《离散数学》课程实验报告3 求关系的自反、对称和传递闭包

**1、题目简介**

在离散数学和计算机科学中，关系的性质是一个核心概念，它涉及到多个方面，例如集合、逻辑和算法设计等。自反性、对称性和传递性是关系的三个基本性质，它们对于理解和描述系统中对象之间的相互关系至关重要。

定义关系的自反性、对称性和传递性。

**自反闭包：**

提供生成自反闭包的算法和示例。

**对称闭包：**

展示如何计算给定关系的对称闭包。

**传递闭包：**

分析Warshall算法，展示如何快速计算传递闭包。

**2、解题思路**

设计一个矩阵类，提示用户输入行和列然后初始化矩阵，调用select方法，选择需要进行的闭包操作，然后编写构造自反，传递，对称关系矩阵的函数。具体的构造方法参考书上的公式。

1、自反是将矩阵R∪

2、传递采用定义来做，传递闭包=R1+R2+R3+……+Rn.

3、对称是将矩阵R∪

1、自反：将矩阵的主对角线全部置为1；

2、关系矩阵加上其转置矩阵得到（逻辑加）

**3、数据结构设计**

这个作业的核心数据是矩阵，是一个典型的二维结构，所以可以用二维的vector容器来存储，在代码中我定义了一个vector<vector<int>> vec 来存储矩阵，并进行矩阵的相关运算。可以方便的获取矩阵中各个点位的数据。

class Matrix

{

public:

Matrix(int row, int column);

Matrix();

void select();

void show();

void Reflexive();

void symmetry();

void transfer();

void add(Matrix& m1, Matrix m2);

void multiply(Matrix& m1, Matrix m2);

int getnum(Matrix m1, Matrix m2, int row, int column);

private:

int row;

int column;

vector<vector<int>> vec;//存放矩阵

vector < vector<int>> temp;

};

**4、核心算法**

void Matrix::Reflexive()//自反矩阵算法

{

temp= vec;

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

temp[i][i] = 1;

show();

select();

}

void Matrix::symmetry()//对称矩阵算法

{

temp = vec;//用临时的一个矢量来运算

for (int i = 0; i < row; i++)//先得到vec的逆

for (int j = 0; j < column; j++)

temp[j][i] = vec[i][j];

for (int i = 0; i < row; i++)//然后将vec和vec的逆逻辑相加

for (int j = 0; j < column; j++)

{

temp[i][j] = temp[i][j] + vec[i][j];

if (temp[i][j] > 1)

temp[i][j] = 1;

}

show();//展示对称闭包的关系矩阵

select();

}

void Matrix::transfer()

{

Matrix mtemp=\*this;

for (int i = 2; i <=row; i++)

{

Matrix m1 = \*this;//复制一个

for (int j = 2; j <= i; j++)

{

Matrix m2 = \*this;

multiply(m1, m2);//m1\*m2

}

add(mtemp, m1);

}

temp = mtemp.vec;

show();

select();

}

void Matrix::add(Matrix& m1, Matrix m2)

{

for (int i = 0; i < m1.row; i++)

{

for (int j = 0; j < m1.column; j++)

{

m1.vec[i][j] = m1.vec[i][j] + m2.vec[i][j];

if (m1.vec[i][j] > 1)

m1.vec[i][j] = 1;

}

}

}

void Matrix::multiply(Matrix& m1, Matrix m2)

{

Matrix temp=m1;

for (int i = 0; i < m1.row; i++)

{

for (int j = 0; j < m1.column; j++)

{

m1.vec[i][j] = getnum(temp, m2, i, j);

}

}

}

int Matrix::getnum(Matrix m1, Matrix m2,int row,int column)

{

int sum = 0;

for (int i = 0; i < m1.row; i++)

{

sum += m1.vec[row][i] \* m2.vec[i][column];

}

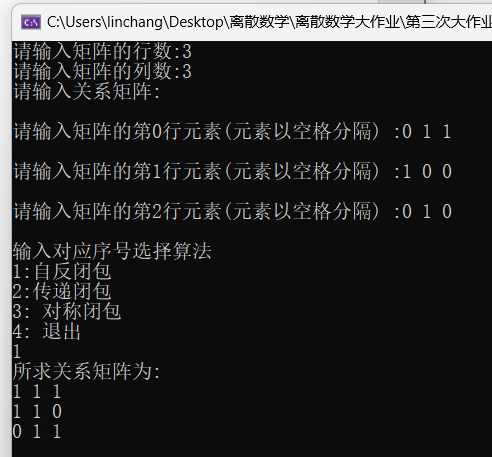
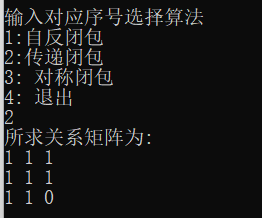
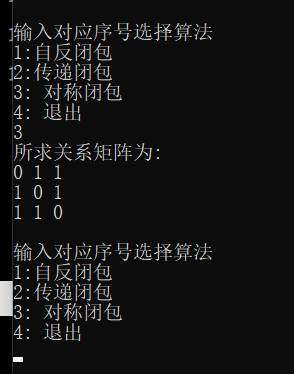
if (sum >= 1)

return 1;

else

return 0;

}**结果展示：**

**** ****

**5、心得体会**

理论与实际的桥梁：开始时，我对关系的自反、对称和传递性有了基本的理解，但在深入研究它们的具体应用时，我才真正意识到这些性质在计算机科学和其他领域的广泛应用。这提醒我，理论知识不仅仅是抽象的概念，当正确应用时，它们具有强大的实用价值。

细节决定成败：在算法实现阶段，我遇到了许多预料之外的挑战。某些理论在实际操作中并不容易实现，需要反复调试和优化。这教会了我，即使是最小的细节，也可能对项目的成功产生重大影响。

持续学习的必要性：尽管我之前已经有了一些离散数学的背景知识，但在项目进行中，我仍然遇到了许多我之前不熟悉的问题。我必须不断学习和研究，以确保项目的顺利进行。