**深 圳 大 学 实 验 报 告**

**课程名称： 算法设计与分析**

**实验名称： 多种排序算法的算法实现及性能比较**

**学院： 计算机与软件学院 专业： 软件工程**

**报告人： 徐霞 学号： 2015150290 班级： 3**

**同组人： 无**

**指导教师： 杨烜**

**实验时间： 2017/9/1-2017/9/22**

**实验报告提交时间： 2017/9/22**

**教务处制**

1. **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。

1. **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；

4、 分别以n=100, n=1000, n=10000, n=100000，n=1000000，重复2的实验（横坐标取对数）；

5、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。

6、画出理论效率分析的曲线和实测的效率曲线，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。

**三．实验步骤与结果**

1. 选择排序：

伪代码：

Select\_sort(arr)

for i=0 to length(arr)

min=arr[i];

j=i+1;

for j to length(arr) //从待排序数组中选出最小的数

if min>arr[j]

min=arr[j];

index=j;

arr[index]=arr[i];

arr[i]=min;

运行结果：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10000 | 20000 | 30000 | 40000 | 50000 | 60000 | 70000 | 80000 |
| Time/ms | 37.65 | 153.45 | 345.7 | 619.9 | 955.85 | 1375.15 | 1870.3 | 2460.35 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | 100 | 1000 | 10000 | 100000 | 1000000 |
| Time/ms | 0 | 0.36 | 37.5 | 3644.85 | 379264 |

2. 冒泡排序：

伪代码：

Bubble\_sort(arr)

for i=0 to length(arr)-1

for j=0 to length(arr)-1-i

//将较大的数字往后移

if arr[j]>arr[j+1]

temp=arr[j]

arr[j]=arr[j+1]

arr[j+1]=temp;

运行结果：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10000 | 20000 | 30000 | 40000 | 50000 | 60000 | 70000 | 80000 |
| Time/ms | 139.6 | 640.15 | 1503.8 | 2719.25 | 4299.8 | 6230.8 | 8488.8 | 11177.5 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | 100 | 1000 | 10000 | 100000 | 1000000 |
| Time/ms | 0 | 0.8 | 138.65 | 17319 | 1836030 |

3. 合并排序：

伪代码：

Merge(arr, l, m, r)

//合并两个子数组

for i=0,j=0, k=l to r

if L[i]<=R[j]

arr[k]=L[i];

i++;

else

arr[k]=R[j];

j++;

Merge\_sort(arr, l, r)

if l<r

m=(l+r)/2;

Merge\_sort(arr,l,m); //递归将大数组拆分成小数组进行求解

Merge\_sort(arr,m+1,r);

Merge(arr,l,m,r);//将处理好的子数组合并

运行结果：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10000 | 20000 | 30000 | 40000 | 50000 | 60000 | 70000 | 80000 |
| Time/ms | 2.5 | 5.35 | 7.8 | 10.5 | 13.15 | 16.1 | 18.75 | 21.7 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | 100 | 1000 | 10000 | 100000 | 1000000 |
| Time/ms | 0 | 0.3 | 2.6 | 27.9 | 288 |

4. 快速排序：

伪代码：

Quick\_sort(arr , l, r)

if l<r

i=l,j=r,x=arr[l];

while i<j

while i<j &&arr[j]>=x

j--;

if i<j

arr[i++]=arr[j]

while i<j&&arr[i]<x

i++;

if i<j

arr[j--]=arr[i];

arr[i]=x;

//将数组拆分成子数组进行求解

Quick\_sort(arr,l,i-1);

Quick\_sort(arr,i+1,r);

运行结果：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10000 | 20000 | 30000 | 40000 | 50000 | 60000 | 70000 | 80000 |
| Time/ms | 0.6 | 1.25 | 1.85 | 2.8 | 3.55 | 4 | 4.75 | 5.65 |

5. 插入排序：

伪代码：

Insert\_sort(arr)

for j=1 to len

key=arr[j]

i=j-1;

while i=0 &&arr[i]>key

arr[i+1]=arr[i]

i--;

arr[i+1]==key;

运行结果：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10000 | 20000 | 30000 | 40000 | 50000 | 60000 | 70000 | 80000 |
| Time/ms | 15.3 | 60.3 | 138.6 | 245.5 | 388.8 | 548.9 | 745.9 | 969.3 |

**三．实验分析**

1. 选择排序：

理论效率分析：

选择排序：**每次从待排序数组中选择出最小的数据，顺序放在数组的前面。**

第一次选择：在n个数中找到最小的数字，比较n-1次

第二次选择：在n-1个数中找到最小的数字，比较n-2次

第三次选择：在n-2个数中找到最小的数字，比较n-3次

…….

第n-1次选择：在2个数中找到最小的数字，比较1次

**T(n)=1+2+3+…+n-1=n(n-1)/2 ∈O(n2)**

实际效率与理论效率对比：

从图中可以看出，曲线趋势基本上符合n2的趋势。

n=10000→20000：37.65\*4=150.6≈153.45（2.85）

n=10000→30000：37.65\*9=338.85≈345.7 （6.85）

n=10000→40000：37.65\*16=602.4≈619.9 （17.5）

……

理论与实际误差分析：

从曲线图中可以看出，曲线趋势基本符合O(n2),计算出来的理论效率略大于实际效率，此算法除了比较操作之外，还有交换操作（赋值操作），交换操作会随着n的增大而增大，最终体现在时间上。理论效率是在理想状态下计算出来的效率，实际上程序在运行过程中会­­受到电脑硬件的影响，所以实际效率要略低于理论效率

2. 冒泡排序：

理论效率分析：

冒泡排序：从头开始两两比较，将较大的数据往后移，一次之后就会将最大的数字移到最后一位，两次之后就会把第二大的数字排到倒数第二位，重复步骤直到排到第一位。

第1次：循环n-1次，排好第n位；

第2次：循环n-2次，排好第n-1位；

第3次：循环n-3次，排好第n-2位；

……

第n-1次：循环1次，排好第2位；

**T(n)=1+2+3+…+n-1=(n-1)(n-2)/2 – 1 ∈O(n2)**

实际效率与理论效率对比：

n=10000→20000：139.6\*4=558.4≈640.15

n=10000→30000：139.6\*9=1256.4≈1503.8

n=10000→40000：139.616=2233.6≈2719.25

……

理论与实际误差分析：

从曲线图中可以看出，冒泡排序曲线趋势基本符合O(n2),随着n的增大，我们发现，实际效率与理论效率的差值越来越大。这可能是因为冒泡排序程序中的交换数值的操作随着n的增大而增大，体现在时间上，加上电脑本身的状态的影响，使得实际效率要低于理论效率。

3. 合并排序：

理论效率分析：

合并排序：**将一个待排序数组分成两个子数组进行排序，两个子数组又可以再分成子数组进行排序，直到不可再分为止，最后将处理好的子数组合并，得到原问题的解。**

1. 通过递归将数组拆分成若干子数组
2. 对子数组进行排序并将子数组合并
3. 最后得到排好序的数组。

**T(n)= 1, n=1;**

**2T(n/2)+n,n>1;**

**令n=2k,k=logn;**

**T(n)=2(2T(n/4)+n/2)+n=4T(n/4)+2n=4(2T(n/8)+n/4)+2n**

**=8T(n/3)+3n=……=n+nlogn∈O(nlogn);**

实际效率与理论效率对比：

当n→cn时（c为任意正数），

[cnlog(cn)] /(nlogn)=[cn(logc + logn)] / (nlogn)

=c(logc +logn)/logn ≈c

所以计算理论效率时：

n=10000→20000：2.5\*2 =5.0≈5.35

n=10000→30000：2.5\*3 =7.5≈7.8

n=10000→40000：2.5\*4=10.0≈10.5

理论与实际误差分析：

合并排序效率是O(nlogn),从图中也可以看出,实际效率曲线也基本符合线性增长的。理论效率要高于理论效率，可能是因为在计算理论效率时忽略的log项，还有就是电脑对程序的影响等等。

4. 快速排序：

理论效率分析：

快速排序： **从数组中挑出一个元素，以此为基准重新排序数列，所有元素比基准小的摆放在基准前面，所有比基准大的摆放在基准的后面，递归把基准两边的子数组排序，最后的结果不需要合并，因为子数组都是原址排序的。**

快速排序的平均运行时间更接近于其最好情况，即尽可能的平衡的划分。

**T(n)= 1, n=1;**

**2T(n/2)+n,n>1;**

**令n=2k,k=logn;**

**T(n)=2(2T(n/4)+n/2)+n=4T(n/4)+2n=4(2T(n/8)+n/4)+2n**

**=8T(n/3)+3n=……=n+nlogn∈O(nlogn);**

实际效率与理论效率对比：

当n→cn时（c为任意正数），

[cnlog(cn)] /(nlogn)=[cn(logc + logn)] / (nlogn)

=c(logc +logn)/logn ≈c

所以计算理论效率时：

n=10000→20000：0.6 \*2=1.2≈1.25

n=10000→30000：0.6 \*3 =1.8≈1.85

n=10000→40000：0.6\*4=2.4≈2.8

……

理论与实际误差分析：

从图中可以看出，实际效率曲线基本符合线性增长，略低于理论效率，可能是因为快速排序使用了递归调用，而递归调用需要时间，还可能有电脑因素的影响，所以理论效率略高于实际效率。

5. 插入排序：

理论效率分析：

插入排序：每次向已经排好序的数组中插入一个数字。

实际效率与理论效率对比：

最坏情况分析：

当数组已反向排序时，有tj=j；所以

T(n)=c1\*n+c2\*(n-1)+c3\*(n-1)+c4\*(2+3+…+n)+c5\*(1+2+…+n-1)+c6\*(1+2+…+n-1)+c7\*(n-1)=(c4/2+c5/2+c6/2)n2+(c1+c2+c3+c7+c4/2-c5/2+c6/2)-(c2+c3+c4+c7)∈O(n2);

平均情况，tj=j/2，大约检查数组的一半，此时

T(n)=c1\*n+c2\*(n-1)+c3\*(n-1)+c4/2(2+3+…+n)+c5/2\*(1+2+…+n-1)+c6/2\*(1+2+…+n-1)+c7\*(n-1)=(c4/4+c5/4+c6/4)n2+(c1+c2+c3+c7+c4/4-c5/4+c6/4)-(c2+c3+c4/2+c7)∈O(n2);

从图中可以看出，曲线的趋势基本符合n2.

n=10000→20000：14.75 \*4=59≈60.3

n=10000→30000：14.75\*9=132.75≈133.95

n=10000→40000：14.75 \*16=236≈239.55

……

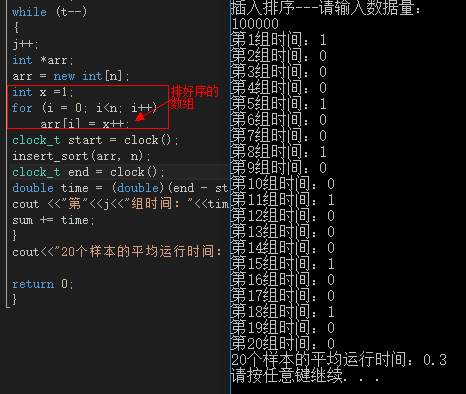
理论与实际误差分析：

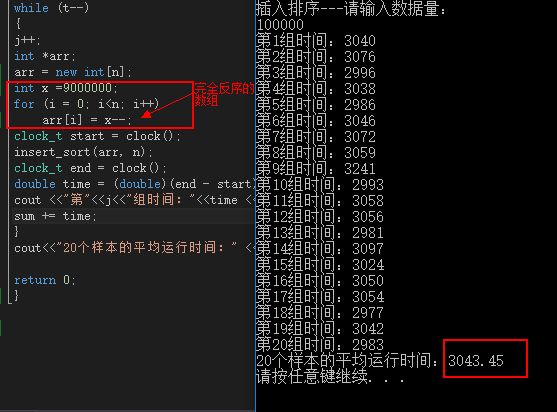
从图中可以看出，快排的实际效率曲线基本符合O(n2)的增长趋势，理论效率与实际效率相差不大，基本上吻合了，其中n=70000的时候实际效率与理论效率相差比较大，可能与当时电脑的状态相关。

**6. 五种排序算法效率对比分析：**

**在这次实验中应该考虑到，尽管是同一个算法，对于相同规模的n，不同的数据也会产生很大的差异，即最好情况与最差情况，举个例子：**

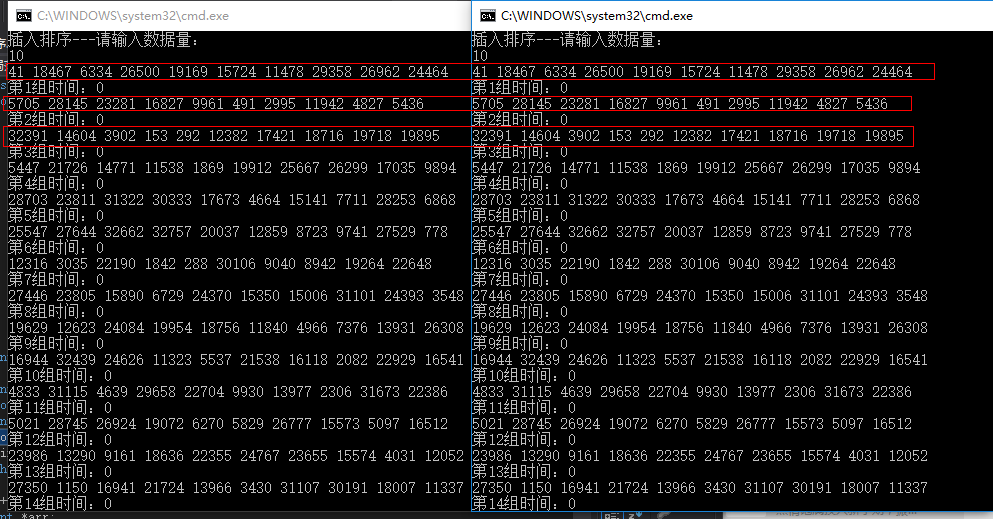
对于插入排序，当排一个已经排好序的数组与排一个反序的数组时：





从运行结果可以看出，**对于n=100000规模的数据，对于同一个排序算法，运行效率也有很大差异，所以在实验过程中我们必须避免这种情况的发生，使得对于不同算法，同一规模的数据使用的是同一套数据。**

通过查阅资料，我知道**rand()产生的随机数在每次运行的时候都是与上一次相同的。若要不同,用函数srand()初始化它**。在这里我们为了避免上面描述的这种情况的发生，应该不要使用srand().测试一下：



发现两次运行程序产生的随机数数据及顺序都是一样的。

冒泡：O(n2)、选择：O(n2)、插入：O(n2)算法的比较：

同样是O(n2)的效率，可是差别却很大，插入最快，冒泡最慢。首先我们要知道O(n2)只是一个上界而已，3 n2，100 n2，1000 n2，都是O(n2)，所以这三种排序算法虽然都是O(n2)效率，可是却有很大的差别。冒泡之所以耗时长，是因为他的比较次数达到了O(n2)，再加上交换次数的影响，随着n增大，收到交换次数影响越大。而选择比插入耗时多的原因在于选择排序每一趟从序列中选出最小的值时，都要与序列其他的全部元素进行比较，而插入排序只需要找到比当前值大的值就可以了。

合并：O(nlogn)，快排：O(nlogn) 算法的比较：

同样是O(nlogn)的效率，二者之间还是有差距。快排效率明显高于合并，二者都是使用的分治策略排序，但差别就在于：合并排序最后要进行数组合并操作，而快速排序不需要，因为子数组都是原址排序的。

五种排序算法整合在一张图中如下：

在同一张表中可以明显看到各种排序算法效率的差距。

**四．实验心得**

本次实验涉及的算法在之前的数据结构中就有接触，但未深入去分析其效率。实验过程中不得操作电脑，不然会给实验数据带来误差，导致在进行算法效率分析时出现让人不可解释的数据。通过这次的实验，我才深深体会一个好的算法带来的优越性，知道优化程序多么重要。

|  |
| --- |
| 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。