

# 重 庆 大 学

## 学 生 实 验 报 告

实验课程名称 数据结构与算法

开课实验室 DS1501

学 院 软件学院 年级 2021 专业班 软件 X 班

学 生 姓 名 XXX 学 号 2021XXXX

开 课 时 间 2022 至 2023 学年第 1 学期

总 成 绩	
教师签名	XXX

软件学院制

# 《数据结构与算法》实验报告

开课实验室：DS1501

2022 年 10 月 27 日

学院	软件学院	年级、专业、班	2021 级软件工 程 X 班	姓名	XXX	成绩	
课程 名称	数据结构与算法	实验项目 名 称	2022-2023 学年第一学期数 据结构与算法上机练习 003	指导教师	XXX		
教 师 评 语	教师签名：  年 月 日						

## 一、实验目的

- 请自行复习,并尝试编码和掌握教材第 7 章内排序的三种 N2 算法,具体实现插入排序(必选)、冒泡排序和选择排序(二者选一),以及实现希尔排序 Shellsort(必选)。
- 在实验报告中附上以上规定所选算法的关键代码,并具体分析相应算法的运行时间代价,按时提交实验报告。

## 二、使用仪器、材料

PC 微机;  
Windows 操作系统, VS2022 编译环境;

## 三、实验步骤

- 1、实现插入排序(必选)、冒泡排序和选择排序(二者选一),以及实现希尔排序 Shellsort(必选);
- 2、具体分析相应算法的运行时间代价,完成实验报告并提交。

#### 四、实验过程原始记录(数据、图表、计算等)

实现插入排序（必选）、冒泡排序和选择排序（二者选一），以及实现希尔排序 Shellsort（必选）；

##### （1）插入排序

①关键代码：

```
void Insertion_sort(int n, int list[]) {
    int times = 0;
    cout << "插入排序前数组为：" << endl;
    for (int i = 0; i < n; i++) {
        cout << list[i] << " ";
    }
    cout << endl;
    for (int i = 1; i < n; i++) {
        for (int j = i; j > 0; j--) {
            if (list[j] < list[j - 1])
            {
                swap(list[j], list[j - 1]);
                times++;
            }
        }
    }
    cout << "插入排序后数组为：" << endl;
    for (int i = 0; i < n; i++) {
        cout << list[i] << " ";
    }
    cout << endl;
    cout << "交换次数为：" << times << endl;
}
```

②验证程序：

```
int main() {
    int n;
    int *list;
    list = new int();
    cout << "请输入数组长度：" << endl;
    cin >> n;
    cout << "请输入数组元素：" << endl;
    for (int i = 0; i < n; i++) {
        cin >> list[i];
    }
    Insertion_sort(n, list);
}
```

③结果：



```
Microsoft Visual Studio 调试控制台
请输入数组长度：
7
请输入数组元素：
20 22 11 3 7 12 4
插入排序前数组为：
20 22 11 3 7 12 4
插入排序后数组为：
3 4 7 11 12 20 22
交换次数为：15
```

④时间复杂度分析：

最好的情况下，即数组本身就是有序的，则只需要进行  $n-1$  次比较，由于每次都是  $\text{arr}[i] \geq \text{arr}[i+1]$ ，无需移动，时间复杂度为  $O(n)$ 。

最坏的情况下，是排序表逆序，需要比较  $[(n+2)(n-1)/2]$  次，移动  $[(n+4)(n-1)/2]$  次。

如果排序记录是随机的，根据元素完全随机分布的原则，平均比较和移动的次数约为  $(n^2/4)$  次。

综上所述，插入排序算法的时间复杂度为  $O(n^2)$ 。

## (2) 冒泡排序

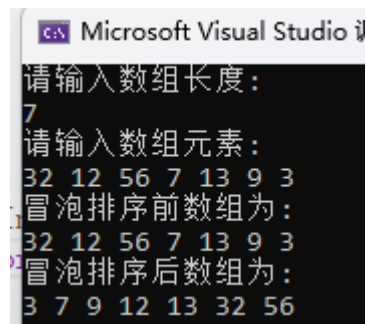
①关键代码：

```
void Bubble_sort(int *list, int n){
    for (int i = 0; i < n - 1; ++i)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (list[j]>list[j + 1])
            {
                swap(list[j], list[j + 1]);
            }
        }
    }
}
```

②验证程序：

```
int main()
{
    int n;
    int *list;
    list = new int();
    cout << "请输入数组长度：" << endl;
    cin >> n;
    cout << "请输入数组元素：" << endl;
    Input(list, n);
    cout << "冒泡排序前数组为：" << endl;
    Output(list, n);
    cout << endl;
    Bubble_sort(list, n);
    cout << "冒泡排序后数组为：" << endl;
    Output(list, n);
    return 0;
}
```

③结果：



```
Microsoft Visual Studio
请输入数组长度:
7
请输入数组元素:
32 12 56 7 13 9 3
冒泡排序前数组为:
32 12 56 7 13 9 3
冒泡排序后数组为:
3 7 9 12 13 32 56
```

④时间复杂度分析：

最好的情况下，初始状态有序度是  $n*(n-1)/2$ ，无需进行交换；

最坏的情况下，初始状态有序度是 0，需要进行  $n*(n-1)/2$  次交换；

平均情况下，根据元素完全随机分布的原则，可以认为需要  $n*(n-1)/4$  次交换操作，比较操作则比交换操作还要多。

综上所述，冒泡排序算法的时间复杂度是  $O(n^2)$ 。

### (3) 希尔排序

①关键代码：

```
void Shell_sort(int *list, int n)
{
    for (int gap = n / 2; gap > 0; gap /= 2)
    {
        for (int i = 0; i < gap; i++)
        {
            for (int j = i + gap; j < n; j += gap)
            {
                if (list[j] < list[j - gap])
                {
                    int k, temp;
                    k = j - gap;
                    temp = list[j];
                    while (k >= 0 && list[k] > temp)
                    {
                        list[k + gap] = list[k];
                        k -= gap;
                    }
                    list[k + gap] = temp;
                }
            }
        }
    }
}
```

②验证程序：

```
int main()
{
    int n;
    int *list;
    list = new int();
    cout << "请输入数组长度：" << endl;
    cin >> n;
    cout << "请输入数组元素：" << endl;
    Input(list, n);
    cout << "希尔排序前数组为：" << endl;
    Output(list, n);
    cout << endl;
    Shell_sort(list, n);
    cout << "希尔排序后数组为：" << endl;
    Output(list, n);
    return 0;
}
```

③结果：



```
Microsoft Visual Studio 调试控制台
请输入数组长度：
7
请输入数组元素：
17 11 5 4 6 2 3
希尔排序前数组为：
17 11 5 4 6 2 3
希尔排序后数组为：
2 3 4 5 6 11 17
```

④时间复杂度分析：

希尔排序的时间复杂度是与选中的增量有关的；

一般来说， $\{2^k, 2^{(k-1)}, \dots, 4, 2, 1\}$  这种增量序列并不是很好的增量序列，使用这个增量序列的时间复杂度(最坏情形)是  $O(n^2)$ ，Hibbard 提出了另一个增量序列  $\{1, 3, 7, \dots, 2^{k-1}\}$  (质数增量)，这种序列的时间复杂度(最坏情形)为  $O(n^{1.5})$ ，而 Sedgewick 也提出了几种增量序列，其中最好的一个序列是  $\{\dots, 109, 41, 19, 5, 1\}$ ，其最坏情形运行时间为  $O(n^{1.3})$ 。

综上所述，一般认为希尔排序的时间复杂度为  $O(n^{1.5})$  (取序列为  $\{\dots, 121, 40, 13, 4, 1\}$ )。

## 五、实验结果及分析

结果都已对应显示在原始数据记录中，结果都与预期的分析符合。