项目说明文档

离散数学课程项目

——Warshall算法求关系的传递闭包

作 者 姓 名： 陈奕名

学 号： 2351883

指 导 教 师： 李冰

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 1 项目分析

## 1.1 项目要求

手动输入矩阵阶数，然后手动输入关系矩阵，利用Warshall算法(简便)求解传递闭包。

# 2 项目设计

## 2.1 数据结构设计

本程序的核心数据结构是 二维数组，用来表示图的关系矩阵（也称为邻接矩阵）。在程序中，我们使用了 std::vector 来动态管理矩阵的数据，因为它提供了灵活的内存管理并能动态扩展尺寸。

vector<vector<int>> matrix：一个二维向量，用于表示关系矩阵。矩阵的元素为 0 或 1，matrix[i][j] = 1 表示从节点 i 到节点 j 有直接的关系或路径，matrix[i][j] = 0 表示没有路径。矩阵的行数和列数由用户输入，表示图的节点数（n个节点）。矩阵的大小为 n x n，其中 n 为节点的数量，每行每列对应一个节点的关系。

## 2.2 算法设计

程序的主要目标是计算给定关系矩阵的 传递闭包。为了完成这一目标，采用了 Warshall 算法。该算法是一种基于动态规划的算法，用于计算图的传递闭包。具体步骤如下：

1.输入矩阵：

用户输入一个 n x n 的矩阵，表示一个有向图，其中 matrix[i][j] = 1 表示从节点 i 到节点 j 有直接路径，0 表示没有直接路径。

2.初始化：

初始化一个矩阵 matrix，并将用户输入的矩阵存储到该矩阵中。

3.Warshall 算法：

对每个中间节点 k，检查是否可以通过节点 k 使得从节点 i 到节点 j 存在路径。如果存在这样的路径，则通过更新 matrix[i][j] 来反映传递闭包的关系。

使用三层嵌套循环：

第一个循环遍历所有的中间节点 k。

第二个循环遍历所有的起始节点 i。

第三个循环遍历所有的终止节点 j。

对每一对节点 (i, j)，检查是否通过节点 k 可以增加一条新的路径。如果可以，就更新 matrix[i][j]。

4.输出传递闭包：

在计算完成后，输出更新后的矩阵，即为传递闭包后的矩阵。

# 3 代码实现

#include <iostream>

#include <vector>

using namespace std;

// 函数：输出矩阵

void printMatrix(const vector<vector<int>>& matrix) {

int n = matrix.size();

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

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

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

}

cout << endl;

}

}

// Warshall 算法计算传递闭包

void warshall(vector<vector<int>>& matrix) {

int n = matrix.size();

// 使用中间节点 k 更新关系矩阵

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

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

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

// 如果 i->k 和 k->j 都有路径，则 i->j 也有路径

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

}

}

}

}

int main() {

int n;

cout << "请输入矩阵阶数: ";

cin >> n;

vector<vector<int>> matrix(n, vector<int>(n));

// 逐行输入矩阵

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

cout << "请输入矩阵的第 " << i << " 行元素（元素以空格分隔）：";

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

cin >> matrix[i][j];

}

}

// 计算传递闭包

warshall(matrix);

// 输出传递闭包结果

cout << "传递闭包后的关系矩阵：" << endl;

printMatrix(matrix);

return 0;

}

# 4 测试

## 4.1 一般情况

**测试用例**：

4

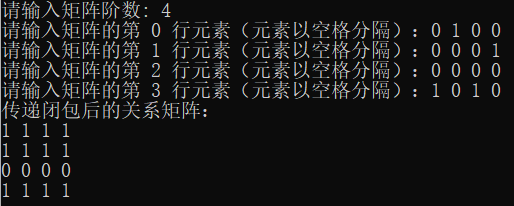
0 1 0 0

0 0 0 1

0 0 0 0

1 0 1 0

**实验结果:**



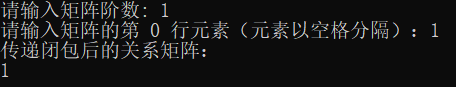
## 4.2 只有一个元素情况

**测试用例**：

1

1

**实验结果:**



# 5 心得体会

通过这次作业，我对传递闭包的概念及其应用有了更深的理解。

首先，程序通过二维数组（vector<vector<int>>）来表示关系矩阵，这使得矩阵的存储与操作变得简单且高效。在输入矩阵时，通过逐行输入的方式，用户可以更加直观地理解每一行代表的图的节点关系。程序的设计遵循了清晰的结构，使得输入、计算和输出分离，易于扩展和维护。

Warshall 算法的核心思想是利用动态规划思想，逐步更新关系矩阵，使得最终得到的矩阵包含了所有可能的路径，这与图的传递闭包的定义是完全一致的。通过三层嵌套循环对每一对节点 (i, j)，检查是否能通过中间节点 k 形成一条新的路径。这让我深刻认识到，传递闭包不仅仅是简单的图遍历，而是通过对已有路径的不断推导来形成新的路径。

在实现过程中，我也意识到算法的时间复杂度是 O(n^3)，当节点数较大时，计算效率可能成为瓶颈。因此，如何在实际应用中优化算法或改用其他更高效的算法是值得思考的问题。

总的来说，这次作业不仅加深了我对图算法特别是传递闭包的理解，还提高了我的编程能力，尤其是在处理二维数组和应用动态规划思想方面。通过实现这个程序，我学会了如何将理论算法转化为实际可执行的代码，并进一步加深了对算法效率和优化的思考。