

小白专场：如何建立图

浙江大学 陈 越

用邻接矩阵表示图

$$G[i][j] = \begin{cases} 1 & \text{若 } \langle v_i, v_j \rangle \text{ 是 } G \text{ 中的边} \\ 0 & \text{否则} \end{cases}$$

```
typedef struct GNode *PtrToGNode;  
struct GNode {  
    int Nv;    /* 顶点数 */  
    int Ne;    /* 边数  */  
    WeightType G[MaxVertexNum][MaxVertexNum];  
    DataType Data[MaxVertexNum]; /* 存顶点的数据 */  
};  
typedef PtrToGNode MGraph; /* 以邻接矩阵存储的图类型 */
```

MGraph初始化

- 初始化一个有VertexNum个顶点但没有边的图

```
typedef int Vertex; /* 用顶点下标表示顶点,为整型 */
MGraph CreateGraph( int VertexNum )
{
    Vertex V, W;
    MGraph Graph;

    Graph = (MGraph)malloc(sizeof(struct GNode));
    Graph->Nv = VertexNum;
    Graph->Ne = 0;

    /* 注意: 这里默认顶点编号从0开始, 到(Graph->Nv - 1) */
    for (V=0; V<Graph->Nv; V++)
        for (W=0; W<Graph->Nv; W++)
            Graph->G[V][W] = 0; /* 或INFINITY */

    return Graph;
}
```

向MGraph中插入边

```
typedef struct ENode *PtrToENode;
struct ENode {
    Vertex V1, V2;      /* 有向边<v1, v2> */
    WeightType Weight;  /* 权重 */
};
typedef PtrToENode Edge;

void InsertEdge( MGraph Graph, Edge E )
{
    /* 插入边 <v1, v2> */
    Graph->G[E->V1][E->V2] = E->Weight;

    /* 若是无向图, 还要插入边<v2, v1> */
    Graph->G[E->V2][E->V1] = E->Weight;
}
```

完整地建立一个MGraph

■ 输入格式

Nv Ne

V1 V2 Weight

.....

```
MGraph BuildGraph()
{
    MGraph Graph;
    Edge E;
    Vertex V;
    int Nv, i;

    scanf("%d", &Nv);
    Graph = CreateGraph(Nv);
    scanf("%d", &(Graph->Ne));
    if ( Graph->Ne != 0 ) {
        E = (Edge)malloc(sizeof(struct ENode));
        for (i=0; i<Graph->Ne; i++) {
            scanf("%d %d %d",
                &E->V1, &E->V2, &E->Weight);
            InsertEdge( Graph, E );
        }
    }

    /* 如果顶点有数据的话, 读入数据 */
    for (V=0; V<Graph->Nv; V++)
        scanf(" %c", &(Graph->Data[V]));

    return Graph;
}
```

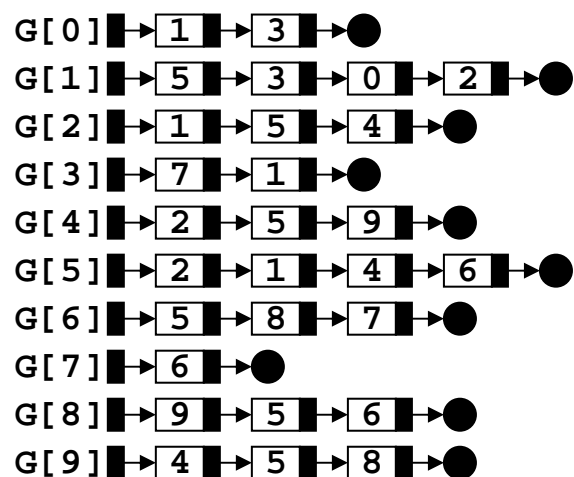
如果不要这么麻烦.....

```
int G[MAXN][MAXN], Nv, Ne;
void BuildGraph()
{   int i, j, v1, v2, w;

    scanf("%d", &Nv);
    /* CreateGraph */
    for (i=0; i<Nv; i++)
        for (j=0; j<Nv; j++)
            G[i][j] = 0; /* 或INFINITY */
    scanf("%d", &Ne);
    for (i=0; i<Ne; i++) {
        scanf("%d %d %d", &v1, &v2, &w);
        /* InsertEdge */
        G[v1][v2] = w;
        G[v2][v1] = w;
    }
}
```

用邻接表表示图

- 邻接表: $G[N]$ 为指针数组, 对应矩阵每行一个链表, 只存非0元素



```
typedef struct GNode *PtrToGNode;
struct GNode {
    int Nv;      /* 顶点数 */
    int Ne;      /* 边数 */
    AdjList G;   /* 邻接表 */
};
```

```
typedef struct Vnode{
    PtrToAdjVNode FirstEdge;
    DataType Data; /* 存顶点的数据 */
} AdjList[MaxVertexNum];
/* AdjList是邻接表类型 */
```

```
typedef PtrToGNode LGraph;
/* 以邻接表方式存储的图类型 */
```

```
typedef struct AdjVNode *PtrToAdjVNode;
struct AdjVNode {
    Vertex AdjV; /* 邻接点下标 */
    WeightType Weight; /* 边权重 */
    PtrToAdjVNode Next;
};
```

LGraph初始化

- 初始化一个有VertexNum个顶点但没有边的图

```
typedef int Vertex; /* 用顶点下标表示顶点,为整型 */
LGraph CreateGraph( int VertexNum )
{
    Vertex V, W;
    LGraph Graph;

    Graph = (LGraph)malloc(sizeof(struct GNode));
    Graph->Nv = VertexNum;
    Graph->Ne = 0;

    /* 注意: 这里默认顶点编号从0开始, 到(Graph->Nv - 1) */
    for ( V=0; V<Graph->Nv; V++ )
        Graph->G[V].FirstEdge = NULL;

    return Graph;
}
```


向LGraph中插入边

```
void InsertEdge( LGraph Graph, Edge E )
{   PtrToAdjVNode NewNode;
```

```
    /* ***** 插入边 <v1, v2> ***** */
```

```
    /* 为v2建立新的邻接点 */
```

```
    NewNode = (PtrToAdjVNode)malloc(sizeof(struct AdjVNode));
```

```
    NewNode->AdjV = E->V2;
```

```
    NewNode->Weight = E->Weight;
```

```
    /* 将v2插入v1的表头 */
```

```
    NewNode->Next = Graph->G[E->V1].FirstEdge;
```

```
    Graph->G[E->V1].FirstEdge = NewNode;
```

```
    /* ***** 若是无向图，还要插入边 <v2, v1> ***** */
```

```
    /* 为v1建立新的邻接点 */
```

```
    NewNode = (PtrToAdjVNode)malloc(sizeof(struct AdjVNode));
```

```
    NewNode->AdjV = E->V1;
```

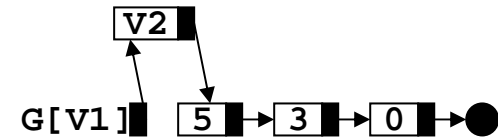
```
    NewNode->Weight = E->Weight;
```

```
    /* 将v1插入v2的表头 */
```

```
    NewNode->Next = Graph->G[E->V2].FirstEdge;
```

```
    Graph->G[E->V2].FirstEdge = NewNode;
```

```
}
```



完整地建立一个LGraph

```
LGraph BuildGraph()  
{    LGraph Graph;
```

..... 还会有区别吗?

```
}
```