* **题目：求随机顶点到达顶点数量的平均值**
* **班级：2021211312 姓名：杨天宇 学号：2021211281** 分工： **1,2**
* **班级：2021211312 姓名：王昊钰 学号：2021211282** 分工： **3,4**
* **班级：2021211312 姓名：李志茂 学号：2021211289** 分工： **5,6**

1. 需求分析

输入输出形式：

程序能达到的功能

输出结果

1. 概要设计

解决思路：

数据结构类型：

主程序流程

函数调用：

1. 详细设计

数据类型：

邻接矩阵 int arc[V][V];

详细操作：

int main (void)

{

int V,E;

cout << "input the number of V and E" << endl; // 输入顶点数和边数

cin >> V >> E;

int arc[V][V];

memset(arc,0,sizeof(arc));

int \*arcpoint = (int\*)arc;

if (init\_arc(arcpoint,V,E)) // 随机生成一个邻接矩阵

{

cout << "initialize error,too much arc" << endl; // 输入边数过多，无法生成简单有向图

}

warshall(arcpoint,V); //用warshall算法求传递闭包

cout << "The average value is" ;

printf(" %.4f \n",average(arcpoint,V)); //输出平均值，保留4位小数

return 0;

}

int init\_arc(int \*arc ,int V, int E)

{

int col,raw; // 生成边行列号

int count = 0; //记录成功生成边数

random\_device rd; // 生成随机数种子

minstd\_rand r\_eng(rd()); //随机数生成器

if (E > V\*(V-1))

return 1;

while (count<E)

{

col = r\_eng() % V;

raw = r\_eng() % V; //确定行列号

if (col == raw) //不包含自环

continue;

else if (arc[raw\*V+col] ) //不包含平行边

continue;

else

{

arc[raw\*V+col] = 1;

count ++;

}

}

return 0;

}

void warshall(int \*arc, int V)

{

int i,j,k;

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

{

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

{

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

{

if(arc[j\*V+i]&&arc[i\*V+k])

arc[j\*V+k] = 1; //warshall算法

}

}

}

}

float average(int \*arc, int V)

{

int count = 0;

int i,j;

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

{

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

{

if (arc[i\*V+j])

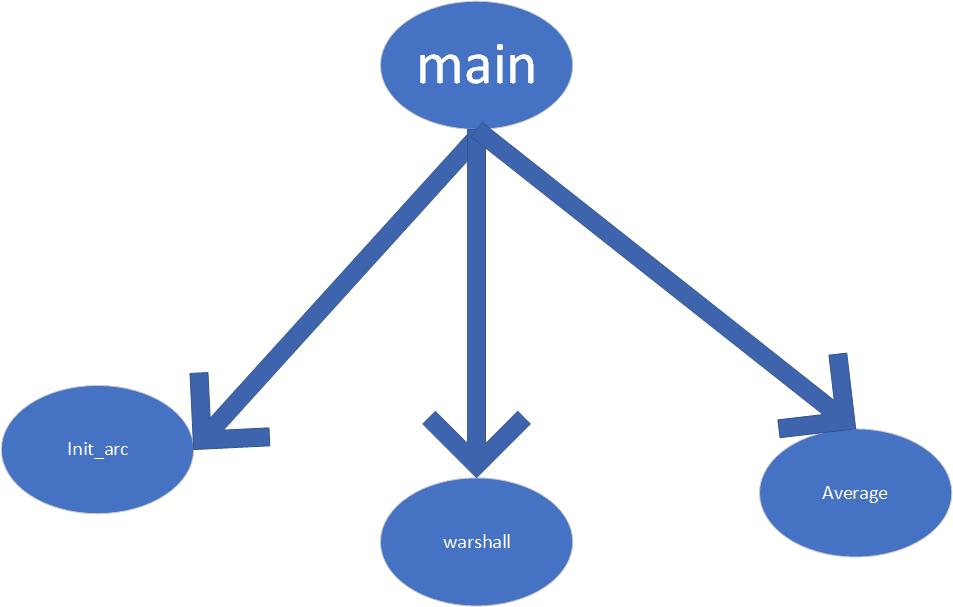
count++; // 记录矩阵中1的个数

}

}

return (float)count/V; //返回可到达顶点数量的平均值

}



4、调试分析报告

遇到的问题：

传递不定长度的二位数组到函数时无法成功传递。

解决方案:

将二维数组视作一个指针，压缩成一维处理。

回顾讨论：

首先生成一个符合要求的简单有向图，之后利用warshall算法求传递闭包得出可到达点，最后统计可到达点的平均值

init\_arc时间复杂度：O(E);空间复杂度：O（1）。

warshall时间复杂度：O(V^3);空间复杂度：O（1）。

Average时间复杂度：O(V^2);空间复杂度：O（1）。

改进设想：

在检测到矩阵全为1时可以使warshall算法提前退出，减少计算次数

经验和体会：

离散数学是数据结构的重要前导知识，有了算法知识的加成，编写相应程序十分简单。

1. 用户使用说明
2. 测试结果