



HARBIN INSTITUTE OF TECHNOLOGY

哈尔滨工业大学

集合论与图论实验报告

实验项目名称 Warshell 算法求传递闭包 成绩评定

实验项目类型: 研究型 实验时间: 2017/05/20

学生姓名: 班级: 学号:

学院: 计算机学院 专业: 软件工程 任课教师: 刘峰

(一) 实验背景与意义

关系的传递闭包(transitive closure of a relation)是, 集合论的基本概念之一。从数学的角度来看, 这类联系就是某个集合中元素之间存在的关系。一个二元关系可能具有某种性质, 也可能不具有这种性质。关系闭包就是使一个二元关系变成具有指定性质的关系, 并且还要满足的最小条件。

传递闭包最直接的应用是在判断大于或小于关系中, 例如已知 $a > b$, $b > c$, 自然很容易可以推测出 a 与 c 之间的大小关系。另外, 传递闭包在关系数据库中也有很多应用, 如最短路径选择, 最省工时加工流程, 谱系查询等。

(二) 实验内容

问题描述:

1. 自然语言描述

假设在中国各省份之间有些有飞机航线, 显然这些航线是有向的, 有些之间没有直接相连。假设游客在其中一个城市 A , 是否存在一条路径可以让游客到达 B 城市。

2. 数学语言描述

将有向图中的初始路径描述为邻接矩阵 A , 邻接矩阵中 $A[i, j]$ 表示 i 到 j 是否直接可达, 若直接可达, 则 $A[i, j]$ 记为 1, 否则记为 0; 两个有向图中 i 到 j 有路径表示从 i 点开始经过其他点 (或者不经过其他点) 能够到达 j 点, 如果 i 到 j 有路径, 则将 $T[i, j]$ 设置为 1, 否则设置为 0; 有向图的传递闭包表示从邻接矩阵 A 出发, 求的所有节点间的路径可达情况, 该矩阵就为所要求的传递闭包矩阵。

算法实现:

1. Warshall 算法的基本思想:

对每个结点 (从第一列开始), 找出所有具有到此结点的有向边的结点 (即该列中元素为 1 的所在行的结点), 再将这些结点所在行同该结点所在行进行逻辑加后作为这些结点所在的新行 (添加新的有向边) (反映了如果这些结点没有到其它结点的有向边, 但有到该结点的有向边, 再通过该结点间接到达其它结点, 根据传递闭包的定义, 这些结点就必然有一条有向边到达其它结点。)

设 R 是集合上的二元关系, M_r 是 R 的关系矩阵

(1) 置新矩阵 $A := M_r$

(2) 置 (列) $j := 1$

(3) 对所有的 $i (1 \leq i \leq n)$

如 $A(i, j) = 1$, 则对 $k = 1, 2, \dots, n$

$A(i, k) = A(i, k) \cup (A(i, j) \cap A(j, k))$

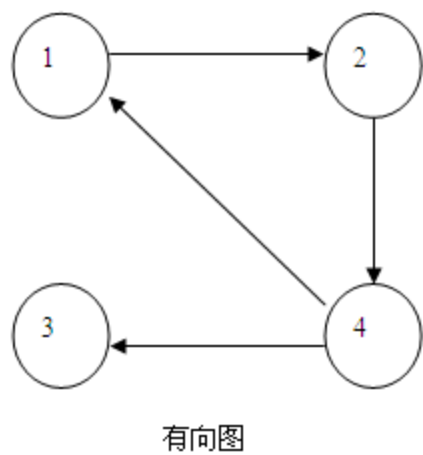
(即将 A 的第 i 行与 A 的第 j 行进行逻辑加后送回 A 的第 i 行)

(4) $j := j + 1$

(5) 如 $j \leq n$ 转 (3), 否则停止。

2. 图示：

有向图为：



由该有向图可以得到初始的邻接矩阵为：

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

邻接矩阵

那么 warshall 传递闭包算法的目的就是由邻接矩阵出发，进行探索求出最终的传递闭包：

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

传递闭包（待求）

(三) 实验环境

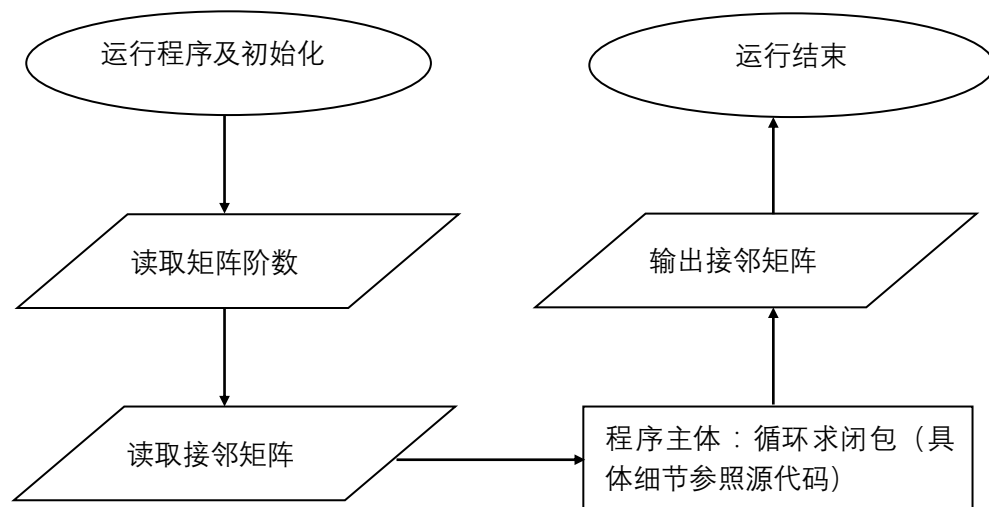
操作系统：Win10
编译器：Visual Studio Community 2015

(四) 程序设计

数据结构设计：

序号	字段名	字段含义	类型	长度	默认值	说明
1	A	接邻矩阵	Int[][] 二维数组	64 位	0	1 表示直接可达，0 表示不能直接到达

程序流程图：

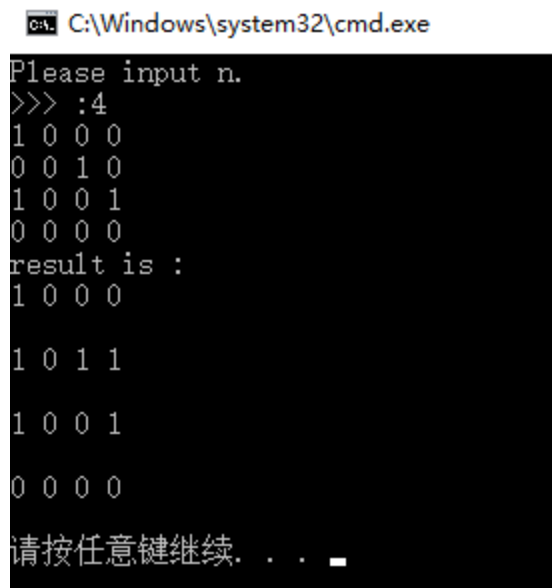


源程序的全部代码：

文件：源.c

```
1  #include<stdio.h>
2
3  main() {
4      int n, i, j, k;
5      int A[50][50] = { 0 };
6      scanf("%d", &n);
7      for (i = 0; i < n; i++) {
8          for (j = 0; j < n; j++) {
9              scanf("%d", &A[i][j]);
10             }
11         }
12     }
13     for (i = 0; i < n; i++) {
14         for (j = 0; j < n; j++) {
15             for (k = 0; k < n; k++)
16                 A[j][k] = A[j][k] || (A[j][i] && A[i][k]);
17         }
18     }
19     for (i = 0; i < n; i++) {
20         for (j = 0; j < n; j++) {
21             printf("%d", A[i][j]);
22         }
23         puts("\n");
24     }
25 }
26
27
28 }
```

（五）运行结果及评价



```
C:\Windows\system32\cmd.exe
Please input n.
>>> :4
1 0 0 0
0 0 1 0
1 0 0 1
0 0 0 0
result is :
1 0 0 0

1 0 1 1

1 0 0 1

0 0 0 0

请按任意键继续. . .
```

Warshell 算法：

时间复杂度：由于有一个三重循环，所以其时间复杂度为 $O(n^3)$

空间复杂度：由于程序中不存在递归，所以空间负载度为 $O(1)$

1. 结论

Warshall 算法适用于求解一个邻接矩阵的传递闭包矩阵。因为闭包矩阵是包含该邻接矩阵的所有传递关系的最小矩阵，若 A_1 与 A_n 相连，则必存在 $(A_1, A_{i1}) \in A, (A_{i1}, A_{i2}) \in A, \dots, (A_{ik}, A_n) \in A$ ，使得 (A_1, A_n) 属于闭包矩阵中。因此，warshall 算法对于求解传递闭包矩阵问题是可行的。