**离散数学课程设计**

**项目说明文档**

**Warshall算法求传递闭包**

作 者 姓 名： 苏家铭

学 号： 2151299

指 导 教 师： 唐剑锋

学院、 专业： 软件学院 软件工程

同济大学

**Tongji University**



**目录**

[1 项目分析 1](#_Toc32459)

[1.1 项目背景 1](#_Toc1034)

[1.2项目要求 1](#_Toc26071)

[1.2.1 功能要求 1](#_Toc17736)

[1.2.2 输入要求 1](#_Toc19480)

[1.2.3 输出要求 1](#_Toc27948)

[1.2.4 项目实例 1](#_Toc15000)

[2 项目设计及实现 2](#_Toc3274)

[2.1 数据结构设计思路 2](#_Toc16769)

[2.2 类设计 2](#_Toc20843)

[2.2.1 链表 2](#_Toc18353)

[2.3 项目算法 3](#_Toc8766)

[2.3.1 实现思路 3](#_Toc3110)

[2.3.2 代码实现 4](#_Toc3252)

[3 项目测试 4](#_Toc3943)

[3.1 一般情况 4](#_Toc31909)

[3.2 交集为空的情况 5](#_Toc3479)

[3.3 完全相交的情况 5](#_Toc26284)

[3.4 其中一个序列完全属于交集的情况 5](#_Toc13791)

[3.5 其中一个序列为空的情况 6](#_Toc32352)

[3.6 输入非法字符的情况（健壮性） 6](#_Toc29253)

[4 算法性能分析 7](#_Toc8015)

[4.1 正确性 7](#_Toc27442)

[4.2 可使用性 7](#_Toc1775)

[4.3 可读性 7](#_Toc3126)

[4.4 效率 7](#_Toc11991)

[4.5 健壮性 7](#_Toc15299)

# 1 项目分析

## 项目背景

利用矩阵表示关系时求解传递闭包的一般算法的时间复杂度达到O(n4)，这时Warshall提出了一种新式求解传递闭包的算法，能将时间复杂度降为O(n3)，极大地节省时间。Warshall算法又叫floyd-Warshall算法，思想为i->j&&j->k，那么i->k。

## 1.2项目要求

利用Warshall算法求解传递闭包

# 2 **项目设计及实现**

## 2.3 **项目算法**

### 2.3.1 实现思路

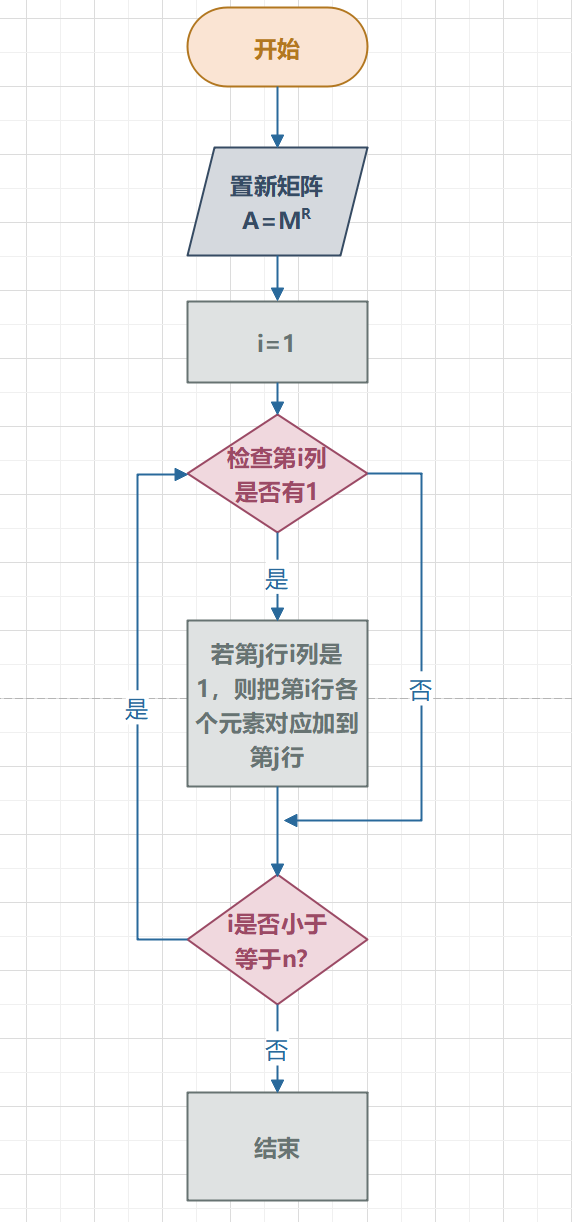
Warshall算法求t(R):

R为二元关系, MR = A,R2的矩阵为A2,...Rk的矩阵为Ak。于是t(R)的矩阵记为MR+=A+A2+...+Ak+...(其中+是逻辑加)

算法步骤如下：

1. 置新矩阵 A=MR;
2. 置i=1;
3. 对所有j，如果A[j,i]=1,则对k=1,2,...,n A[j,k]=A[j,k]+A[i,k];
4. i加1;
5. 如果i<=n,则转到步骤3，否则停止。

流程图如下：



### 2.3.2 代**码实**现

#include<iostream>

using namespace std;

void warshell();

void show();

void init();

void select();

void delete\_shuzu(int\*\* arrx, int n);

int\*\* arr, n;

int main() {

init();//初始化矩阵

show();//显示初始矩阵

cout << "=======================" << endl;

cout << "===Warshell传递闭包====" << endl;

cout << "=======================" << endl;

warshell();//用warshell算法求传递闭包

show();//显示传递闭包矩阵

delete\_shuzu(arr, n);//释放存储空间

return 0;

}

void delete\_shuzu(int\*\* arrx, int n) {

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

delete[]arrx[i];

}

void init() {

cout << "请输入矩阵行(列)数:";

while (1) {

cin >> n;

if (cin.fail() || n < 1) {

cout << "输入矩阵行(列)数非法，请再次输入:" << endl;

cin.clear();

cin.ignore(65535, '\n');

}

else

break;

}

cout << "创建初始矩阵:" << endl;

arr = new int\* [n];

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

arr[i] = new int[n];

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

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

cin >> arr[i][j];

}

}

}

void warshell() {

//(1)i＝0；

//(2)对所有j如果a[j，i]＝1，则对k＝0，1，…，n-1，a[j，k]＝a[j，k]∨a[i，k]；

//(3)i加1；

//(4)如果i<n，则转到步骤2，否则停止

int i = 0;

int j = 0;

int k = 0;

for (i = 0; i < n; i++)//共n次遍历，i是列

{

for (j = 0; j < n; j++)//检查第i列是否有1

{

if (arr[j][i]==1)//若第j行i列是1，则把第i行各个元素对应加到第j行

{

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

{

arr[j][k] = arr[j][k] | arr[i][k];//逻辑加

}

}

}

}

}

void show() {

cout << "矩阵：" << endl;

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

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

cout << arr[i][j] << " ";

}

cout << endl;

}

}

# 3 项目测试

以教材p89例4.10的矩阵为例：



# 4 算法性能分析

## 4.1 正确性

本算法能正确地执行预定的功能和性能要求，由以上版块证明本项目与教材p89例4.10情况比较一致并经过大量数据比对成功，满足正确性。

## 4.2 可使用性

本算法可以很方便地使用，warshall算法求传递闭包的功能已封装在一个函数中。并且该算法有良好的界面和完备的用户文档。

## 4.3 可读性

本算法逻辑清晰、简单、且结构化，所有命名与函数名都具有实际含义，让人见名知义。且算法中包含了大量注释，简要说明了算法功能、输入与输出参数的使用规则、重要数据的作用、算法中各程序段完成的功能。

## 4.4 效率

warshall算法利用三个for循环来解决传递闭包问题，而一般算法则使用了4个for循环，本算法将O(n4)的时间复杂度降为O(n3)，极大地降低了时间成本。

## 4.5 健壮性

本算法对于边界条件，诸如：输入矩阵行/列数非法，小于1的情况有错误判断和重新输入功能。

# 5 实验感想

普通的算法是将一个个矩阵通过相乘计算出n种不同次方并相加，这种算法的时间复杂度非常高：需要n个矩阵相加O(n)，每个nXn的矩阵需要通过矩阵相乘得到O(n2)，每个矩阵中的每个项又需要某行和某列各个对应数的相乘之和O(n)，各个操作之间是嵌套关系，所以总的时间复杂度为O(n4)，非常消耗时间，并且在计算过程中需要若干个辅助矩阵和变量，也非常消耗空间。

而warshall算法很好地解决了这一现象。首先Warshall算法只需要三个整型变量作为辅助，极大地减少了空间的消耗。并且在Warshall算法中只需要三个for循环，时间复杂度为O(n3)，也最大限度地减少了时间的损耗。

这启示着我们在寻求求解一个问题的时候要从多钻研问题，使用不同的方法，不要仅仅局限以一种，要创新，超越前人的思想。就像 Warshall一般，带来了新的算法为后人造福，为社会做出自己的贡献。