同济大学

离散数学说明文档

--求关系的自反、对称和传递闭包

姓　　名：胡正华

学　　号：2353741

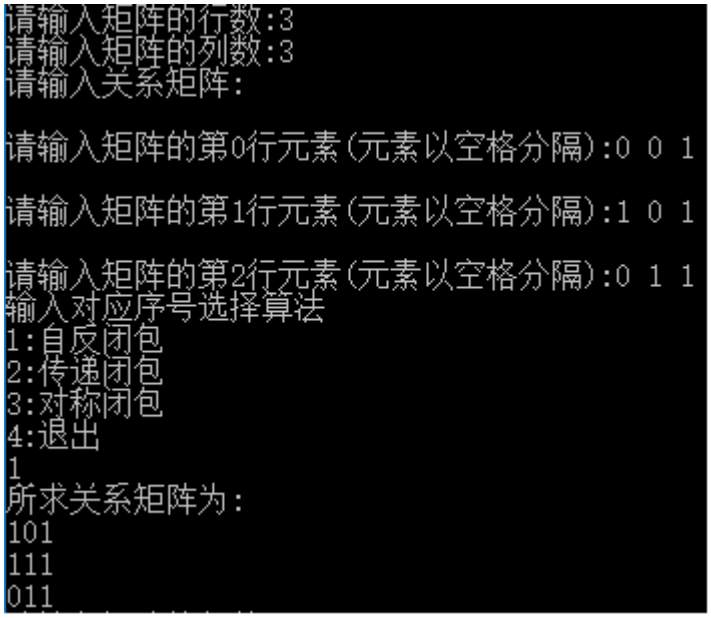
任课教师：李　冰

1. 题目简介

1.1 题目要求

根据给定的关系矩阵和用户指令，求其自反、对称、传递闭包。

1.2 题目样例



2. 解题思路

该题的核心共两部分：菜单界面实现选择和求三种闭包。

2.1 自反闭包

自反关系要求矩阵中的对角线元素都为 1。即对于所有的 iii，关系矩阵的第 iii 行第 iii 列的元素必须为 1。

判断方法：检查矩阵的每个对角元素，如果所有对角元素均为 1，则该关系是自反的。如果存在对角元素为 0，则关系不是自反的。

2.2 对称闭包

对称关系要求矩阵关于主对角线对称，即若关系中(a,b)(a, b)(a,b)存在，则(b,a)(b, a)(b,a)也应当存在。

判断方法：检查关系矩阵中的元素，如果对于每一对元素(i,j)(i, j)(i,j)，矩阵的第 iii 行第 jjj 列元素与第 jjj 行第 iii 列元素相等，则关系是对称的。如果存在不等的元素，则关系不是对称的。

2.3 传递闭包

传递闭包是指如果关系中有(a, b)(a, b)(a, b)和(b, c)(b, c)(b, c)，则应添加关系(a, c)(a, c)(a, c)。传递闭包的核心是通过“间接”路径来增强矩阵。

3. 代码实现

3.1 菜单

```
void selectMenu()
{
    cout << "请输入矩阵的行数: ";
    cin >> n;
    cout << "请输入矩阵的列数: ";
    cin >> d;
    cout << "请输入关系矩阵:\n";
    for (i = 0; i < n; i++)
    {
        cout << "请输入矩阵的第" << i + 1 << "行元素(元素以空格分隔):
";      for (j = 0; j < d; j++)
            cin >> s[i][j];
    }

    cout << "输入对应序号选择算法\n";
    cout << "1: 自反闭包\n";
    cout << "2: 传递闭包\n";
    cout << "3: 对称闭包\n";
    cout << "4: 退出\n";
    cin >> z;

    switch (z)
    {
    case 1: zifan(s); break;
    case 2: chuandi(s); break;
    case 3: duichen(s); break;
    case 4: std::exit(0); break;
    default: cout << "无效选择，请重新选择\n"; selectMenu(); break;
    }
}
```

3.2 自反闭包

```
void zifan(int s2[][MAX])
{
    for (i = 0; i < n; i++)
        s2[i][i] = 1;  // 自反闭包：将主对角线元素置为1
    output(s2);
    selectMenu();
}
```

3.3 传递闭包

```
void duichen(int s2[][MAX])
{
    int s1[MAX][MAX];
    for (i = 0; i < n; i++)
        for (j = 0; j < d; j++)
            s1[j][i] = s2[i][j];  // 求转置矩阵

    for (i = 0; i < n; i++)
        for (j = 0; j < d; j++)
        {
            s2[i][j] = s2[i][j] + s1[i][j];  // 对称闭包：矩阵加转置矩阵
            if (s2[i][j] > 1)
                s2[i][j] = 1;  // 确保元素为0或1
        }

    output(s2);
    selectMenu();
}
```

4. 心得体会

在本次实验中，我认真复习了课堂知识，熟练掌握了自反、对称和传递三种关系，同时找到其和关系矩阵的对应关系。另外在求解的过程中，我还对程序的运行效率进行了分析。我发现在求解传递闭包的过

程中，传统的算法效率并不高，这也为后续的用 Warshall 算法求解传递闭包的题目做了铺垫。