# 1 题目简介

给定表示关系的矩阵

例： 0 1 0 0

0 0 1 1

1 1 0 1

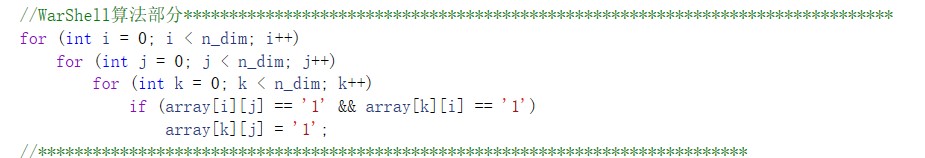
1 0 0 0

用WarShell算法求出其表示的关系的传递闭包，并用矩阵表示。

# 2 解题思路及算法分析

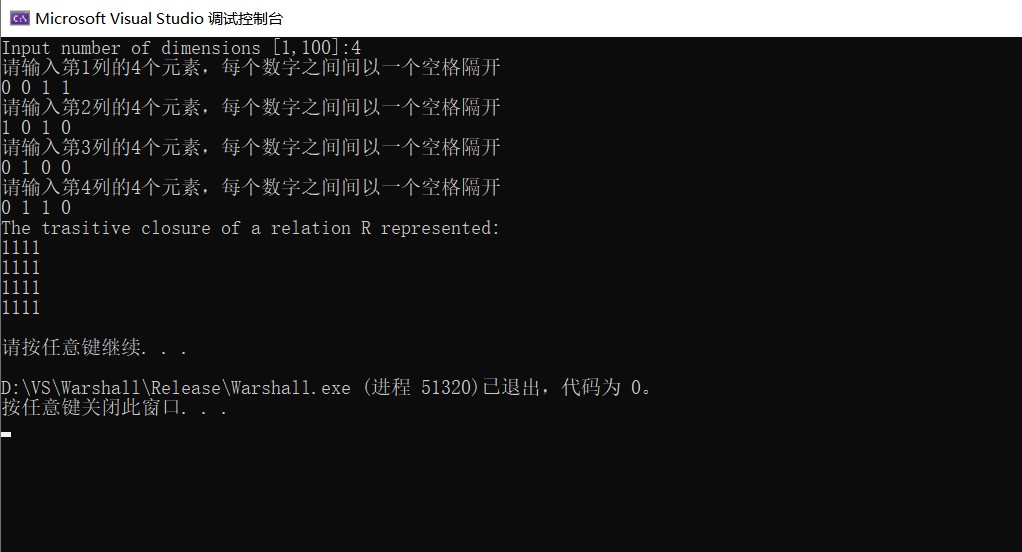
WarShell算法的核心思想是：如果顶点i与j之间联通，那么对于任意的k,如果k与i之间也连通，那么k与j之间也是连通的，事实上，这一思想也是基于“传递”概念本身的。所以，程序本质上要做的事情就是，利用二维数组模拟矩阵，并通过两层循环结构遍历关系矩阵，接着再用一层循环将k值从0遍历到n\_dim-1，模拟上述过程。当整个二维数组遍历完毕时，也就得到了传递闭包。

WarShell算法事实上用程序语言表述也十分简洁：

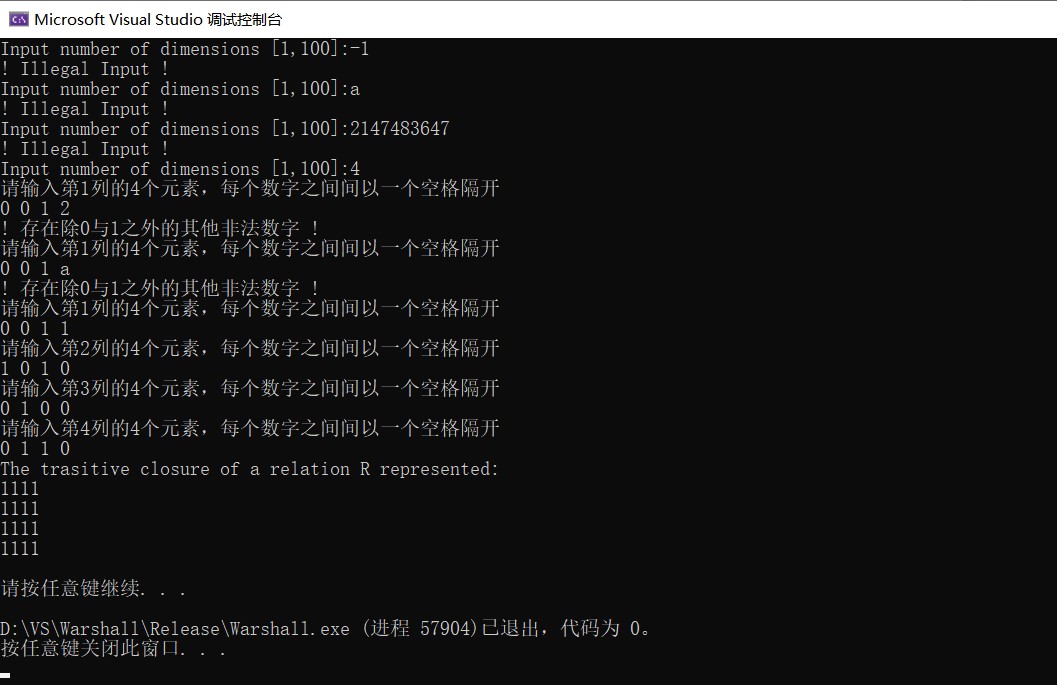


# 实验结果：

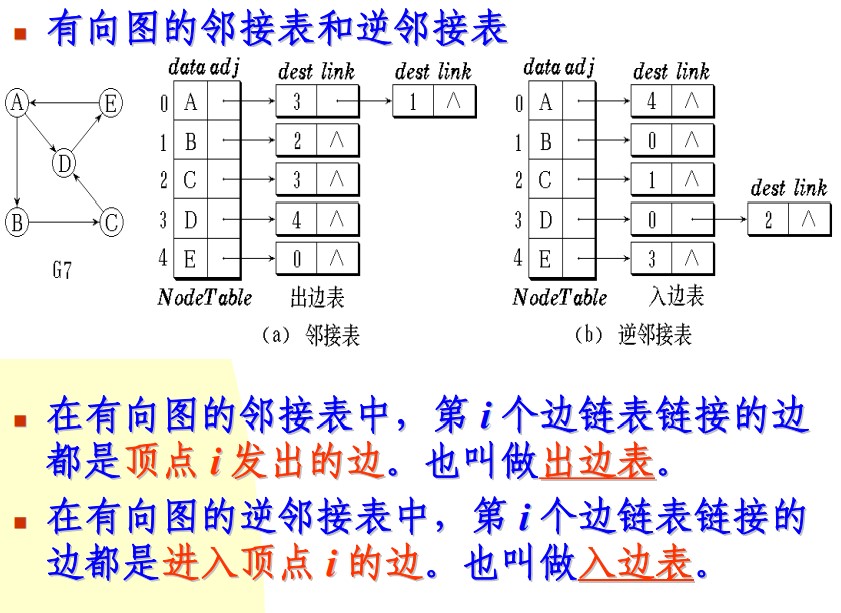
对称闭包：



此外，为了使增强程序的稳定性与健壮性，本程序中对一系列非法输入情况进行了如下处理：

1. 综合考虑程序执行时间与空间开销以及实际情况，本程序将关系矩阵上限设定在了100，如果用户输入了非数字字符或者数字是非正数或者超过了上限，程序会提示用户重新输入；
2. 由于关系矩阵中只有0、1两种数字，所以当用户意外输入非数字字符或者输入了0、1以外的数字，程序会忽略非法字符所在列的数字，提示用户重新输入该列数字。

直观上，WarShell算法通过三层循环计算传递闭包，将时间复杂度降为O(n3)，相比于普通算法的O(n4)有了一定提升，但是，三次方级的时间复杂度以及平方级别的空间开销依旧不是非常理想的状态。具体分析可以知道，前两层循环由邻接矩阵这一种图的存储形式决定，最后一层循环由遍历寻找中间节点决定，因此，在目前的邻接矩阵这一种数据结构下，并不存在很大的优化空间，这里提供基于邻接表的数据结构计算传递闭包的一种可能思路：



如图所示，利用邻接表计算传递闭包依旧需要平方级别的时间复杂度，但是，在遍历过程中，或许可以直接通过链表指针指向来实现传递关系，而无需再进行一层循环。

此外，动态内存申请是不可以直接开辟二维数组的，本程序便提供了一种典型的间接开辟二维数组的方法：先开辟n\_dim大小的指针数组，接着再开辟n\_dim个n\_dim大小的数组并分别用n\_dim的指针指向其地址，从而实现了动态内存申请开辟二维数组。当然，需要切记的一点是，动态申请开辟，即通过new关键字申请的内存空间并不能被随着程序的结束而自动释放，而需要delete关键字手动释放。