# 

## 《算法分析理论及应用》课程实验报告

**班级：软工182 姓名：邓棋 学号：2018081062**

**一、实验题目**

1. 编码实现选择排序。
2. 编码实现冒泡排序。
3. 编码实现鸡尾酒排序。

**二、实验内容**

**1. 选择排序。（完成实验代码、伪代码）**

选择排序开始的时候，扫描整个序列，找到整个序列的最小记录和序列中的第一个记录交换，从而将最小记录放到它在有序区的最终位置上，然后再从第二个记录开始扫描序列，找到n-1个序列中的最小记录，再和第二个记录交换位置。一般地，第i趟排序从第i个记录开始扫描序列，在n-i+1（1≤i≤n-1）个记录中找到关键码最小的记录，并和第i个记录交换作为有序序列的第i个记录。请完成选择排序，并且编码实现。

**2. 冒泡排序。（完成实验代码、伪代码）**

冒泡排序的基本思想就是：从无序序列头部开始，进行两两比较，根据大小交换位置，直到最后将最大（小）的数据元素交换到了无序队列的队尾，从而成为有序序列的一部分；下一次继续这个过程，直到所有数据元素都排好序。算法的核心在于每次通过两两比较交换位置，选出剩余无序序列里最大（小）的数据元素放到队尾。

**3. 进阶提高题。（选做）小组讨论、编程完成鸡尾酒排序法。**

鸡尾酒排序法，又名双向冒泡排序法，算法传统冒泡法的一点改进。但是对于鸡尾酒排序，算法的时间复杂度与空间复杂度并没有改进。

　　不同的是排序的交换次数。某些情况下鸡尾酒排序比普通冒泡排序的交换次数少。比如{2,3,4,1},鸡尾酒排序只需交换2次，而冒泡排序需要三次。总体上，鸡尾酒排序可以获得比冒泡排序稍好的性能。但是完全逆序时，鸡尾酒排序与冒泡排序的效率都非常差。

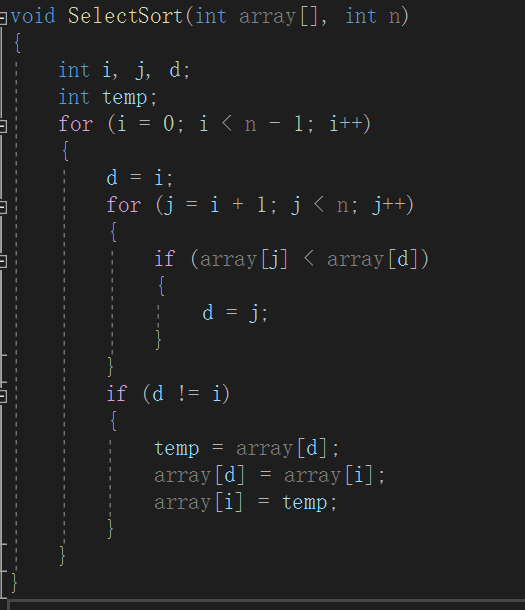
　　鸡尾酒排序的思想就是在从前往后依次循环依靠邻近数据交换实现结果的同时，依次从后往前循环数据交换。前者交换获取未排序最大值，而后者交换获取未排序最小值。

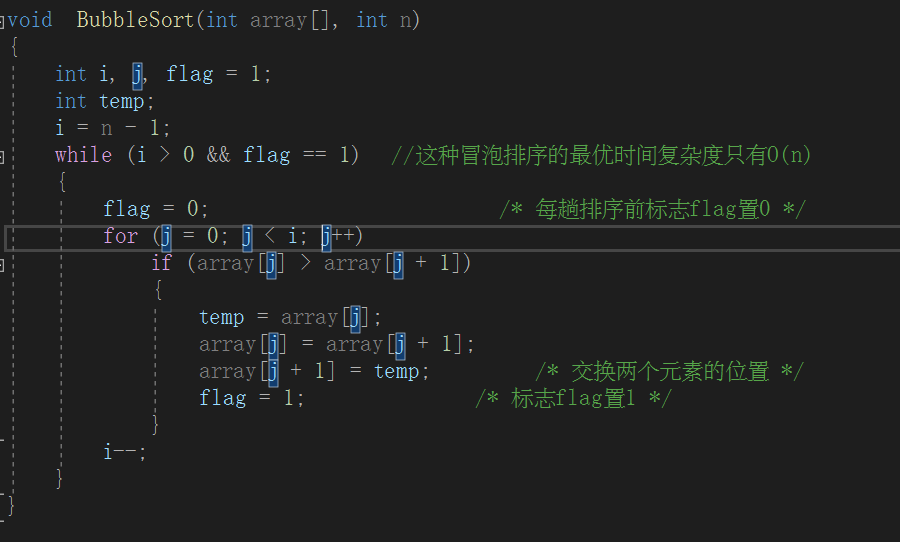
**三、实验目的**

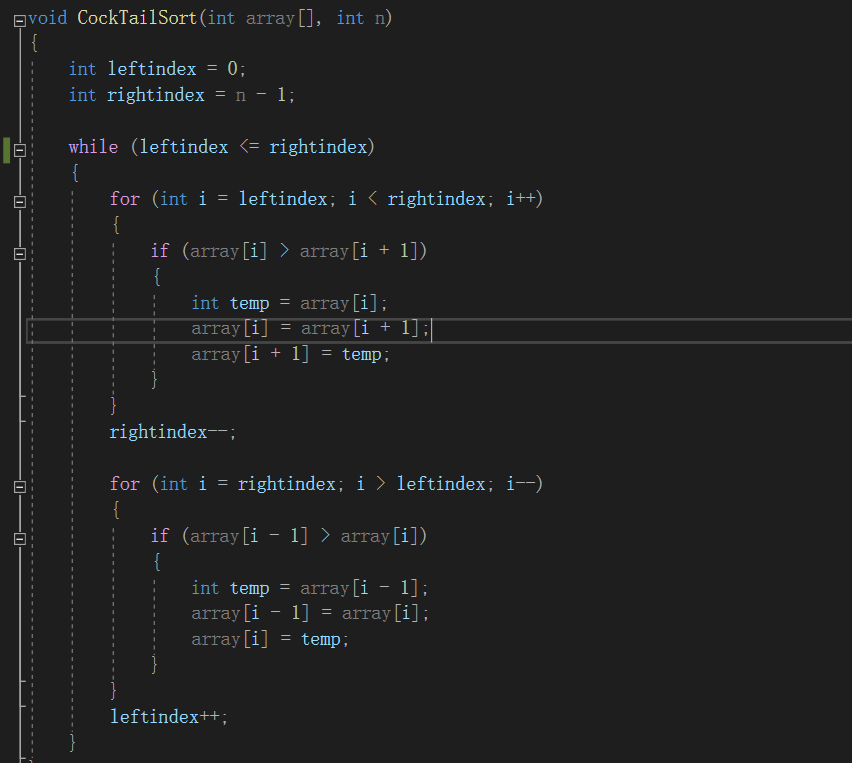
1. 理解蛮力法的思想，算法策略。

2. 重点掌握排序问题中的蛮力法

**四、实验代码**

**1、选择排序**

**2、冒泡排序**

**3、鸡尾酒排序**

**五、实验总结**

书写以下实验总结：1、算法伪代码编写；2、算法设计策略描述；3、算法时空复杂度分析。4、遇到的问题及解决方法。

1. **算法伪代码编写**

**选择排序：**

for i=0 to n-1

{

for j=i+1 to n

{

if(array[j]>array[i]) swap(array[j],array[i]) ;

}

}

**冒泡排序：**

i=n-1;

while(i>0 && flag!=0)

{

for j=0 to i

{

if(array[j]>array[j+1]) swap(array[j], array[j+1]);

flag = 1;

}

}

**鸡尾酒排序：**

while(leftindex <= rightindex)

{

for i=leftindex to rightindex

{

if(array[i]>array[i+1]) swap(array[i],array[i+1]);

}

rightindex--;

for i=rightindex to leftindex

{

if(array[i-1]>array[i]) swap(array[i-1], array[i]);

}

leftindex++;

}

1. **算法设计策略描述**

**选择排序：**用一个下标来标记每次循环一开始遍历的元素，然后依次往后遍历，只要元素比一开始标记的小，就将标记改为当前下标，然后交换位置，将每次能遍历的最小的数放置在每次遍历的最前端，这样不断选择，使无序的数列变为有序。

**冒泡排序：**使用双层循环，外层的第一次循环，内层循环使n个数据都两两比较了一次，并将此数列中的数交换位置，使得最大的数不断后移，使它变成最后一个数；外层的第二次循环，内层循环就只需要考虑n-1个数了，因为此时数列最末尾的数已经是最大的数了，不需要再纳入考虑范围，再将第二大的数挪到倒数第二个位置……外层循环每循环完一次就会浮出一个最大数，依次往后排，同样的，小数也不断浮到数列前面，循环完毕后，数列也就有了顺序。

**鸡尾酒排序：**从前往后依次循环依靠邻近数据交换实现结果的同时，依次从后往前循环数据交换，前者交换获取未排序最大值，而后者交换获取未排序最小值，将冒泡排序的过程双向实现。

1. **算法时空复杂度分析**

**选择排序：**最优情况为数列一开始就是有序的（比如说从小到大），最差情况为数列一开始逆序，无论最优还是最差，都得进行遍历，其平均时间复杂度为O(n^2)。其空间的开销主要在于用于交换位置的临时变量，最优情况下临时变量不会占用内存，最差情况下要生成[n(n-1)]/2个临时变量，但它的生命周期短，相当于每次需要交换的时候重复占用哎内存，所以其平均空间复杂度为O(1)。

**冒泡排序:** 冒泡排序的时间复杂度主要依赖于内外循环的时间开销,最优的情况就是开始已经排好序了，后续循环不需要进行交换，只有外层循环纳入时间开销，则时间开销为：[n(n-1)]/2，最优情况的时间复杂度为O(n^2)，这是对于一般的两层for循环，像是在此题中，用一个标志来判断是否继续循环，最优时间复杂度为O(n);最差的情况就是开始时元素全是逆序的，每一次排序都要交换两个元素，时间花销为：[3n(n-1)]/2，最差情况的时间复杂度为O(n^2)，冒泡排序的平均时间复杂度为O(n^2)。冒泡排序的空间复杂度全在于交换元素时临时变量（temp）作占的内存空间，同样的最优情况仍是不需要交换位置，临时变量不会分配内存，最优情况空间复杂度为0；最差的情况仍是元素的初态为逆序的，每次内层循环都会调用if语句块，一共有[n(n-1)]/2个临时变量生成，但是临时变量的生命周期很短，只在if语句块内，语句完毕后给它分配的内存会被释放，这样来看，n个临时变量都是反复地占用了同一块内存，此时的空间复杂度为O(1)。综上，冒泡排序的平均空间复杂度为O(1)。

**鸡尾酒排序：**鸡尾酒排序的时空复杂度与冒泡排序类似，最优情况为一开始就是有序地，最差情况为一开始逆序，最优情况的时间复杂度为O(n)，最差情况的时间复杂度为O(n^2),平均时间复杂度为O(n^2)。空间的主要开始也是用于交换元素的临时变量所占，最优为0，最差为O(1)，其平均空间复杂度为O(1)。

1. **遇到的问题及解决方法**

鸡尾酒排序的外部循环判断条件，应该写为while(leftindex <= rightindex)或者while(leftindex < rightindex)，不能写为while(leftindex != rightindex)，每次将最大的数和最小的数分别放置在数列尾和数列头之后，leftindex和rightindex都会自增1或自减1，减小需要遍历的范围，直到它们相等时停止，如果将条件设为不等，相等之后会继续自增自减，使得左边界比又边界还大，造成数组越界问题，和二分法选择的思想有些类似。

**六、算法策略的英文描述（字数>200）**

**Selection Sort:**

Using a subscript to mark the elements which are traversed at the beginning of each loop, and then traverse them backwards. As long as the element is smaller than the initial marked element, the mark is changed to the current subscript. And then swaps this two elements. Placing the smallest element which can be traversed each time in the front of sequence. So that the selection continues, making the unordered sequence into order.

**Bubble Sort:**

Using double loop .The inner loop loops the n data one by one and swaps the numbers in the sequence to make the largest number move backwardly until making it become the last one when the first loop of the outer loop. Then, when the second loop of the outer loop, the inner loop only needs to consider n-1 numbers. Because the largest number of the sequence already in the end, no need to be considered again. And then the second largest number will be moved to the penultimate position......

The outer loop will float a maximum number after each loop, and then put them in the back row. Similarly, the small number will be floated to the front of this sequence. After the loop is completed, the data in the sequence has been sorted .

**Cocktail Sort:**

While the results are achieved by looping adjacent data exchange from the front to the back, loop data exchange from the back to the front. The former swaps for the unsorted maximum, and the latter swaps for the unsorted minimum. Bidirectional implementation of bubble sort process.