说明

1、程序使用Visual Studio 2019编写；

2、程序从文件读入数据部分使用了输入重定向，与直接输入格式相同；

3、程序测试数据读入文件已附在程序源代码及输入文件目录中；

4、搜索算法递归和非递归实现得到的搜索序列略有差异，但遍历结果中输出的生成森林相同；

5、程序通过宏定义设置最大顶点数为50；

6、程序进行了菜单设计，功能需通过选择再执行。

算法分析

一、两种建立算法的时间复杂度以及存储结构的空间占用情况分析

1、有向图邻接矩阵：

（1）时间复杂度：

初始化邻接矩阵--O(n^2)；

读入并储存顶点信息--O(n)；

读入并储存邻接边信息--O(e)；

综上，T = O(n^2 + n + e)，由e <= n^2得T = O(n^2)。

（2）空间占用情况：

无论有多少条边邻接矩阵都要占用n^2个单元，故S = O(n^2)。

2、有向图邻接表：

（1）时间复杂度：

读入并储存顶点信息、边表初始化--O(n)；

读入并储存邻接边信息--O(e)；

综上，T = O(n + e)。

（2）空间占用情况：

存储信息分为顶点数据和边数据两部分，故S = O(n + e)；

总体来看，邻接矩阵适用于稠密图，而邻接表适用于稀疏图。

二、搜索算法分析

（1）时间复杂度：

深度优先搜索和广度优先搜索整个过程均要遍历所有结点，在邻接表上搜索最坏情况要走完所有的顶点和边，在邻接矩阵上沿着各边搜索顶点的过程中最坏情况需要遍历整个矩阵。

在邻接表上搜索：T = O(n + e);

在邻接矩阵上搜索：T = O(n^2)。

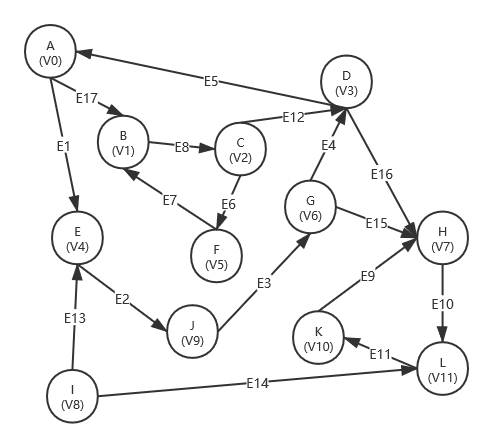
（2）空间复杂度：

广度优先搜索和深度优先搜索都要使用Visited数组，广度优先搜索还要使用队列。邻接表和邻接矩阵上搜索时这些辅助变量使用情况类似，故搜索算法的空间复杂度均为O(n)。

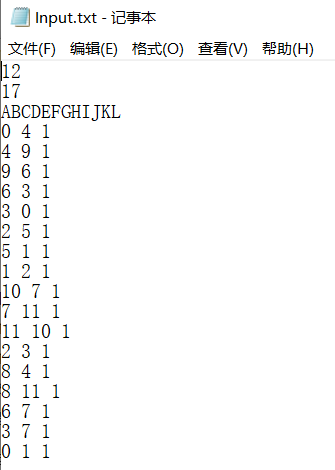
程序测试

1、输入数据

按照作业要求，自己设计了12个顶点、17条边的有向图如下图。



输入文件内容：

数据含义：

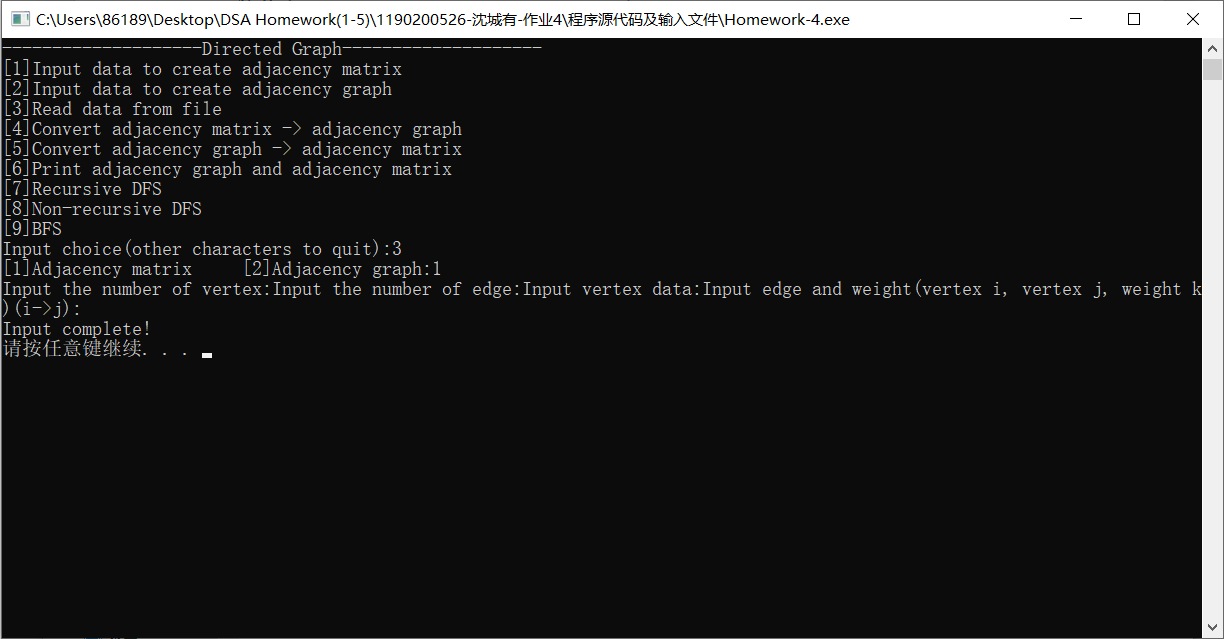
第一行、第二行：顶点数、边数；

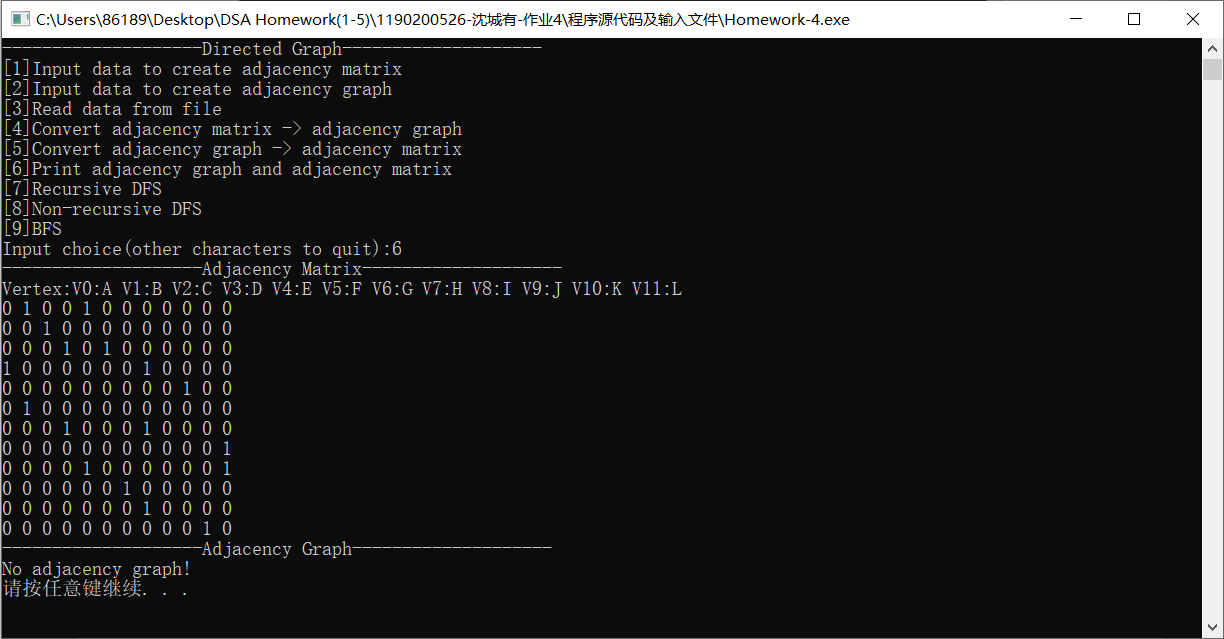
第三行：各顶点名称（或数据，程序中定义为char）；

第四行至末尾：边的出发顶点、边的指向顶点、边的权值。

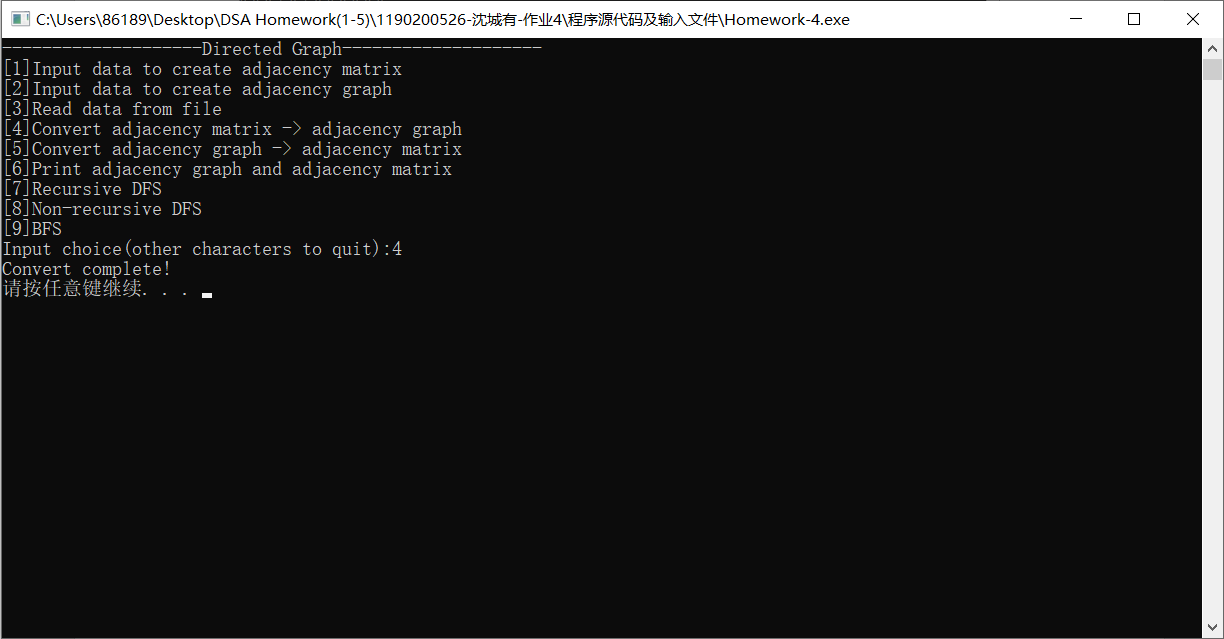
2、文件读入、读入结果显示及邻接矩阵、邻接表间转换

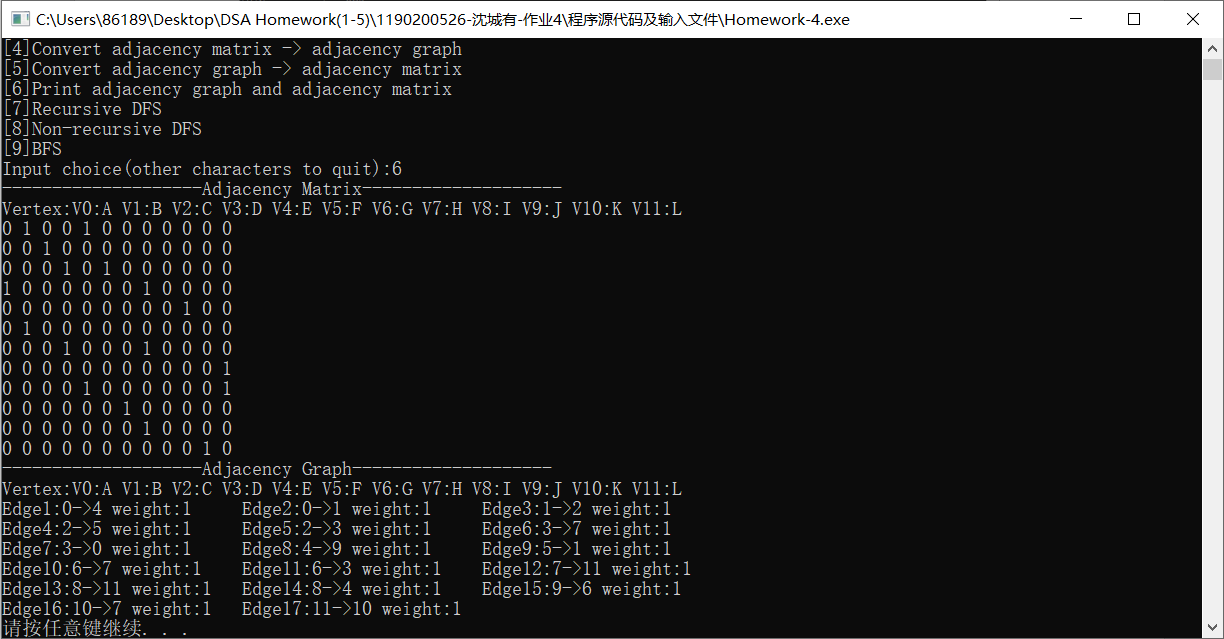
（1）读入并建立邻接矩阵



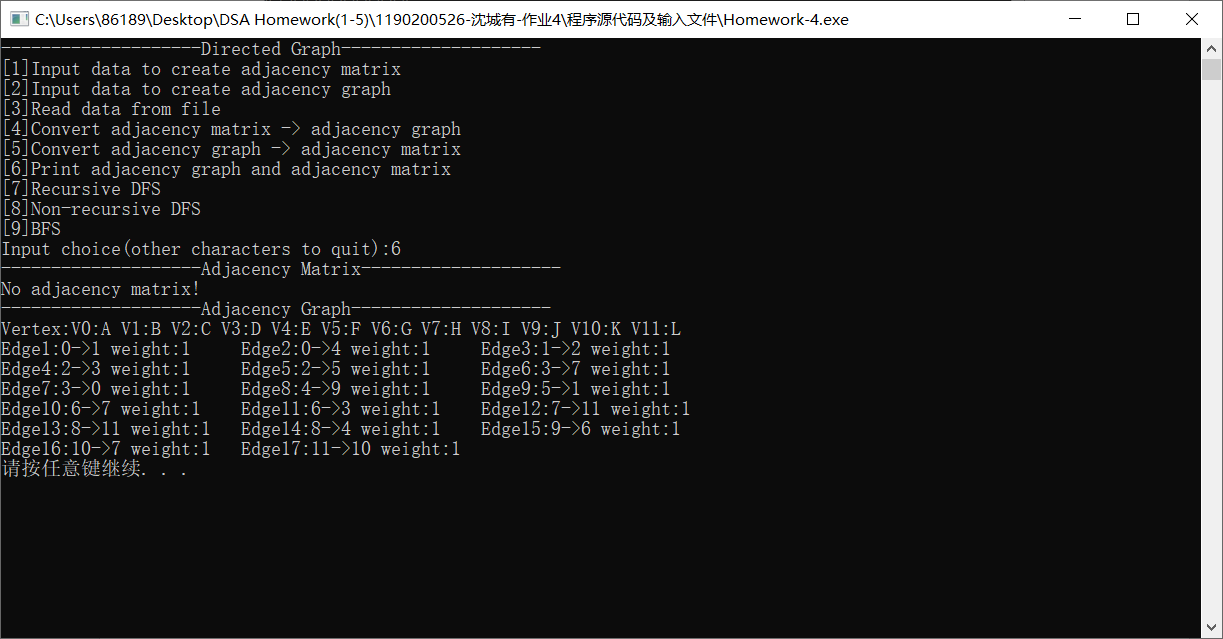
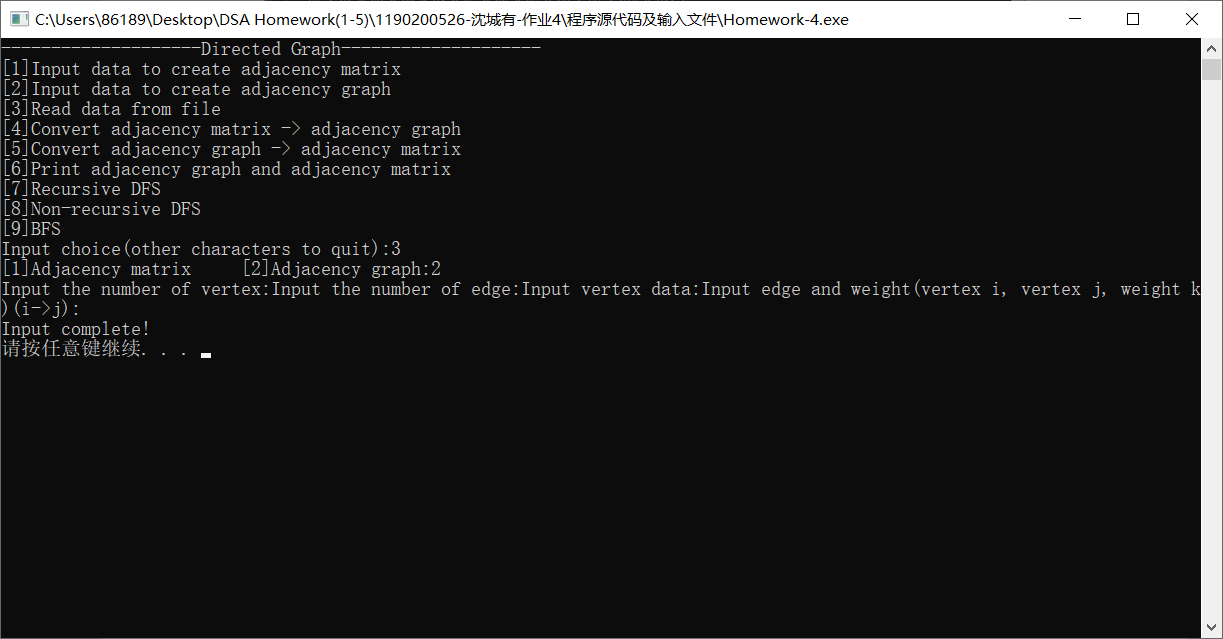


（2）邻接矩阵转换为邻接表

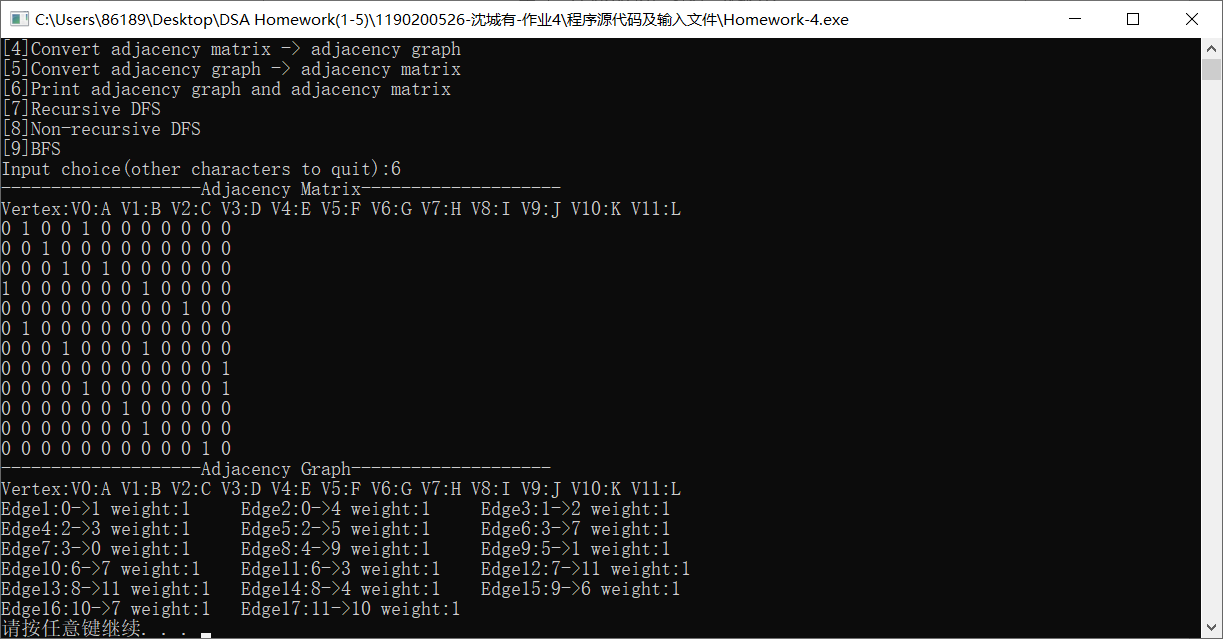
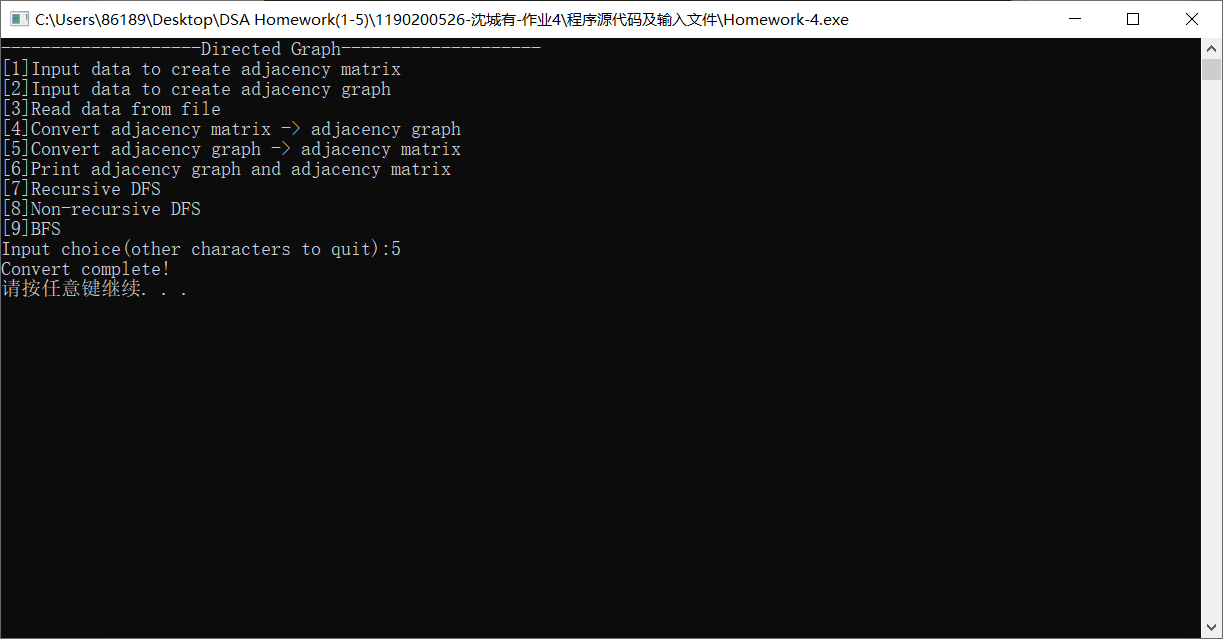




（3）读入并建立邻接表

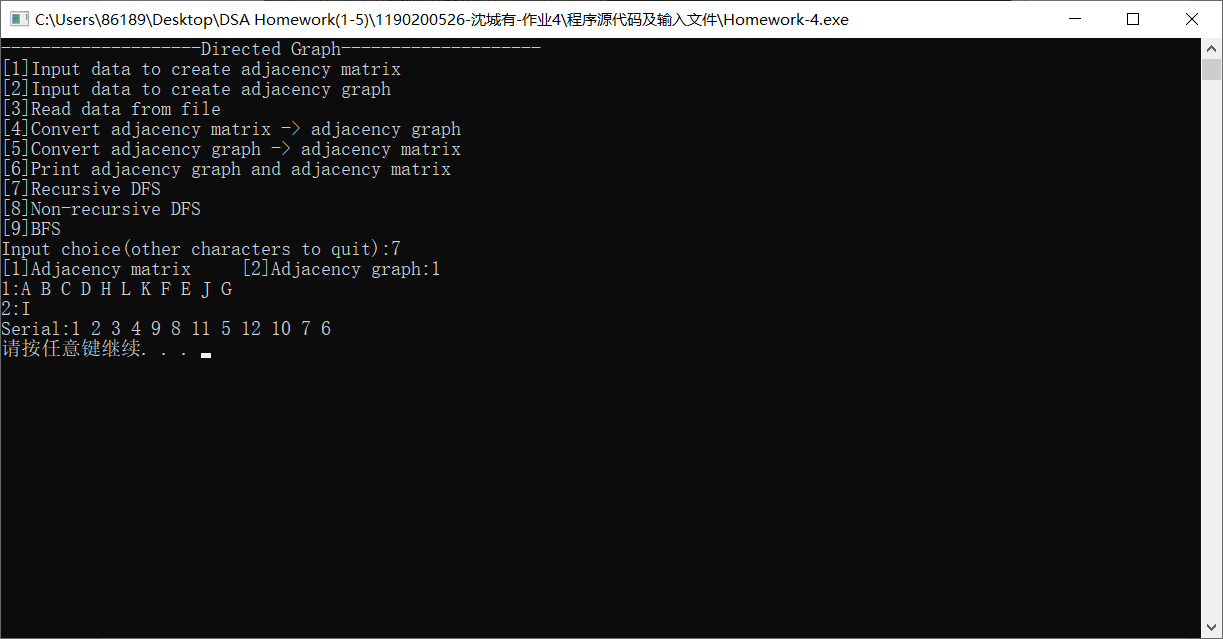


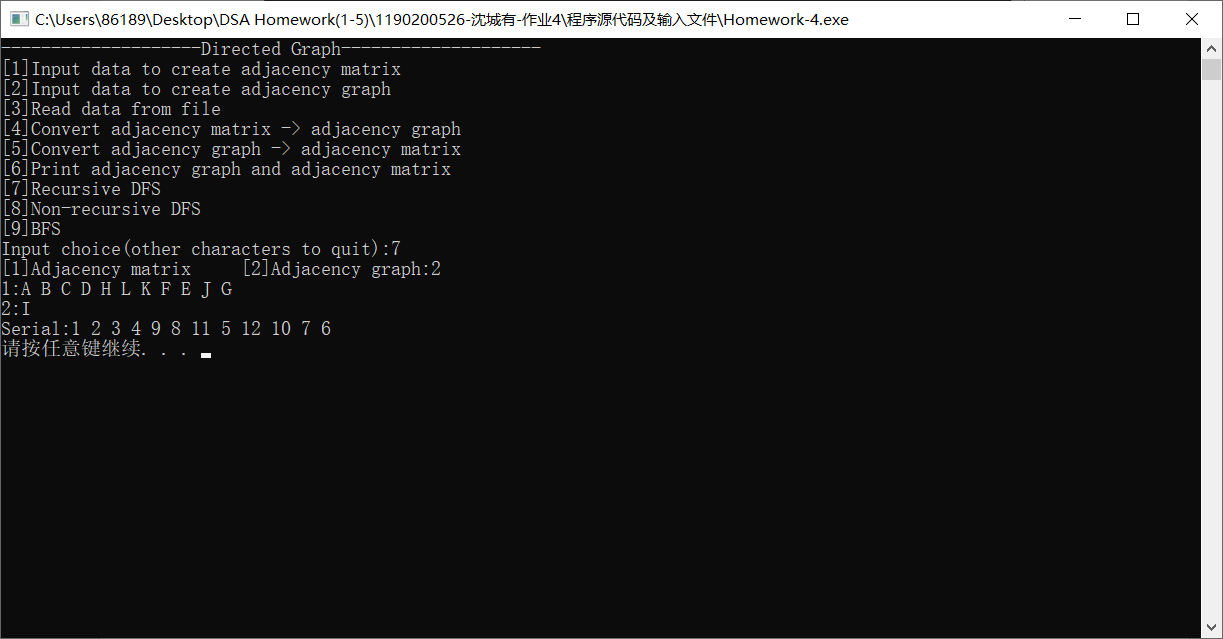
（4）邻接表转换为邻接矩阵



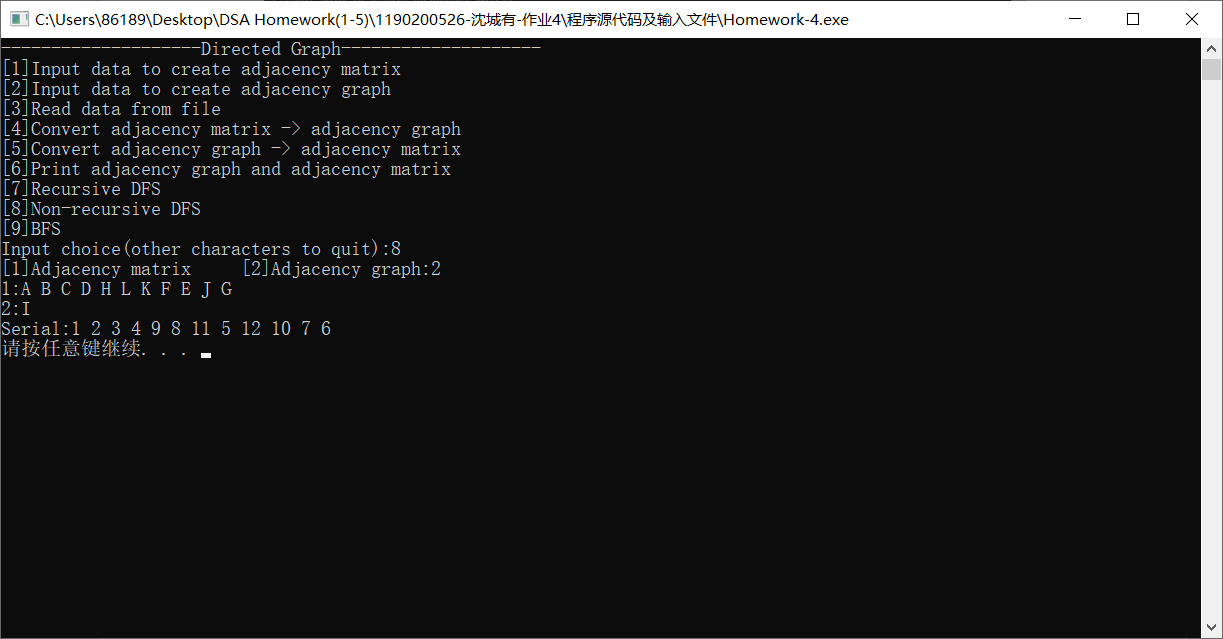
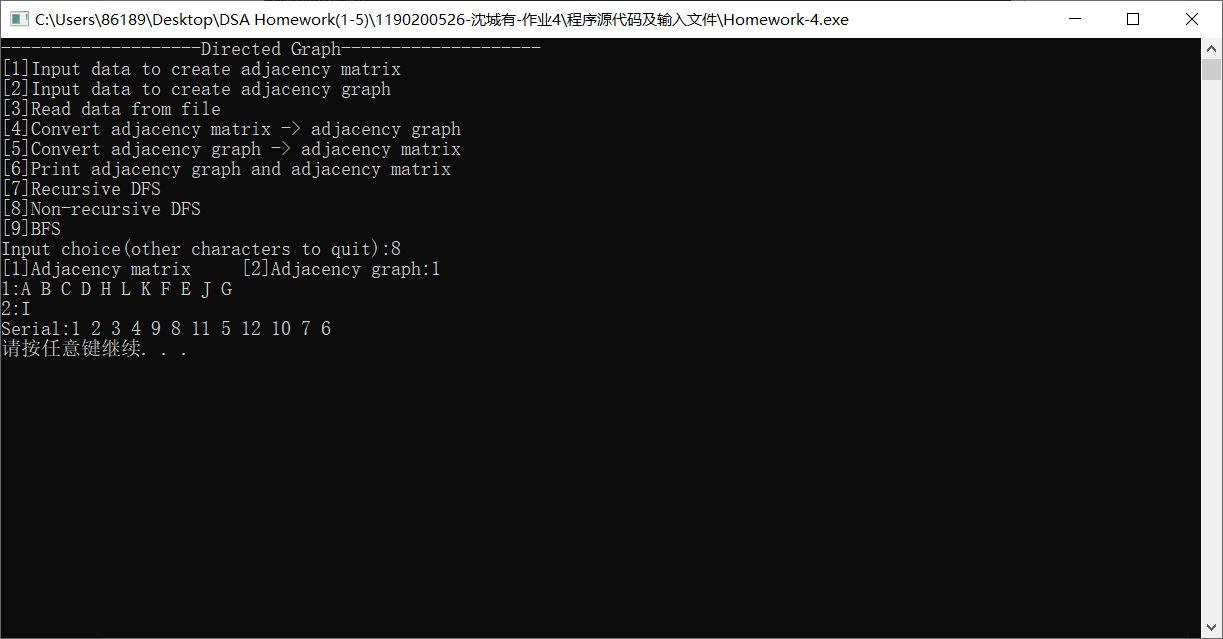
3、深度优先搜索

（1）递归

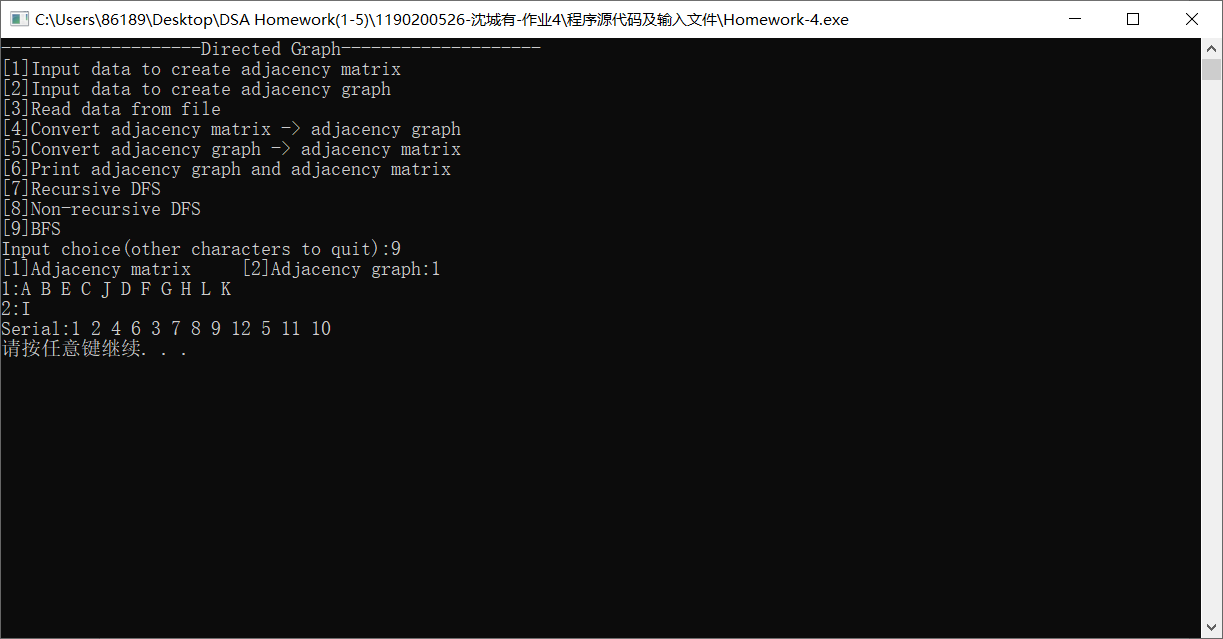


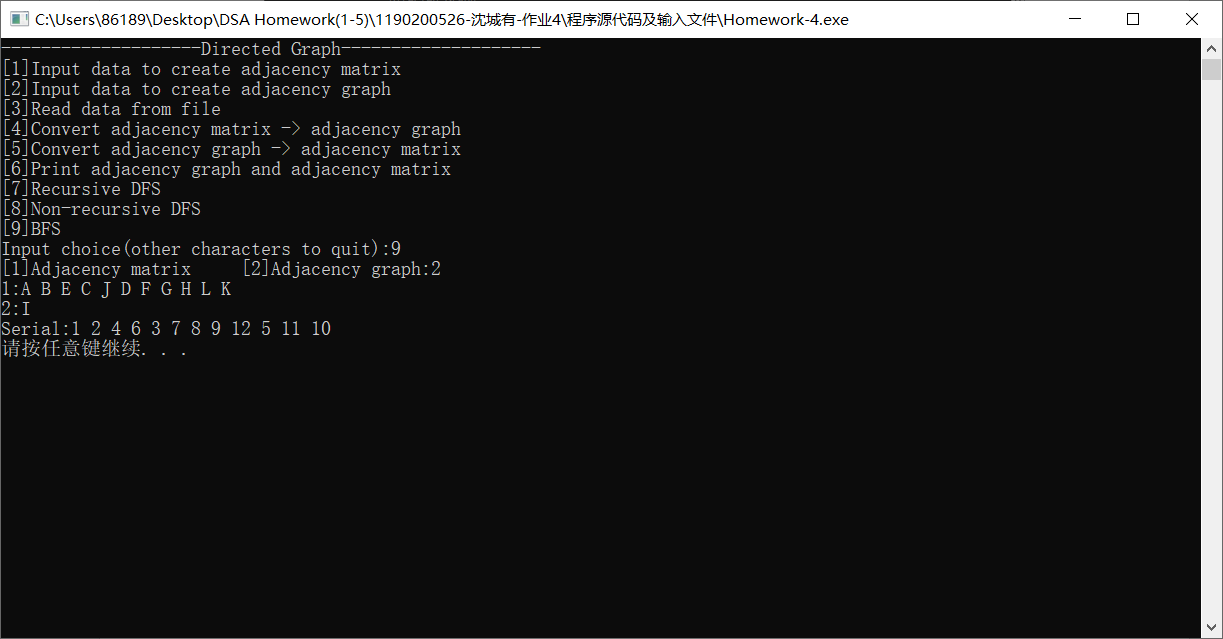


（2）非递归



4、广度优先搜索





总结：通过与图形对比分析，确定程序运行结果正确。