本节内容

图的存储

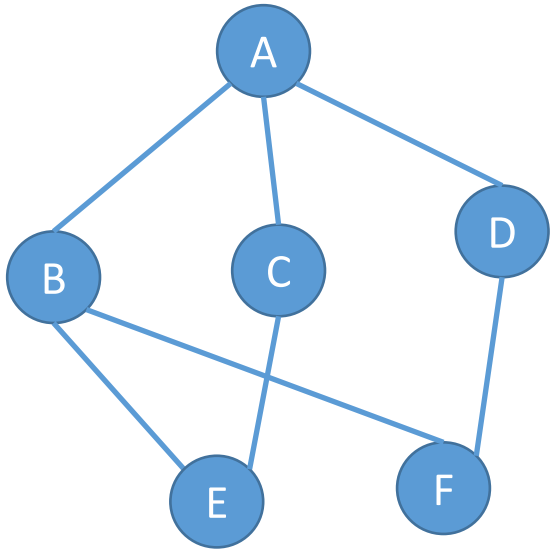
邻接矩阵法

知识总览

- 图的存储
  - 邻接矩阵
  - 邻接表
  - 十字链表
  - 邻接多重表

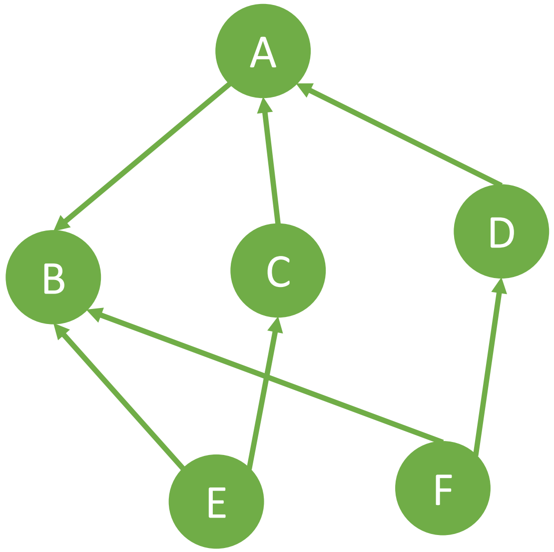
图的存储——邻接矩阵法

无向图



	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	1	0
D	1	0	0	0	0	1
E	0	1	1	0	0	0
F	0	1	0	1	0	0

有向图



	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	0	0	0	0
C	1	0	0	0	0	0
D	1	0	0	0	0	0
E	0	1	1	0	0	0
F	0	1	0	1	0	0

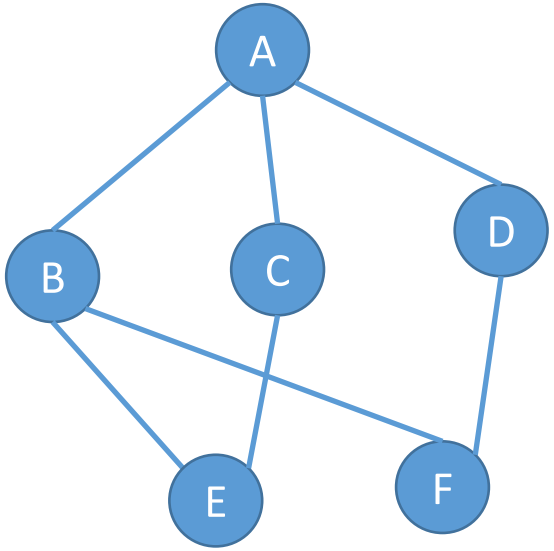
```
#define MaxVertexNum 100                          //顶点数目的最大值
typedef struct{
    char Vex[MaxVertexNum];                       //顶点表
    int Edge[MaxVertexNum][MaxVertexNum];         //邻接矩阵，边表
    int vexnum,arcnum;                            //图的当前顶点数和边数/弧数
} MGraph;
```

顶点中可以存更复杂的信息

可以用 bool型或枚举型变量表示边

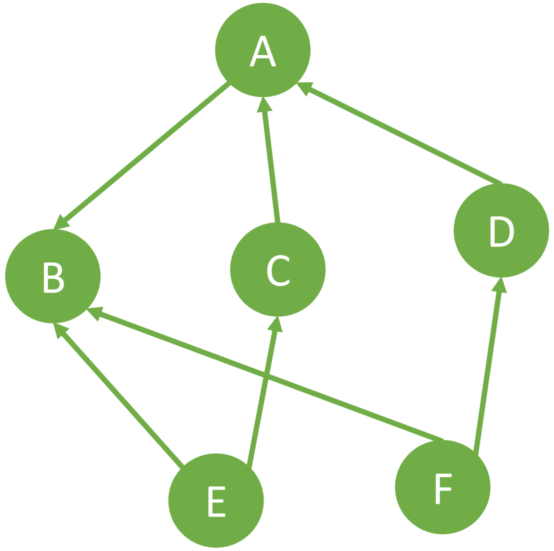
图的存储——邻接矩阵法

无向图



	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	1	0
D	1	0	0	0	0	1
E	0	1	1	0	0	0
F	0	1	0	1	0	0

有向图



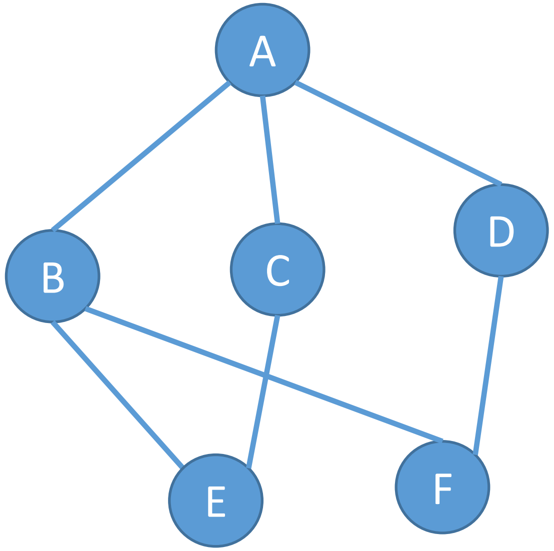
	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	0	0	0	0
C	1	0	0	0	0	0
D	1	0	0	0	0	0
E	0	1	1	0	0	0
F	0	1	0	1	0	0

结点数为n的图$G=(V,E)$的邻接矩阵$\boldsymbol{A}$是$n\times n$的。将G的顶点编号为$v_1, v_2,\ldots, v_n$，则

$$A[i][j]=\begin{cases}1, & 若(v_i,v_j)或\langle v_i,v_j\rangle是E(G)中的边\\0, & 若(v_i,v_j)或\langle v_i,v_j\rangle不是E(G)中的边\end{cases}$$

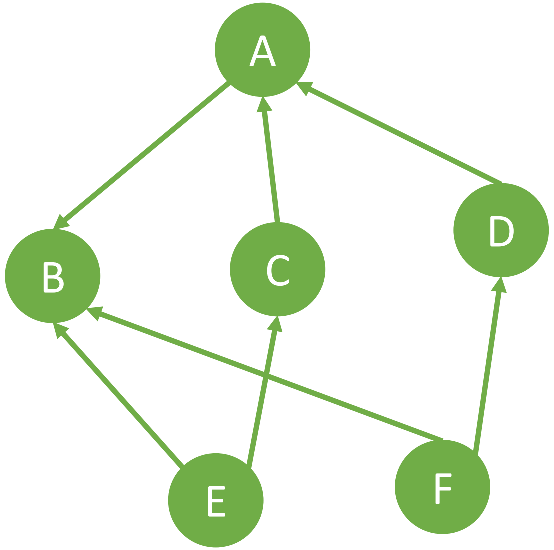
图的存储——邻接矩阵法

无向图



	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	1	0
D	1	0	0	0	0	1
E	0	1	1	0	0	0
F	0	1	0	1	0	0

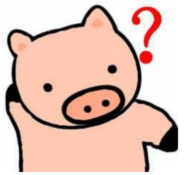
有向图



	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	0	0	0	0
C	1	0	0	0	0	0
D	1	0	0	0	0	0
E	0	1	1	0	0	0
F	0	1	0	1	0	0

第i个结点的度 = 第i行（或第i列）的非零元素个数

第i个结点的出度 = 第i行的非零元素个数
第i个结点的入度 = 第i列的非零元素个数
第i个结点的度 = 第i行、第i列的非零元素个数之和



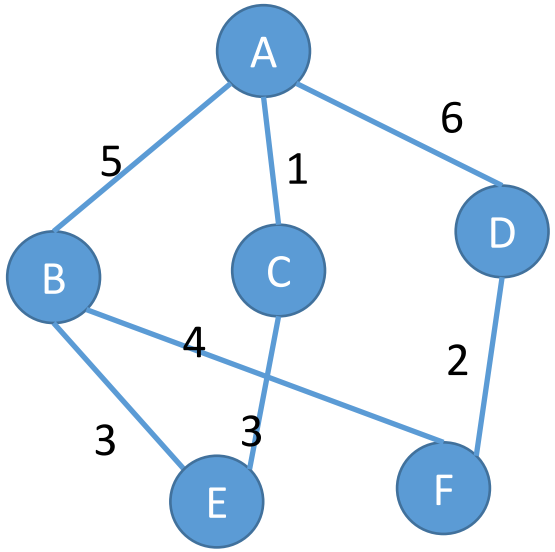
思考：如何求顶点的度、入度、出度？

如何找到与一个顶点相连的边/弧？

邻接矩阵法求顶点的度/出度/入度的时间复杂度为 O(|V|)

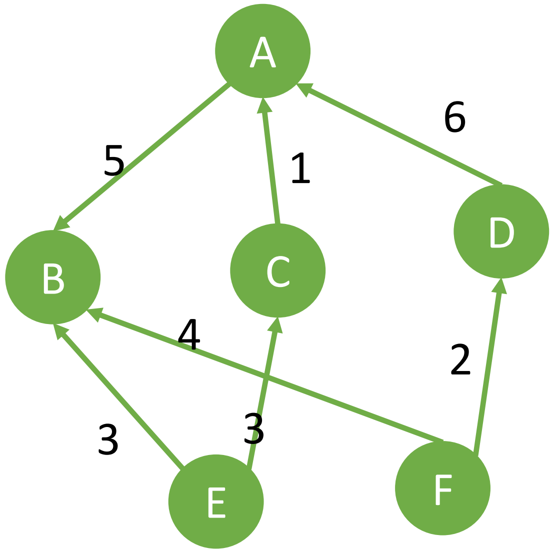
邻接矩阵法存储带权图（网）

无向网



	A	B	C	D	E	F
A	∞	5	1	6	∞	∞
B	5	∞	∞	∞	3	4
C	1	∞	∞	∞	3	∞
D	6	∞	∞	∞	∞	2
E	∞	3	3	∞	∞	∞
F	∞	4	∞	2	∞	∞

有向网



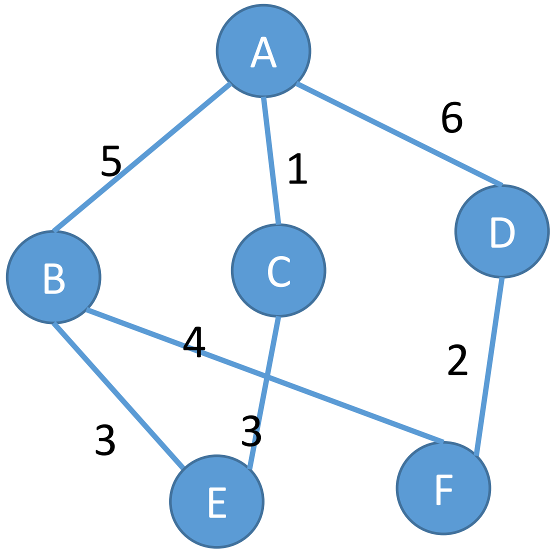
	A	B	C	D	E	F
A	∞	5	∞	∞	∞	∞
B	∞	∞	∞	∞	∞	∞
C	1	∞	∞	∞	∞	∞
D	6	∞	∞	∞	∞	∞
E	∞	3	3	∞	∞	∞
F	∞	4	∞	2	∞	∞

可用int的上限值表示“无穷”

```
#define MaxVertexNum 100                          //顶点数目的最大值
#define INFINITY 最大的int值                       //宏定义常量"无穷"
typedef char VertexType;                          //顶点的数据类型
typedef int EdgeType;                             //带权图中边上权值的数据类型
typedef struct{
    VertexType Vex[MaxVertexNum];                 //顶点
    EdgeType Edge[MaxVertexNum][MaxVertexNum];    //边的权
    int vexnum,arcnum;                            //图的当前顶点数和弧数
}MGraph;
```

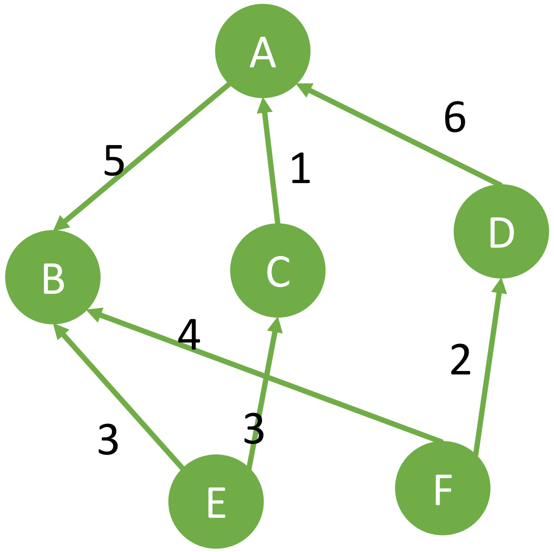
邻接矩阵法存储带权图（网）

无向网



	A	B	C	D	E	F
A	0	5	1	6	∞	∞
B	5	0	∞	∞	3	4
C	1	∞	0	∞	3	∞
D	6	∞	∞	0	∞	2
E	∞	3	3	∞	0	∞
F	∞	4	∞	2	∞	0

有向网



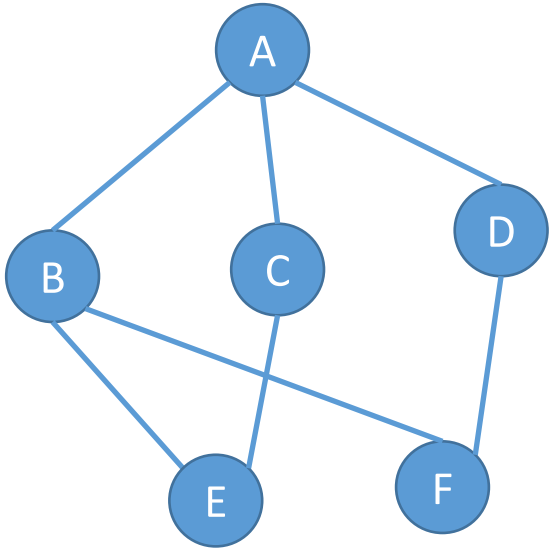
	A	B	C	D	E	F
A	0	5	∞	∞	∞	∞
B	∞	0	∞	∞	∞	∞
C	1	∞	0	∞	∞	∞
D	6	∞	∞	0	∞	∞
E	∞	3	3	∞	0	∞
F	∞	4	∞	2	∞	0

可用int的上限值表示“无穷”

```
#define MaxVertexNum 100                            //顶点数目的最大值
#define INFINITY 最大的int值                         //宏定义常量"无穷"
typedef char VertexType;                            //顶点的数据类型
typedef int EdgeType;                               //带权图中边上权值的数据类型
typedef struct{
    VertexType Vex[MaxVertexNum];                   //顶点
    EdgeType Edge[MaxVertexNum][MaxVertexNum];      //边的权
    int vexnum,arcnum;                              //图的当前顶点数和弧数
}MGraph;
```

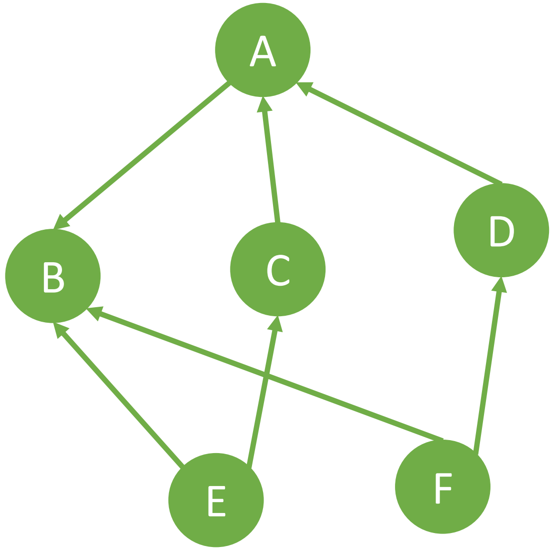
邻接矩阵法的性能分析

无向图



	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	1	0
D	1	0	0	0	0	1
E	0	1	1	0	0	0
F	0	1	0	1	0	0

有向图



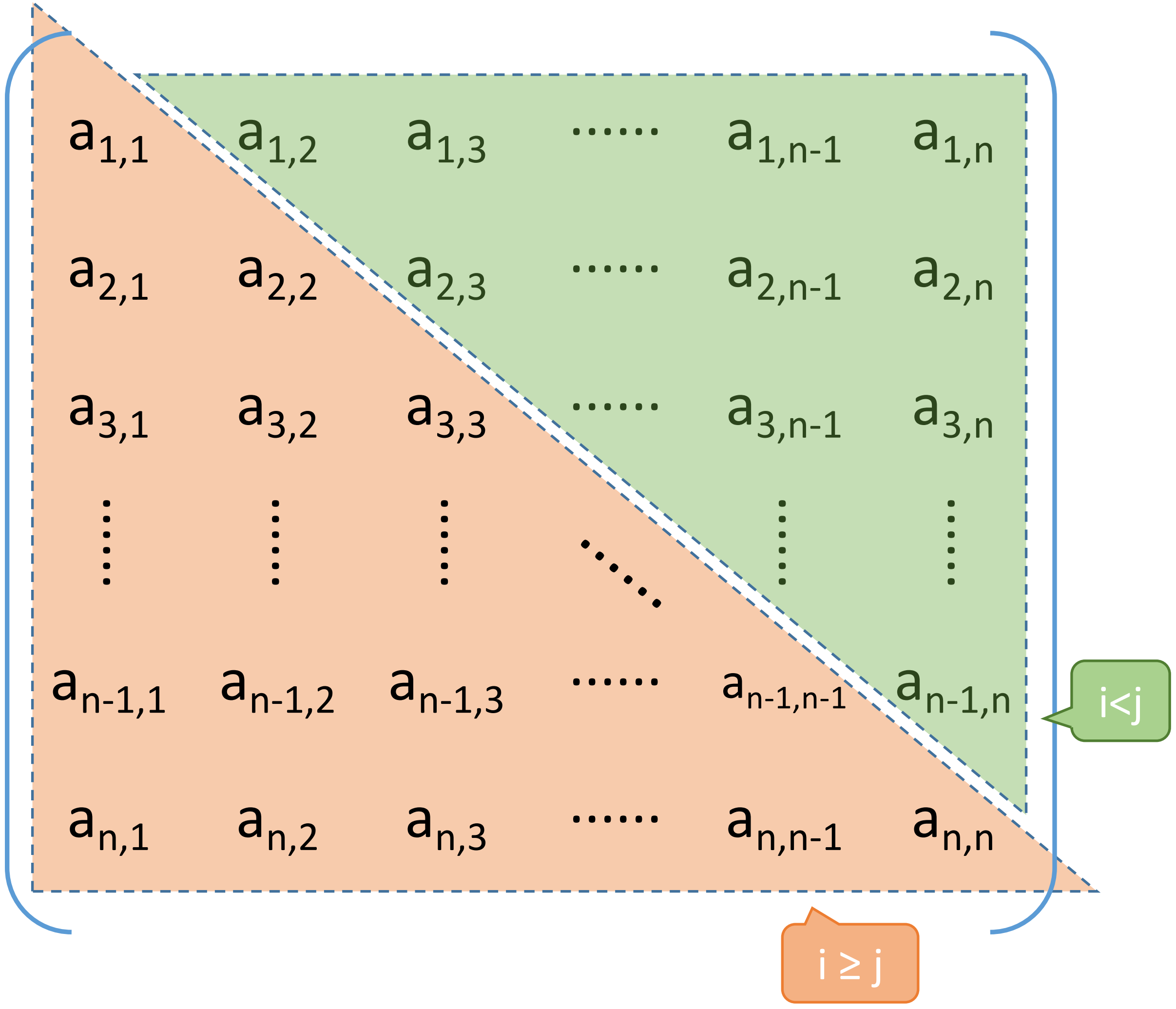
	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	0	0	0	0
C	1	0	0	0	0	0
D	1	0	0	0	0	0
E	0	1	1	0	0	0
F	0	1	0	1	0	0

空间复杂度：$O(|V|^2)$ ——只和顶点数相关，和实际的边数无关

适合用于存储稠密图

无向图的邻接矩阵是对称矩阵，可以压缩存储（只存储上三角区/下三角区）

回顾：对称矩阵的压缩存储



策略：只存储主对角线+下三角区

按行优先原则将各元素存入一维数组中。

B[0]	B[1]	B[2]	B[3]	….		B[$\frac{n(n+1)}{2}$-1]
$a_{1,1}$	$a_{2,1}$	$a_{2,2}$	$a_{3,1}$	……	$a_{n,n-1}$	$a_{n,n}$

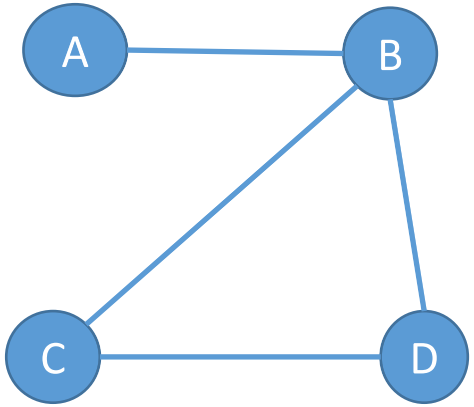
矩阵下标 → 一维数组下标

$a_{i,j}$ → B[k]

$a_{i,j} = a_{j,i}$ （对称矩阵性质）

$$k=\begin{cases}\frac{i(i-1)}{2}+j-1, & i\geqslant j\text{（下三角区和主对角线元素）}\\ \frac{j(j-1)}{2}+i-1, & i<j\text{（上三角区元素}a_{ij}=a_{ji}\text{）}\end{cases}$$

邻接矩阵法的性质



	A	B	C	D
A	0	1	0	0
B	1	0	1	1
C	0	1	0	1
D	0	1	1	0

设图G的邻接矩阵为A（矩阵元素为0/1），则A^n的元素$A^n[i][j]$等于由顶点i到顶点j的长度为n的路径的数目

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

$$A^2[1][4] = a_{1,1}\ a_{1,4} + a_{1,2}\ a_{2,4} + a_{1,3}\ a_{3,4} + a_{1,4}\ a_{4,4} = 1$$

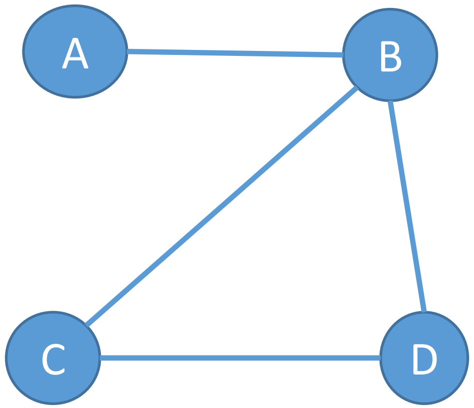
$$A^2[2][2] = a_{2,1}\ a_{1,2} + a_{2,2}\ a_{2,2} + a_{2,3}\ a_{3,2} + a_{2,4}\ a_{4,2} = 3$$

$$A^2[3][3] = a_{3,1}\ a_{1,3} + a_{3,2}\ a_{2,3} + a_{3,3}\ a_{3,3} + a_{3,4}\ a_{4,3} = 1$$

$$A^2[1][2] = a_{1,1}\ a_{1,2} + a_{1,2}\ a_{2,2} + a_{1,3}\ a_{3,2} + a_{1,4}\ a_{4,2} = 1$$

$$A^2 = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{3} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{2} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{2} \end{bmatrix}$$

邻接矩阵法的性质



	A	B	C	D
A	0	1	0	0
B	1	0	1	1
C	0	1	0	1
D	0	1	1	0

设图G的邻接矩阵为A（矩阵元素为0/1），则A^n的元素$A^n[i][j]$等于由顶点i到顶点j的长度为n的路径的数目

$$A^3 = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{3} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{2} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{2} \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{3} & \mathbf{1} & \mathbf{1} \\ \mathbf{3} & \mathbf{2} & \mathbf{4} & \mathbf{4} \\ \mathbf{1} & \mathbf{4} & \mathbf{2} & \mathbf{3} \\ \mathbf{1} & \mathbf{4} & \mathbf{3} & \mathbf{2} \end{bmatrix}$$

知识回顾与重要考点

邻接矩阵法要点回顾：

- 如何计算指定顶点的度、入度、出度（分无向图、有向图来考虑）？时间复杂度如何？
- 如何找到与顶点相邻的边（入边、出边）？时间复杂度如何？
- 如何存储带权图？
- 空间复杂度——O(|V|²)，适合存储稠密图
- 无向图的邻接矩阵为对称矩阵，如何压缩存储？
- 设图G的邻接矩阵为A（矩阵元素为0/1），则An的元素An[i][j]等于由顶点i到顶点j的长度为n的路径的数目

欢迎大家对本节视频进行评价~

学员评分：6.2.1 邻接矩阵法

扫一扫二维码打开或分享给好友

– 腾讯文档 –

可多人实时在线编辑，权限安全可控

公众号：王道在线

b站：王道计算机教育

抖音：王道计算机考研