

1001010101010000100111110101000011010101011010110101010101010100

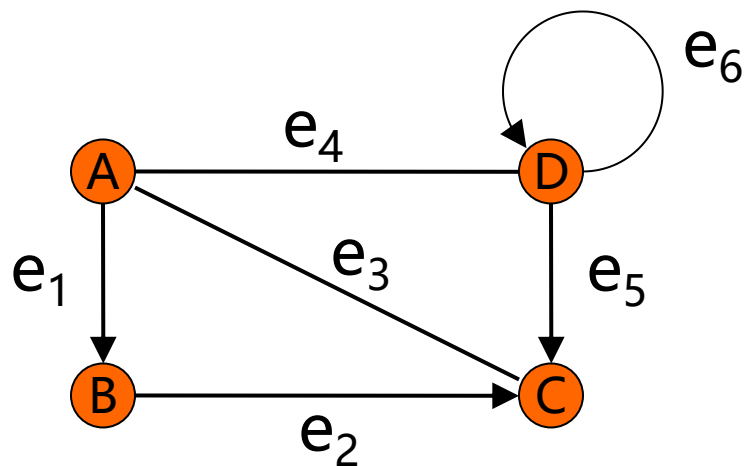
图论与代数结构

初识图论与道路/回路

崔 勇

清华大学计算机系
网络技术研究所

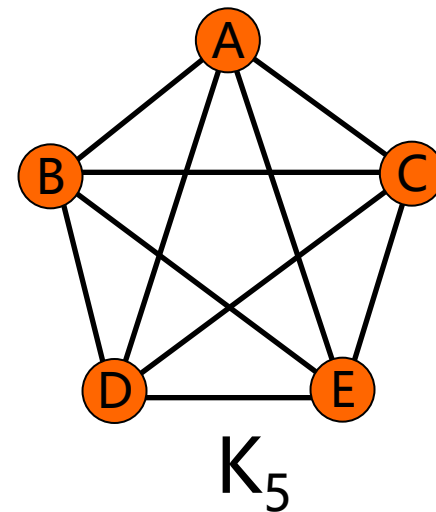
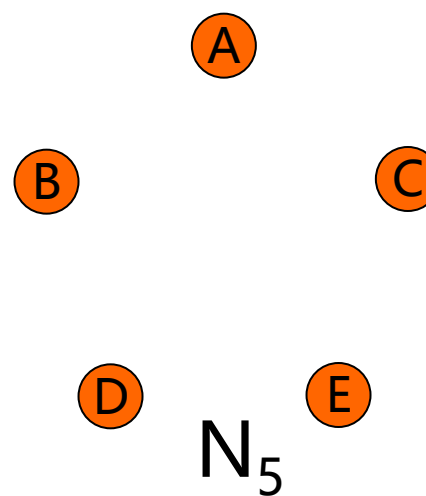
前情回顾



- A是 e_1 的始点，B是 e_1 的终点
- A是B的直接前驱，C是B的直接后继
- A, C是相邻结点，是无向边 e_3 的端点
- e_6 是自环，与D关联，没有重边
- 度数（出度，入度） $\sum_{v \in V(G)} d(v) = 2m$

简单图

- 无重边、无自环的无向图



- 空图 N_n ，完全图 K_n ，二分图，完全二分图 $K_{m,n}$

主要内容

- 图的运算（并、交、对称差、同构）
- 图的代数表示方法
 - 各种方法的特点（优缺点）
- 道路与回路
 - 概念和定义（带弦回路证明）
 - 判定：三个核心算法

学好？
不是学会知识
而是学会创新



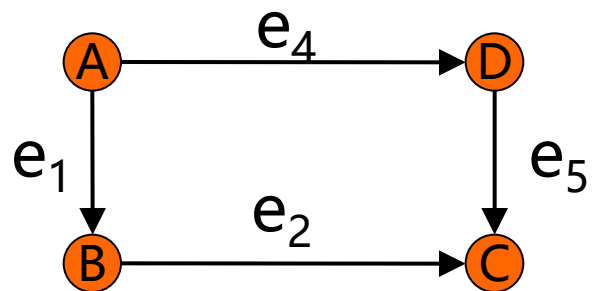
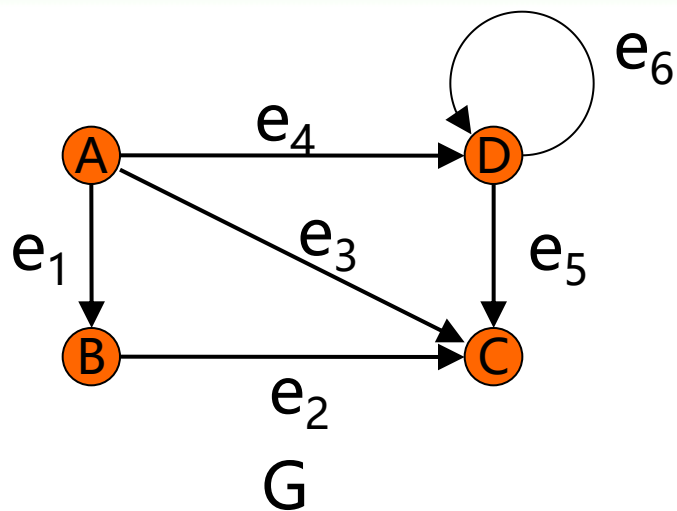
图的定义

站在巨人肩膀上发明创造
集合：子集、运算

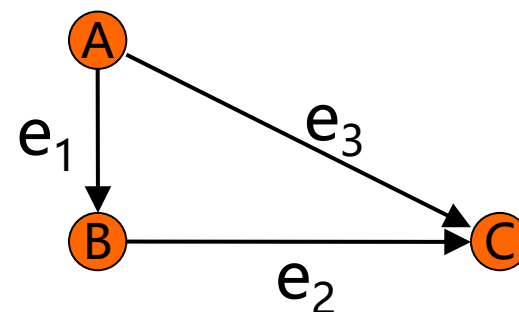
- 子图

- 对图 $G=(V,E)$ 与 $G'=(V',E')$ ，若 V' 包含于 V ， E' 包含于 E ，则称 G' 为 G 的一个子图（真子图）
- 特别若 $V=V'$ ，则称 G' 为 G 的支撑子图或生成子图
- 若 E' 包含了 G 在结点子集 V' 的所有边，则称 G' 为 G 的导出子图
- G 的空支撑子图与 G 本身称为 G 的平凡子图

图的定义



G 的支撑子图

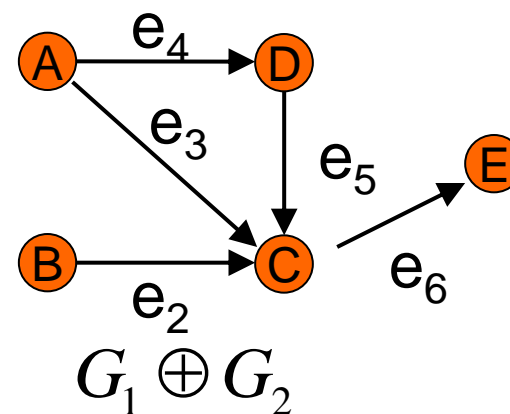
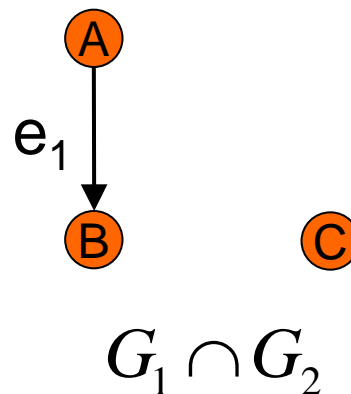
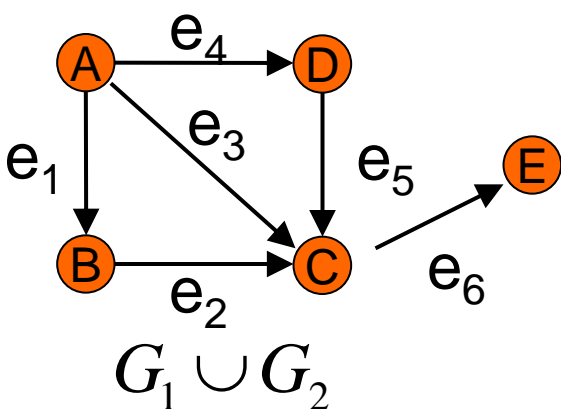
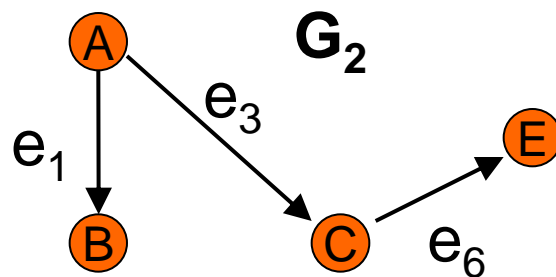
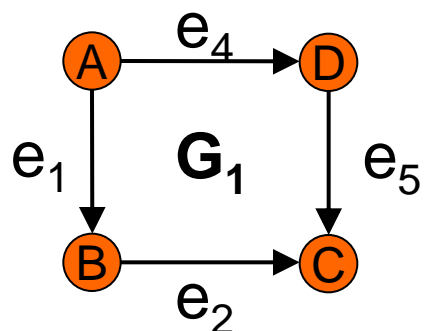


G 的导出子图

(边全吗?)

图的定义

- 图的并、交、对称差
 - 两个图 $G_1=(V_1, E_1)$, $G_2=(V_2, E_2)$



$$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$$

$$G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$$

$$G_1 \oplus G_2 = (V_1 \cup V_2, E_1 \oplus E_2)$$

是否可能增加其他形式的定义 (如点交边并) ?



图的定义

- 图的差
 - $G - H$
 - 对于 G 的子图 H , $G - H$ 表示在 G 中删除 H 中的各条边得到的图
 - 补图
 - 对于简单图 G , 完全图 $K_n - G$, 称为 G 的补图。
 - $G - v$ 是 G 的导出子图, $G - e$ 是 G 支撑子图。



图的定义

- 对于无向图G, 结点v的邻点集

$$\Gamma(v) = \{u \mid (u, v) \in E\}$$

- 有向图呢?
- 设v是有向图G的一个结点, 则

$$\Gamma^+(v) = \{u \mid (v, u) \in E\}$$

$$\Gamma^-(v) = \{u \mid (u, v) \in E\}$$

分别称为v的直接后继集（外邻集）与直接前趋集（内邻集）

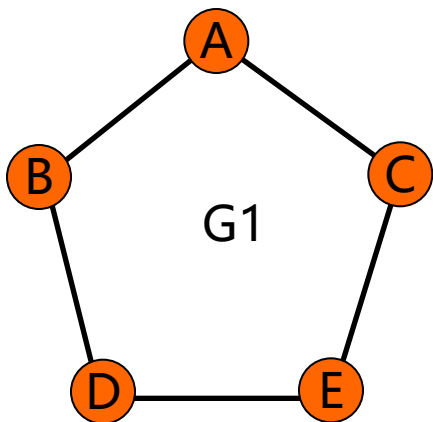
图的定义

相同的数，相同的集合，**相同的图**？长得一样？

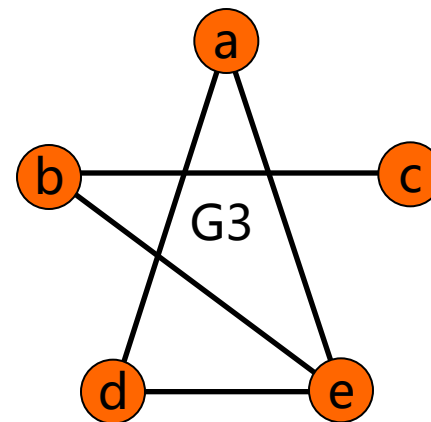
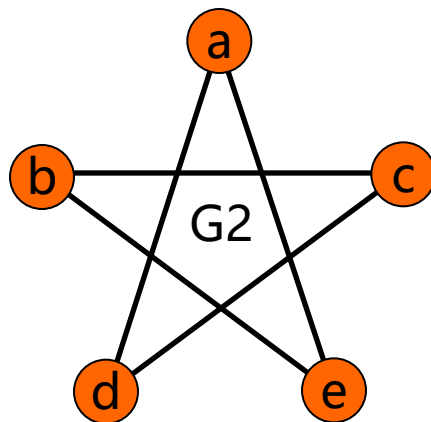
- 同构

- 两个图 $G_1=(V_1, E_1)$, $G_2=(V_2, E_2)$, 若在 V_1 与 V_2 之间存在双射 f , 当且仅当 (u,v) 为 G_1 的边时, $(f(u),f(v))$ 为 G_2 的边。称图 G_1 与 G_2 同构。记作 $G_1 \cong G_2$

透过现象
看本质？



$f(A)=a, f(B)=d, f(C)=e, f(D)=c, f(E)=b$

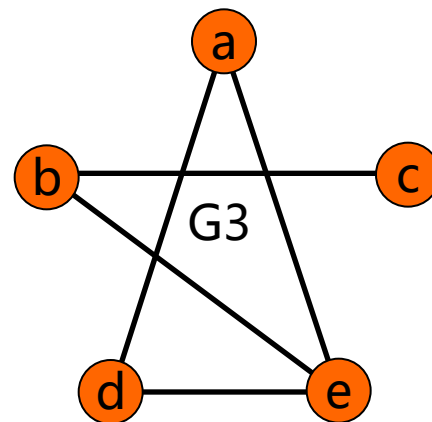
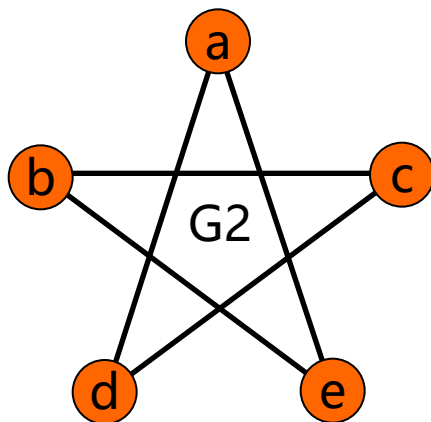
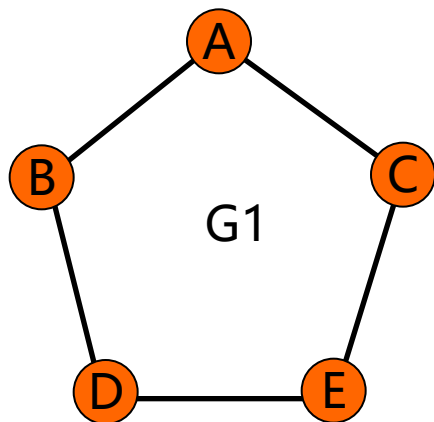


不同构

试试
总结泛化？

图的定义

- 同构的性质



- 若 $G_1 \cong G_2$, 则有

- 顶点数、边数相同: $n=5, m=5$
- 结点度非减序列相同: $G1(2,2,2,2,2), G3(1,2,2,2,3)$
- 存在同构的导出子图: G3存在三角形导出子图

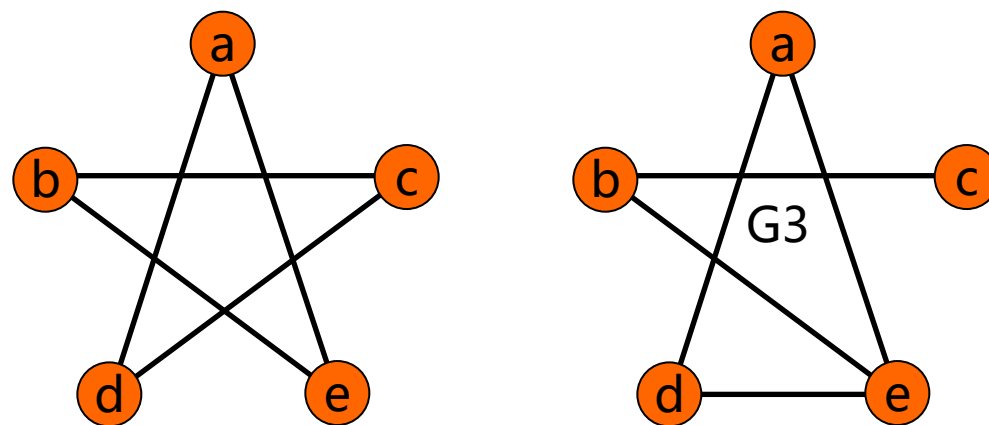


主要内容

- 图的定义、运算（续）
- 图的代数表示方法
 - 各种方法的特点（优缺点）
- 道路与回路
 - 概念和定义（带弦回路）
 - 判定：三个核心算法

图的代数表示

- 火眼金睛 v.s. 计算机处理
- 计算机擅长/不擅长什么?
- 图的代数表示方法
 - 邻接矩阵
 - 权矩阵
 - 关联矩阵
 - 边列表
 - 正向表
 - 邻接表

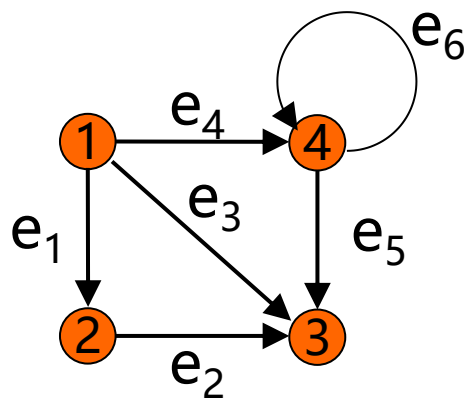


如何实现图的代数表示?

图的代数表示

- 发明：邻接矩阵（点&点）

$$A = [a_{ij}]_{n \times n} \quad a_{ij} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & \text{other} \end{cases}$$



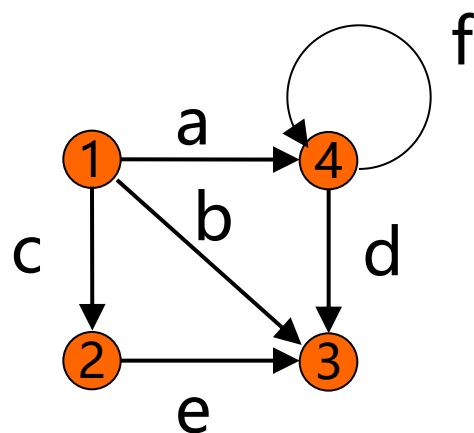
$$\begin{bmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 1 & 1 & 1 \\ v_2 & 0 & 0 & 1 & 0 \\ v_3 & 0 & 0 & 0 & 0 \\ v_4 & 0 & 0 & 1 & 1 \end{bmatrix}$$

- 自己的发明有什么特点？
 - 对于有向图的邻接矩阵中， v_i 的正度和 v_i 的负度？
 - 第*i*行的1的个数表示 v_i 的正度，第*i*列的1的个数表示 v_i 的负度
 - 邻接矩阵可表示自环，但是不能表示重边

图的代数表示(2)

- 权矩阵
 - 赋权图用权矩阵表示

表示重边?
两条带宽为3的边
两条时延为3的边



$$\begin{bmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & c & b & a \\ v_2 & 0 & 0 & e & 0 \\ v_3 & 0 & 0 & 0 & 0 \\ v_4 & 0 & 0 & d & f \end{bmatrix}$$

除了点&点外,
还有什么?
继续发明.....

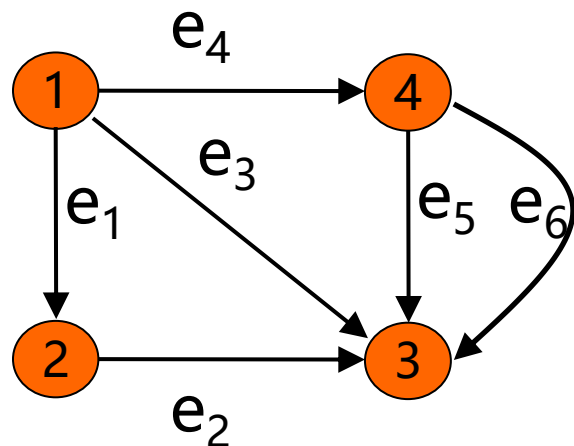
$$A = [a_{ij}]_{n \times n} \quad a_{ij} = \begin{cases} w_{ij} & (v_i, v_j) \in E \\ 0 & \text{other} \end{cases}$$

图的代数表示(3)

- 发明：关联矩阵 (n点&m边)

– 有向图

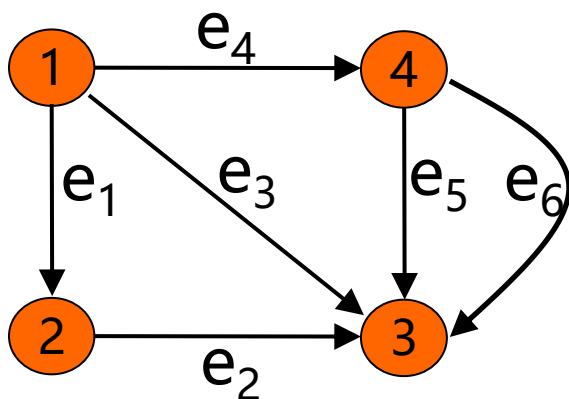
$$B = [b_{ij}]_{n \times m} \quad b_{ij} = \begin{cases} 1 & e_j = (v_i, v_k) \in E \\ -1 & e_j = (v_k, v_i) \in E \\ 0 & \text{其它} \end{cases}$$



$$\begin{bmatrix} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ v_1 & 1 & 0 & 1 & 1 & 0 & 0 \\ v_2 & -1 & 1 & 0 & 0 & 0 & 0 \\ v_3 & 0 & -1 & -1 & 0 & -1 & -1 \\ v_4 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix}$$

图的代数表示(4)

• 关联矩阵



$$\begin{bmatrix}
 & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\
 v_1 & 1 & 0 & 1 & 1 & 0 & 0 \\
 v_2 & -1 & 1 & 0 & 0 & 0 & 0 \\
 v_3 & 0 & -1 & -1 & 0 & -1 & -1 \\
 v_4 & 0 & 0 & 0 & -1 & 1 & 1
 \end{bmatrix}$$

– 关联图性质 (有向图)

- 每列只有一个1和一个-1
- 每行中1的个数为相应结点的正度, -1个数为负度
- 能表示重边, 不能表示自环

图的代数表示(5)

- 关联矩阵

- 有向图

$$B = [b_{ij}]_{n \times m} \quad b_{ij} = \begin{cases} 1 & e_j = (v_i, v_k) \in E \\ -1 & e_j = (v_k, v_i) \in E \\ 0 & \text{其它} \end{cases}$$

- 无向图

$$B = [b_{ij}]_{n \times m} \quad b_{ij} = \begin{cases} 1 & e_j \text{与} v_i \text{关联} \\ 0 & \text{其它} \end{cases}$$

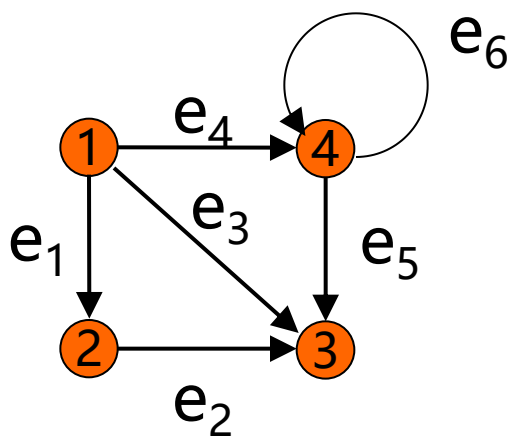
图的代数表示(6)

邻接矩阵

$$\begin{bmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 1 & 1 & 1 \\ v_2 & 0 & 0 & 1 & 0 \\ v_3 & 0 & 0 & 0 & 0 \\ v_4 & 0 & 0 & 1 & 1 \end{bmatrix}$$

关联矩阵

$$\begin{bmatrix} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ v_1 & 1 & 0 & 1 & 1 & 0 & 0 \\ v_2 & -1 & 1 & 0 & 0 & 0 & 0 \\ v_3 & 0 & -1 & -1 & 0 & -1 & -1 \\ v_4 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix}$$



- 基本表示的唯一性
 - 邻接矩阵与关联矩阵表示图是唯一的
- 能否有其他的表示方法?
 - 点和点 or 点和边
 - 边和边?

图的代数表示(7)

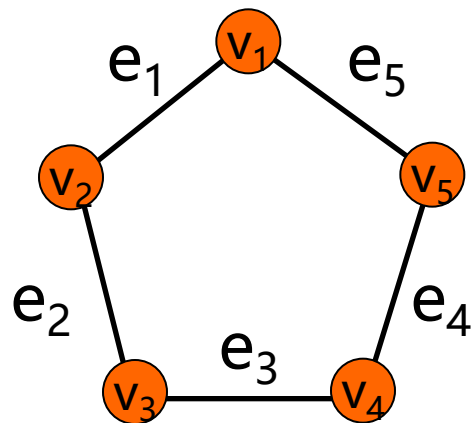
邻接矩阵

$$\begin{bmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 1 & 1 & 1 \\ v_2 & 0 & 0 & 1 & 0 \\ v_3 & 0 & 0 & 0 & 0 \\ v_4 & 0 & 0 & 1 & 1 \end{bmatrix}$$

关联矩阵

$$\begin{bmatrix} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ v_1 & 1 & 0 & 1 & 1 & 0 & 0 \\ v_2 & -1 & 1 & 0 & 0 & 0 & 0 \\ v_3 & 0 & -1 & -1 & 0 & -1 & -1 \\ v_4 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix}$$

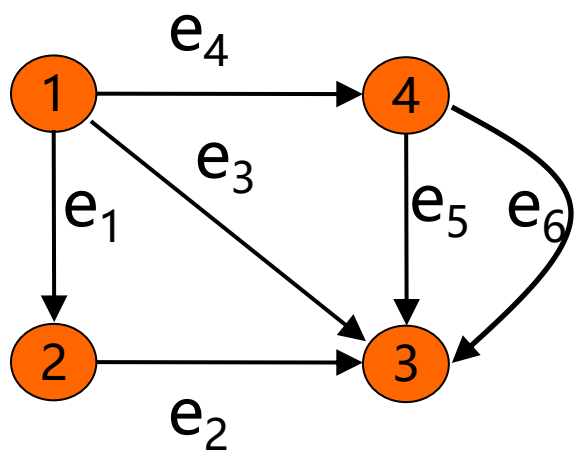
- 基本矩阵表示存在什么问题？
 - 不能表示重边或自环
 - 在计算机上存储邻接矩阵与关联矩阵时，将占据较大的存储空间并可能增加计算复杂度（稀疏矩阵）
- 因此引入边列表、正向表、逆向表、邻接表等



图的代数表示(8)

• 边列表

- 关联矩阵：透过现象看本质，连问三个为什么
- 对列进行压缩？



信息量的核心是非零元的位置

	e_1	e_2	e_3	e_4	e_5	e_6
v_1	1	0	1	1	0	0
v_2	-1	1	0	0	0	0
v_3	0	-1	-1	0	-1	-1
v_4	0	0	0	-1	1	1

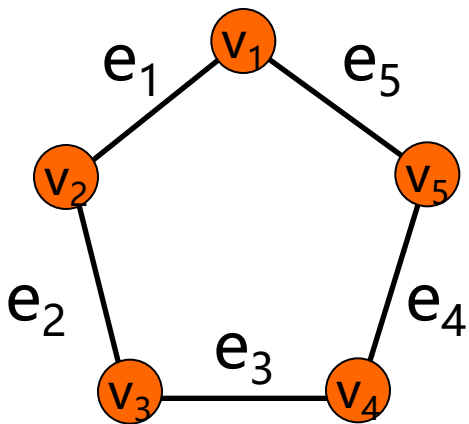
$$A : \begin{pmatrix} 1 & 2 & 1 & 1 & 4 & 4 \end{pmatrix}$$

$$B : \begin{pmatrix} 2 & 3 & 3 & 4 & 3 & 3 \end{pmatrix}$$

图的代数表示(9)

- 边列表

- 对关联矩阵的列进行压缩
- 边列表由两个 m 维向量 A 和 B 组成
- 当对 G 的结点与边进行编号后
- 对第 k 条边 $e_k=(v_i, v_j)$, 则 $A(k)=i$, $B(k)=j$, 即 $A(k)$ 存放第 k 条边始点的编号, $B(k)$ 存放其终点标号



–对赋权图怎么办?

用 m 维向量 Z 存放权, $Z(k)=w_k$ 。

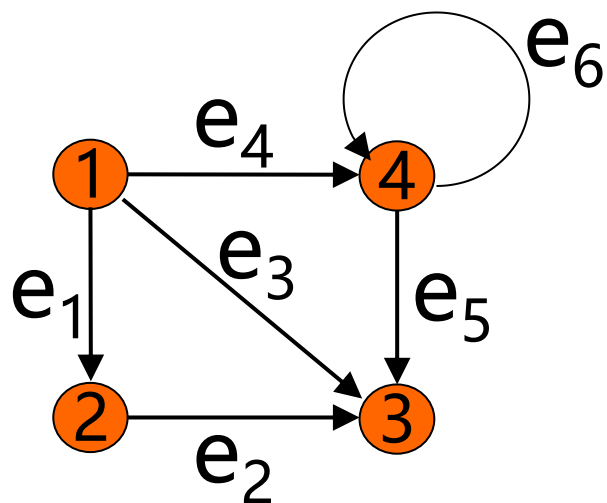
$A: (1\ 2\ 3\ 4\ 5)$

$B: (2\ 3\ 4\ 5\ 1)$

图的代数表示(10)

• 正向表

- 如何优化邻接矩阵?
- 对行进行压缩?
- 本质: 后继从哪开始?



直接后继节点?
排列一起?

2	3	4	3	3	4
---	---	---	---	---	---

	v_1	v_2	v_3	v_4
v_1	0	1	1	1
v_2	0	0	1	0
v_3	0	0	0	0
v_4	0	0	1	1

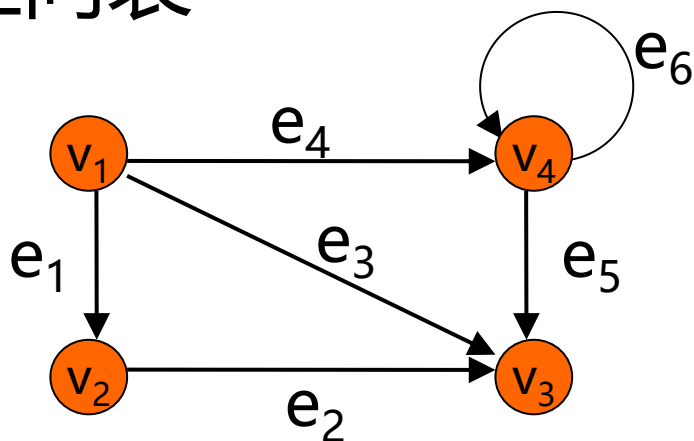
图的代数表示(11)

- 正向表

- 对邻接矩阵的行压缩
- 正向表将每个节点的直接后继集中在一起存放，有向图的正向表由一个 $(n+1)$ 维向量A，一个 m 维向量B组成
- 当对G的结点与边进行编号后， $A(i)$ 表示结点 v_i 的第一个后继在B中的地址，B中存放这些后继结点的编号， $A(n+1)=m+1$

图的代数表示(12)

• 正向表



是 v_4 吗?

A:
位置指针

v_1	v_2	v_3	v_4	v_{end}
1	4	5	5	7

B:
后继节点

2	3	4	3	3	4
---	---	---	---	---	---

	v_1	v_2	v_3	v_4
v_1	0	1	1	1
v_2	0	0	1	0
v_3	0	0	0	0
v_4	0	0	1	1

正向表, 存在如下关系:

$$d^+(v_i) = A(i+1) - A(i)$$

$$A(i) = \sum_{j=1}^{i-1} d^+(v_j) + 1$$

从 $B(A(i))$ 到 $B(A(i+1)-1)$ 的任一个值, 都是 v_i 的直接后继。

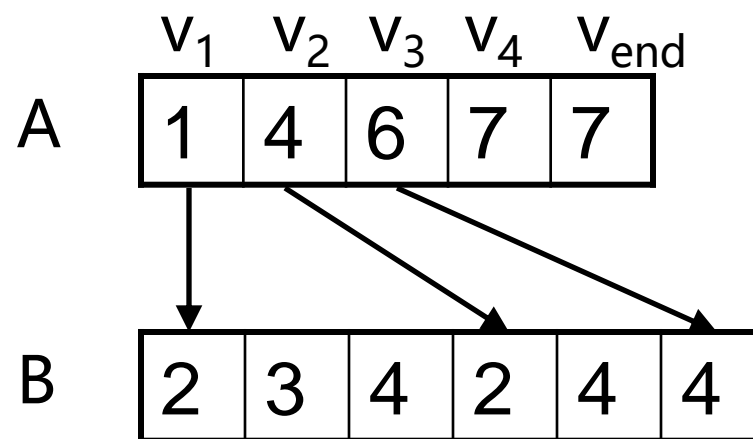
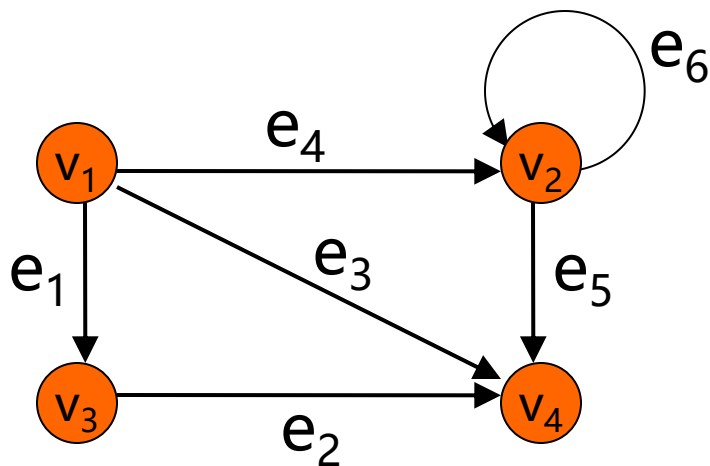
对赋权图:

用 m 维向量 Z 存放权 $Z(k) = w_k$

图的代数表示(13)

• 无向图的正向表

- 对于无向图，由于边没有方向性，所以B中存放的是相应邻结点的编号，因此B与Z都要扩充为 $2m$ 维的向量。

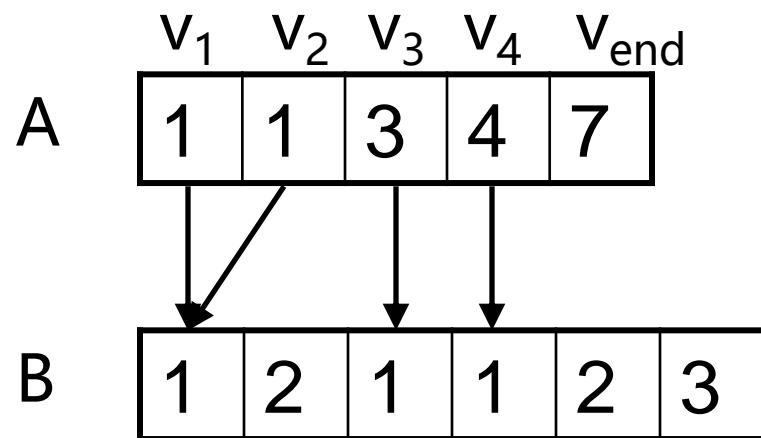
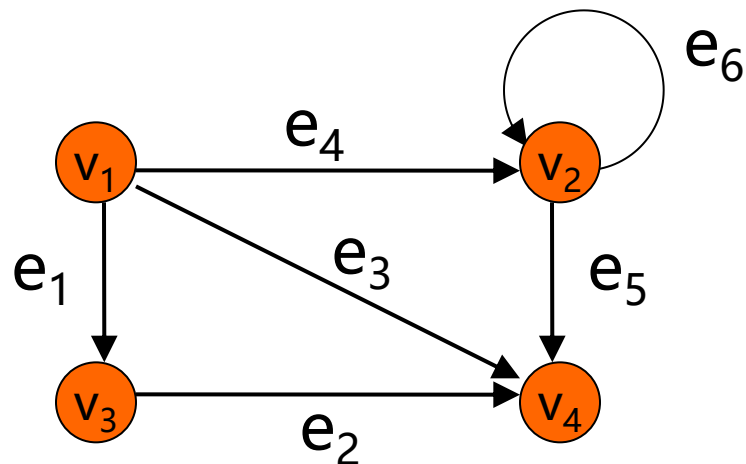


有了正向表（出度/后继），还应该.....

图的代数表示(14)

• 逆向表

– 将每个结点的直接前趋集中在一起存放。



优缺点？对于图的动态变化缺少灵活性：去掉 e_4 ？
 如何提高灵活性以适应图的动态变化？加减边？

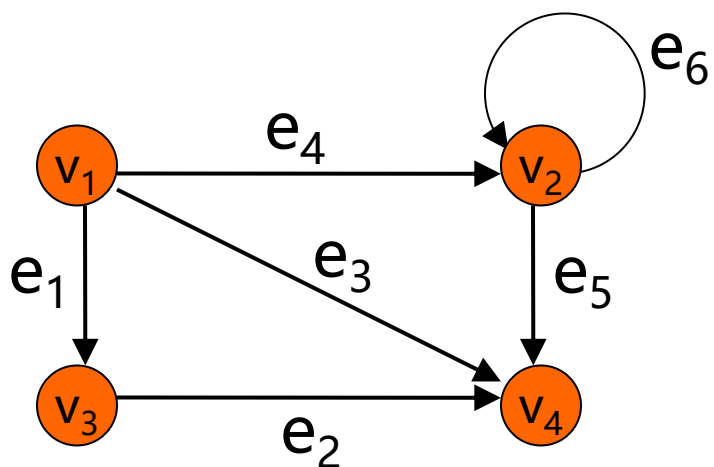
图的代数表示(15)

- 邻接表

- 采用单链表结构表示一个图
- 对每个结点 v_i 用一个表结点表示
- 表结点由三个域a、b、c组成
 - 邻结点域a中存放邻结点的编号
 - 数据域b中存放相应边的数值（权）
 - 链域c存放下一个表结点的地址指针

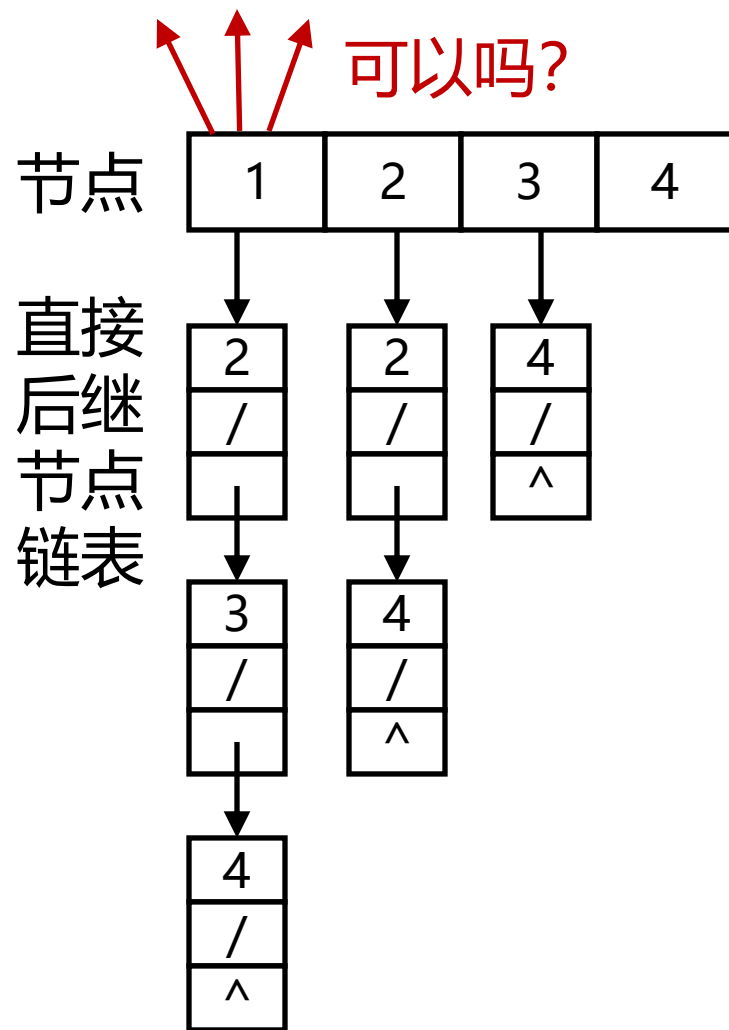
图的代数表示(16)

邻接表举例



a	• 邻结点域a中存放邻结点的编号
b	• 数据域b中存放相应边的数值 (权)
c	• 链域c存放下一个表结点的地址指针

如何进一步提高灵活性?



图的代数表示(17)

- 图的代数表示方法

- 邻接矩阵

- 关联矩阵

- 边列表

- 正向表

- 邻接表

简单灵活

避免稀疏矩阵

折衷
空间v.s.灵活

- 思考

- 各自特点：重边、自环、空间、处理方法、相互转换

- 小试牛刀：干掉第一个实验

100101010101000010011111010100011010101011010110101010101010100

图论与代数结构

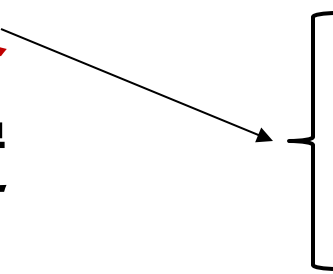
道路与回路基础

崔 勇

清华大学计算机系
网络技术研究所

清华大学计算机系网络技术研究所

第二章 道路与回路

- 道路与回路的定义
 - 道路与回路的判定
 - 欧拉道路与回路
 - 哈密顿道路与回路
 - 旅行商问题与分支定界法
 - 最短路径
 - 关键路径
 - 中国邮路
- 
- 道路与回路的定义
 - 割点、割边和块
 - 顶点与边的连通度

道路与回路的定义

- 有向道路的定义

发明一下?

- 有向图G的一条有向道路P是一个边序列 $(e_{i_1}, e_{i_2}, \dots, e_{i_q})$ 满足

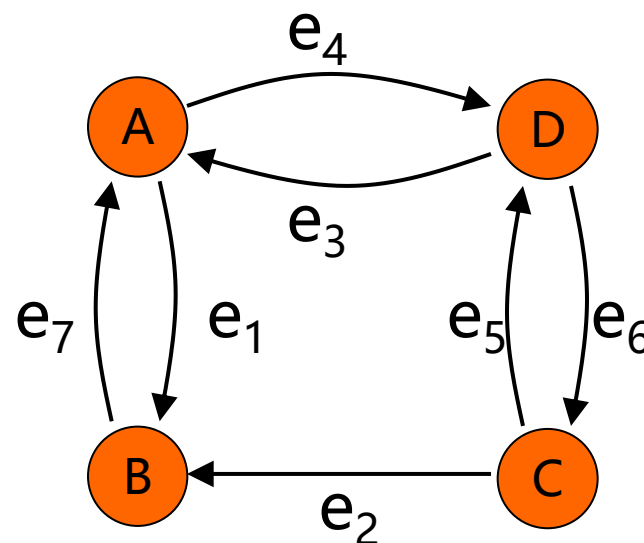
$$e_{i_k} = (v_{i_{k-1}}, v_{i_k}), k = 1, 2, \dots, q$$

利用已有定义
定义要严格

- 有向回路

- 若在图G的道路P中, $v_{i_0} = v_{i_q}$
- 则称P是G的一条有向回路。

不断细致如何细分?



寻找
特殊情况

道路与回路的定义(2)

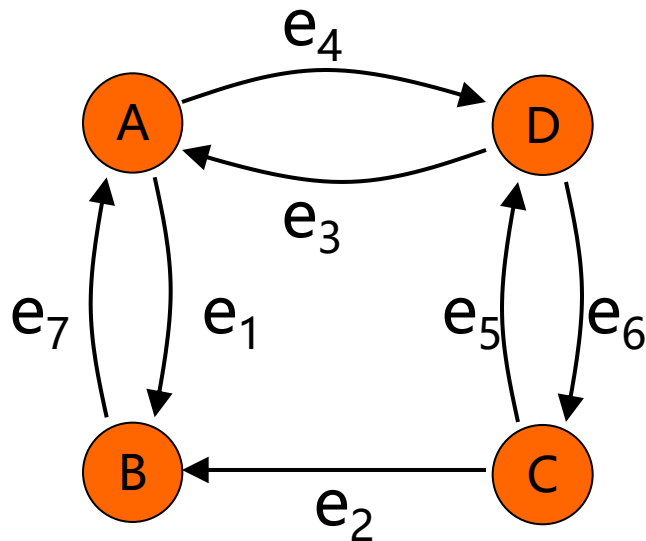
- 简单有向道路和回路

- P中的边没有重复出现，则称P为简单有向道路或回路

点可否重复？

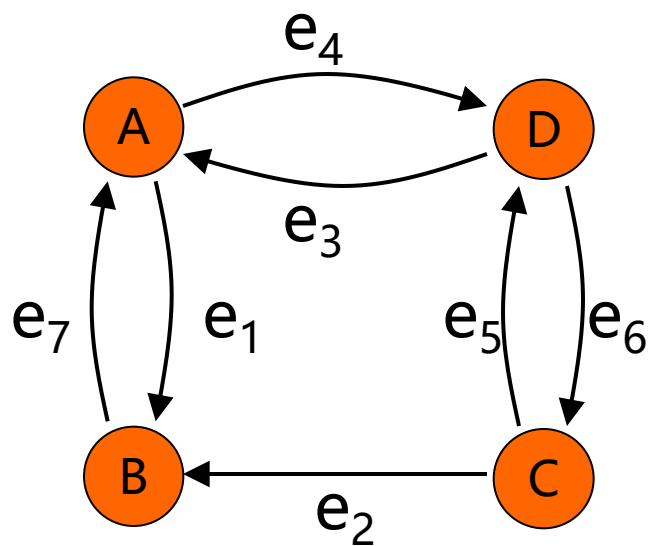
- 初级有向道路和回路

- P中的边和结点均不重复出现，
则称P为初级有向道路或回路，
简称为路或回路。



道路与回路的定义(3)

- 有向道路与回路



$(e_4, e_3, e_4, e_6, e_5, e_3, e_1)$ 有向道路

$(e_4, e_3, e_4, e_6, e_5, e_3)$ 有向回路

$(e_4, e_6, e_5, e_3, e_1)$ 简单道路

(e_4, e_6, e_5, e_3) 简单回路

(e_4, e_6, e_2) 初级道路

(e_4, e_6, e_2, e_7) 初级回路

道路与回路的定义(4)

- 无向图的道路与回路
 - 道路（链）与回路（圈）
 - 简单道路与回路（边不重复）
 - 初级道路与回路（顶点不重复）
- 与有向图的道路、回路定义类似，其区别只是无向图中的边没有方向。

道路与回路的定义(5)

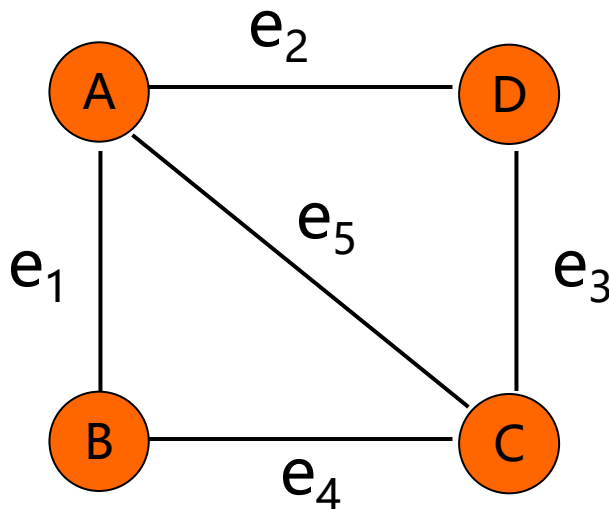
- 弦

- 设 C 为简单图 G 中含结点数大于3的一个初级回路，若结点 v_i 和 v_j 在 C 中不相邻，则称 (v_i, v_j) 是 C 的一条弦。

$$C = (e_1, e_2, e_3, e_4)$$

e_5 是 C 的一条弦

给出牛定义后呢？



存在性：三人行必有吾师！

道路与回路的定义(6)

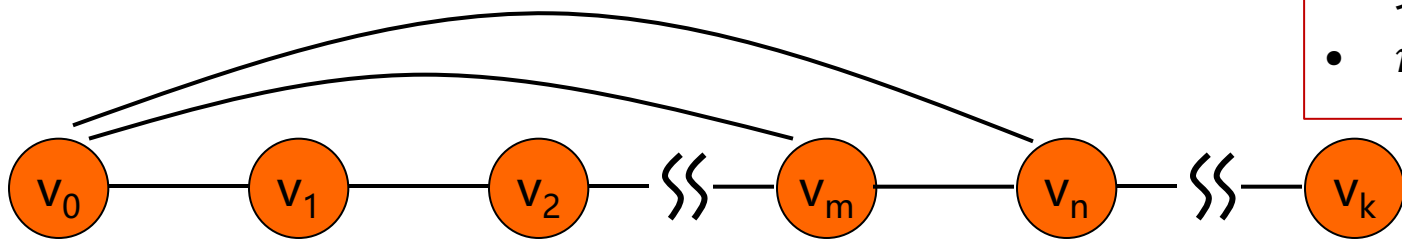
• 例

– 若 G 中每一点的度大于等于3，则 G 中必含带弦的回路。

– 证明（构造法）：

- 假设 G 中的一条极长的初级道路 P 为 $P = (e_1, e_2, \dots, e_k)$ $e_i = (v_{i-1}, v_i)$
- 由于 P 是极长道路， v_0 的所有邻结点均在此道路上
- v_0 的度不小于3，所以除了 v_1 以外， v_0 至少与 P 上的另外两个结点相连
- 设其为 $v_m, v_n, n > m$ ，则 $(e_1, e_2, \dots, e_n) + (v_n, v_0)$ 是一个初级回路，边 (v_0, v_m) 是其一条弦。

- P50.4简单图 G 中有 $n \geq 4$ 且 $m \geq 2n - 3$ ，则 G 含带弦回路
- 数学归纳法（假设）+ 构造
- $n + 1$ 与 n 关系（存在 $d \leq 2$ ？）



道路与回路的定义(7)

- 例：二分图

- 设 $G=(V,E)$ 是无向图，如果 $V(G)$ 可以划分为子集 X 和 Y ，使得对所有的 $e=(u,v)$, u 和 v 分属于 X 或 Y ，则称 G 为二分图。

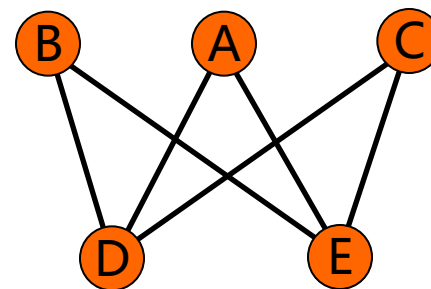
- 如果二分图中有回路，回路边数有什么特点？

- 答案：二分图回路的边数为偶数

- 证明：

- 设 C 是二分图 G 的回路

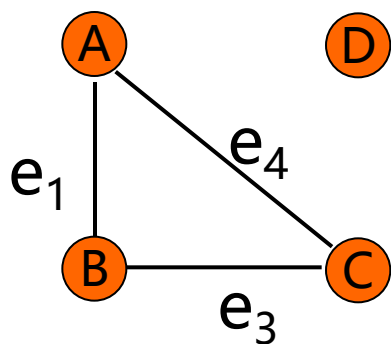
- 假设 C 的起点是 $v_0 \in X$ ，根据二分图的性质，回路从 v_0 出发后，经过奇数条边到达 Y ，经过偶数条边到达 X ，因此需要偶数条边才能回到 v_0 。



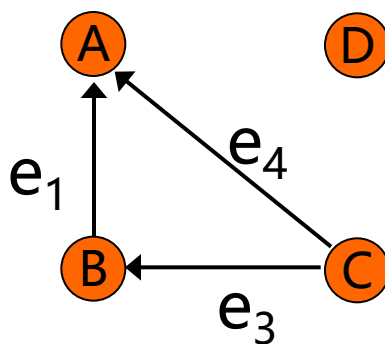
道路与回路的定义(8)

- 连通图、非连通图

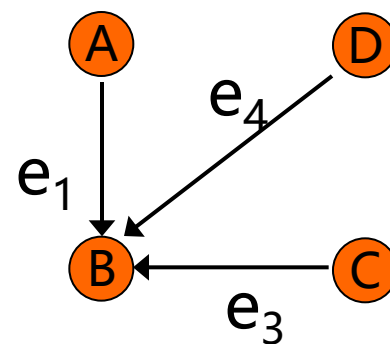
- 无向图 G 的任意两个结点之间都存在道路，就称 G 为连通图，否则称 G 为非连通图
- 对于有向图，若不考虑其边的方向，即视之为无向图，若它是连通的，则称 G 是连通图。



非连通图



非连通图



连通图

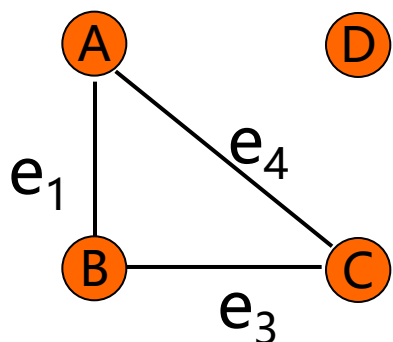
牛定义后?
存在性?

什么图必
连通?

道路与回路的定义(9)

- 极大连通子图

- 若连通子图 H 不是 G 的任何连通子图的真子图，称 H 是 G 的极大连通子图，或连通支



有几个连通支？

有两个连通支，结点集分别是 $\{A, B, C\}$, $\{D\}$

存在性：什么样的图必联通？

道路与回路的定义(10)

- 例

- 若 G 是简单图, 当 $m > \frac{(n-1)(n-2)}{2}$ 时, G 是连通图。
- 证 (反证法):
- 假定 G 非连通, 则至少存在2个连通支,
不妨设不相连子图 $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$.
- 其中 $|V_1| = n_1$, $|V_2| = n_2$, $|E_1| = m_1$, $|E_2| = m_2$
故有 $n_1 + n_2 = n$, $m_1 + m_2 = m$.
因为 G 是简单图, 所以 G_1 , G_2 也都是简单图

道路与回路的定义(11)

- 证 (续) :

- 反证法, 假定G非连通, G_1 、 G_2 都是简单图

- 有 $m_1 \leq \frac{n_1(n_1-1)}{2}$

$$m_2 \leq \frac{n_2(n_2-1)}{2}$$

$$\therefore m \leq \frac{n_1(n_1-1)}{2} + \frac{n_2(n_2-1)}{2}$$

能发明出来吗?

$$\because n_1, n_2 \leq n-1$$

$$\therefore m \leq \frac{(n-1)(n_1-1+n_2-1)}{2} = \frac{(n-1)(n-2)}{2}$$

- 与已知条件 $m > \frac{(n-1)(n-2)}{2}$ 矛盾, 因此G连通

割点, 割边和块

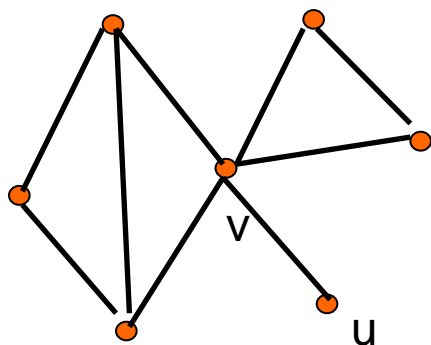
- 割点
 - 设 v 是 G 的一个顶点, 如果 $G-v$ 的连通分支数比 G 多, 称 v 是 G 的一个割点
- 割边
 - 设 e 是 G 的一条边, 若 $G' = G - e$ 比 G 的连通支数增加, 则称 e 是 G 的一条割边
- 块
 - 图 G 没有割点的极大连通子图称为块

割点, 割边和块

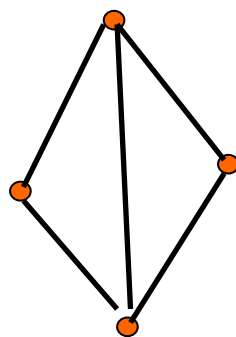
- 例

- 图(a)中 v 是割点, 它有三个子块

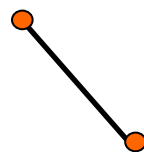
- U 是不是割点?



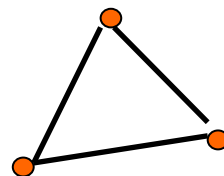
(a)



B_1



B_2



B_3

(b)

有割点必然有割边? NO!
有割边必然有割点?

第二章 道路与回路

- 道路与回路的定义
- 道路与回路的判定
- 欧拉道路与回路
- 哈密顿道路与回路
- 旅行商问题与分支定界法
- 最短路径
- 关键路径
- 中国邮路

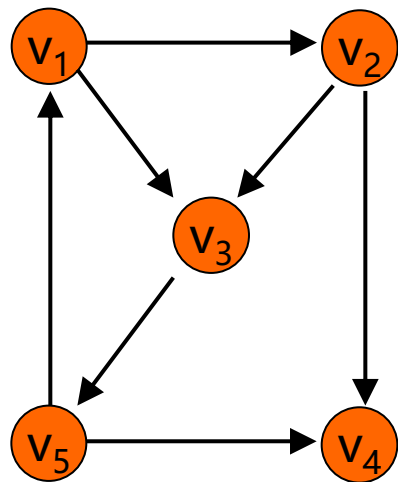
连通图的牛定义后

性质？存在性？
连通图的判定？

- 邻接矩阵法判定
- 广探法BFS
- 深探法DFS

道路与回路的判定

- 用邻接矩阵或搜索法判定两结点间是否存在通路



三步能到吗?

$v_1 \rightarrow v_4$ 的可达性

$v_1 - v_4$, 两步能到吗?

$$P = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

道路与回路的判定

- 邻接矩阵判定法:

- 设图G的邻接矩阵 $A = (a_{ij})$

- 若 $a_{ij} = 1$

- 表示 (v_i, v_j) 为G的一条边, 则 v_i, v_j 间有道路

- $A^2 = (a_{ij}^{(2)})$ $a_{ij}^{(2)} = a_{ik} \cdot a_{kj}$ 之和 ($k=1 \dots n$)

- 若 $a_{ij}^{(2)}$ 不为零

- 当且仅当存在 k , 使得 $a_{ik} = a_{kj} = 1$

- (v_i, v_k) 和 (v_k, v_j) 为G的边

- v_i, v_j 间有长度为2的道路

$$P = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

道路与回路的判定(2)

- 邻接矩阵判定法(续):
 - 若 $A^l = (a_{ij}^{(l)}) (l \leq n)$ 当中 $a_{ij}^{(l)}$ 不为零, 表示存在 v_i, v_j 间有长为 l 的道路。
 - n 步判断可达性的表示方法?
 - $P = (p_{ij}) = A + A^2 + A^3 + A^4 + \dots + A^n$
 - **P的含义**: 一般若 $p_{ij} = t$ 从 v_i 有 t 条道路到达 v_j , $p_{ij} = 0$, n 步内从 v_i 不能到达 v_j , 则在 G 中不存在从 v_i 到达 v_j 的路。

道路与回路的判定(3)

- 邻接矩阵判定法(续):

- $P=(p_{ij})=A+A^2+A^3+A^4+\dots+A^n$

- n 步是否足够?

- 基于抽屉原理的构造?

- 若从 v_i 经过 l 步 ($l \geq n$)能到达 v_j , 根据抽屉原理, 必在该路中有相同的 v_k , 即存在回路, 删掉这段回路, 仍存在从 v_i 到达 v_j 的路。

- 因此有 v_i, v_j 间有道路, 当且仅当 p_{ij} 不为0

道路与回路的判定(4)

- 若只关心 v_i 与 v_j 之间有无道路可用逻辑运算法
 - $a_{ij}^{(l)} = \bigvee_{k=1, n} (a_{ik}^{(l-1)} \wedge a_{kj})$
 - 图G的**道路矩阵**:
$$P = A \vee A^2 \vee A^3 \vee A^4 \vee \dots \vee A^n$$
- 更“简单”的算法
 - 计算道路矩阵 – Warshall算法

道路与回路的判定(5)

- Warshall算法

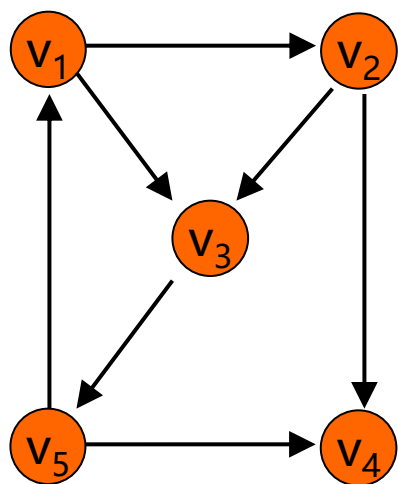
```
begin
  i = 1 to n
    j=1 to n
      k=1 to n
         $p_{jk} = p_{jk} \vee (p_{ji} \wedge p_{ik})$ 
      end
    end
  end
```

- 对内部循环变量j,k, 逐一更新 p_{jk} , 即 (v_j, v_k) 的可达性 (是否能找到经过 v_i 的路径)
- 遍历所有的 v_i , 不断更新P矩阵

道路与回路的判定(6)

- 例

- 使用Warshall算法计算下图的道路矩阵
- 依次循环*i,j,k*计算: $p_{jk} = p_{jk} \vee (p_{ji} \wedge p_{ik})$



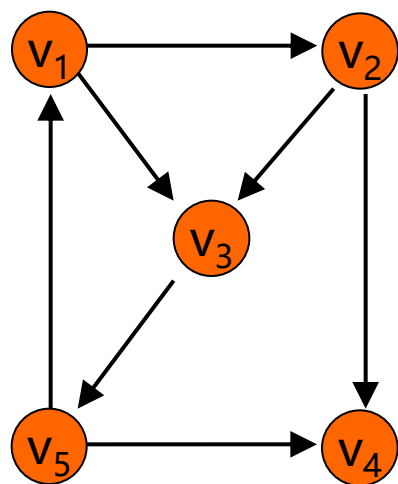
人工执行算法
(结合图)

$$P = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

道路与回路的判定(6)

• 例

- 使用Warshall算法计算下图的道路矩阵
- 依次循环*i,j,k*计算: $p_{jk} = p_{jk} \vee (p_{ji} \wedge p_{ik})$



人工执行算法
(结合图)

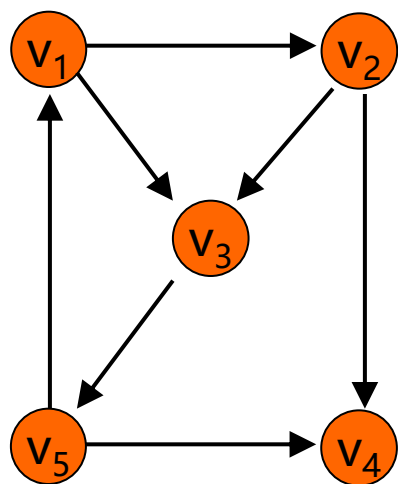
$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$i = 1 \quad \begin{matrix} j=5 \\ k=2,3 \end{matrix}$$

道路与回路的判定(6)

• 例

- 使用Warshall算法计算下图的道路矩阵
- 依次循环*i,j,k*计算: $p_{jk} = p_{jk} \vee (p_{ji} \wedge p_{ik})$



人工执行算法
(结合图)

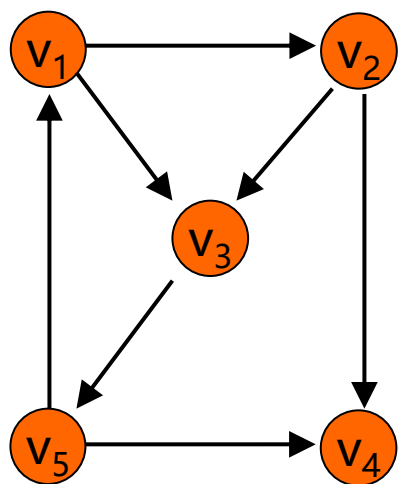
$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$i = 2 \quad \begin{matrix} j=1,5 \\ k=3,4 \end{matrix}$$

道路与回路的判定(6)

• 例

- 使用Warshall算法计算下图的道路矩阵
- 依次循环*i,j,k*计算: $p_{jk} = p_{jk} \vee (p_{ji} \wedge p_{ik})$



人工执行算法
(结合图)

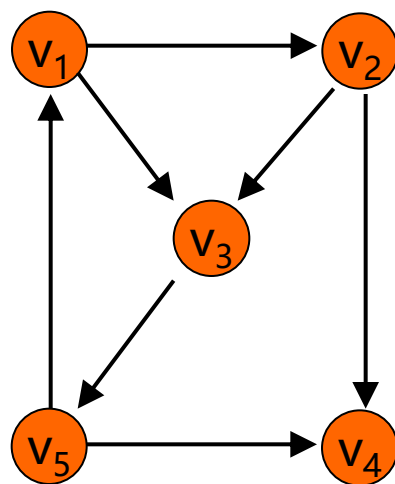
$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$i = 3 \quad \begin{matrix} j=1,2,5 \\ k=5 \end{matrix}$$

道路与回路的判定(6)

• 例

- 使用Warshall算法计算下图的道路矩阵
- 依次循环*i,j,k*计算: $p_{jk} = p_{jk} \vee (p_{ji} \wedge p_{ik})$



人工执行算法
(结合图)

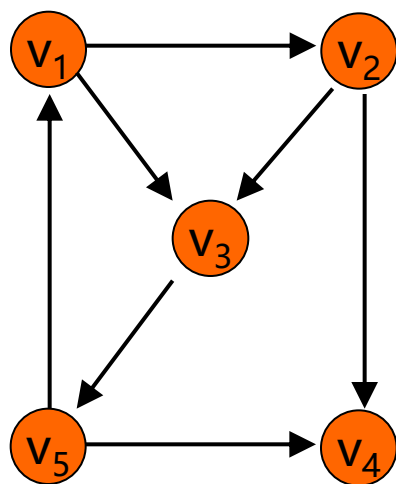
$$\begin{bmatrix}
 0 & 1 & 1 & 1 & 1 \\
 0 & 0 & 1 & 1 & 1 \\
 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 0 & 0 \\
 1 & 1 & 1 & 1 & 1
 \end{bmatrix}$$

$i = 4$ $j = 1, 2, 5$
 $k = /$

道路与回路的判定(6)

• 例

- 使用Warshall算法计算下图的道路矩阵
- 依次循环*i,j,k*计算: $p_{jk} = p_{jk} \vee (p_{ji} \wedge p_{ik})$



人工执行算法
(结合图)

$$\begin{bmatrix}
 1 & 1 & 1 & 1 & 1 \\
 1 & 1 & 1 & 1 & 1 \\
 1 & 1 & 1 & 1 & 1 \\
 0 & 0 & 0 & 0 & 0 \\
 1 & 1 & 1 & 1 & 1
 \end{bmatrix}$$

$$i = 5 \quad \begin{matrix} j=1,2,3,5 \\ k=1,2,3,4,5 \end{matrix}$$

补充-算法复杂度

- 时间复杂度

- 一般是指问题随规模的增长算法所需消耗的运算时间的增长趋势。
- 问题规模即要处理的数据增长时，基本操作要重复执行的次数必定也会增长。
- 我们关心这个执行次数以什么样的数量级增长
- 基本操作的执行次数是问题规模 n 的一个函数 $T(n)$

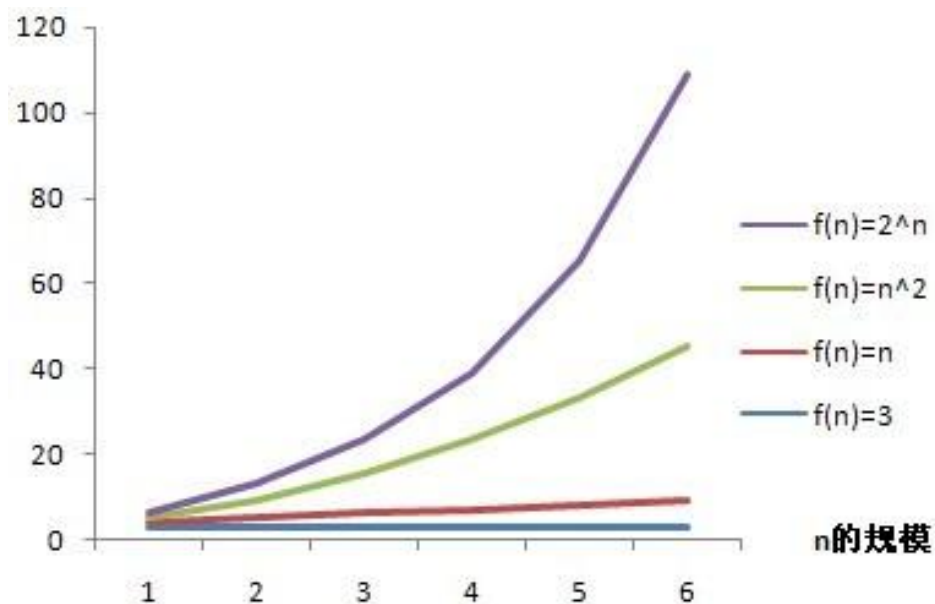
可惜我们很难得到 $T(n)$ ☹，怎么办呢？

补充-算法复杂度

- 时间复杂度
 - 同数量级函数
 - 考虑辅助函数 $f(n)$
 - 使得当 n 趋近于无穷大时, $T(n)/f(n)$ 的极限值为不等于零的常数, 则称 $f(n)$ 是 $T(n)$ 的同数量级函数
 - 时间复杂度
 - 若存在辅助函数 $f(n)$, 与 $T(n)$ 是同数量级函数, 记作 $T(n) = O(f(n))$, 称 $O(f(n))$ 为算法的渐进时间复杂度, 简称时间复杂度
 - 如 $f(n) = C$ 或 $\log n$ 或 n 或 n^k 或 k^n ($k > 1$)

补充-算法复杂度

- 不同复杂度的比较



举例说明算法复杂度的概念

10000, $N+500$

$10N$, N^2 , N^3

$N!$, N^N

N^a , b^N ,

某问题复杂度为 $N!$ 当 $N=100$ 时, 用世界上最快的超级计算机“神威·太湖之光”, 需要1s完成计算。
当问题规模增长10%, 计算用时多少?

约 $100^{10}s=10^{12}$ 年



补充-算法复杂度

道路矩阵: $P = A \vee A^2 \vee A^3 \vee A^4 \vee \dots A^n$

- 计算道路矩阵 – Warshall算法

```
begin
  i=1 to n
    j=1 to n
      k=1 to n
         $p_{jk} = p_{jk} \vee (p_{ji} \wedge p_{ik})$ 
      End
    End
  End
```

算法复杂度

$O(n^3)$

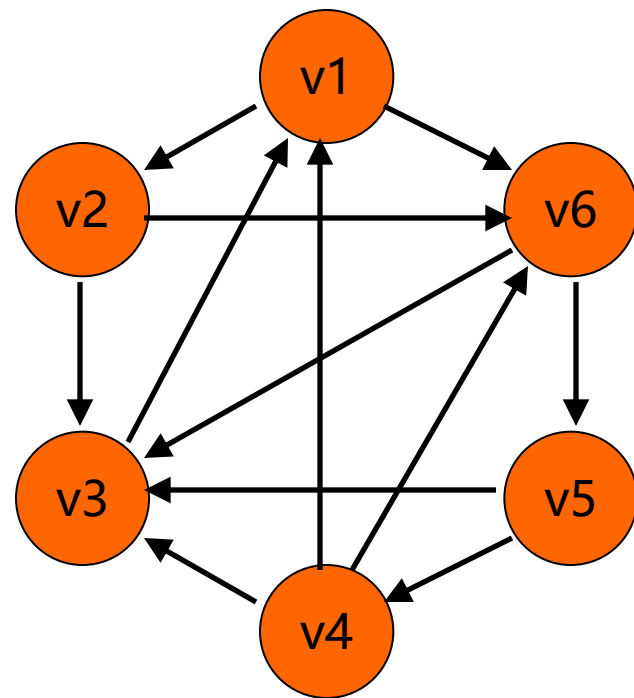
道路与回路的判定(7)

- 判断两个结点间有无道路的方法
 - 基于邻接矩阵的Warshall算法
 - 广探法(Breadth First Search)
 - 深探法(Depth First Search)

道路与回路的判定(8)

- 广探法(BFS)

- BFS(Breadth First Search), 是从 G 的任一结点 v_1 开始, 找它的直接后继集 $\Gamma^+(v_1)$, 记为 A_1
- 对 A_1 中的每一个结点分别找它们的直接后继集, 这些第二批后继集的并记为 A_2
- 依此类推, 直至达到目的结点。
- 可能存在的问题?
 - 回路避免

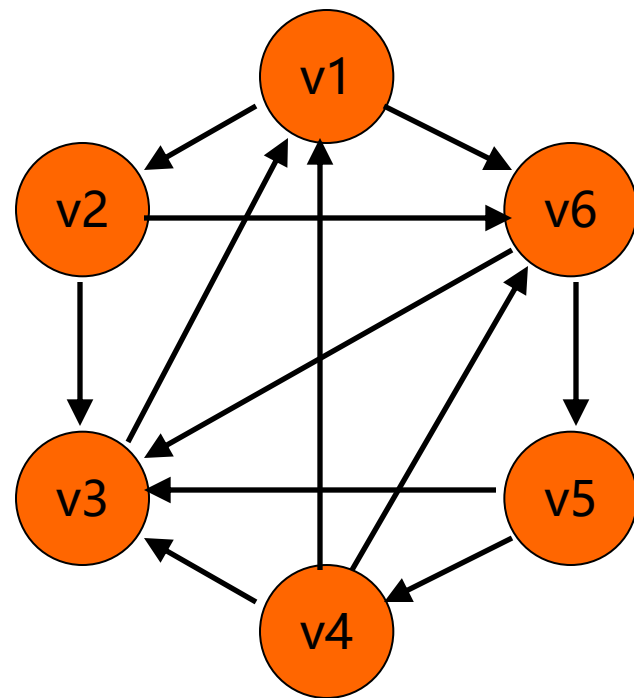


道路与回路的判定(9)

- 广探法 (BFS)

- 为避免结点的重复搜索可对结点进行标记

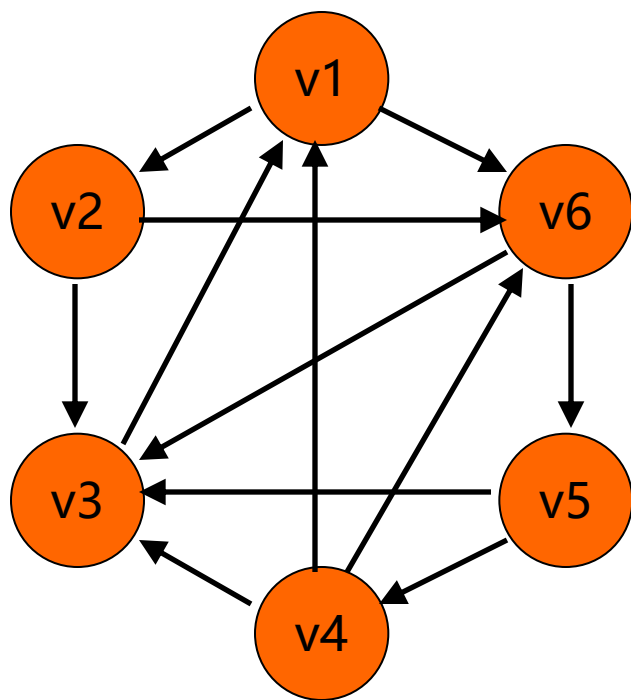
- 开始时所有结点标记为0
 - 搜索时若新搜到的结点标记为0, 则加入后继集, 同时将其标记改为1
 - 搜索时若新搜到的结点标记为1, 则忽略该点



道路与回路的判定(10)

• 例

– 用BFS找下图中 v_1 到 v_4 的一条道路。



访问节点

$$\Gamma^+(v_1) = \{v_2, v_6\}$$

$$\Gamma^+(v_2) = \{v_3, v_6\}$$

$$\Gamma^+(v_6) = \{v_3, v_5\}$$

$$\Gamma^+(v_3) = \{v_1\}$$

$$\Gamma^+(v_5) = \{v_3, v_4\}$$

后继集

$$A_1 = \{v_2, v_6\}$$

$$A_2 = \{v_3, v_5\}$$

$$A_3 = \{v_4\}$$

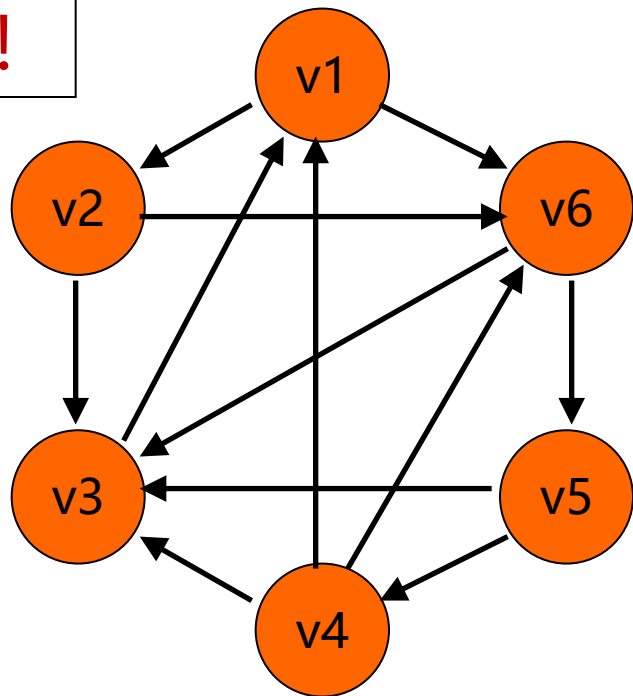
路径信息?

struct {flag, hop, prior} v[n]

图论与代数结构

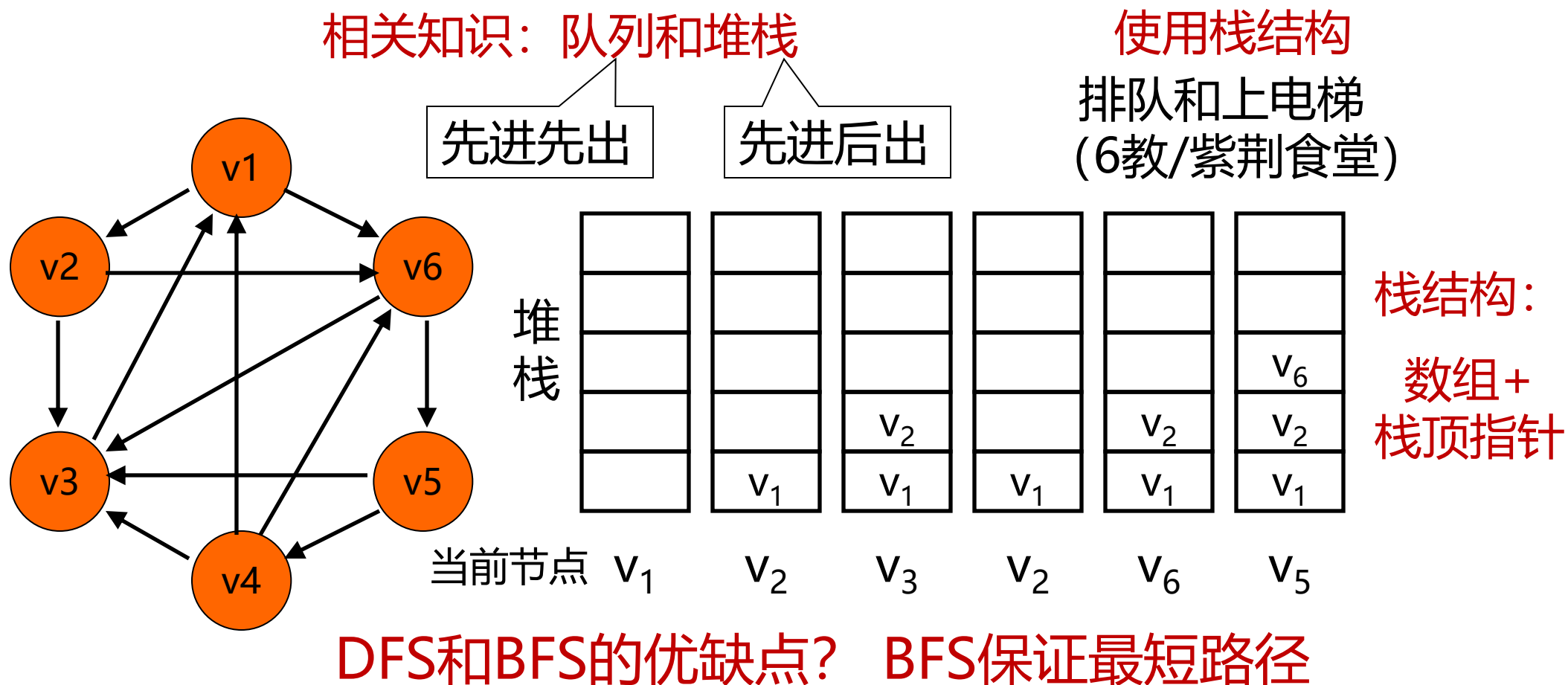
- 深探法(DFS)

- ## 回头时要能回得去！



道路与回路的判定(12)

- 例：使用DFS找出下图中 v_1 到 v_4 的道路



总结（道路和回路）

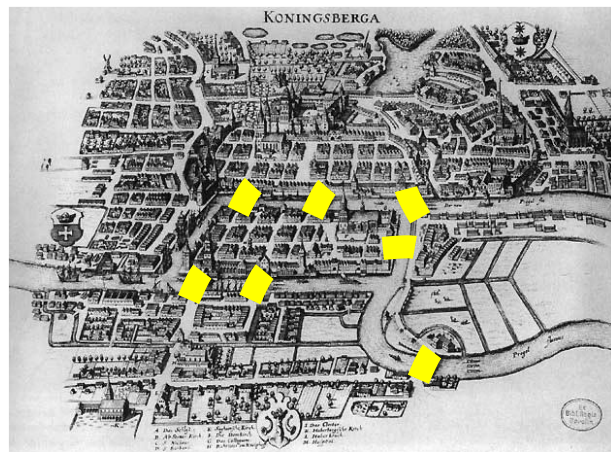
- 从一般性定义到特殊
 - 首尾相接：严格而巧妙的形式化定义
 - 简单道路、初级道路（边/点是否重复）
 - 带弦回路（极长初级道路）
 - 特殊：极大连通子图，边足够多必连通
 - 邻接矩阵判定道路：Warshall算法
 - 算法复杂度：100, $10N$, N^2 , N^3 , $N!$
 - 判定/寻找道路：广探法、深探法

下一讲提要

- 欧拉道路与回路
- 哈密顿道路与回路
- 旅行商问题与分支定界法



欧拉

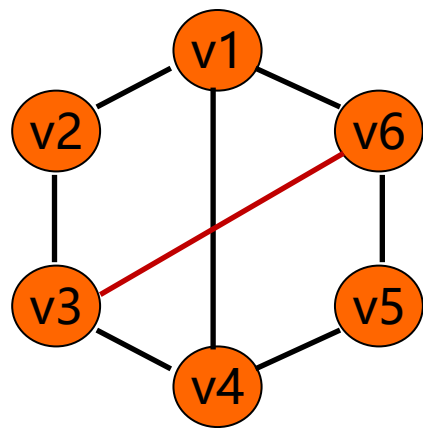


建模、算法、证明、发明

第二周作业

- 课程作业

- 习题一/P14 (图的运算) : 16题
 - 习题二/P50 (道路/回路) : 3, 4, 5题
 - 补充题: 采用传统矩阵乘法, 道路矩阵P的计算复杂度是多少?
 - P50.5: 设G是不存在三角形的简单图,
- 解析题干: 不存在三角形, 任2点度数 $\leq n$





第二周作业

- 引导实验
 - 不计入成绩，但推荐同学参考、尝试
- 第一次编程实验
 - **最短路**和欧拉回路二选一
 - DDL：第六周周日 23:59:59 前