

电子科技大学

计算机专业类课程

实验报告

课程名称：数据结构与算法

学院专业：计算机科学与工程学院

学生姓名：韩博宇

学号：2019040708023

指导教师：戴波

日期：2020 年 9 月 28 日

电子科技大学

实验报告

实验一

一、实验室名称：

电子科技大学清水河校区主楼 A2-412

二、实验项目名称：

生成前 n 个自然数的一个随机置换

三、实验原理：

通过对程序语句的执行次数进行分析，计算得出算法的时间复杂度 $O(n)$

四、实验目的：

分析不同算法的时间复杂度，并通过“生成前 n 个自然数的一个随机置换”的实验进行验证。

五、实验内容：

利用三种不同时间复杂度算法来实现前 n 个自然数的一个随机置换。

六、实验器材（设备、元器件）：

硬件平台：

CPU: Intel (R) Core (TM) i7-9750H CPU @ 2.60GHz

内存：16G

软件平台：

操作系统: Windows 10

开发环境: Dev-C++ 5.11

七、实验步骤:

1. 问题描述:

①题目:

背景:

假设要生成前 n 个自然数的一个随机置换, 如 $\{4, 3, 1, 5, 2\}$ 和 $\{3, 1, 4, 2, 5\}$ 就是一个合法置换;

但 $\{5, 4, 1, 2, 1\}$ 就不是, 因为 1 出现 2 次而 3 没有。假设我们有一个随机数生成器 $\text{RandInt}(i, j)$, 它以相同概率生成 i 到 j 之间的整数, 闭区间, 可以取到 i 与 j 。

下面是三个算法。

(1) 如下填入从 $A[0]$ 到 $A[N-1]$ 的数组 A : 为了填入 $A[i]$, 生成不同于 $A[0], A[1], \dots, A[i-1]$ 之间的随机数时, 才将其填入 $A[i]$

(2) 同算法 1, 但保留一个称为 **Used** 数组的附加数组, 当一个随机数 **Ram** 最初被放入数组 A 的时候, $\text{Used}[\text{Ram}] = 1$. 这就是说,

当用一个随机数填入 $A[i]$ 时, 用 **Used** 数组来测试该随机数是否已经被使用。

(3) 首先填写数组使得 $A[i] = i + 1$; 然后

For ($i = 1; i < N; i++$)

Swap ($\&A[i], \&A[\text{RandInt}(0, i)]$)

要求实现以上三个算法的函数: (1 对应 (1) 中的算法, 请务必不要弄混)

`void RandomPermutation1(int n);`

`void RandomPermutation2(int n);`

`void RandomPermutation3(int n);`

在函数中输出为前 n 个自然数的一个随机置换, 用, 号分隔

必须使用教师提供的随机函数, 函数原型如下:

`int RandInt(int i, int j);`

在文件中加入: `extern int RandInt(int i, int j);` 即可

(在测试的时候可以使用 `random` 函数, 但是在向系统提交时必须改成调用 `RandInt` 产生随机数)

注意:

1、逗号为英文输入法中逗号;

2、输入 n 不为正整数时, 输出 **error**

3、在输出的结果后多输出 ", 0" (不含外侧引号) 表示输出结束,

除此之外, 其余任意多余输出视为错误。

例如:

1. 当参数 $n = 5$ 时, 如果数组的前 5 个元素为 1, 2, 3, 4, 5, 则在 5 后加上英文的逗号和 0,

即输出格式为

1, 2, 3, 4, 5, 0

2. 例如 $n = 10$ 时，数组的 10 个元素输出为 2, 8, 5, 1, 10, 9, 3, 6, 7, 4, 但是需在结果后增加一个 0

因此最终输出为：

2, 8, 5, 1, 10, 9, 3, 6, 7, 4, 0

② 题目分析

本题要求使用三种不同算法实现“生成前 n 个自然数的一个随机置换”。其中，第一种方法通过将新产生的随机数与已存在的 $A[0]$, $A[1]$, ..., $A[i-1]$ 比较，若与已有数都不相同才将其填入 $A[i]$ 这种方法来实现；第二种方法通过 Used 数组记录已使用过的数据，避免后面重复使用；第三种方法通过使用 Swap 函数随机交换已有数据的顺序，实现生成随机置换的目的。

题目中还规定了输入、输出要求和非法值处理要求，在实验时要关注这些问题。

2. 算法分析与概要设计

输入：一个正整数 n

输出：在函数中输出为前 n 个自然数的一个随机置换，用, 号分隔，结果后增加一个 0

输入转化为输出步骤：

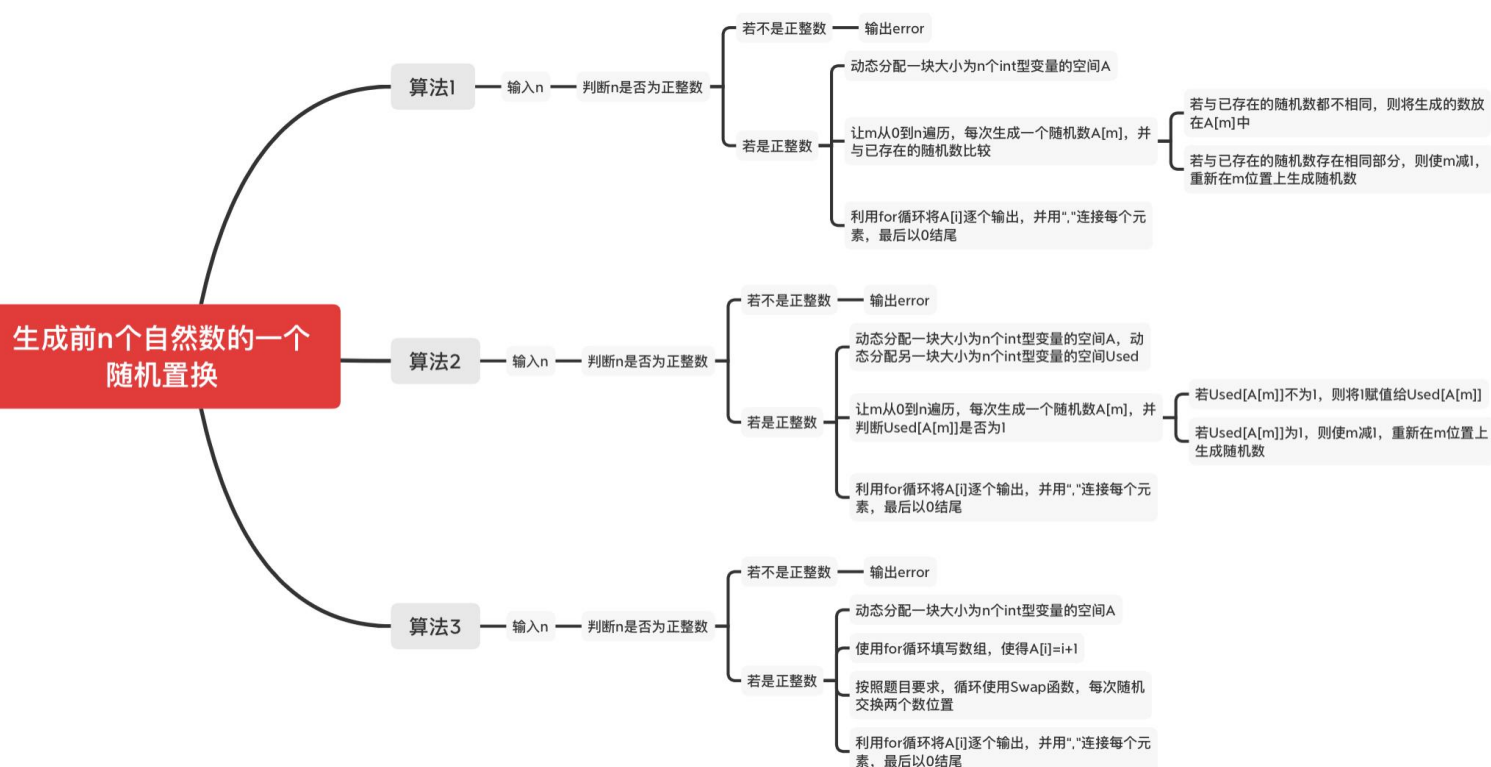


图 1 输入转化为输出步骤

3. 核心算法的详细设计与实现

①算法一流程图：

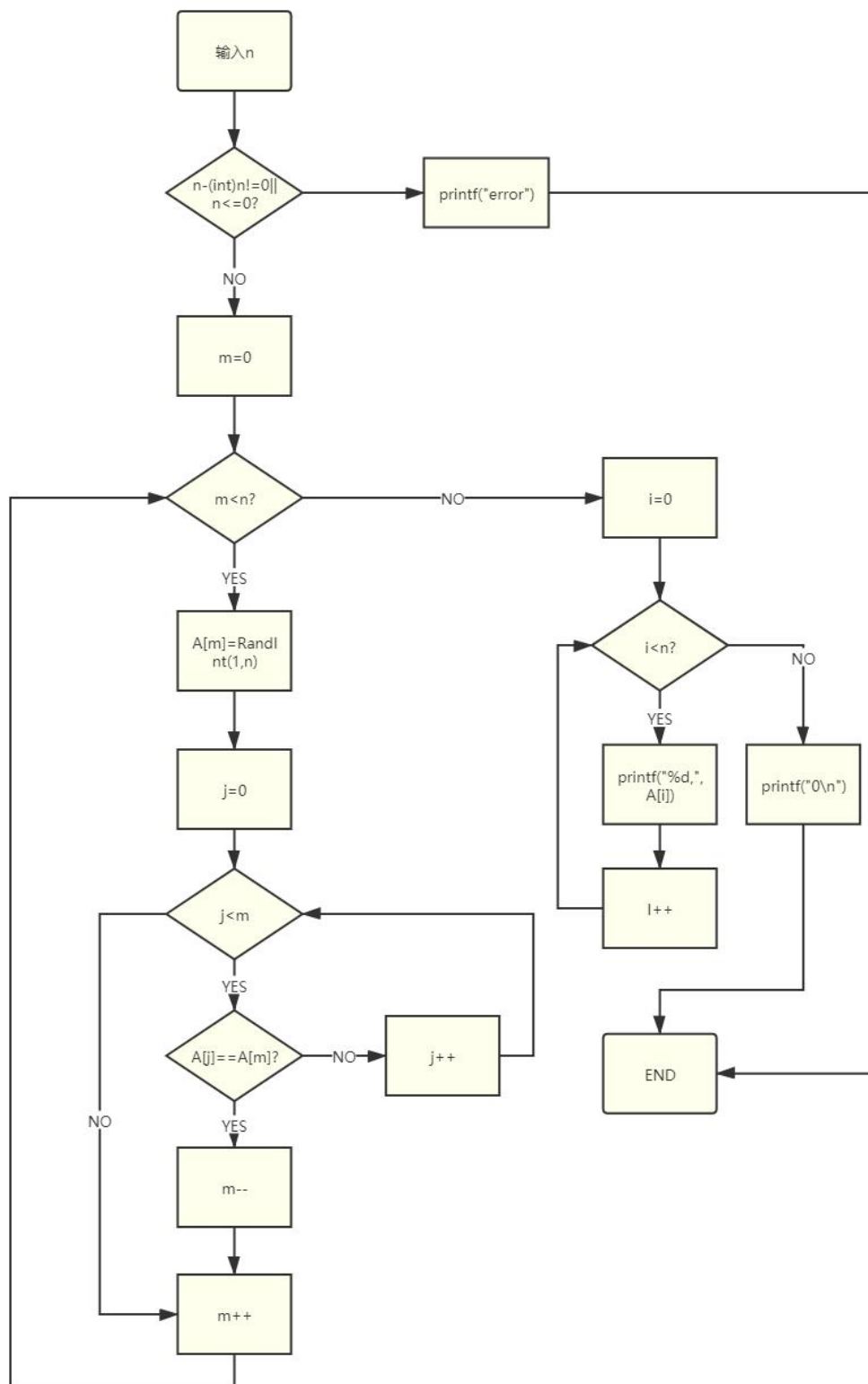


图 2 算法 1 流程图

②算法二流程图：

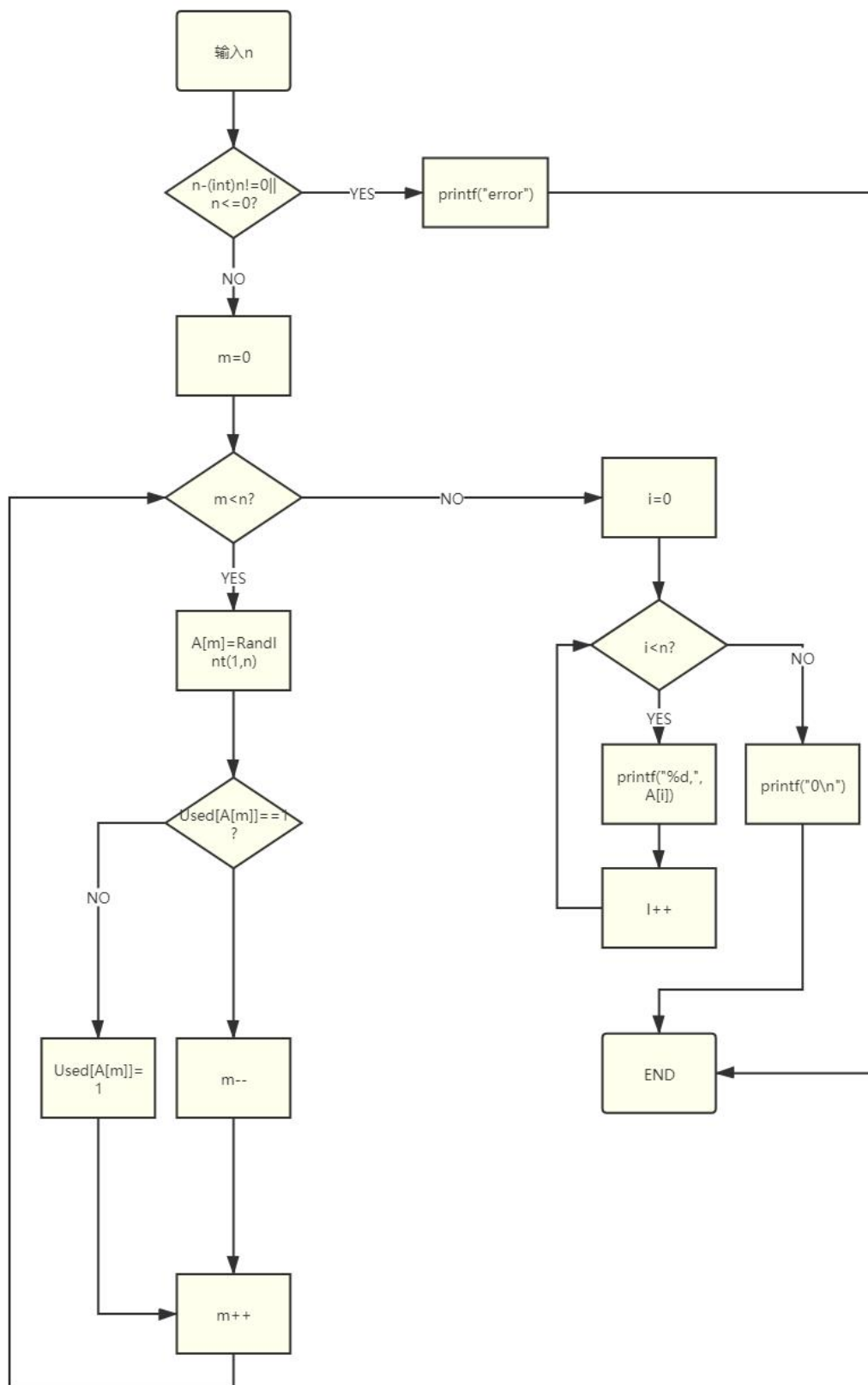


图 3 算法 2 流程图

③算法三流程图：

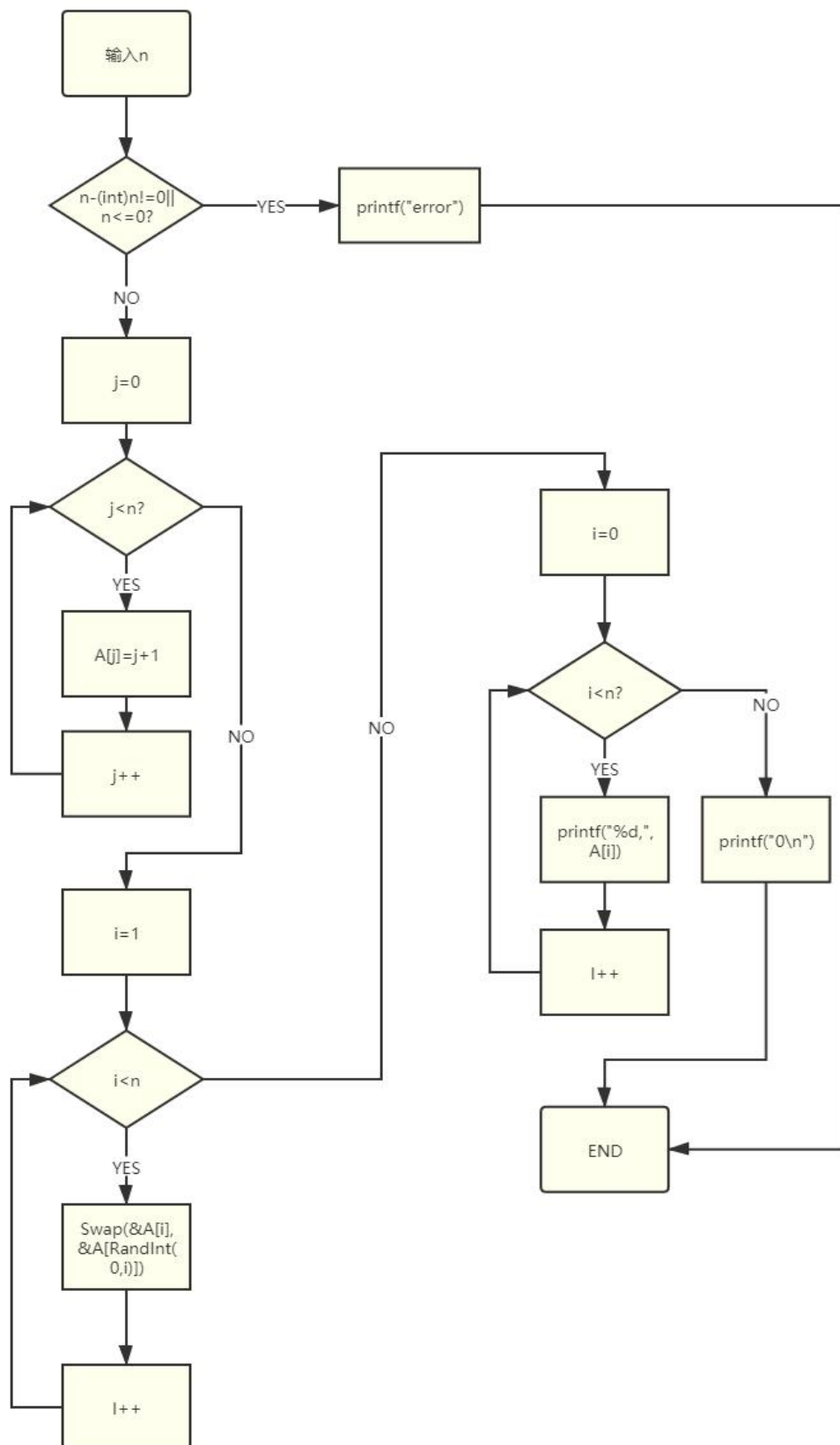


图 4 算法 3 流程图

八、实验数据及结果分析：

代码编写完成后，我从输入为正常值、错误值、边界值三方面对代码进行了测试，得到如下的测试结果：

n	算法 1 所用时间 (s)	算法 2 所用时间 (s)	算法 3 所用时间 (s)
10	0.002	0.001	0.000
100	0.008	0.001	0.000
1000	0.083	0.001	0.0004
10000	1.750	0.008	0.001
20000	12.006	0.023	0.002
40000	19.863	0.105	0.016
80000	79.065	0.238	0.033
160000	363.882	0.605	0.060
320000	数据过大	1.09	0.133

表 1 不同 n 值下的三种算法所需时间








错误类型	输入的错误值	输入及输出截图
输入数值为负数	-1	
	-100	
输入数值为小数	10.5	
	-6.5	
输入数值为非数字	a	
	&	
	_10	

表 2 输入错误情况下测试结果


边界值	输入及输出截图
0	

表 3 输入边界值情况下测试结果

在实验前，我对三种算法的时间复杂度进行了理论值的计算。算法 1 时间复杂度为 $O(n^2)$ ，算法 2 时间复杂度为 $O(n)$ ，算法 3 时间复杂度为 $O(n)$ 。

通过实验，我们得出了实验数据，下面我将分析得到的实验数据。

由表 1 可知，对于输入同样的 n 值，三种算法完成所需的时间不同。算法 3 最快，算法 2 次之，算法 1 最慢。且随着 n 值增加，算法 1 所需时间增长最快；当 n 足够大时，算法 1 所需时间明显高于算法 2 和算法 3；甚至当 n 为 320000 时，算法 1 由于数据过大无法计算。实验结果与时间复杂度理论值分析相符。

由表 2 可知，对于错误的输入，无论错误是负数、小数，还是非数字，本程序可以给出报错，不会由于输入错误出现程序运行错误的情况。

由表 3 可知，对于输入边界值 0，程序会报错，不会由于输入边界值出现程序运行错误的情况。

九、总结及心得体会：

在编写三个函数过程中，既让我理解和学会了解决同一问题的不同算法，也进一步锻炼了我程序设计的能力。通过本次实验我深刻理解了不同算法在运行过程中时间复杂度的巨大差异，明白了选取好的算法可以大幅降低运行时间的道理。本次实验的缺点在于：由于编写程序时没有提前注释 i, j, k, m, n 等变量的具体用途，在使用时自己错误的混乱使用各个变量，导致程序无法正常运行；由于我是第一次使用码图系统，不了解系统测试程序的原理，以至于在本地运行正确的程序在码图系统提交后一直出现被扣分的情况，花费了大量时间才修改到满分。

十、对本实验过程及方法、手段的改进建议及展望：

希望本实验在题目中加入码图系统测试要点，例如：测试时不需要 `main` 函数，对 n 的判断应放在各个函数中。避免本地编译正确但提交出现扣分的情况。此外，题目中建议给出一些测试数据，方便同学们测试。

报告评分：

指导教师签字：