南开大学

Nankai University

南　开　大　学

网 络 空 间 安 全 学 院

数据结构实验报告

第一次作业

罗功成 1910487

年级：2019 级

专业：信息安全

指导教师：王玮

2021 年 9 月 25 日

摘要

基于 C++ 编程，不同排序规则下，根据是否借助辅助空间来确定不同算法的时间复杂度和空间复杂度

关键字：时间复杂度，空间复杂度，排序，动态内存分配

目录

一、 实验流程

(一) 实验内容和目的

已知一个长度为 n 的数组，数值范围为 10-99，对该数列进行排列.

排列规则:

1. 个位数大小作为第一排序标准
2. 个位数大小相同时，以十位数的大小作为第二排序标准。

要求:

1. 按照上述规则下，从小到大排列。
2. 给出算法设计思想，并用 C++ 实现。
3. 需要分别用依赖和不依赖辅助空间两种方法，来实现对数组的排列
4. 分别对两种方法的时间复杂度和空间复杂度进行分析。

(二) 实验原理和思路

1. 辅助空间可以理解为除了存放初始数据的数组外，额外需要的数组。
2. n 不确定，需要用动态内存空间分配的方式，用数组 a 存放初始数据。
3. 从小到大排序，考虑使用冒泡排序
4. 由于新的排序算法先比个位，个位相同比十位，个位比十位优先，也就相当于说是原来数据的个位在排序中应该算是“十位”，而原来的十位相当于排序中“个位”。
5. 因此，需要将原来的数据进行十位和个位的分离，然后给个位乘 10 再加上十位构成新的数字序列，在这个数列用冒泡排序，排好后再分离它的十位和个位，将它再对应还原成原来的数值。
6. 用辅助空间，可以考虑用 b,c,d 三个和存放原数据 a 等长且可变长度 n 的数组，分别存放十位，个位，还有反转长度后的数组
7. 不用辅助空间，也可以将个位，十位，反转三个属性用临时变量 temp 和数组 a 存放转换

(三) 实验核心代码

有辅助数组下排序算法

```
for (int i = 0; i < n; i++)//借助辅助空间
  {
      c[i] = a[i] / 10;//10 位,取10位要除以, 不是%模！！
      b[i] = a[i] - c[i] * 10;//个位
  }
  //规定 相当于是个位十位反转, 那么也可以考虑反转后排序, 在对应反转回来
  // 后面开始B当十位, C当个位。
```

```
    for (int i = 0; i < n; i++)
    {
        d[i] = b[i] * 10 + c[i];//反转数组
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = n - 1; j > i; j--)
        {
            if (d[j - 1] > d[j])
            {
                int temp1 = d[j];
                d[j] = d[j - 1];
                d[j - 1] = temp1;
            }
        }
    }
    for (int i = 0; i < n; i++)
    {
        int p = d[i] / 10;
        a[i] = p + (d[i] - p * 10) * 10;
    }
```

无辅助数组下排序算法

```
   for (int i = 0; i < n; i++)
    {
        int temp1 = a[i] / 10;//1号是十位数,变为新的个位数
        int temp2 = a[i] - 10 * temp1;//2号是个位数, 变成新的十位数
        a[i] = temp2 * 10 + temp1;
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = n - 1; j > i; j--)
        {
            if (a[j - 1] > a[j])
            {
                int temp = a[j];
                a[j] = a[j - 1];
                a[j-1] = temp;
            }
        }
    }

    for (int i = 0; i < n; i++)
    {
        int temp3 = a[i] / 10;//现在的十位其实是原来的个位
```

```
24          int temp4 = a[i] - 10 * temp3;//个位变回原来十位
25          a[i] = temp4 * 10 + temp3;
```

（四） 实验结果与分析

1. 依赖辅助空间的结果如图2所示

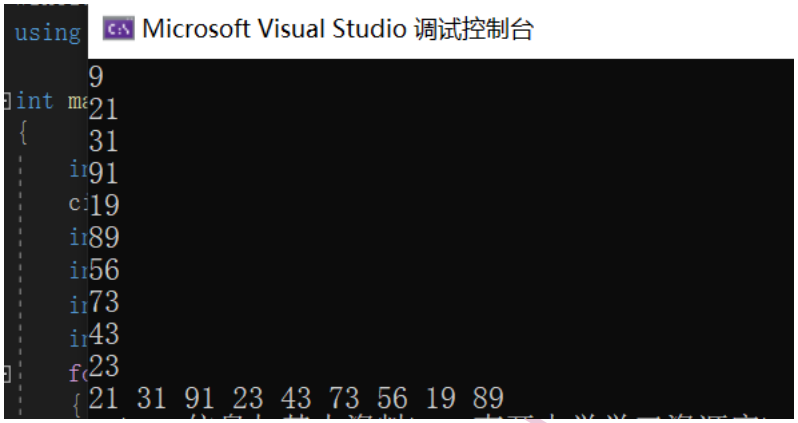

图 1: 借助 bcd 三个辅助数组的结果

2. 不依赖辅助空间的结果如图2所示

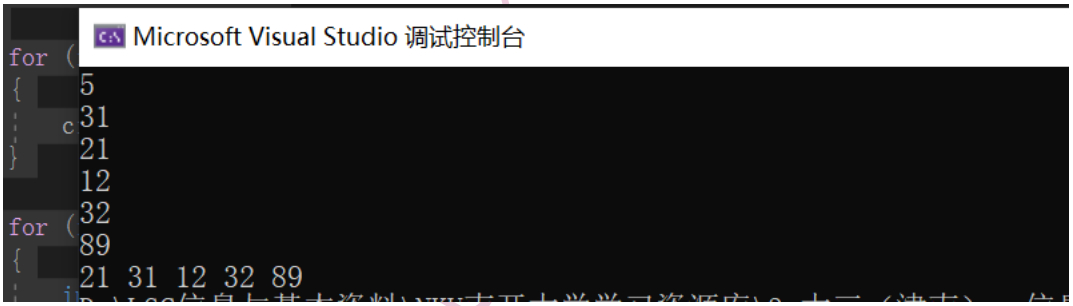

图 2: 只用临时变量 temp 的结果

可见，上述两种算法的核心代码均可以满足本次实验的各项要求。

二、 实验探究问题解析

1. 借助辅助空间排序：

时间复杂度上：O（n^2）

进行了 2n 次加法，2n 次减法，4n 次乘法，2n 次除法，n(n-1)/2 比较，交换可能 0 到 n-1 次.

空间复杂度上：O(n)

用到了 abcd 四个可变等长度 n 的数组，

Sp=n*(addr(a)+addr(b)+addr(c)+addr(d)+4*sizeof(int))

2. 不借助辅助空间排序：

时间复杂度上:O（n^2）
进行了 2n 次加法，2n 次减法，4n 次乘法，2n 次除法，n(n-1)/2 比较，交换可能 0 到 n-1 次.
空间复杂度上：O(n)
只用到了 a 一个可变等长度 n 的数组，存放初始数值
Sp=n*(addr(a)+sizeof(int))
两者的时间复杂度 tp 均为对应各核心操作次数与该操作一次所用时间乘积之和

三、 总结和收获

1. 对于冒泡排序，动态数组的代码实现进行了编程方面的复习。
2. 通过是否依赖辅助空间的两种算法的对比，对课程学习中的时间复杂度和空间复杂度有了更为直观的认识。
3. 对课程学习中的时间复杂度和空间复杂度、实例特征、可变部分大小、关键操作的确认、渐进记法等和本实验相关的课程知识进行了复习和巩固。