# 算法设计实验报告



|  |  |
| --- | --- |
| **实验名称：** | **递归与分治策略综合实验** |
| **姓名：** | **蒋雪枫** |
| **班级：** | **2017211314** |
| **学号：** | **2017213508** |
| **专业：** | **网络工程** |
| **指导教师：** | **叶文老师** |

**2019年10月16日**

1. 综述

递归，即间接或直接调用自身的思想；分治，即分而治之，通过减小问题的规模来提高我们求解问题的效率与时间。在学生这两年的计算机方向的学习中，深深感受到这两种算法思想的重要性。同学们不论是在参加各种算法比赛，还是在假期去企业实习(当然大部分是刷题准备面试的时候)，都写过不少相关的代码，虽然或许并没有有意去深入了解其背后的思想。本月中，我们跟随叶文老师的教学，通过一些典型的案例深入研究其算法思想。在本次实验中，我们完成了快速排序(以及其随机化处理)，归并排序(以及其非递归实现)，线性时间选择以及平面最近点对的算法，并通过LTE基站的较为工程化的数据来辅助验证了算法的正确性。

比起代码本身，算法的思想才是最为重要的，所以报告中未贴出全部代码，只贴出了核心算法部分，源代码见附件。

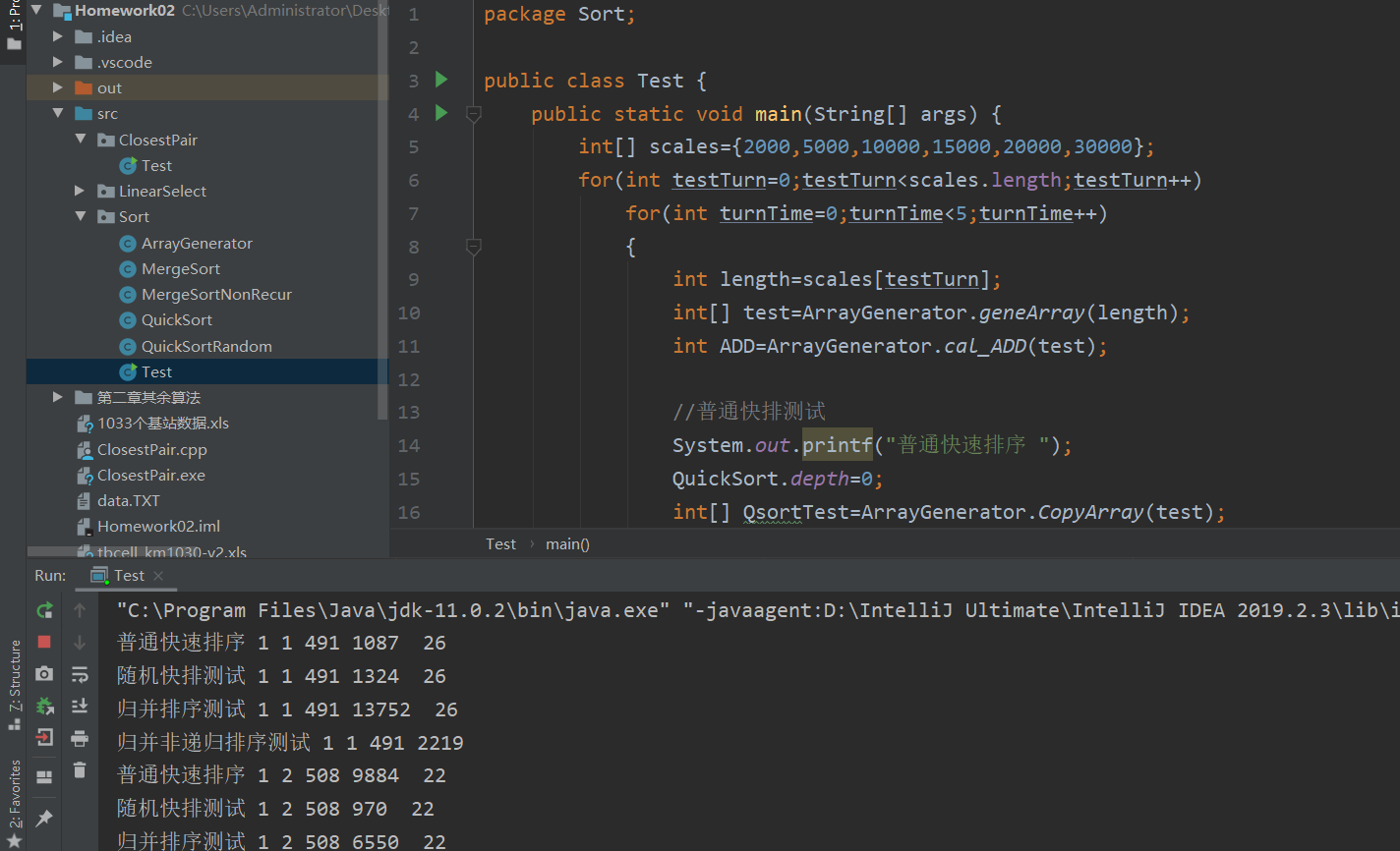
1. 文件说明

本次作业主要使用Java完成，最近点对算法使用C++算法完成。在主文件目录下，src目录下是Sort package(内有四种算法以及计算其运行效率的测试类Test，其余的Java包同理)、LinearSelect package以及ClosestPair package。主文件目录还有ClosestPair.cpp以及其对应的可执行文件，使用C++11编写。其余是一些算法所需要的文件资源。另外，学生在这里也另外实现了第二章所讲的其他算法，在另外的附录中，兴趣使然： ）

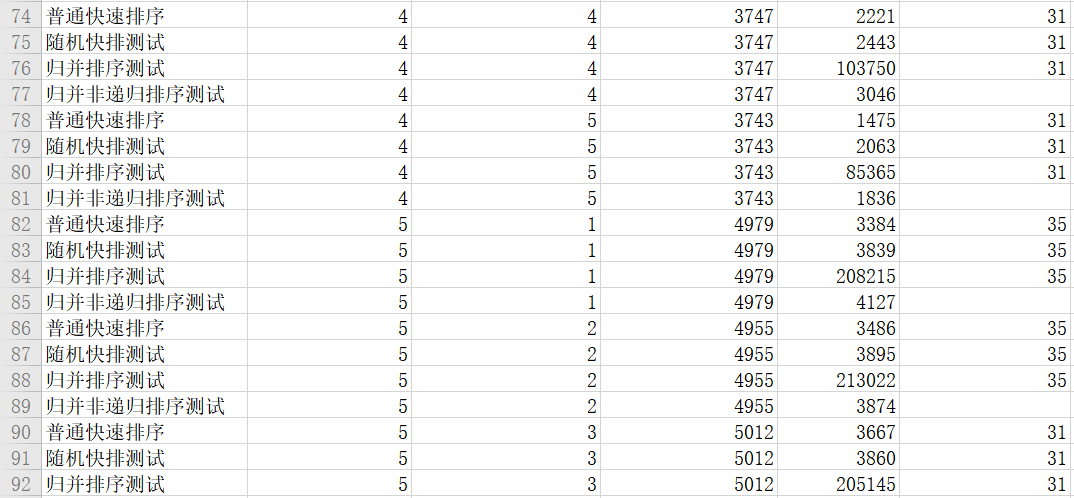
1. 四种排序算法运行以及结果说明

首先，在我们的测试类中，我们生成了不同长度的数据，一共有6种长度，并且每一组长度我们都测试了五组样本。拿第一行举例子，1-1为外部测试的轮数与样本序号，491为该样本的Average Distinct Degree，1087为运行时间，单位为微妙，26为递归最深层次。

排序复杂度使用O(n2)方法遍历统计，递归深度通过设置全局变量实现，并且在调用自身时候传递参数k+1，用于记录当前深度情况。

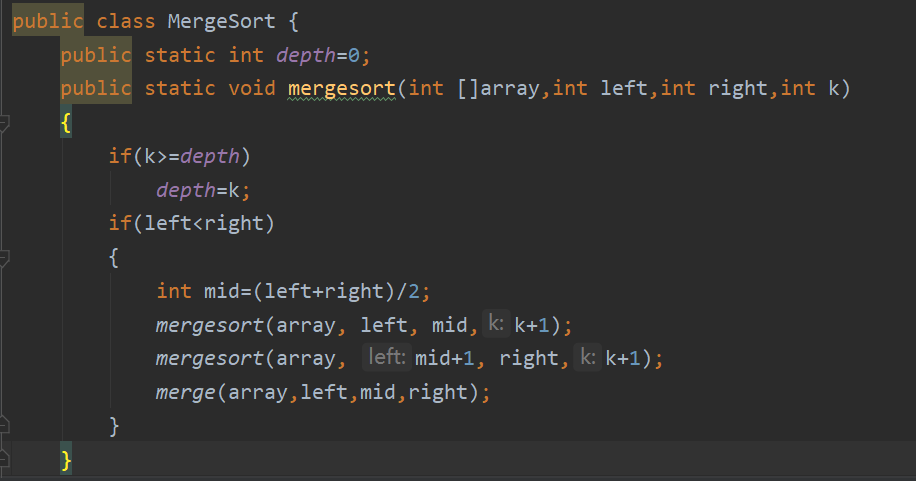


对于排序算法的运行时间详情，均已经把数据连接到文件夹当中，可以进行查看。这里截图仅为部分，但基本服从整体的分布。

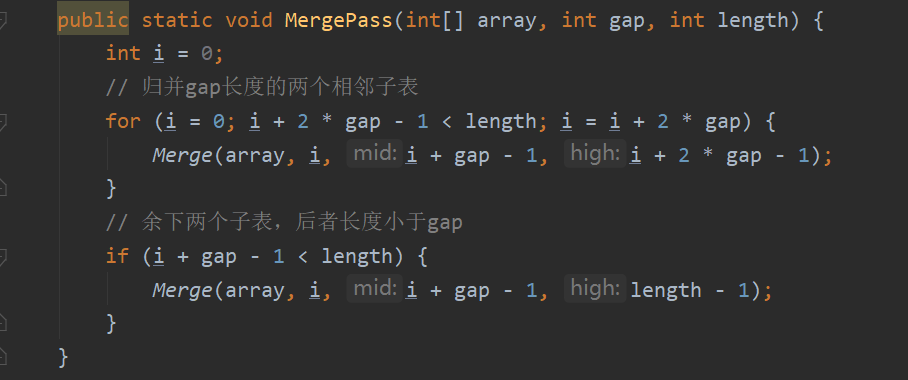


在学生自己编写的代码中，测试发现递归式的归并排序时间较长，再咨询了其余同学后，得知他们也遇到了同样的情况，我将他们用Java实现的代码也丢进了我的测试类，发现时间都很长，比较新奇，至于原因，我们之后在总结部分再谈。而其余算法都差不太多，甚至觉得随机优化的快速排序甚至整体上比不过最初的快速排序，原因我们也在后面的实验总结来谈。

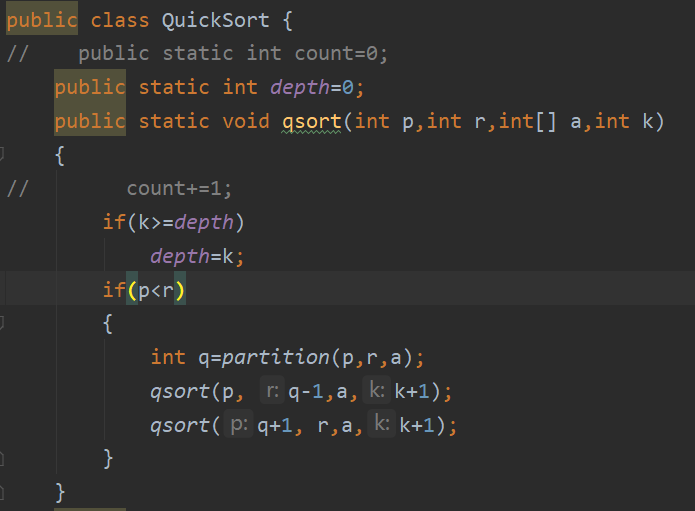
这里附上这四种排序的核心算法：



归并排序



非递归的归并排序中MergePass



核心代码为四行的快速排序(其核心是划分函数)

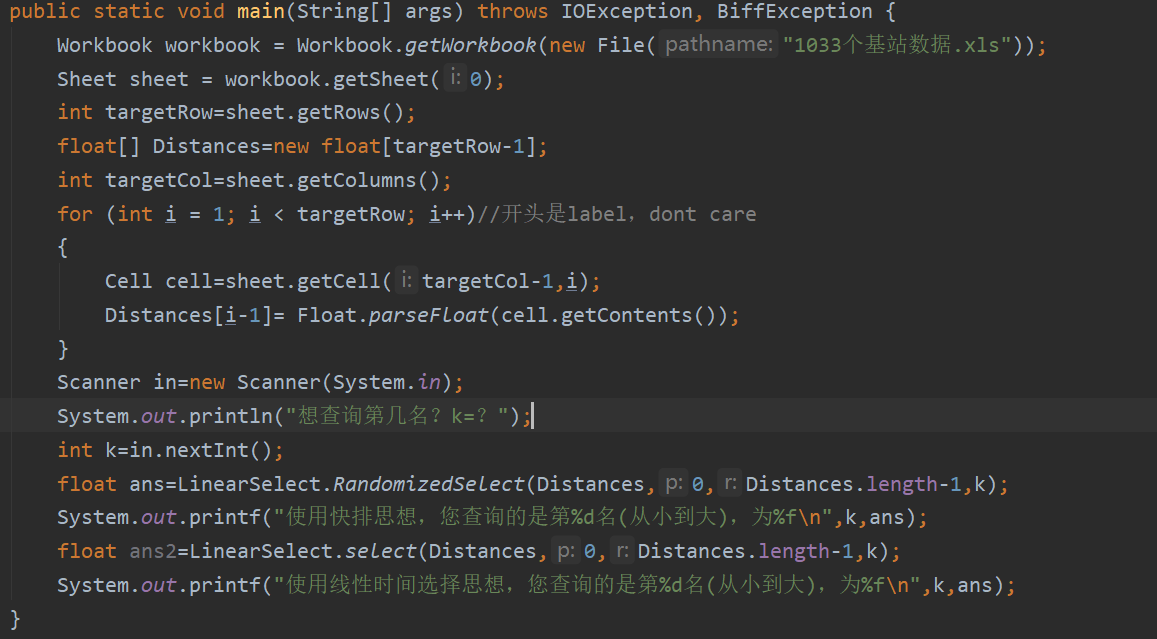


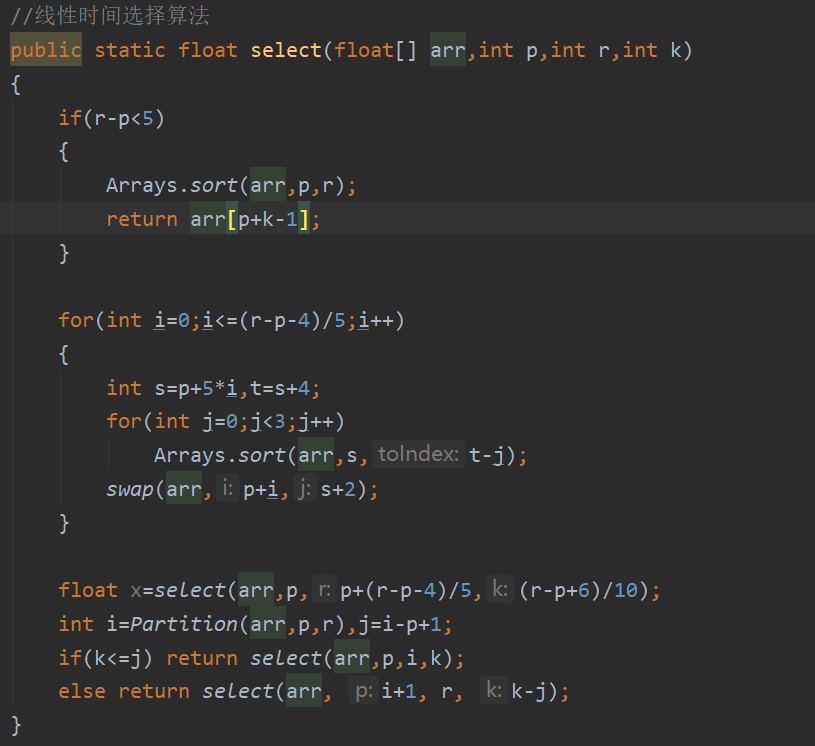
随机优化的快速排序

1. 线性时间选择以及最近点对问题

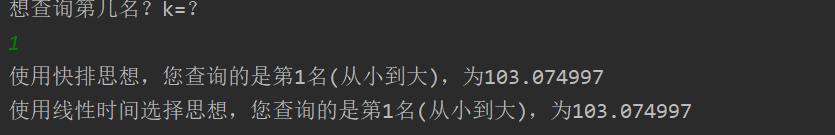
对于本问题，让我学习到的最重要的一个是算法本身的思想，一个是Java与Excel读写的交互。在实现时，学生采用了快排思想和书上线性时间选择求MinK的两个实现，结果一致，时间无疑是O(n)优于O(n2)。

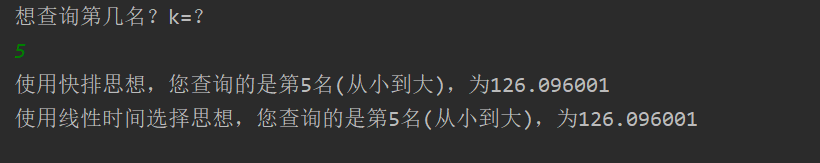
有一说一，线性时间选择算法将数学和算法结合得非常巧妙，要是没有学这一章，估计这辈子都想不到。同时，主定理也很重要，只是课上老师没怎么强调。当然，其实借助快排思想也挺快的，只是没有优化到O(n)。另一方面，我们可以使用最大最小堆（Python内的heapq可以很快实现这一要求）来解决MinK问题，当然它用得最多的情况还是滑动窗口。

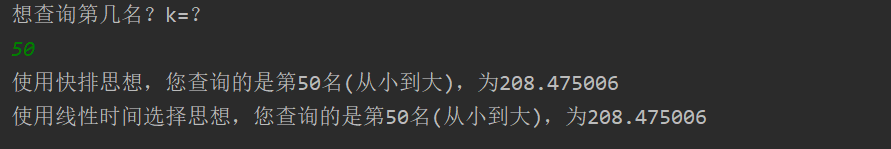




当我们查询k=5，50，1的时候，结果分别如下：





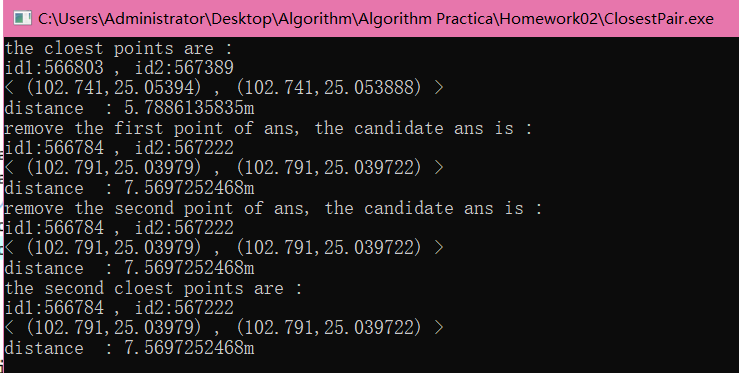


随后是最近点对问题求解：



而最近点对问题也是递归思想的一次升华，当我们最开始开始读到分开为两个区域求出d，再在横坐标的Xmid-d，Xmid+d区间内继续寻优，考虑到该区域可能集中了所有点，所以我们进一步优化，利用鸽巢原理，筛选周围的点。之后的学术论文中，甚至提出了可以继续优化的想法，这种想法是“画圆”，但其意义也没有”鸽巢原理“思想具有重大意义了。

使用C++11实现的最后版本的最近点对，结果如下：



1. 问题分析与思考
2. 为什么随机优化的快速排序会整体显得更慢？学生认为，这里是我们在排序的时候都需要传参，Java是以数组整体进行传参，而我们在外面调用了又一层函数，可能就耽误了很多时间。可以进行代码优化。
3. 为什么有时候刚开始排序的时候，第一轮第一组时间会久一些？学生认为是生成Random类并第一次调用的时候比较费时间，并且另外编写了用于验证该想法的测试脚本也佐证了这一猜想。所以在最后分析的时候，可以不用参考第一次调用Random的第一个数组长度的第一轮第一个样本。
4. 为什么最后用C++实现的最近点对？老实说，用Java没有De出Bug在哪里，只知道是在递归调用closepair方法的时候爆栈了(throws StackOverflow Exception)。经过思考，学生认为，这并不是代码的问题，而就是爆栈了，因为传参都是开的对象数组。或许可以通过JVM调优来解决该问题。
5. 最近点对问题为什么跑出来是0？因为有一对基站经纬度一样。本次运行时是去掉了其中一个基站的信息，因为个人觉得莫得意义。有一说一，其实用KD tree解决这种问题也挺快的，实在不行暴力O(n2)对于这种数据量也是可以做到的。但该算法的思想的确给了我很大的启发。
6. 排序的结果在哪？见文件夹内的excel文件。一目了然，无需多于分析。
7. 还可以怎么优化？可能需要多写点Java代码，读点JVM了。
8. 这些算法都是最佳的算法吗？显然不是。但是它们都很有意义，给学生以启发，只是随着后面我们高级数据结构和算法的使用，或许会有更方便的解决方案。

【**备注】详细的代码在文件夹中。**

1. 实验小结

以后再说起递归与分治，再也不会只知道Hanoi塔五行核心代码的例子和递归版的归并排序了。另外，也在实践中提高了自己Java编程与使用断点调试Debug的能力，也是头一次使用jxl这个jar包，在idea中导入，实现了Excel与Java之间的交互。