**绪论作业**

**一、你能结合自己的编程实践，谈一下数据结构和算法之间的关系吗？**

**数据结构和算法的关系：**

**算法**是面向问题而规定的一个有限长的操作序列，而程序是算法的具体语言实现。

而数据结构的研究内容为：

1. 数据对象的结构形式，各种数据结构的性质及其计算机中表示。
2. 每种数据结构上定义的基本操作和**算法**。
3. **算法**的效率，数据结构的应用和数据排序、查找等方面的技术。

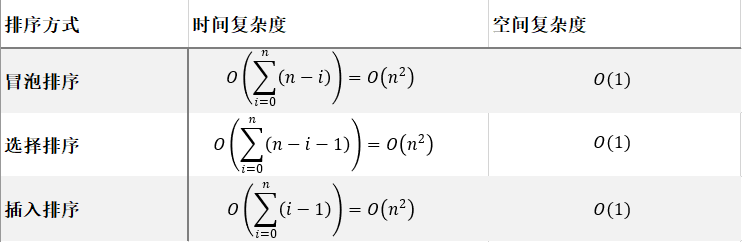
**由此可见，数据结构研究的是组织数据的方式，数据结构是算法的基础。**好的算法离不开一个优良的数据结构，一个好的数据结构可以降低算法的时间和空间复杂度，提升算法效率，同时也有可能使针对数据的一些基本操作变得简单，更容易设计出用户友好型程序，也可增强程序的健壮性等。

**二、简单排序算法**

1、编程实现(直接) 冒泡排序算法、(直接)选择排序算法和(直接)插入排序算法，分析其时间和空间复杂度。

三种排序算法的思路为：

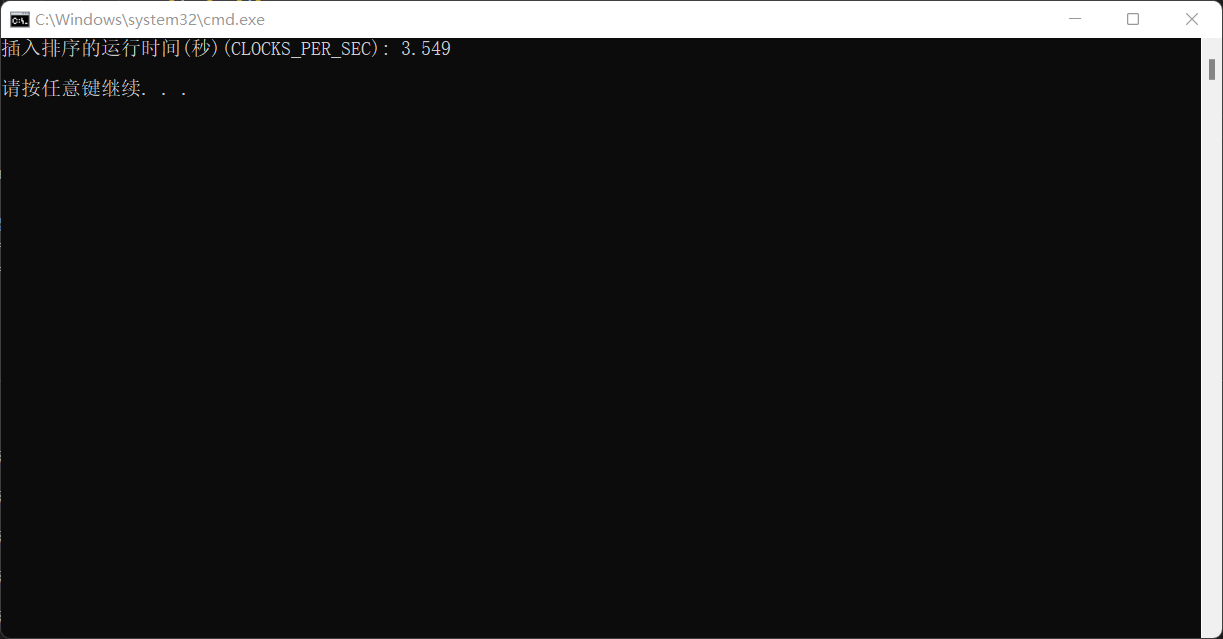
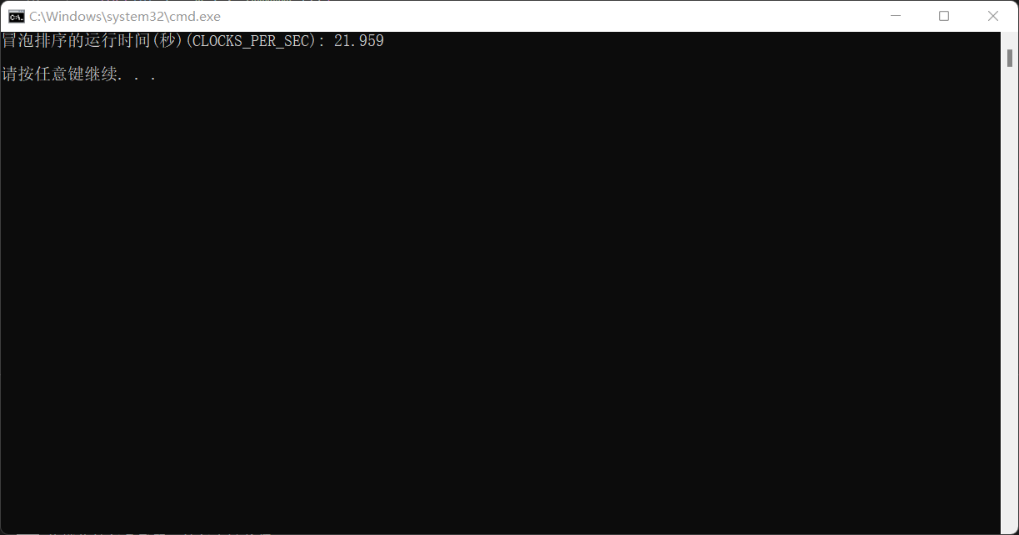
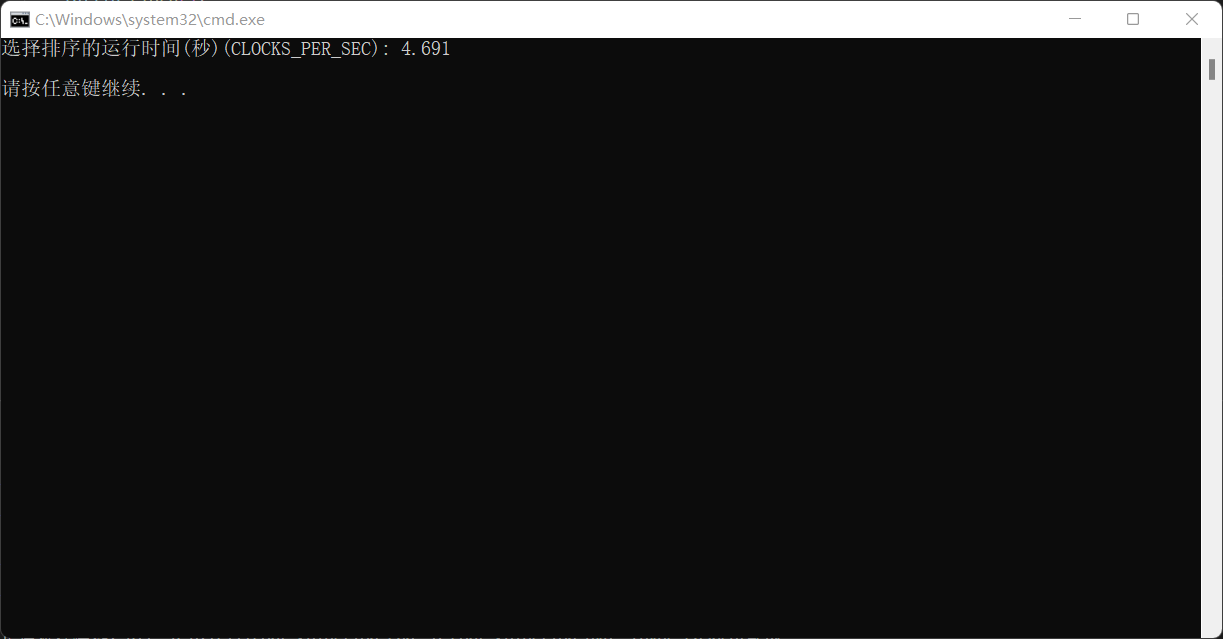
1. 冒泡排序：重复地走访过要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。
2. 选择排序：首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。以此类推，直到所有元素均排序完毕。
3. 插入排序：通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。



以随机生成的n组数据为例：由实际代码可计算最差时间和空间复杂度如下表所示。

随机生成100000组数据，通过实验的方法验证上述算法时间复杂度的分析结论。

**结果如下：**

****

由图可见，针对同一组数据，插入排序的效率最高，选择排序其次，冒泡排序最次。在时间复杂度和空间复杂度近似的情况下，运行时间却有较大区别。**原因是因为三种排序方式虽然数量级相同，但是每次循环的语句长度及其执行时间并不相同。**在较小规模的排序并不能看出差距，但在较大规模下产生的差距还是比较可观的。

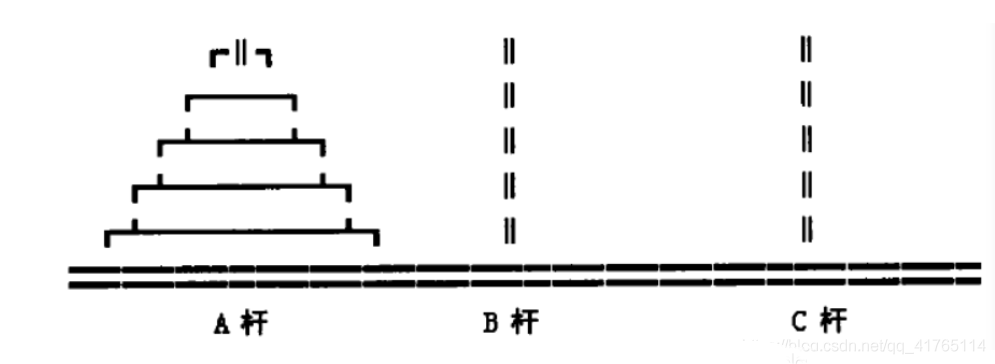
2、编写(直接) 冒泡排序、(直接)选择排序算和(直接)插入排序的**递归**算法，分析其时间和空间复杂度。

上述三种直接循环排序算法程序中均存在两层循环嵌套的情况，因而递归实现的思路主要有两种思路，**第一种**为仅针对外层循环的递归，即递归与循环嵌套；**第二种**是针对内层与外层双重循环的递归，即程序中没有循环，但针对插入排序现阶段并没有通过编程实现，原因是插入排序中的内层循环存在难以通过递归判断结束的值的比较。

由程序得出三种递归排序算法时间复杂度均为O(n2)，第一种思路的空间复杂度为O(n)，第二种思路的空间复杂度为O(n2)。

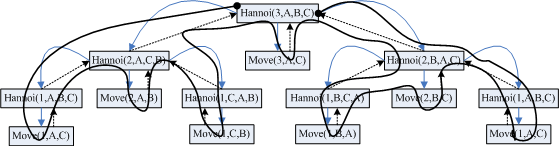
由此可以看出递归实现排序算法的弊端是，**大大提高了空间复杂度，且时间复杂度并无明显减少，甚至循环节内基本语句的执行次数还有所增加，针对上述第二种思路尤为明显。**

因而在执行较大量级数据的排序时，递归实现会出现**栈溢出**的情况。在实际执行过程中，第一种思路大概能实现30000组数据的排序，第二种思路仅能实现200组左右的数据排序。由此可见相较于直接循环排序，递归实现效率是大打折扣的。

**三、汉诺塔问题**

1、编程实现汉诺塔问题的递归算法，并分析程序的时间和空间复杂度。

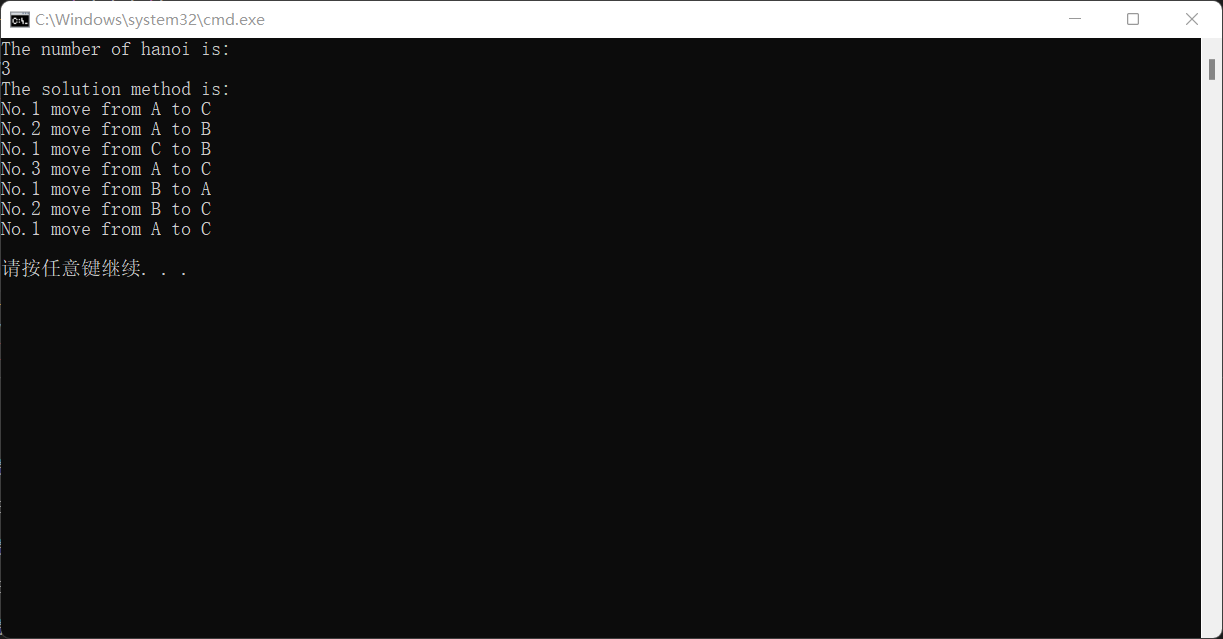
针对此汉诺塔问题的主要思路如图所示（以n=3为例，图来源于CSDN）：

1. 以C盘为中介，从A杆将1至n-1号盘移至B杆；
2. 将A杆中剩下的第n号盘移至C杆；
3. 以A杆为中介，从B杆将1至n-1号盘移至C杆。

由此来分析此程序的时间复杂度和空间复杂度，

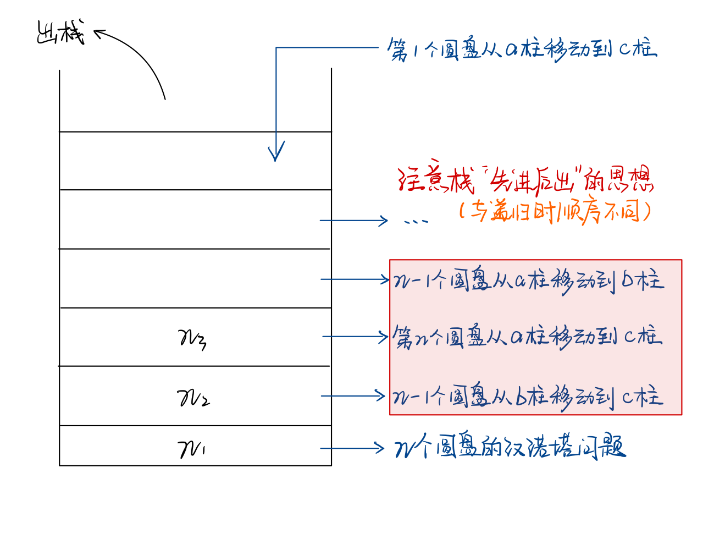
设盘子个数为n时，需要T(n)步，把A柱上n-1个盘子移到B柱，需要T(n-1)步，A柱最后一个盘子移到C柱一步，B柱上n-1个盘子移到C柱上T(n-1)步。

得递推公式T(n)=2T(n-1)+1，因此**汉诺塔问题的时间复杂度为O(2n)。而空间复杂度为O(n)，**因为运行时栈中最多会同时保存n个函数的调用信息。

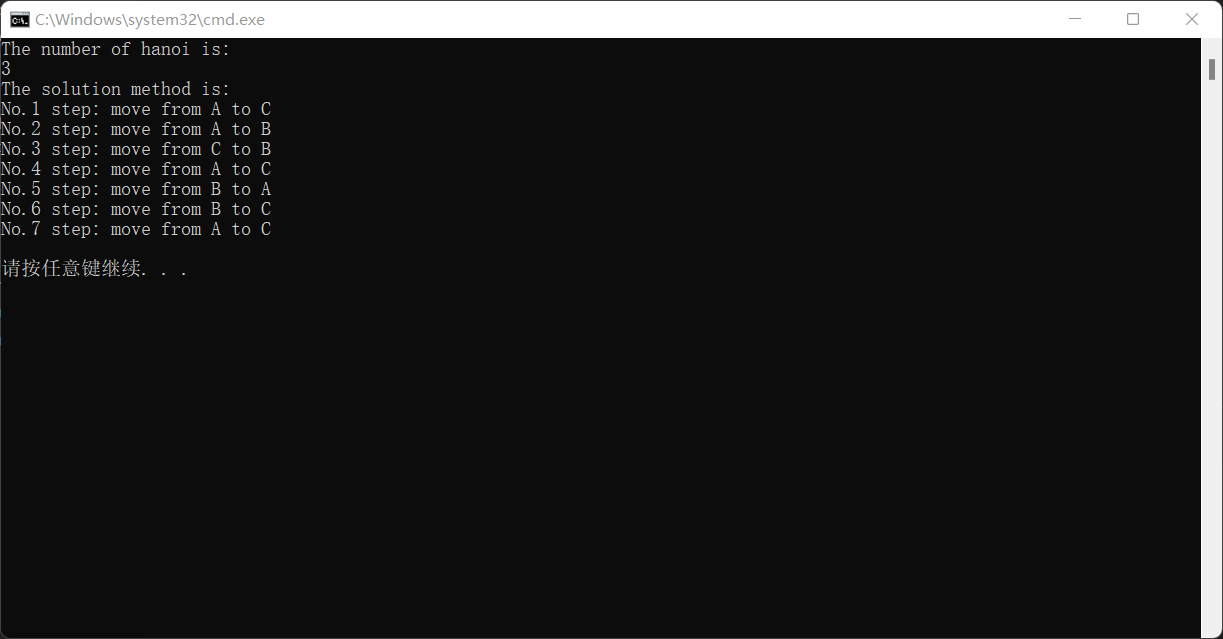
具体运行结果如下：

2、你能将汉诺塔的递归算法改写成非递归算法吗？并分析其时间和空间复杂度。

按照上述思想，在不使用递归的情况下，我们可以通过**栈**来实现汉诺塔的非递归算法，即利用栈的“先进后出”的性质，将每一步的圆盘移动操作，由后置前的**逆推**依次存入到栈中，随后依次弹出，给出解决方案，具体思路如下图。值得注意的是，在逆推的过程中由于其性质的特殊性，**进入栈的顺序与递归时恰好相反**。

然后再来分析此程序的时空复杂度，

此程序并未减少盘子移动的次数，因而时间复杂度仍为O(2n)，但由于此顺序栈相较于递归时函数间网状的相互传递结构在同一时刻的空间占用更多，数值上等于函数的调用次数，即O(2n)，执行效率上落后于传统递归法。

具体运行结果如下：

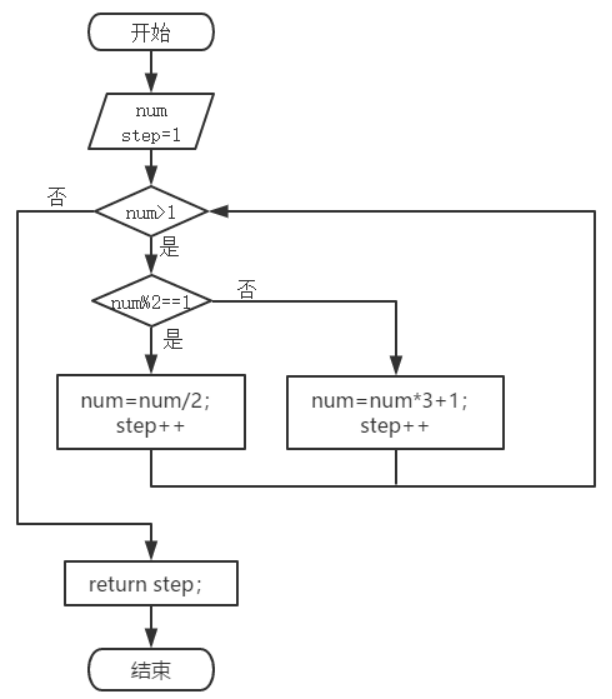
查阅资料知，**解集树法**会是更优的算法，即采用一个高度为n的二叉树的数据结构表示移动方案，从而可以计算出任意一步移动方法，且时间复杂度为O(n2n)，空间复杂度为O(1)，大大提高了算法的效率。此算法留待笔者在之后的学习中实现。

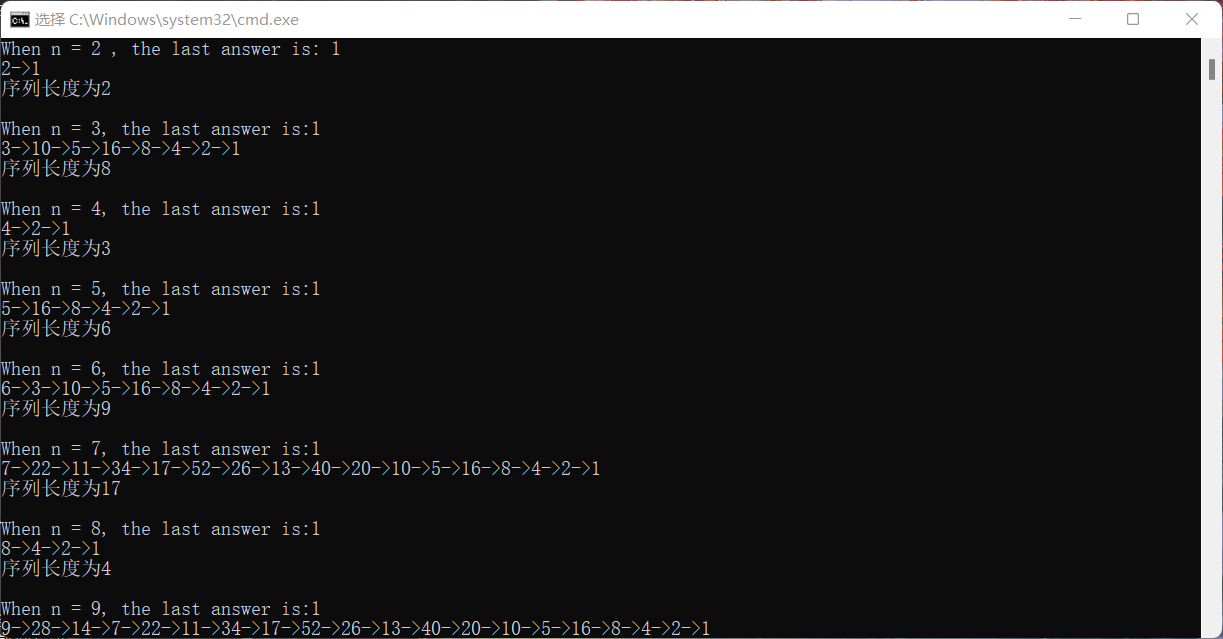
**四、 角谷猜想（冰雹猜想）**

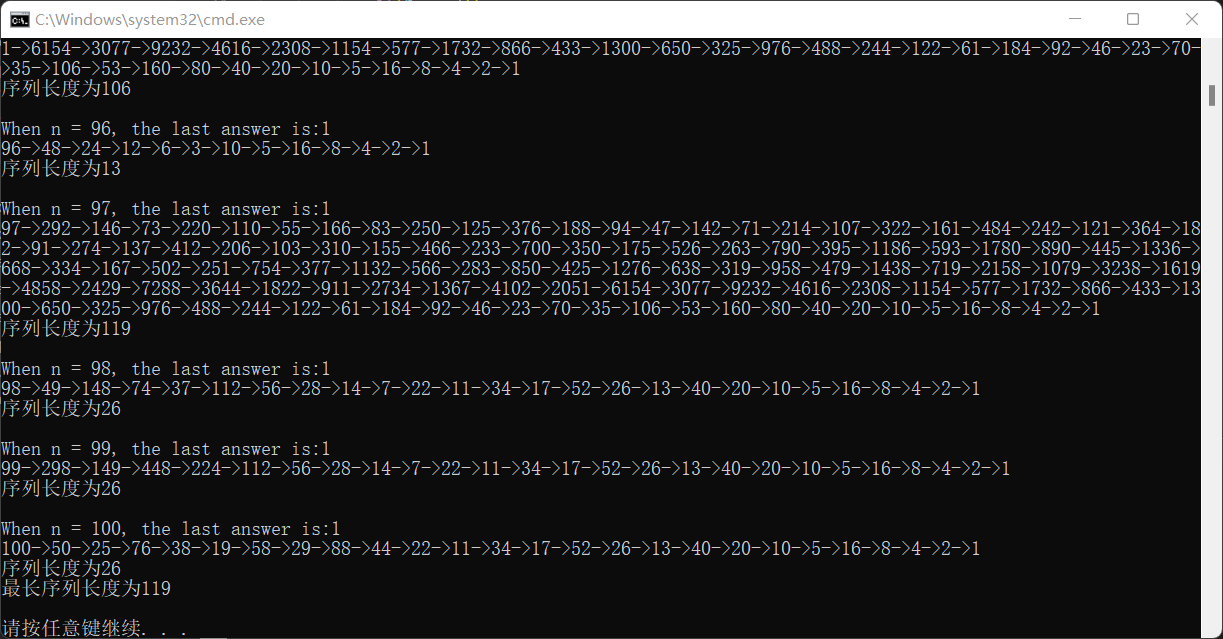
对于任何正整数n，若是偶数，则除以2；若是奇数，则乘以3加上1。这样得到一个新的整数，如果继续进行上述处理，则最后得到的数一定是1吗？

1、编写程序对100以内的整数进行测试验证，并找出其中最长的序列。

本题笔者主要通过两种算法实现了角谷猜想的验证，第一种为通过连续的循环判断其奇偶，实现对n的数据处理，并通过数组的形式存储处理过程，算法流程如图所示；第二种是通过递归的方式实现，但问题依然是空间的占用导致此算法效率略低于第一种，同时在处理验证较大数字是否满足猜想的时候会可能导致栈溢出的情况。





最终部分运行结果如下：

可得最长序列为数字97的角谷数列，长度为119（包含97数字本身），并且验证了1-100以内的整数均符合角谷猜想。