**深 圳 大 学 实 验 报 告**

**课程名称：­ 算法设计与分析**

**实验项目名称： 排序算法性能分析**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 霍永凯**

**报告人：罗泽鸿 学号：2017153005 班级： 计科03**

**实验时间： 3.14-3.18**

**实验报告提交时间： 3.18**

**教务处制**

|  |  |
| --- | --- |
| **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。 | |
| **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。 | |
| **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（以下讲解以升序为例）。**   **1.选择排序**  1）选择排序的原理  选择排序是通过将数组中的元素，按位置下标从小到大的顺序，把他们逐一从无序变为有序的排序算法。算法思路如下，假设数组下标从0开始，共有n个数，则第一次，从n个数中找出最小的那个数，然后放到下标为0的位置，即第一个位置，此时数组的前1个元素已有序；第二次，从下标为1开始的n-1个数中找出最小的那个数，然后放到下标为1的位置，即第二个位置，此时数组的前2个元素已有序；重复以上这些步骤，一直到数组的下标为n-1开始的1个数找出最小的那个数，并把它放在下标为n-1的位置，使得最终数组的前n个元素已有序。到此为止，数组已排序完毕。注意，由于数组到下标为n-1的位置时，未排序的元素只剩下一个元素，此时无论怎么样都不需要更换位置了，所以数组下标遍历时，只需从0到n-2即可完成排序。  2）选择排序伪代码  from i=0 to i=n-2 n-1  pos=i; n-1  flag=1 c5 \*  from j=i+1 to j=n-1 c1=sum(n-1-i (i=0 to i=n-2))=n\*(n-1)/2  if a[pos]>a[j] c2=c1  then pos=j; c3  if a[j]<a[j-1] c6 \*  flag=0 c7 \*  if flag==1 c8 \*  break c9 \*  if(i!=pos) n-1  swap(a[i],a[pos]) c4  3）选择排序的理论时间效率  从以上伪代码可知，选择排序的需要两重循环来完成排序。在最好情况下，考虑加\*代码，数组已按从小到大排好，此时对于每个下标i，都不需要进行交换且有每个a[j]都>=a[j-1]，c5=0 c7=c7=c8=c9=1 c6=c2=c1=n-1此时c3=0 c4=0，所以总的时间复杂度为T=n-1+5；在最坏情况下，不考虑加\*代码，初始数组按降序排序，此时c3=c2 c4=3\*(n-1) 所以总的时间复杂度为T=6\*(n-1)+3\*n\*(n-1)/2=(3\*n/2+6)\*(n-1)。在最好情况下，选择排序时间复杂度为O(n)，在最坏情况下，选择排序的时间复杂度为O()。  **2.冒泡排序**  1）冒泡排序的原理  冒泡排序是通过把数组中的每一个元素，按数组下标从小到大的形式，通过两两交换，把大的元素逐次往数组下标大的位置上移动的排序算法。算法思路如下，假设数组下标从0开始，共有n个数。那么第一次数组的两两交换从下标0开始，如果遇到下标i的值比下标i+1的值大，那么这两个数需要进行交换，一直交换到下标n-2和n-1；第二次数组的两两交换从下标0开始，一直交换到下标n-3和n-2；重复以上交换过程，一直到第n-1次，对下标0和下标1进行交换。到此为止，此数组已完成排序。  2）冒泡排序伪代码  from i=0 to i=n-2 n-1  Flag=1 c4 \*  from j=0 to j=n-i-2 c1=sum(n-i-1(i=0 to i=n-2))=n\*(n-1)/2  if a[j]>a[j+1] c2=c1  swap(a[j],a[j+1]) c3  Flag=0 c5 \*  if flag==1 c6 \*  Break; c7 \*  3）冒泡排序的理论时间效率  从以上伪代码可知，冒泡排序的需要两重循环来完成排序。在最好情况下，数组已按从小到大排好，考虑加了\*号的代码，此时对于每一对相邻的元素，都不需要交换，此时c2=c1=n-1 c5=c3=0 c4=1c6=c7=1且在i=0循环后就跳出整个循环了，所以总的时间复杂度为T=4+2\*(n-1)；在最坏情况下，数组按降序排序，不考虑加\*的代码，此时对于每一个两两相邻的数组，都需要进行交换，此时c3=c2\*3 所以总的时间复杂度为T=n-1+5\*n\*(n-1)/2=(5\*n/2+1)\*(n-1)。在最好情况下，冒泡排序时间复杂度为O(n)，在最坏情况下，冒泡排序的时间复杂度为O()。  **3.合并排序**  1）合并排序的原理  合并排序是利用分治的思想，将数组平均分为两部分分别进行排序，排好序后再对两个部分进行合并排序的排序算法。此排序需要再开另外两个数组来临时储存元素。算法思路如下，先把数组分成两部分分别递归进行排序，递归的终止条件即为当前数组的左边界下标不小于右边界下标。对数组左半部分、右半部分排序后，再对整个数组排序，最终完成排序，概况来说，就是分解、求解、合并。  2）合并排序伪代码  mergeSort( left, right)  if left+1<right  mid=(left+right)/2 c1  mergeSort(left,mid) c2  mergeSort(mid,right) c3  merge(left,mid,right) c4  merge(left,mid,right)  n1=0  n2=0  from i=left to i=mid-1  L[n1++]=a[i]  from i=mid to i=right  R[n2++]=a[i]  i=0  j=0  from k=left to k=right-1  if L[i]<R[j]  a[k]=L[i] i++  else  a[k]=R[j] j++  3）合并排序的理论时间效率  从以上伪代码可知，合并排序需要用到二分递归，对与长度为n的数组需要递归次。由于归并排序用分治思想来排序的，它排序的时间复杂度与数组原来的排序情况没有关系。所以归并排序的最好情况、最坏情况以及平均情况时间复杂度都是相同的。归并排序总时间为分解时间+求解时间+合并时间。分解时间就是把上面伪代码中的c1 c2和c3操作，时间复杂度为3个时间单位，求解时间就是把规模为n的问题规模分成两个规模为n/2的子问题，即c2 c3里子问题的时间，时间为2T(n/2) 而合并时间为merge操作，即c4对应的步骤需要的时间，所需时间为O(n)级别，所以总时间为T(n)=2T(n/2)+O(n)。用递归树的思想求该T(n)假设第一层所需时间为x，则第二层所需时间为x/2+x/2=x，第3层所需时间为x/4+x/4+x/4+x/4=x，每一层所需时间都为x，由上面分析得要递归次，所以共需x\*次。  由此可得，在最好、最坏情况下，合并排序的时间复杂度都为O（）次。   1. **插入排序** 2. 插入排序的原理   插入排序是通过将数组中的元素，按长度从1到n的顺序，使他们逐一从小的已排好序的子序列，更新到这个数组都有序的排序算法。算法思路如下，假设数组从0开始，共有n个数，那么在第一次，对数组的第2个元素，即下标为1的元素进行操作，如果该元素比下标为0的元素小，则把它如在该元素前；在第二次，对数组的下标为2的元素进行操作，把它插入到前2个元素的恰当位置，使得前3个元素有序；重复以上操作过程，一直到把下标为n-1的元素插入到前n-1个数中，使得最终n个数有序。至此，就已完成了插入排序。   1. 插入排序伪代码   from i=1 to i=n n-1  tmp=a[i] j=i-1 (n-1)\*2  while tmp<a[j] c1  a[j+1]=a[j] j=j-1 c2  a[j+1]=tmp n-1   1. 插入排序的理论时间效率   从以上伪代码可知，插入排序需要有一个两个循环来完成排序，在最好情况下，数组已按从小到大排好，此时对于每个a[i]，a[i]都>=a[j] c1=n-1 c2=0  所以总的时间复杂度T=5\*(n-1)，在最坏情况下，数组按降序排序，次数对于每个当前的a[i]，a[i]都<=a[j](j<i)，即i前面的所有元素，位置都要移动，此时c2=2\*n\*(n-1)/2 c1=(n+2)\*(n-1)，所以总的时间复杂度为  T=4\*(n-1)+(n+2)\*(n-1)+2\*n\*(n-1)/2=(2\*n+6)\*(n-1) 在最好情况下，插入排序的时间复杂度为O(n)，在最坏情况下，插入排序时间复杂度为O()。   1. **快速排序**   1）快速排序的原理  快速排序是利用分治的思想，以边界元素的值（这里以左端点为例）为标杆，把数组分成小于该标杆元素的左部分和大于该标杆元素的右部分，然后再分别给两部分的子序列进行相同操作，直到相邻两个元素都实现有序的排序算法。算法思路具体如下，先取当前数组的左边界对应的元素作为分界线，把大于它的元素都放在标杆元素右边，把小于它的元素都放在标杆元素左边，然后递归处理标杆元素左边的子序列和标杆元素右边的子序列，直到所有相邻元素都有序为止。最终即可完成排序。   1. 快速排序伪代码   quickSort(l,r)  if l<r  m=partition(l,r)  quickSort(l,m-1)  quickSort(m+1,r)  partition(l,r)  v=a[l]  while l<r  while l<r And a[r]>=v  r--  a[l]=a[r]  while l<r And a[l]<=v  l++  a[r]=a[l]  a[l]=v  return l  3）快速排序的理论时间效率  从以上伪代码可知，快速排序需要将数组进行划分，在对两部分进行递归排序。假设数组下标从0开始，数组长度为n，在最好情况下，每次划分都很均衡，那么其递归树的深度为次，第一次划分将数组分为两部分，其划分时间为n，可知在最好情况下T(n)=2T(n/2)+n T(1)=0 由此可推出T(n)=O()；在最坏情况下，数组的排序为降序排序，每次划分只得到一个比上一次划分少了一个数的子序列，整个过程需要执行n-1次调用，且第i次划分需要n-i次关键字的比较，才能找出划分枢轴，整个过程所需时间为T==n\*(n-1)/2，所以在最坏情况下的时间复杂度为O()。     1. **各排序算法的实验结果**   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；这里以n=10000, n=20000, n=30000, n=40000, n=50000做为数据规模的大小为例，以下排序过程，时间记录都以毫秒为单位。   1. **选择排序**   以下是用选择排序在不同数据规模下的平均时间耗费情况    **图1.1选择排序在不同数据规模的时耗**  根据以上数据整理成表格和曲线，以便对数据进行分析    **表1.2选择排序在不同数据规模的时耗**    **图1.3选择排序在不同数据规模的时耗**  **在函数图形上**，函数的增长速度基本满足二次增长函数模型，与其最坏情况下时间复杂度O()相吻合；  **与理论对比**，利用上述理论分析的时间T=6\*(n-1)+3\*n\*(n-1)/2=(3\*n/2+6)\*(n-1)  来计算（10000起点时间取相同来计算），可得到以下实际耗时和理论耗时，其相对误差大概为0.01%的规模，误差相对较小，即实验得到的排序时间与理论情况基本相同；从下述函数图像和数据对比，还可发现，实际耗时与理论耗时相互浮动，无明显大小关系。      **表1.4选择排序的实际耗时与理论耗时比较**    **图1.5选择排序的实际耗时与理论耗时比较**   1. **冒泡排序**   以下是用冒泡排序在不同数据规模下的平均时间耗费情况    **图2.1冒泡排序在不同数据规模下的时耗**  根据以上数据整理成表格和曲线，以便对数据进行分析    **表2.2冒泡排序在不同数据规模下的时耗**      **图2.3冒泡排序在不同数据规模下的时耗**  **在函数图形上**，上述冒泡排序函数的增长速度基本满足二次增长函数模型，与其最坏情况下时间复杂度O()相吻合；  **与理论对比**，利用上述理论分析的时间T=n-1+5\*n\*(n-1)/2=(5\*n/2+1)\*(n-1)来计算（10000起点时间取相同来计算），可得到以下实际耗时和理论耗时，其相对误差大概为10%到40%的范围，误差比较大，且由以下图像对比和表格可看出，理论耗时明显低于实际耗时。不考虑实验的偶然误差，导致该现象的可能原因是冒泡排序的实际耗时总体会比理论值高，与选择排序对比，选择排序在实际耗时会小一些。    **表2.4冒泡排序的实际耗时与理论耗时比较**    **图2.5冒泡排序的实际耗时与理论耗时比较**   1. **合并排序**   以下是用合并排序在不同数据规模下的平均时间耗费情况    **图3.1合并排序在不同数据规模下的时耗**  根据以上数据整理成表格和曲线，以便对数据进行分析    **表3.2合并排序在不同数据规模下的时耗**      **图3.3合并排序在不同数据规模下的时耗**  **在函数图形上**，上述合并排序函数的增长速度相对缓慢，与其理论情况下时间复杂度O()大致吻合；  **与理论对比**，利用上述理论分析的理论时间T=来计算（10000起点时间取相同来计算），可得到以下实际耗时和理论耗时，其相对误差范围为25%-29%，不同数据规模的误差相距范围比较比较小，且其实际耗时明显比理论耗时小，可知合并函数在实际排序过程实际排序时间会比理论时间少，和冒泡、选择排序相比排序时间缩小很多。      **表3.4合并排序的实际耗时与理论耗时比较**    **图3.5合并排序的实际耗时与理论耗时比较**   1. **插入排序**   以下是用插入排序在不同数据规模下的平均时间耗费情况    **图4.1插入排序在不同数据规模下的时耗**  根据以上数据整理成表格和曲线，以便对数据进行分析    **表4.2插入排序在不同数据规模下的时耗**      **图4.3插入排序在不同数据规模下的时耗**  **在函数图形上**，上述插入排序函数的增长速度基本满足二次增长函数模型，与其最坏情况下时间复杂度O()相吻合；  **与理论对比**，利用上述理的时间T=4\*(n-1)+(n+2)\*(n-1)+2\*n\*(n-1)/2=(2\*n+6)\*(n-1)来计算（10000起点时间取相同来计算），可得到以下实际耗时和理论耗时；由以下图像和表格数据对比可知，插入排序时间与理论排序时间的相对误差为1%-17%，且随着数据规模的增大，实际耗时与理论耗时相差越来越多，同是O()的时间复杂度，插入排序的实际时间复杂度比选择、冒泡小一些。      **表4.4插入排序的实际耗时与理论耗时比较**    **图4.5插入排序的实际耗时与理论耗时比较**  5、**快速排序**  以下是用快速排序在不同数据规模下的平均时间耗费情况    **图5.1快速排序在不同数据规模下的时耗**  根据以上数据整理成表格和曲线，以便对数据进行分析    **表5.2快速排序在不同数据规模下的时耗**      **图5.3快速排序在不同数据规模下的时耗**  **在函数图形上**，上述快速排序函数的增长速度相对缓慢，与其理论情况下时间复杂度O()大致吻合；  **与理论对比**，利用上述理论分析的理论时间T=来计算（10000起点时间取相同来计算），可得到以下实际耗时和理论耗时，与理论耗时相比，实际耗时的误差范围为11%左右，且实际耗时比理论耗时低，并随着数据规模的增大，实际耗时与理论耗时的时间相差越来越大，同样是时间复杂度为O()的排序，快速排序比合并排序的实际消耗时间低一些。      **表5.4快速排序的实际耗时与理论耗时比较**    **图5.5快速排序的实际耗时与理论耗时比较**   1. **各排序实际消耗时间的对比**   汇总以上各排序在10000到50000的数据规模下的实际消耗时间，把它们的耗时整理如下图表一起对比，可主观的感受不同排序的算法耗时差别。首先，相比时间复杂度为O()的选择、冒泡、插入排序，时间复杂度为O()的合并排序、快速排序所消耗时间完成低了很多。选择、冒泡、插入排序的时间复杂度都是在1000ms以上的，而合并、快速排序所需时间都是在10ms的级别的。而同样是时间复杂度为O()的排序算法，冒泡排序消耗时间相对较高，在50000的规模上，平均消耗时间为13s左右；选择排序次之，在50000的规模上，平均消耗时间为4s左右；相比下，插入排序最优，在50000的数据规模下，消耗时间为2-3s。    **图6.1各排序在不同数据规模下的时间复杂度**  再单独考虑时间复杂度为O()的合并排序、快速排序，如下图，无论是在小规模的比较，还是大规模数据的比较，快速排序所消耗时间都比合并排序少。在数据规模为50000时，合并排序所需的平均时间为12ms左右，而快速排序平均所需时间为9.5ms左右。    **图6.2快速排序与合并排序在不同数据规模的耗时**   1. **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。**   首先第一想到的方法就是堆排序的升序排序，因为堆排序每次排序就是从当前剩下的数中挑选最大的那个数，放在数组的最后端，维护一个数组的最大堆，所以只需要10次堆排序的筛选，就可以从10亿个数中挑出10个最大的。  但是，我们知道，排序中最快的，也是我们平常用最多的，无外乎就是快排了，那这里能否尝试用快排解决这个问题？快排的思想及原理我们前面已介绍了，这里的10亿个数挑出10个最大的，也就是我们只需考虑对快排划分后的的子序列排后面的，前面的不包括最后10个数的序列，我们就不用排序了。原本快排是这样的：    现在我们考虑将其左半部分的序列是否需要排序进行判断下，如果前面的子序列面的不包括最后10个数，那么就不用对其排序：    对改进后的快排，我们用堆排产生的最大的10个数，来和同样的数据快排产生的最大10个数进行比较，结果如下： |
| 两种排序产生的结果是一样的，因此我们可利用这改进过的快排来解决我们的问题。现在我们来看下两种方法处理同样的数据规模所需要的时间，这里的时间单位为毫秒，结果如下：      由以上可知，在数据规模为10000到50000时，堆排需要3到20毫秒的时间，而堆排只需要不足1毫秒的时间。        当数据规模增大时，如上图，当数据规模为100000、200000、300000、400000时，堆排需要40到200多毫秒的时间，而快排只需不到10毫秒的时间；当数据为1000000时，堆排需要600多毫秒，而快排只需要20多毫秒。可以预见，当数据规模10亿时，快排的速度优越性将会更加明显。  由实验结果我们可以得出，用改进过的快排来解决这个问题，所需时间远少于用堆排来实现，可见，快排是解决该问题的比较优良的方案。 |
| **实验结果和小结：**  1 **掌握了**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 **掌握了**不同排序算法时间效率的经验分析方法，完成理论分析与经验分析以及理论与实际的对比。  3 **不同的**时间复杂度，排序时间的差别会很大，特别是当数据规模增大时，这种差异会越来越明显。  4 **相同的**时间复杂度，实际排序时间也会有所差异，和其他时间复杂度都为O()的排序算法相比，冒泡实际排序时间较长，插入排序时间较短；同样是时间复杂度为O(n)的时间复杂度，快速排序的实际耗时会短一些，而合并排序的时间会长一些。  5 **在实验**的过程，由于产生的数据范围太小，或者不够随机，也会导致一些违背常理的结果，刚开始我的随机数只是用了一个rand()函数，发现合并排序、快速排序的耗时都偏高了，且合并排序的耗时比快速排序的耗时还高，经仔细的查找资料发现，rand()的数据范围在某些系统上只有0-32767，这会导致当数组数据规模为40000甚至是50000时，产生的数据会有很多重复的，使得实验数据没有代表性。而我们知道，在数组有多个元素相同时，快速排序作为一种不稳定的排序，在处理这类数组时间上会偏高一些，从而有上述现象，以及快速排序的排序速度比合并还慢的情况。后来把随机生成函数稍作修改（具体请看源代码），各排序的耗时则相对正常多了。 |
| **指导教师批阅意见：**  **成绩评定：**  **指导教师签字：**  年 月 日 |
| 备注： |