石家莊鐵道大學
SHIJIAZHUANG TIEDAO UNIVERSITY

在线开放课程

数据结构— 图

图的存储结构—邻接表

主讲 ： 石玉晶

目录

如何存储图?

考虑图的定义，图是由顶点和边组成的，分别考虑如何存储顶点、如何存储边。

一、邻接表表示法

- 邻接表是图的一种**链式存储结构**。
- 边的表示：为图中**每个顶点建立一个单链表**，单链表中的结点表示依附于该顶点的**边**(实际存储的是：与该边相关联的另一个顶点)。

单链表边结点(vi,vj)

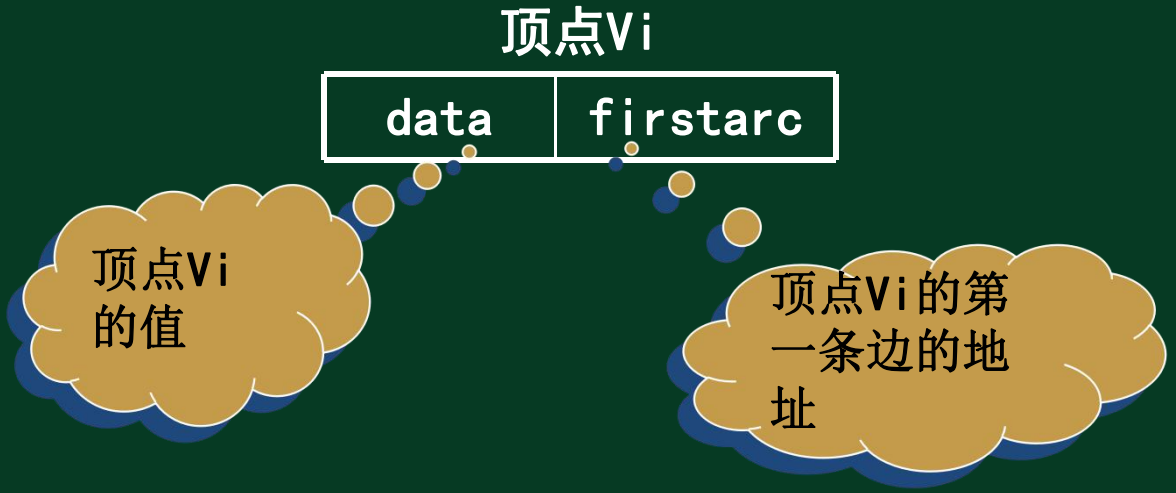
与Vi关联的边的另一端的顶点Vj

当前边的信息，如权值

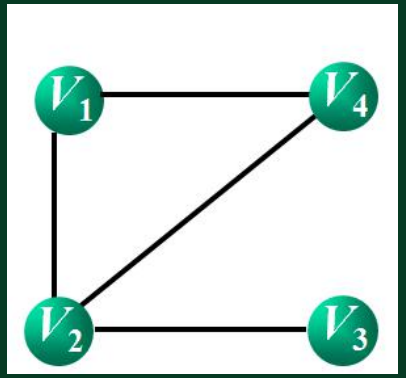
与Vi关联的下一个边的节点地址

一、邻接表表示法

- 邻接表是图的一种**链式存储结构**。
- 用**一维数组**存储图的顶点信息。

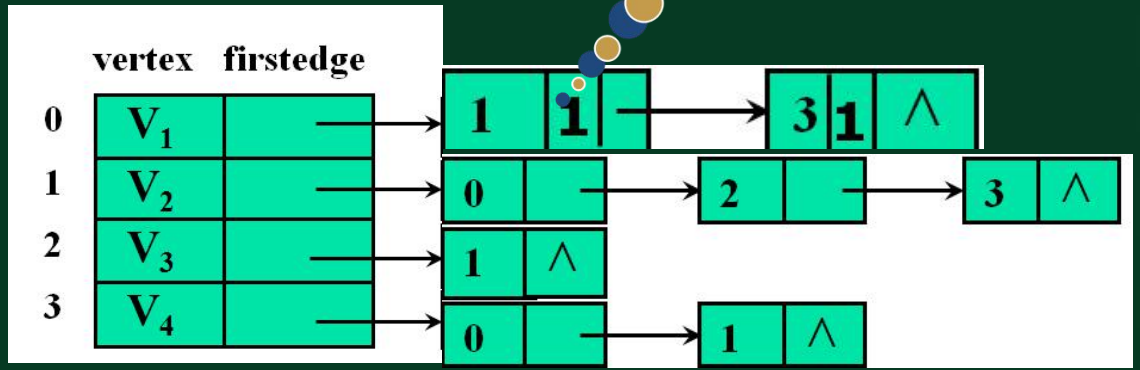


一、邻接表表示法



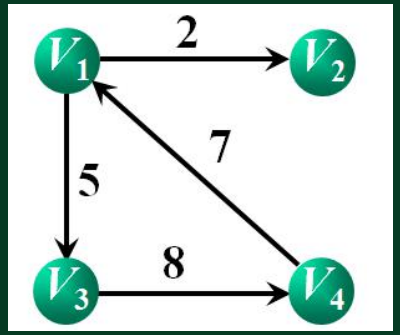
边没有权值的话，可以省略

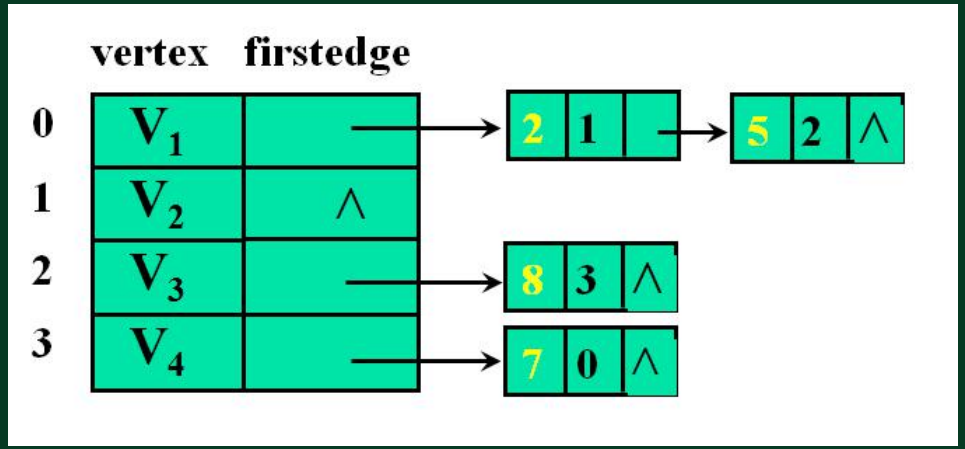
顶点=



一、邻接表表示法

- 网图的邻接表





一、邻接表表示法

图的邻接表存储表示

```
#define  MVNum     100;  // 最大顶点个数
typedef struct ArcNode{  //边节点
    int  adjvex;           // 该边所指向的顶点的下标
    struct ArcNode   *nextarc;    // 下一条边的节点地址
    OtherInfo       info;       // 该边相关信息
    } ArcNode;
typedef struct VNode {    // 顶点信息
    VertexType    data;
    ArcNode    *firstarc;          // 指向第一条依附该顶点的边
} VNode, AdjList[MVNum];  //顶点一维数组类型
typedef struct {
    AdjList vertices;                 // 顶点数组
    int vexnum, arcnum;               // 图的当前顶点数和边数
    } ALGraph;
```

二、使用邻接表表示法创建无向图

- 【算法思想】
  - （1）输入总顶点数和总边数。
  - （2）依次输入点的信息，存入顶点一维数组中。
  - （3）输入边信息（边的两端顶点和info），创建该边的节点，并加入到邻接表中。

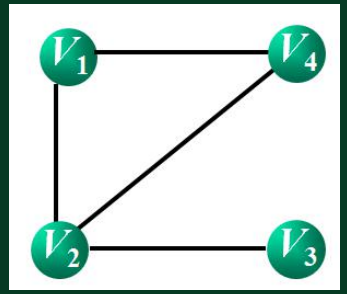
二、使用邻接表表示法创建无向图


```
Status CreateUDG(ALGraph &G){
    cin>>G.vexnum>>G.arcnum;       //依次输入顶点个数和边个数
    for(i = 0;i<G.vexnum; ++i) {   //依次输入顶点的信息
          cin>>G. vertices[i].data;
          G. vertices[i].firstarc=NULL;
       }
    /* 以下代码为创建边*/
      for(k = 0;k<G.arcnum;++k){     //构造邻接表
       cin>>v1>>v2;
        i = LocateVex(G, v1);  j = LocateVex(G, v2);
        p1=new ArcNode;       //创建边节点，并加入到V1的邻接表中
        p1->nextarc=G.vertices[i].firstarc;
        G.vertices[i].firstarc=p1;
        //因为是无向图，所以还要再创建一个边节点，加入到v2的邻接表中
            // 代码省略
     }//for
}
```

二、使用邻接表表示法创建无向图

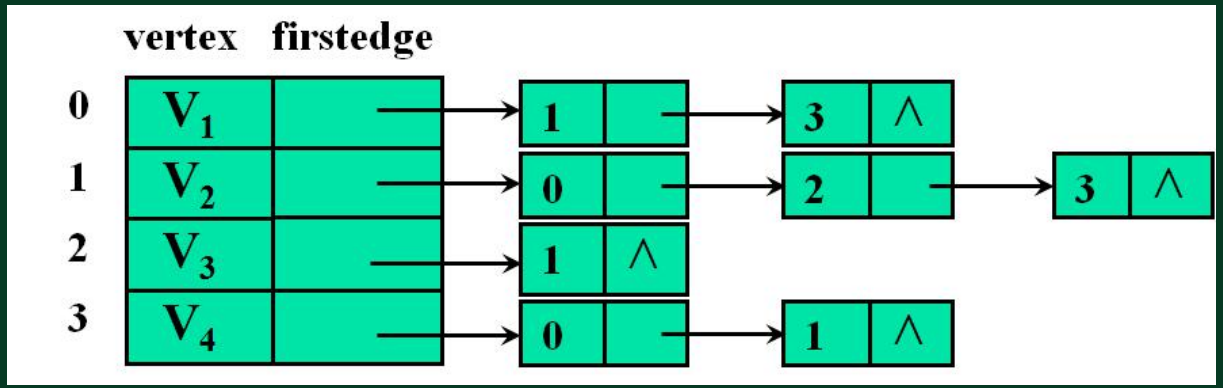
- 算法的时间复杂度
  - $O(n+e)$
- 如果创建的是有向图，则更加简单，每读入一个顶点对序号<i，j>，仅需生成一个边节点p1，并插入到Vi的邻接表头部即可。

三、邻接表相关算法



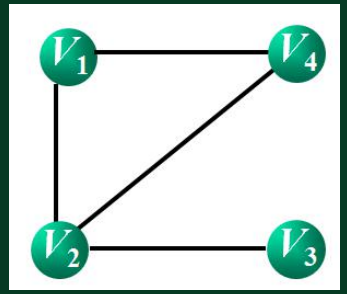
- 无向图的邻接表

  如何求顶点 i 的度？



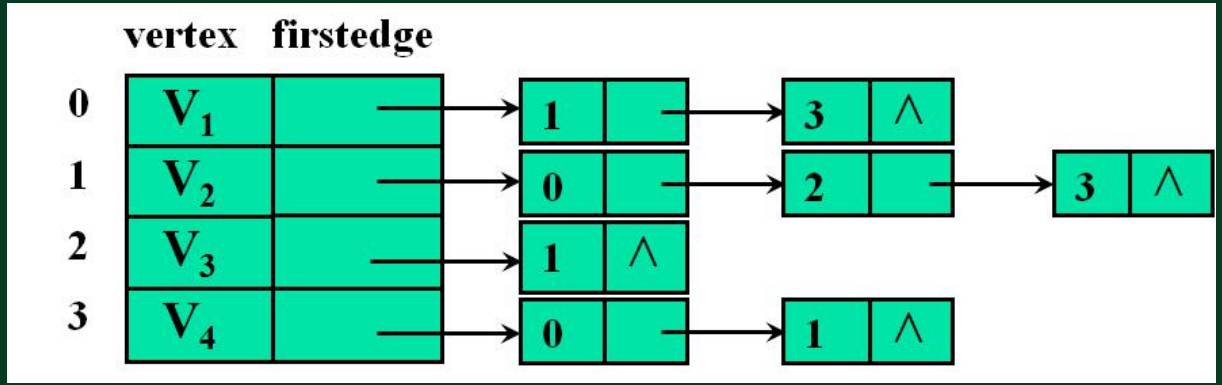
顶点i的边表中结点的个数。

三、邻接表相关算法



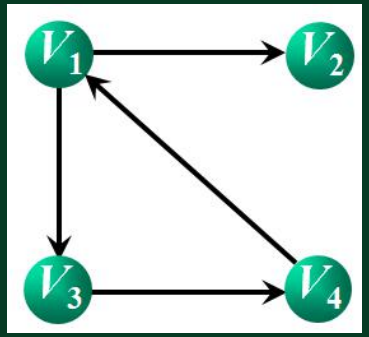
- 无向图的邻接表

  如何判断顶点 *i* 和顶点 *j* 之间是否存在边?



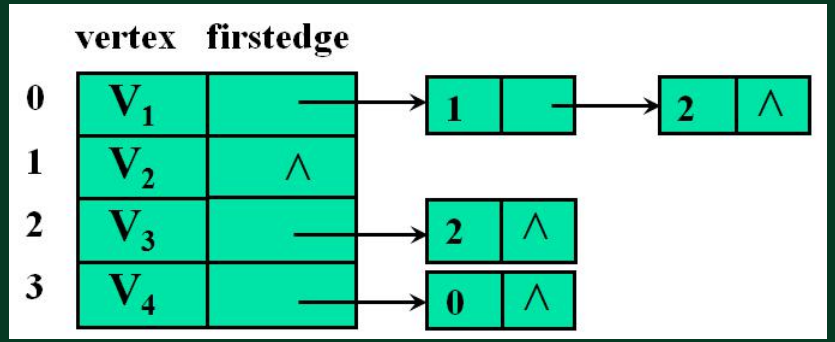
测试顶点 *i* 的边表中是否存在终点为 *j* 的结点。

三、邻接表相关算法



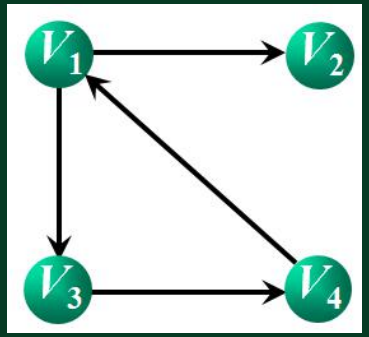
- 有向图的邻接表

  如何求顶点 i 的出度？



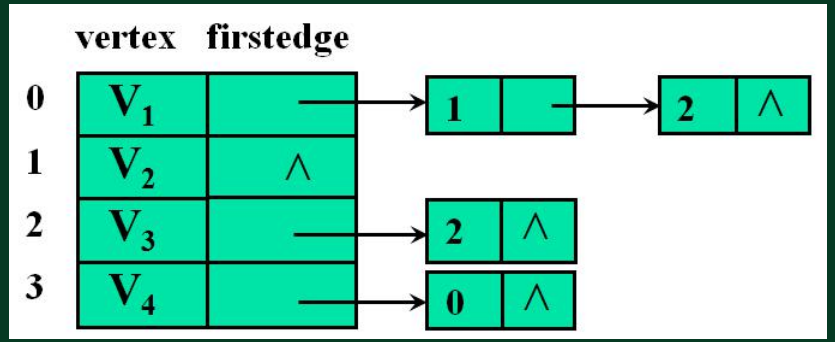
顶点i的边表中结点的个数。

三、邻接表相关算法



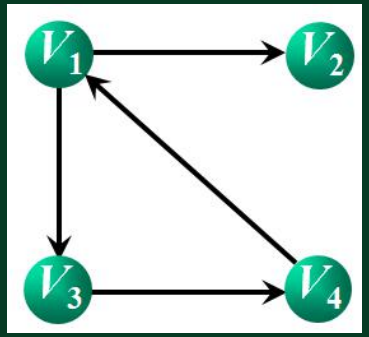
- 有向图的邻接表

  如何求顶点 i 的入度？



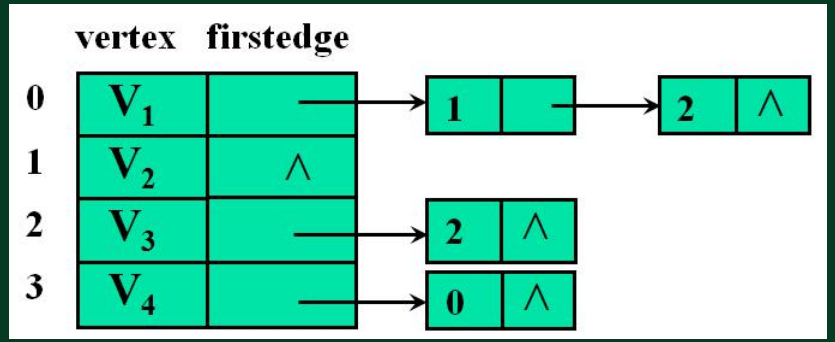
各顶点的边表中以顶点 i 为终点的结点的个数。

三、邻接表相关算法



- 有向图的邻接表

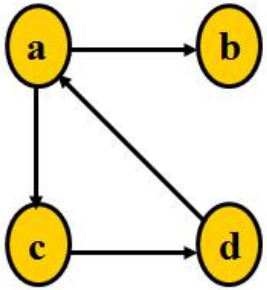
  如何求顶点 i 的所有邻接点？



遍历顶点 i 的边表，该边表中的所有终点都是顶点 i 的邻接点。

四、邻接表表示法总结

- 优缺点
  - **优点**：空间较省；无向图容易求各顶点的度；有向图容易求顶点的出度；
  - **缺点**：求有向图顶点的入度则不容易，要遍历整个表。
  - 为了求顶点的入度，有时可设逆邻接表（指向某顶点的邻接点链接成单链表）。



	data	firstarc	adjvex	next
0	a	→	3	^
1	b	→	0	^
2	c	→	0	^
3	d	→	2	^

逆邻接表

小结

- 图的存储结构——邻接表表示法
- 使用邻接表表示图时的若干算法

- 邻接表适合于稀疏图，邻接矩阵适合于稠密图

小结

- 接下来学习：
- 图的遍历—深度优先搜索

谢谢！