《数据结构》课程实践报告

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 院、系 | 计算机学院 | | 年级专业 | 大二 | 姓名 | 张芷涵 | 学号 | 1827405114 |
| 实验布置日期 | |  | | 提交  日期 |  | | 成绩 |  |

课程实践实验6：排序算法的实现及性能测试及比较

## 一、问题描述及要求

排序算法的实现及性能测试及比较

[问题描述]在书中，各种内部排序算法的时间复杂度分析结果只给出了算法执行时间的阶，或大概执行时间。试通过具体数据比较各种算法的关键字比较次数和记录移动次数，以取得直观感受。

要求：

（1）编写程序创建一些整数文件用于排序。创建的这些文件可以根据需要生成不同的长度，如长度分别为20，200和2000，以正序、逆序、随机顺序的方式创建这些文件，通过把所有这些测试数据保存在文件中（而不是每次在测试程序时用随机数生成），可以使用同样的数据去测试不同的方法，因此会更易于比较这些方法的性能。

（2）数据表采用顺序存储结构，实现插入排序，选择排序，希尔排序，归并排序，快速排序，堆排序等排序，并对这些算法的实现效率进行比较和分析。

（3）排序的指标包含：运行时间，比较次数，移动次数。

## 二、概要设计

**可包含以下内容，可根据实际情况取舍。**

**（1）对实验内容的理解和二次概括。**

**（2）给出系统的功能列表（可以用图示或清单的形式）**

**（3）程序运行的界面设计 （可以用图示等方式，如：首先出现屏幕提示，请用户选择输入配置的方式，1从键盘输入活单元坐标2……3….然后用户）**

**（4）确定总体设计思路，采用何种数据结构，设计哪些类，各类的作用 ，类方法的介绍，类之间的关系描述**

**（5）程序结构设计，包括：对已有程序的使用，自己将设计哪些程序文件，各部分关系描述。**

这部分最核心的工作是：描述解决问题所需用到的数据结构和算法。——给出明确地确定该结构及对应算法的理由。

此外，这部分可以给出面向对象相关的设计，例如实现完整功能所涉及的类及对象间的总体关联关系等。

1. 按照要求，设计一个创建数据的函数，用顺序表存储整数，设计各种排序算法：插入排序、选择排序、希尔排序、快排、归并排序、堆排。
2. 运行界面：
3. 修改自己实现过的顺序存储实现的线性表类，，引入头文件<list.h>。修改之处是，将原本大小固定的数组，改为动态申请的数组。这样可以比较较大数据量时，各种算法时间性能。

创建数据的函数按用户需求创建整数文件。

读取数据的函数将数据赋给顺序表。

各种排序函数将所给的顺序表排序。

## 三、详细设计

**主函数、关键算法的设计（可以用流程图等手段表示）**

**着重对难点部分进行分析。**

1. 创建数据函数的设计：
2. 插入排序：
3. 选择排序：
4. 希尔排序：
5. 快排：

划分函数：

1. 归并排序：

从中间将表划分为两个子表，分别递归排序两个子表，再合并。

合并函数：

1. 堆排：

建立堆：从最后一个非叶子节点开始，依次取出之前所有元素，调整加入到堆中

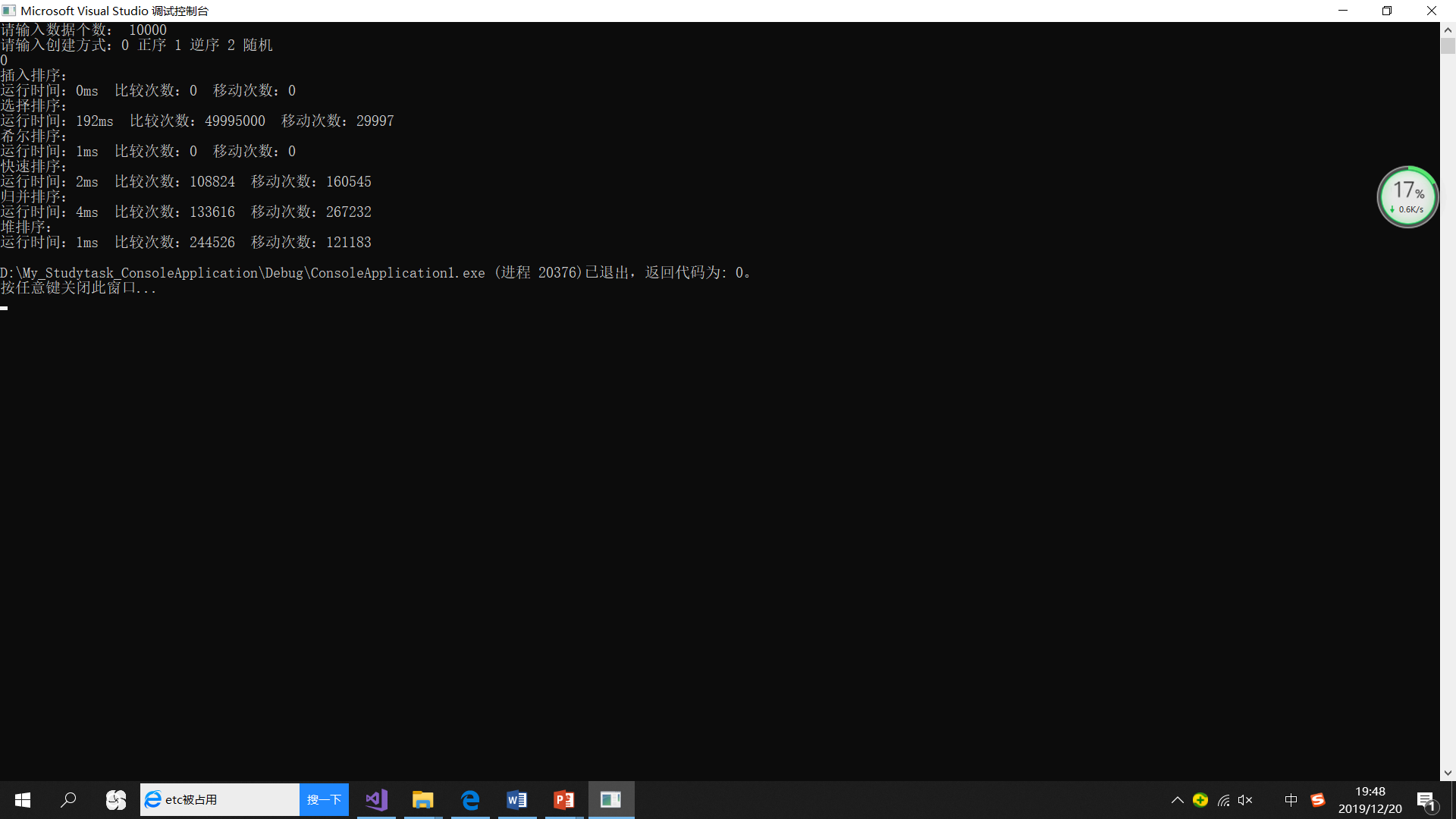
调整堆：

## 四、实验结果

紧扣题目要求设计提供相应的测试方法和结果。可以给出具体的测试用例，每个测试用例一般可列出：

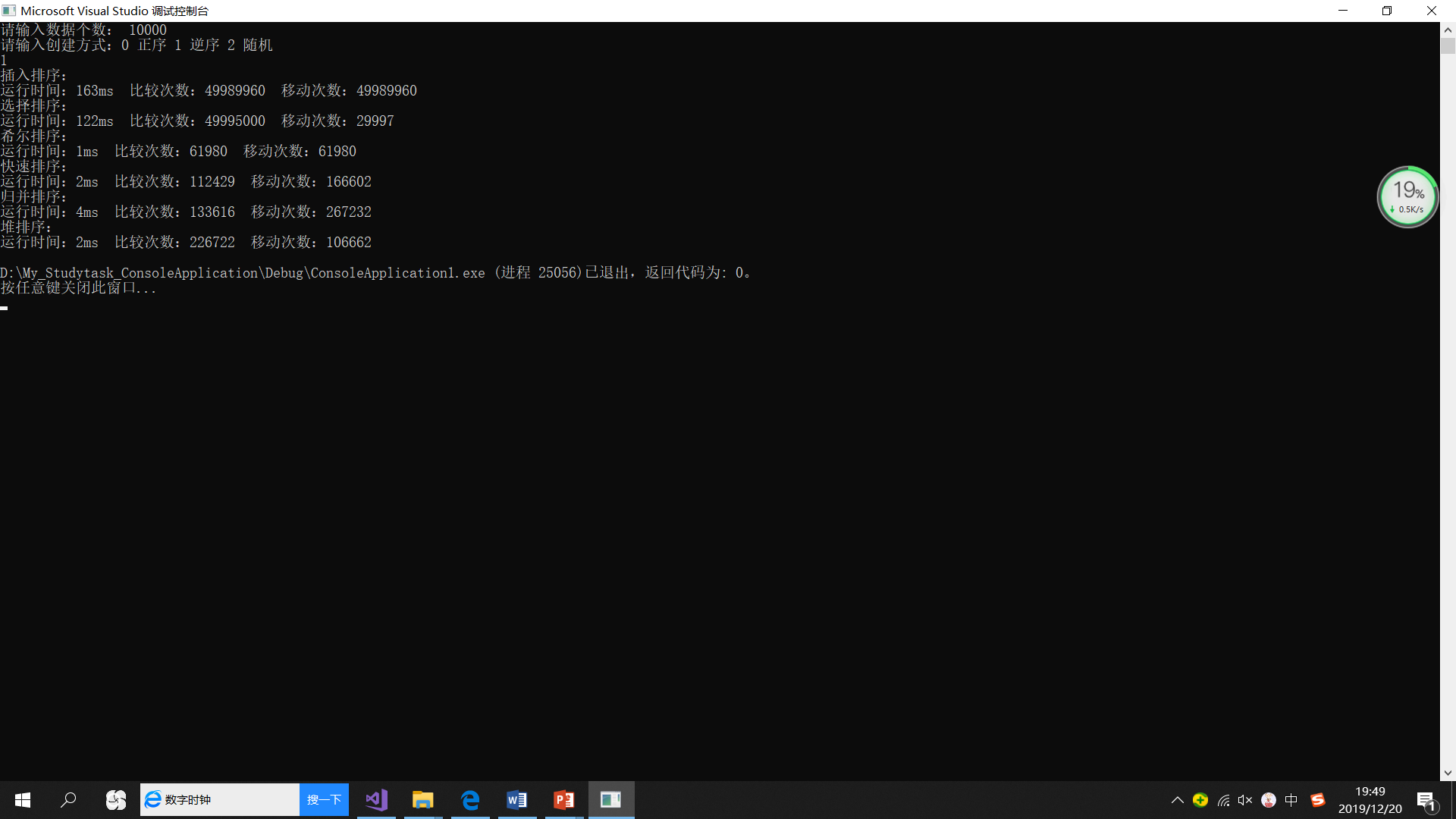
* 测试输入：设计一组输入数据
* 测试目的：针对具体哪方面的漏洞
* 正确输出：对应该输入，算法正确时的输出内容
* 实际输出：实际测试得到的输出 **粘贴输入及运行情况截图**
* 错误原因：如果实际输出与正确输出不符合，分析可能的错误原因
* 测试结论：通过、已改正和待修改三种。

1. 正序 10000



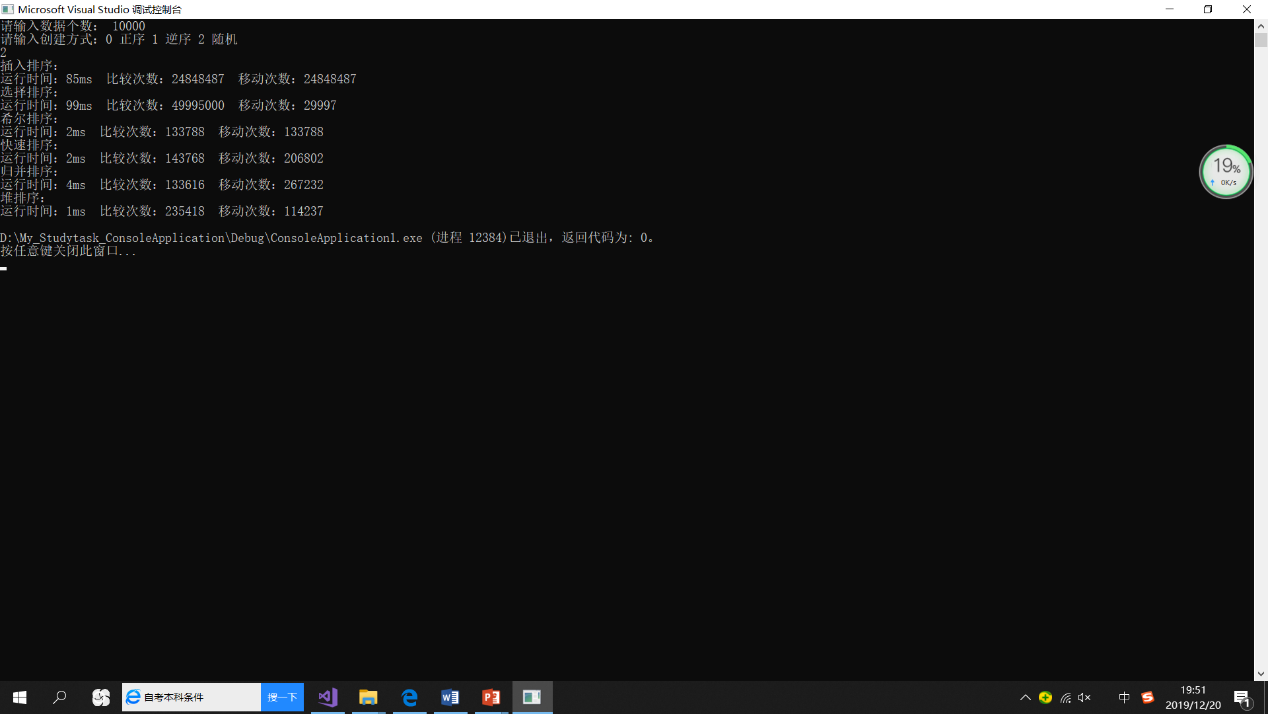
测试插入排序的最好情况

1. 逆序 10000



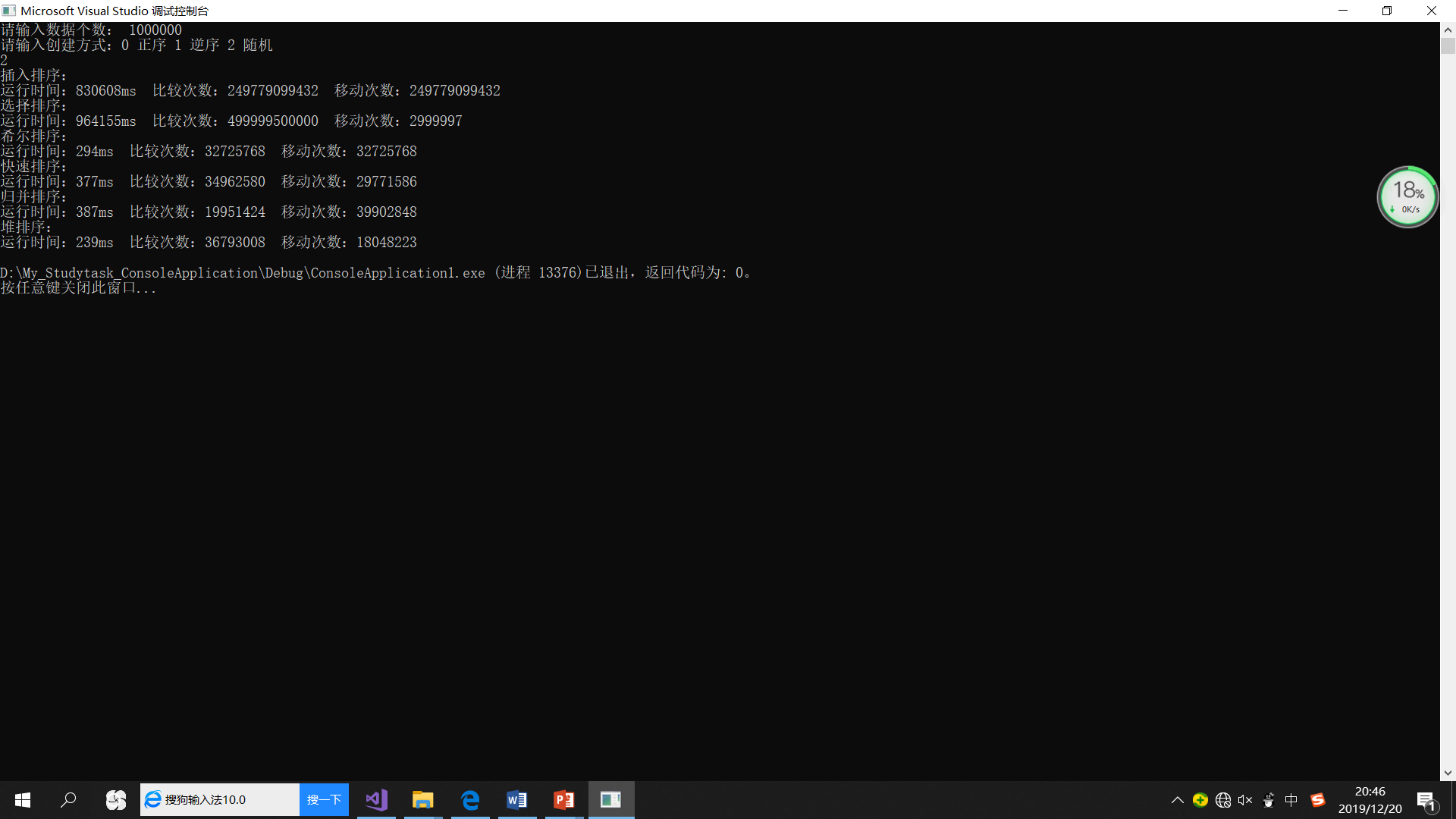
测试插入排序的最坏情况

1. 随机 10000



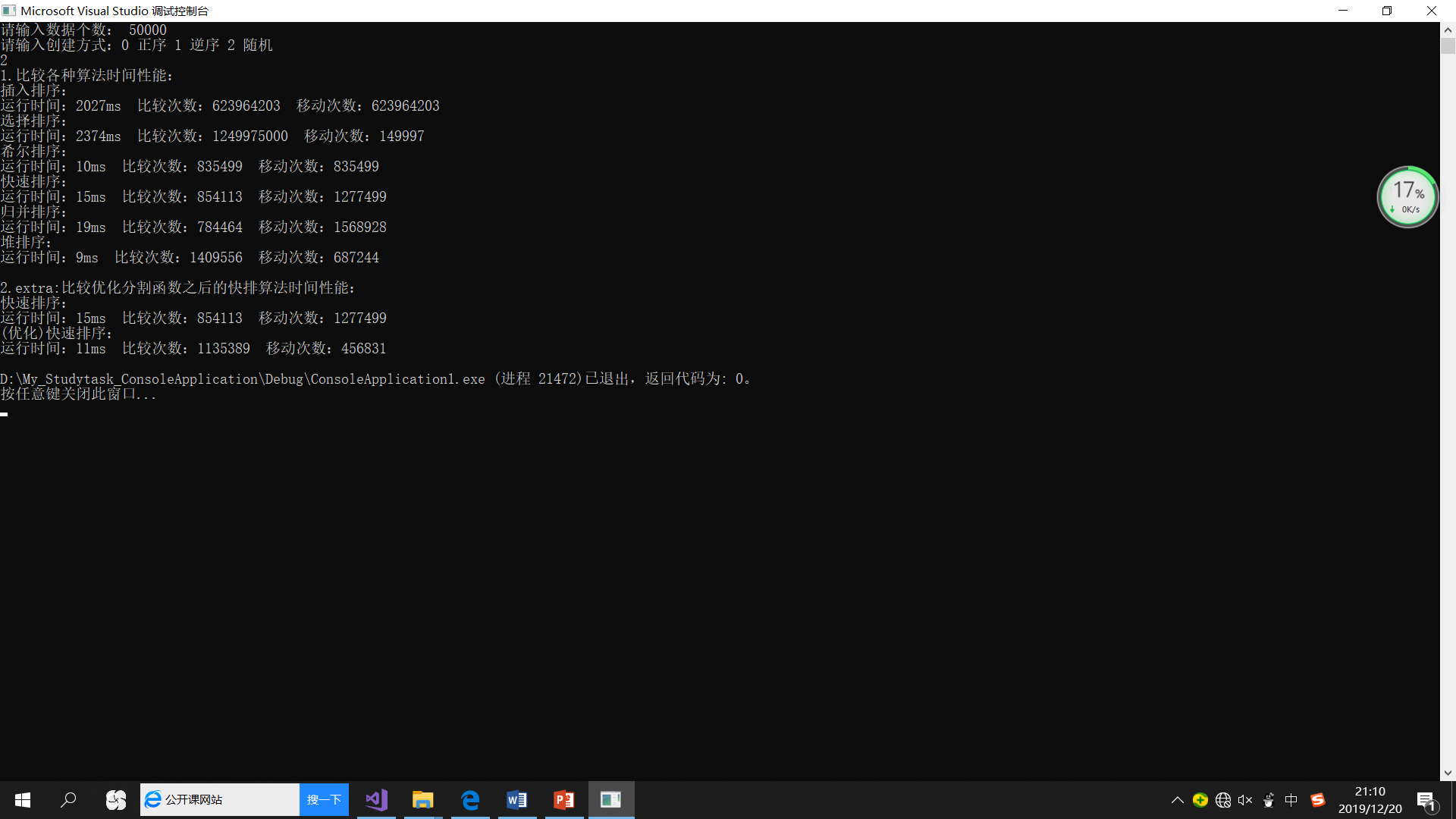
测试一般情况下，各排序时间性能

1. 随机 1000000



测试较大数据量时，各排序时间性能

1. 随机 50000



比较优化分割函数之后的快排算法和原来的时间性能

## 五、实验分析与探讨

这部分是实验报告最出彩的地方，主要应包含两方面：

测试结果的分析：解释测试策略，对得到的结果进行分析，总结算法的时空复杂度，得出对算法性能方面分析的结论

不局限于题目的一些限制条件和使用算法的要求，探讨更多解决问题的途径等。

**实验设计、实现过程中遇到的问题，将在分析问题、解决问题的整个过程中遇到的问题罗列出来，并写出解决问题的方法。**

1.测试结果的分析：

插入排序:

算法设计时，将其放在第一个不大于它的位置，因此是稳定的。

平均比较次数和移动次数均为1/4\*n^2+O(n)级别.

最好情况发生在表正序时，此时时间复杂度为O(n),比较次数为n-1次，移动次数为0次。

最坏情况发生在表反序时，此时时间复杂度为O(n^2),比较次数为n(n-1)/2次，移动次数为n(n-1)/2+2(n-1)次。

选择排序：

与表初始状态无关。平均比较了（1+2+…+n-1）=n(n-1)/2次，1/2\*n^2+O（n）级别。平均移动了3\*（n-1）次，3\*n+O(1)级别。由于移动次数较少，当记录元素较大时，选择排序比插入排序更为合适。而记录元素较小，移动花费时间不多时，插入排序由于比较次数少，会更好。

希尔排序：

执行时间和增量序列的选择有关。算法不稳定。比较次数少于直接插入排序，执行时间更快。因为开始时增量较大，分组较多，每组的记录数目少，各组内直接插入较快，后来增量逐渐缩小，分组数逐渐减少，而各组的记录数目逐渐增多，但由于已经相对有序，所以新一趟排序也较快。实验中，希尔排序比较快，不输给快排和归并。

快速排序：

快排不稳定，时间复杂度是O（nlogn）。平均比较次数约为1.39n\*logn+O(n)，移动次数约为0.69n\*logn+O(n).理论上，快排的最坏情况，会降为O（n^2）。

归并排序：

归并排序稳定，时间复杂度O（nlogn）。我采用的实现方法，比较次数约为O（nlogn）,移动次数为O（2nlogn）.但是在顺序表下，和其他算法相比，时间性能不是最快的。在链表下，这种算法是速度最快的算法之一。

堆排序：

堆排算法时间复杂度也是O（nlogn）级别，不稳定、是原地算法，适用于顺序表时间性能较高。建立堆大约是O（n）级别。算法的核心在于调整堆。理论上，最坏情况下，比较次数为2\*nlogn+O(n)，移动次数为nlogn+O(n)。堆排的有序部分的生成，是从最后一个元素位置往前增长。

2. 按照书上提示，快排的分割函数可以优化，将平均交换次数从1/2\*n降到1/6\*n。优化之后，我的实验结果和理论差不多。

## 六、小结

给出设计实现过程的小结和展望

**任何感想、收获、建议 完成了哪些内容，未完成哪些内容，选做部分有否完成**

**程序的局限性，进一步学习和完善的方向**

感想、收获：

按照题目要求，我通过具体数据比较各种算法的关键字比较次数和记录移动次数，对各种算法的时间性能有了直观感受。利用程序根据用户需求创建不同长度的一些整数文件用于排序。并以正序、逆序、随机顺序的方式创建这些文件，通过把所有这些测试数据保存在文件中，可以使用同样的数据去测试不同的方法，因此会更易于比较性能。数据表采用顺序存储结构，实现插入排序，选择排序，希尔排序，归并排序，快速排序，堆排序。

基于比较的排序算法的时间上限是O（nlogn），归并、快排、堆排都是这个级别的，快排、堆排不稳定，而归并排序是稳定的。快排不适用于链式存储的线性表，因为链式存储不能随机读取，获得元素花费时间较长。归并排序在链式存储下很快，顺序存储时，需要用辅助空间帮助合并表。插入排序是稳定的。相对于插入排序，选择排序比较次数多，移动次数少。希尔排序是改进之后的插入排序。通过分组排序先让表相对有序，然后再排序。时间性能和分组有关。在我的几个测试样例中，希尔排序执行时间和递归版本的快排、归并排序差不多。

除了题目要求，我还按照书上提示，完成了对快排分割函数的优化。这种方法使用双指针，从表两端同时遍历，当两边均与pivot有逆的大小关系时，才彼此交换。因此，书上通过概率数学证明了优化的原因。

程序的局限性：可以考虑实现一个功能菜单，让用户自己选择排序算法并查看执行效果。

## 附录：源代码

(注：源代码打包成\*.zip文件，独立上传)