



# 南昌大学实验报告

学生姓名： 丁俊      学 号： 8003119100      专业班级： 信安 193 班

实验类型： ☒ 验证 ☐ 综合 ☐ 设计 ☐ 创新      实验日期： 6.3      实验成绩：         

## 一、实验项目名称

排序

## 二、实验目的

学习插入排序和二分、表排序以及排序的时间复杂度

## 三、实验任务

- 1、请设计直接插入排序算法函数 `void insertSort(int a[],int n)`，对  $a[1]\cdots a[n]$  进行升序排序。并测试在不同数据规模下的排序效率。
- 2、请设计二分插入排序算法函数 `void binInsertSort(int a[],int n)`，对  $a[1]\cdots a[n]$  进行升序排序。并测试在不同数据规模下的排序效率。
- 3、请设计 shell 排序算法函数 `void shellSort(int a[],int n)`，对  $a[1]\cdots a[n]$  进行升序排序。并测试在不同数据规模下的排序效率。

## 四、主要仪器设备及耗材

**Dec++5.15    windows10**

## 五、实验步骤

### 1、直接插入排序测试

```
1. #include "ArrayIo.h"
2. #include <time.h>
3. #define N 500000
4.
5. void insertSort(int a[], int n) {
6.     int i, j;
7.     for (i = 2; i <= n; i++) {
8.         a[0] = a[i]; // a[0]不存储数据，只做临时数据用来比较
9.         j = i - 1;
10.        // 让j往左移动
11.        while ( a[j] > a[0]) {
12.            a[j + 1] = a[j]; // 大于 a[0]的数不断向右移动，腾出空位
13.            j--;
14.        }
15.        // 因为上面最后面 j 减去了 1，所以要加回来
16.        a[j + 1] = a[0];
17.    }
18. }
19.
20. int main() {
21.     clock_t start, finish;
22.     start = clock();
23.     int a[N + 1], n;
24.     printf("数据初始化...\n");
25.     n = readData(a, N, "data1.txt");
26.     printf("%d 个数据排序中...\n", n);
27.     insertSort(a, n);
28.     saveData(a, n, "out.txt");
29.     printf("排序结束,排序结果保存在 out.txt 文件中\n");
30.     finish = clock();
31.     double d = (double)(finish - start) / CLOCKS_PER_SEC;
32.     printf("the time cost is %lf\n", d);
33.     return 0;
34. }
```

`double d = (double)(finish - start) / CLOCKS_PER_SEC;` 用来计算程序运行时间

## 2、二分插入排序

```
1. #include "Arrayio.h"
2. #define N 50000      /*N 为数据量大小，因 data1.txt 中只有 50 万个数，所以自行设定 N
   值时需让 N<=500000*/
3.
4. /*请将本函数补充完整，并进行测试*/
5. void binInsertSort(int a[], int n) {
6.     int mid, left, right, i, j;
7.     for (i = 2; i <= n; i++) {
8.         left = 1;
9.         right = i - 1;
10.        while (left <= right) {
11.            mid = (left + right) / 2;
12.            if (a[i] < a[mid])
13.                right = mid - 1;
14.            else
15.                left = mid + 1;
16.        }
17.        //插入的位置是 left
18.        a[0] = a[i];
19.        for (j = i - 1; j >= left; j--)
20.            a[j + 1] = a[j];
21.        a[left] = a[0];
22.    }
23.
24. }
25.
26. int main() {
27.     clock_t start, finish;
28.     start = clock();
29.     int a[N + 1], n;          /*数据存储在 a[1]...a[N]中*/
30.     printf("数据初始化...\n");
31.     n = readData(a, N, "data1.txt"); /*从 data1.txt 中读入 N 个整数存入数组 a, n
   为实际读入的数据个数*/
32.     printf("%d 个数据排序中...\n", n);
33.     binInsertSort(a, n);
34.     saveData(a, n, "out.txt");    /*排序结果存放在 out.txt 文件中*/
35.     printf("排序结束，排序结果保存在 out.txt 文件中.\n");
36.     finish = clock();
37.     double d = (double)(finish - start) / CLOCKS_PER_SEC;
38.     printf("the time cost is %lf\n", d);
```

```
39.     return 0;
40. }
```

### 3、shell 排序

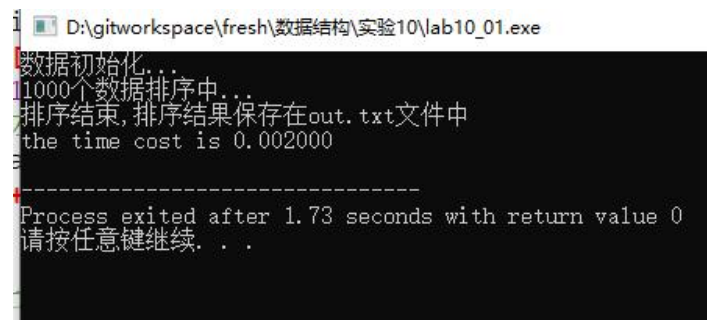
```
1. #include "Arrayio.h"
2. #include "time.h"
3. #define N 1000      /*N 为数据量大小, 因 data1.txt 中只有 50 万个数, 所以自行设定 N 值
   时需让 N<=500000*/
4.
5. /*请将本函数补充完整, 并进行测试*/
6. void shellSort(int a[], int n) {
7.     int i, j, d;
8.     d = n / 2;
9.     while (d >= 1) {
10.        for (i = d + 1; i <= n; i++) {
11.            //每一趟将 a[i]插入到 a[1]..a[i-1]
12.            a[0] = a[i];
13.            j = i - d;
14.            while ( j >= 1 && a[j] > a[0]) {
15.                a[j + d] = a[j];
16.                j = j - d;
17.            }
18.            a[j + d] = a[0];
19.        }
20.
21.        d = d / 2;
22.    }
23.
24. }
25.
26. int main() {
27.     clock_t start, finish;
28.     start = clock();
29.     int a[N + 1], n;          /*数据存储在 a[1]...a[N]中*/
30.     printf("数据初始化...\n");
31.     n = readData(a, N, "data1.txt"); /*从 data1.txt 中读入 N 个整数存入数组 a, n
   为实际读入的数据个数*/
32.     printf("%d 个数据排序中...\n", n);
33.     shellSort(a, n);
34.     saveData(a, n, "out.txt");    /*排序结果存放在 out.txt 文件中*/
35.     printf("排序结束, 排序结果保存在 out.txt 文件中.\n");
36.     finish = clock();
37.     double d = (double)(finish - start) / CLOCKS_PER_SEC;
38.     printf("the time cost is %lf\n", d);
```

```
39.     return 0;  
40. }
```

## 六、实验结果

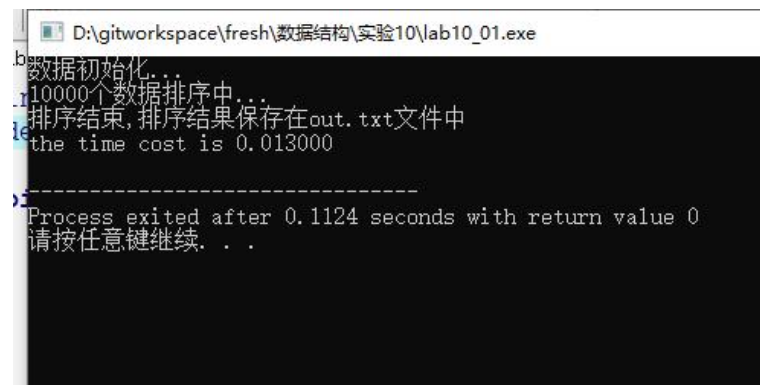
### 1、直接插入排序

当 N=1000 时：



```
D:\gitworkspace\fresh\数据结构\实验10\lab10_01.exe  
数据初始化...  
1000个数据排序中...  
排序结束, 排序结果保存在out.txt文件中  
the time cost is 0.002000  
-----  
Process exited after 1.73 seconds with return value 0  
请按任意键继续. . .
```

当 N=10000 时：



```
D:\gitworkspace\fresh\数据结构\实验10\lab10_01.exe  
数据初始化...  
10000个数据排序中...  
排序结束, 排序结果保存在out.txt文件中  
the time cost is 0.013000  
-----  
Process exited after 0.1124 seconds with return value 0  
请按任意键继续. . .
```

当 N=50000 时：



```
D:\gitworkspace\fresh\数据结构\实验10\lab10_01.exe  
数据初始化...  
50000个数据排序中...  
排序结束, 排序结果保存在out.txt文件中  
the time cost is 0.368000  
-----  
Process exited after 0.4606 seconds with return value 0  
请按任意键继续. . .
```

当 N=100000 时：

```
D:\gitworkspace\fresh\数据结构\实验10\lab10_01.exe
数据初始化...
1000000个数据排序中...
排序结束,排序结果保存在out.txt文件中
the time cost is 1.222000
-----
Process exited after 1.563 seconds with return value 0
请按任意键继续. . .
```

当 N=500000 时:

```
D:\gitworkspace\fresh\数据结构\实验10\lab10_01.exe
数据初始化...
500000个数据排序中...
排序结束,排序结果保存在out.txt文件中
the time cost is 8.749000
-----
Process exited after 8.827 seconds with return value 0
请按任意键继续. . .
```

从上面可知，有下表：

N 的取值	time 时间/s
1000	0.002
10000	0.013
50000	0.368
100000	1.222
500000	8.749

直接插入排序算法的时间复杂度是  $O(n^2)$ ，从上表可知超过几十万的数据规模该算法的效率就会变得很低，因为要花费更多的时间去比较和挪动位置。

## 2、二分插入排序

直接有下表格：

N 的取值	time 时间/s
1000	0.001
10000	0.014
50000	0.34
100000	1.157
500000	8.169

从上表可知二分排序应该会更快点，但是这里有读入数据和写入数据的干扰。

### 3、shell 排序算法

直接有下表格：

N 的取值	time 时间/s
1000	0.001
10000	0.004
50000	0.019
100000	0.044
500000	0.208

从上表可以看出 shell 排序比插入排序快得多，即使很高得数据量所花时间也非常短。

### 七、思考讨论题或体会或对改进实验的建议

### 八、参考资料