**数据结构实验十二 图的存储实现**

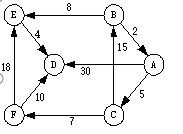
**一、实验目的**

1、 理解图的存储结构与基本操作；

2、 掌握图的创建过程

**二、实验内容**

**1.**请把下图以邻接矩阵的结构存储，并打印输出此邻接矩阵。



图的创建代码参考教材例题.

提示:首先构建图的逻辑模型，得出该图的顶点集和边集，调用相应的函数生成图的邻接矩阵，并打印出邻接矩阵。

2.用邻接表存储上图，并输出邻接表。（此题为选做）

三、【实验源代码】

 两个头文件加主文件

Adjmatrix.h

#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE 6

#define MAXVEX 50

#define INF 0

typedef struct MGraph {

    char vexs[MAXSIZE];

    int arc[MAXSIZE][MAXSIZE];

    int numVertexes;

    int numEdges;

}MGraph;

void createMGraph(MGraph \*G) {

    int i, j, k, w;

    G->numEdges = 9;

    G->numVertexes = 6;

    char vexs[] = {'A', 'B', 'C', 'D', 'E', 'F'};

    for (i = 0; i < G->numVertexes; i++) {

        G->vexs[i] = vexs[i];

    }

    for (i = 0; i < G->numVertexes; i++) {

        for (j = 0; j < G->numVertexes; j++) {

            G->arc[i][j] = INF;

        }

    }

    int edges[][3] = {{0,2,5}, {0,3,30}, {1,0,2}, {1,4,8}, {2,1,15}, {2,5,7}, {4,3,4}, {5,3,10}, {5,4,18}};

    for (k = 0; k < G->numEdges; k++) {

        i = edges[k][0];

        j = edges[k][1];

        w = edges[k][2];

        G->arc[i][j] = w;

    }

}

void printMGraph(MGraph G) {

    int i, j;

    for (i = 0; i < G.numVertexes; i++) {

        for (j = 0; j < G.numVertexes; j++) {

            printf("%d ", G.arc[i][j]);

        }

        printf("\n");

    }

}

 adjList.h

#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE 6  // 顶点数

#define MAXVEX 50  // 存储顶点的最大数量

typedef struct ENode {

    int adjvex;

    int weight;

    struct ENode \*nextarc;

}ENode;

typedef struct VNode {

    char data;

    ENode \*firstarc;

}VNode, AdjList[MAXSIZE];

typedef struct LGraph {

    AdjList adjList;

    int numVertexes;

    int numEdges;

}LGraph;

void createLGraph(LGraph \*G) {

    int i, j, k, w;

    G->numEdges = 9;

    G->numVertexes = 6;

    char vexs[] = {'A', 'B', 'C', 'D', 'E', 'F'};

    for (i = 0; i < G->numVertexes; i++) {

        G->adjList[i].data = vexs[i];

        G->adjList[i].firstarc = NULL;

    }

    int edges[][3] = {{0,2,5}, {0,3,30}, {1,0,2}, {1,4,8}, {2,1,15}, {2,5,7}, {4,3,4}, {5,3,10}, {5,4,18}};

    for (k = 0; k < G->numEdges; k++) {

        i = edges[k][0];

        j = edges[k][1];

        w = edges[k][2];

        ENode \*edge = (ENode \*)malloc(sizeof(ENode));

        edge->adjvex = j;

        edge->weight = w;

        edge->nextarc = G->adjList[i].firstarc;

        G->adjList[i].firstarc = edge;

    }

}

void printLGraph(LGraph G) {

    int i;

    ENode \*p;

    for (i = 0; i < G.numVertexes; i++) {

        printf("%d %c ", i, G.adjList[i].data);

        p = G.adjList[i].firstarc;

        while (p != NULL) {

            printf("-> %d(%d) ", p->adjvex, p->weight);

            p = p->nextarc;

        }

        printf("\n");

    }

}

Graph.c

#include <stdio.h>

#include <stdlib.h>

#include "adjMatrix.h"

#include "adjList.h"

int main() {

    MGraph G1;

    createMGraph(&G1);

    printf("Adjacency matrix:\n");

    printMGraph(G1);

    printf("\n");

    LGraph G2;

    createLGraph(&G2);

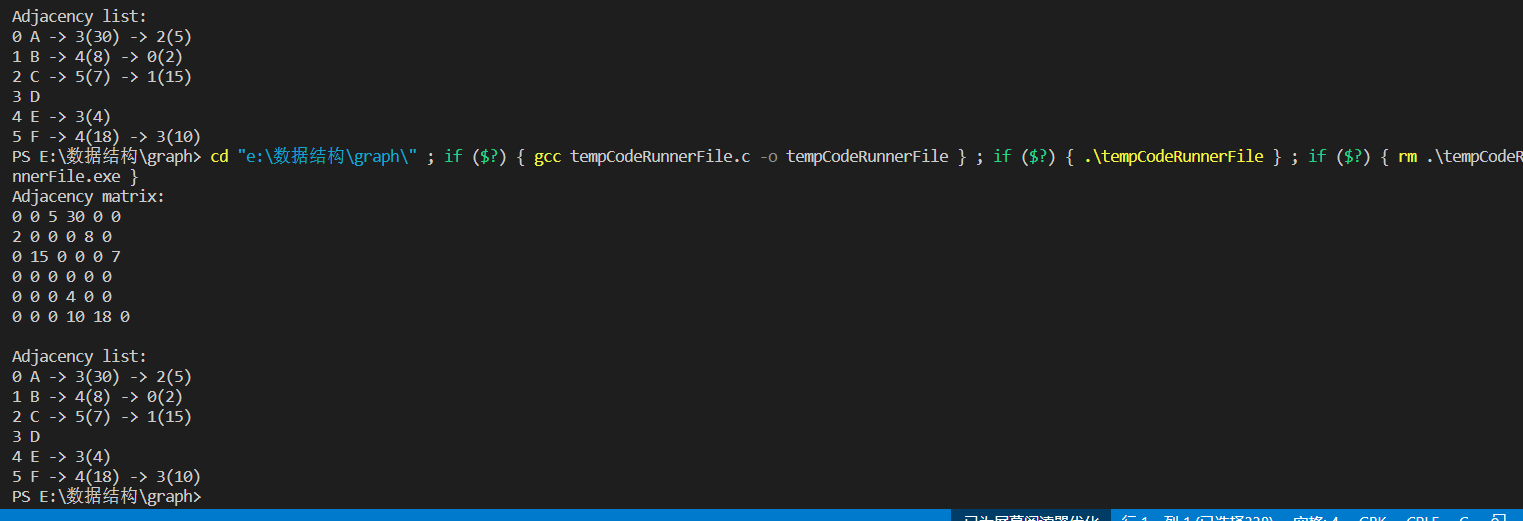
    printf("Adjacency list:\n");

    printLGraph(G2);

    return 0;

}

四、【实验结果】



五、【实验总结】

1. 顶点没有放在表里面，简单用一个数组来存了

2. 带权图的邻接表不用无穷来代替，矩阵看起来难看，用0来代替了无限了

