电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2017221302006

姓 名 周玉川

（实验） 课程名称 数据结构

理论教师 刘勇国

实验教师 肖老师

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告（2）**

**学生姓名：** 周玉川  **学 号：2017221302006**  **指导教师： 刘勇国，肖老师**

**实验地点：**清水河基础实验大楼 **实验时间：2018.6.15**

**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：**

|  |
| --- |
| //快速排序和折半查找 by Zhou Yuchuan  #include <bits/stdc++.h>  using namespace std;  #define END 999999  #define Info char  #define MAX 100  typedef struct{  int key;  Info data;  }RecType;  void QuickSort(RecType R[],int l,int r);  int partition(RecType R[],int l,int r);  int BinarySearch(RecType R[],int l,int r,int data);  int UpperBound(RecType R[],int l,int r,int data);  int LowerBound(RecType R[],int l,int r,int data);  void ReadFile(FILE\*fp,RecType R[],int \*n);  void Disp(RecType R[],int n);  void ChooseFind(RecType R[],int n)  {    printf("\n");  printf("Three choice and operation prompt:\nBinarySort: B/b\nUpperBound: U/u\nLowerBound: L/l\n");  printf("Please choose the method of sort: ");  char ch;  scanf("%c",&ch);  printf("\n");  printf("Enter the number you want to search: ");  int num;scanf("%d",&num);printf("\n");  switch(ch)  {  case 'B':case 'b':printf("The location of %d is %dth\n",num,BinarySearch(R,0,n,num));break;  case 'U':case 'u':printf("The location of %d is %dth\n",num,UpperBound(R,0,n,num));break;  case 'L':case 'l':printf("The location of %d is %dth\n",num,LowerBound(R,0,n,num));break;  default: printf("Illeagal Input\n");break;  }  }  int main(void)  {  FILE\*fp;  if ((fp=fopen("in.txt","r"))==NULL)  {  perror("");  exit(EXIT\_FAILURE);  }  RecType R[MAX];int n=0;  memset(R,0,sizeof(R));  ReadFile(fp,R,&n);fclose(fp);  printf("Before sort: ");Disp(R,n);  QuickSort(R,0,n);printf("After sort: ");Disp(R,n);  ChooseFind(R,n);  return 0;  }  void QuickSort(RecType R[],int l,int r)  {  int i;  if (l<r-1)  {  i=partition(R,l,r);  QuickSort(R,l,i);  QuickSort(R,i,r);  }  }  int partition(RecType R[],int l,int r)  {  int i=l,j=r-1;  RecType tmp=R[i];  while (i<j)  {  while (i<j&&R[j].key>=tmp.key)  j--;  R[i]=R[j];  while (i<j&&R[i].key<=tmp.key)  i++;  R[j]=R[i];  }  R[i]=tmp;  return i+1;  }  int BinarySearch(RecType R[],int l,int r,int data)  {  int mid;  while (l<r)  {  mid=(l+r)/2;  if (data==R[mid].key) return mid+1;  else if (R[mid].key>data)  r=mid;  else  l=mid+1;  }  printf("NOT Find!\n");  exit(EXIT\_FAILURE);  }  int UpperBound(RecType R[],int l,int r,int data)  {  int mid;  while (l<r)  {  mid=(l+r)/2;  if (R[mid].key<=data)  l=mid+1;  else r=mid;  }  if (R[l-1].key!=data)  {  printf("NOT Find!\n");  exit(EXIT\_FAILURE);  }  return l;  }  int LowerBound(RecType R[],int l,int r,int data)  {  int mid;  while (l<r)  {  mid=(l+r)/2;  if (R[mid].key>=data)  r=mid;  else l=mid+1;  }  if (R[l].key!=data)  {  printf("NOT Find!\n");  exit(EXIT\_FAILURE);  }  return l+1;  }  void ReadFile(FILE\*fp,RecType R[],int\*n)  {  int i=0;  int num;  while (fscanf(fp,"%d",&num)!=EOF){  R[i++].key=num;  (\*n)++;  }  R[i].key=END;  }  void Disp(RecType R[],int n)  {  int i;  for (i=0;i<n;i++)  {  printf("%d%c",R[i].key,i==n-1?'\t':' ');  }  printf("The sum of number is %d\n",n);  } |

**九、程序运行结果：**

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **（1）实验结果1：** | | **结果评价: 成功排序，界面正常。** | |
| |  | | --- | | **（2）实验结果2：** | |
| |  | | --- | | **结果评价：排序后，进行查找操作，有三种折半查找方式可供选择，创新。** | | **（3）实验结果3：** | |
| |  | | --- | | **结果评价: 选择正常折半查找，结果符合事实。** | |
| |  | | --- | | **实验结果3：** | | **结果评价：选择Upper Bound方式，查找同一个数，结果与上一个不同，就是因为这个算法会返回该数字的最大下表，符合期望。** | |

**十、实验结论：**

**试验成功，结论正确。**

**十一、总结及心得体会：**

1. 认真研究书本，虽然对新手来说有些困难，但是认真吸收后会受益无穷。
2. 平时多写代码，多练练，不能只在上机的时候才敲敲键盘，切勿眼高手低，没有简单的知识，只有勤快的学生。
3. 怀抱着兴趣去写代码。
4. 虽然实验偏简单，但是还是能找到亮点以及自己学习的地方。比如这次想到了采用三种折半查找方式。
5. 继续加油。