《离散数学》课程实验报告文档

题目：Warshall 算法求传递闭包

姓名：　赵卓冰

学号：　2252750

专业：　软件工程

年级：　2023 级

指导教师：唐剑锋

2024 年 12 月 2 日

1. 实验报告：基于Warshall算法的传递闭包计算

1.1. 实验内容

输入一个关系矩阵，利用 Warshall 算法计算其传递闭包并输出结果。

- **关系矩阵**：表示一个二元关系的有向图，其中元素为 0 或 1。
- **传递闭包**：在图中，如果从节点 A 到节点 B 存在路径，则传递闭包矩阵对应位置为 1。

1.2. 实验原理和方法

1.2.1. 实验原理

1. **关系矩阵**：
   - 给定一个关系矩阵 R，其中 $R[i][j] = 1$ 表示节点 i 到节点 j 存在一条直接路径。
   - 如果 $R[i][j] = 0$，表示两节点间无直接路径。
2. **传递闭包**：
   - 如果节点 i 能通过若干中间节点到达节点 j，则传递闭包矩阵 T 中 $T[i][j] = 1$。
   - 初始传递闭包矩阵 T 等于输入关系矩阵 R。
3. **Warshall算法**：
   - **核心思想**：利用动态规划思想，逐步通过中间节点更新路径可达性。
   - 对于每个中间节点 k：
     - 遍历所有起点 i 和终点 j，检查是否通过 k 存在从 i 到 j 的路径：
       $T[i][j] = T[i][j] \vee (T[i][k] \wedge T[k][j])$。

1.2.2. 实验方法

1. **输入矩阵**：
   - 用户输入一个 $n \times n$ 的关系矩阵。
2. **实现Warshall算法**：
   - 使用三重循环，依次遍历中间节点 k，起点 i，终点 j，更新传递闭包矩阵。
3. **输出结果**：
   - 输出传递闭包矩阵。

1.3. 数据结构设计

1.3.1. Matrix类

1. **矩阵存储**：
   - 使用 `vector<vector<int>>` 存储矩阵数据。
   - 提供矩阵的输入、输出、转置、加法、乘法等操作。
2. **成员函数**：
   - **构造函数**：初始化矩阵大小并将元素设置为 0。
   - **Trans**：计算矩阵的转置。
   - **operator+**：逻辑加法（并集）。
   - **operator***：矩阵逻辑乘法。
   - **Input**：用户输入矩阵数据。
   - **Disp**：打印矩阵内容。
3. **代码片段**：

```
class Matrix {
private:
    vector<vector<int>> data; // 存储矩阵数据的二维vector
    size_t size;              // 矩阵大小

public:
    // 构造函数：初始化矩阵
    Matrix(const size_t input_size) {
        size = input_size;
        data.assign(size, vector<int>(size, 0));
    }

    // 输入矩阵数据
    void Input() {
        for (int i = 0; i < size; ++i) {
            for (int j = 0; j < size; ++j) {
                cin >> data[i][j];
            }
```

```
        }
    }

    // 打印矩阵
    void Disp() {
        for (int i = 0; i < size; ++i) {
            for (int j = 0; j < size; ++j) {
                cout << data[i][j] << ' ';
            }
            cout << endl;
        }
    }
};
```

1.4. 算法实现

1.4.1. Warshall算法

Warshall 算法是计算传递闭包的经典算法，通过逐步增加中间节点，更新路径的可达性。

1.4.1.1. 算法步骤

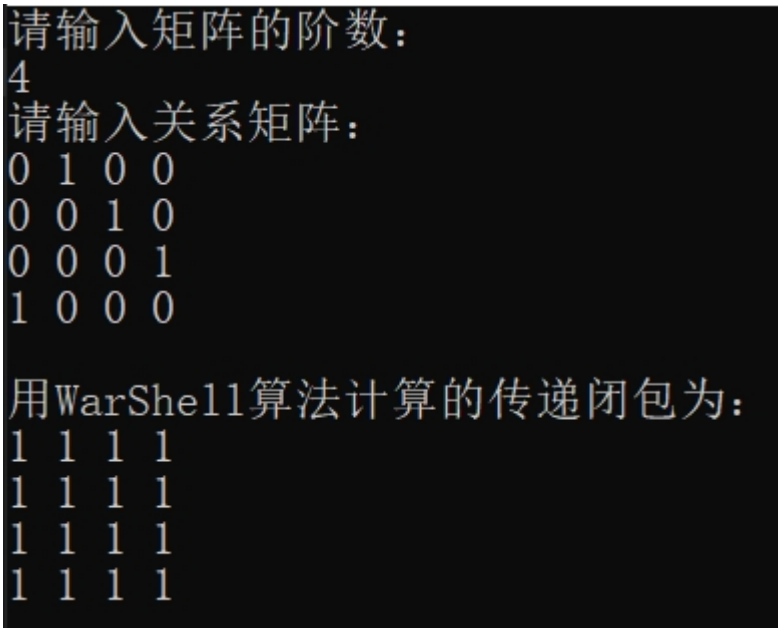
1. 初始化传递闭包矩阵 T 为关系矩阵 R。
2. 遍历所有中间节点 k。
3. 对于每对起点 i 和终点 j:
    - 如果从 i 到 j 可以通过 k 达到，则 $T[i][j] = 1$。

1.4.1.2. 代码实现

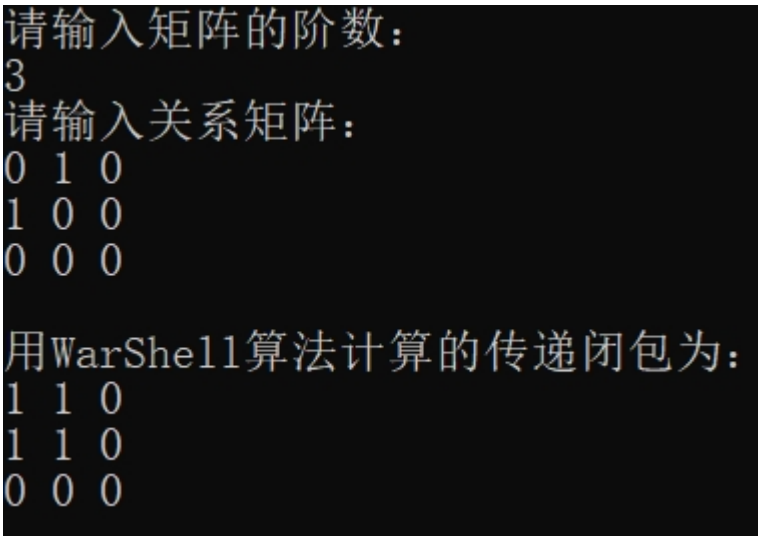
```
Matrix Warshell(Matrix& matrix) {
    Matrix result(matrix); // 初始化传递闭包矩阵
    for (int k = 0; k < matrix.Size(); ++k) {
        for (int i = 0; i < matrix.Size(); ++i) {
            for (int j = 0; j < matrix.Size(); ++j) {
                result[i][j] = result[i][j] || (result[i][k] &&
result[k][j]);
            }
        }
    }
    return result;
}
```

1.5. 测试结果

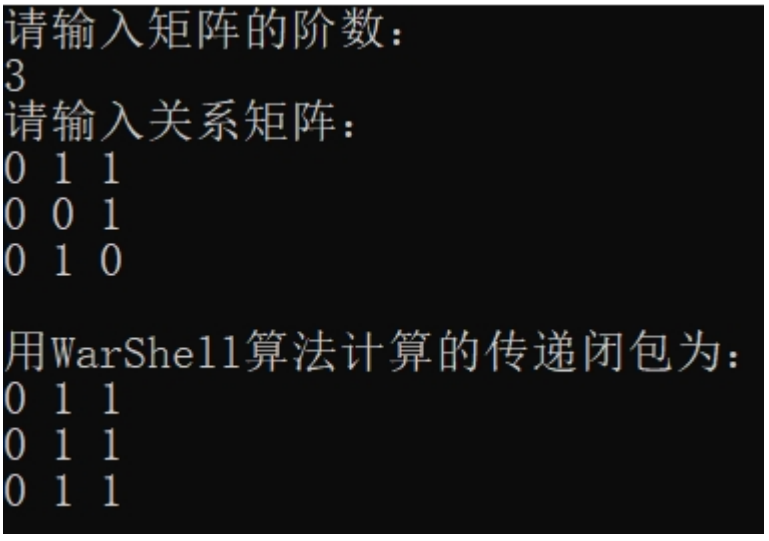
测试用例1



测试用例2



测试用例3



1.6. 项目流程图

```
+------------------------+
|       输入矩阵阶数       |
```

1.7. 心得体会

通过本实验，我深入理解了传递闭包的概念和 Warshall 算法的实现。在实验中，我学习了矩阵的逻辑运算以及如何使用动态规划思想逐步构建传递闭包。以下是我的收获：

1. **算法思想**：
   - Warshall 算法通过动态规划逐步增加中间节点，优化了路径可达性判断的效率。
2. **数据结构设计**：
   - 使用 `vector<vector<int>>` 存储矩阵数据，实现了矩阵的逻辑运算（加法、乘法）和基本操作。
3. **实践与优化**：
   - 实现过程中，我注意到可以通过减少不必要的判断优化算法效率。
   - 学会了使用面向对象编程封装矩阵操作，提升代码复用性和可读性。

本实验帮助我巩固了算法设计与矩阵运算的知识，为进一步研究图论算法奠定了基础。