阅读：21,304　　作者：解学武

图的邻接多重表存储法（超详细）

上一节　　下一节

前面讲过，无向图的存储可以使用邻接表，但在实际使用时，如果想对图中某顶点进行实操（修改或删除），由于邻接表中存储该顶点的节点有两个，因此需要操作两个节点。

为了提高在无向图中操作顶点的效率，本节学习一种新的适用于存储无向图的方法——邻接多重表。

> 注意，邻接多重表仅适用于存储无向图或无向网。

邻接多重表存储无向图的方式，可看作是邻接表和十字链表的结合。同邻接表和十字链表存储图的方法相同，都是独自为图中各顶点建立一张链表，存储各顶点的节点作为各链表的首元节点，同时为了便于管理将各个首元节点存储到一个数组中。各首元节点结构如图 1 所示：

图 1 邻接多重表各首元节点的结构示意图

图 1 中各区域及其功能为：

- data：存储此顶点的数据；
- firstedge：指针域，用于指向同该顶点有直接关联的存储其他顶点的节点。

从图 1 可以看到，邻接多重表采用与邻接表相同的首元节点结构。但各链表中其他节点的结构与十字链表中相同，如图 2 所示：

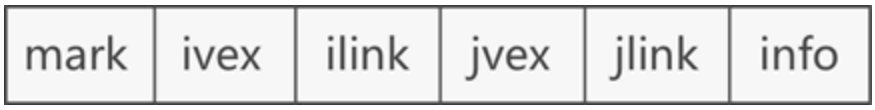

图 2 邻接多重表中其他节点结构

图 2 节点中各区域及功能如下：

- mark：标志域，用于标记此节点是否被操作过，例如在对图中顶点做遍历操作时，为了防止多次操作同一节点，mark 域为 0 表示还未被遍历；mark 为 1 表示该节点已被遍历；
- ivex 和 jvex：数据域，分别存储图中各边两端的顶点所在数组中的位置下标；
- ilink：指针域，指向下一个存储与 ivex 有直接关联顶点的节点；
- jlink：指针域，指向下一个存储与 jvex 有直接关联顶点的节点；
- info：指针域，用于存储与该顶点有关的其他信息，比如无向网中各边的权；

综合以上信息，如果我们想使用邻接多重表存储图 3a) 中的无向图，则与之对应的邻接多重表如图 3b) 所示：

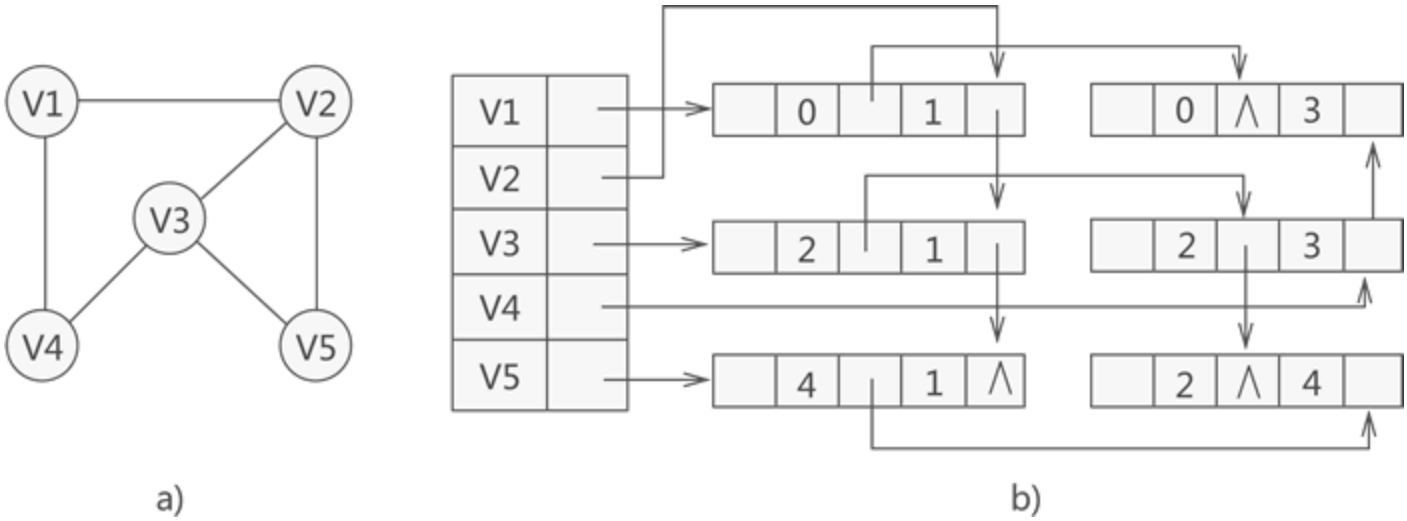

图 3 无向图及其对应的邻接多重表

从图 3 中，可直接找到与各顶点有直接关联的其他顶点。比如说，与顶点 V1 有关联的顶点为存储在数组下标 1 处的 V2 和数组下标 3 处的 V4，而与顶点 V2 有关联的顶点有 3 个，分别是 V1、V3 和 V5。

图 3 中邻接多重表的整体结构转化为 C 语言代码如下所示：

```
#define MAX_VERTEX_NUM 20                    //图中顶点的最大个数
#define InfoType int                         //边含有的信息域的数据类型
#define VertexType int                       //图顶点的数据类型
typedef enum {unvisited,visited}VisitIf;     //边标志域
typedef struct EBox{
    VisitIf mark;                            //标志域
    int ivex,jvex;                           //边两边顶点在数组中的位置下标
    struct EBox * ilink,*jlink;              //分别指向与ivex、jvex相关的下一个边
    InfoType *info;                          //边包含的其它的信息域的指针
}EBox;
typedef struct VexBox{
    VertexType data;                         //顶点数据域
    EBox * firstedge;                        //顶点相关的第一条边的指针域
}VexBox;
typedef struct {
    VexBox adjmulist[MAX_VERTEX_NUM];//存储图中顶点的数组
    int vexnum,degenum;//记录途中顶点个数和边个数的变量
}AMLGraph;
```

上一节 下一节