数据结构课程大作业

报告

姓 名：王语晨

学 号：2022104842

专 业：计算机科学与技术

欧拉图助手

王语晨

1. 大作业的功能及特点

欧拉图助手功能在于，能准确接收用户输入的图，帮助判断输入的图是否为欧拉图，如果是则给出欧拉通路/回路。用户可以选择使用提供的可视化操作窗口，用户可以通过鼠标点击进行图的创建，程序可以绘制通路，操作可以回退，有简单的操作界面。欧拉图助手可以完成建立图到给出欧拉通路的全过程，帮助用户解决关于欧拉图的问题。

其特点在于，可以准确完成欧拉图的判断和通路的寻找并展示。

最大的创新在于加入了可视化窗口，用户可以利用鼠标交互，突破以往只能键盘输入的局限，加入按钮等新的数据结构，利用额外的第三方库，进行图形界面的绘制，可以根据用户鼠标点击位置绘制图的结点，可以根据鼠标左键点击一步一步生成通路，有较强的观赏性，更加直观，使对欧拉图不熟悉的用户可以不用面对冰冷的结点序列，可以体会通路过程。复位按钮和结束按钮，可使可视化过程按用户需求重复，适用于画错节点，没看懂，误触情况，简单的界面，用户不会有使用的困难。

1. 大作业需求分析

该程序旨在帮助用户解决一笔画问题，只要有图程序就可以给出判断和找到的通路，并根据用户需要提供可视化窗口进行鼠标交互操作。

用户对象为一笔画游戏有通关困难希望有通关“秘籍”的玩家、学习离散数学图论欧拉图相关知识有困难或不想自己苦苦找通路的同学、出完考试题进行简单验证的老师、上课教学进行演示的老师等等。

基于以上，我所设计的欧拉图助手需要有以下功能：

1、可以接收用户输入的图，并保存。

2、可以判断图是否为欧拉图或半欧拉图。

3、给出可以演示的可视化界面，并具有完善的基本功能，可回到主程序中重新建图等。

4、具有菜单化界面，可不断接受新的图，有一定健壮性。

1. 大作业的设计

**1、主要数据结构**

图（使用邻接矩阵储存）：

#define MAXSIZE 20

typedef enum{DG,UDG} GraphKind;

//有向图，无向图（因为一笔画问题不关心权重）

typedef struct ArcCell{

VRType adj; //顶点关系，若i->j连通则相应adj>0，值为平行边数

InfoType \*info; //顶点信息

}ArcCell,AdjMatix[MAXSIZE][MAXSIZE];//邻接矩阵

typedef struct{

VertexType vex[MAXSIZE];//结点

AdjMatrix arcs; //邻接矩阵

int vexnum,arcnum;//结点数，边数

GraphKind kind;//图的类型

}MGraph;

栈（顺序表）：

typedef struct{

SElemType \*base;

SElemType \*top;

int stacksize;

}SqStack;

入栈

Status Push(SqStack &S,SElemType e){

//栈顶插入e

if(S.top-S.base>=S.stacksize)

{//栈满追加空间……}

\*S.top++=e;

return OK;

}

出栈

Status Pop(SqStack &S,SElemType &e){

栈顶出栈，给e,返回OK，否则返回ERROR

if(S.top==S.base)

return ERROR;

e=\*--S.top;

return OK;

}

可视化界面：

鼠标消息（库中自带）{

坐标x,y;

鼠标信息（左键点击，按住，右键等等）;

}

自建：

按钮

typedef struct Button {

int x, y, width, height;//坐标&长高

COLORREF color;//按钮颜色

char context[500];//按钮文本

}Button,\*PtrButton;

文本框

typedef struct Text {

int x, y;//坐标

int width;//长

int height;//高

char context[500] ;//文本

}Text, \* PtrText;

**2、详细设计的算法**

**核心算法——查找欧拉路径（DFS改版，原始DFS也提供也可，但效率不高：**遍历所有边，每遍历一个边则删除该边继续遍历，如果中间过程还没有遍历所有边就无法继续遍历了，则往前回溯继续遍历。该算法时间复杂度是指数级别。**）：**

**Hierhoizer算法：**

**意思是相当于找很多环的并集**

1. 选择任一顶点为起点（半欧拉图需要选择两个度数为奇数的结点中的一个），遍历所有相邻边。
2. 深度搜索，访问相邻顶点。将经过的边都删除。
3. 如果当前顶点没有相邻边，则将顶点入栈。
4. 栈中的顶点倒序输出，就是从起点出发的欧拉迹、欧拉环游/回路。

Stack Hierhoizer (int use[N][N], Graph G, Stack S, int x)

{

/\*use是接收用的邻接矩阵，避免改变G中的邻接矩阵，G是已有的图，S是路径，x是路径起点\*/

//由于在此之前判断是否欧拉图，确保一定有通路，所以一定能找到。

int i;

for (i = FirstNod(use, G, x); i >= 0; i = NextNod(use, G, x, i))//遍历x的邻结点

{

use[x][i]--;//删除这条边

if (G.Gkind)//根据图的类型灵活

use[i][x]--;

S = Hierhoizer (use, G, S, i);//递归

}

S = PushStack(S, x);//把x放入路径（得到的是路径逆序）

return S;

}

/\*第一个邻结点\*/

int FirstNod(int use[N][N], Graph G, int x) {

int i;

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

if (use[x][i])

return i;

return -1;

}

/\*找以x为起点终点为i下一个的邻结点\*/

int NextNod(int use[N][N], Graph G, int x, int i) {

int j;

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

if (use[x][j])

return j;

return -1;

}

菜单switch

第一次只能选建图，第二次（建图成功后）只能选计算路径，之后才能选可视化。

计算路径中包含判断是否是欧拉图/半欧拉图

int JudgeEulerD(Graph G, int\* num) {

/\*有向图判断是否有欧拉通路\*/

//若欧拉图\*num为0，若半欧拉图（起点终点不同）\*num为2

int start, end, deg, i, j, flagout = 0, flagin = 0;

\*num = 0;//初始化非规范结点0个

start = 0;///这里初始化起点为0

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

{

deg = 0;

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

{

deg = deg + G.GraphElem[i][j] - G.GraphElem[j][i];

//入度+出度和是否为0

}

if (deg != 0)

{

if (deg == 1)//入度多一

{

start = i;//起点变成入度多的这个

flagout++;

}

else if (deg == -1) //出度多一

{

end = i;

flagin++;

}

(\*num)++;

}

}

if (flagout == 1 && flagin == 1)//充分条件

\*num = 2;

return start;

}

/\*无向图判断是否有欧拉通路\*/

int JudgeEulerUD(Graph G, int\* num) {

//\*num为奇数度数顶点个数

int start, deg, i, j;

\*num = 0;

start = 0;///这里初始化为0

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

{

deg = 0;

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

{

deg += G.GraphElem[i][j];

}

if (deg % 2 == 1)///奇度顶点

{

start = i;

(\*num)++;

}

}

return start;

}

主判断+找通路

if (G.Gkind)

start = JudgeEulerUD(G, num);

else

start = JudgeEulerD(G, num);

if (\*num == 0 || \*num == 2)

{

S.length = 0;

S.top = 0;

S = Hierhoizer(G.GraphElem, G, S, start);

if (\*num == 2)

printf("欧拉通路为：");

else

printf("欧拉回路为：");

可视化部分：

添加easyx第三方库，官网下载，很顺利。

添加头文件#include<graphics.h>

设置结点文本

PtrText SetText(PtrText text, ExMessage m, int i) {

//鼠标消息m，节点编号i

char chs[3];

text = (PtrText)malloc(sizeof(Text));

text->x = m.x;//文本位置起始在鼠标点击的位置上

text->y = m.y;

text->width = 5;

text->height = 5;

strcpy(text->context, "V\0");//显示V0/V1…

sprintf(chs, "%d", i);//把字符转成int

strcat(text->context, chs);

return text;

}

创建按钮

PtrButton createButton(int x,int y,int width,int height,char\* str) {

PtrButton button = (PtrButton)malloc(sizeof(Button));

assert(button);

button->x = x;

button->y = y;

button->width = width;

button->height = height;

button->color = RGB(204, 213, 240);

strcpy(button->context,str);

return button;

}

画按钮

void drawButton(PtrButton button) {

int xx, yy,textw,texth;

setfillcolor(button->color);//按钮填充色

fillrectangle(button->x,button->y,button->x+button->width,button->y+button->height);//画矩形

setbkmode(TRANSPARENT);//勾边

textw = textwidth(button->context);

texth = textheight(button->context);

xx = button->x + (button->width - textw) / 2;

yy = button->y + (button->height-texth) / 2;//文字居中

settextcolor(BLACK);//文字颜色

outtextxy(xx,yy,button->context);//打印文字

}

判断鼠标是否在按钮里

int inButton(PtrButton button, ExMessage m) {

//在里面就变色，不在就变回来

if (m.x > button->x && m.x<button->x + button->width

&& m.y>button->y && m.y < button->y + button->height)

{

button->color = RGB(236, 244, 255);

return 1;

}

button->color = RGB(204, 213, 240);

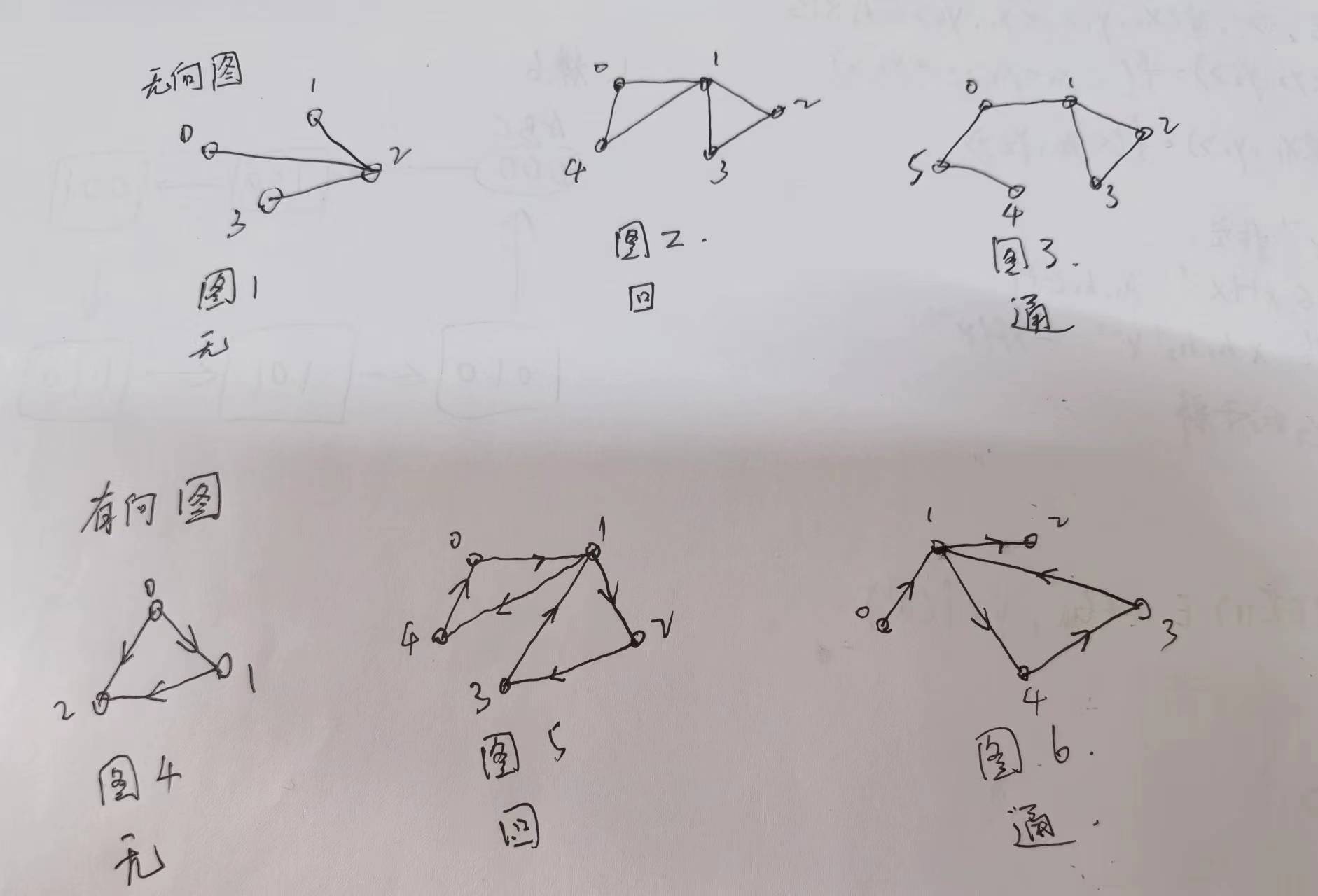
return 0;

}

1. 测试

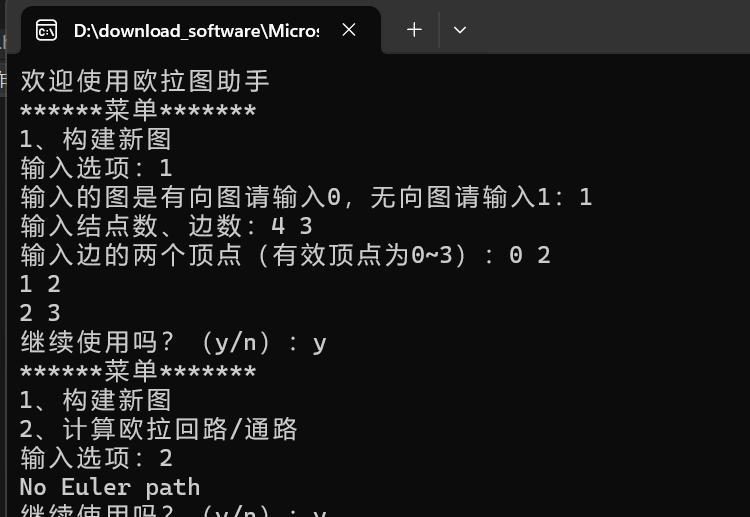
**1、测试用例**

测试用例分为有向图和无向图，每种图个测试三张，一张为无欧拉通路的，一张为有欧拉回路，一张为只有欧拉通路没有欧拉回路的。具体图例如下：

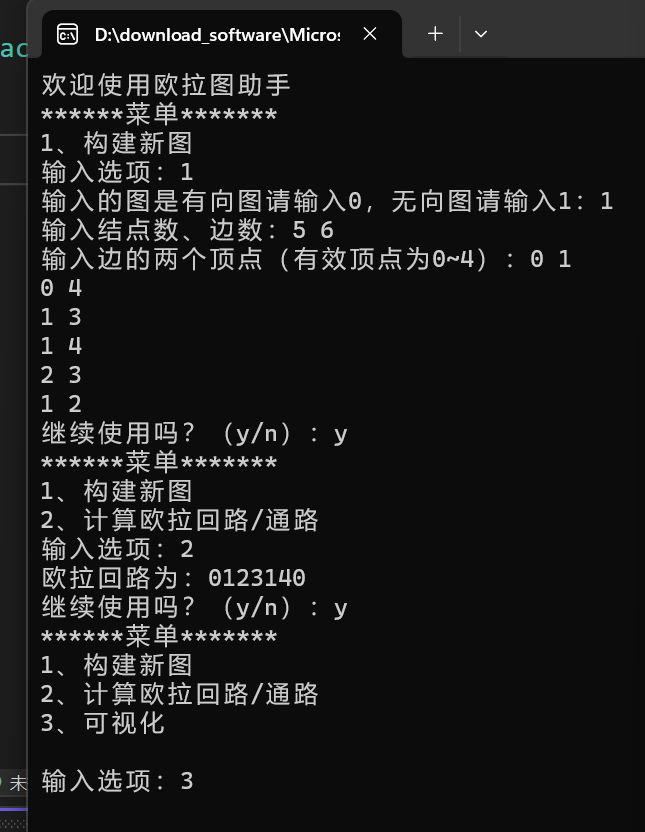
****

**2、测试结果和分析**

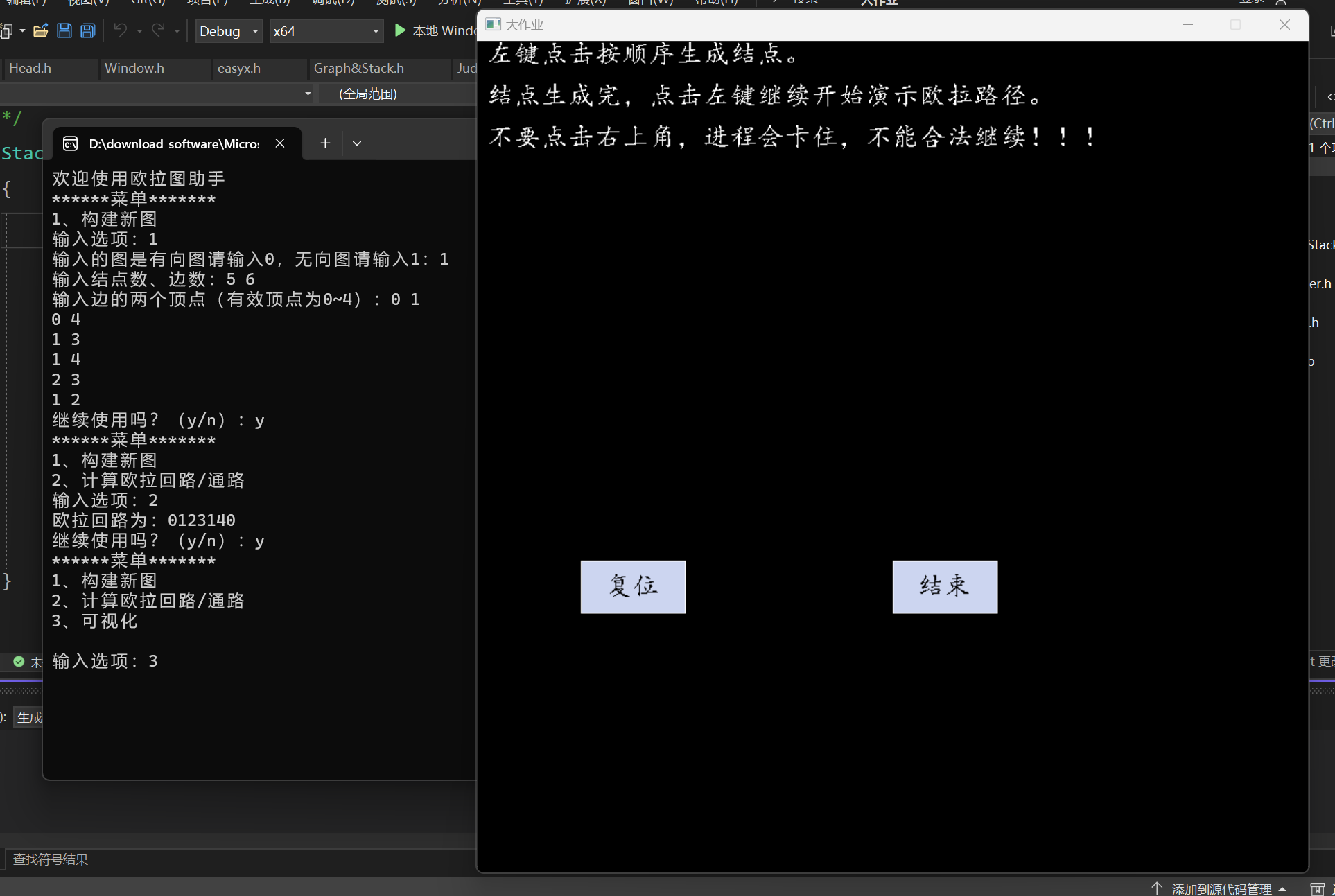
图一：

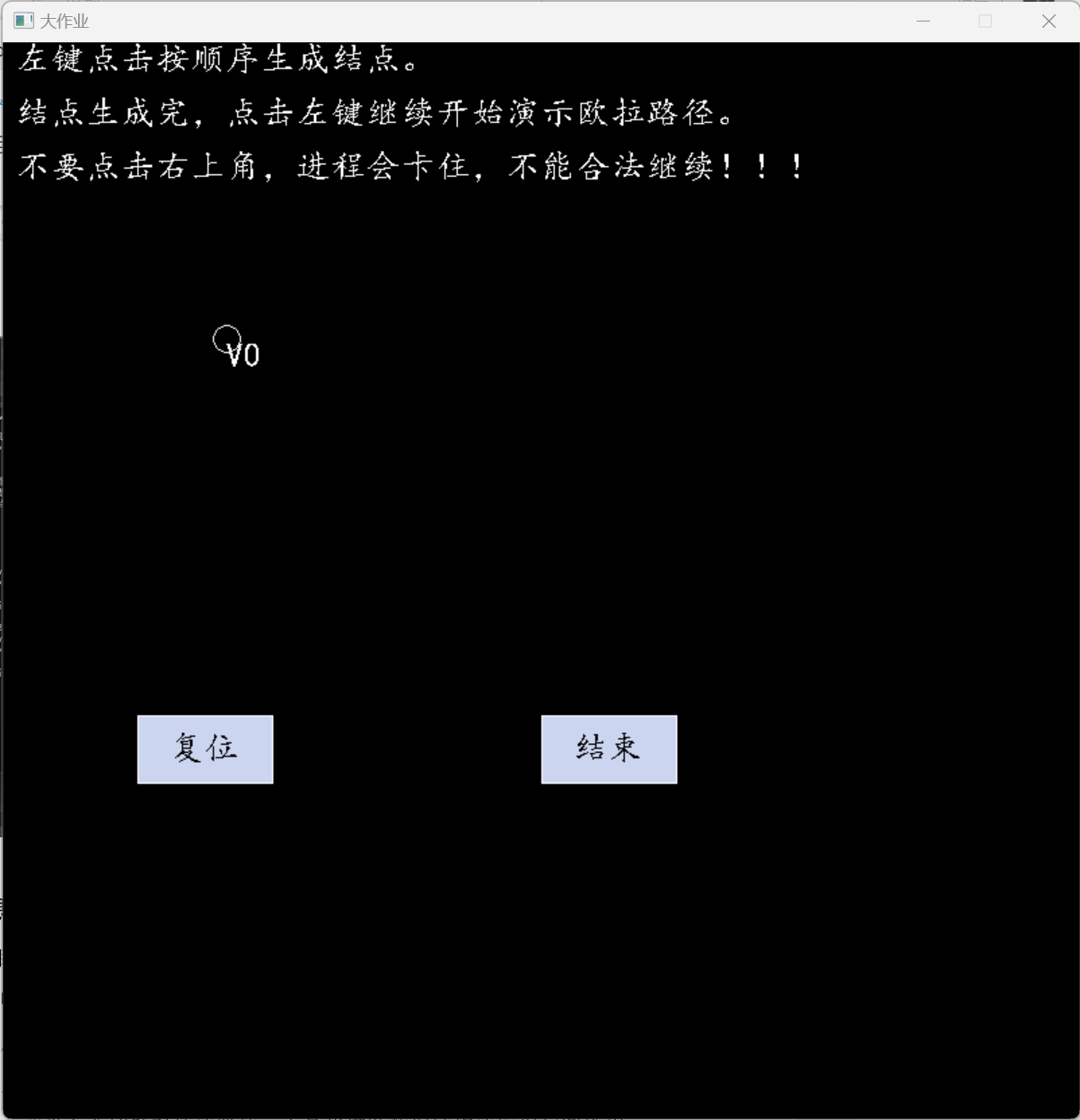


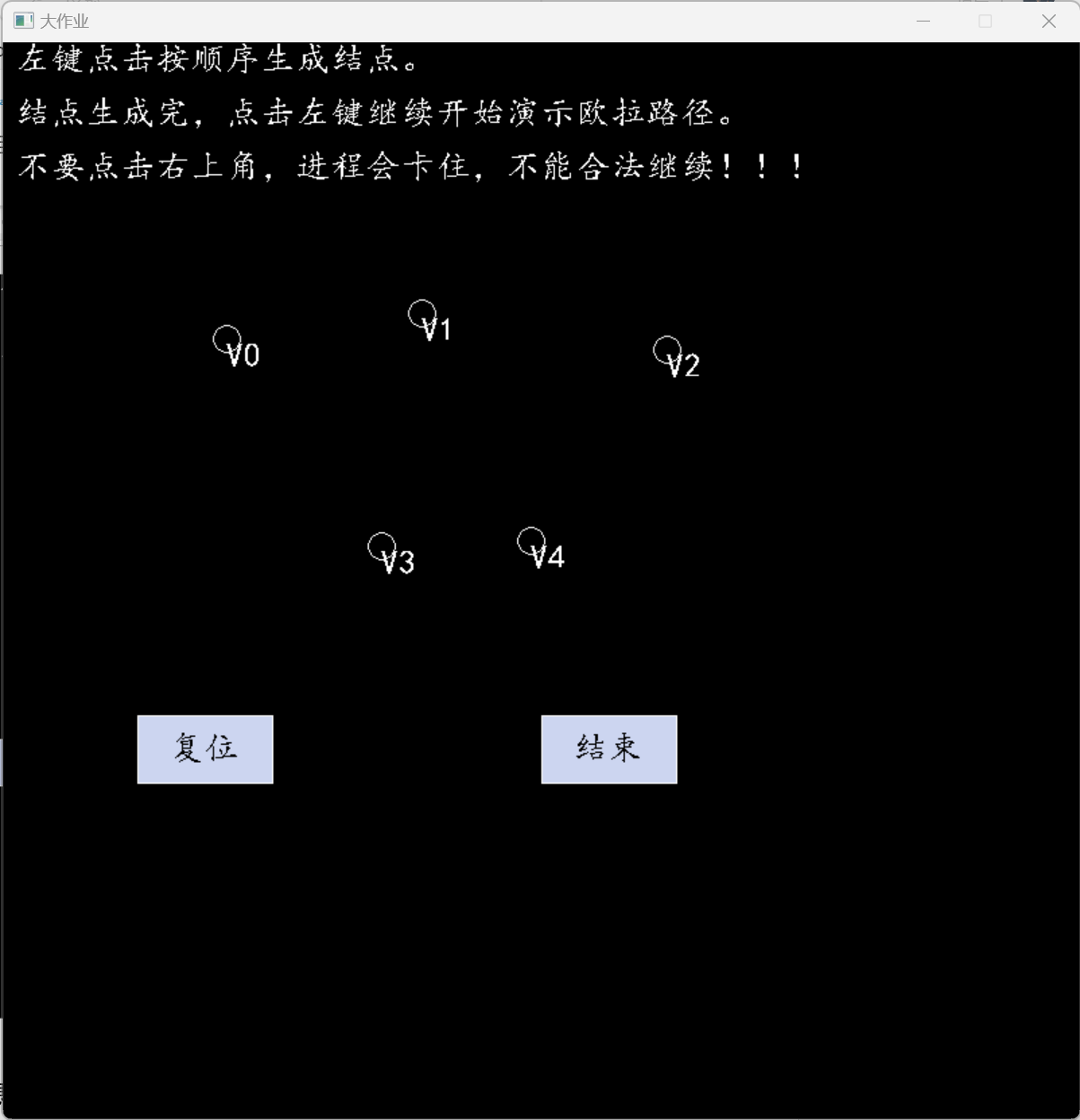
图二：

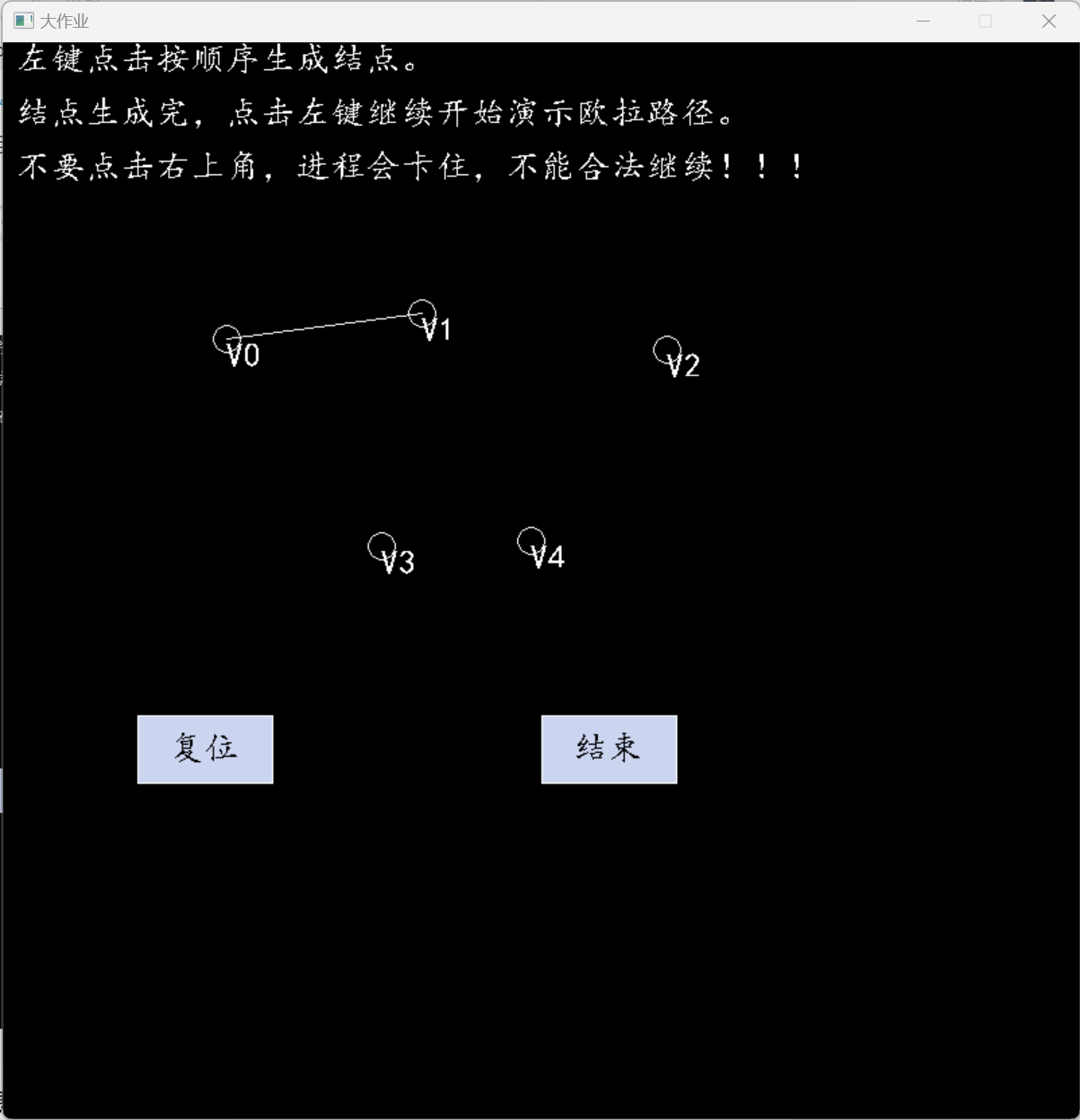


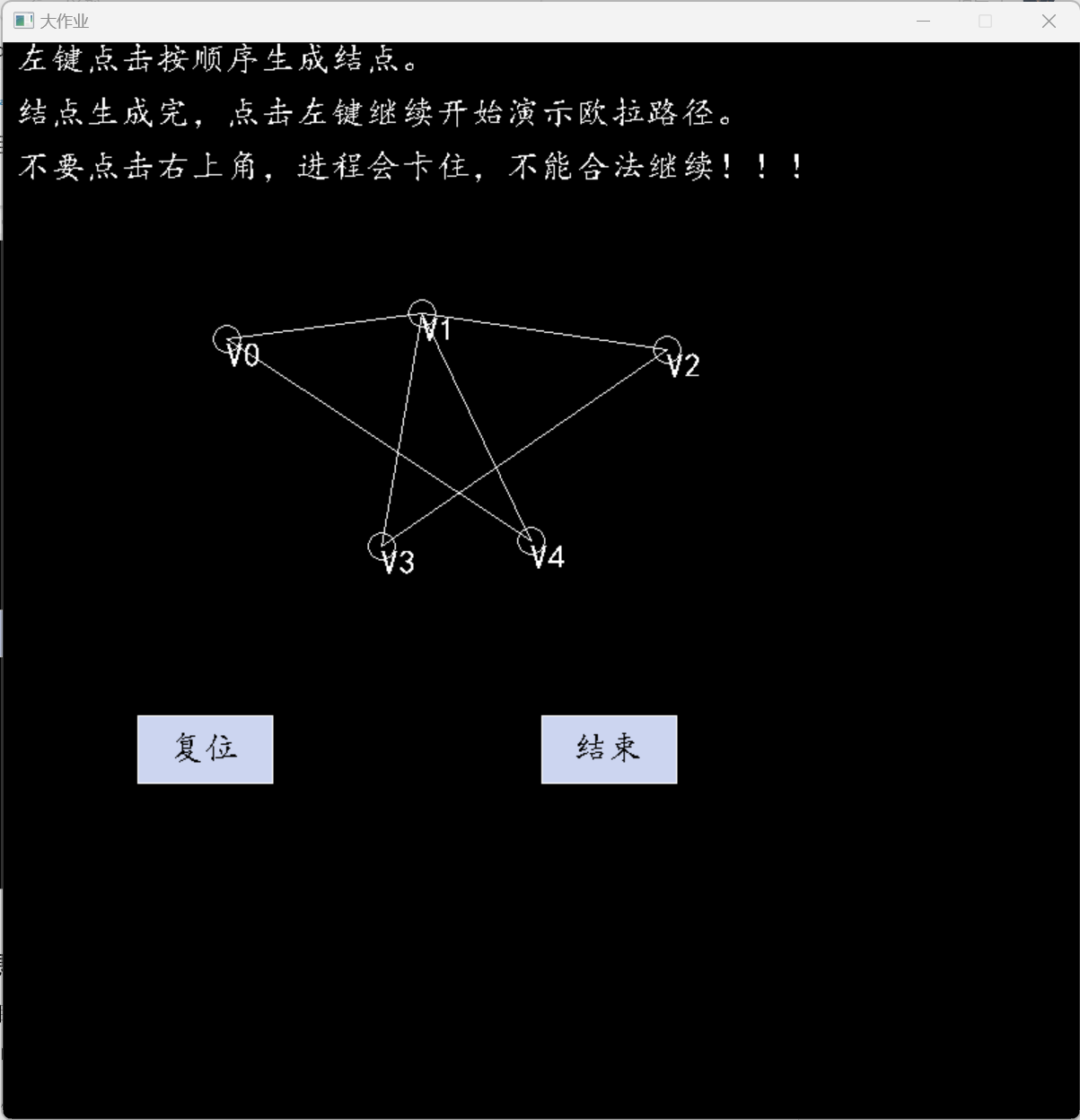
可视化过程（就展示这一个，其他的略，想看可以自己跑）：

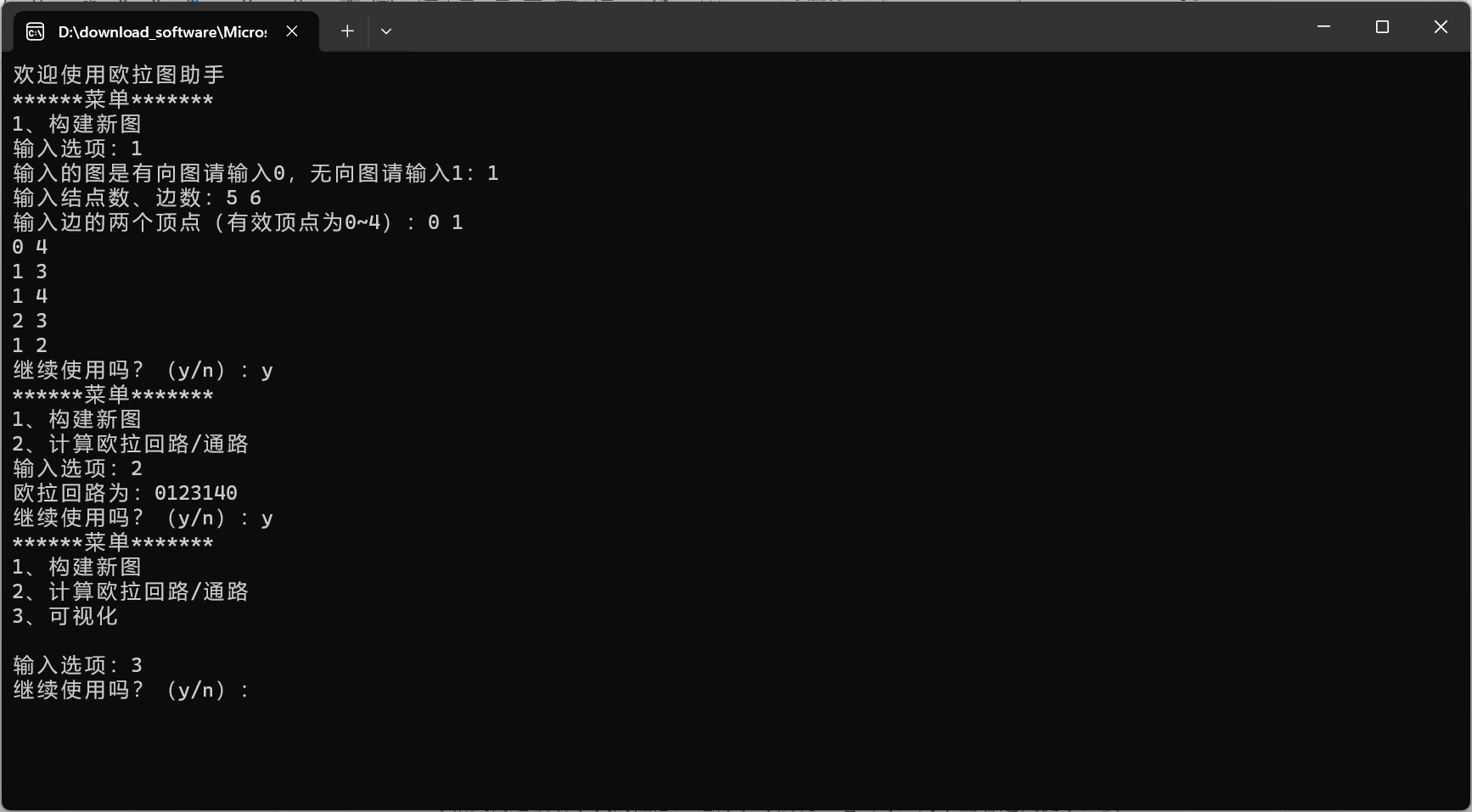




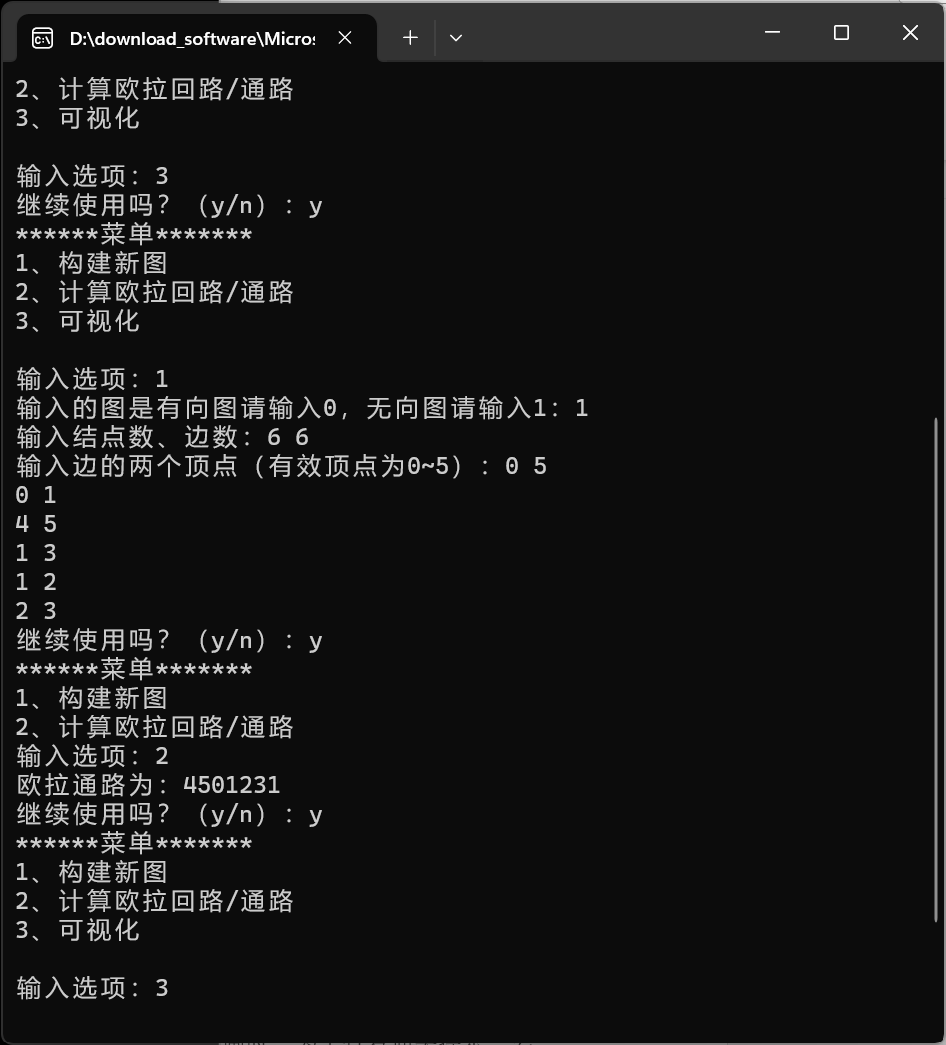


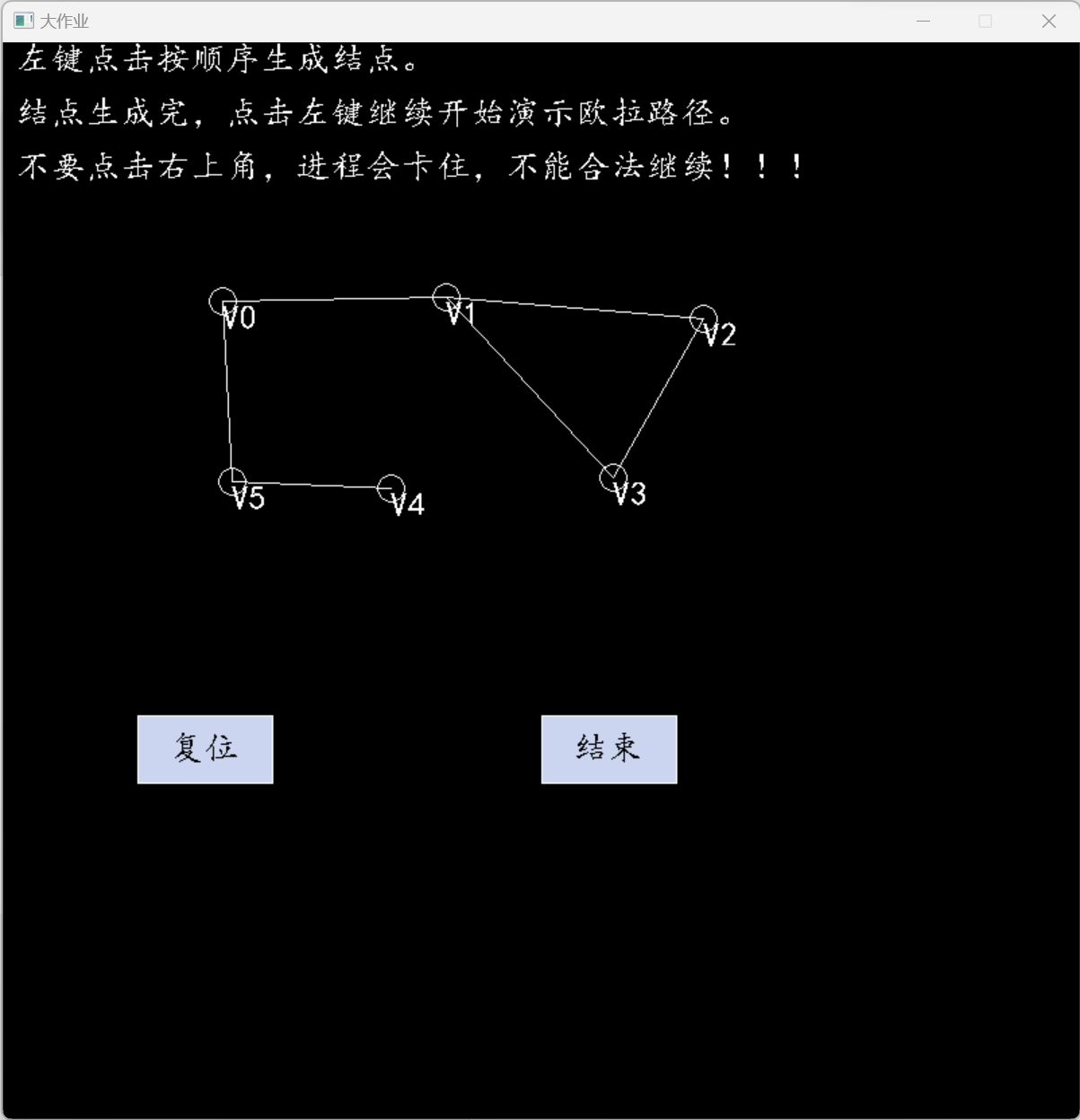






图三：（接着图二使用，验证了连续使用性）

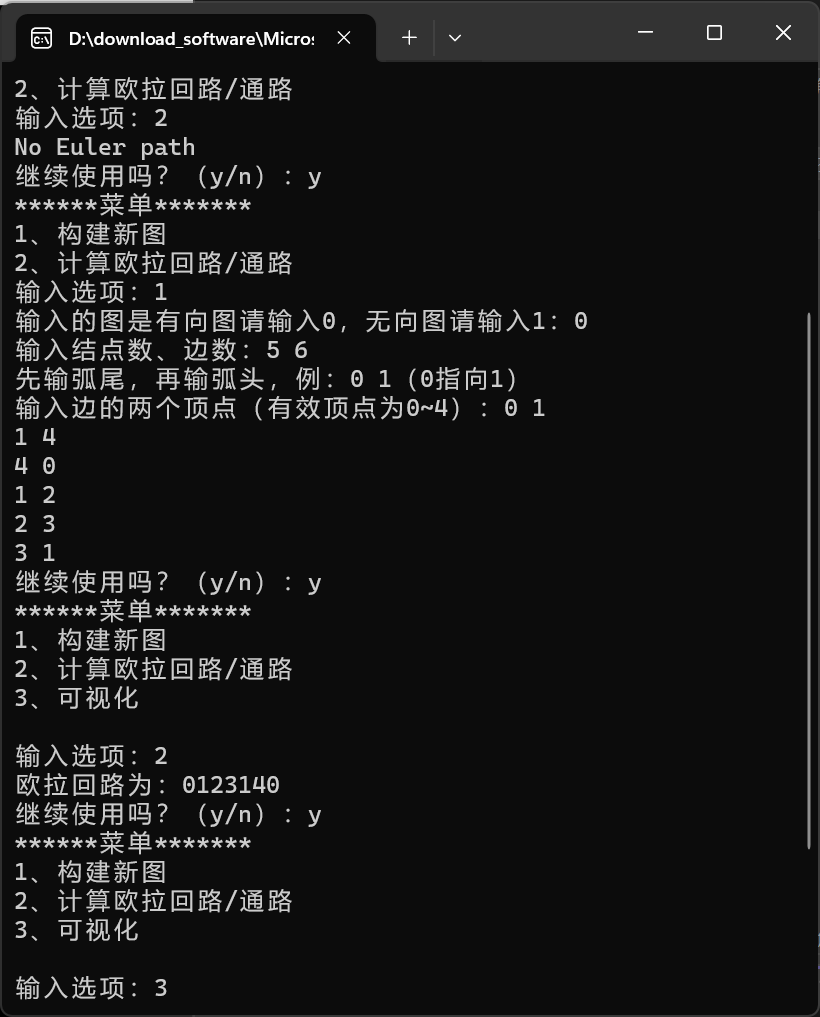


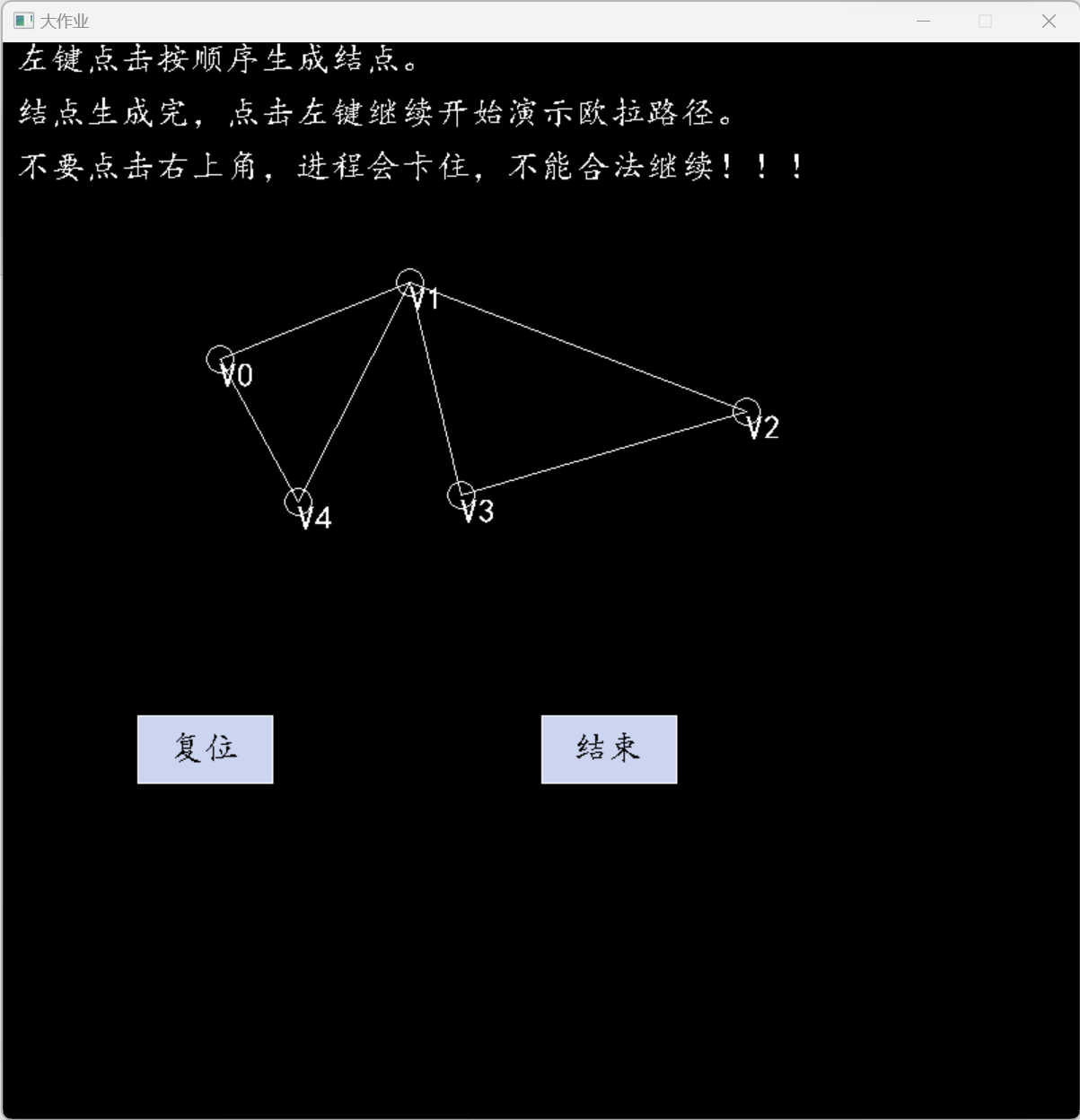


图四：

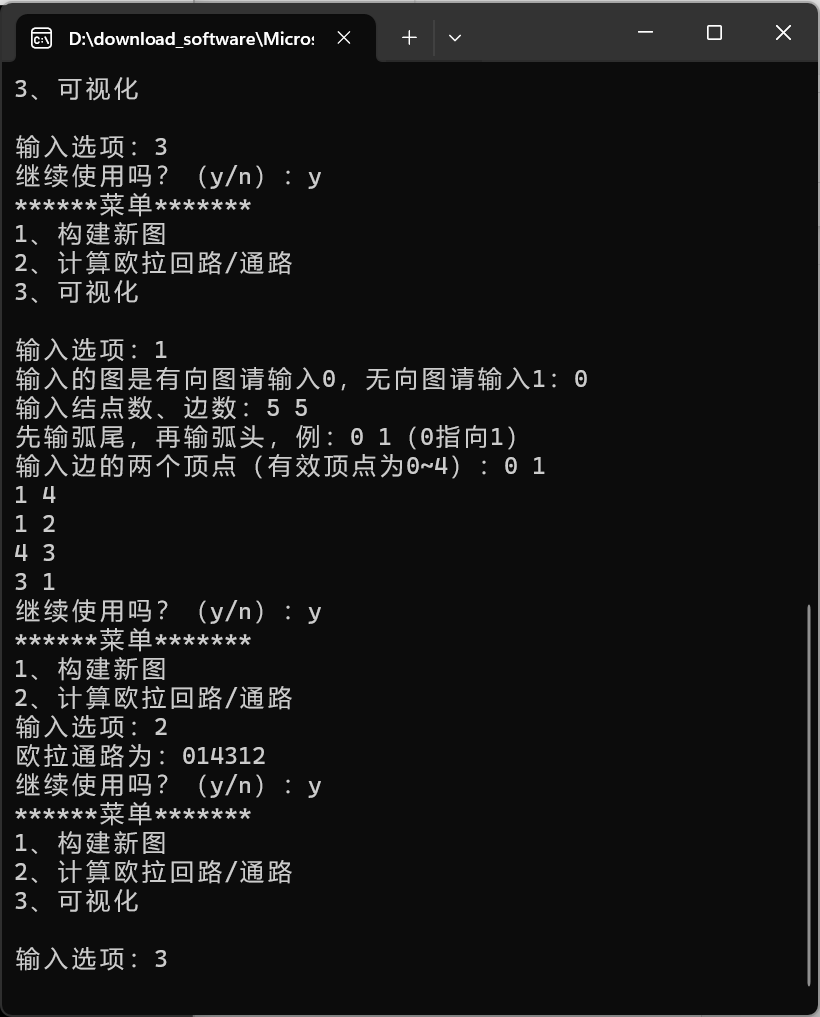


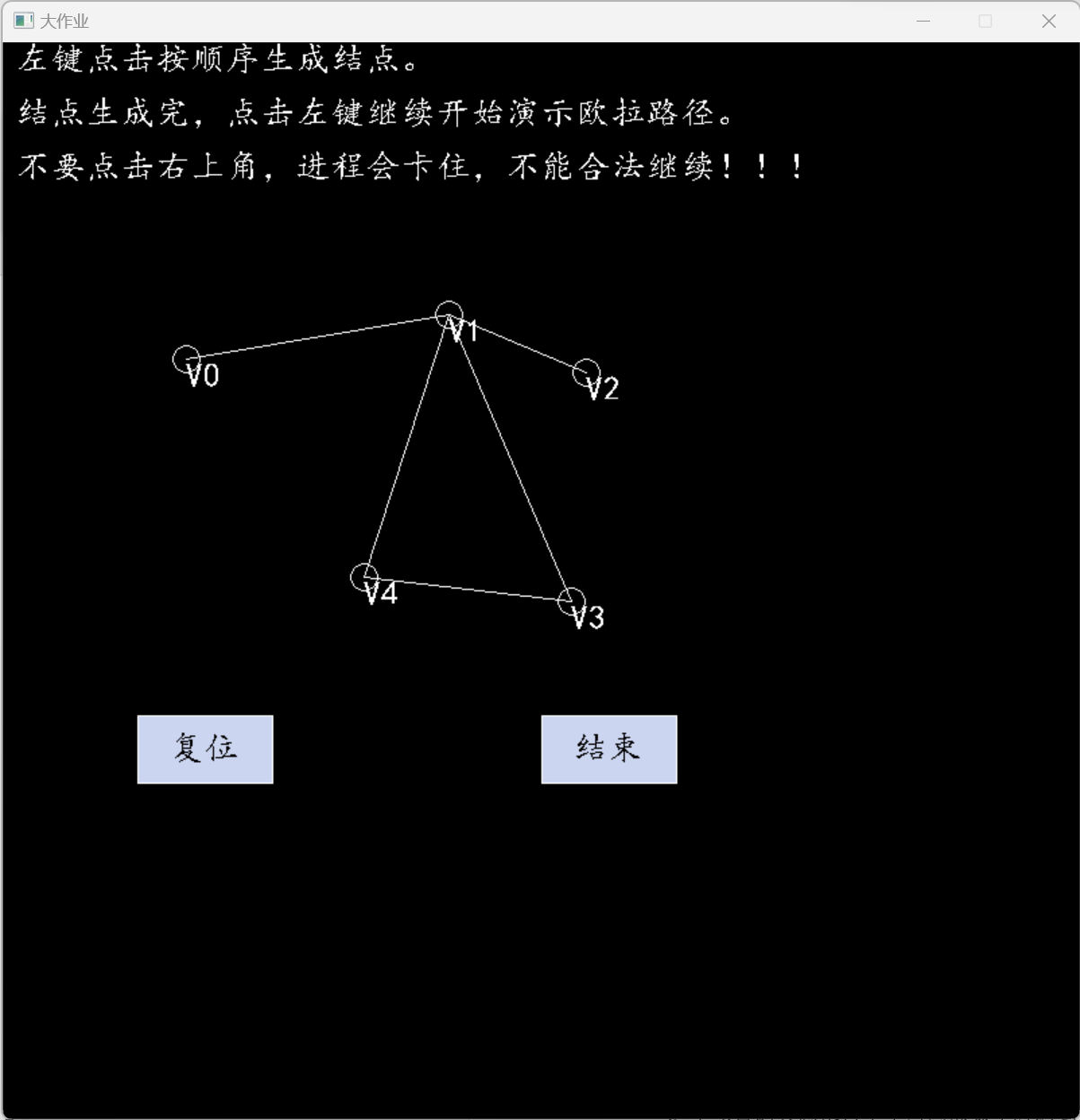
图五：





图六：





1. 感想与体会

这个项目的灵感来自于学习离散数学时遇到的欧拉图，当时提到的一笔画等相关问题吸引了我的注意，记得小时候玩一笔画时，为了查看通关提示，要钱或是观看广告，很费劲，而且上课提供的节点数较多的例子，也是一眼看不出路径的，虽然有找的技巧，但总是费时费力。同时，数据结构课程中，只将其作为引子，并没有介绍针对他的算法。于是我借助数据结构的知识与图论相结合，开发了这样的欧拉图助手，数据结构提供了邻接矩阵的存储方式以及深度搜索算法（DFS），图论提供了判断欧拉图的充要条件，以及寻找途径的技巧。

整个程序可以完成接收建立图，判断欧拉图，寻找欧拉路径，可视化步骤展示。基本解决了与欧拉图有关的一系列操作，只要涉及欧拉图，本程序可以一条龙服务，服务于不同人群。

对于我来说这个选题不仅锻炼了我对于图的处理能力，更将深度搜索与欧拉图相结合，让我认识到深度优先搜索不仅是查找，更是随需求而变，有千变万化。

同时，这也是我第一次尝试课外的内容，比如从0学习使用可视化图形库esayx，在学习过程中也遇到了很多问题，但不断的尝试、调整，终于实现了预期中的目标，给出了满意的答卷。

1. 参考文献

[1]黄立.确定欧拉回路的一种方法[J].承德民族师专学报,1995（02）：32.

DOI:10.16729/j.cnki.jhnun.1995.02.010.

1. 附录

源程序清单

头文件：

Graph&Stack.h

Head.h

Hierhoizer.h

Judge.h

Window.h

源文件：

main.cpp

（如果不想用Hierhoizer算法，可以用

DFS欧拉通路算法无向图（不包含判断）.cpp

Fleury算法.cpp）

(这三种是比较常见的欧拉图算法，效率：H>F>DFS)

Head.h

#pragma once

#include<stdio.h>

#include<stdlib.h>

#include<graphics.h>

#include<assert.h>

#define N 10

#include"Graph&Stack.h"

#include"Judge.h"

#include"Hierhoizer.h"

#include"Window.h"

Graph&Stack.h

typedef int GraphKind;

typedef struct Stack {

int StackElem[N];

int top;

int length;

}Stack;

typedef struct Graph {

int GraphElem[N][N];

//char vex[N];//结点为字符再启用

int vexnum, arcs;

GraphKind Gkind;

}Graph;

//入栈

Stack PushStack(Stack S, int x) {

S.StackElem[S.top] = x;

S.top++;

S.length++;

return S;

}

Hierhoizer.h

/\*第一个邻结点\*/

int FirstNod(Graph G, int x) {

int i;

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

if (G.GraphElem[x][i])

return i;

return -1;

}

/\*找下一个邻结点\*/

int NextNod(Graph G, int x, int i) {

int j;

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

if (G.GraphElem[x][j])

return j;

return -1;

}

/\*核心算法，Hierhoizer，利用了DFS

相当于很多环的并集

\*/

Stack Hierhoizer(Graph &G, Stack S, int x)

{

int i;

for (i = FirstNod(G, x); i >= 0; i = NextNod(G, x, i))

{

G.GraphElem[x][i]--;

if (G.Gkind)

G.GraphElem[i][x]--;

S = Hierhoizer(G, S, i);

}

S = PushStack(S, x);

return S;

}

Judge.h

/\*有向图判断是否有欧拉通路\*/

int JudgeEulerD(Graph G, int\* num) {

int start, end, deg, i, j, flagout = 0, flagin = 0;

\*num = 0;

start = 0;///这里初始化为0

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

{

deg = 0;

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

{

deg = deg + G.GraphElem[i][j] - G.GraphElem[j][i];

}

if (deg != 0)

{

if (deg == 1)

{

start = i;

flagout++;

}

else if (deg == -1)

{

end = i;

flagin++;

}

else

(\*num) = 3;

(\*num)++;

}

}

if (flagout == 1 && flagin == 1)

\*num = 2;

return start;

}

/\*无向图判断是否有欧拉通路\*/

int JudgeEulerUD(Graph G, int\* num) {

int start, deg, i, j;

\*num = 0;

start = 0;///这里初始化为0

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

{

deg = 0;

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

{

deg += G.GraphElem[i][j];

}

if (deg % 2 == 1)///奇度顶点

{

start = i;

(\*num)++;

}

}

return start;

}

Window.h

#pragma once

typedef struct Button {

int x, y, width, height;

COLORREF color;

char context[500];

}Button,\*PtrButton;

typedef struct Text {

int x, y;

int width;

int height;

char context[500] ;

}Text, \* PtrText;

PtrText SetText(PtrText text, ExMessage m, int i) {

char chs[3];

text = (PtrText)malloc(sizeof(Text));

text->x = m.x;

text->y = m.y;

text->width = 5;

text->height = 5;

strcpy(text->context, "V\0");

sprintf(chs, "%d", i);//问题点（还改了设置）

strcat(text->context, chs);

return text;

}

PtrButton createButton(int x,int y,int width,int height,char\* str) {

PtrButton button = (PtrButton)malloc(sizeof(Button));

assert(button);

button->x = x;

button->y = y;

button->width = width;

button->height = height;

button->color = RGB(204, 213, 240);

strcpy(button->context,str);

return button;

}

void drawButton(PtrButton button) {

int xx, yy,textw,texth;

setfillcolor(button->color);

fillrectangle(button->x,button->y,button->x+button->width,button->y+button->height);

setbkmode(TRANSPARENT);

textw = textwidth(button->context);

texth = textheight(button->context);

xx = button->x + (button->width - textw) / 2;

yy = button->y + (button->height-texth) / 2;

settextcolor(BLACK);

outtextxy(xx,yy,button->context);

}

int inButton(PtrButton button, ExMessage m) {

if (m.x > button->x && m.x<button->x + button->width

&& m.y>button->y && m.y < button->y + button->height)

{

button->color = RGB(236, 244, 255);

return 1;

}

button->color = RGB(204, 213, 240);

return 0;

}

//可视化按步骤显示生成路径

void show(Graph G,int\* road) {

int i = 0,j=0;

PtrText text = NULL;

Text info;

ExMessage\* vex;

ExMessage temp;

ExMessage m;

char button1name[] = "复位";

char button2name[] = "结束";

PtrButton reset, end;

initgraph(800, 800);

BeginBatchDraw();

vex = (ExMessage\*)malloc(sizeof(ExMessage) \* G.vexnum);

reset = createButton(100, 500, 100, 50, button1name);//问题点：const char\* 类型的实参与char\* 类型的形参不兼容

end = createButton(400, 500, 100, 50, button2name);

info.x = 10; info.y = 0;

//strcpy(info.context,"左键点击按顺序生成结点。\n结点生成完，点击左键继续开始演示欧拉路径。\n不要点击右上角，进程会卡住，不能合法继续！！！\n");

//问题点：没有自动换行功能

while (1)

{

settextstyle(25, 0, "楷体");

settextcolor(WHITE);

outtextxy(info.x, info.y, "左键点击按顺序生成结点。");

outtextxy(info.x, info.y + 40, "结点生成完，点击左键继续开始演示欧拉路径。");

outtextxy(info.x, info.y + 80, "不要点击右上角，进程会卡住，不能合法继续！！！");

drawButton(reset);

drawButton(end);

if (i < G.vexnum)

{

peekmessage((vex + i), EM\_MOUSE);

if (inButton(reset, vex[i])&& vex[i].message == WM\_LBUTTONDOWN)

{

for (i = G.vexnum; i >= 0; i--)

{

vex[i].message = 0;

vex[i].x = -1;

vex[i].y = -1;

}

i = 0;

//flushmessage(EX\_MOUSE);

cleardevice();

}

else if (inButton(end, vex[i])&& vex[i].message == WM\_LBUTTONDOWN)

{

closegraph();

return;

}

else if (vex[i].message == WM\_LBUTTONDOWN)

{

circle(vex[i].x, vex[i].y, 10);

settextcolor(WHITE);

text = SetText(text, vex[i], i);

outtextxy(text->x, text->y, text->context);

i++;

}

}

else

{

setlinecolor(WHITE);

setlinestyle(10);

if (j < G.arcs)

{

peekmessage(&temp, EM\_MOUSE);

if (inButton(reset, temp)&& temp.message == WM\_LBUTTONDOWN)

{

for (i = G.vexnum; i >= 0; i--)

vex[i].message = 0;

j = 0;

temp.message = 0;

temp.x = -1;

temp.y = -1;

i = 0;

m.message = 0;//问题点：复位后显示不全

//flushmessage(EX\_MOUSE);

cleardevice();

}

else if (inButton(end, temp)&& temp.message == WM\_LBUTTONDOWN)

{

closegraph();

return;

}

else if (temp.message == WM\_LBUTTONDOWN)

{

line(vex[road[j]].x, vex[road[j]].y, vex[road[j + 1]].x, vex[road[j + 1]].y);

j++;

temp.message = 0;

}

}

else

{

peekmessage(&m, EM\_MOUSE);

if (inButton(reset, m)&& m.message == WM\_LBUTTONDOWN)

{

for (i = G.vexnum-1; i >= 0; i--)

{

vex[i].message = 0;

vex[i].x = -1;

vex[i].y = -1;

}

j = 0;

temp.message = 0;

temp.x = -1;

temp.y = -1;

i = 0;

m.message = 0;

flushmessage(EX\_MOUSE);

cleardevice();

}

else if (inButton(end, m) && m.message == WM\_LBUTTONDOWN)

{

closegraph();

return;

}

}

//直接退出

//{

// peekmessage(&temp, EM\_MOUSE);

// if (temp.message == WM\_LBUTTONDOWN)

// break;

//}

}

FlushBatchDraw();

}

EndBatchDraw();

closegraph();

return;

}

main.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS -1

/\*原始灵感

void dfs(int x)

{

for (int i = head[x]; i; i = nex[i])

{

int y = to[i];

if (vis[i])continue;

vis[i] = 1;

dfs(y);

}

ans.push\_back(x);

}

\*/

#include"Head.h"

int main() {

Graph G = {0};

Graph Gcopy;

Stack S;

int i,j,v0,v1, start,choice=1,count=1;

int\* num;

int road[N];

char go='y',ch;

printf("欢迎使用欧拉图助手\n");

while(go=='y')

{

printf("\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\n");

switch (count) {

case 1:

printf("1、构建新图\n");

printf("输入选项：");

scanf("%d", &choice);

if (choice != 1)

{

printf("输入错误！\n");

exit(0);

}

count=2;

break;

case 2:

printf("1、构建新图\n");

printf("2、计算欧拉回路/通路\n");

printf("输入选项：");

scanf("%d", &choice);

if (choice != 1&&choice!=2)

{

printf("输入错误！\n");

exit(0);

}

count=3;

break;

default:

printf("1、构建新图\n");

printf("2、计算欧拉回路/通路\n");

printf("3、可视化\n");

printf("\n");

printf("输入选项：");

scanf("%d", &choice);

if (choice == 1)

count = 2;

}

switch(choice)

{

case 1:

printf("输入的图是有向图请输入0，无向图请输入1：");

scanf("%d", &G.Gkind);

if (G.Gkind != 0 && G.Gkind != 1)

{

printf("输入错误\n");

exit(0);

}

printf("输入结点数、边数：");

scanf("%d%d", &G.vexnum, &G.arcs);

//邻接矩阵全置0

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

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

G.GraphElem[i][j] = 0;

//输入边

if (!G.Gkind)

printf("先输弧尾，再输弧头，例：0 1（0指向1）\n");

printf("输入边的两个顶点（有效顶点为0~%d）：", G.vexnum - 1);

for (i = 0; i < G.arcs; i++)

{

scanf("%d%d", &v0, &v1);

if (v0 < 0 || v0 >= G.vexnum || v1 < 0 || v1 >= G.vexnum)

printf("输入有错，请检查\n");/////////停止

G.GraphElem[v0][v1]++;

if (G.Gkind)

G.GraphElem[v1][v0]++;

}

break;

//判断是否有欧拉通路

case 2:

num = (int\*)malloc(sizeof(int));

if (G.Gkind)

start = JudgeEulerUD(G, num);

else

start = JudgeEulerD(G, num);

//找通路

if (\*num == 0 || \*num == 2)

{

S.length = 0;

S.top = 0;

Gcopy = G;

S = Hierhoizer(Gcopy, S, start);

if (\*num == 2)

printf("欧拉通路为：");

else

printf("欧拉回路为：");

i = 0;

while (S.top != 0) {

S.top--;

road[i] = S.StackElem[S.top];

printf("%d", S.StackElem[S.top]);

S.length--;

i++;

}

printf("\n");

}

else

{

printf("No Euler path\n");

count = 2;

}

break;

//可视化

case 3:

show(G, road);

default:

break;

}

printf("继续使用吗？（y/n）：");

while ((ch = getchar()) != '\n' && ch != EOF);

scanf("%c", &go);

}

return 0;

}