《离散数学》课程实验报告

3-求关系的自反、对称和传递闭包

一、题目原理简介

定义：

包含给定的元素，并且具有指定性质的最小的集合,称为关系的闭包。设 R 是非空集合 A 上的关系，R 的自反（对称、传递）闭包是 A 上的关系 R'，且 R'满足以下条件：

①R'是自反（对称、传递）的；

②R 包含于 R';

③对于 A 上任何包含 R 的自反（对称、传递）关系 R"都有 R'包含于 R"。

定理：①自反闭包 $r(R)=R\cup R^0$（自反闭包关系矩阵的对角线全为 1）；

②对称闭包 $s(R)=R\cup R^{-1}$（对称闭包关系矩阵关于主对角线对称）；

③传递闭包 $t(R)=R\cup R^2\cup R^3\cdots\cdots$

本项目要求引导用户输入关系矩阵的行数以及内部各元素，形成 n*n 的关系矩阵，选择算法（1-自反闭包、2-传递闭包 3-对称闭包、4-退出），系统输出想要求得的闭包关系矩阵或退出程序。

二、解题思路

2.1 相关变量的声明与类型选定

仔细阅读题目并进行思考，本项目需要用到的变量主要有以下几个：

①R 的关系矩阵 Matrix[ROW][COLUMN]，为一个 bool 型的二维全局数组，并且大小为 ROW*COLUMN。其中 ROW 为宏定义的 100，COLUMN 同样为宏定义的 100（因为关系矩阵必须为方阵）。bool 型限制了该矩阵元素只能为 0 和 1，因而需要对初始的错误输入进行处理。

②键盘输入的矩阵行数 row 为一个 int 型的全局变量。该变量决定了 Matrix 实际使用部分的大小为 row*row。由于 row 必须小于宏定义 ROW，所以也要对 row 的错误输入或者超额输入进行判定和杜绝，否则会出现溢出。

③键盘输入的选项 choice 为一个 int 型的局部变量，用以选择输出自反、对称、传递闭包矩阵或者退出程序。所以 choice 的取值在{1,2,3,4}，超出范围的取值需要进行错误处理。

2.2 求解自反闭包

根据自反闭包的定义和相关定理可知,求解自反闭包矩阵只需要在原矩阵上将对角线上的元素全都置为 1，即 Matrix[i][i]=true。

2.3 求解对称闭包

根据对称闭包的定义和相关定理可知，求解对称闭包只需遍历矩阵，将矩阵中第 i 行 j 列的元素与第 j 行 i 列的元素进行逻辑加（与和），即 Matrix[i][j]=Matrix[i][j]||Matrix[j][i]。

2.3 求解传递闭包

根据传递闭包的定义和相关定理可知，求解传递闭包只需遍历矩阵，当 Matrix[i][j]=true 时，继续遍历考察 Matrix[j][k]==true? 若是，则 Matrix[i][k]=true。符合传递的定义：若<i,j>属于关系，且<j,k>属于关系，则令<i,k>也属于关系。

三、代码与运行结果

```
/*2151133 孙韩雅*/
#include <iostream>
using namespace std;
#define ROW 100//宏定义矩阵的行数最大值
#define COLUMN 100//宏定义矩阵的列数最大值
bool Matrix[ROW][COLUMN] = { 0 };
int row = 0;//方阵的行数（列数）

void output(bool Matrix[][COLUMN]) {//输出关系矩阵
	cout << "所求关系矩阵为:" << endl;
```

```
	for (int i = 0; i < row; i++){
		for (int j = 0; j < row; j++)
			cout << Matrix[i][j] << ' ';
		cout << endl;
	}
}
void reflexive(bool Matrix[][COLUMN]) {//求自反闭包
	//将对角线元素置为1
	for (int i = 0; i < row; i++)
		Matrix[i][i] = true;
	output(Matrix);
}
void symmetric(bool Matrix[][COLUMN]) {//求对称闭包
	for (int i = 0; i < row; i++) {//原矩阵和对称矩阵相加
		for (int j = 0; j < row; j++)//逻辑加
			Matrix[i][j] = Matrix[i][j] ||
Matrix[j][i];
	}
	output(Matrix);
}
void transitive(bool Matrix[][COLUMN]) {//求传递闭包
	int i = 0, j = 0, k = 0;//循环控制变量
		for (i = 0; i < row; i++) {
			for (j = 0; j < row; j++) {
				if (Matrix[i][j]) {//若<i, j>属于关
系，进入语句
					for (k = 0; k < row; k++) {
						if (Matrix[j][k])//若
<j,k>也属于关系，进入语句
							Matrix[i][k] =
true;//令<i,k>也属于关系
					}
				}
			}
		}
	output(Matrix);
}
void solution(bool Matrix[][COLUMN]) {
PART1:
	cout << "请输入矩阵的行(列)数（正整数且不超过100）:";
	cin >> row;
	if (cin.fail() || row < 0 || row>100) {
		cout << "输入错误，请重新输入！" << endl << endl;
		cin.clear();
		cin.ignore(100, '\n');
		goto PART1;
	}
	int i = 0, j = 0;//循环控制变量
	for (i = 0; i < row; i++){
		PART2:
		cout << endl;
		cout << "请输入关系矩阵的第" << i + 1 << "行元素
(0或1，元素以空格分隔）:";
		for (j = 0; j < row; j++) {
			cin >> Matrix[i][j];
			if (cin.fail()) {
				cout << "输入错误，请重新输入该行元
素！" << endl;
				cin.clear();
				cin.ignore(100, '\n');//清空缓冲区
				goto PART2;//重新输入该行元素
			}
		}
	}
	cout << endl;
	cout << "1:自反闭包" << endl;
	cout << "2:对称闭包" << endl;
	cout << "3:传递闭包" << endl;
	cout << "4:退出" << endl;
	int choice = 0;//用于存储用户键盘输入的操作选项
	while (choice != 4) {
		cout <<endl<< "输入对应序号选择算法：" << endl;
		cin >> choice;
		if (cin.fail() || choice > 4 || choice < 1) {
			cout << "输入错误，请重新输入！" << endl;
			cin.clear();
			cin.ignore(100, '\n');//清空缓冲区
			continue;
		}
		switch (choice) {
		case 1:
			reflexive(Matrix);
			break;
		case 2:
			symmetric(Matrix);
			break;
		case 3:
			transitive(Matrix);
```

```
                break;
            }
        }
        cout << "程序已退出！" << endl;
}

int main(){
        solution(Matrix);
        return 0;
}
```

运行结果：

```
请输入矩阵的行(列)数（正整数且不超过100）:3

请输入关系矩阵的第1行元素(0或1，元素以空格分隔）:1 0 1

请输入关系矩阵的第2行元素(0或1，元素以空格分隔）:0 0 1

请输入关系矩阵的第3行元素(0或1，元素以空格分隔）:0 1 0

1:自反闭包
2:对称闭包
3:传递闭包
4:退出

输入对应序号选择算法：
1
所求关系矩阵为:
1 0 1
0 1 1
0 1 1

输入对应序号选择算法：
2
所求关系矩阵为:
1 0 1
0 1 1
1 1 1

输入对应序号选择算法：
3
所求关系矩阵为:
1 1 1
1 1 1
1 1 1

输入对应序号选择算法：
4
程序已退出!
```

四、体会与心得

在了解闭包的概念以及自反（对称、传递）闭包的求解办法和矩阵特征之后，放到代码上求解相关矩阵还是比较简单的。根据题目的要求，我也能在脑海中大概构思出需要的几个重要变量，以及每一个闭包求解的过程。所以本题对于我而言还算是比较能轻易完成的。

当然完成的难度不大，但是在细节处理以及清洁代码、增强健壮性的过程仍需要精益求精。老师给的代码从正常人的逻辑思维角度完成了求解，但是较为繁琐，有些地方可以进行大量的删减，包括变量的开辟也较多，其实从头到尾只需要用到上述的三个变变量和最多三个循环控制变量就足够。所以我根据我的进一步思考，将变量的使用所见到最少，也给代码的占存减少了许多。在删减变量的同时，我还对变量的命名作了更详细的补充，有些变量仅仅用字母代替，可读性不强，也容易混淆，所以我在理解的基础上将变量改成了我喜欢并且易懂的名称。

最后示例代码遗漏了对于输入错误的判断，在本题输入内容较多的情况下健壮性较低，所以我对应于每个变量的输入要求进行了增补，使得代码能够容许输入错误的发生。