SE106 Homework5B Report

熊伟伦 5120379076

1. 分析算法时间复杂度

归并排序需要将数据每次分成 2 份，因此一共需要进行 $\log_2 N$ 次分配操作，每次操作都要对所有数据进行重新排序，时间复杂度为 N ，因此时间复杂度为 $N \log N$ 。

快速排序与之类似，从最好情况来看（每次取到中位数）每次的大小数据分列时间复杂度为 $\log_2 N$ ，对每次数据的重新排列的时间复杂度为 N ，所以一般情况平均下来时间复杂度也为 $N \log N$ 。最差情况每次取得的 pivot 为最大或最小值，时间复杂度为 $N * N$ 。

2. 运行时间数据

具体运行环境和时间数据见文件 sort_time_data

运行环境：

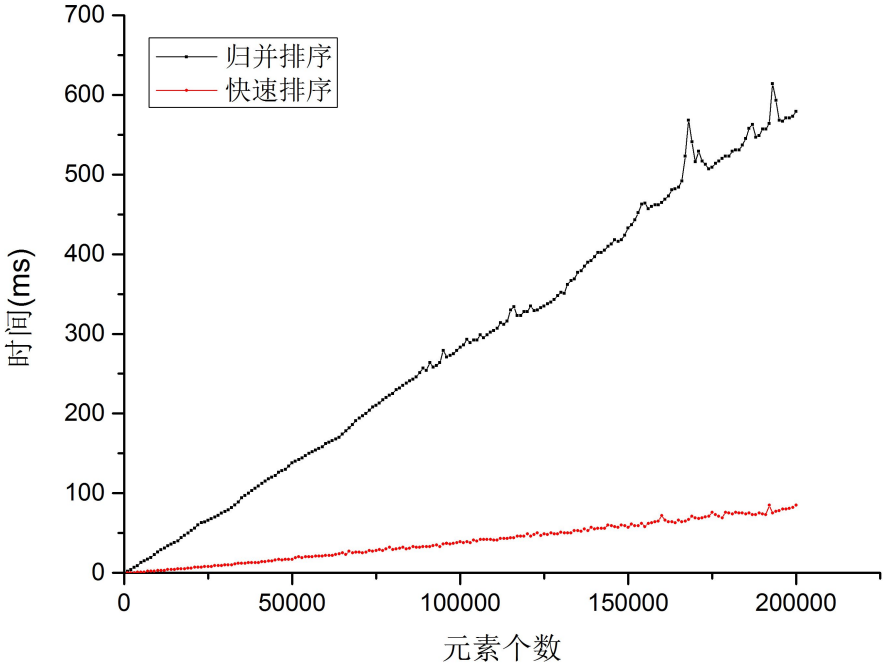
Windows 7

Intel Core i5-3210M

MinGW 4.8　g++ 4.8.0

测试数据个数从 1000 到 200000，每间隔 1000 个元素测试一次
每个排序共测试 200 组数据
每组数据测 10 次取平均值

得到图像如下 （来自 OriginPro 9.0)



3. 数据分析

两种排序时间上的差异大致是常数倍的关系，并没有因为元素个数的增加产生差异，符合都为 $N \log N$ 的时间复杂度。

并且时间与元素个数的图像接近直线，但随着元素个数的增加略有上升。符合 $N \log N$ 的特性，因此可认为时间复杂度的分析正确。

从以上数据看出我写的快速排序比归并排序快很多（约 7 倍），虽然理论上归并排序的元素操作次数和快速排序最好情况一样，一般情况下快速排序的操作次数比归并排序多，但是快速排序依然比归并排序快。对此分析原因。

原因 1. 归并排序（非 inplace 版本）申请了额外的内存空间，读写内存的次数更多，耗费了更多的时间。并且在程序执行的时候从任务管理器查看所占用的内存，归并排序确实占用了比快速排序更多的内存。

原因 2. 归并排序先从底层开始排序，最后越来越多的元素进行合并，越后期操作的元素个数越多，越对 cache 不友好。

而快速排序则相反，从顶层开始排，最终排序的元素个数越来越少，操作次数越来越多，这样比较而言，比归并排序对 cache 友好的多。

但从维基百科的“排序算法”词条中看到，测试得到的快速排序（inplace 版本）比归并排序（非 inplace 版本）快大约 3 倍，而我的数据快了 7 倍。

对比代码，可能的原因是我为了方便使用了 vector 来保存生成的随机数，而在归并排序

的实现中大量使用了 push_back 函数，开销比较大，所以我所写的归并排序比快速排序慢很多。

4. 总结

归并排序和快速排序本质上都是分块递归的思想进行排序。

不同点在于归并排序是自下而上的，先进行分组，再排好后从下至上合并。而快速排序是自上而下的，先分组，排好之后再分组，不用再次合并。

虽然归并排序看似是最合理的分区间，理论上的操作次数比快速排序少，但因为操作顺序的不同，造成了额外内存开销和 cache 友好的差异，同时导致了速度上的差异。

归并排序也有空间复杂度为 O(1)的版本，但是需要进行更多的排序次数。