|  |  |
| --- | --- |
| 总评成 绩 |  |

算法设计说明书

题目分类（难）：1

学生姓名：秦嘉豪 学号：1807004120

1. 软件使用手册
2. 开发环境：包括软件软、硬件说明及开发语言

【编程软件】

VS 2017、Easy-x图形库。

【开发语言】 C++

1. 使用软件所需要的环境和插件安装说明：

【软件使用说明】

使用时电脑应安装好VS系列中任意一个编译器，并支持Easy-X编译运行，同时设置好图片资源的路径，即可进行应用。

1. 运行程序的文件名和软件使用方法简单介绍：

【文件名】：闭包问题

【使用方法】：

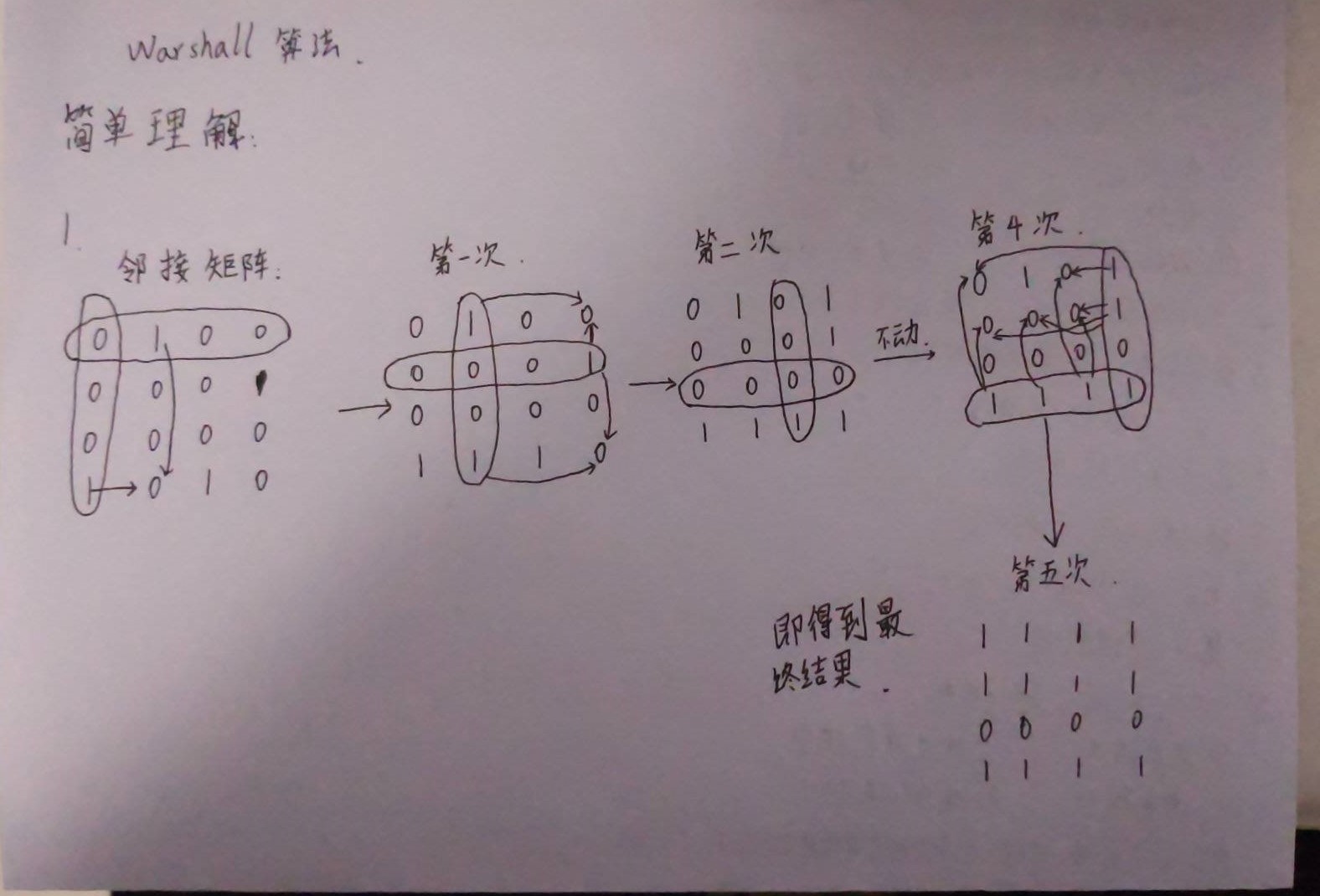
* ：打开编译器，设置好图片资源的路径，检查程序。
* ：运行程序，在弹出的输入框中依次输入顶点个数、邻接矩阵，程序框中会同步显示输入的数值，如果输入有误的话关闭重新输入。
* ：输入完成后，即可显示每次闭包的结果以及最终的结果以供查看。

1. 算法原理介绍

【核心算法介绍】：

本次算法大作业题目的核心算法为——Warshall算法

1. 【分析过程】  
    我们的目的是通过 Warshall算法求有向图的传递闭包。那么我们需要从有向图中得到该图的邻接矩阵，然后再通过邻接矩阵求解传递闭包。  
    该算法有一个定理，Rij=Rik∪Rkj，其中ijk均是从0~n，n为顶点数。简单说就是寻找十字，如果r[i][k]与r[k][j]均为1或者r[i][j]原本就为1，就把r[i][j]赋值为1，这样使用三重循环最终即可得到传递闭包。



1. 【输入/输出】  
   程序输入为：首先输入图的顶点个数，然后再依次输入邻接矩阵，在界面上会相应显示出输入的内容，如果输入有误关闭重新输入即可。  
   程序输出为：程序再输入完毕后，会依次输出每次变换的矩阵，最终会输出一个最终结果，即所求的传递闭包。
2. 【Floyd-Warshall算法原理】  
   Warshall算法的原理是动态规划。设**D**i,j,k 为从i到j的只以(1~k)集合中的节点的最短路径的长度。1. 若最短路径经过点k，则**D**i,j,k = **D**i,k,k-1 + **D**k,j,k-1; 2.若最短路径不经过点k，则**D**i,j,k = **D**i,j,k-1。**D**i,j,k = min(**D**i,j,k-1, **D**i,j,k-1 + **D**k,j,k-1)。
3. 【项目设计进度说明】  
   2020年03月31日 布置题目，并对题目进行分析；  
   2020年04月6日~04月10日 算法设计、编写代码；  
   2020年04月11日 对算法进行分析和测试；  
   2020年04月23日~04月25日 撰写算法设计说明书；  
   2020年04月29日 验收
4. 核心代码（按模块顺序，每一个模块先做简要功能说明及输入输出介绍，核心代码尽可能少）
5. 【输入模块】  
   输入模块使用InputBox函数弹出对话框输入数据，然后再转换为int型变量进行数据处理。

TCHAR s[3];

InputBox(s, 3, \_T("请输入图中节点数目："), \_T("节点输入"), \_T("00"), 0, 0, false);

num = \_ttoi(s);

InputBox(s, 3, \_T("请输入邻接矩阵："), \_T("输入"), \_T("00"), 0, 0, false);

r[i][j] = \_ttoi(s);

1. 【输出模块】  
   输出使用outtextxy函数，将每个过程的矩阵依次输出。

for (i = 1; i <= num; i++)

for (j = 1; j <= num; j++)

r[i][j] = temp[i][j];

outtextxy(x, y, "第");

TCHAR s[2];

\_stprintf(s, \_T("%d"), k);

outtextxy(x + 20, y, s);

outtextxy(x + 40, y, "次转换：");

for (i = 1; i <= num; i++) {

y += 15;

for (j = 1; j <= num; j++)

{

\_stprintf(s, \_T("%d"), r[i][j]);

outtextxy(x, y, s);

x += 20;

}

x = 0;

}

y += 40;

1. 【Warshall算法模块】  
   采用Warshall算法进行处理计算，使用三重for循环模拟推导过程，最终得到结果。

void Warshall(int num, int \*\*r)

{

int i, j, k;

int \*\*temp = (int \*\*)malloc(sizeof(int \*) \* (num + 1));

for (i = 0; i < num + 1; i++)

temp[i] = (int \*)malloc(sizeof(int) \* (num + 1));

y = 130;

for (k = 1; k <= num; k++) {//三重循环实现Warshall算法

for (i = 1; i <= num; i++)

for (j = 1; j <= num; j++)

temp[i][j] = (r[i][j]) || (r[i][k] & r[k][j]);

}

}

1. 【主函数模块】  
   主函数模块包含对窗口的绘制、图像的加载、字体的设置、函数的调用以及最终结果的输出。

int main()

{

int i, j, num;

initgraph(800, 650);

loadimage(&img, \_T("D:\\Program Files (x86)\\VS2017projects\\Navigation\_consultation\\Navigation\_consultation\\001.jpg"), 800, 650);

putimage(0, 0, &img);

gettextstyle(&f); // 获取当前字体设置

\_tcscpy(f.lfFaceName, \_T("黑体")); // 设置字体

setbkmode(TRANSPARENT);

settextcolor(BLACK);

settextstyle(&f); // 设置字体样式

TCHAR s[3];

InputBox(s, 3, \_T("请输入图中节点数目："), \_T("节点输入"), \_T("00"), 0, 0, false);

num = \_ttoi(s);

outtextxy(x, y, "顶点个数为：");

outtextxy(x + 100, y, s);

outtextxy(x, y + 20, "邻接矩阵：");

int \*\*r = (int \*\*)malloc(sizeof(int \*) \*(num + 1));

for (i = 0; i < num + 1; i++)

r[i] = (int \*)malloc(sizeof(int) \*(num + 1));

outtextxy(x, y + 20, "邻接矩阵：");

y = 40;

for (i = 1; i < num + 1; i++) {

for (j = 1; j < num + 1; j++)

{

InputBox(s, 3, \_T("请输入邻接矩阵："), \_T("输入"), \_T("00"), 0, 0, false);

r[i][j] = \_ttoi(s);

outtextxy(x, y, s);

x += 20;

}

y += 20;

x = 0;

}

Warshall(num, r);

y = 530;

outtextxy(x, y, "最终闭包：");

for (i = 1; i < num + 1; i++)

{

y += 15;

x = 0;

for (j = 1; j < num + 1; j++)

{

\_stprintf(s, \_T("%d"), r[i][j]);

outtextxy(x, y, s);

x += 20;

}

}

system("pause");

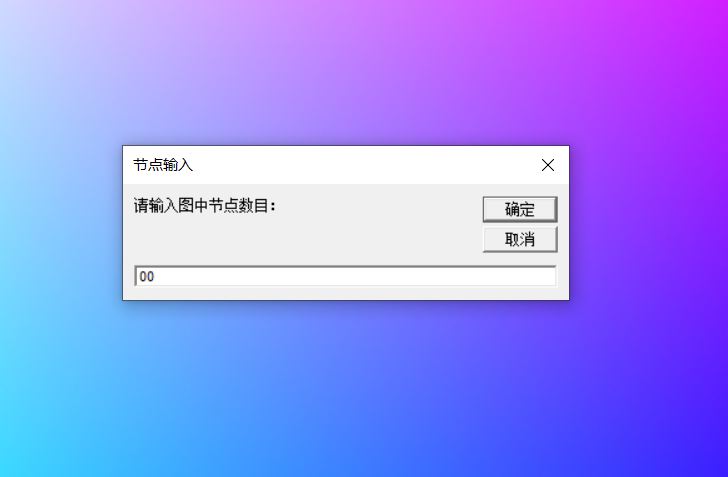
closegraph();

return 0;

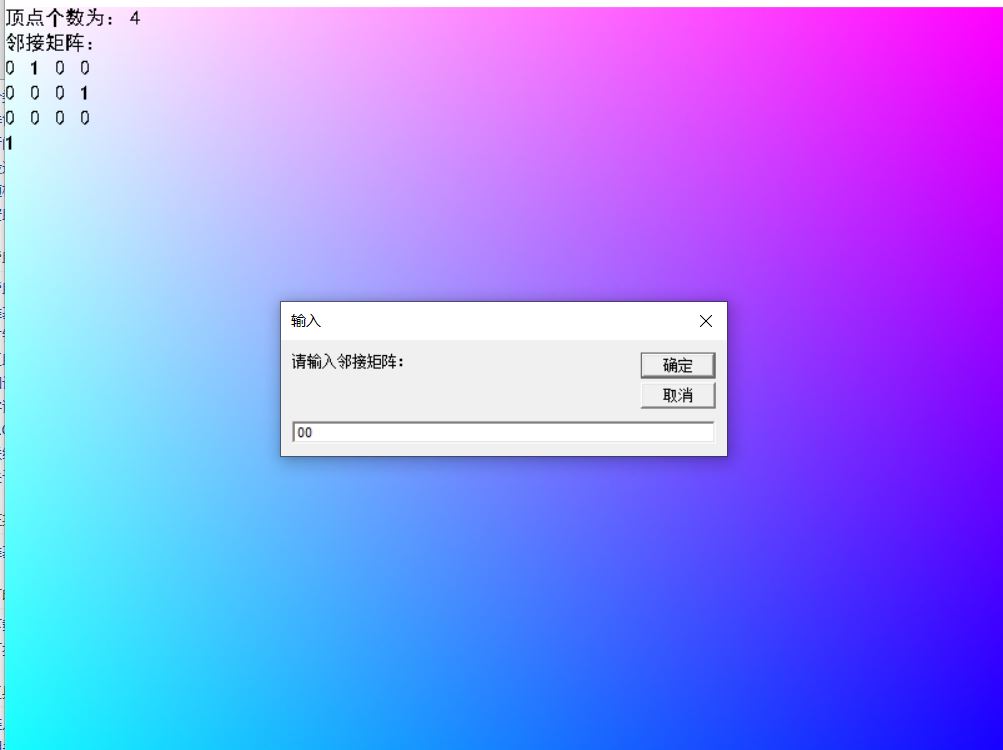
}

1. 运行结果：(必须有截图，否则没分)及对结果的说明

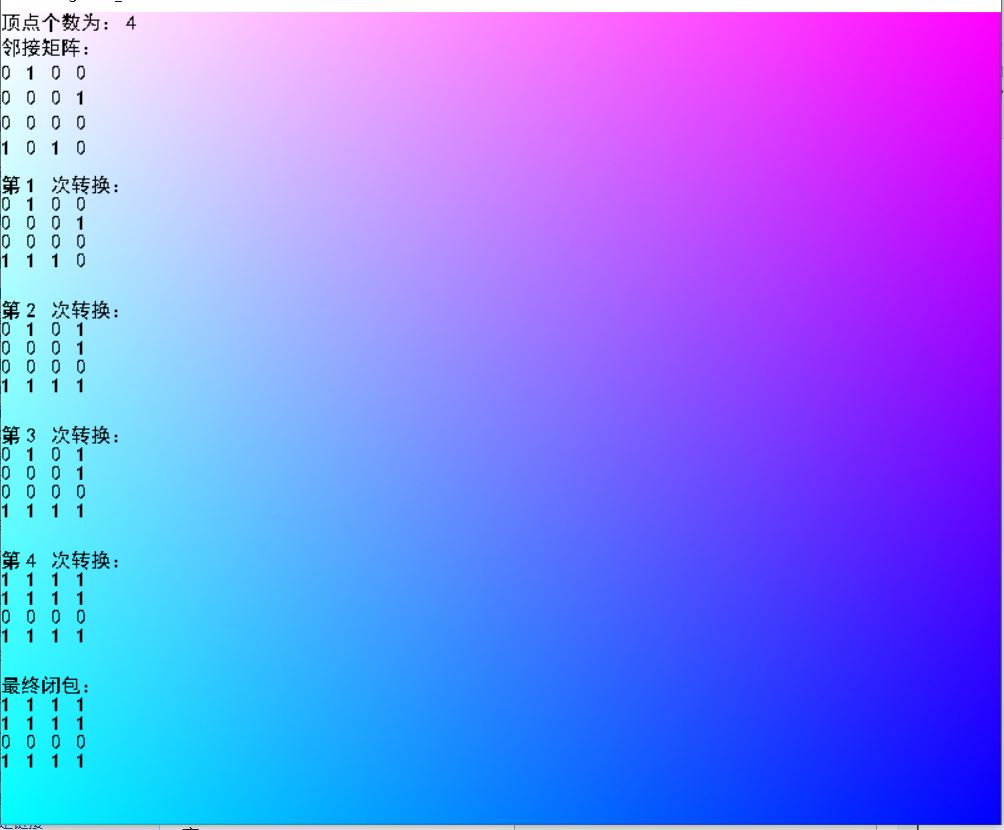
* 【输入节点个数】  
  向该提示框中输入节点数目的个数，点击回车。



* 【输入邻接矩阵】  
  向该提示框中输入邻接矩阵的每个的值，每次只能输入一个值，且每次输入都会在界面中显示，如果输入错误需要关闭重新输入。



* 【结果示意图】  
  输入完毕后该程序会自动计算并以矩阵的形式输出每一次转换的结果，最终得到所求的结果，即为传递闭包。



1. 心得体会

通过本次算法设计深刻的理解了算法在解决实际问题中的重要性，并且深刻理解了Warshall算法的具体计算过程，并且学会了OpenCV的基本使用及可视化显示。在完成的过程中虽然遇到了一些问题，但是在学习的过程中都一一解决了，还锻炼了自身的代码能力。同时也发现了将这些知识运用起来才会更加有趣，虽然在此次设计中还存在很多不足，但是看着这些代码，程序能够按照我们的想法去解决问题，给我的感受还是很不错的。

1. 参考文献

[1] 严蔚敏，吴伟民﹒数据结构（C语言版）﹒北京：清华大学出版.1997

[2] 师智斌﹒算法分析与设计及案例教程﹒北京：清华大学出版社.2014

[3] 孙博文﹒分形算法与程序设计 —— 用Viasual C++实现﹒北京：科学出版社，2014