电子科技大学信息与软件工程学院

实 验 报 告

学　　号　2016220304022

姓　　名　罗悦

（实验）　课程名称　数据结构与算法

理论教师　陈安龙

实验教师　陈安龙

电子科技大学教务处制表

电 子 科 技 大 学

实　验　报　告（4）

学生姓名：罗悦　　**学　号：**2016220304022　　**指导教师：**陈安龙

实验地点：清水河科技实验大楼　　**实验时间：**2017.6.1

一、实验室名称：学校实验中心软件实验室

二、实验项目名称：编程实现快速排序和折半查找算法

三、实验学时：4

四、实验原理：

快速排序的基本思想是：通过一躺排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一不部分的所有数据都要小，然后再按次方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

假设要排序的数组是 A[1]……A[N]，首先任意选取一个数据（通常选用第一个数据）作为关键数据，然后将所有比它的数都放到它前面，所有比它大的数都放到它后面，这个过程称为一躺快速排序。一躺快速排序的算法是：

1）设置两个变量 I、J，排序开始的时候 I：=1，J：=N

2）以第一个数组元素作为关键数据，赋值给 X，即 X：=A[1]；

3）从 J 开始向前搜索，即（J：=J-1），找到第一个小于 X 的值，两者交换；

4）从 I 开始向后搜索，即（I：=I+1），找到第一个大于 X 的值，两者交换；

5）重复第 3、4 步，直到 I=J。

二分法查找（折半查找）的基本思想：

（1）确定该区间的中点位置：mid=（low+high）/2

min 代表区间中间的结点的位置，low 代表区间最左结点位置，high 代表区间最右结点位置

（2）将待查 a 值与结点 mid 的关键字（下面用 R[mid].key）比较，若相等，则查找成功，否则确定新的查找区间：

A)如果 R[mid].key>a，则由表的有序性可知，R[mid].key 右侧的值都大于 a，所以等于 a 的关键字如果存在，必然在 R[mid].key 左边的表中,这时 high=mid-1;

B)如果 R[mid].key<a,则等于 a 的关键字如果存在，必然在 R[mid].key 右边的表中。这时 low=mid；

C)如果 R[mid].key=a，则查找成功。

（3）下一次查找针对新的查找区间，重复步骤（1）和（2）

（4）在查找过程中，low 逐步增加，high 逐步减少，如果 high<low，则查找失败。

五、实验目的：

本实验通过实现快速排序和折半查找算法，使学生理解如何实现快速查找和排序的基本算法思想。通过练习，加强对算法的理解，提高编程能力。

六、实验内容：

（1）实现数据序列的输入;

（2）实现快速排序算法,并对输入的序列排序后输出；

（3）实现折半查找算法,并在步骤(2)排序后的序列上,进行任意地查找,并输出查询结果。

七、实验器材（设备、元器件）：

PC 机一台，装有 C 语言集成开发环境。

八、数据结构与程序：

```
#include<stdio.h>
#define MAX 100

int list[MAX];
int l;

void read(){
    int i;
    printf("请输入要进行排序的表的长度：\n");
    scanf("%d",&l);
    printf("请依次录入数据：\n");
    for(i=1;i<=l;i++){
        scanf("%d",&list[i]);getchar();
    }
}

int Quick_Partition(int k,int j){  //快速排序的一次划分，左端位置为 i，右端位
```

```
置为 j
    list[0]=list[k];
    while(k<j){
        while(k<j&&list[j]>=list[0])j--;  //j 往左移
        if(k<j){
            list[k]=list[j];
            k++;
        }                                 //比支撑点关键字小的记录交换到左边
        while(k<j&&list[k]<=list[0])k++;  //i 往右移
        if(k<j){
            list[j]=list[k];
            j--;
        }                                 //比支撑点关键字大的记录交换到右边
    }
    list[k]=list[0];    //支撑点记录置入正确位置
    return k;           //返还支撑点位置
}

void Quick_Sort(int s,int t){      //对表 list 做快速排序
    int i;
    if(s<t){
        i=Quick_Partition(s,t);    //调用划分过程将表一分为二
        Quick_Sort(s,i-1);         //递归调用快速排序过程
        Quick_Sort(i+1,t);         //递归调用快速排序过程
    }
}

void out(){
    int i;
    for(i=1;i<=l;i++){
        printf("%d\t",i);
    }
    printf("\n");
    for(i=1;i<=l;i++){
        printf("%d\t",list[i]);
    }
    printf("\n");
}

int BinSearch(int K){
    int low=1;
    int hight;
    int mid;
    hight=l;
```

```
    while(low<=hight){
        mid=(low+hight)/2;
        if(list[mid]==K){
            return mid;
        }
        if(list[mid]>K){
            hight=mid-1;
        }
        else{
            low=mid+1;
        }

    }
    return 0;
}

int main(){
    int s,t,K,m;
    read();
    printf("初始表为：\n");
    out();
    printf("请输入要进行排序的段落的第一个数据的序号：");
    scanf("%d",&s);getchar();
    printf("请输入要进行排序的段落的最后一个数据的序号：");
    scanf("%d",&t);getchar();
    Quick_Sort(s,t);
    printf("进行了快速排序后的表为：\n");
    out();
    printf("请输入你想查找的数据：\n");
    scanf("%d",&K);
    m=BinSearch(K);
    if(m!=0){
        printf("所查找的数据在表中的位置为：%d\n",m);
    }
    else{
        printf("所查找的数据在表中不存在！\n");
    }
    return 0;
}
```

九、程序运行结果：

图 1-1

图 1-1 先录入需要排序的表的长度，并依次录入数据，并打印初始表。

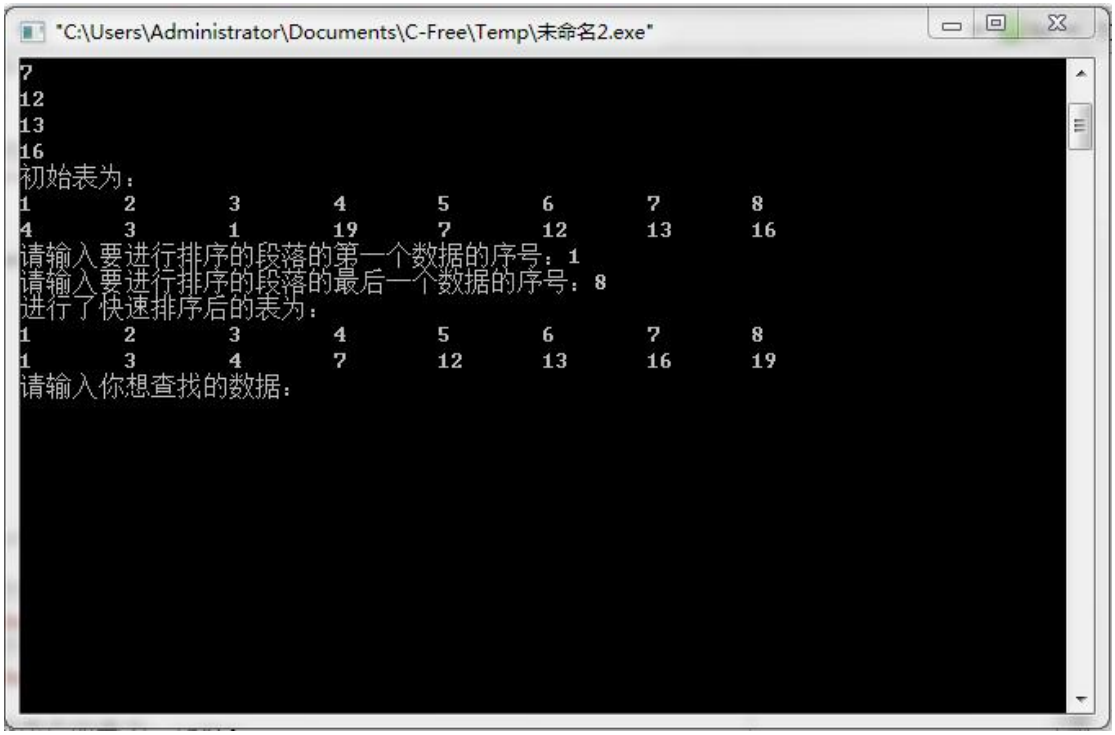

图 1-2

图 1-2 先输入需要进行排序的第一个数据的序号和最后一个需要排序的序号，并对序号内的数据进行快速排序，并打印排序后的表。

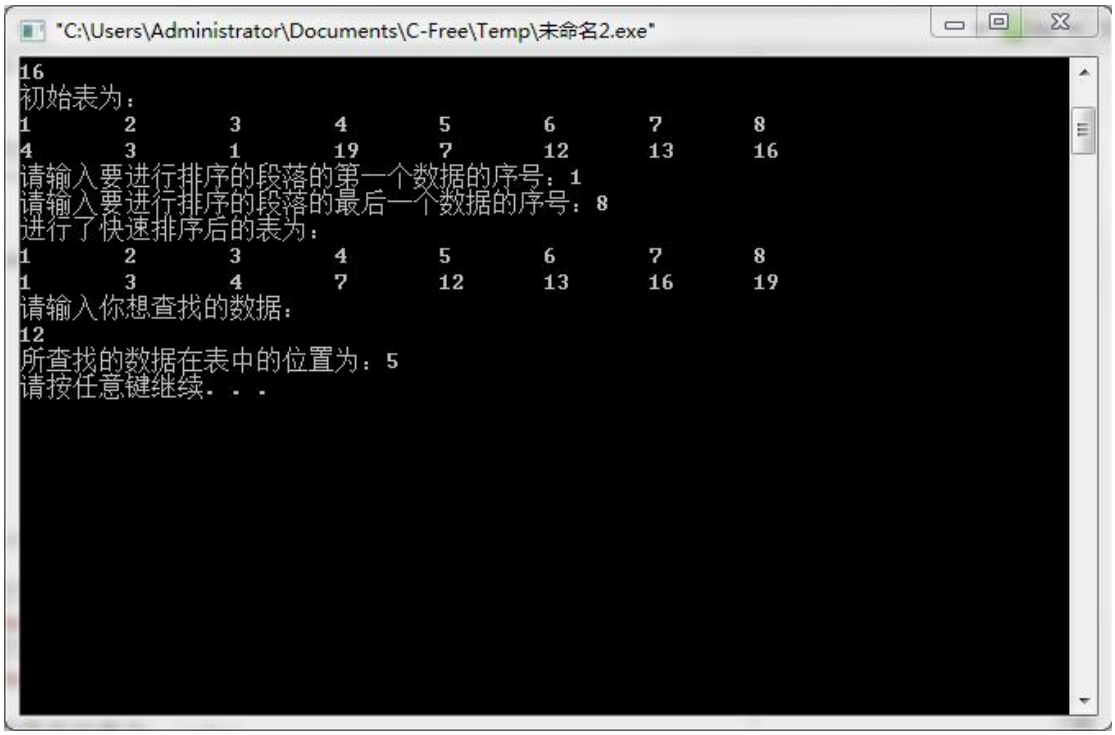

图 1-3

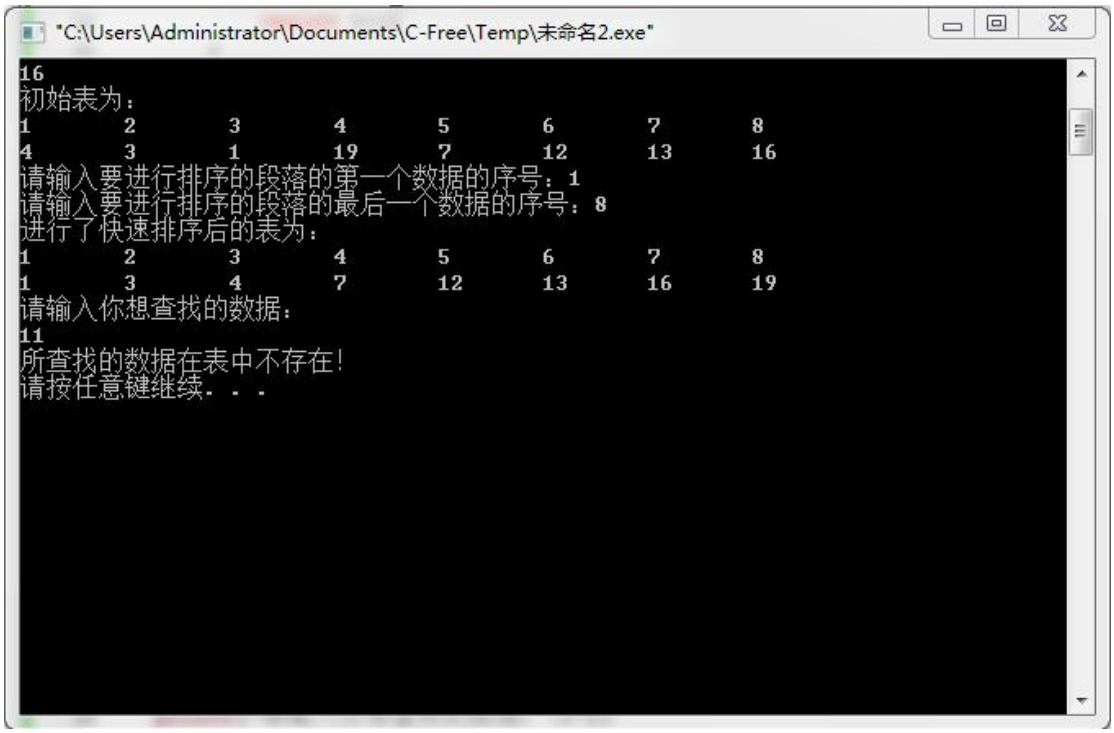

图 1-4

图 1-3 和图 1-4 任意输入所要查找的数据，若表中含有此数据，则打印出它的序号，若没有此数据，则打印“所查找的数据在表中不存在”。

十、实验结论：

本次实验通过快速排序的方法把原始表按数值大小的升序重新打印一遍。并且用折半查找的方法准确的找到了表中所需要的数据的位置。使得实验完成。

十一、总结及心得体会：

本次实验运用了快速排序，通过一躺排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一不部分的所有数据都要小，然后再按次方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。这样的算法比较简便，使得实验编程代码显得更加简洁。快速排序是冒泡排序的改进版，是目前已知的最快的排序方法。它的优点是极快，数据移动少，缺点是不稳定。但通过这次实验我也了解到了最快的算法的技巧性，使得我以后能在自己编写的程序运用这些简便的思想，获得突破与创新。

二分法查找（折半查找）的优点是比较次数少，查找速度快，平均性能好；其缺点是要求待查表为有序表，且插入删除困难；因此，折半查找方法适用于不经常变动而查找频繁的有序列表。首先，假设表中元素是按升序排列，将表中间位置记录的关键字与查找关键字比较，如果两者相等，则查找成功；否则利用中间位置记录将表分成前、后两个子表，如果中间位置记录的关键字大于查找关键字，则进一步查找前一子表，否则进一步查找后一子表。重复以上过程，直到找到满足条件的记录，使查找成功，或直到子表不存在为止，此时查找不成功。

此次实验让我的编程能力又得到了一个提升，并且能够仔细得处理好每一个细节，写好每一个程序。