完成日期：2016.11.24

* **问题描述：**

实验04：实现Floyd算法，并求所示有向图中各顶点之间的最短路径及其长度。

* **算法思想：**

采用邻接矩阵形式存储图，D[][]存储权值。采用P[][]二元数组内线性表方式存储Path。采用数字方式存储路径，打印时再转换成字母。

按照Floyd算法，改变Path。

* **设计描述：**

（1）

typedef struct{

char vexs[MAX\_VERTEX\_NUM];//顶点向量

int arcs[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];//邻接矩阵

int vexnum,arcnum;//图的当前顶点数和弧数

GraphKind kind; //种类

}MGraph,\*MGPtr;

typedef struct{

int stop[MAX\_VERTEX\_NUM+1];

int ListLen;

}DisMatrix,\*Dis;//线性表存储路径

（2）int insert(Dis p,int i,int e)//用来实现权值为无穷时，结点路径的更新，Floyd函数调用

int contact(Dis p1,Dis p2,Dis p3)//用来实现路径更新,Flyod函数调用

void FLOYD(MGPtr G1,int D1 [][MAX\_VERTEX\_NUM],Dis P1[][MAX\_VERTEX\_NUM])//实现Floyd算法

void Printf(MGPtr G1,int D2[][MAX\_VERTEX\_NUM],Dis P2[][MAX\_VERTEX\_NUM])

（3）

算法描述：

Insert(Dis &p,i，e)//在i位置插入e

{ if(i不符合要求) return ERROR;

i之后的结点后移；

在i位置插入e；

长度++；

}

Int contact(Dis p1,Dis p2,Dis &p3)

{ 把p1中所有传入p3;

While(p2数据未读取完)

{ if p2中数据和p3重复

不存储；

Else

P2的此个数据传入p3;

}

更新P3长度;

}

void ShortestPath\_FLOYD1(Mgraph G, PathMatrix &Path,

DistanceMatrix &D)

{ for (v = 0; v < n; v++)

for (w = 0; w < n; w++) {

D[v][w] = G.arcs[v][w];

path[v][w] = φ;

if (D[v][w] < INFINITY)

path[v][w] = [v]+[w];

}

for (k = 0; k < n; k++)

for (v = 0; v < n; v++)

for (w = 0; w < n; w++)

if (D[v][k] + D[k][w] < D[v][w]) {

D[v][w] = D[v][k] + D[k][w];

Path[v][w] = Path[v][k]+Path[k][w];

}

}

* **源代码：**

（实验04.c）

#include<stdio.h>

#include<stdlib.h>

#define INFINITY 1000//最大值无穷

#define MAX\_VERTEX\_NUM 20//最大顶点个数

#define ERROR 0

#define OK 1

typedef enum{DG,DN,UDG,UDN}GraphKind; //{有向图，有向网，无向图，无向网}

typedef struct{

char vexs[MAX\_VERTEX\_NUM];//顶点向量

int arcs[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];//邻接矩阵

int vexnum,arcnum;//图的当前顶点数和弧数

GraphKind kind;

}MGraph,\*MGPtr;

typedef struct{

int stop[MAX\_VERTEX\_NUM+1];

int ListLen;

}DisMatrix,\*Dis;//线性表存储路径

int insert(Dis p,int i,int e)//线性表中插入结点

{ int j;

if((i<0)||(i>p->ListLen))//i不符合要求

return ERROR;

for(j=p->ListLen-1;j>=i-1;j--)

{ p->stop[j+1]=p->stop[j];}//后面结点后移

p->stop[i]=e;//插入

p->ListLen++;//长度加一

return OK;

}

int contact(Dis p1,Dis p2,Dis p3)//路径更新

{ int i=0,j;

int flag;

int p1count=0,p2count=0;

for(p1count=0;p1count<p1->ListLen;p1count++)

{ p3->stop[p1count]=p1->stop[p1count];

i++;

}//把路径1（P[v][k]）先存入要更新的路径

while(p2count<p2->ListLen)

{ flag=0;

for(j=0;j<p1count;j++)

{ if(p3->stop[j]==p2->stop[p2count])

{flag=1;

}//如果路径2(p[k][w])有与路径1重复的部分，则不重复存储

}

if(flag==0)

{

p3->stop[i]=p2->stop[p2count];

i++;

}//存入路径2不重复部分

p2count++;

}

p3->ListLen=i;//更新路径线性表长度

}

void FLOYD(MGPtr G1,int D1 [][MAX\_VERTEX\_NUM],Dis P1[][MAX\_VERTEX\_NUM])

{ int v,w,k;

int count;

for(v=0;v<G1->vexnum;v++)

for(w=0;w<G1->vexnum;w++)

{ count=0;

D1[v][w]=G1->arcs[v][w];//距离长度开始为权值

P1[v][w]=(Dis)malloc(sizeof(DisMatrix));

P1[v][w]->stop[0]=-1;//代表无法联通

P1[v][w]->ListLen=1;//开始忘了设置，到后面更新路径时报错

//

if(D1[v][w]<INFINITY)

{P1[v][w]->stop[0]=v;

P1[v][w]->stop[1]=w;

P1[v][w]->ListLen=2;

}//有权值的地方路径预设

}

for(k=0;k<G1->vexnum;k++)

for(v=0;v<G1->vexnum;v++)

for(w=0;w<G1->vexnum;w++)

if(D1[v][k]+D1[k][w]<D1[v][w])//发现更短距离

{ D1[v][w]=D1[v][k]+D1[k][w];//距离长度更新

//

if(P1[v][w]->stop[0]==-1)//如果开始为无穷，则更新路径

{ P1[v][w]->stop[0]=v;

P1[v][w]->stop[1]=w;

P1[v][w]->ListLen=2;

insert(P1[v][w],1,k);

}

else{

contact(P1[v][k],P1[k][w],P1[v][w]);//路径更新

}

//

//P1[v][w]->stop=P1[v][k]+P1[k][w];

}

}

void Printf(MGPtr G1,int D2[][MAX\_VERTEX\_NUM],Dis P2[][MAX\_VERTEX\_NUM])//打印

{ int v,w,k;

for(v=0;v<G1->vexnum;v++)

for(w=0;w<G1->vexnum;w++)

{if(v!=w&&D2[v][w]<INFINITY)//不打印自己到自己以及无法联通的

{

printf("%c到%c的最短路径是",65+v,65+w);//转换为字母

for(k=0;k<=P2[v][w]->ListLen-1;k++)

{printf(" %c ",P2[v][w]->stop[k]+65);

}

printf("长度是%d",D2[v][w]);

printf("\n");

}

}

}

main()

{ MGPtr G;

G=(MGPtr)malloc(sizeof(MGraph));

Dis Path[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];

int D[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];

printf("请输入图的顶点数：");

scanf("%d",&G->vexnum);

printf("请输入图的弧数：");

scanf("%d",&G->arcnum);

printf("请输入邻接矩阵：\n");

int i,j;

//读取邻接矩阵

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

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

{scanf("%d",&G->arcs[i][j]);}

FLOYD(G,D,Path);

Printf(G,D,Path);

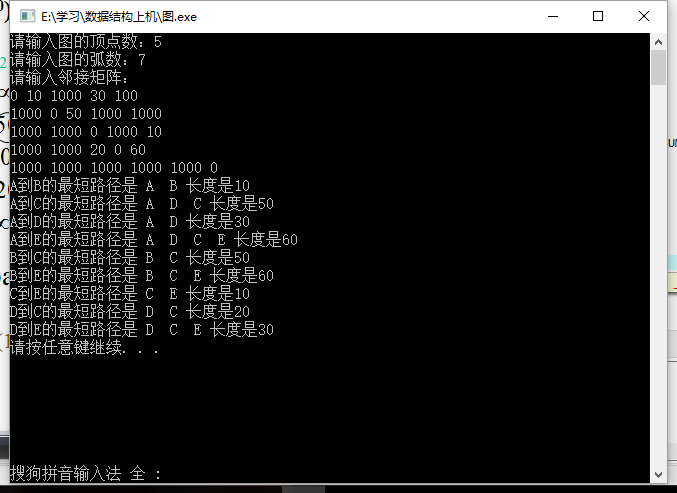
system("pause");

return 0;

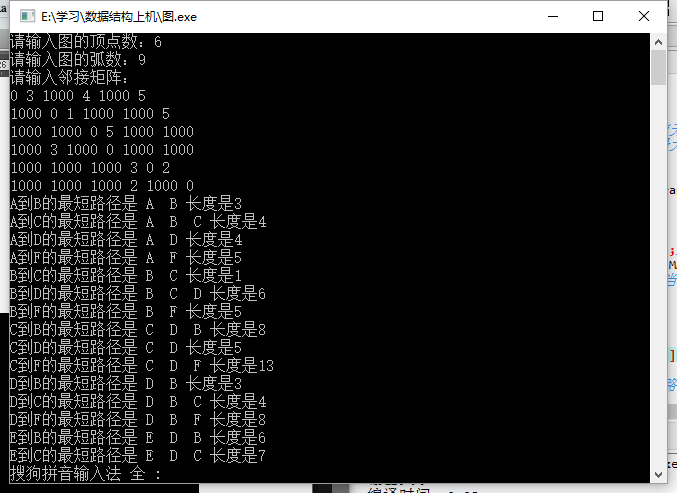
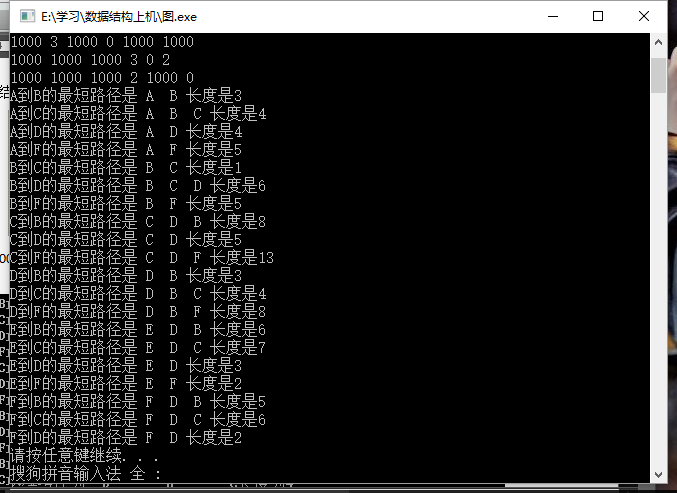
}

* **测试结果：**

1.（PPT上例子）



2.网上找的例子



* **用户使用说明：**

运行环境DevC++.

按照要求输入。输入形式为邻接矩阵。权值为无穷的用1000或以上数字表示。自己和自己的权值为0。

* **心得体会：**

按照老师PPT上给的算法写的，所以开始时求距离的D矩阵时正确的，但是求路径Path有问题。

把Path改成线性表后，存入时没有分配内存，所以只能读取第一行，所以就每次都要分配空间。

之后发现开始距离矩阵为无穷的结点无法更新Path，后来发现没有初设ListLen，所以会出现停止运行。（但是在U盘里隔离运行可以，但是路径输出为空白）

改过来后，发现输出不对。

对算法理解有误，路径更新并不是向原路径加一个结点，而是把P1[v][k]和P1[k][w]路径合并。

合并时还要注意重复部分不要重复存储。