算法实验一：排序

姓名：易惠雄

学号：PB14011025

**1：实验要求**

实验1：排序 n 个元素，元素为随机生成的长为1..16的字符串（字符串均为英文小写字母），n的取值为：3^2，3^4，3^6，3^8，3^10，3^12；

算法：直接插入排序，堆排序，归并排序，快速排序，SHELL排序（5个增量，分别是：1，3，7，15，21）。

实验2：排序n个元素，元素为随机生成的1到65535之间的整数，n的取值为：3^3，3^5，3^7，3^9，3^11，3^13；

算法：冒泡排序，快速排序，归并排序，基数排序，计数排序。

**2：实验环境**

编译环境：dev-C++5.9.2

机器内存：8G

时钟主频：

系统：win7

**3：实验过程**

**实验1：**

**首先创造3^13个随机实验数据**，每个实验数据为由a到z的字符组成的长度在1到16的字符串组成，将数据写到F:\\PB14011025-project1\\ex1\\input\\input\_strings.txt的文本文件中(这部分由create-number程序完成)

**接着对于每个排序算法，相同的操作有：**

(1):从上面创造的文本文件F:\\PB14011025-project1\\ex1\\input\\input\_strings.txt.txt中读取指定个数的字符串存入字符串数组str1[n]中（n为字符串个数）

（2）每个排序执行前，读取时间LARGE\_INTEGER t1,t2,tc;

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

每个排序执行完，读取时间QueryPerformanceCounter(&t2);

那么排序所用时间为：t = (t2.QuadPart - t1.QuadPart)\*1.0/tc.QuadPart

（3）每个排序执行后，将排好序的数组存入相对应的文本文件中

(4)所有指定长度的排序都完成后，将每个算法所用的时间写入相对应的文本中。

**最后每个算法的思想如下（具体算法可见4中的关键代码截图）**

**直接插入排序**：二维字符串数组a[0][17]到a[i-1][17]已排序（i>=1）(已从小到大排列)，现在要对a[i][17]进行排序，先令str[17] = a[i][17],j=i-1比较str和a[j][17],若str>=a[j][17]则跳出循环；若str<a[j][17],则令a[j+1][17] = a[j][17], ,j --;

J<0跳出循环。最后令a[j+1][17] = i即可，这时二维字符串数组a[0][17]到a[i][17]已排序。

**堆排序**：调整堆，即将二叉树的每个根节点都大于等于它的子节点；在建堆过程中，令前【n/2】个为非叶子节点，后面的为叶节点，建立二叉树，后调用调整堆函数，此时数组中最大元素位于二叉树的根节点。排序即将建好后的二叉树的二叉树根节点元素和数组中最后一个元素交换，对A的前n-1个元素调用调整堆

函数，反复即排序完成。

**归并排序**：将数组A[1…n]平分为2部分A[1…n/2]和A[n/2+1…n],对它们分别调用归并排序，然后将排好的两部分连接在一起。连接函数Merge中，令数组L和R分别复制这两部分(最后添加一个侦察元素)，一一比较，写入A[1..n]中即可。

**快速排序**：对于划分元素的寻找为Partition函数。它实现的是对于一个数组A[p..r]找到一个中间元素q，使得q前面的元素都小于等于它，后面的元素大于等于它。

然后就对A[p..q] A[q+1…r]进行快速排序即可。

**SHELL排序**（5个增量，分别是：1，3，7，15，21）:将A[1...n]按下标进行分组，下标/gap相同的元素分为一组。对每组进行直接插入排序。要求gap最后为1.

这里令gap依次为21 15 7 3 1即可

**实验2：**

**创造数据过程和实验1类似，**不过将字符串改为整数即可；

**每个排序相同操作也和实验1类似**，不同点也在于将字符串改为整数；

**最后每个算法的思想如下（具体算法可见4中的关键代码截图）**：

**快速排序和归并排序与实验1相同**，

**冒泡排序**：先从第一个元素开始，两两比较，若前一元素大于后面的元素，则交换，一轮排序下来，最后的一个元素最大；此后与前相同，不过比较次数减一；最后一次比较后，即排序完成。

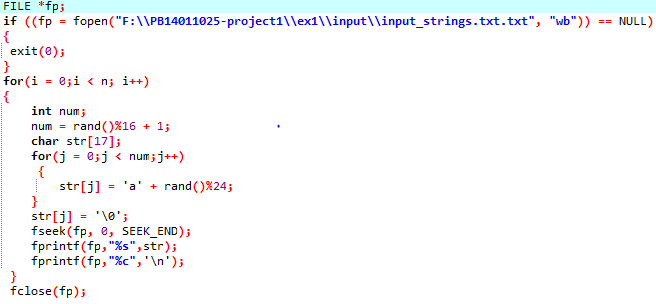
**计数排序**：对于数组A[1…n]，用C[k]记录A[1…n]中某一个值的个数，然后变换C[k],C[k] = C[k] + C[k-1],使得C[k]记录的是数组A中小于等于某一个值的个数，然后B[1..n]存储排序好数据的数组，令B[C[A[k]]]=A[k],C[A[k]]--,然后k从n到1即可，B即为排序好的数组；

**基数排序**：即将数分为5个位（此处int值最高为5位），现对低位进行排序（可使用计数排序），再对高位排序，即排序完成.

**4：实验过程代码关键截图（结合文字说明）**

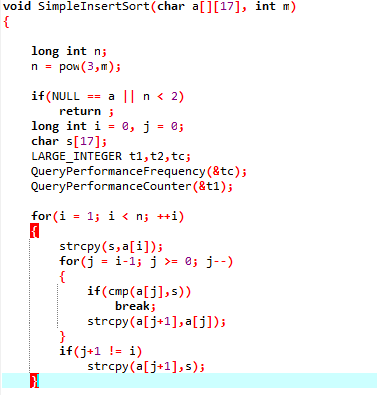
**实验1：**

**创造3^13个随机实验数据**：



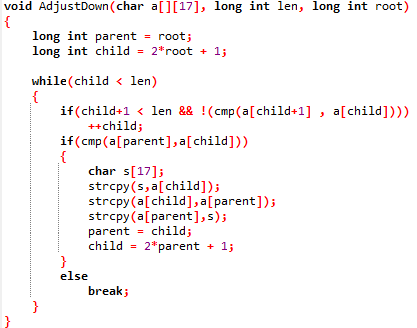
**文字说明**：通过随机函数产生a-z的字母和字符串长度，得到一个随机字符串；然后循环3^13次，即3^13个数据，写入指定文件中

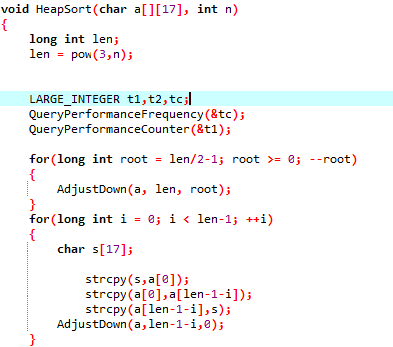
**直接插入排序：**



**文字说明**：函数中参数a为待排数组，当a[0…i-1]为已排时，让a[0…i]也变成已排

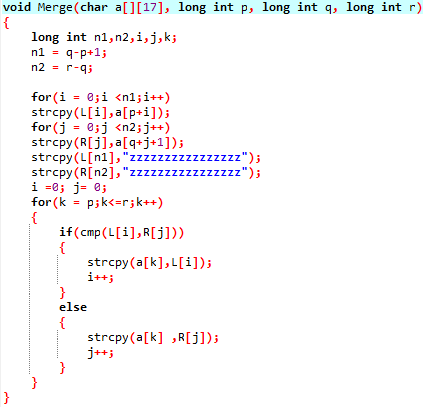
**堆排序**：

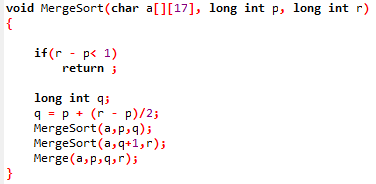




**文字说明:**Ajustdown函数可使堆变成最大堆，HeapSort函数先创造一个堆，然后区堆的根与最后一个元素替换，然后使前n-1个元素为最大堆，循环上述！

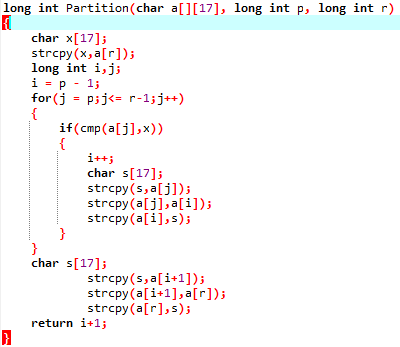
**归并排序**：

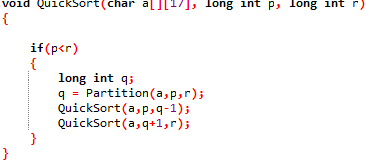




**文字说明**：MergeSort函数对数组a进行排序，先对前n/2个元素排序，后对第n/2+1到n元素排序，然后调用merge函数将两个已排序列连接。

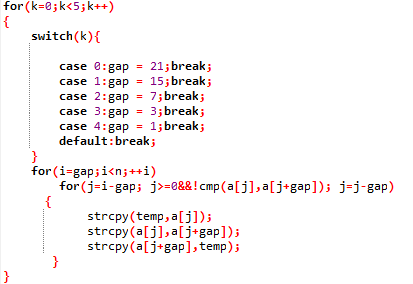
**快速排序**：





**文字说明**：将a[0…n-1]分为两部分，a[0…p]和a[p+1…n-1]，p的选择：p大于等于它前面的元素，小于等于它后面的元素，然后递归即可

**Shell排序**：



**文字说明**：gap为增量，每次对序号%gap 相同的序列进行直接插入排序，最后gap需为1

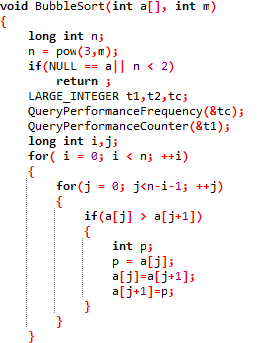
**实验2：**

**创造数据**：



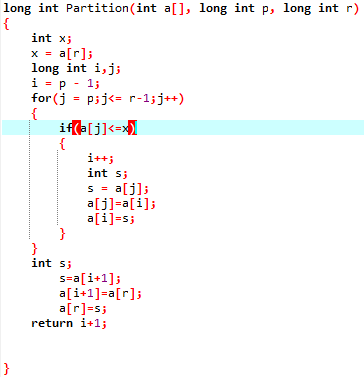
**文字说明**：同1

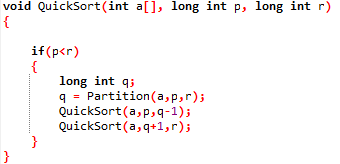
**冒泡排序**：



**文字说明**：1次循环可将比较过的元素中的最大值排到末尾

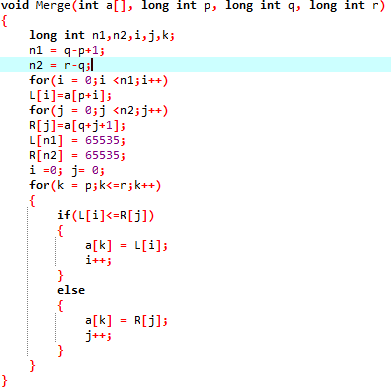
**快速排序**：

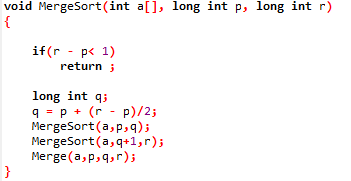




**文字说明**：同1

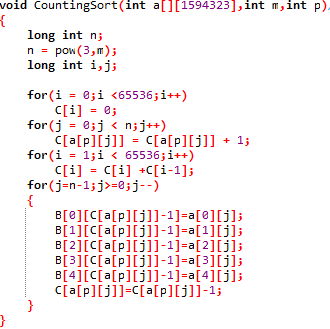
**归并排序**：

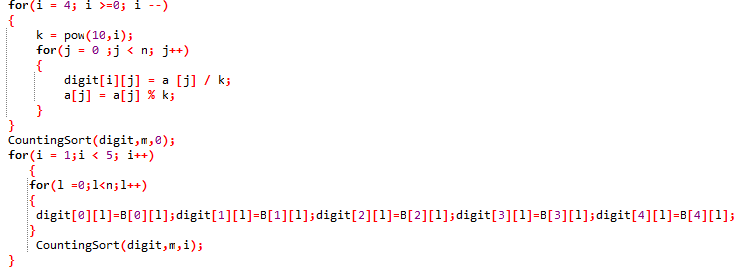




**文字说明**：同1

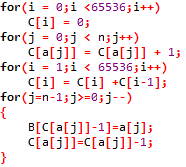
基数排序：





**文字说明**：先将数组A按每位改为二维数组digit[5][n]，然后依次对digit[0][n], digit[1][n], digit[2][n], digit[3][n], digit[4][n]进行计数排序

**计数排序**：



**文字说明**：C[1…k]开始保存的是数组A中等于k的元素个数，之后为数组A中小于等于k的元素个数，B[1…n]为已排好顺序的数组A。

**5：实验结果分析：**

**实验1**：

**直接插入排序**：

数据如下：

数据量n^2（10^8) 排序时间（s）

0.00000081 0.000005

0.00006561 0.000159

0.00531441 0.012602

0.43046721 0.513421

34.86784401 39.050001

排序时间随数据量n^2的变化曲线：

由R^2=1知时间t和n^2成线性关系

具体此处为t = 1.1197\*10^(-8) n^2 + 9.5\*10^(-11)

与课本中直接插入排序性能为O(n^2)相符合

**堆排序：**

数据如下：

数据量nlog(3)n(10^6) 时间（s）

0.000018 0.000007

0.000324 0.000121

0.004374 0.001348

0.052488 0.01445

0.590490 0.162303

6.377292 1.797565

排序时间随数据量nlog(3)n的变化曲线

由R^2 = 1,知，t与nlog(3)n成线性关系

t = 2.819 \*10(-7) nlog(3)n – 8 \* 10^(-10)

与课本中堆排序平均时间复杂度为O(n lgn)一致

**归并排序：**

数据如下：

数据量nlog(3)n(10^6) 时间（s）

0.000018 0.000012

0.000324 0.000105

0.004374 0.001268

0.052488 0.014584

0.590490 0.150408

6.377292 1.581230

排序时间随数据量nlog(3)n的变化曲线

由R ^2 =1知，t与 nlog(3)n成线性关系

t = 2.478 \*10(-7) nlog(3)n + 1.1 \* 10^(-9)

与课本中堆排序平均时间复杂度为O(n lgn)一致

**快速排序；**

数据如下：

数据量nlog(3)n(10^6) 时间（s）

0.000018 0.000006

0.000324 0.000092

0.004374 0.000918

0.052488 0.00896

0.590490 0.107854

6.377292 2.338617

排序时间随数据量nlog(3)n的变化曲线

由R ^2 =0.9979知，t与 nlog(3)n成线性关系

t = 3.69 \*10(-7) nlog(3)n – 2.26 \* 10^(-8)

与课本中堆排序平均时间复杂度为O(n lgn)一致

**Shell排序：**

数据如下：

数据量nlog(3)n(10^6) 时间（s）

0.000018 0.000007

0.000324 0.000096

0.004374 0.00194

0.052488 0.0768

0.590490 3.730708

6.377292 333.991910

排序时间随数据量nlog(3)n的变化曲线

由R ^2 =0.9937知，t与 nlog(3)n成线性关系

t = 5.2953 \*10(-5) nlog(3)n – 25.6989 \* 10^(-6)

与课本中堆排序平均时间复杂度为O(n lgn)一致

**对每个算法综合分析：**

Log(3)n 堆 插入 merge quick shell

2 0.000007 0.000005 0.000012 0.000006 0.000007

4 0.000121 0.000159 0.000105 0.000092 0.000096

6 0.001348 0.012602 0.001268 0.000918 0.00194

8 0.01445 0.513421 0.014584 0.00896 0.0768

10 0.162303 39.05 0.150408 0.107854 3.730708

12 1.797565 1.58123 2.338617 333.99191

系列1：堆 2：插入 3：归并 4：快排 5 ：shell排序

输入规模<=3的2次方时，5种算法相差不大，插入排序最优

输入规模在3的4次方到3的6次方之间时，shell排序最优，merge排序，堆排序，quick排序相差不大，处于中间水平，插入排序最慢！

输入规模大于3的6次方时，堆排序和merge排序，quick排序最优，相差不大，shell排序较差，插入排序最差！

与课本上介绍的算法渐进性能相符！

**实验2：**

**冒泡排序：**

数据如下：

数据量n^2（10^9) 排序时间（s）

0.000000729 0.000005

0.000059049 0.000253

0.004782969 0.018771

0.387420489 2.142307

31.381059609 195.777136

排序时间随数据量n^2的变化曲线：

由R^2=1知时间t和n^2成线性关系

具体此处为t = 6.2409\*10^(-9) n^2 -7.11\*10^(-11)

与课本中直接插入排序性能为O(n^2)相符合

**快速排序**：

数据如下：

数据量nlog(3)n(10^7) 时间（s）

0.0000081 0.000003

0.0001215 0.000043

0.0015309 0.000379

0.0177174 0.005626

0.1948617 0.049083

2.0726199 0.845431

排序时间随数据量nlog(3)n的变化曲线

由R ^2 =0.9987知，t与 nlog(3)n成线性关系

t = 4.097 \*10(-8) nlog(3)n – 6.1 \* 10^(-10)

与课本中堆排序平均时间复杂度为O(n lgn)一致

**归并排序**：

数据如下：

数据量nlog(3)n(10^7) 时间（s）

0.0000081 0.000010

0.0001215 0.000047

0.0015309 0.000680

0.0177174 0.005501

0.1948617 0.060360

2.0726199 0.664347

排序时间随数据量nlog(3)n的变化曲线

由R ^2 =1知，t与 nlog(3)n成线性关系

t = 3.206 \*10(-8) nlog(3)n – 4 \* 10^(-11)

与课本中堆排序平均时间复杂度为O(n lgn)一致

**计数排序**：  
数据如下：

n+k(k=65536)(10^8) 时间（s）

0.00065563 0.000731

0.00065779 0.000615

0.00067723 0.000883

0.00085219 0.001029

0.00242863 0.005216

0.01614959 0.049503

排序时间随n+k的变化曲线:

由R ^2 =0.9995知，t与 n+k成线性关系

t = 3.1594\*10(-8) (n+k) – 1.6\* 10^(-11)

与课本中堆排序平均时间复杂度为O(n +k)一致

**基数排序**：

数据如下：

d(n+k)(d = 5,k=65536)(10^8) 时间（s）

0.00327815 0.003140

0.00328895 0.005638

0.00338615 0.003677

0.00426095 0.008907

0.01214315 0.061149

0.08074795 0.548735

排序时间随d（n+k）的变化曲线:

由R ^2 =0.9999知，t与 d(n+k)成线性关系

t = 7.0445\*10(-8) (n+k) – 2.05\* 10^(-10)

与课本中堆排序平均时间复杂度为O(d(n +k))一致

**对每个算法综合分析：**

Log(3)n 冒泡 计数 归并 快排 基数

3 0.000005 0.000731 0.000010 0.000003 0.003140

5 0.000253 0.000615 0.000047 0.000043 0.005638

7 0.018771 0.000883 0.00068 0 0.000379 0.003677

9 2.142307 0.001029 0.005501 0.005626 0.008907

11 195.777136 0.005216 0.060360 0.049083 0.061149

13 0.049503 0.664347 0.845431 0.548735

1:冒泡 2：计数 3：归并 4 ：快排 5：基数排序

输入规模<=3的5次方时，快速排序，归并排序，冒泡排序都较快，其中快速排序最快；基数排序和计数排序都很慢，基数排序最慢！

输入规模在3的7次方到3的9次方之间时，计数排序，归并排序，快速排序均很快，基数排序较慢，冒泡排序很慢！

输入规模大于3的9次方时，计数排序最快，归并排序，快速排序，基数排序接近，都较快！冒泡排序极慢！

与课本上介绍的算法渐进性能相符！

**6：实验心得**

通过本次实验，让我完全熟悉了各种排序的算法，和其实现过程中要注意的问题；

通过本次实验，让我明白，当数据较多又知道数据范围时，用计数排序(O(n+k))效果极佳；

而归并排序，堆排序，快速排序等O(nlgn)算法在数据多或少时均不错！

对于直接插入排序，冒泡排序，当数据量很少时，效果很好！

通过本次实验数据的处理，让我形象的看见了各种算法的时间复杂度，意义重大！

通过完成此次实验，让我知道让算法变成可执行程序，还需很多调试过程，此次试验让写代码能力有了一定程度提高！

所以，本次实验收获颇多！