**深 圳 大 学 实 验 报 告**

**课程名称： 算法设计与分析**

**实验名称： 排序算法性能分析**

**学院： 计算机与软件学院 专业： 软件工程**

**报告人： 蔡晓鑫 学号： 2017151019**

**班级： 软件工程01**

**指导教师： 杨煊**

**实验时间： 2018/3/7——2015/3/25**

**实验报告提交时间： 2018/**

**教务处制**

## 一、实验目的

1. 掌握选择排序、冒泡排序、合并排序、快速排序、插入排序算法原理
2. 掌握不同排序算法时间效率的经验分析方法，验证理论分析与经验分析的一致性。

## 二、实验概述

排序问题要求我们按照升序排列给定列表中的数据项，目前为止，已有多种排序算法提出。本实验要求掌握选择排序、冒泡排序、合并排序、快速排序、插入排序算法原理，并进行代码实现。通过对大量样本的测试结果，统计不同排序算法的时间效率与输入规模的关系，通过经验分析方法，展示不同排序算法的时间复杂度，并与理论分析的基本运算次数做比较，验证理论分析结论的正确性。

## 三、实验内容。

1、实现选择排序、冒泡排序、合并排序、快速排序、插入排序算法；

2、以待排序数组的大小n为输入规模，固定n，随机产生20组测试样本，统计不同排序算法在20个样本上的**平均运行时间**；

3、分别以n=10000, n=20000, n=30000, n=40000, n=50000等等，重复2的实验，画出不同排序算法在20个随机样本的平均运行时间与输入规模n的关系，如下图1所示；

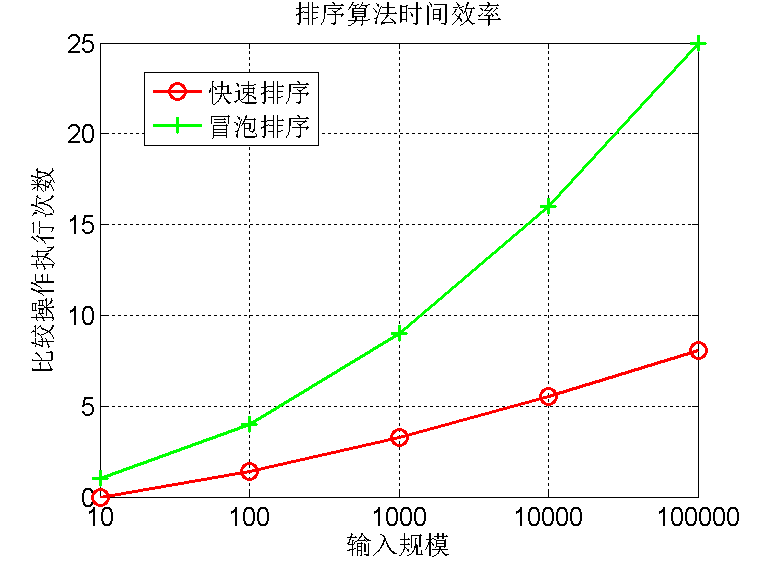


图1. 时间效率与输入规模n的关系图

4、画出理论效率分析的曲线和实测的效率曲线，注意：由于实测效率是运行时间，而理论效率是基本操作的执行次数，两者需要进行对应关系调整。调整思路：以输入规模为10000的数据运行时间为基准点，计算输入规模为其他值的理论运行时间，画出不同规模数据的理论运行时间曲线，并与实测的效率曲线进行比较。经验分析与理论分析是否一致？如果不一致，请解释存在的原因。

5、现在有10亿的数据（每个数据四个字节），请快速挑选出最大的十个数，并在小规模数据上验证算法的正确性。

## 四、提交材料

1. 代码：排序算法及注释
2. 实验报告：按照实验报告要求完成完整的实验报告。实验报告要包括五大部分：1）问题描述；2）求解问题的算法原理描述，包括算法的实现细节的解释；3）算法实现的核心伪代码（不可贴源码）；4）算法测试结果及效率分析；5）对求解这个问题的经验总结，注意不要写个人心得，要从算法解决问题的角度写你得到的结论是什么。

3、讲解用PPT

**一、实现选择排序、冒泡排序、合并排序、快速排序、插入排序算法；**

1. **选择排序**
2. **算法思想：  
   每一趟都从待排序序列中选出最小（或最大）元素，与待排序序列首元素交换，直至整个序列有序。**
3. **伪代码：**

**//以升序排序为例**

**//数组从1开始**

**selectSort(Array)**

**for i = 1 to Array.len – 1**

**index = i**

**for j = i + 1 to Array.len**

**if Array[j] < Array[index]**

**index = j**

**exchange Array[i] and Array[index]**

1. 冒泡排序
2. 算法思想：

每一趟都在待排序序列中沿着一个方向，对相邻元素进行比较，交换，使待排序序列的最大值（最小值）上浮到待排序序列的末尾。

1. 伪代码：

**//以升序排序为例**

**//数组从1开始**

**bubbleSort(Array)**

**for i = 1 to Array.len – 1**

**for j = 1 to Array.len – i – 1**

**if Array[j] > Array[j+1]**

**exchange Array[j] and Array[j+1]**

1. **合并排序**
2. **算法思想：**

**利用分治法的思想，将待排序序列不断划分成等长的子序列，直至序列长度为1，再按原划分路径回归合并成有序子序列，直到合并为原序列等长的有序序列。**

1. **伪代码：**
2. **“分”：序列（子序列）划分成等长子序列**

**mergeSort(Array, left, right)**

**if left < right**

**mid = (left + right) / 2**

**mergeSort(Array, left, mid)**

**mergeSort(Array, mid + 1, right)**

**//该操作属于“治”，当序列为1时，回归合并**

**merge(Array, left, mid, right)**

**1**

1. **“治”：当序列长度划分至1，也就认为有序的情况下，对两个相邻的子序列按序进行合并**

**merge(Array, left, mid, right)**

**//left\_array：暂存 Array[left : mid] 的数组**

**//right\_array：暂存 Array[mid : right]的数组**

**//left\_array, right\_array 的末尾都设置哨兵(一个不能被超越的值)，防止越界**

**i = 1 j = 1**

for k = left to right

if left\_array[i] <= right\_array[j]

Array[k] = left\_array[i++]

else

Array[k] = right\_array[j++]

1. 快速排序：
2. 算法思想：

通过一趟排序，将序列划分成两个部分，其中一部分的元素全部大于另一部分的元素，不断重复子任务，直至整个序列有序。

1. 伪代码：
2. 对序列（子序列）进行划分

Divide(Array, left, right)

p = Array[left]

while left < right

while left < right and p <= Array[right]

right = right – 1

if left < right

Array[left] = Array[right]

left = left + 1

while left < right and p > Array[left]

left = left + 1

if left < right

Array[right] = Array[left]

right = right – 1

Array[left] = p

Return left

1. 重复子任务

quickSort(Array, left, right)

if left < right

index = Divide(Array, left, right)

quickSort(Array, left, index-1)

quickSort(Array, index+1, right)

1. 插入排序：
2. 算法思想：  
   假定已有一部分序列有序，将待排序序列的第一个元素按序插入有序序列，直至整个序列有序
3. 伪代码：

insertSort(Array)

for i = 2 to Array.len

key = Array[i]

j = i – 1

while j > 0 and Array[j] > key

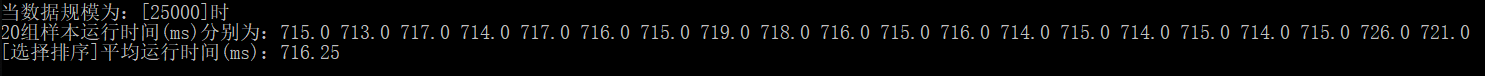
Array[j+1] = Array[j]

j—

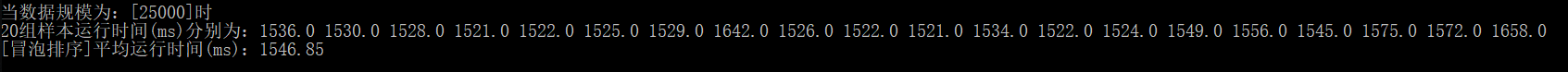
Array[j] = key

**二、以待排序数组的大小n为输入规模，固定n，随机产生20组测试样本，统计不同排序算法在20个样本上的平均运行时间**

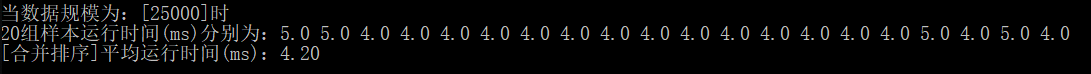
1. 选择排序：716.25 ms



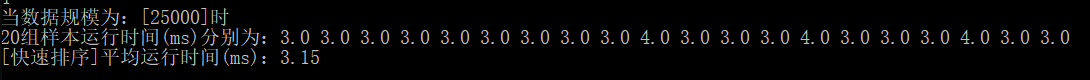
1. 冒泡排序：1546.85 ms



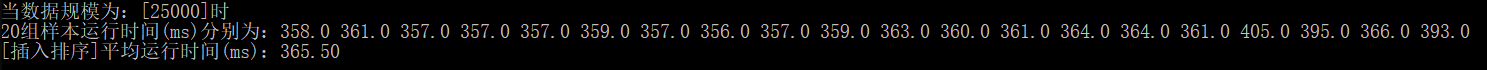
1. 合并排序：4.20 ms

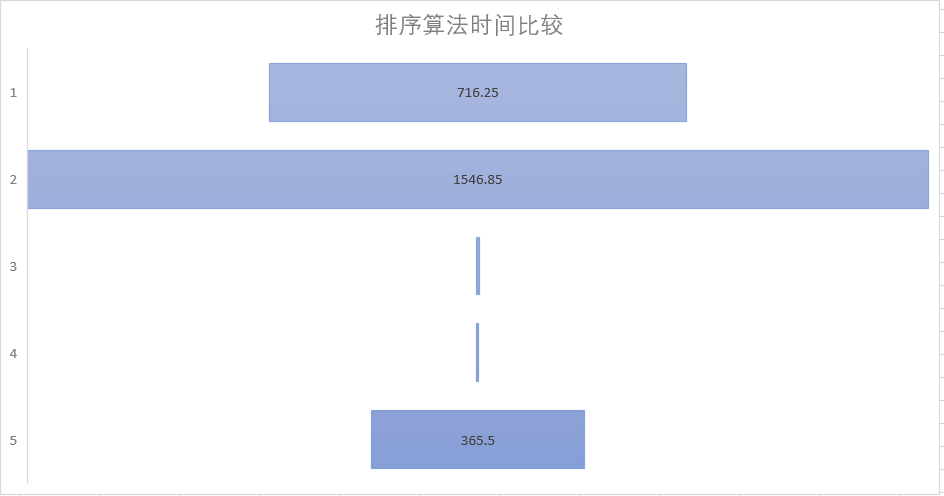


1. 快速排序：3.15 ms



1. 插入排序：365.50 ms





1.选择排序

2.冒泡排序

3.合并排序

4.快速排序

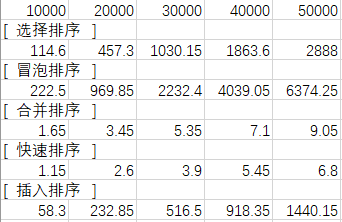
5.插入排序

3.15

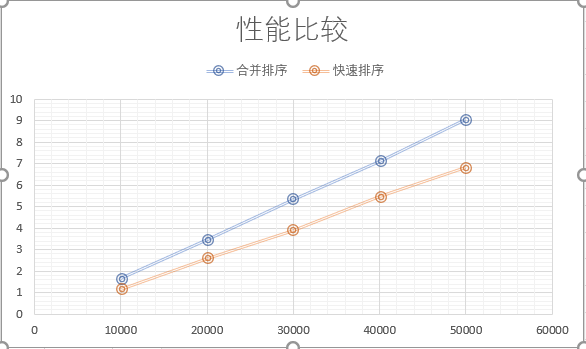
4.20

**三、分别以n=10000, n=20000, n=30000, n=40000, n=50000等等，重复2的实验，画出不同排序算法在20个随机样本的平均运行时间与输入规模n的关系**

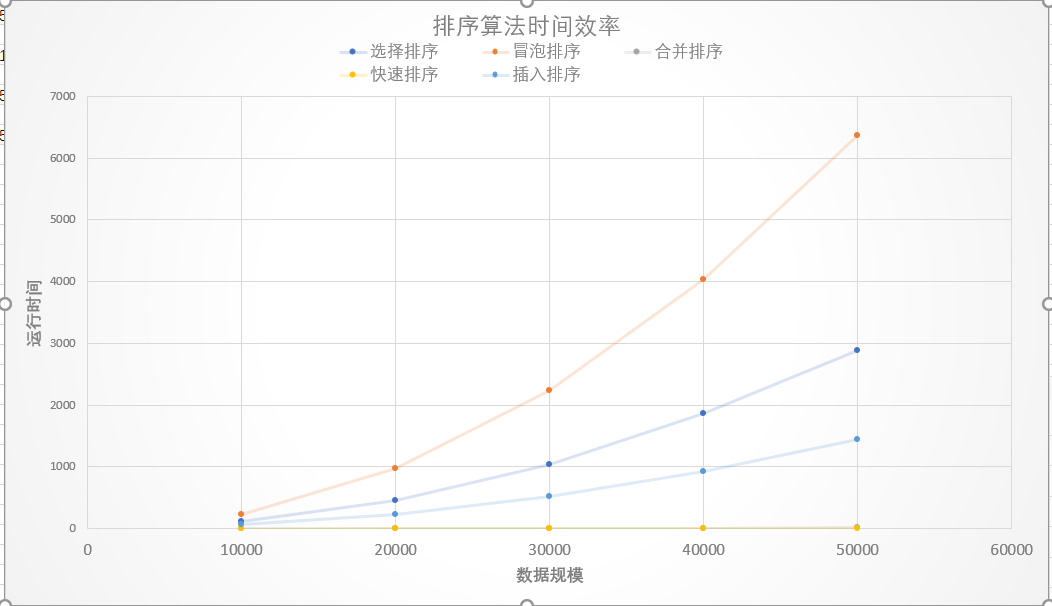
1. **不同算法在不同数据规模下的平均运行时间**

****

1. **快速排序与合并排序**

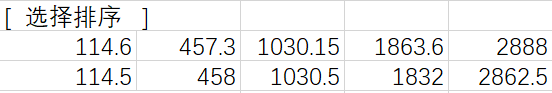
****

1. **五种排序算法比较**

****

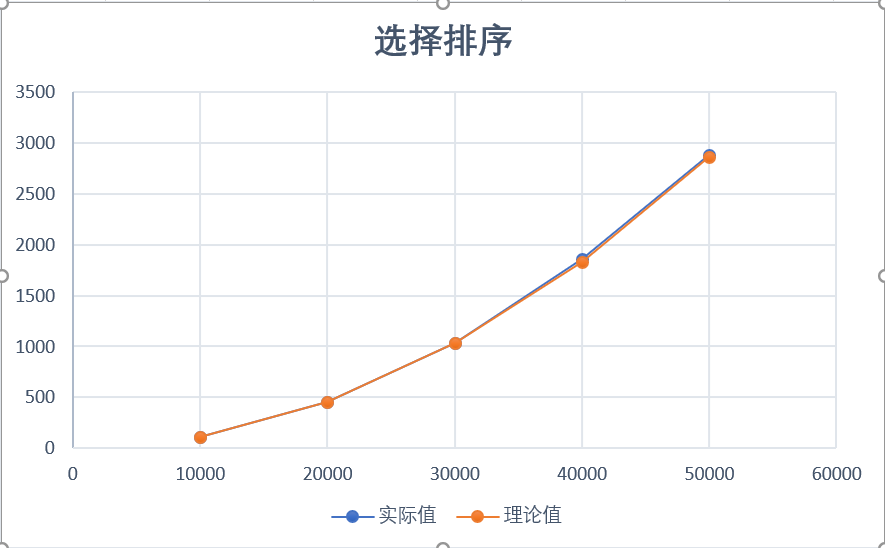
**四、画出理论效率分析的曲线和实测的效率曲线，注意：由于实测效率是运行时间，而理论效率是基本操作的执行次数，两者需要进行对应关系调整。调整思路：以输入规模为10000的数据运行时间为基准点，计算输入规模为其他值的理论运行时间，画出不同规模数据的理论运行时间曲线，并与实测的效率曲线进行比较。经验分析与理论分析是否一致？如果不一致，请解释存在的原因。**

1. **选择排序：**



**理论值**

**实际值**

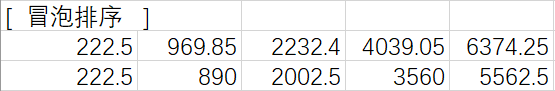


1. **分析：**

**从图像看，实际值散点图与理论值散点图能很好的拟合，基本符合θ(n²)**

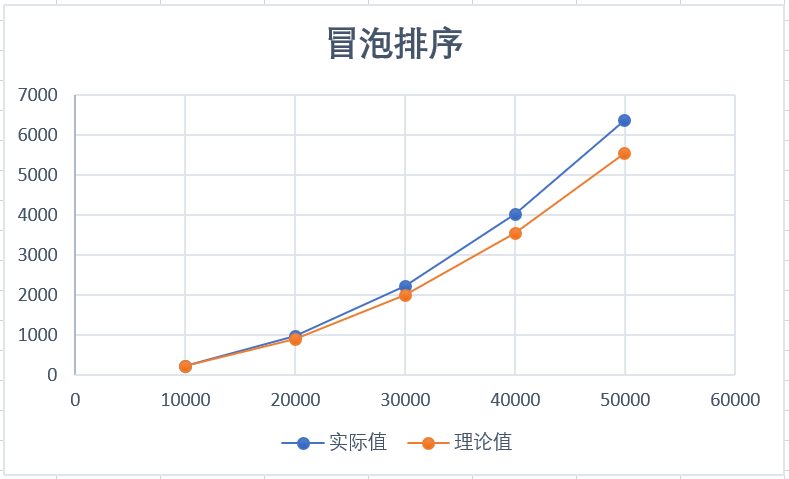
**从数据上看，实际值略大于理论值，可能是在相对小的数据规模下，电脑性能会有一定的影响。**

1. **冒泡排序**

****

**理论值**

**实际值**

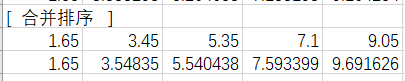
****

1. **分析：**

**从图像看，实际值散点图基本符合二阶多项式的增长趋势，基本符合θ(n²)**

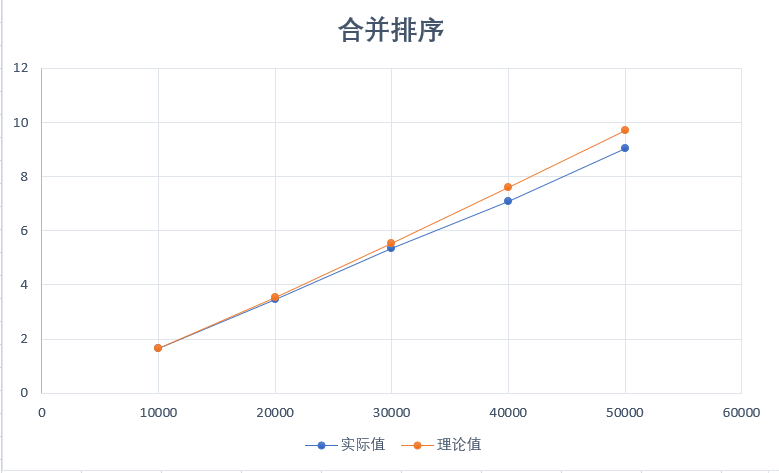
**从数据上看，实际值随着数据规模的增长，偏差逐渐拉大，这是由于数据交换指令达到一定数量规模而加大系统开销，但这部分开销时间在时间复杂度的计算中属于忽略部分。**

1. **合并排序：**

****

**实际值**

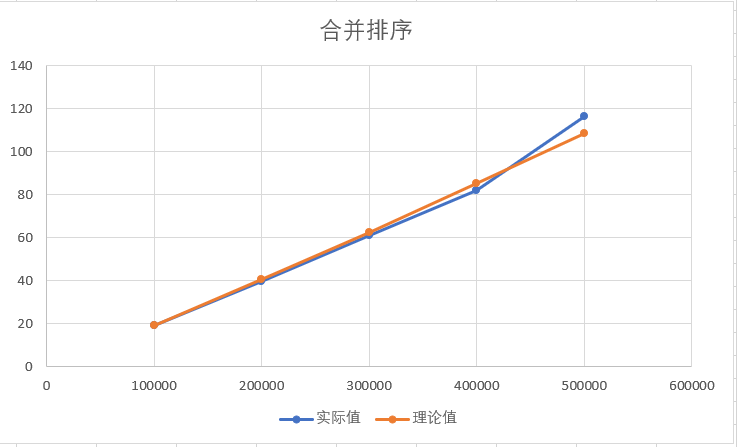
**理论值**

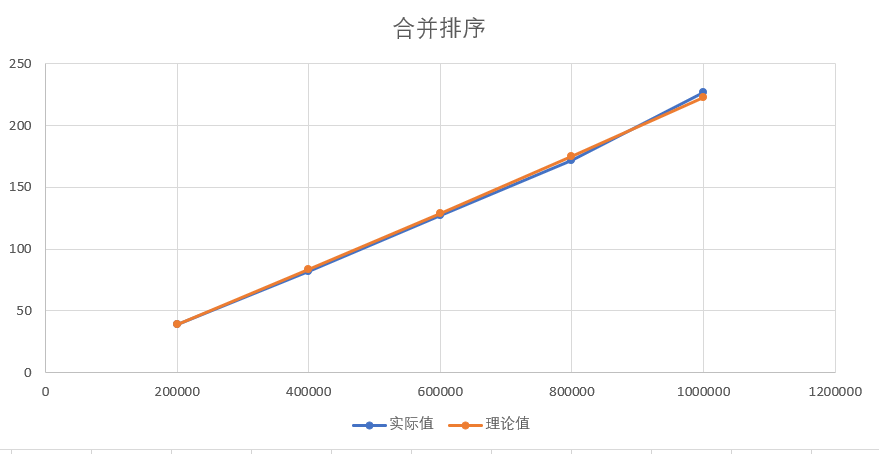
****

1. **数据分析：**

**从图像上看，实际值散点图基本符合线性增长，与θ(nlogn)的算法复杂度吻合**

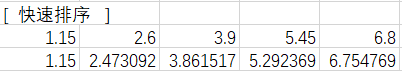
**从数据上看，实际值小于理论值，因为小规模的数据递归深度浅，所以递归的复杂度logn的时间消耗占的比重小，从而导致基准点实际值偏小。通过增大数据规模，可以减少这种误差的影响，实际值和理论值会更加拟合**

****数据规模10w-50w

数据规模20w-100w

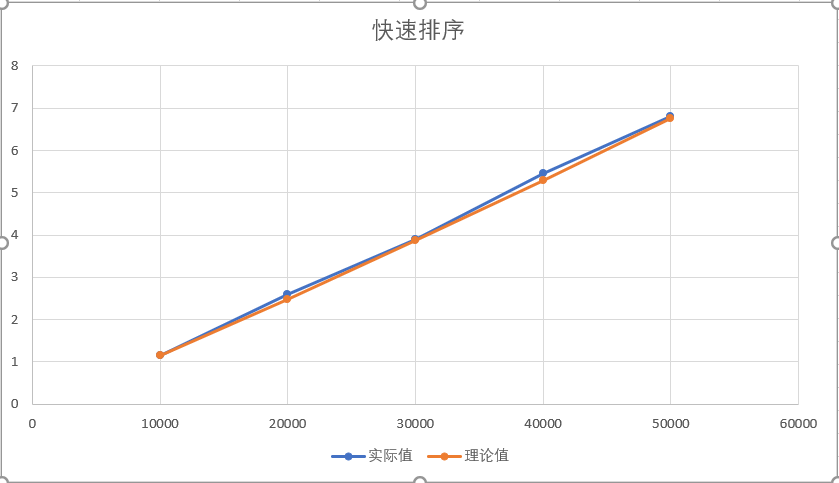
**在数据规模大的情况下，实际值和理论值基本一致，但实际值会偏大，原因可能是临时数组的分配，拷贝，递归层数增加导致系统消耗增加，从而增加运行时间**

1. **快速排序**

****

**理论值**

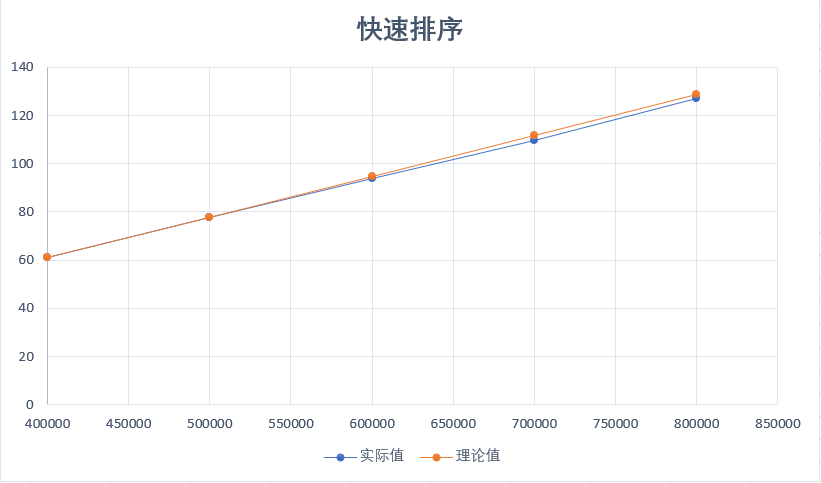
**实际值**

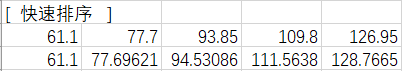
****

1. **数据分析：**

**从图像上看，实际值散点图和理论值散点图能很好的拟合，符合θ(nlogn)算法复杂度**

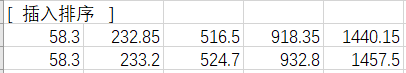
**从数据上看，实际值偏大于理论值，原因可能是数据的交换加大系统消耗，但是这部分消耗时间在算法复杂度的计算中被忽略。**

**40w-80w**

****

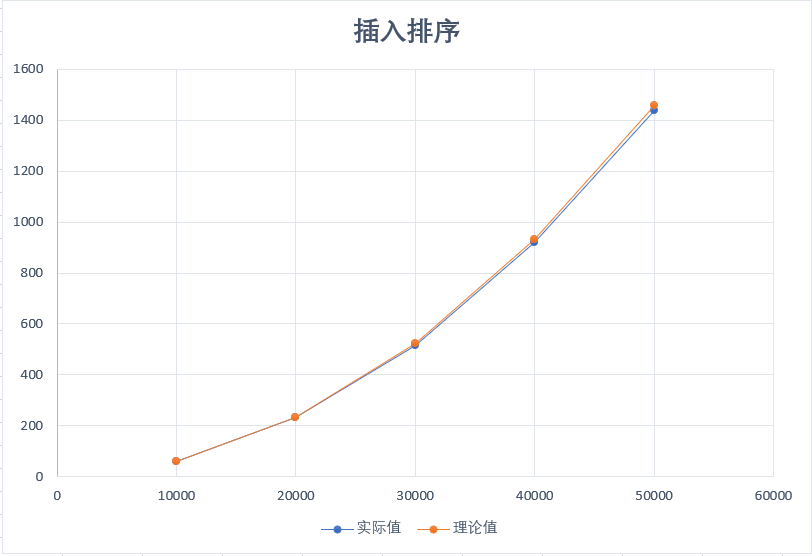
**达到一定数据规模后，理论值与实际值散点图能很好的拟合**

1. **插入排序**

****

**实际值**

**理论值**

****

1. **数据分析**

**从图像上看，实际值与二次增长趋势吻合，符合θ(n²)**

**从数据上看，实际值偏小于理论值，理论值是在数据规模1w的情况下推算出来的，在小数据规模下的排序时间受机器性能影响大，在此情况下1w的运行时间可能比理论情况偏小，从而导致推演的各个理论值偏小**

**四、现在有10亿的数据（每个数据四个字节），请快速挑选出最大的十个数，并在小规模数据上验证算法的正确性**

1. **设计算法思路：**
2. **借用快速排序特点**

**选定基准值后，数组划分成大于基准值/小于基准值**

1. **结合实际任务**

**不需完全排序，所以小于基准值的部分可弃，近似排除法**

1. **数据规模问题**

**分治思想，把10亿数据划分成100个一千万的子数组，每个子数组选取10个最大值**

1. **极端情况考虑**
2. **基准值选取过偏，所得的最大值元素小于10。在另一边的数组中执行子任务，再划分出一定数量的最大值元素，使得每个子任务都能得到10个最大值元素**
3. **编写伪代码**

**Index = getKey(Array,left,right)**

**If Index > 10**

**Quick\_halfSort(Array,left,Index-1)**

**Else if Index==10**

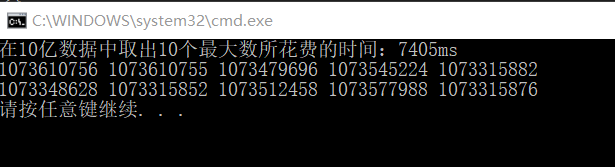
**get Top 10**

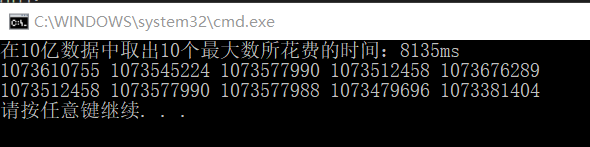
**Else**

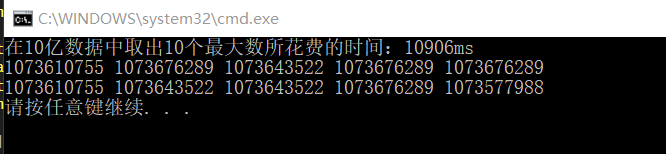
**get Top Index**

**Quick\_halfSort(Array,Index+1,right)//获得剩余的Top(10-Index)**

1. **运行结果**

****

****

****

**五、总结：**

1. 排序算法分析方面：用合理的数据获取较为准确的基准点能得到更好的拟合曲线。一般是根据算法的复杂度确定T1的数据规模，对于冒泡，插入，选择排序来说，运行时间比较长，所以选取的数据规模相对于快排，合并排序的数据规模小。快排和归并在选取1w作为基准点时，得到的理论时间往往小于实际时间，因为运行时间短，时间误差大，同时受到机器运行效率的影响的比重也会加大。
2. TopK问题求解：

分析问题很重要，通过对问题的分析，了解解决问题的最小代价。题目的要求很简单，就是在一堆数据中选取十个最大的数，那么避开全排序的坑，其实就变成了淘汰制，确定门槛，缩小搜索范围。但是有需要借助排序的思想，符合淘汰制，只有快排，每次挑选出一个值，弃掉小于的部分，直到得到10个数。这也就是所设计算法的任务。再而是分析数据规模，10e9一次性是存不下的，所以又得从基础的算法思想入手-分治法。分成100个1kw的子序列，每个子序列找10个值，最后整合再找出10个数，此时便是答案。

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |