数据结构实验五

张笑辰

机1901

201930310107

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define MaxVertexNum 100 // 最大顶点数设为100

#define INFINITY 65535 // ∞设为双字节无符号整数的最大值

typedef int Vertex; // 顶点下标（用顶点下标表示顶点）

typedef int WeightType; // 边的权值设为整型

typedef char DataType; // 顶点存储的数据类型设为整型

typedef struct GNode\* PtrToGNode; // 图结点

struct GNode {

int Nv; // 顶点数

int Ne; // 边数

WeightType G[MaxVertexNum][MaxVertexNum]; // 邻接矩阵

DataType Data[MaxVertexNum]; // 顶点数据，如果顶点无数据，则可以不必定义此变量

};

typedef PtrToGNode MGraph; // 以邻接矩阵存储的图类型

typedef struct ENode\* PtrToENode; // 边指针的定义

struct ENode { // 边的结构的定义

Vertex V1, V2; // 有向边<V1, V2>

WeightType Weight; // 权重

};

typedef PtrToENode Edge; // 边的定义（边的指针的别名）

MGraph CreateGraph(int VertexNum)

{

// 初始化一个有VertexNum个顶点，但没有边的图

Vertex V, W; // 表示顶点下标的临时变量

MGraph Graph;

Graph = (MGraph)malloc(sizeof(struct GNode)); // 创建图

Graph->Nv = VertexNum;

Graph->Ne = 0;

// 使用INFINITY初始化图的邻接矩阵的每个坐标点

// 邻接矩阵坐标编号：0 --- (Graph->Nv - 1)

for (V = 0; V < Graph->Nv; V++)

{

for (W = 0; W < Graph->Nv; W++)

{

Graph->G[V][W] = INFINITY;

}

}

return Graph;

}

// 插入边 - 本质上是针对邻接矩阵表示的边赋权

void InsertEdge(MGraph Graph, Edge E)

{

// 邻接矩阵为对称矩阵

Graph->G[E->V1][E->V2] = E->Weight;

// 如果是无向网图，邻接矩阵为对称矩阵

Graph->G[E->V2][E->V1] = E->Weight;

}

MGraph BuildGraph()

{

MGraph Graph;

Edge E;

Vertex V;

int Nv, i;

printf("输入顶点个数\n");

scanf\_s("%d", &Nv); // 读入顶点个数

Graph = CreateGraph(Nv); // 初始化有Nv个顶点但没有边的图

printf("输入边数\n");

scanf\_s("%d", &(Graph->Ne)); // 读入边数

if (Graph->Ne != 0) {

E = (Edge)malloc(sizeof(struct ENode));

for (i = 0; i < Graph->Ne; i++)

{

printf("输入边(V1, V2, Weight):\n");

scanf\_s("%d,%d,%d", &E->V1, &E->V2, &E->Weight); // 通过输入创建一条边

InsertEdge(Graph, E); // 将边插入图中

}

}

for (V = 0; V < Graph->Nv; V++)

{

printf("输入顶点:\n");

scanf\_s(" %c", &(Graph->Data[V])); // 输入各顶点的数据

printf("输入顶点值:%d\n", Graph->Data[V]);

}

return Graph;

}

void TraverseGraph(MGraph G)

{

printf("顶点数:\n", G->Nv);

for (int i = 0; i < G->Nv; i++)

{

printf("[");

for (int j = 0; j < G->Nv; j++)

{

printf(" %d(%c, %c) ", G->G[i][j], G->Data[i], G->Data[j]);

}

printf("]\n");

}

}

int main()

{

MGraph Graph = BuildGraph();

TraverseGraph(Graph);

return 0;

}

实验总结：无向图就为邻接矩阵关于对角线对称，边可有信息，实现过程较简单。输入总顶点数和总边数，依次输入点的信息存入顶点表中，是每个表头结点的指针域初始化为NULL  
，创建邻接表。依次输入每条边依附的两个顶点，确定这两个顶点的序号i和j之后，将此边结点分别插入Vi和Vj对应的两个边链表头部