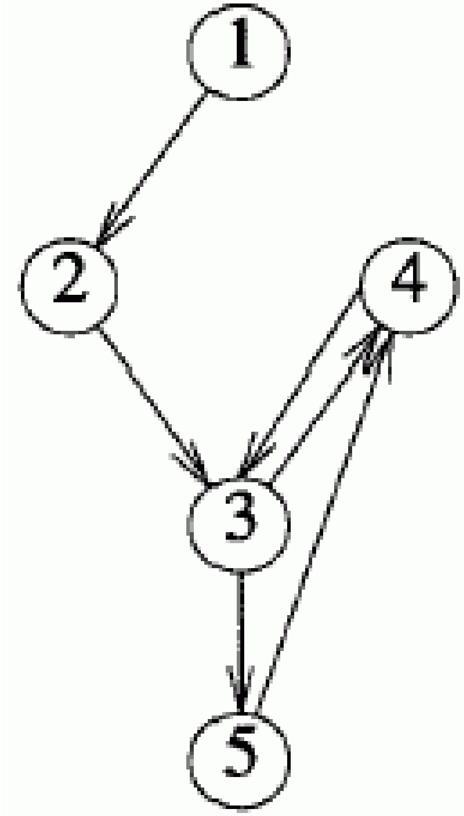
$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & \boxed{1} & 1 & 1 & 1 \\ 2 & \boxed{1} & \boxed{1} & 1 & 0 \\ 3 & \boxed{1} & \boxed{0} & \boxed{1} & 1 \end{array}$$

数组实现示例（续）



	1	2	3	4	5	6	7
1	1	1	1	0	0	0	0
2	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	1	1	0	0
5	0	0	0	0	1	1	1
6	0	0	0	0	1	1	1

数组实现示例（续）



$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 0 & 1 & 1 & 1 \\
4 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

可能的优化

- 矩阵元素取值只是0或1，只需一个二进制位即可保存
  - n(n-1)/8字节，节省16倍
  - **整数操作→位操作**，存储、检索复杂
- 无向图是对称矩阵，只保存上（下）三角即可
  - n(n-1)/16字节

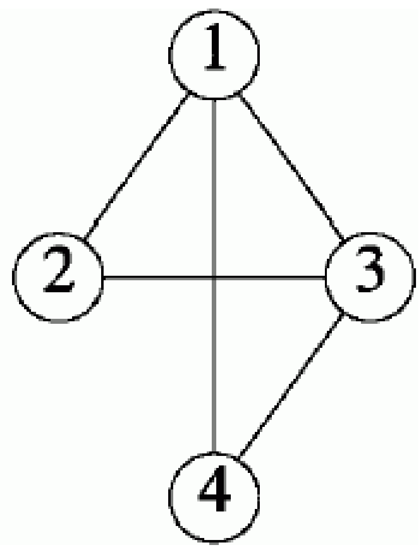
时间复杂性

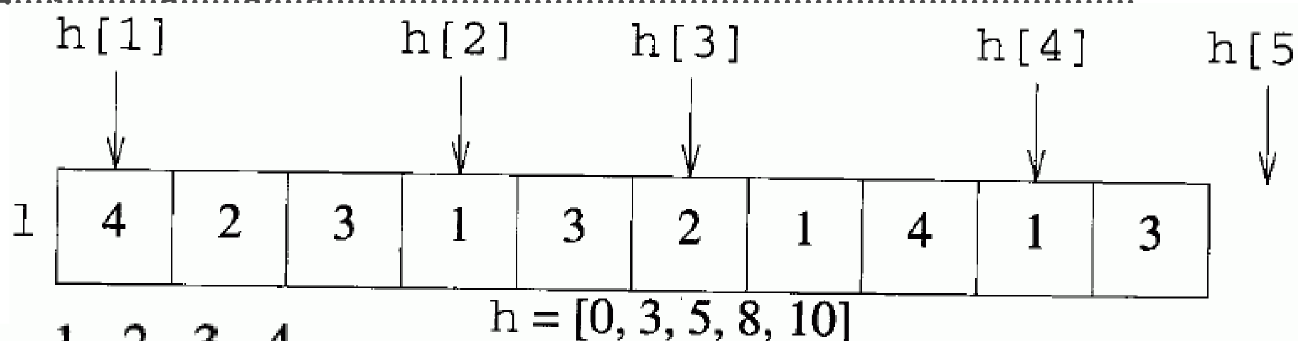
- 求给定节点的邻接节点集合，$\Theta(n)$
- 求图中总的边数，$\Theta(n^2)$
- 增加、删除一条边，$\Theta(1)$

存储方式2：邻接压缩表

- 图较为“稀疏”，邻接矩阵空间浪费
- 邻接压缩表，packed-adjacency-list
  - 使用一维数组h[0:n+1]，*l*[0:x]
  - 有向图：x=e-1；无向图：x=2e-1
  - *l*：保存邻接顶点集合
    顶点1的邻接顶点，顶点2的邻接顶点，...
  - h[i]：顶点i邻接顶点集合在*l*中的起始位置

邻接压缩表示例

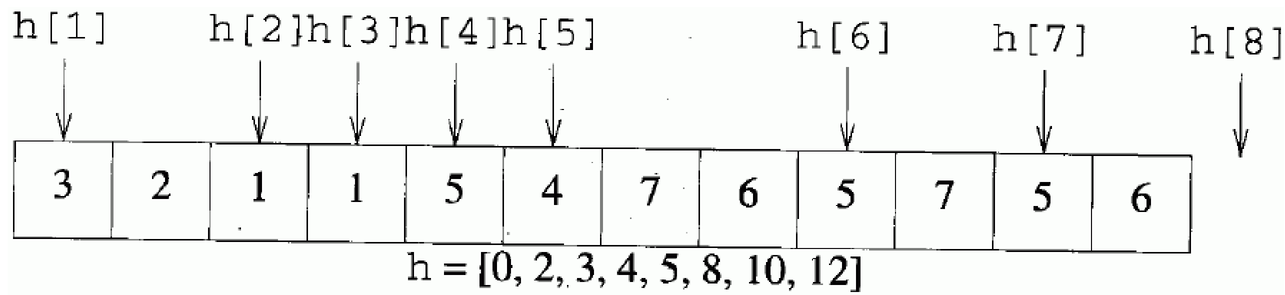


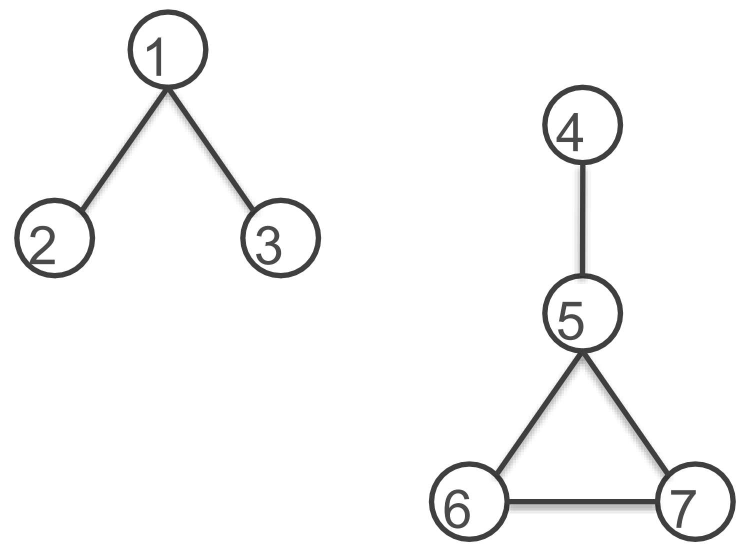


$$
\begin{array}{c}
\begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array}
\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}
\end{array}
$$

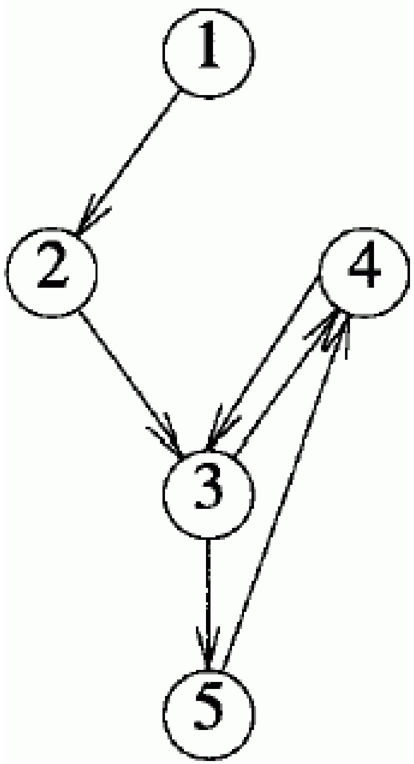
邻接压缩表示例（续）

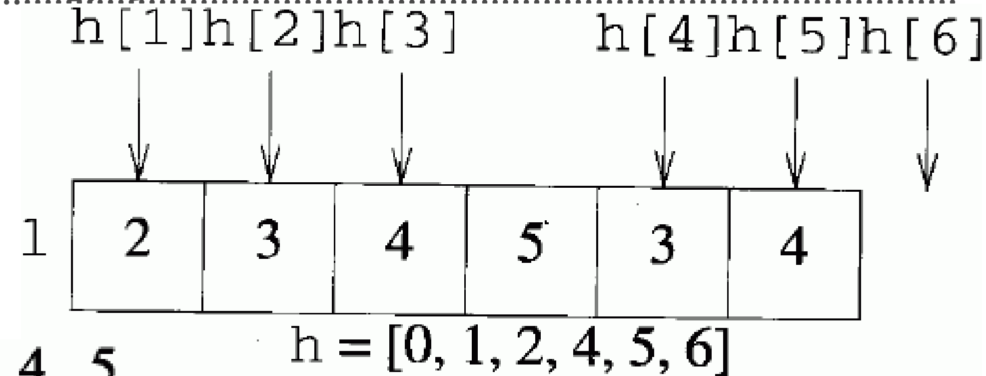
$$
\begin{array}{c|ccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
6 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
7 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
\end{array}
$$





邻接压缩表示例





$$
\begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 0 & 0 & 1 & 1 \\
4 & 0 & 0 & 1 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

优化

- 数组元素的压缩
  - h：取值范围0~2e， $\lceil \log_2(2e+1) \rceil$ 位即可
  - *l*：取值范围1~n， $\lceil \log_2 n \rceil$ 位即可
  - 总空间：

$$(n+1)\lceil \log_2(2e+1) \rceil + 2e\lceil \log_2 n \rceil = O((n+e)\log n)$$

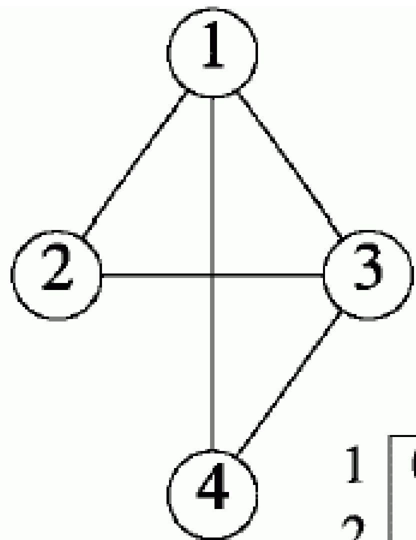
时间复杂性

- e远远小于n^2→空间复杂性远优于邻接矩阵
- 顶点i的度h[i+1]-h[i]，边总数h[n+1]/2，$\Theta(1)$
- 增加、删除一条边，O(n+e)

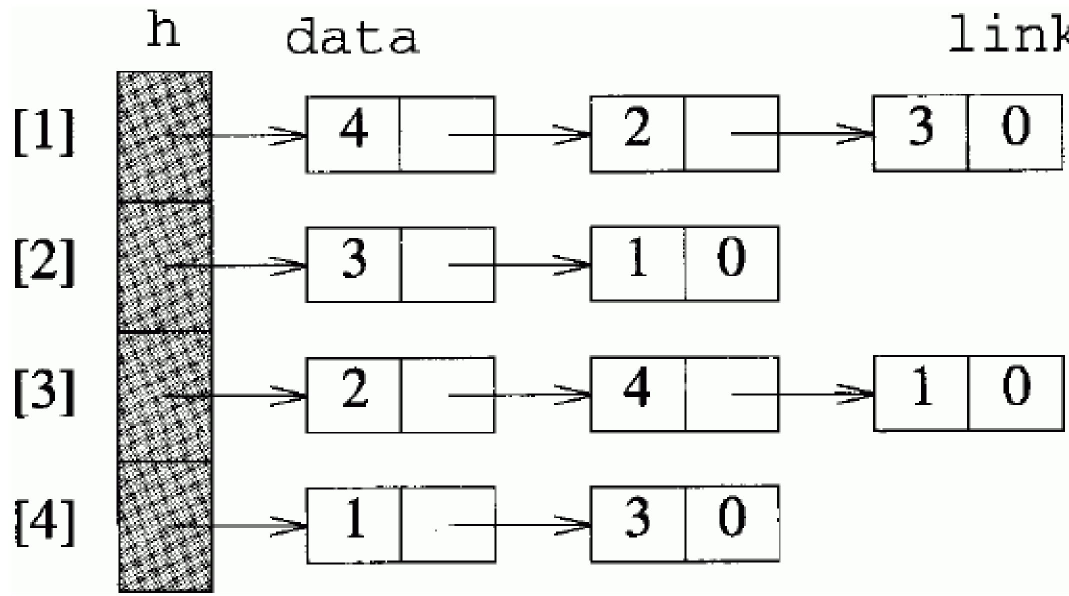
存储方式3：邻接链表

- 邻接压缩表插入、删除复杂
- 邻接链表（linked-adjacency-list）
  - 邻接表（邻接顶点集合）用链表保存
  - Chain<int>类型的数组h
    h[i]——顶点i的邻接表

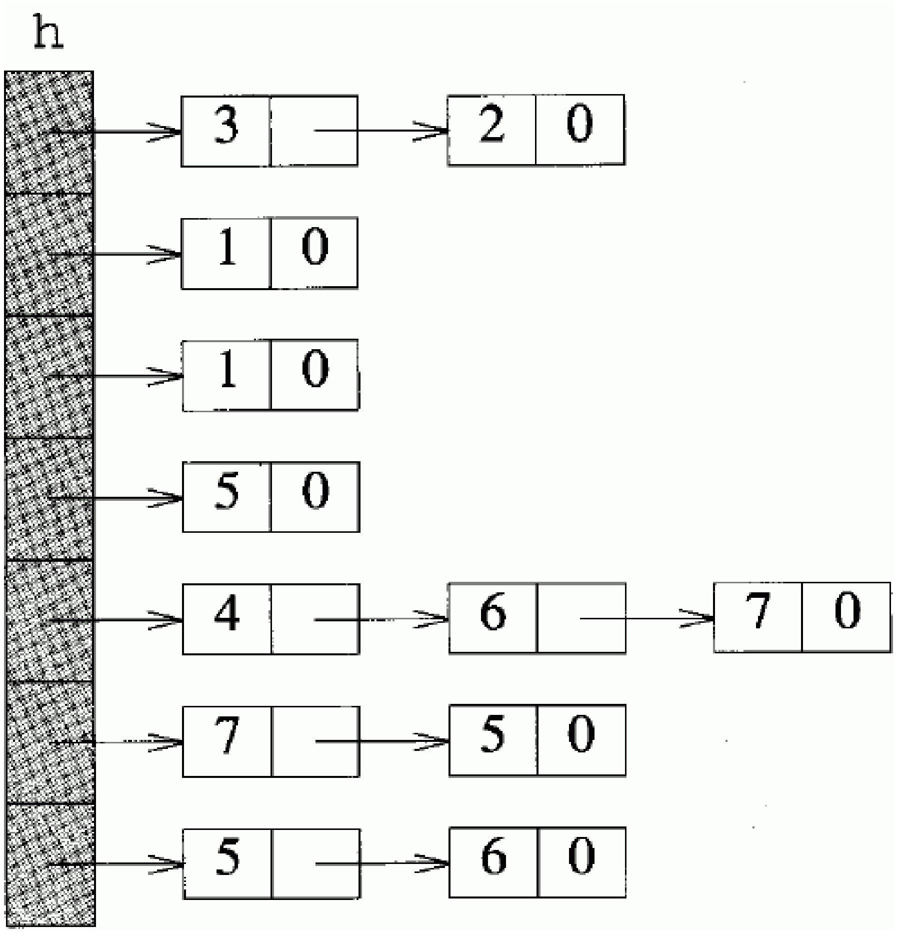
邻接链表示例



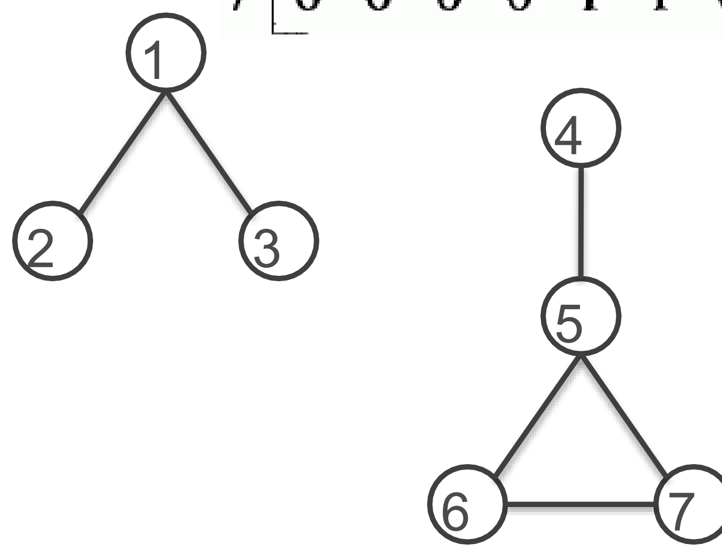
$$\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 1 & 1 \\ 2 & 1 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 1 & 0 & 1 & 0 \end{array}$$



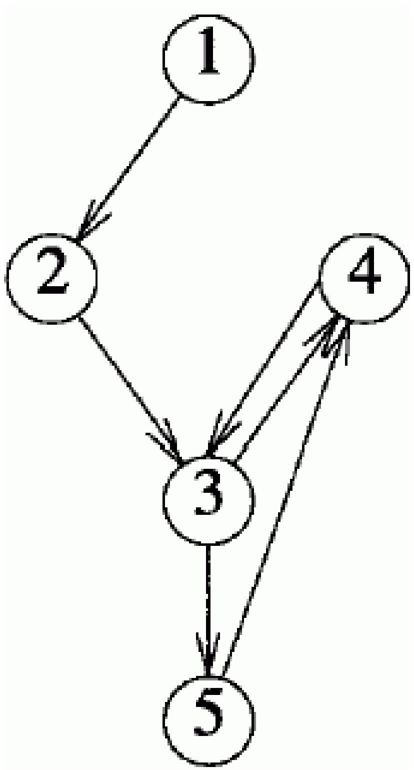
邻接链表示例（续）



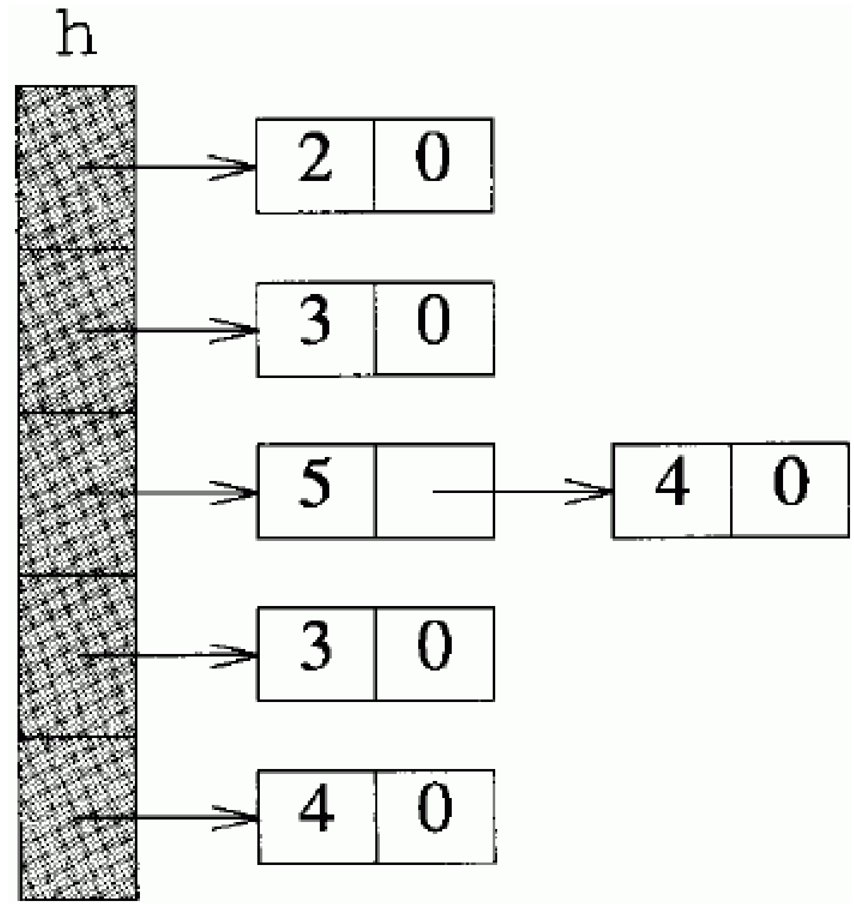
$$
\begin{array}{c|ccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
6 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
7 & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}
$$



邻接链表示例（续）



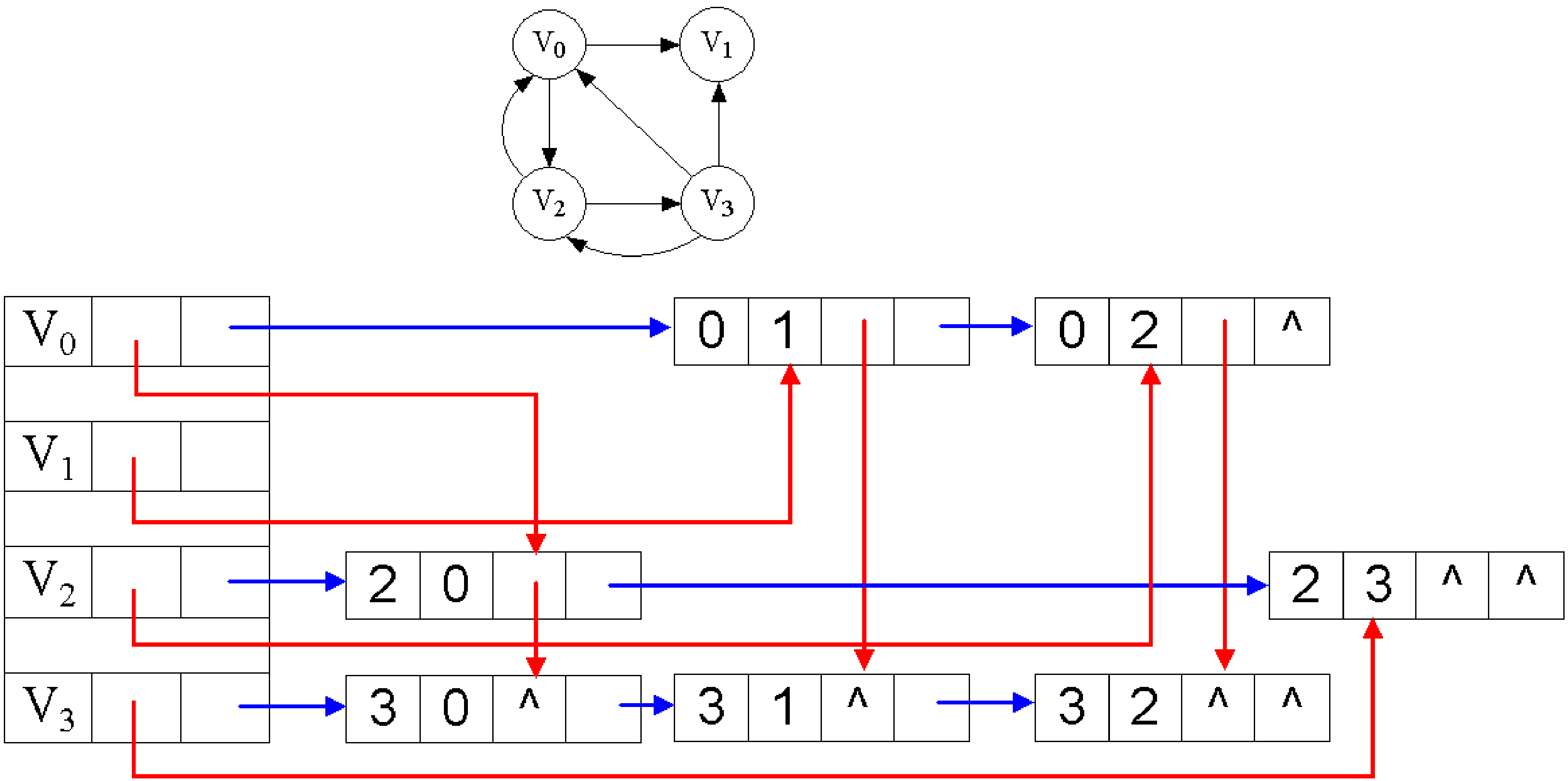
$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}
\end{array}
$$



复杂性分析

- 空间复杂性
  - 2(n+m+1)，有向图：m=e，无向图：m=2e
- 时间复杂性
  - 插入、删除边高效
  - 求邻接顶点集合：Θ(n)
  - 求边的总数：Θ(e)

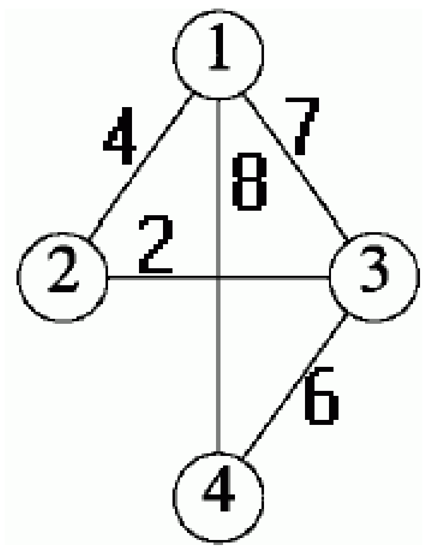
存储方式4：十字链表



网络的描述

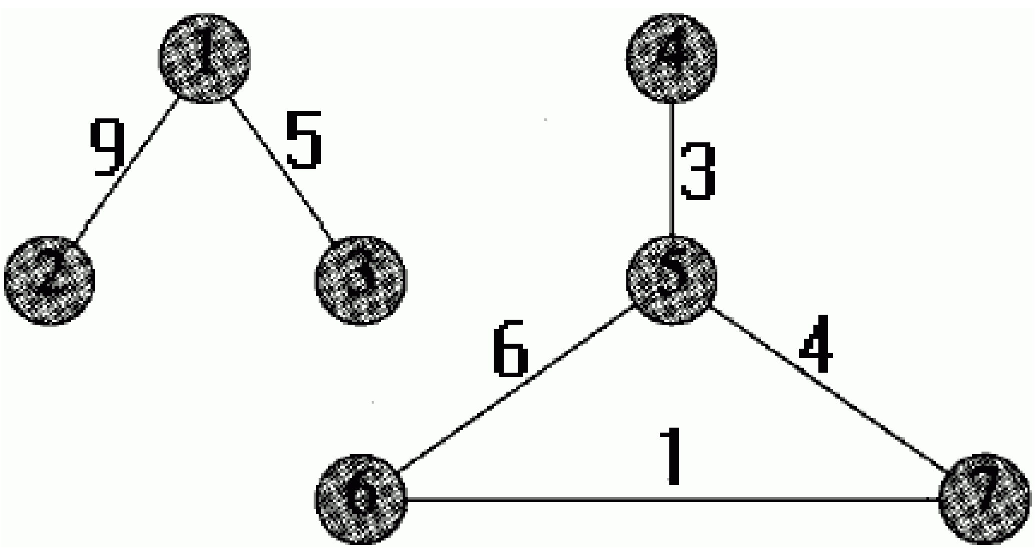
- 图描述简单扩充，描述边的耗费
- 耗费邻接矩阵（cost-adjacency-matrix）C
  - A(i, j)=1, C(i, j)——对应边的耗费（权重）
  - A(i, j)=0, C(i, j)=∞——不存在边

耗费邻接矩阵例



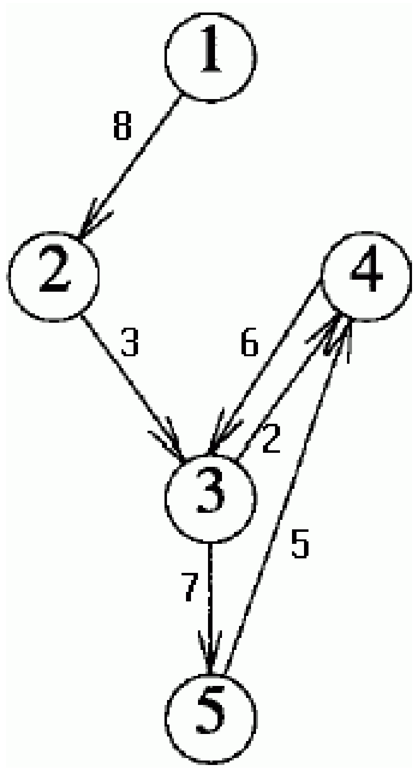
$$
\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
\hline
1 & \infty & 4 & 7 & 8 \\
2 & 4 & \infty & 2 & \infty \\
3 & 7 & 2 & \infty & 6 \\
4 & 8 & \infty & 6 & \infty
\end{array}
$$

耗费邻接矩阵例（续）



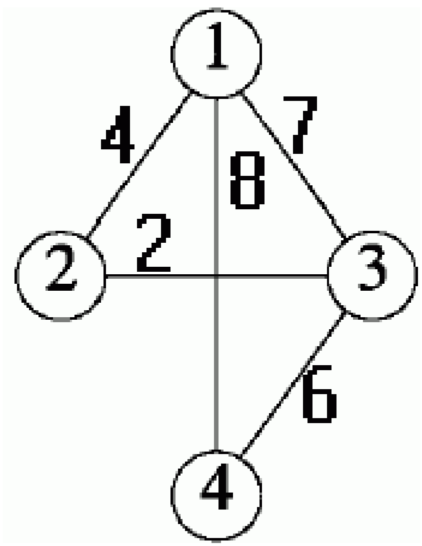
$$
\begin{array}{c}
 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\left[\begin{array}{ccccccc}
\infty & 9 & 5 & \infty & \infty & \infty & \infty \\
9 & \infty & \infty & \infty & \infty & \infty & \infty \\
5 & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & 3 & \infty & \infty \\
\infty & \infty & \infty & 3 & \infty & 6 & 4 \\
\infty & \infty & \infty & \infty & 6 & \infty & 1 \\
\infty & \infty & \infty & \infty & 4 & 1 & \infty
\end{array}\right]
\end{array}
$$

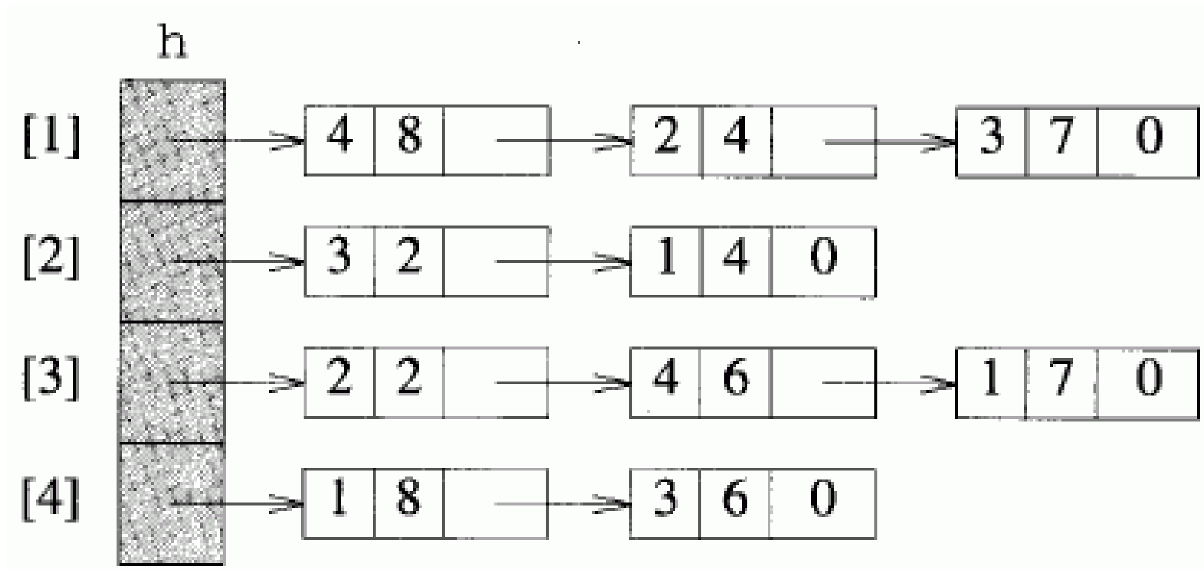
耗费邻接矩阵例（续）



$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
\infty & 8 & \infty & \infty & \infty \\
\infty & \infty & 3 & \infty & \infty \\
\infty & \infty & \infty & 2 & 7 \\
\infty & \infty & 6 & \infty & \infty \\
\infty & \infty & \infty & 5 & \infty
\end{bmatrix}
\end{array}
$$

邻接链表实现





小结

- 邻接矩阵适用于稠密图、邻接表适用于稀疏图

- 需要熟练掌握前三种表示方法

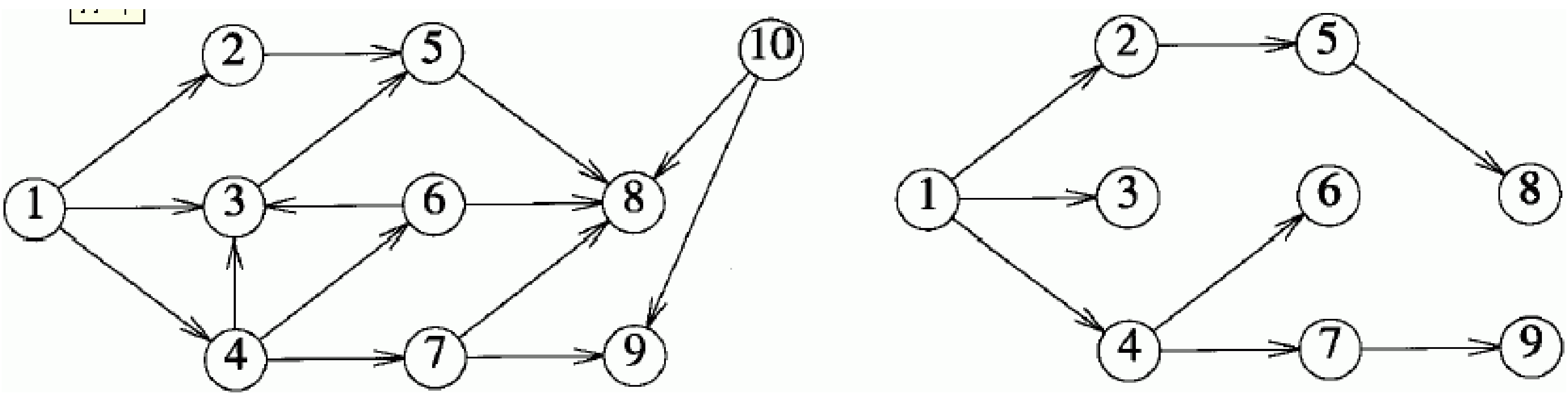
主要内容

- 图的基本概念
- 图的存储及基本操作
- 图的遍历
- 最小生成树

图的遍历

- 从一个给定节点开始，访问所有可达节点，且每个顶点仅访问一次
- 宽度优先搜索（Breadth-First Search，BFS）
  - 开始顶点1
  - 1可达的顶点集合{2，3，4}
  - {2，3，4}可达的顶点集合{5，6，7}
  - {5，6，7}可达的顶点集合{8，9}

宽度优先搜索示例



宽度优先搜索算法伪代码

```
/ /从顶点v 开始的宽度优先搜索
把顶点v标记为已到达顶点；
初始化队列Q，其中仅包含一个元素v;
while（Q不空）{
  从队列中删除顶点w;
  令u 为邻接于w 的顶点；
  while（u）{
        if（ u 尚未被标记）{
              把u 加入队列；
              把u 标记为已到达顶点； }
        u = 邻接于w 的下一个顶点；
  }
}
```

定理12-1

- **定理12-1**
  设N 是一个任意的图、有向图或网络，
  v 是N 中的任意顶点
  上述伪代码能够标记从v 出发可以到达的所有顶点（包括顶点v）。

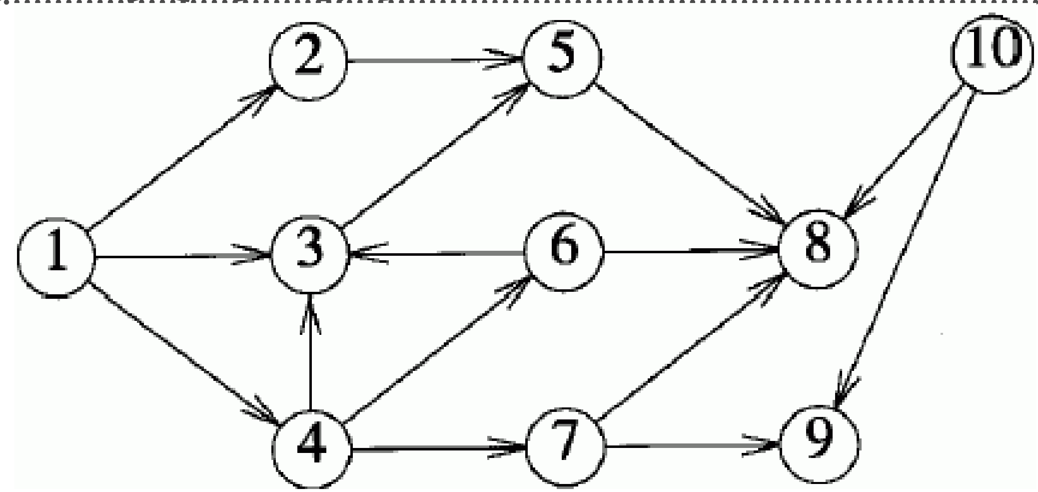
复杂性分析

- 可达顶点都被标记
- 每个顶点只加入队列一次，也只删除一次——处理一次
- 处理顶点——遍历它所有邻接顶点
- 邻接矩阵——Θ(sn)
- 邻接链表——$\Theta(\sum_i d_i^{out})$

深度优先搜索

- Depth-First Search，DFS
  - v为开始顶点，首先标记v
  - 选择一个与v邻接，且尚未标记的顶点u
  - 像处理v一样对u进行处理——DFS递归调用
  - 对u的处理完毕后，选择另一个与v相邻且未标记的顶点，继续搜索
  - 若不存在，搜索中止

深度优先搜索例



1→2→5→8

→3

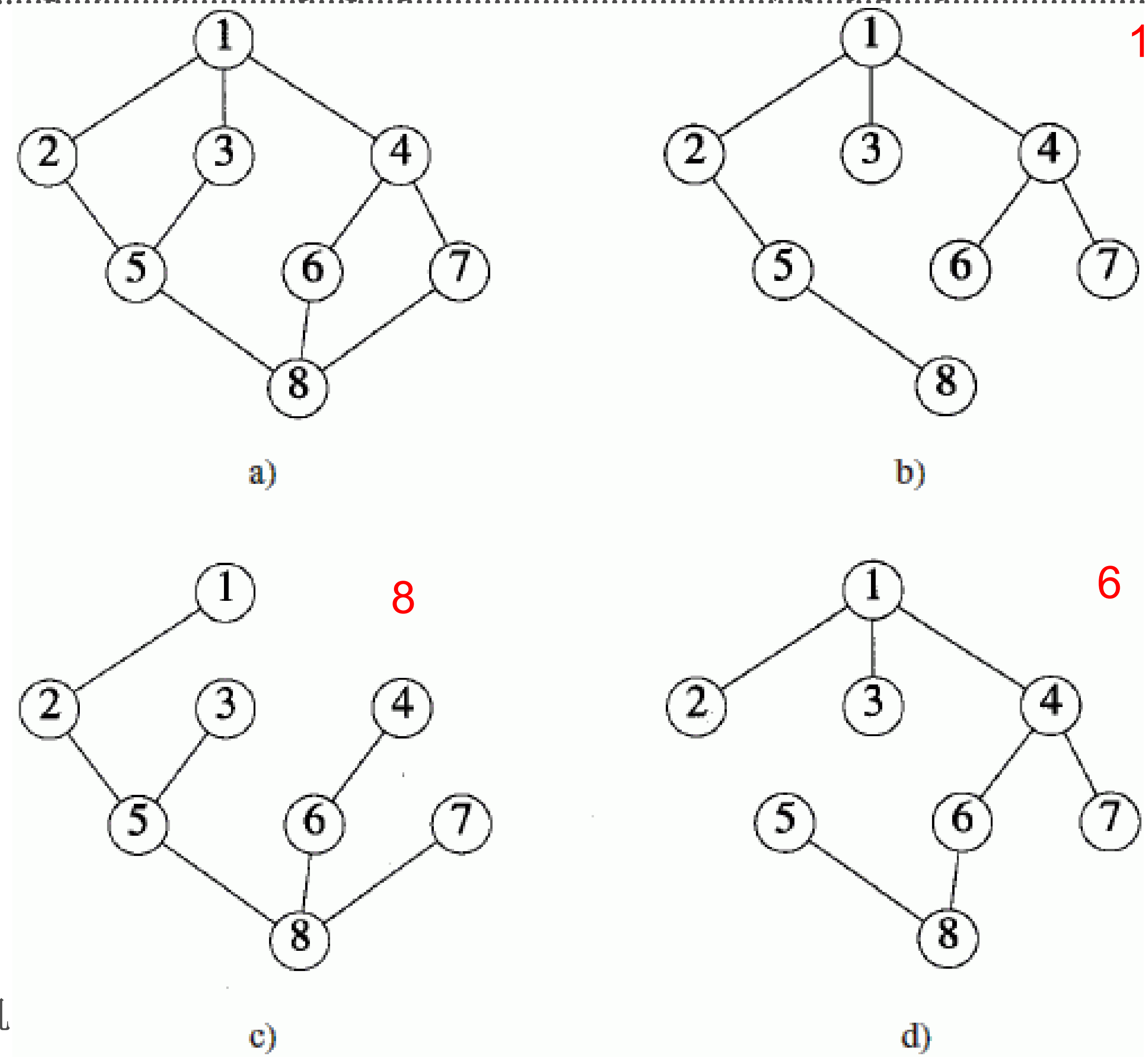
→4→6

→7→9

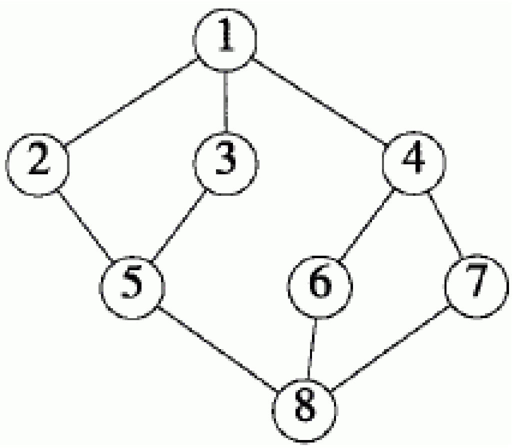
生成树

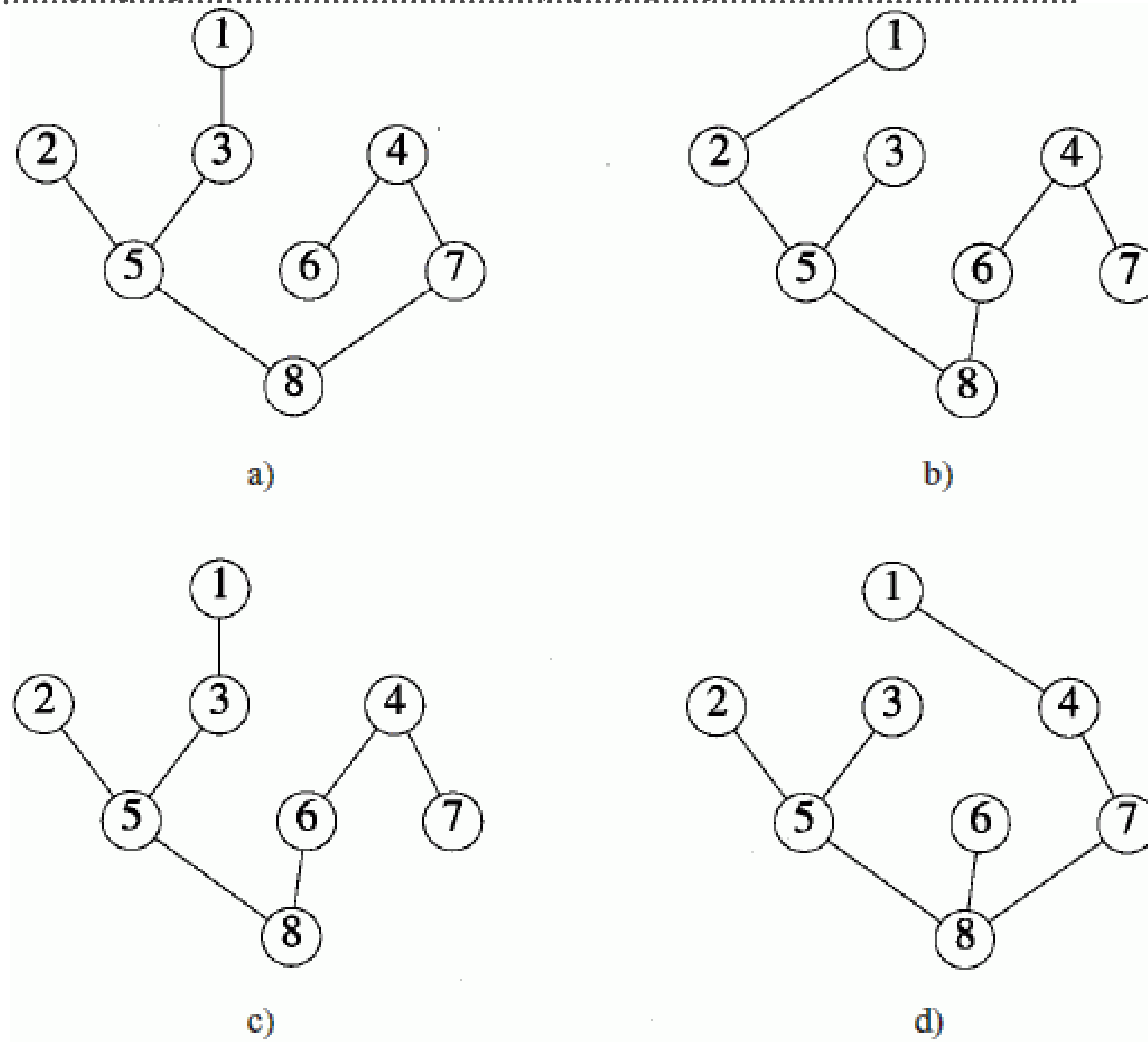
- 对连通图进行BFS，所有顶点都被标记
- 到达一个新的顶点，要通过相应的边
- 恰好n-1条边——连通子图——生成树

宽度优先搜索构造生成树



深度优先搜索构造生成树





主要内容

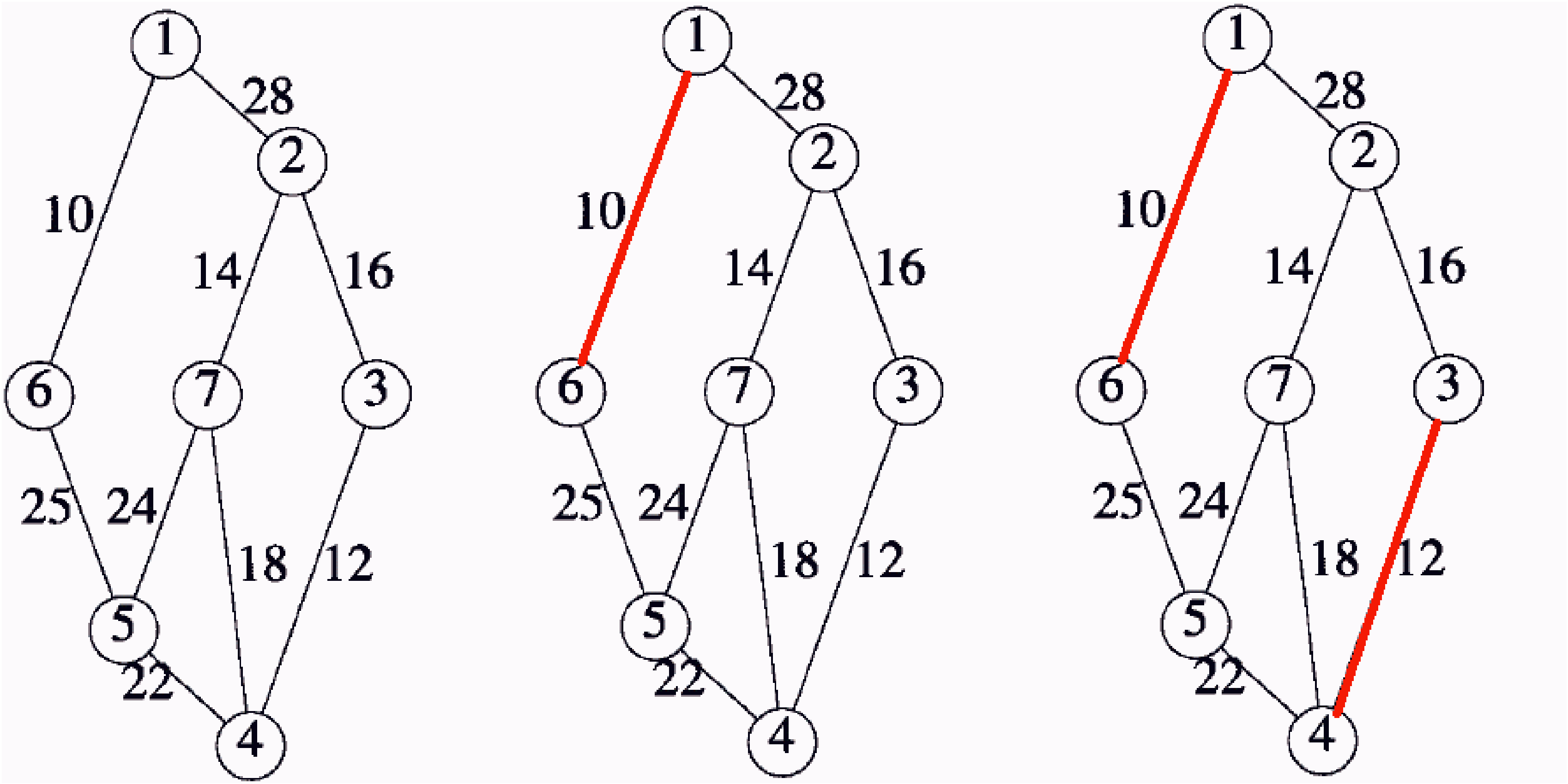
- 图的基本概念
- 图的存储及基本操作
- 图的遍历
- 最小生成树

最小耗费生成树

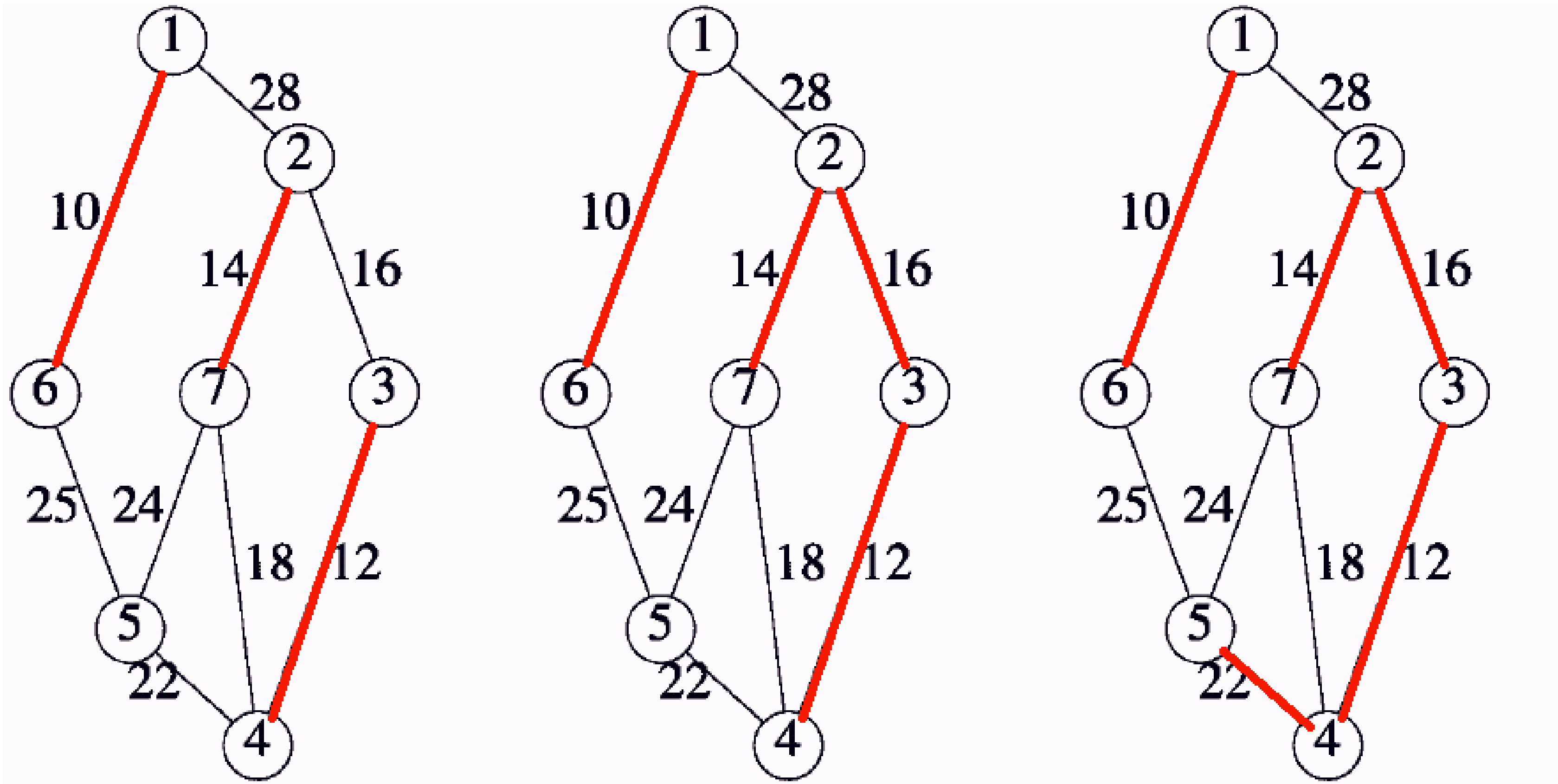
- 问题关键：如何选择生成树的n-1条边

1. Kruskal
   - 每个步骤选择一条边加入生成树
   - 贪心准则：不会产生环路，且耗费最小
   - 可按耗费递增顺序考察每条边
     - 若产生环路，丢弃
     - 否则，加入

例

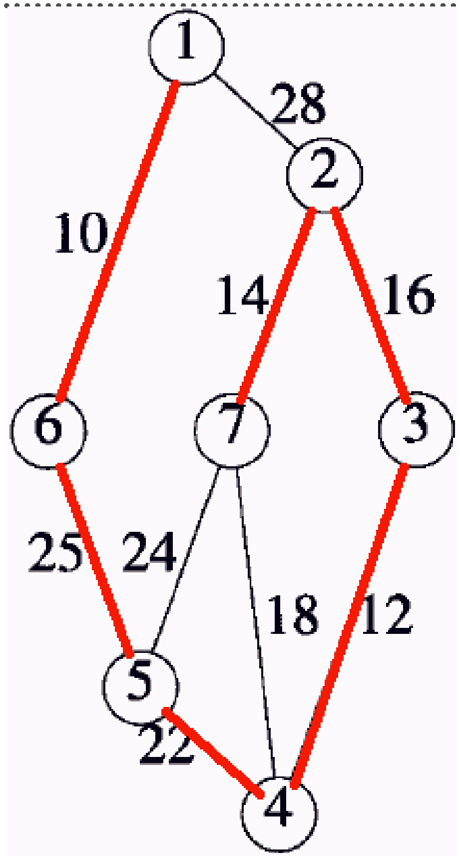

1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4
1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4
1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4

例（续）



1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4
1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4
1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4

例（续）



伪代码

/ /在一个具有*n* 个顶点的网络中找到一棵最小生成树

令*T*为所选边的集合，初始化*T*=Φ

令E为网络中边的集合

while（*E*≠Φ）&&（|*T*|≠*n*-1）{

令(*u*, *v*)为*E*中代价最小的边

E=*E*-{（*u*, *v*）} / /从*E*中删除边

if（(*u*, *v*)加入*T*中不会产生环路）将(*u*, *v*)加入*T*

}

i f（|*T*| == *n*-1） *T*是最小耗费生成树

else 网络不是连通的，不能找到生成树

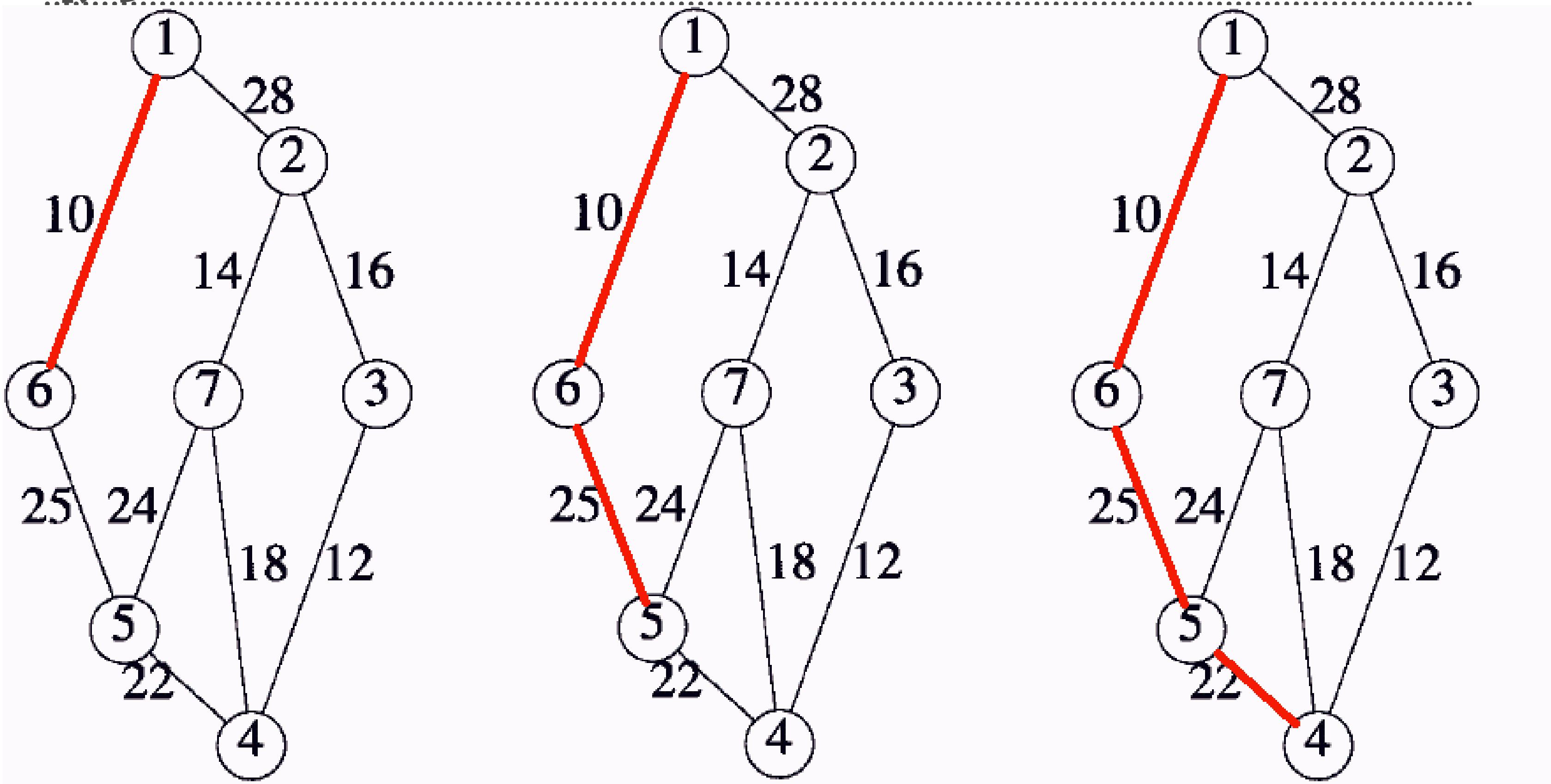
Prim算法

- 贪心准则
  - 加入后仍形成树，且耗费最小
  - 始终保持树的结构——Kruskal算法是森林
- 算法过程
  - 从单一顶点的树T开始
  - 不断加入耗费最小的边(u，v)，使T∪{(u，v)}仍为树——u、v中必然有一个已经在T中，另一个不在T中

Prim算法伪代码

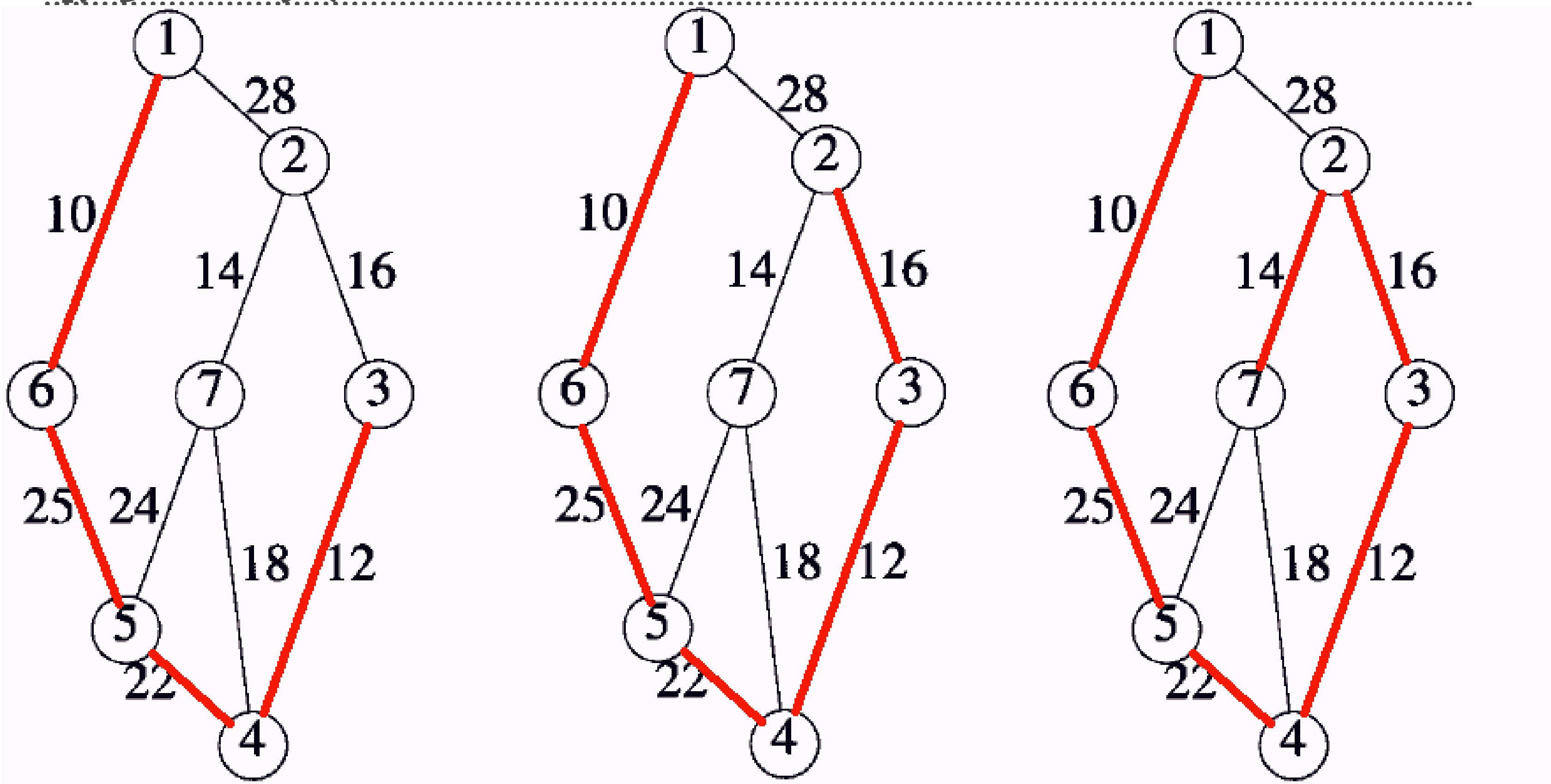
```
//假设网络中至少具有一个顶点
设T为所选择的边的集合，初始化T=Φ
设TV为已在树中的顶点的集合，置TV={1}
令E为网络中边的集合
while（E<>Φ）&&（|T|<>n-1）{
  令(u, v)为最小代价边，其中u∈TV, v∈TV
  if（没有这种边）break
  E=E-{(u, v)}//从E中删除此边
  在T中加入边(u, v)
}
if（|T|==n-1）T是一棵最小生成树
else 网络是不连通的，没有最小生成树
```

例



1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4
1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4
1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4

例（续）



1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4
1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4
1
28
2
10
14
16
6
7
3
25
24
18
12
5
22
4

本章结束