**实验1 Sorting**

1. **实验要求**

实验内容：排序n个元素，元素为随机生成的1到65535之间的整数，n的取值为： ，，，，，；

算法：插入排序，堆排序，快速排序，基数排序，计数排序。

用适当的方法或工具记录算法在执行中所消耗的时间，根据不同输入规模时记录的数据，画出算法在不同输入规模下的运行时间曲线图；比较曲线是否与课本中的算法渐进性能，给出分析。

比较分析不同的排序算法的时间曲线， 根据图表分析各种排序算法在不同输入规模下的优劣，并分析与课本上介绍的算法渐进性能的相符程度。

1. **实验环境**

编译环境：Windows 10

机器内存：8G

时钟主频：28GHz

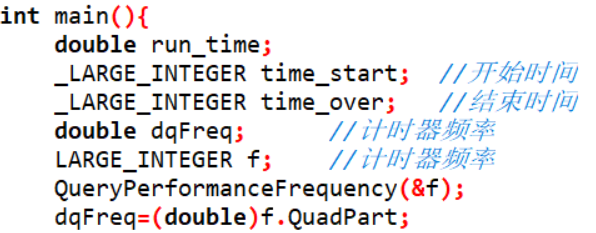
1. **实验过程**

本次实验采用C语言编写，在Dev C++下运行。

根据实验要求，首先生成行的随机数据，并将其写入input文件夹的input\_integer.txt中。在本次实验中，我采用srand函数生成随机数，其中rand()函数生成0-32767范围的整数，为了满足实验要求，可以先判断rand()生成的数是否能被2整除，再决定是否加上32768，如此以生成1-65535范围的随机数。

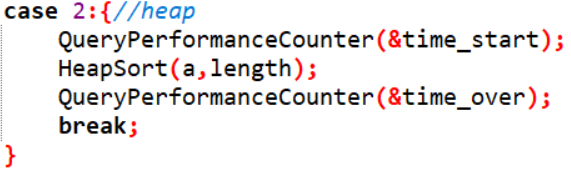
接下来编写5个不同的排序算法，根据书本算法结合C语言中的数组实际情况进行微调。

1. **实验关键代码截图（结合文字说明）**



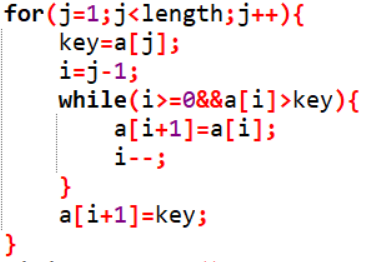
此部分用于排序程序的计时，可以精确到微妙。需要包括头文件(#include<windows.h>)。

计时示范：

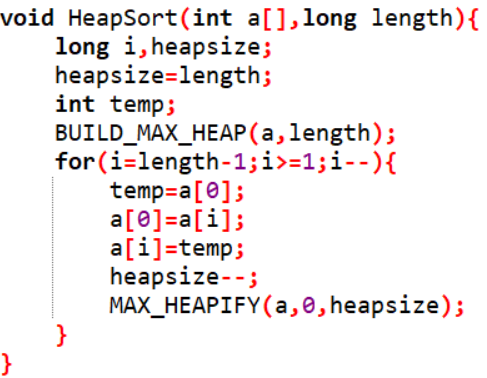


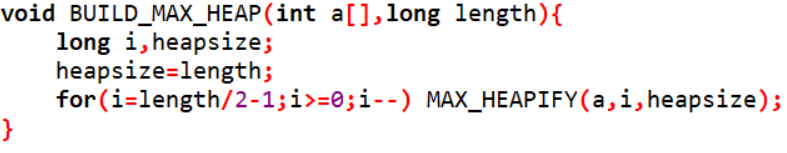
在排序开始和结束处记录时间，之后将两者相减即排序算法所用时间。

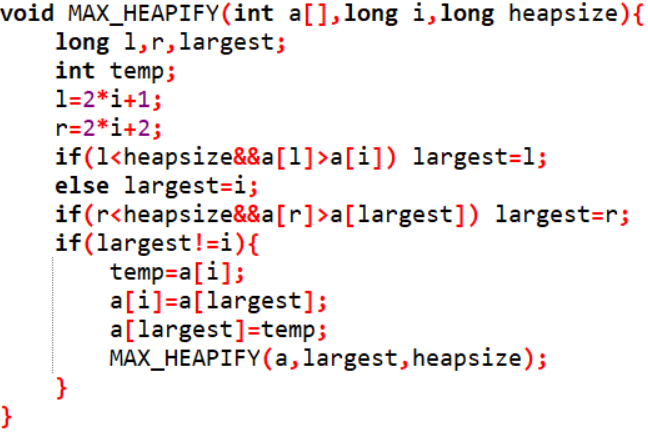
算法部分：



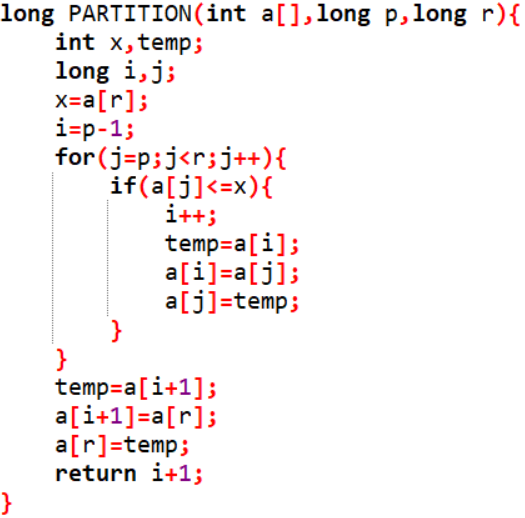
此部分即插入排序，与书本上基本一致，运行效率较低。

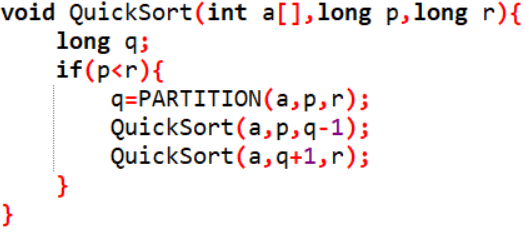




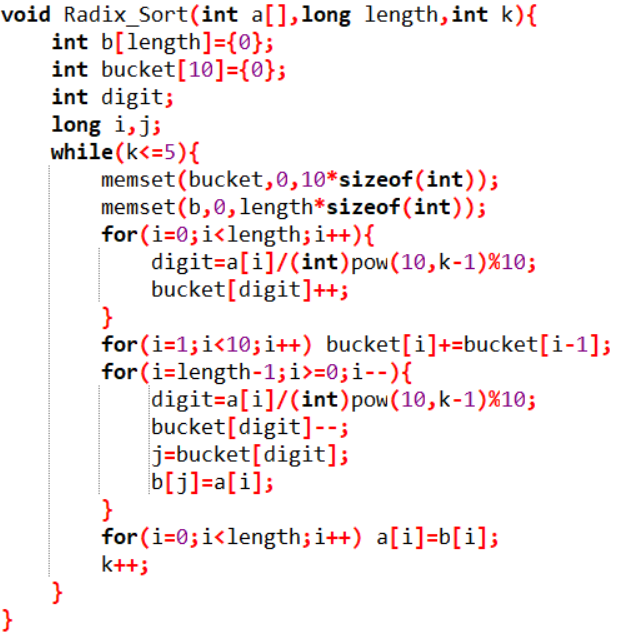


该算法为堆排序算法，与书上基本相同，关键的区别在于书本上的数组下标从1开始，而C的数组下标从0开始，故二叉树的左孩子右孩子应当表示为2\*i+1和2\*i+2。

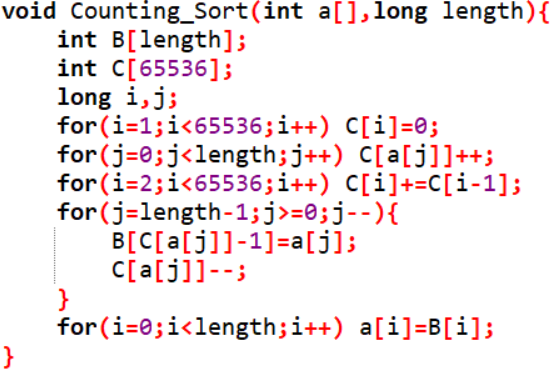




快速排序代码与书上也基本相同，同样需要注意下标的起始，除此以外由于数组的长度最大能达到131072，故下标的数据类型应当定义为长整型long。

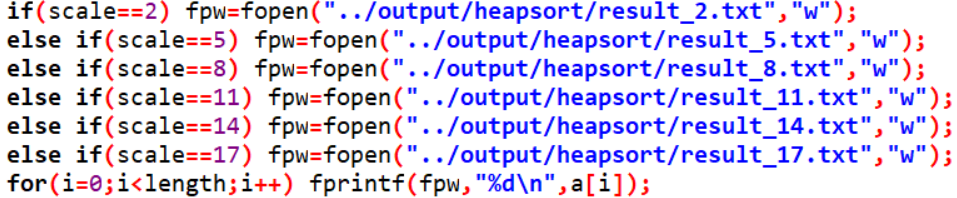


基数排序在书上没有具体算法伪代码，此处自己实现。首先定义两个数组b和bucket，分别用来存储排序后数据和相应索引下数据的个数。之后将数据排入桶，并调整桶数组使其便于计算桶中元素经过一次重排之后的下标。如此重复5次(本次实验随机数最大为65535，是一个5位数，只需要重复5次)，即可完成重排。



计数排序与书上伪代码基本相同，同样需要注意下标的问题。

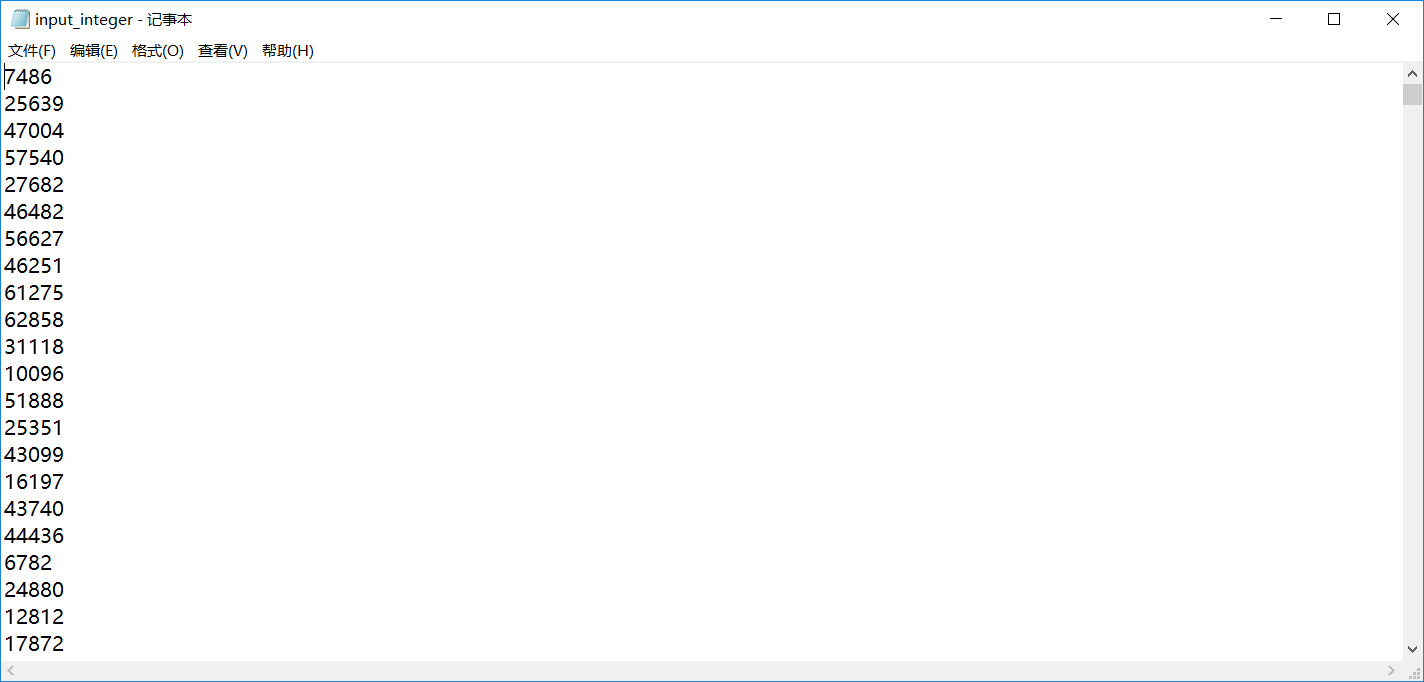
最后是结果输出部分，采用如下代码：



将排序后的数组输出到相应目录下的txt文件中，并将同一个算法的排序时间输出到一个txt中。

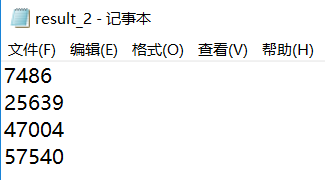
1. **实验结果、分析（结合相关数据图表分析）**

实验初始随机数据：

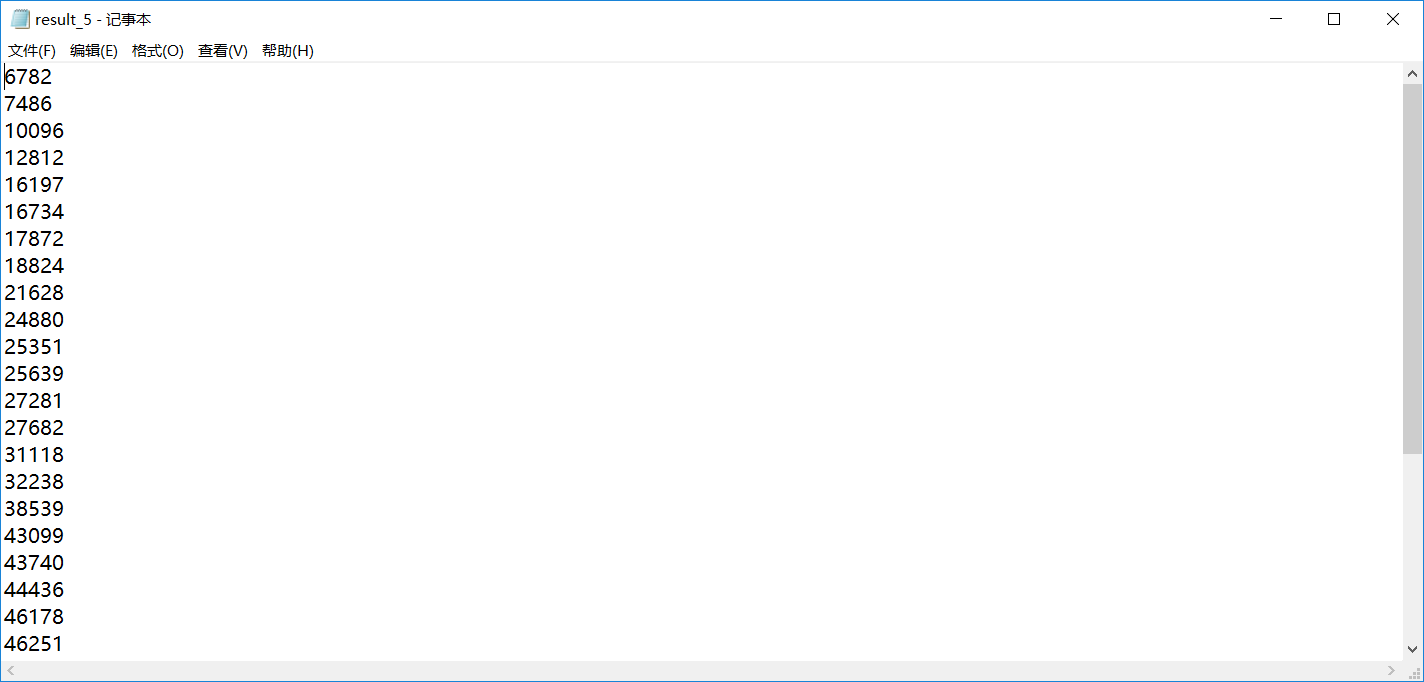


实验结果输出：

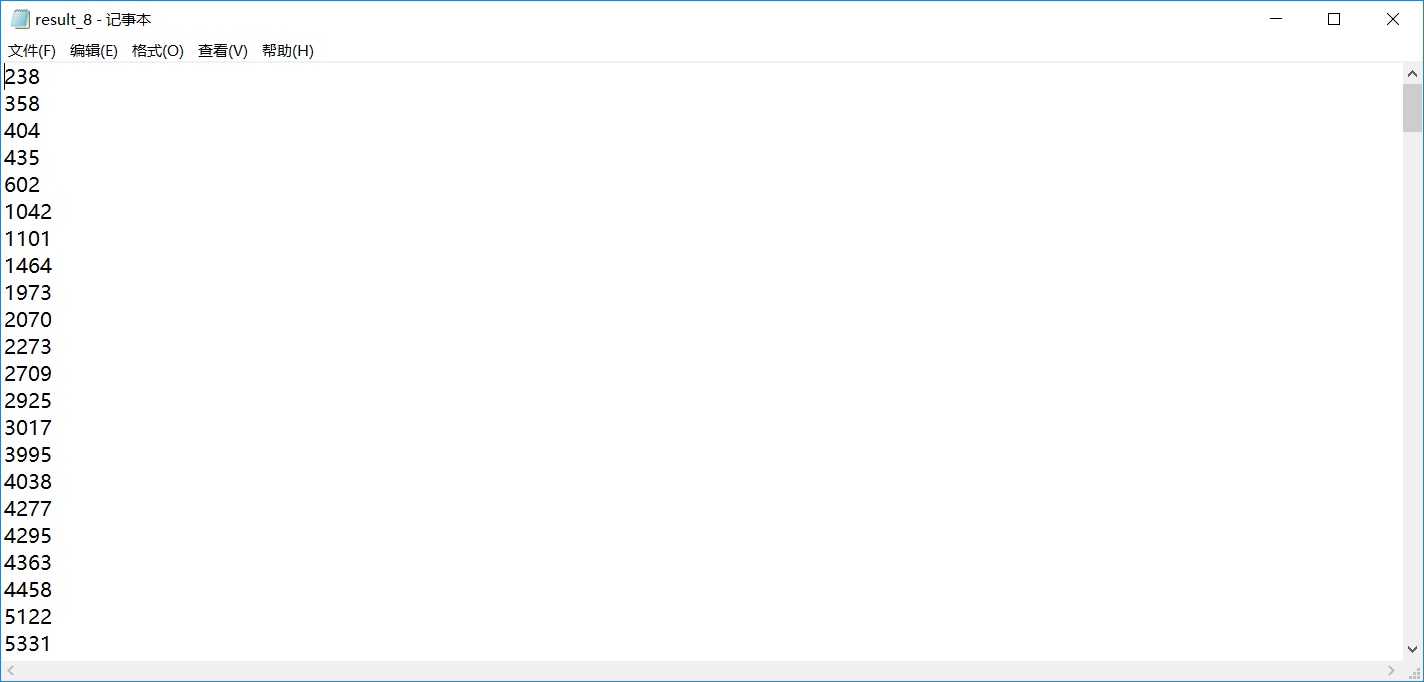
result\_2:



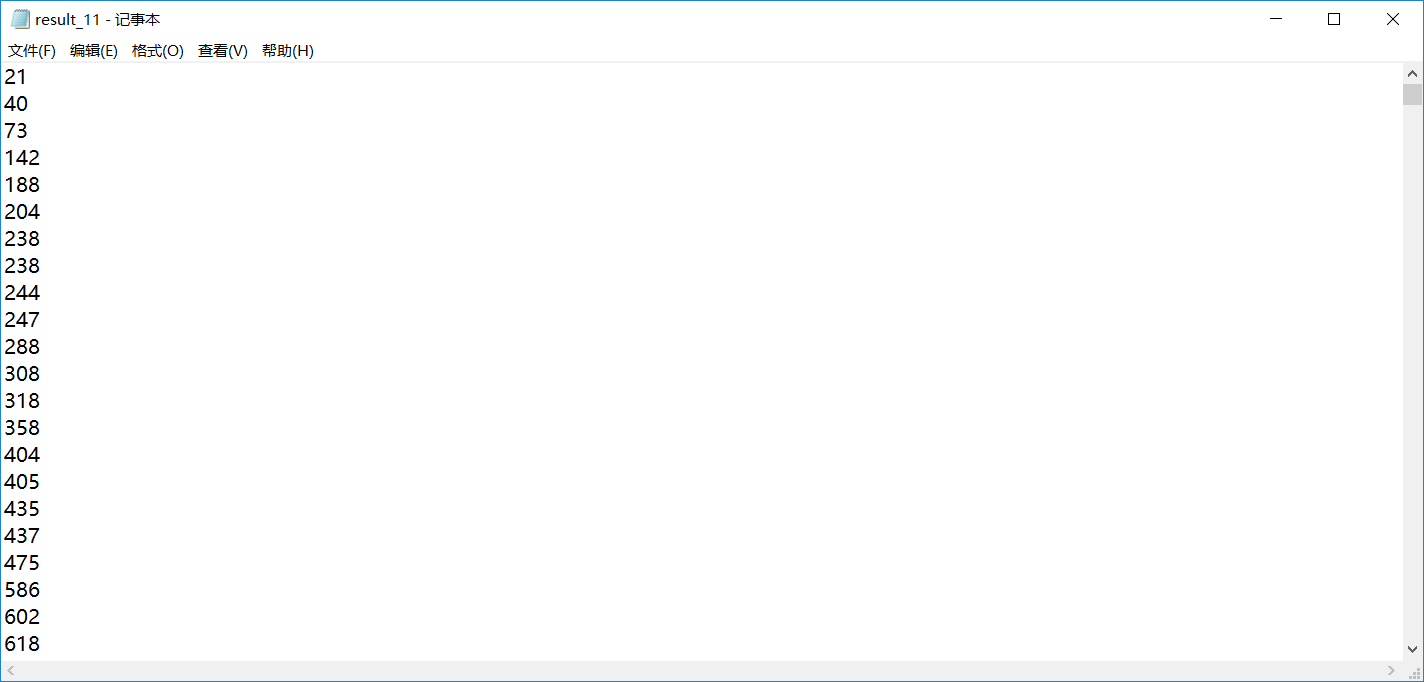
result\_5.txt:



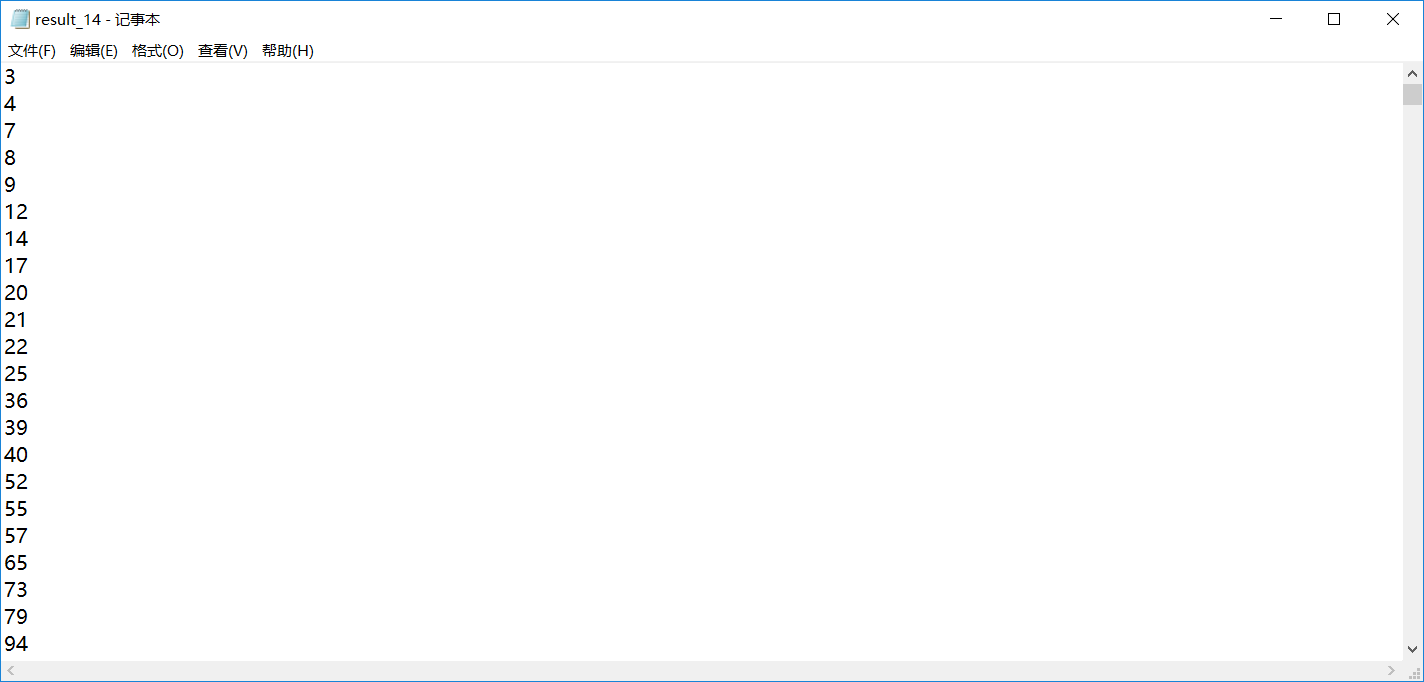
result\_8.txt:



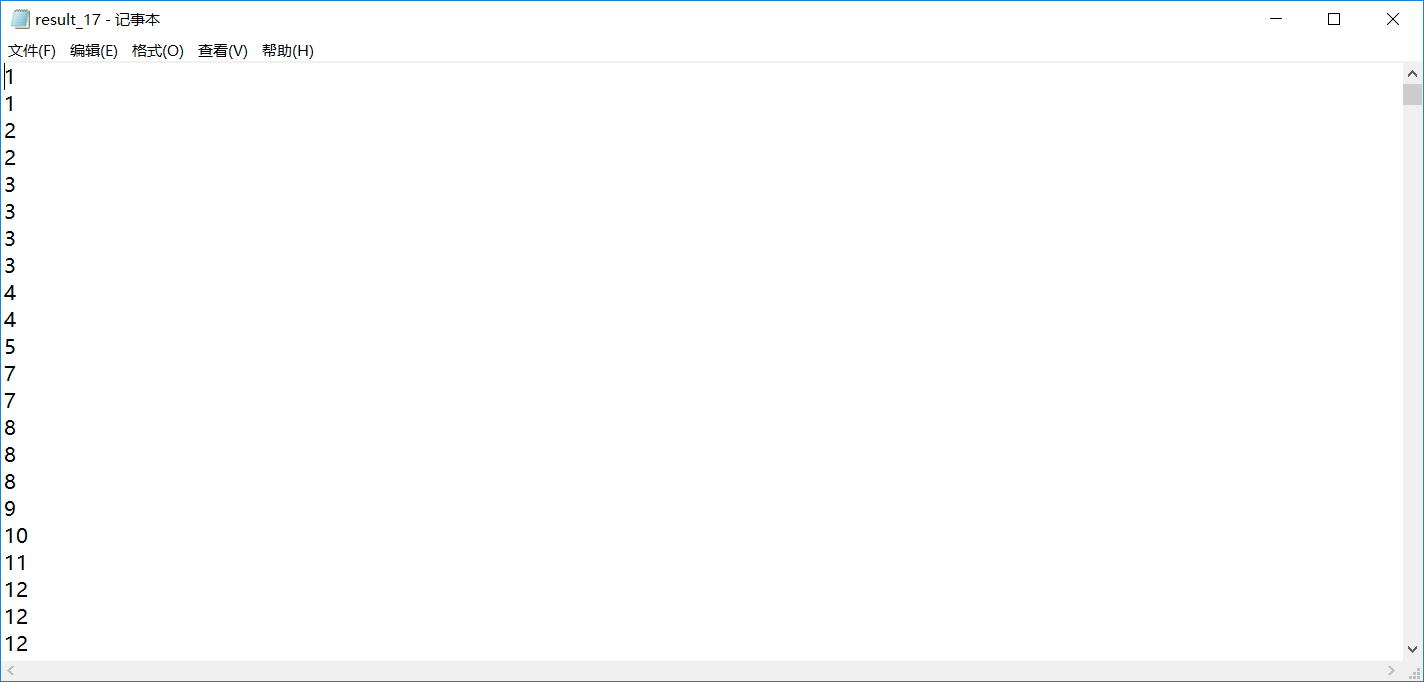
result\_11.txt:



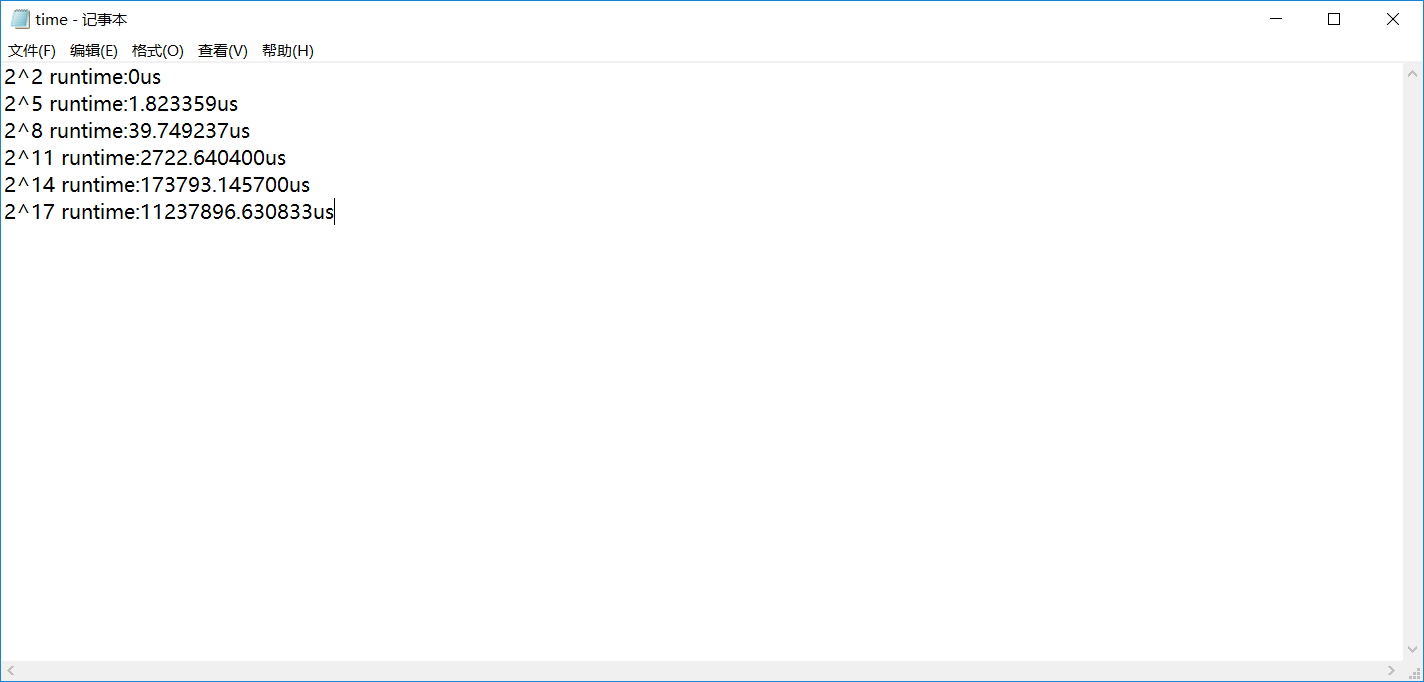
result\_14.txt:



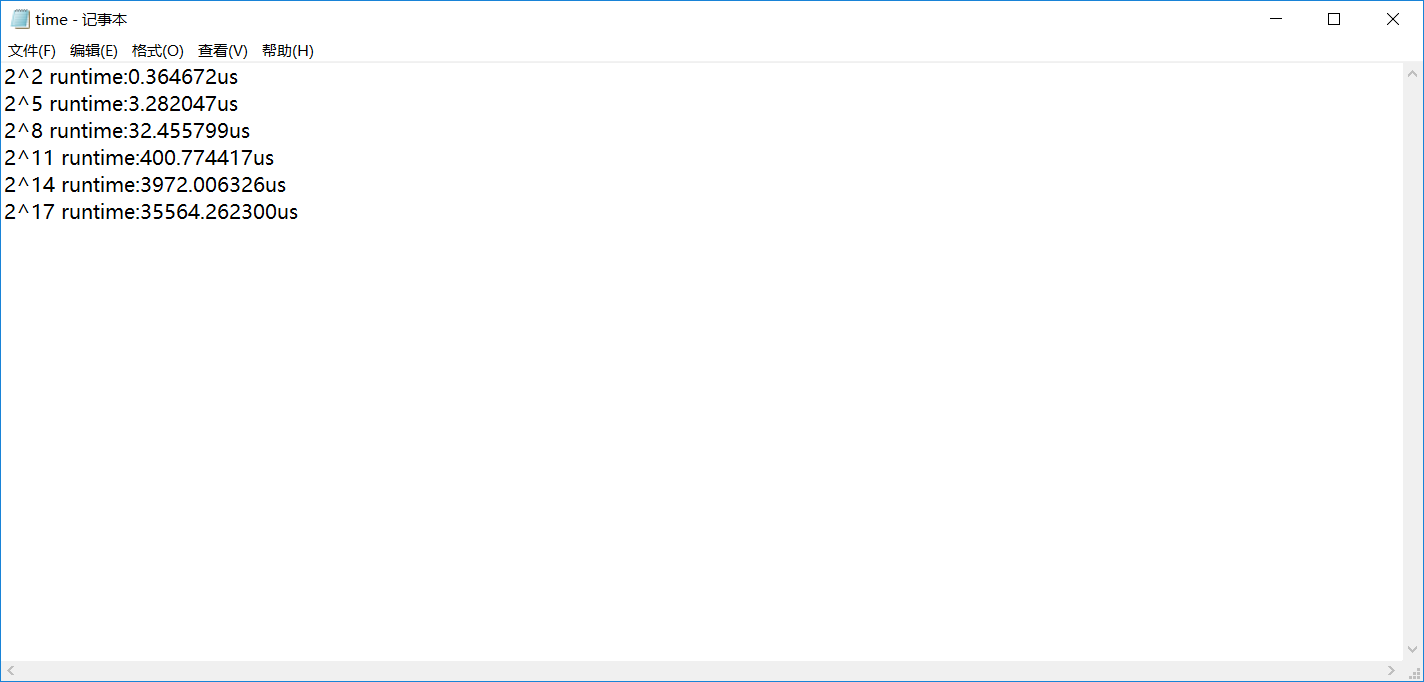
result\_17.txt:



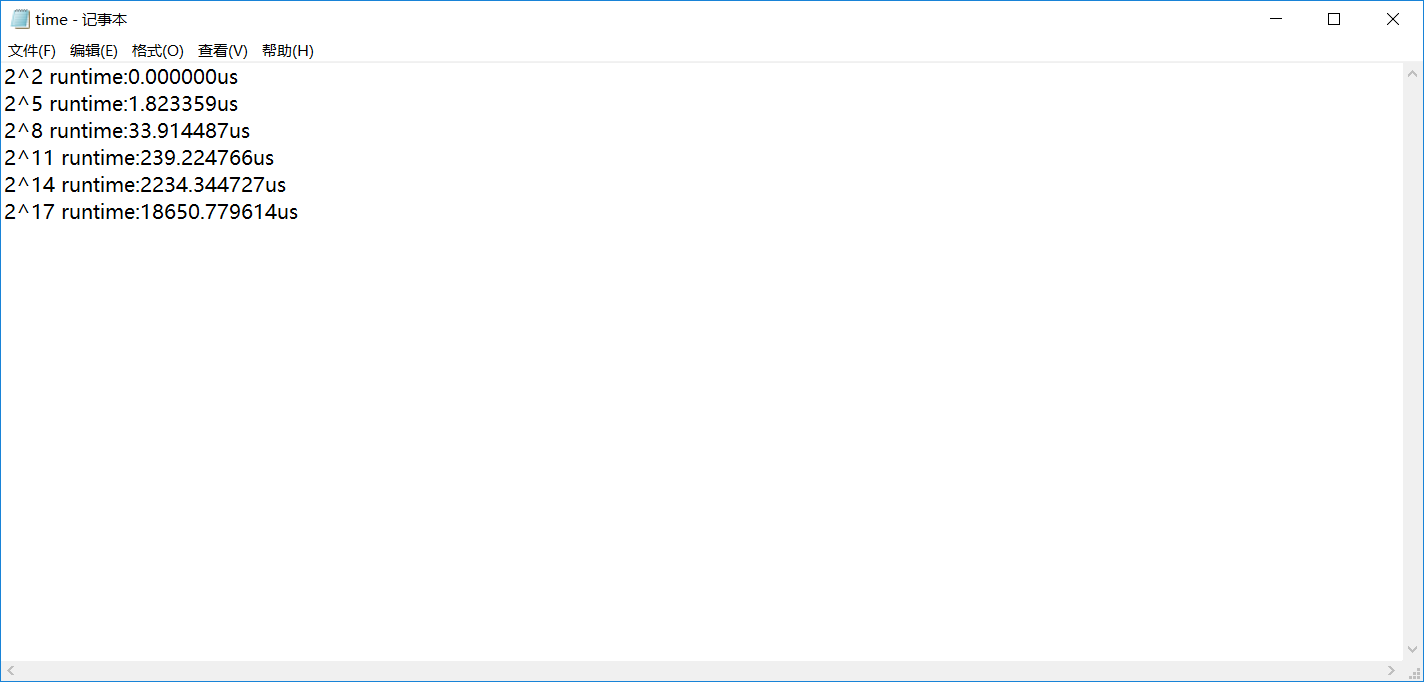
插入排序算法的时间：



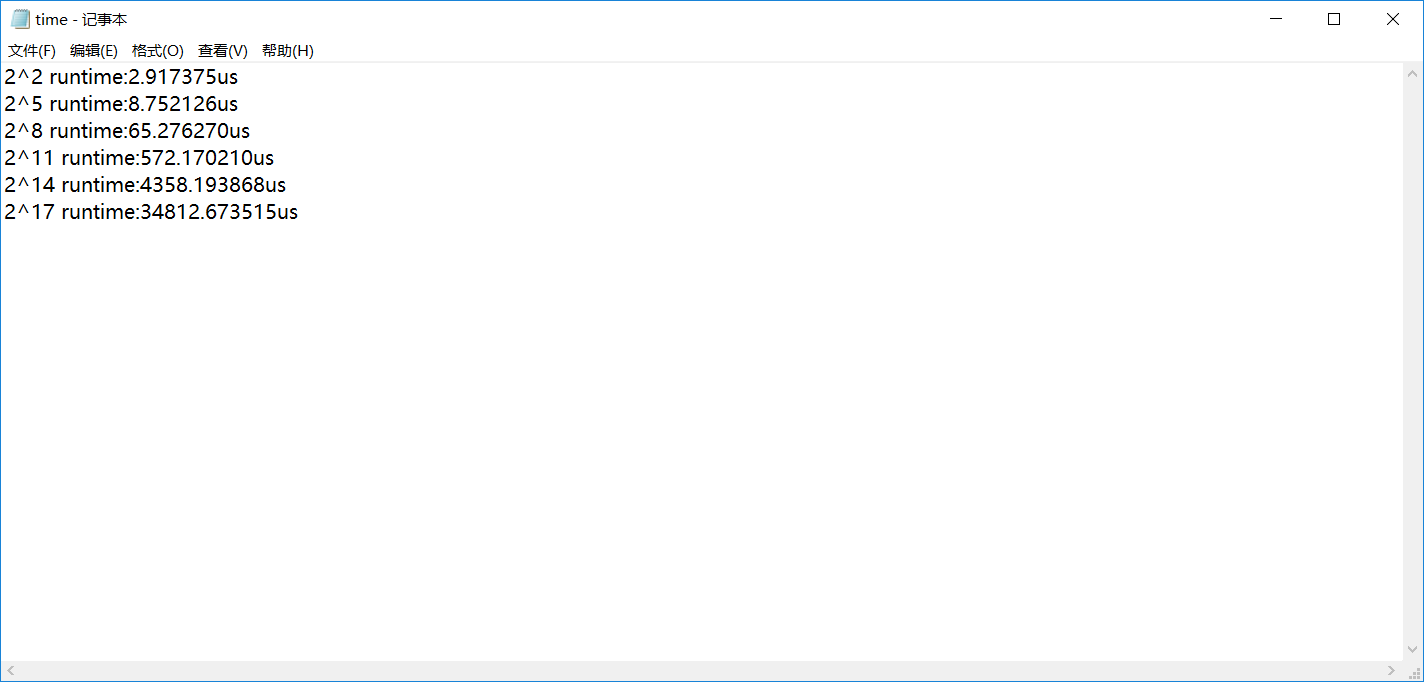
堆排序算法的时间：



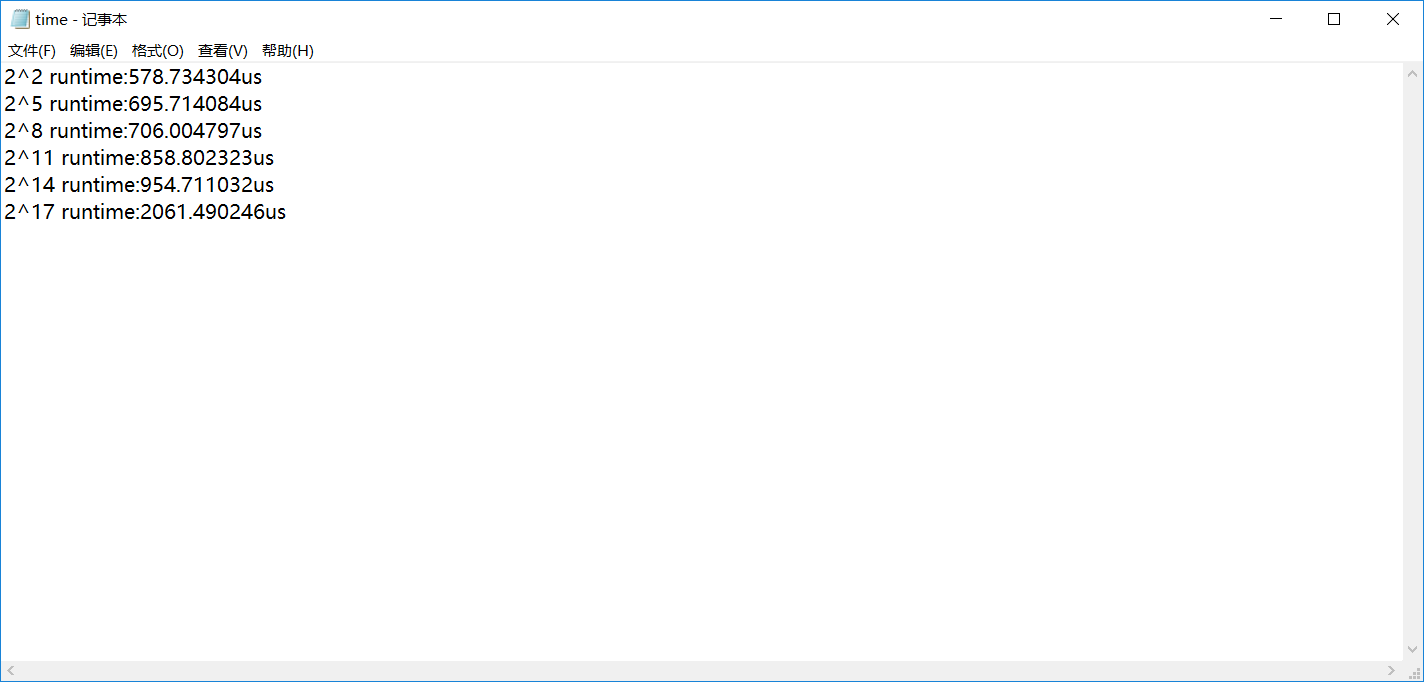
快速排序算法的时间：



基数排序算法的时间：



计数排序算法的时间：



使用Excel作图并分析：

1.插入排序：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 插入排序 | | | | |
| n | n^2 | rt(us) | lg(n^2) | lg(rt) |
| **4** | **16** | **0** | **4** | **-** |
| **32** | **1,024** | **2** | **10** | **1** |
| **256** | **65,536** | **40** | **16** | **5.322** |
| **2,048** | **4,194,304** | **2723** | **22** | **11.411** |
| **16,384** | **268,435,456** | **173793** | **28** | **17.407** |
| **131,072** | **17,179,869,184** | **11237897** | **34** | **23.422** |

通过图可见符合情况较好。

2.堆排序：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 堆排序 | | | | |
| n | n\*lgn | rt(us) | lg(n\*lgn) | lg(rt) |
| **4** | **8** | **1** | **3** | **0** |
| **32** | **160** | **3** | **7** | **1．585** |
| **256** | **2048** | **32** | **11** | **5** |
| **2,048** | **22528** | **401** | **14** | **8.647** |
| **16,384** | **229376** | **3972** | **18** | **11.956** |
| **131,072** | **2228224** | **35564** | **21** | **15.118** |

符合情况较好。

3.快速排序：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 快速排序 | | | | |
| n | n\*lgn | rt(us) | lg(n\*lgn) | lg(rt) |
| **4** | **8** | **1** | **3** | **0** |
| **32** | **160** | **2** | **7** | **1** |
| **256** | **2,048** | **34** | **11** | **5.087** |
| **2,048** | **22,528** | **239** | **14** | **7.900** |
| **16,384** | **229,376** | **2234** | **18** | **11.125** |
| **131,072** | **2,228,224** | **18651** | **21** | **14.187** |

除去前面几个点，符合情况较好。

4.基数排序：

|  |  |  |  |
| --- | --- | --- | --- |
| 基数排序 | | | |
| n | rt(us) | lg(n) | lg(rt) |
| **4** | **3** | **2** | **1.585** |
| **32** | **9** | **5** | **3.170** |
| **256** | **65** | **8** | **6.022** |
| **2,048** | **572** | **11** | **9.160** |
| **16,384** | **4358** | **14** | **12.089** |
| **131,072** | **34813** | **17** | **15.087** |

符合情况较好。

5.计数排序：

|  |  |  |  |
| --- | --- | --- | --- |
| 计数排序 | | | |
| n | rt(us) | lg(n) | lg(rt) |
| **4** | **578** | **2** | **9.175** |
| **32** | **696** | **5** | **9.443** |
| **256** | **706** | **8** | **9.463** |
| **2,048** | **859** | **11** | **9.746** |
| **16,384** | **955** | **14** | **9.899** |
| **131,072** | **2061** | **17** | **11.009** |

图线较为反常，符合情况不好，可能是由实验数据规模较小时为常数时间引起的。

1. **实验心得**

通过本次实验，我加深了对这5种排序算法的理解，提高了编程熟练度，掌握了程序计时的方法，进一步练习了绘图分析。