**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： 实验1 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： 刘刚**

**报告人：黄志鹏 学号：2017303008 班级：计科1班**

**实验时间： 2020年4月 1日 - 2020年4月 7日**

**实验报告提交时间： 2020年4月 7日**

**教务部制**

**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。
3. **实验步骤与结果**
4. **问题描述**

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。

1. **算法原理描述**
   1. **冒泡排序**

其基本思路为，将相邻的两个元素进行比较，若排序方式不符合，则将其进行交换，

数组比较的顺序可以从前往后，或是从后往前，若是将数组竖直排列，就如气泡向上升，故而称为冒泡排序。

原理如下：

1. 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
2. 对每一对相邻元素做同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。
3. 针对所有的元素重复以上的步骤，除了最后一个。
4. 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。
   1. **选择排序**

选择排序是一种简单直观的[排序算法](https://baike.baidu.com/item/%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95/5399605" \t "https://baike.baidu.com/item/%E9%80%89%E6%8B%A9%E6%8E%92%E5%BA%8F/_blank)，选择排序是不稳定的排序方法。

原理如下：

1. 第一次从待排序的[数据元素](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0/715313" \t "https://baike.baidu.com/item/%E9%80%89%E6%8B%A9%E6%8E%92%E5%BA%8F/_blank)中选出最小（或最大）的一个元素，存放在序列的起始 位置。
2. 从剩余的未排序元素中寻找到最小（大）元素，放到已排序的序列的末尾。
3. 以此类推，直到全部待排序的数据元素的个数为零。
   1. **插入排序**

插入排序，一般也被称为直接插入排序。对于少量元素的排序，它是一个有效的算法 。插入排序是一种最简单的[排序](https://baike.baidu.com/item/%E6%8E%92%E5%BA%8F/1066239" \t "https://baike.baidu.com/item/%E6%8F%92%E5%85%A5%E6%8E%92%E5%BA%8F/_blank)方法，它的基本思想是将一个记录插入到已经排好序的有序表中，从而一个新的、记录数增1的有序表。

原理如下：

1. 假设前面n-1(其中n>=2)个数已经是排好顺序的。
2. 将第n个数插到前面已经排好的序列中，然后找到合适自己的位置，使得插入第n 个数的这个序列也是排好顺序的。
3. 按照此法对所有元素进行插入，直到整个序列排为有序的过程。
   1. **快速排序**

通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以[递归](https://baike.baidu.com/item/%E9%80%92%E5%BD%92/1740695" \t "https://baike.baidu.com/item/%E5%BF%AB%E9%80%9F%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95/_blank)进行，以此达到整个数据变成有序[序列](https://baike.baidu.com/item/%E5%BA%8F%E5%88%97/1302588" \t "https://baike.baidu.com/item/%E5%BF%AB%E9%80%9F%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95/_blank)。

原理如下:

1. 首先设定一个分界值，通过该分界值将数组分成左右两部分。
2. 将大于或等于分界值的数据集中到数组右边，小于分界值的数据集中到数组的左边。此时，左边部分中各元素都小于或等于分界值，而右边部分中各元素都大于或等于分界值。
3. 然后，左边和右边的数据可以独立排序。对于左侧的数组数据，又可以取一个分界值，将该部分数据分成左右两部分，同样在左边放置较小值，右边放置较大值。右侧的数组数据也可以做类似处理。
4. 重复上述过程，可以看出，这是一个递归定义。通过递归将左侧部分排好序后，再递归排好右侧部分的顺序。当左、右两个部分各数据排序完成后，整个数组的排序也就完成了。
   1. **归并排序**

归并排序是建立在归并操作上的一种有效的排序算法,该算法是采用分治法的一个非常典型的应用。将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。若将两个有序表合并成一个有序表，称为二路[归并](https://baike.baidu.com/item/%E5%BD%92%E5%B9%B6/253741" \t "https://baike.baidu.com/item/%E5%BD%92%E5%B9%B6%E6%8E%92%E5%BA%8F/_blank)。归并排序是一种稳定的排序方法。

原理如下：

1. 申请空间，使其大小为两个已经[排序](https://baike.baidu.com/item/%E6%8E%92%E5%BA%8F" \t "https://baike.baidu.com/item/%E5%BD%92%E5%B9%B6%E6%8E%92%E5%BA%8F/_blank)序列之和，该空间用来存放合并后的序列。
2. 设定两个[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88" \t "https://baike.baidu.com/item/%E5%BD%92%E5%B9%B6%E6%8E%92%E5%BA%8F/_blank)，最初位置分别为两个已经排序序列的起始位置。
3. 比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动 指针到 下一位置。
4. 重复步骤3直到某一指针超出序列尾。
5. 将另一序列剩下的所有元素直接复制到合并序列尾。
6. **算法实现的核心伪代码**
   1. **冒泡排序**

for i= 0 to n-1

for j=0 to n-1-i

if a[j]>a[j+1]

交换a[j],a[j+1]

* 1. **选择排序**

for i=0 to n-1

min=i

for j= i+1 to n-1

if ele[min]>ele[j] min=j

交换ele[i],ele[min]

* 1. **插入排序**

for i=1 to n

pos = i-1 //有序数列的最后一位

tmp=arr[i]

while(pos>=0 && arr[pos] > tmp){

arr[pos + 1] = arr[pos]

pos--

}

arr[pos + 1] = tmp //插入

* 1. **快速排序**

if(low >= high) 直接返回

    int i = low, j = high, tmp = arr[i]

    while (i < j) //将比tmp大的往后挪，小的往前挪

        while (i < j && arr[j] >= tmp)

            j--

        if (i < j)

            arr[i++] = arr[j]

        while (i < j && arr[i] < tmp)

            i++

        if (i < j)

            arr[j--] = arr[i]

    arr[i] = tmp;

    QuickSort(arr, low, i - 1); //递归调用

    QuickSort(arr, i + 1, high);

* 1. **归并排序**

MergeSort(arr,low,high) //分割为小问题

if(low+1<high)

m = (low + high) / 2;

MergeSort(arr, low, m);

MergeSort(arr, m, high);

Merge(arr, low, m, high);

Merge(arr,low,middle,high)

n1=middle-low

n2=high-middle

new array a1[n1],a2[n2]

for i=0 to n1-1

a1[i]=arr[p+i]

for j=0 to n2-1

a2[j]=arr[q+1+j]

i=j=0,k=low

for k=low to high

if (i < n1 && a1[i] <= a2[j])

arr[k] = a1[i++];

else if (j < n2 && a1[i] >= a2[j])

arr[k] = a2[j++];

else if (i == n1 && j < n2)

arr[k] = a2[j++];

else if (j == n2 && i < n1)

arr[k] = a1[i++];

1. **算法测试结果及效率分析**

分别以N=10000 20000 30000 40000 50000 为基准对各种算法进行测试。得到的运行时间表如下：

表4.1 各算法运行时间表(ms)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N\ms | 冒泡排序 | 选择排序 | 插入排序 | 快速排序 | 归并排序 |
| 10000 | 291.667 | 145.667 | 87.9 | 1 | 14.9 |
| 20000 | 1264.33 | 562.667 | 318.2 | 2.5 | 28.5 |
| 30000 | 3029 | 1307.67 | 698.4 | 3.7 | 43.2 |
| 40000 | 5203.67 | 2240 | 1228.1 | 4.9 | 66.5 |
| 50000 | 8109.67 | 3567.33 | 1936 | 6.1 | 79.7 |

由表4.1可做出效率曲线图如下：

图4.1 效率曲线图

分析：

从以上的数据可以得出，冒泡排序、选择排序、插入排序的运行效率是一个量级的，随着数据量的增加，所需时间成指数级增长，但由于冒泡排序的比较次数和交换次数相较于其余两种多，故其运行效率相较于低一些，而插入排序的平均交换次数比较次数比之选择排序低，故而效率更加高。

同时，对于解决数据量级较大的问题，归并排序和快速排序都是较为有效的方法，其中由于快速排序算法中，含有更少的数据赋值操作，而归并排序，进行合并操作时，需要额外一个数组作为中介，所以快速排序相较于归并排序会更快些。

根据表4.1可得到理论算出各个算法的理论时间表，如下：

表4.2 理论时间表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N\ms | 冒泡排序 | 选择排序 | 插入排序 | 快速排序 | 归并排序 |
| 10000 | 291.667 | 145.667 | 87.9 | 1 | 14.9 |
| 20000 | 1166.668 | 582.668 | 351.6 | 2 | 29.8 |
| 30000 | 2625.003 | 1311.003 | 791.1 | 4.755 | 70.848 |
| 40000 | 4666.672 | 2330.672 | 1406.4 | 8 | 119.2 |
| 50000 | 7291.675 | 3641.675 | 2197.5 | 11.610 | 172.984 |

根据表4.2可得各算法理论效率分析的曲线和实测的效率曲线图，如下所示：

图4.2 冒泡排序理论曲线与效率曲线图

图4.3 选择排序理论曲线与效率曲线图

图4.4 插入排序理论曲线与效率曲线图

分析：

冒泡排序、选择排序、插入排序都是O(n^2)的时间复杂度，由图4.2，4.3，4.4可知，该三种算法实测值与理论值的误差较小，其中冒泡排序和插入的误差随着数据规模的增大而增大，主要原因是除了比较操作之外，赋值操作也随着n的增加逐渐增大，从而导致误差增大。

图4.5 快速排序理论曲线与效率曲线图

图4.6 归并排序理论曲线与效率曲线图

分析：

由图4.5和4.6可知，快速排序和归并排序的效率实测值和理论值相差较大，经过分析，快速排序和归并排序在对某些特定序列的数组排序时，会有较快运行效率，如快速排序形成的二叉树接近于平衡二叉树。

综上可知，快速排序在处理数据规模巨大的问题由较大的优势，故选用快速算法。本次实验中，数据规模为十亿，如使用一般的数组声明方法，会导致栈空间溢出，使算法无法正常运行下去。故本次实验，当使用了N为一千万数据规模时，排序时间5762ms，根据快速排序的时间复杂度o（nlog2n）可推算出当N为十亿时所用时间约为38277ms。

1. **实验心得**

通过本次实验，加深了我对排序算法的认识，同时也学习了通过理论和实测分析算法的时间效率，验证了理论分析与经验分析的一致性。

同时，在进行算法设计中，若控制数据读写、修改的操作尽量减少，对于算法性能提升也具有一定效果，合理分析和利用算法能极大的提高效率。

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

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。