**《离散数学》Warshall算法求传递闭包项目文档**

1. **题目简介**

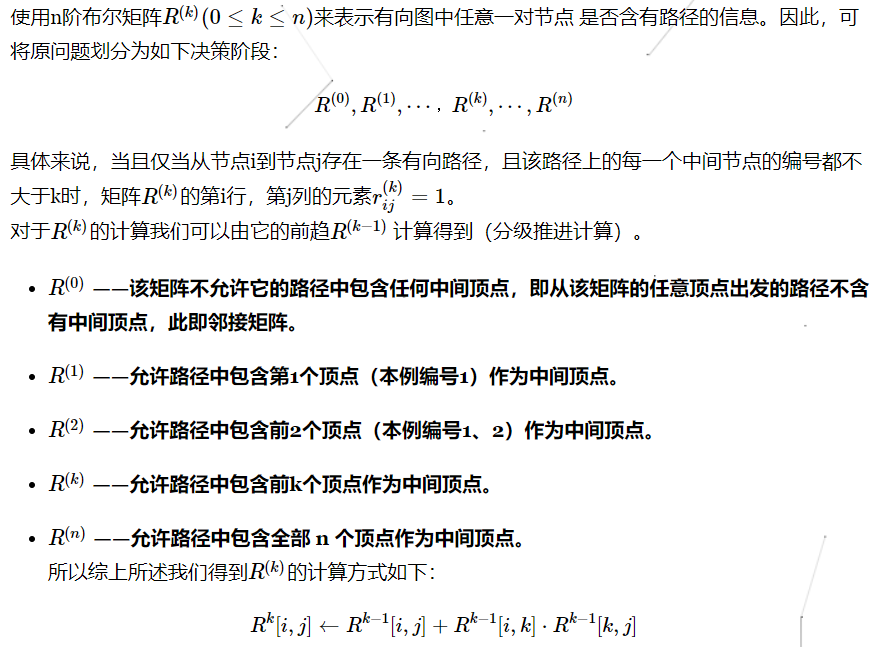
不同于第三题的传统方法求传递闭包，使用warshal1算法求传递闭包。

1. **解题思路**

**2.1实验原理**

从数学上来说，传递闭包是在集合上求包含关系的最小传递关系。从关系图的角度来说，就是如果原关系图上有从一点到另一点的路径，则传递闭包的关系图上就应该有从该点到另一点的边.

此传统算法的时间复杂度为O（n^3）。



**2.2 设计程序**

根据功能要求，先设计main函数具体实现，其中包括健壮性处理，输入判断；还包括输入提示等。

主要部分为根据选项调用相应的函数；

cout << "1.warShall方法求传递闭包" << endl;

cout << "2.退出程序" << endl;

cout << "请输入你的选择:";

int choice;

//输入错误处理

while (cin >> choice)

{

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

{

cout << "输入错误，请重新输入！" << endl;

cin.clear();

cin.ignore(1024, '\n');

}

else

break;

}

switch (choice) {

case 1:

warShall(matrix, n);

break;

case 2:

break;

default:

break;

}

return 0;

1. **核心算法**

**3.1 求传递闭包算法实现**

void warShall(vector<vector<int>>matrix, int n) {

//最外层循环代表以i为内部顶点，可以经过i到达其它点

for (int i = 0; i < n; i++) {

//里面这两层循环代表由j点出发经i点到达k点

for (int j = 0; j < n; j++) {

for (int k = 0; k < n; k++) {

//所以中间用按位或来判断，只要有一个成立就行

matrix[j][k] = matrix[j][k] || (matrix[j][i] && matrix[i][k]);

}

}

}

output(matrix, n);

}

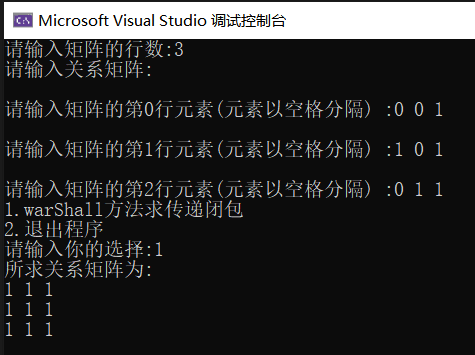
**3.2 求传递闭包算法描述**

使用三层循环嵌套，最外层循环代表以i为内部顶点，可以经过i到达其它点，里面两层循环代表由j点出发经i点到达k点，由j到k无非就是两种情况，j直接到k或者j到i,i再到k,所以中间用逻辑运算符或来判断，只要有一个是1，则证明两点之间存在有效路径，然后就直接赋值matrix[j][k]为1，表明最终传递闭包图上，两点之间有边直接相连。其中，matrix[j][i] && matrix[i][k]表示两点之间有经过i的路径存在。

1. **所用数据结构**

Vector数组：相当于动态开辟数组，不需要考虑数组溢出问题，不会造成空间浪费，使用方便；用来存放关系矩阵。

1. **实验结果**



1. **心得体会**

和传统算法相比，warshall算法更加简洁易懂，并且时间复杂度降低。warshall算法通俗的来说就是通过n^3操作来判断矩阵中所有可以联通的点，例如现在有1到2，2到3，3到4，4到5，若想要判断1与4连通需要判断1与3连通，而想要判断1与5连通需要先判断1与4连通，所以需要进行n^3来暴力判断所有的点。显然 Warshall和 Floyd算法在竞赛里是很少用到，时间复杂度太大。