

# 实验一实验报告

2112514 辛浩然

## 实验题目

对数列的奇偶部分分别排序

## 实验内容

对数列使用快速排序分别对奇数和偶数进行非降序排序，在不使用新数组的前提下，奇数全部置于数组头部，偶数全部置于数组尾部。

## 实验代码

```
#include <iostream>
using namespace std;
void quickSort(int* a, int i, int j) //快速排序
{
    if (j <= i || j <= 0)
        return;
    int m = i, n = j, s = a[m];
    while (m < n)
    {
        while (m < n && a[n] >= s)
        {
            n--;
        }
        //从右向左找到比标准数小的数
        if (m < n)
        {
            a[m++] = a[n];
        }
        while (m < n && a[m] < s)
        {
            m++;
        }
        //从左向右找到比标准数大的数
        if (m < n)
```

```

        {
            a[n--] = a[m];
        }
    }
    a[m] = s;
    //经过这一轮的排序, m左侧都比它小, 右侧都比它大
    quickSort(a, i, m - 1);
    quickSort(a, m + 1, j);
    //再从m左右两侧分别进行排序
}

void Classify(int* a, int n)
{
    if (n <= 1)
        return;
    int i = 0, j = n - 1;
    while (i < j)
    {
        while (a[j] % 2 == 0 && i < j)
            j--;
        while (a[i] % 2 && i < j)
            i++;
        if (i != j)
            swap(a[i], a[j]);
        i++;
        j--;
    }
}

int main()
{
    int n;
    cin >> n;
    int* a = new int[n];
    for (int i = 0; i < n; i++)
        cin >> a[i];
    for (int i = 0; i < n; i++)
    {
        if (a[i] < 0)
        {
            cout << "ERROR";
            return 0;
        }
    }
    Classify(a, n); //奇偶分离

```

```

    int f = 0;
    while (a[f] % 2 && f < n)
        f++;
    //最后一个奇数的位置为f-1
    quickSort(a, 0, f - 1);
    quickSort(a, f, n - 1);
    //奇偶数分别排序
    for (int i = 0; i < n; i++)
        cout << a[i] << ' ';
    delete[] a;
    return 0;
}

```

## 实验测试

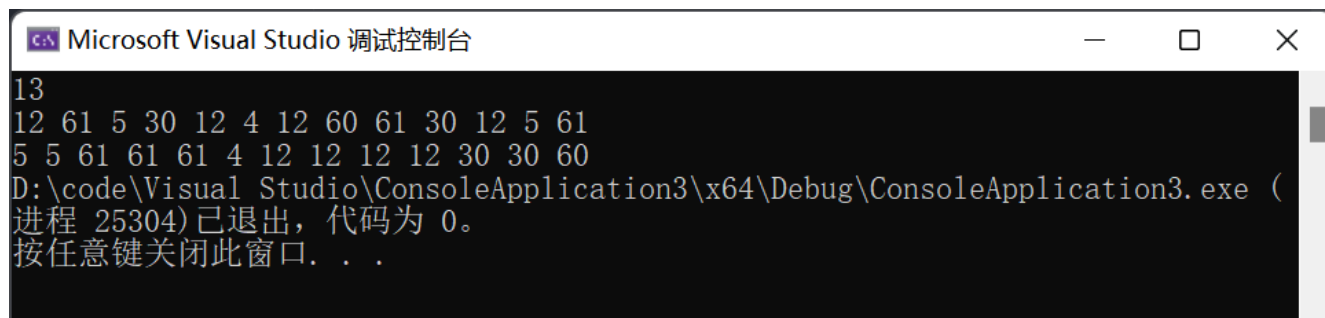
### 测试用例一

输入：

13

12 61 5 30 12 4 12 60 61 30 12 5 61

输出：



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio 调试控制台". The console output is as follows:

```

13
12 61 5 30 12 4 12 60 61 30 12 5 61
5 5 61 61 61 4 12 12 12 12 30 30 60
D:\code\Visual Studio\ConsoleApplication3\x64\Debug\ConsoleApplication3.exe (
进程 25304)已退出，代码为 0。
按任意键关闭此窗口. . .

```

### 测试用例二

输入：

10

10 9 8 7 6 5 4 3 2 1

输出：

```
Microsoft Visual Studio 调试控制台
10 9 8 7 6 5 4 3 2 1
1 3 5 7 9 2 4 6 8 10
D:\code\Visual Studio\ConsoleApplication3\x64\Debug\ConsoleApplication3.exe (
进程 16512)已退出，代码为 0。
按任意键关闭此窗口. . .
```

## 测试用例三

输入：

12

9 7 5 3 1 7 11 13 15 101 9 9

输出：

```
Microsoft Visual Studio 调试控制台
9 7 5 3 1 7 11 13 15 101 9 9
1 3 5 7 7 9 9 9 11 13 15 101
D:\code\Visual Studio\ConsoleApplication3\x64\Debug\ConsoleApplication3.exe (
进程 24120)已退出，代码为 0。
按任意键关闭此窗口. . .
```

## 算法分析

### 分离奇偶数

一边从数组头部开始向后遍历，当遇到偶数时停止；同时一边从数组尾部开始向前遍历，当遇到奇数时停止。此时交换两数。然后两边继续遍历，直到头尾相遇。如此，将奇数集中于数组的头部，偶数集中于数组的尾部。

### 快速排序算法

快速排序算法是一种分治的算法。设定一个标准数，将数组分成比标准数大和比标准数小的两部分，当每部分都有序，那么整个数组就是有序的。

具体实现：对于一个数组，将其首个元素定为比较标准数。从尾部向前遍历，找到比标准数小的数，把这个数放在标准数的位置；从头部向后遍历，找到比标准数大的数，把这个数放在上个数的位置；以此类推，直到头部头与尾部头相等，将标准数放在最终位置。这样，标准数左侧都比它小，标准数右侧都比它大。然后再对标准数左右两侧分别递归调用本函数。

### 复杂度分析

## 时间复杂度

1.奇偶分离函数：由程序分析可知，该函数对于该数组相当于仅仅进行了一次遍历，时间复杂度为 $O(n)$

2.快速排序函数：

最差时间复杂度：

每次选的基准值都是最大或最小的数，每次只多了一个有序数，相当于冒泡排序。

与标准数的比较次数为 $n-1+n-2+\dots+1$ 次，即 $n(n-1)/2$ 次，最差时间复杂度为 $O(n^2)$

最好时间复杂度：

每次的标准数刚好平分数组。

规模为 $n$ 的时间复杂度：

$T[n] = 2T[n/2] + n-1$  ( $n-1$ 为第一次平分数组的比较次数,即平分长度为 $n$ 的数组的比较次数)

迭代得第 $m$ 次递归： $T[n]=2^{(m-1)}T[n/2^{(m-1)}]+mn-2^m+1$

令 $n/2^{(m-1)}=1$ 得到 $m=\log n+1$

由于 $T[1]=0$ ,

故 $T[n]=nT[1]+(\log n+1)n-2n+1=n\log n-2n+1$

最好时间复杂度为 $O(n\log n)$

平均时间复杂度为 $O(n\log n)$

3.综上，整个程序时间复杂度为 $O(n\log n)$

## 空间复杂度

1.奇偶分离函数：空间复杂度为 $O(1)$

2.快速排序函数：

最差空间复杂度：

每次选的基准值都是最大或最小的数，每次只多了一个有序数，相当于冒泡排序

递归深度为 $n-1$ ,空间复杂度为 $O(n)$

最好空间复杂度：

由上分析，递归深度为 $\log n + 1$ ，空间复杂度为 $O(\log n)$

平均空间复杂度为 $O(\log n)$

3. 综上，整个程序空间复杂度为 $O(\log n)$

## 心得总结

1. 学习掌握快速排序方法，在实验应用中出现了一些bug。出现的原因在于对一些细节的处理不到位，比如在获取数组奇偶数分界点时导致数组访问越界。由于对快速排序的掌握并不是很好，出现了一些问题，在debug中对快速排序理解也更加深入；

2. 通过实验代码的复杂度分析对时间和空间复杂度的理解更加深入。对于快速排序时间复杂度的分析遇到一些困难，通过查找资料学习到分析方法；

3. 在代码编写和复杂度分析之后，意识到今后代码的编写，要注意程序的复杂度，尽可能地降低运行时间和运行内存。很多时候会面临二者的取舍，根据实际需要选择优化时间或空间复杂度。