班级—— 2014211314

姓名—— 袁振宇

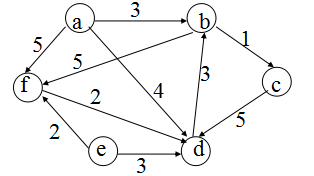
学号—— 2014211529

分工情况——个人独立完成

完成日期——2015.12.20

**1.问题描述：**

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。



1. **明确规定：**
2. 输入形式：首先输入顶点数和弧数，再按顺序输入顶点，最后输入各条弧的弧头弧尾和相应权值；
3. 输出形式：输出各顶点间的最短路径；
4. 程序所能达到的功能：以邻接矩阵存储图，并用Floyd算法求出各顶点间的最短路径。
5. **算法思想：**

实现Floyd算法求最短路径。

1. **设计描述：**
2. 图结构体；
3. 主函数调用int LocateVex(MGraph G, string vex),返回该顶点的序号，调用ShortestPath\_Floyd(MGraph G, int D[][MAX\_VERTEX\_NUM], string P[][MAX\_VERTEX\_NUM])函数实现Floyd算法求出最短路径；
4. 算法描述：

ShortestPath\_Floyd(MGraph G, int D[][MAX\_VERTEX\_NUM], string P[][MAX\_VERTEX\_NUM])

{

for(v = 1; v <= G.vexnum; v++)

for(w = 1; w <= G.vexnum; w++)

{

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

P[v][w] = '\*';

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

P[v][w] = G.vexs[v] + G.vexs[w];

}

for(k = 1; k <=G.vexnum; k++)

for(v = 1; v <= G.vexnum; v++)

for(w = 1; w <= G.vexnum; w++)

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

{

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

string S = P[k][w];

S = S.substr(1,S.length()-1);

P[v][w] = P[v][k] + S;

}

}

LocateVex(MGraph G, string vex)

{

for( i=1; i <=G.vexnum; i++)

{

if( vex == G.vexs[i])

return i;

}

return 0;

}

1. **源程序**

#include<iostream>

#include<cstdlib>

using namespace std;

#define INFINITY 1000 //最大值

#define MAX\_VERTEX\_NUM 100 //最大顶点数

#define TRUE 1

#define FALSE 0

typedef enum{ //四种图类型

DG, DN, UDC, UDN

}GraphKind;

typedef struct MGraph{

string vexs[MAX\_VERTEX\_NUM]; //顶点向量

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

int vexnum, arcnum; //图当前顶点数和弧数

GraphKind kind; //图的种类标志

}MGraph;

void ShortestPath\_Floyd(MGraph G, int D[][MAX\_VERTEX\_NUM], string P[][MAX\_VERTEX\_NUM])

{

int v, w, k;

for(v = 1; v <= G.vexnum; v++)

for(w = 1; w <= G.vexnum; w++)

{

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

P[v][w] = '\*';

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

P[v][w] = G.vexs[v] + G.vexs[w];

}

for(k = 1; k <=G.vexnum; k++)

for(v = 1; v <= G.vexnum; v++)

for(w = 1; w <= G.vexnum; w++)

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

{

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

string S = P[k][w];

S = S.substr(1,S.length()-1);

P[v][w] = P[v][k] + S;

}

}

int LocateVex(MGraph G, string vex)

{

for(int i=1; i <=G.vexnum; i++)

{

if( vex == G.vexs[i])

return i;

}

return 0;

}

int main()

{

MGraph G;

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

string P[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM], S;

int v, w, k, weight;

string V1, V2;

cout << "请输入图的顶点数和弧数:" << endl;

cin >> G.vexnum >> G.arcnum;

if(G.vexnum >= MAX\_VERTEX\_NUM)

{

cout << "顶点过多(输入100以下个顶点),请重新输入顶点数和弧数:";

cin >> G.vexnum >> G.arcnum;

}

cout << "请按顶点序号顺序输入顶点:" << endl;

for(v = 1; v <= G.vexnum; v++)

cin >> G.vexs[v];

for(v = 1; v <= G.vexnum; v++) //初始化邻接矩阵

for(w = 1; w <= G.vexnum; w++)

{

if(v == w)

G.arcs[v][w] = 0;

else

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

}

cout << "请输入各弧尾弧头及其路径的权值:" << endl;

for(k = 1; k <= G.arcnum; k++) //输入相关联的顶点及其权值

{

cin >> V1 >> V2 >> weight;

v = LocateVex(G, V1);

w = LocateVex(G, V2);

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

}

ShortestPath\_Floyd(G, D, P);

cout << "\n每一对顶点的最短路径是:" << endl;

for(v = 1; v <= G.vexnum; v++)

for(w = 1; w <= G.vexnum; w++)

if(v != w && D[v][w] != INFINITY)

{

cout << G.vexs[v] << G.vexs[w] << "间的最短路径为: " << P[v][w]

<< "\t\t\t" << "长度为: " << D[v][w] << endl;

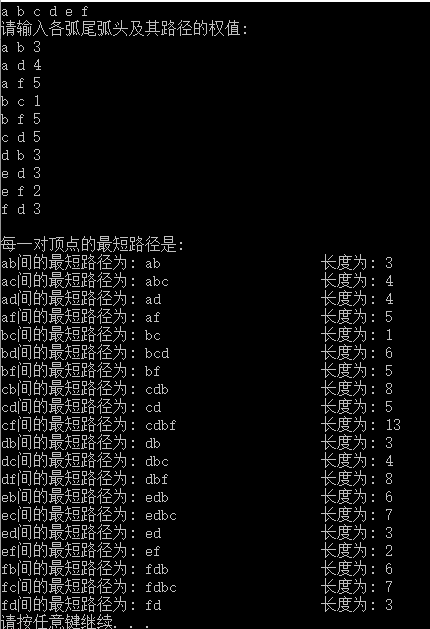
}

system("pause");

return 0;

}

1. **测试结果**



1. **用户使用说明：**

打开DEV-C++,运行程序，输入时一定要按顶点序号顺序输入，其余按程序提示进行操作。

1. **心得体会：**

通过本次试验，我深入体会了Floyd算法求最短路径的方法，在初次实现时，我采用的是输入邻接矩阵的方法，这样输入时非常繁琐，后来进行了改进，直接输入每条弧的弧头弧尾和相应权值，这样操作比较简单，而且易于用户理解。