实验 9 日志

一. 实验设计

实验 9 要求我们实现两个排序算法，并对其在 100,1k,10k,100k,1M 的数据规模下对设计出的排序算法的性能进行分析，计算排序运行时间。由于数据较多可以采用随机数生成的方法去随机生成不同规模下的测试数据，进行不同数据规模下排序所需要的时间，进而对比分析其性能。

实验代码的设计：实验代码中包含一个 set()函数，用作生成测试数据，可以实时根据需要对测试数据进行更新。如果比较不同查找算法的时间消耗，则需注释掉 set(),无需对测试文件进行更新。排序算法分别使用一个函数去实现，在函数内部，每次当开始进行查找时，QueryPerformanceFrequency(获取时间频率)和 QueryPerformanceCounter 进行时间的精确计算。Test()函数通过以文件流对象的为参数的方式，从测试文件中读取测试数据，对读到数组中的数据进行排序操作。主函数利用 for 循环和 switch 语句分别进行 5 次不同规模的数据查找。

实验为了更清晰对排序算法进行比较，实现了三种排序算法，一种较慢的，稳定的冒泡排序，两种较快，一种稳定的归并排序，另外一种不稳定的快速排序。实验代码快速排序和归并排序是在一个 cpp 中实现，冒泡排序单独在一个 cpp 中实现，测试输出文件分别进行存储。

二. 实验过程记录

2019 年 12 月 23 日

快速排序算法的优化实现：快速排序算法在对于数据序列较为有序的情况下，排序的效率较低，可以在实现基本快排的基础上对其进行一个优化处理，选取枢轴关键字的时候，可以选取每次处理序列的中间值，可以在一定程度上减小有序序列在排序过程中蜕变成冒泡排序的时间消耗。实验代码采用的就是取序列中间值来进行处理的。类似于这样的想法，枢轴值还可以用中间三个数，随机选取一个来处理，这样可以大大优化最差情况的时间消耗。时间性能分析：快速排序一般情况下时间复杂度为 0（nlog n），但是当排序序列的初始状态为按照关键字有序排序时，快速排序将蜕化为冒泡排序，其时间复杂度为 0（n^2）。

如何记录时间：

```
QueryPerformanceFrequency(&fre);//获得时钟频率
QueryPerformanceCounter(&start);
Quick_sort(l,0,N-1);
QueryPerformanceCounter(&end);
dft=(double)(end.QuadPart-start.QuadPart)/(double)fre.QuadPart;
```

dft 为测得的时间，注意需要加相关头文件#include <windows.h>。

文件流的输入与输出：头文件#include <fstream>。文件流读取数据：定义一个 ifsteam 对象，用来进行文件的读取即将文件中的数据读取到控制台中。ifstream in(“文件名.txt”,ios::in)//ios::in 表示写入，然后进行读取,例如将文件中的数字读到数组 a[n]中，即 in>>a[i]。文件流写入数据：定义一个 ofsteam 对象，用来进行文件的写入即将输入或者是处理过的信息存入到文件中去。ofstream out（“文件名.txt”,ios::out）//ios::out 表示读取，然后进行写入，

例如将数组 a[n]中的数据写入文件中，即 out<<a[i];out<<"写入数据"。

2019 年 12 月 26 日

归并排序：实验中采用的是 2-路归并排序，基本思想和快速排序类似，即分治法，将一个序列分成两个等长的子序列，分别对这两个子序列递归地调用归并排序算法，再将两个位置相邻的记录的有序子序列归并为一个记录的有序序列。优化处理：对于处理到序列长度较小时可以采用插入排序来处理。时间性能分析：当被排元素数目为 n 时，递归的深度是 log(n),第一层递归可以看做是对一个长度为 n 的数组排序，下一层是对两个长度为 n/2 的数组排序，以此类推，最后一层是对 n 个长度为 1 的子数组进行排序，而在每一层递归中都需要 n 步，所以总的时间代价为 O（nlog n）。

二. 心得和问题

2019 年 12 月 23-27 日

实验 9 对不同规模下的数据进行处理，在处理上基本和实验 8 的思路一样，不同的是对调用的函数进行修改。对两种排序算法进行读取操作，运行完成一种排序后，需要再次打开文件进行读取操作，以免出现前次排序影响下一次排序的时间。

对排序算法进行运算时间的计算时，要注意不要在其中加入 cout,cin 等语句，避免因为 C++输入输出消耗时间过长而影响实验中排序算法的时间计算。