离散数学

项目说明文档

**Warshell算法求传递闭包**

同济大学

Tongji University

姓名： 林觉凯

学号： 2253744

指导老师： 李冰

学院专业： 软件学院 软件工程

1. **实验介绍**

本题是要以C++程序的方式实现用Warshell算法完成传递闭包的构建。

**2.实验要求**

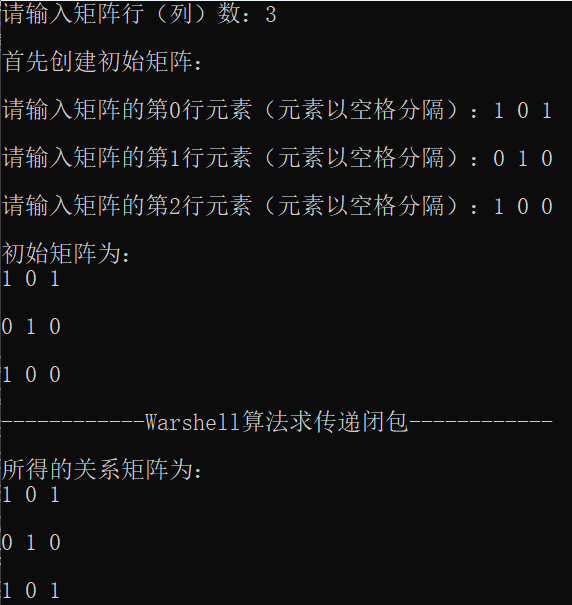
**2.1 输入要求**

要求用户输入关系矩阵的行数（列数），再通过用户指定的矩阵大小提示用户当前应该输入某一行的矩阵具体内容元素(在输入的每一个过程中都需要做相应的错误处理来保证代码的健壮性)。

**2.2 输出要求**

通过Warshell算法求出用户输入的关系矩阵的传递闭包。

**2.3 项目示例**



1. **数据结构设计**

对于存储关系矩阵，我们利用二维数组这一数据结构。但是本题不是采取写死的二维数组大小设计，而是使用动态内存分配来确定最终要求的二维数组的大小并给它分配相应的空间。使用我们首先需要应该指向二维数组的指针int\*arr。arr2[i][j]表示关系矩阵中第i行第j列的元素。

同时，还需要设计row,col,i,j等循环变量来充当计算闭包这一过程中循环操作的计数器功能。

**3.项目实现**

**3.1输入部分的实现**

void Init()

{

cout << "请输入矩阵行（列）数："; //输入行（列）数，并做代码的健壮性检验

while (1)

{

cin >> n;

if (cin.good() == 0 || n < 1)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入矩阵行（列）数错误！请再次输入:" << endl;

continue;

}

break;

}

cout << endl << "首先创建初始矩阵：" << endl; //合理动态分配空间，防止空间浪费

Array = new int\* [n];

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

Array[i] = new int[n];

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

{

cout << endl << "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++)

Array[i][j] = Get\_element();

}

cout << endl;

cout << "初始矩阵为：" << endl;

Output();

}

在输入部分首先需要用户输入行数（列数），然后再根据用户输入的行数（列数）的值来为我们的二维数组动态分配相应内存，这样可以防止空间的浪费。其次就是要输入关系矩阵的具体内容，在输入关系矩阵具体内容的时候依据题目要求需要逐行输入关系矩阵的element，因此也要对输入关系矩阵的element进行错误处理，故我想到了再写一个行数叫Get\_element（如下）对输入关系矩阵的element进行统一的代码健壮性判断。

int Get\_element()

{

int retint;

while (1)

{

cin >> retint;

if (cin.good() == 0 || (retint != 0 && retint != 1))

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入错误！请重新输入！";

continue;

}

break;

}

return retint;

}

**3.2输出部分的实现**

输出部分由Output函数实现，通过遍历矩阵中的每一行每一列，输出关系矩阵内的具体内容。

void Output()

{

for (int i = 0; i < n; i++) //遍历关系矩阵并打印

{

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

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

cout << endl << endl;

}

}

**3.3 Warshell算法求传递闭包的实现**

void Warshell()

{

int col = 0;

int row = 0;

int k = 0;

for (col = 0; col < n; col++) //总共有n次遍历，col是列

{

for (row = 0; row < n; row++) //进行检查第col列是否有1存在

{

if (Array[row][col] == 1)

//若第row行col列是1，则把第col列各个元素对应加到第row行

{

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

Array[row][k] = Array[row][k] | Array[col][k]; //进行逻辑加

}

}

}

}

Warshell算法求传递闭包首先使循环变量i＝0；之后再对所有的循环变量j如果满足Array[j，i]＝1，则对循环变量k＝0，1，…，n-1，Array[j，k]＝Array[j，k]∨Array[i，k]（相加操作）；再使i加1；进行判断，如果满足i<n，则重新跳转到第二个步骤进行相应的循环操作，否则便停止。最后得出用Warshell算法得出的传递闭包。

**4.实验心得**

此次项目是使用一种方便的算法——Warshell算法来求解传递闭包的过程。相较于先前的求传递闭包的算法，我个人认为Warshell算法具有算法代码简洁、过程简单的优点。同时，Warshell算法的时间复杂度较低，因为它没有求矩阵合成运算这个高时间复杂度的步骤。但是这个算法比较抽象，在此次实验的过程中，我在网上查找了相应的实验步骤，对Warshell算法及传递闭包的更加深入的理论知识有了了解。

、

附件(源代码)：

#include <iostream>

using namespace std;

void Init(); //初始化函数

void Output(); //输出1矩阵内容函数

void Warshell(); //Warshell算法求传递闭包

void DeleteArray(); //回收空间，合理释放内存

int\*\* Array, n; //两个全局变量，二维数组指针和行（列）数

//输入矩阵函数（保证输入的矩阵元素只有0和1），代码的健壮性

int Get\_element()

{

int retint;

while (1)

{

cin >> retint;

if (cin.good() == 0 || (retint != 0 && retint != 1))

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入错误！请重新输入：";

continue;

}

break;

}

return retint;

}

//初始化函数，确定矩阵的行（列）数和具体内容

void Init()

{

cout << "请输入矩阵行（列）数："; //输入行（列）数，并做代码的健壮性检验

while (1)

{

cin >> n;

if (cin.good() == 0 || n < 1)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入矩阵行（列）数错误！请再次输入:" << endl;

continue;

}

break;

}

cout << endl << "首先创建初始矩阵：" << endl;

//合理动态分配空间，防止空间浪费

Array = new int\* [n];

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

Array[i] = new int[n];

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

{

cout << endl << "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++)

Array[i][j] = Get\_element();

}

cout << endl;

cout << "初始矩阵为：" << endl;

Output();

}

//输出矩阵具体内容的函数

void Output()

{

//遍历关系矩阵并打印

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

{

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

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

cout << endl << endl;

}

}

//Warshell算法求传递闭包

void Warshell()

{

//(1)首先使i＝0；

//(2)之后再对所有j如果满足Array[j，i]＝1，则对k＝0，1，…，n-1，Array[j，k]＝Array[j，k]∨Array[i，k]；

//(3)再使i加1；

//(4)进行判断，如果满足i<n，则重新跳转到步骤(2)进行循环操作，否则便停止

//(5)最后得出用Warshell算法得出的传递闭包

int col = 0;

int row = 0;

int k = 0;

for (col = 0; col < n; col++) //总共有n次遍历，col是列

{

for (row = 0; row < n; row++) //进行检查第col列是否有1存在

{

if (Array[row][col] == 1)

//若第row行col列是1，则把第col列各个元素对应加到第row行

{

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

Array[row][k] = Array[row][k] | Array[col][k]; //进行逻辑加

}

}

}

}

//回收刚刚分配空间的Array数组，释放内存

void DeleteArray()

{

//逐步释放空间，释放每一行的数组

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

delete[] Array[i];

}

//主函数调用各个解函数来获得最终结果

int main()

{

Init();

cout << "------------Warshell算法求传递闭包------------" << endl << endl;

Warshell();

cout << "所得的关系矩阵为：" << endl;

Output();

DeleteArray();

return 0;

}