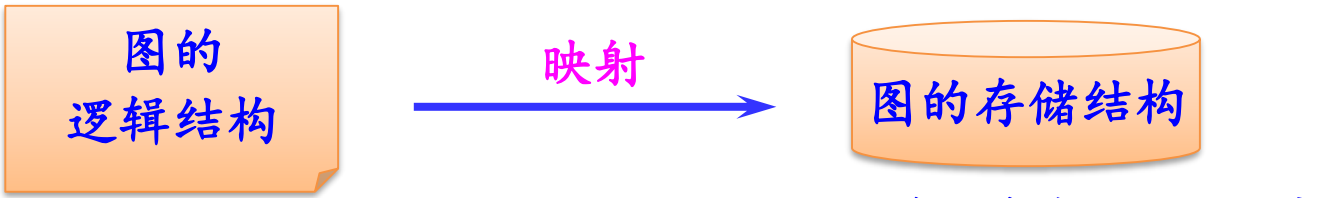
8.2 图的存储结构



- 存储每个顶点的信息
- 存储每条边的信息

图的两种主要存储结构：

- 邻接矩阵
- 邻接表

8.2.1 邻接矩阵存储方法

邻接矩阵是表示顶点之间相邻关系的矩阵。设G=(V，E)是具有n（$n>0$）个顶点的图，顶点的编号依次为0～$n-1$。

G的邻接矩阵A是n阶方阵，其定义如下：

（1）如果G是无向图，则：

A[i][j]=1：若(i,j)∈E(G)　0:其他

（2）如果G是有向图，则：

A[i][j]=1：若<i,j>∈E(G)　0:其他

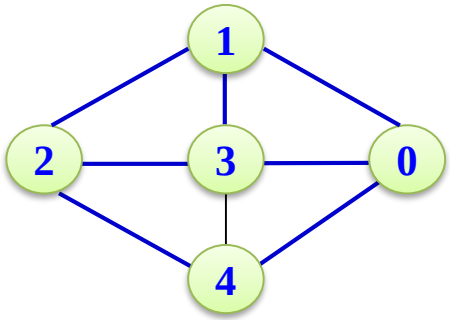
（3）如果G是带权无向图，则：

$A[i][j]= w_{ij}$：若$i \neq j$且$(i,j) \in E(G)$　0：$i=j$　∞：其他

（4）如果G是带权有向图，则：

$A[i][j]= w_{ij}$：若$i \neq j$且$<i,j> \in E(G)$　0：$i=j$　∞：其他

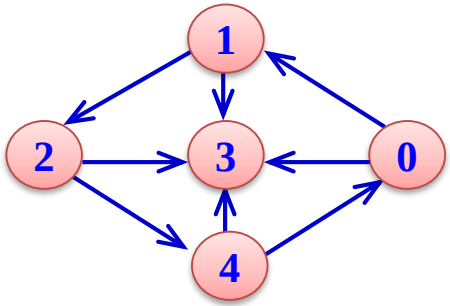
邻接矩阵演示



$A_1=$

	0	1	2	3	4
0	0	1	0	1	1
1	1	0	1	1	0
2	0	1	0	1	1
3	1	1	1	0	1
4	1	0	1	1	0

对称



$A_2=$

	0	1	2	3	4
0	0	1	0	1	0
1	0	0	1	1	0
2	0	0	0	1	1
3	0	0	0	0	0
4	1	0	0	1	0

不对称

邻接矩阵的主要特点：

- 一个图的邻接矩阵表示是唯一的。
- 特别适合于稠密图的存储。

邻接矩阵的存储空间为$O(n^2)$

图的邻接矩阵存储类型定义如下：

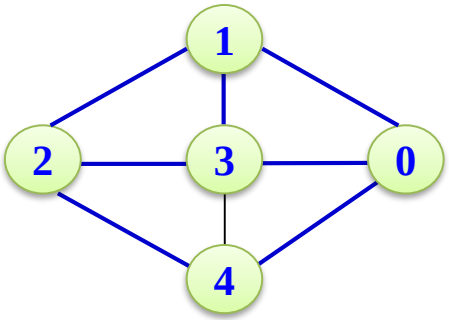
```
#define  MAXV  <最大顶点个数>
typedef struct
{    int no;                          //顶点编号
     InfoType info;                   //顶点其他信息
} VertexType;

typedef struct                        //图的定义
{    int edges[MAXV][MAXV];           //邻接矩阵
     int n，  e;                      //顶点数，边数
     VertexType vexs[MAXV];           //存放顶点信息
}  MGraph;
```

声明顶点的类型

声明的邻接矩阵类型

8.2.2 邻接表存储方法

- 对图中每个顶点*i*建立一个单链表，将顶点*i*的所有邻接点链起来。



顶点0的单链表：1 → 3 → 4 ∧

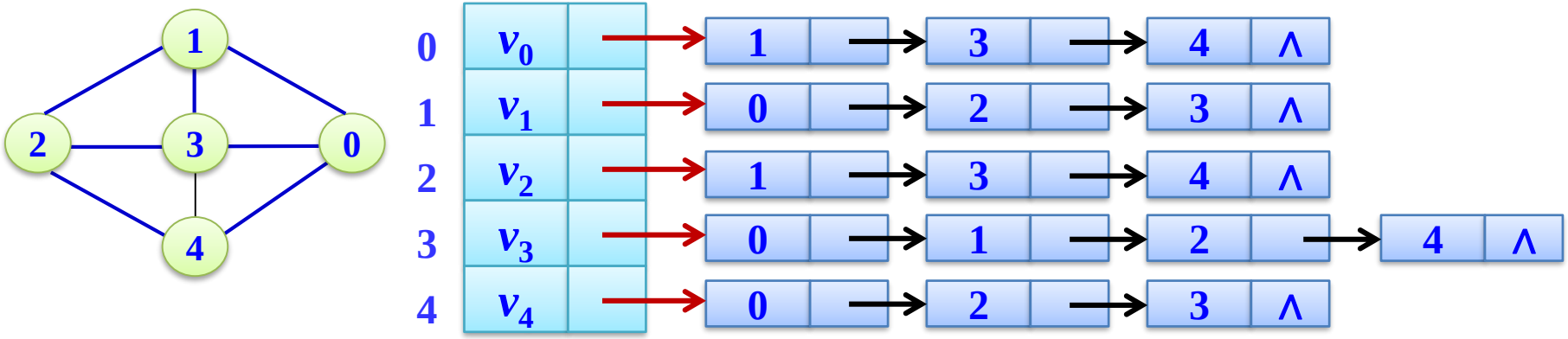
顶点1的单链表：0 → 2 → 3 ∧

顶点2的单链表：1 → 3 → 4 ∧

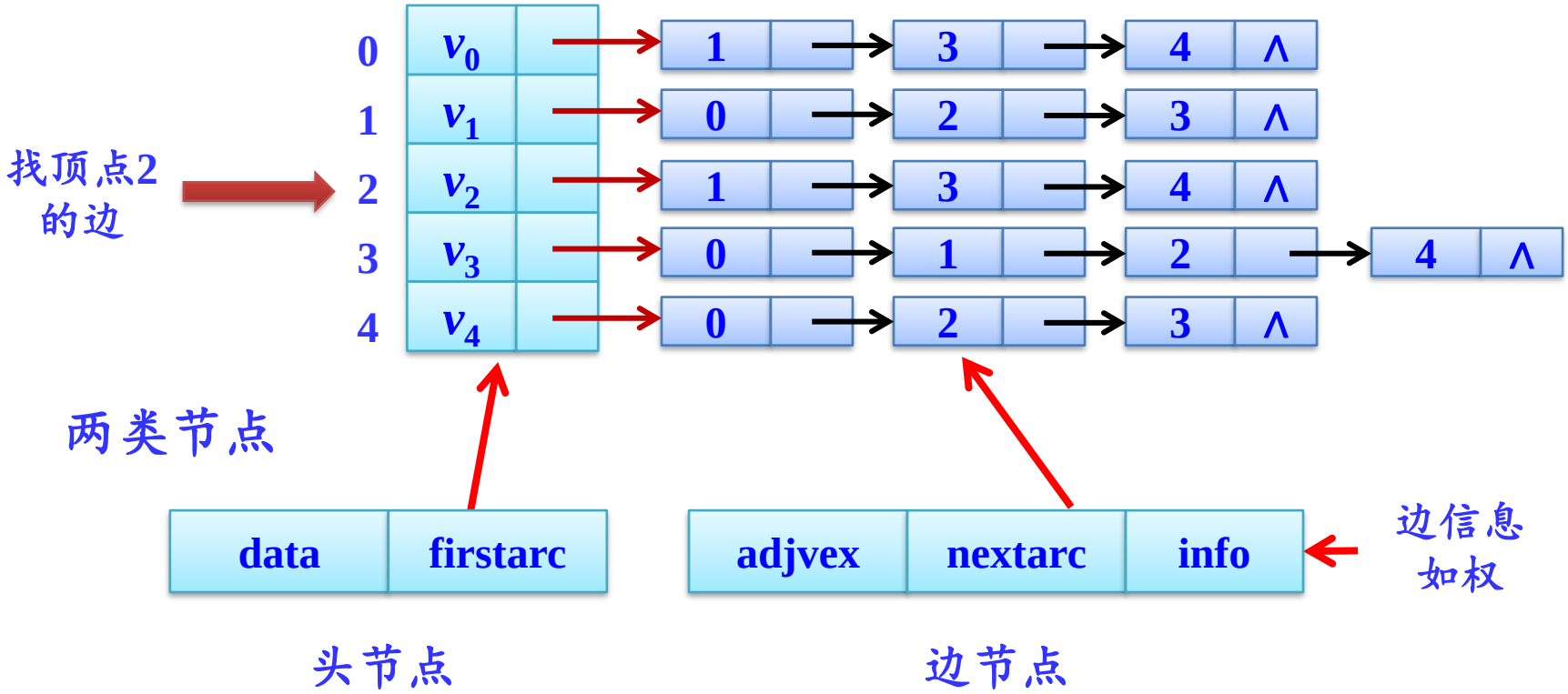
顶点3的单链表：0 → 1 → 2 → 4 ∧

顶点4的单链表：0 → 2 → 3 ∧

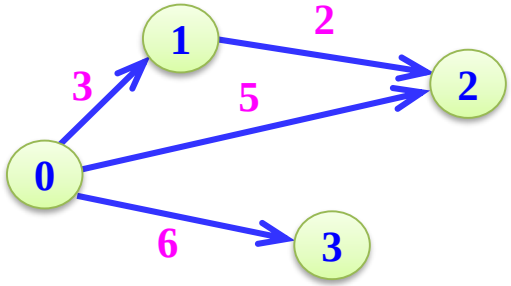
- 每个单链表上添加一个表头节点（表示顶点信息）。并将所有表头节点构成一个数组，下标为*i*的元素表示顶点*i*的表头节点。

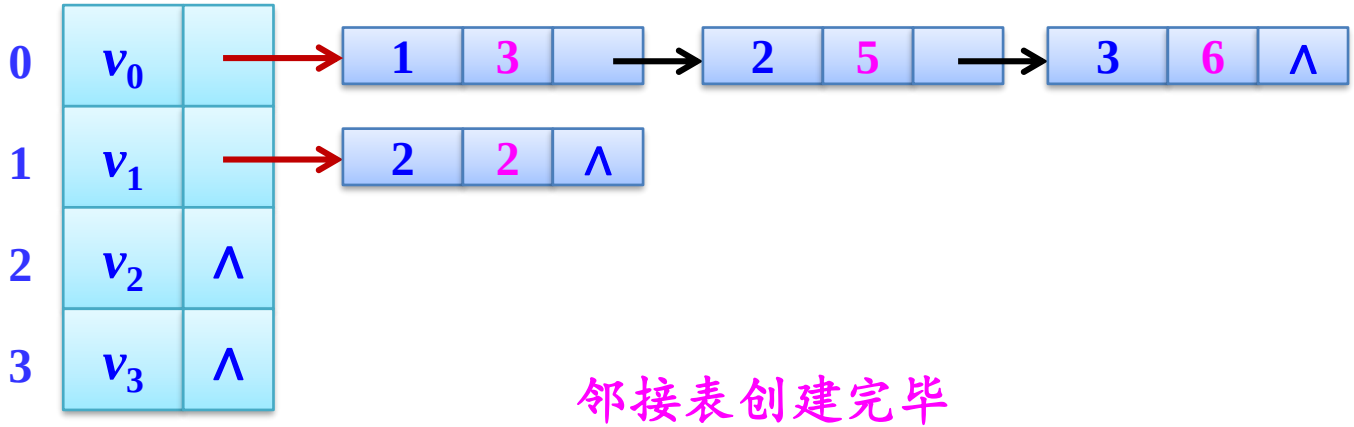


图的邻接表存储方法是一种顺序分配与链式分配相结合的存储方法。



一个网

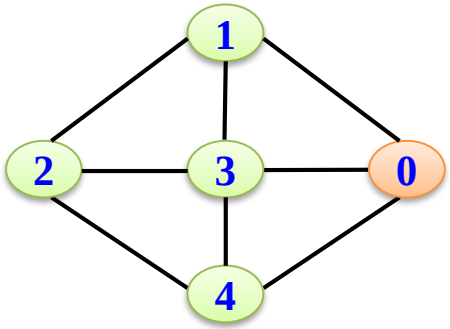


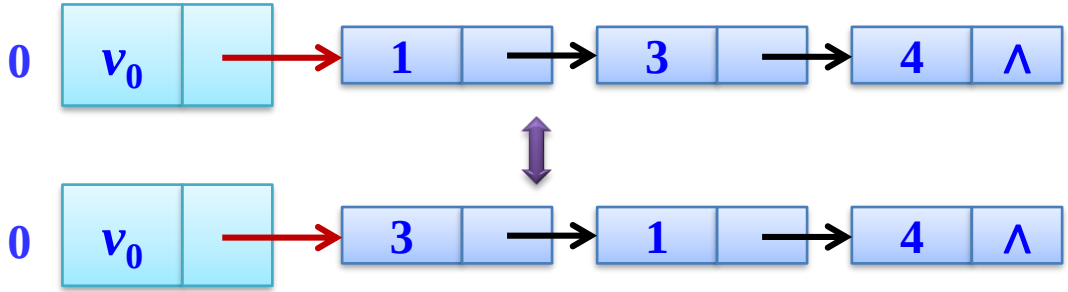


邻接表创建完毕

邻接表的特点如下：

- 邻接表表示不唯一。





- 特别适合于稀疏图存储。

邻接表的存储空间为$O(n+e)$

图的邻接表存储类型定义如下：

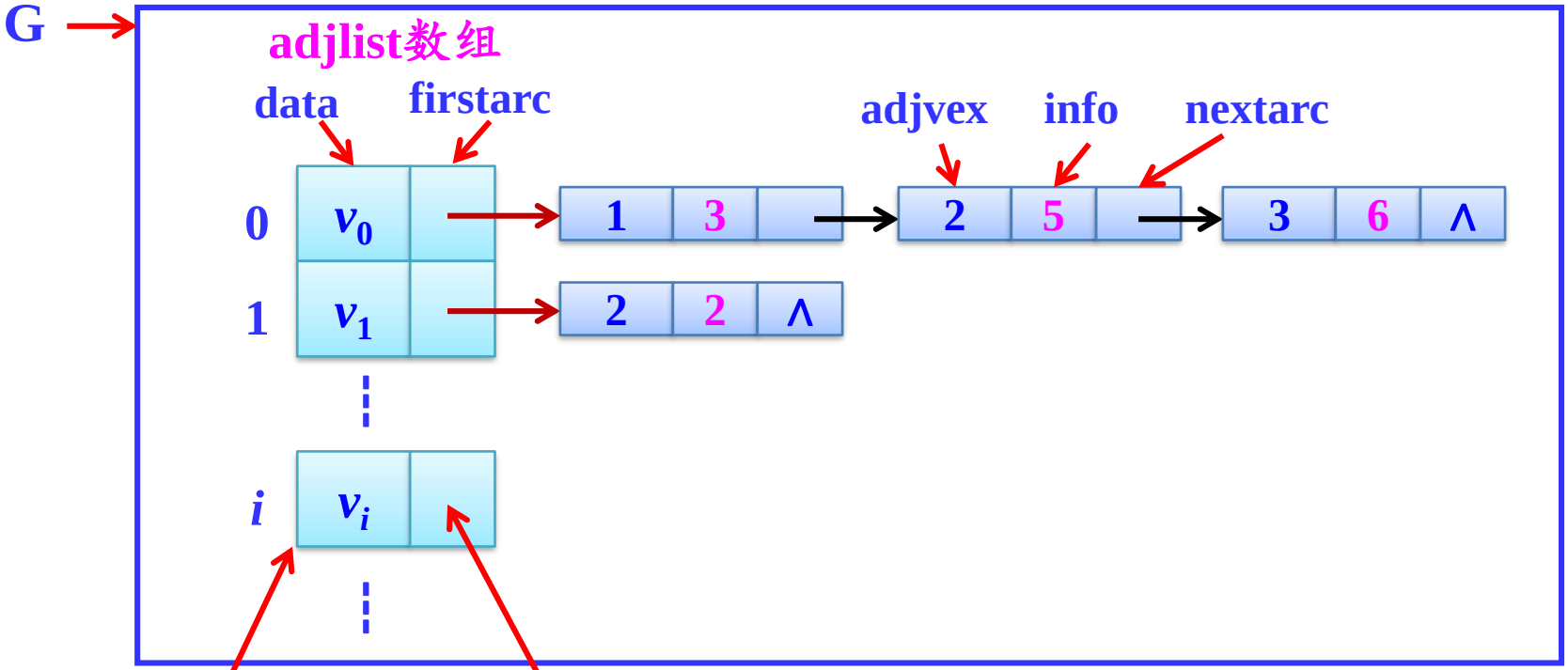
```
typedef struct ANode
{    int adjvex;                    //该边的终点编号
     struct ANode *nextarc;         //指向下一条边的指针
     InfoType info;                 //该边的权值等信息
} ArcNode;

typedef struct Vnode
{   Vertex data;                    //顶点信息
    ArcNode *firstarc;              //指向第一条边
} VNode;

typedef struct
{    VNode adjlist[MAXV] ;          //邻接表
     int n,e;                       //图中顶点数n和边数e
} ALGraph;
```

声明**边节点**类型

声明邻接表**头节点**类型

声明图邻接表类型

一个邻接表通常用指针引用：



引用头节点：G->adjlist[*i*]

引用头节点的指针域：G->adjlist[*i*].firstarc

思考题

图的邻接矩阵和邻接表两种存储结构各有什么优缺点？

——本讲完——