# 

## 《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. 实现快速排序的算法。
2. 实现归并排序的算法。

**二、实验内容**

**1. 快速排序。（完成实验代码、伪代码）**

快速排序（请完成快速排序的设计策略描述、时间复杂度的分析、伪代码以及代码实现）。

**2. 归并排序。（完成实验代码、伪代码）**

归并排序（请完成归并排序的设计策略描述、时间复杂度的分析、伪代码以及代码实现）。

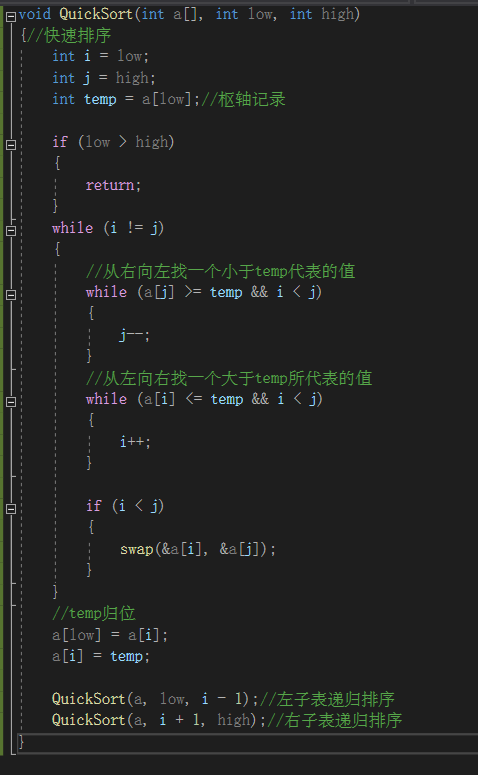
**三、实验目的**

1. 理解分治法的思想，算法策略。

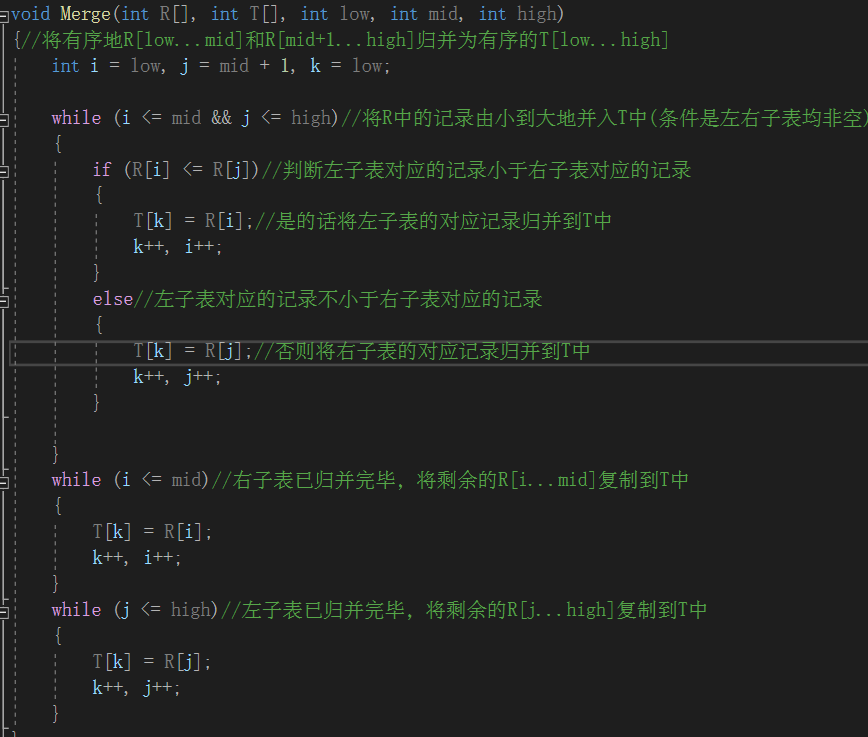
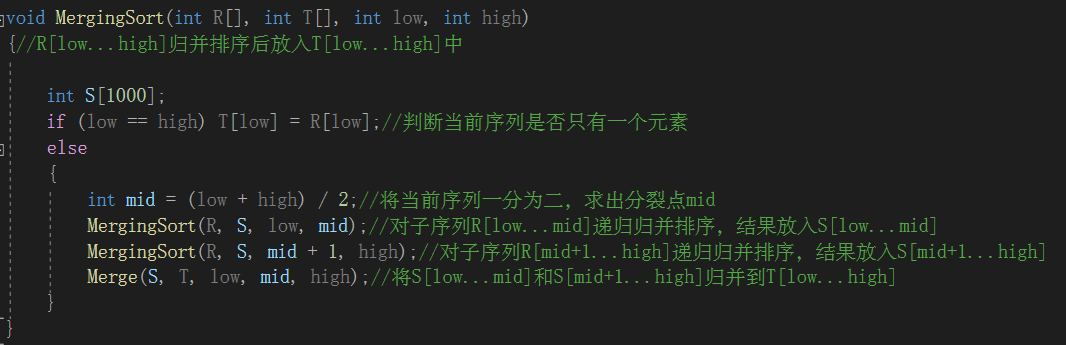
2. 掌握利用分治法解决问题的基本思想，会用高级语言对算法进行描述，并对算法复杂度（时间和空间）进行分析。

**四、实验代码**

**快速排序：**

****

**归并排序：**

****

**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**

**快速排序：**

Partition(A,p,r) //p、r为数组下标

x = A[r] //将最后一个元素作为主元素

i = p-1 // i指向的是比主元素小的位置，

for j = p to r-1 //从第一个元素开始到倒数第二个元素结束，比较确定主元素的位置

do if A[j] <= x

then i = i+1 //如果比主元素小，则把i=i+1的位置上的元素和j位置发现小元素互换

exchange A[i] <-> A[j]

exchange A[i+1]<->A[r] //最终确定主元的位置

return i+1 //返回主元的位置

end

QuickSort(A,p,r)

if p<r

q = Partition(A,p,r) //确定划分位置

QuickSort(A,p,q-1) //子数组A[p...q-1]

QuickSort(Q,q+1,r) //子数组A[q+1...r]

end

**归并排序：**

void Merge( T A,T &B, s,m,e)//将A中的A[s..m]和A[m+1..e]合并成B[s..e]

{

for( j = m+1,k = s; j <=e && s<= m;k++)

{

if( A[j] < A[s] ) B[k] = A[j++];

esle B[k] = A[s++];

}

if( s <= m ) B[k..e] = A[s..m];

if( j <= e) B[k..e] = A[j..e];

}

void MSort(T A,T &B,int s,int t)

{

if(s== t) B[s] = A[s];

else

{

m = (s+t)/2;

MSort(A,B,s,m);

MSort(A,B,m+1,t);

Merge(A,B,s,m,t);

}

}

1. **算法设计策略描述**

**快速排序：**

在待排序的n个记录中任取一个记录（通常选取第一个数据）作为枢轴，设其关键字为pivotkey。经过一趟排序后，把所有关键字小于pivotkey的记录交换到前面，把所有关键字大于pivotkey的记录交换到后面，结果将待排序的记录分成两个子表，最后将枢轴放在分界处位置。然后对左右子表重复上述过程，直到每一个子表只有一个记录时，排序完成。

**归并排序：**

假设初始序列含有n个记录，则可以看成是n个有序的子序列，每个子序列的长度为1，然后两两归并，得到n/2个长度为2或1的序列，在归并的时候依照元素数据大小对其排序（主要是交换位置），；再两两归并，……,如此重复，直到得到一个长度为n的有序序列为止。在这其中会用到一个临时的的辅助空间，数组或者顺序表，将左右子表先放入临时的辅助空间，然后再将其放入目标数组或者顺序表。

1. **算法时空复杂度分析**

**快速排序：**

快速排序的趟数取决于递归树的深度。

最好情况：每一趟排序后都能将记录序列均匀地分割成两个长度大致相等的子表，类似折半查找，在n个元素的序列中，对枢轴定位所需的时间为O(n)，此时的时间复杂度是大约是O(nlog2n)。

最坏情况：在待排序序列已经排好序的情况下，其递归树成为单支树，每次划分只得到一个比上一次少一个记录的子序列。这样，必须经过n-1趟才能将所有的记录定位，而且第i趟需要经过n-i次比较。

平均情况下，快速排序的时间复杂度是O(nlog2n)。

快速排序是递归的，执行时需要有一个栈来存放相应的数据，最大递归调用次数与递归树的深度一致，所以最好情况下的空间复杂度为O(log2n)，最坏情况下为O(n)。

**归并排序：**

当有n个记录时，需进行log2n趟归并排序，每一趟归并，其关键字比较次数不超过n，元素移动次数都是n，因此归并排序的时间复杂度为O(nlog2n)。

用顺序表实现的归并排序，需要和待排序记录个数相等的辅助存储空间，所以空间复杂度为O(n)。

1. **遇到的问题及解决方法**

快速排序是一种不稳定排序，如果我的序列是4,4……4，类似这样的，枢轴、low值代表元素、high值代表元素是相同的，就类似于题设要求的数据，快速排序就会陷入死循环，而且应该用一个临时变量i、j来代替low、high，不能直接用low、high自增自减，这样的话最后枢轴的记录会被改变，陷入死循环……

我的解决办法是不把快速排序分成两个函数，就直接写到一个里面，避免死循环。

**六、算法策略的英文描述（字数>200）**

Quick sort:

Take any one of the n records hasn't be sorted (usually the first data is selected) as the pivot, and set its key to pivotkey. After a round of sorting, all records with keywords less than pivotkey are swapped to the front, and all records with keywords greater than pivotkey are swapped to the back. As a result, the records to be sorted are divided into two subtables. Then the pivot is placed at the boundary . Then repeat the above process for the left and right subtables, until each subtable has only one record, the sorting is completed.

Merging Sort:

Assuming that the initial sequence contains n records, it can be regarded as n ordered subsequences, each subsequence is 1 in length, and then merged in pairs to obtain n / 2 sequences of length 2 or 1. In the merged time, they are sorted according to the size of the element data (mainly the exchange position), and then merged one by one, ..., and so on, until an ordered sequence of length n is obtained. A temporary auxiliary space, an array or a sequential table will be used in this, and the left and right subtables are put into the temporary auxiliary space first, and then they are put into the target array or the sequential table.