**Lab4 图的最短路径**

**唐凯成 PB16001695**

**一.实验要求**

1、创建图的数据结构

·从邻接矩阵读入创建图的数据结构

·运用Dijkstra算法求解两个点之间的最短路径

·使用dot库以输出对应文件

1. 实现压缩程序的GUI界面

·运用Qt（MacOS）来实现GUI界面的工程构建

·通过维护QMainWindow类来完成GUI界面与源程序的链接

**二.实验平台**

Apple LLVM version 9.1.0 (clang-902.0.39.2)

Visual Studio Code v1.29.0

Qt 5.8.0

**三.实验内容**

1、总体设计

根据读入的设置，使用了邻接矩阵来存放图的数据结构，并使用Dijkstra算法来实现了两个点之间的最短路径。

2、图的创建和读入

由于本次的读入文件中已直接设置好读入的要求，因此直接以读入文件中的矩阵大小n来创建一个n\*n的数组用于存放邻接矩阵即可。另外根据读入的格式，还需将矩阵的下半三角又上半三教复制得到。

3、Dijkstra算法求解最短路径

使用Dijkstra算法时考虑到不仅要求解出最短路径的大小，还要能够输出最短路径，故使用多维护一个二维数组，用来记录从起点到每个顶点的路径。

实现时还需要创建一个一维数组，用来标记当前已经到达的点，若到达了该点则将该点的标记位改为true，否则标记位为false。

每轮循环均实现以下操作，遍历到目前已经到达点的周围还没有到达的点，计算起点到该点的最小值，记录本轮中的最小值和准备新到达的点。完成寻找后，将该点在一位数组内标为true，将该点距离起点的距离改为该点+该点前一点到原点的距离，该点的路径为前一点的路径复制过来+加入该点自己。

如此循环，当终点已经为true时即结束循环，输出对应路径即可。

4、GUI界面的实现

由Qt内已经构建好的QMainWindow类直接调用窗口设置，并链接按键、路径到对应的函数即可实现简单的GUI界面，具体的实现界面见实验结果中。

**四.实验代码**

1、Dijkstra.h

#include<stdio.h>

#include<stdlib.h>

#include<iostream>

#include<iomanip>

#include<fstream>

#include<string>

#ifndef DIJKSTRA\_H

#define DIJKSTRA\_H

#define MAX\_VERTEX\_NUM 20

#define INFINITY 1000

typedef struct

{

char vexs[MAX\_VERTEX\_NUM];

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

int vexnum,arcnum;

}MGraph;

bool dijkstra(const char \*ifname);

#endif // DIJKSTRA\_H

2、Dijkstra.cpp

#include"dijkstra.h"

using namespace std;

bool dijkstra(const char \*ifname)

{

int i,j,n,k,start,end,tmp;

long min;

MGraph map;

int P[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM]; // Path Martix

int D[MAX\_VERTEX\_NUM]; // Short Path Table

bool final[MAX\_VERTEX\_NUM]; // Point Color Table

//读入节点信息

ifstream in(ifname);

in>>map.vexnum>>start>>end;

while(!in.eof())

{

for(i=0;i<=map.vexnum-1;i++)

for(j=0;j<=map.vexnum-1;j++)

in>>map.arc[i][j];

}

in.close();

//完善矩阵信息

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

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

map.arc[i][j]=map.arc[j][i];

for(i=0;i<map.vexnum;i++)

for(j=0;j<map.vexnum;j++)

if(map.arc[i][j]<=0)

map.arc[i][j]=INFINITY;

//初始化

for(i=0;i<map.vexnum;i++)

{

final[i]=false;

D[i]=map.arc[start][i];

for(j=0;j<map.vexnum;j++)

P[i][j]=-1;

P[i][start]=0;

}

D[start]=0;

final[start]=true;

//dijkstra loop

for(i=1;i<map.vexnum;i++) // n-1论算法执行

{

min=INFINITY;

for(j=0;j<map.vexnum;j++)

if(!final[j])

if(D[j]<min)

{

n=j;

min=D[j];

}

final[n]=true;

//更改路径信息

int max=0;

for(k=0;k<map.vexnum;k++)

{

if(P[n][k]>=max)

{

tmp=k;

max=P[n][k];

}

}

if(tmp!=n)

P[n][n]=max+1;

for(j=0;j<map.vexnum;j++)

{

if(!final[j]&&(min+map.arc[n][j]<D[j]))

{

D[j]=min+map.arc[n][j];

for(k=0;k<map.vexnum;k++)

if(k!=j)

P[j][k]=P[n][k];

int max;

max=0;

for(k=0;k<map.vexnum;k++)

{

if(P[j][k]>=max&&(P[j][k]!=map.vexnum-1))

max=P[j][k];

}

P[j][j]=max+1;

}

}

}

// 输出答案，图形化界面中无需

/\*printf("Min=%d\nPath:",D[end]);

for(i=0;i<map.vexnum;i++)

for(j=0;j<map.vexnum;j++)

if(P[end][j]==i)

printf("%d ",j);\*/

// 输出dot格式的图像描述文件

ofstream out;

out.open("/Users/tkc/Desktop/test/dijkstra.dot");

out<<"graph first2"<<endl;

out<<"{"<<endl;

out<<"node [shape = circle];"<<endl;

out<<"label=\"MIN:"<<D[end]<<"\t";

out<<"PATH:";

for(i=0;i<map.vexnum;i++)

for(j=0;j<map.vexnum;j++)

if(P[end][j]==i)

out<<j<<"\t";

out<<"\""<<";";

for(i=0;i<map.vexnum;i++)

{

if(P[end][i]>=0)

out<<i<<"[color=red]"<<endl;

else

out<<i<<endl;

}

for(i=0;i<map.vexnum-1;i++)

for(j=i+1;j<map.vexnum;j++)

if(map.arc[i][j]<INFINITY)

out<<i<<"--"<<j<<"[label="<<map.arc[i][j]<<"]"<<";"<<endl;

out<<"}"<<endl;

out.close();

system("./a.sh");

//system("export > ex.txt");

return true;

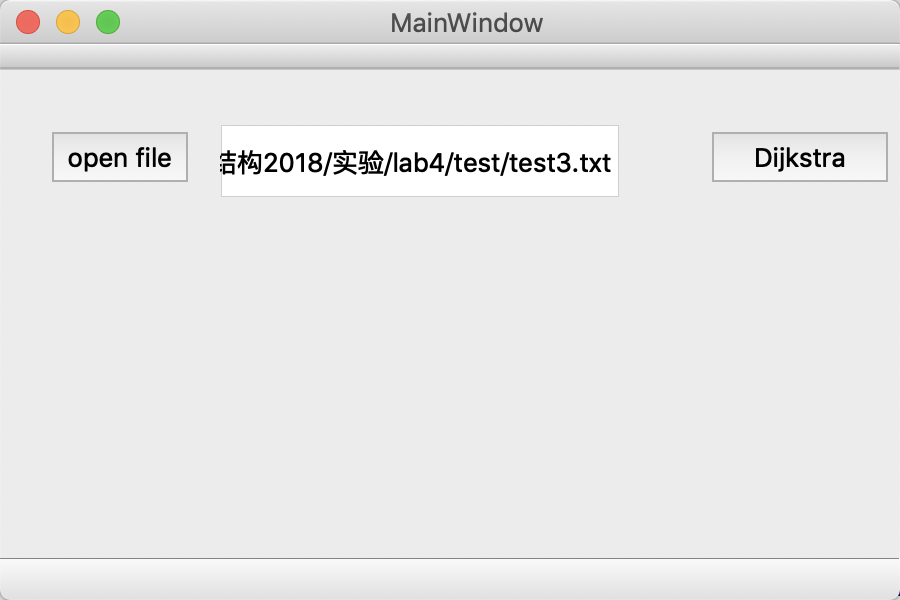
}

**五.实验结果**

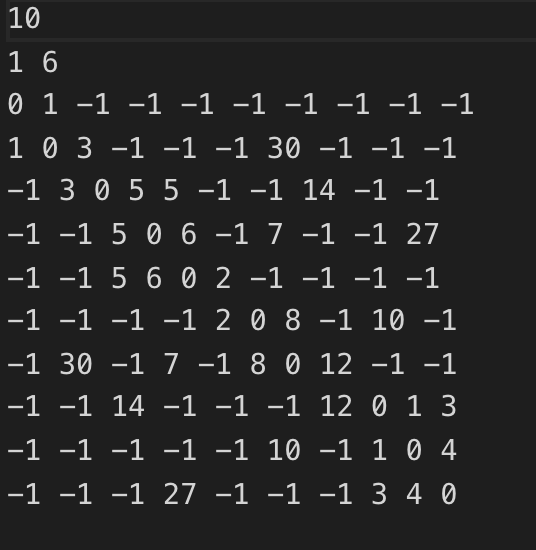
1、GUI界面

由Dijkstra工程，在Qt下构建生成的GUI界面为：

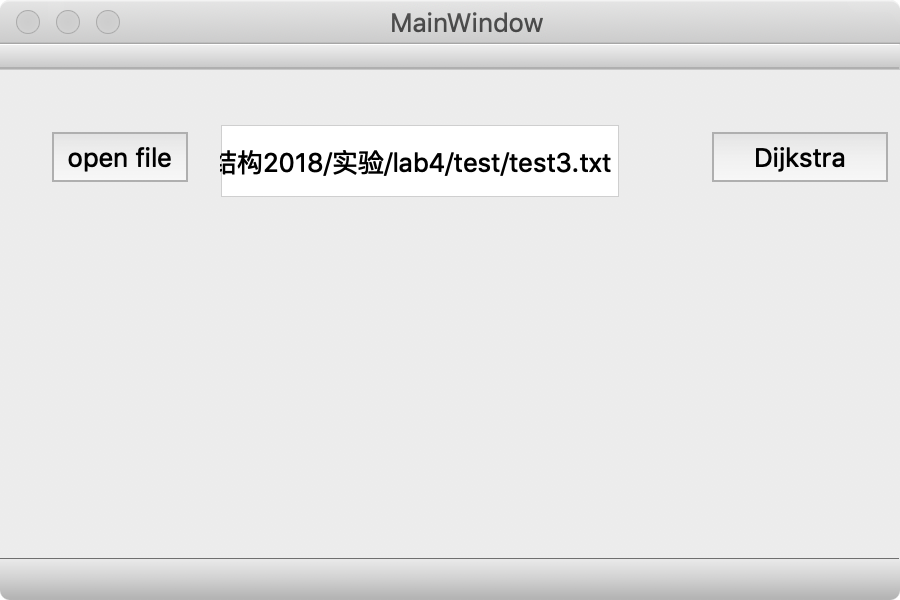
（1）进入界面：

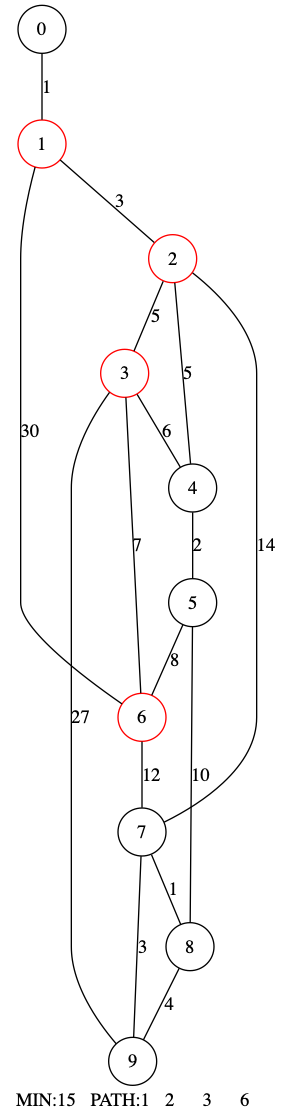


（2）测试文件：



（3）生成结果：





2、实验结果分析

由上述生成的图，验证可得路径为正确最短路径，已在图片中通过红色标出，如上图。

**五.实验总结**

本次实验中除了使用之前的Qt之外，还通过系统dot库来实现绘图指令，在实际处理中发现Qt内直接调用system时没有连接到dot的PATH，因此在脚本文件中另外加入了dot库的PATH后才可以运行。总的来说在练习算法之余也体验了图形化界面的实现，很有收获。