电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2016220203031

姓 名 沈若玥

（实验） 课程名称 数据结构与算法

理论教师 郝宗波

实验教师 李美蓉

**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：**沈若玥  **学 号：2016220203031**

**指导教师：**郝宗波

**实验地点：**清水河科技实验大楼 **实验时间：**2017.06.11

**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>

#include <stdlib.h>

#define MAX 10

void readData(int a[], int\* num) {

    FILE\* fp;

    int c;

    \*num = 0;

    fp=fopen("/Users/ryshen/Desktop/test2.txt","r");

    if(!fp)

    {

        printf("The file does not exist!\n");

        exit(0);

    }

    for (int i = 0; fscanf(fp,"%d",&c)!=EOF; i++) {

        a[i] = c;

        (\*num)++;

    }

    fclose(fp);

}

void swap(int\* a, int\* b){

    int temp;

    temp = \*a;

    \*a = \*b;

    \*b = temp;

}

void quickSort(int a[],int start, int tail) {

    int i, j, x;

    i = start;

    j = tail;

    x = a[start];

    while (i < j) {

        while(a[j] >= x && j > i) {

            j--;

        }

        swap(&a[i], &a[j]);

        while(a[i] <= x && i < j) {

            i++;

        }

        swap(&a[i], &a[j]);

    }

    if (start < tail) {

        quickSort(a, start, i-1);

        quickSort(a, i+1, tail);

    }

}

int binSearch(int a[], int x, int num) {

    int low = 0, high = num-1, mid, time = 0;

    while(low <= high) {

        mid = (low + high) / 2;

        if (x == a[mid]) {

            time++;

            if (a[mid-1] == x || a[mid+1] == x){

                printf("existence & number > 1\n");

            }

            else {

                printf("existence & number = 1\n");

            }

            return 1;

        }

        else if (x < a[mid]) {

            high = mid - 1;

        }

        else {

            low = mid + 1;

        }

    }

    printf("nonexistence\n");

    return 0;

}

t(a, 0, num-1);

    printf("sort:");

    for (int i = 0; i < num; i++){

        printf(" %d", a[i]);

    }

    printf("\n");

    for (int i = 0; i < 3; i++){

        printf("input search data:\n");

        scanf("%d", &x);

        binSearch(a, x, num);

    }

}

 int main() {

    int a[MAX];

    int num, x;

    readData(a, &num);

    printf("before:");

    for (int i = 0; i < num; i++){

        printf(" %d", a[i]);

    }

    printf("\n");

    quickSort(a, 0, num-1);

    printf("sort:");

    for (int i = 0; i < num; i++){

        printf(" %d", a[i]);

    }

    printf("\n");

    for (int i = 0; i < 3; i++){

        printf("input search data:\n");

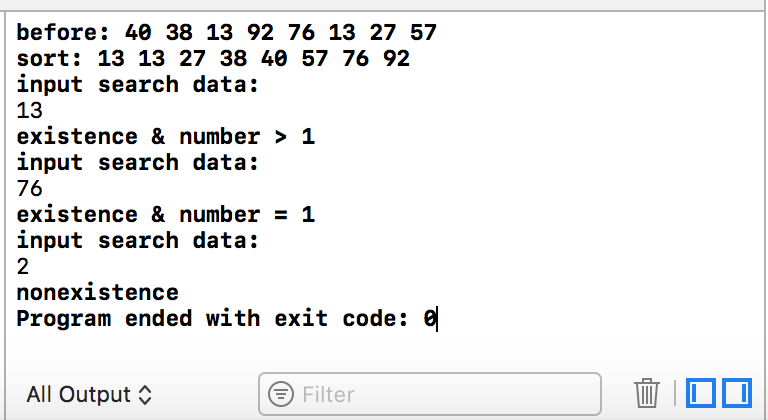
        scanf("%d", &x);

        binSearch(a, x, num);

    }

}

**九、程序运行结果：**

****

**十、实验结论：**

测试正确，程序满足项目需求。

**十一、总结及心得体会：**

总结：此次实验考察了快速排序和折半查找的理解与掌握，快速排序是一种广泛应用的排序方法，是一个递归算法，平均性能优于很多排序方法，但是需要一个栈空间来进行递归