《数据结构与算法》实验报告

实验五：查找和排序

教　师：潘晔

学　生：梁书恺

学　号：2022040906023

时　间：11.9

地　点：科 B119

一、**ex5-1**

1. **问题分析：**

1）题意理解

从键盘输入数组，利用数组生成二叉排序树，再调用中序遍历函数得到有序序列，再分别进行顺序查找、二分查找、二叉排序查找。

2）数据结构设计

```
typedef struct BSTNode
{
    BSTNode *lchild;
    BSTNode *rchild;
    int data;
} BSTNode;
```

3）关键算法思路

```
void createBST(BSTNode **T, int key)
{
    if (*T == NULL)
    {
        *T = (BSTNode *)malloc(sizeof(BSTNode));
        if (*T == NULL)
            errorHandler(4);
        (*T)->data = key;
        (*T)->lchild = (*T)->rchild = NULL;
    }
    else if (key < (*T)->data)
        createBST(&(*T)->lchild, key);
    else if (key > (*T)->data)
        createBST(&(*T)->rchild, key);
}
```

4）健壮性设计

通过调用统一出错处理函数 errorHandler 打印出错信息并跳出程序。

5）性能分析

时间复杂度：O(n);

空间复杂度：O(n);

2. **测试数据与运行结果截图：**

```
请输入序列长度：8
请输入序列：3 13 40 50 10 17 43 70
中序遍历：3 10 13 17 40 43 50 70
请输入要查找的关键字：13
顺序查找结果：查找成功，经过了 2 次比较
二分查找结果：查找成功，经过了 3 次比较
二叉排序树查找结果：查找成功，经过了 1 次比较
```

```
请输入序列长度：-1
输入的序列长度不合法
```

```
请输入序列长度：8
请输入序列：3 13 40 50 10 17 43 70
中序遍历：3 10 13 17 40 43 50 70
请输入要查找的关键字：30
顺序查找结果：查找失败
二分查找结果：查找失败
二叉排序树查找结果：查找失败
```

3. **上机时遇到的问题**

① 问题现象： 二分查找异常 **原因：**在中序递归遍历时未给数组赋值 **解决办法：**使用一个 static 变量记录次数并为数组赋值。

4. 程序代码

ex1.c

ex1.h

二、ex1-2

1. 问题分析：

1）题意理解

输入姓名和学号，根据姓名利用哈希表存储，并在表中查找。

2）数据结构设计

```
typedef struct
{
    char name[6];
    int id;
} HashTable;
```

3）关键算法思路

```
int hashSearch(char *name)
{
    int index = hash(name);
    int i = 0;
    while    (hashTable[index].id    !=    0    &&
strcmp(hashTable[index].name, name) != 0)
    {
        index = (index + 1) % 40;
        i++;
        if (i == 40)
            return -1;
    }
    if (hashTable[index].id == 0)
        return -1;
    else
        return index;
}
```

```
int hash(char *name)
{
    int i, sum = 0;
    for (i = 0; i < strlen(name); i++)
    {
        sum += name[i];
    }
    return sum % 40;
}
```

4）健壮性设计

通过调用统一出错处理函数 errorHandler 打印出错信息并跳出程序。

5）性能分析

时间复杂度：O(1);

空间复杂度：O(n);

2. 测试数据与运行结果截图：

```
请输入 37 个学生的姓名和学号：
hqghu 41 eayln 11478 fdxfi 23281 cvscx 4827 gbwkf 292 qduxw
5447 nfozv 25667 rtkjp 31322 epggx 28253 pnrvy 20037 tmwcy
12316 yycqp 9040 vikef 15890 mznim 24393 kasvw 18756 renzk
16944 cxfxt 16118 sgyps 4639 adpoo 31673 fxzbc 6270 ejuvp
23986 aboyg 23655 oeylf 16941 bnplj 18007 rvipy 14945 myehw
6422 nqrqp 900 xujjl 27624 ovaow 3602 xwhms 4596 cbxco 8281
sfzkv 6900 txdkn 24648 yjyhf 22813 xjswn 20600 kufnu 6224
xzrzb 3195
请输入要查找的姓名：sfzkv
学号：6900
```

```
请输入 37 个学生的姓名和学号：
hqghu 41 eayln 11478 fdxfi 23281 cvscx 4827 gbwkf 292 qduxw
5447 nfozv 25667 rtkjp 31322 epggx 28253 pnrvy 20037 tmwcy
12316 yycqp 9040 vikef 15890 mznim 24393 kasvw 18756 renzk
16944 cxfxt 16118 sgyps 4639 adpoo 31673 fxzbc 6270 ejuvp
23986 aboyg 23655 oeylf 16941 bnplj 18007 rvipy 14945 myehw
6422 nqrqp 900 xujjl 27624 ovaow 3602 xwhms 4596 cbxco 8281
sfzkv 6900 txdkn 24648 yjyhf 22813 xjswn 20600 kufnu 6224
xzrzb 3195
请输入要查找的姓名：liang
查找错误！
```

3. **上机时遇到的问题**

   1 问题现象： 数据结构设计出现问题 **原因：**在节点中存储姓名指针不方便 **解决办法：**在节点中直接存储姓名

4. **程序代码**

   ex2.c

   ex2.h

三、ex1-3

1. **问题分析：**

1）题意理解

编写各种排序算法，对在主程序中输入的一组元素进行排序，并输出每趟的排序结果。

2）数据结构设计

```
int a[10];
```

3）关键算法思路

```
void quickSort(int *a, int low, int high)
{
    if (low < high)
    {
        int pivotpos = partition(a, low, high);
        quickSort(a, low, pivotpos - 1);
        quickSort(a, pivotpos + 1, high);
    }
}
```

4）健壮性设计

对于存在重复元素的序列，程序不会出错。

5）性能分析

时间复杂度：O(1);

空间复杂度：O(1);

2. 测试数据与运行结果截图：

```
原始数据：513 87 512 61 908 170 897 275 653 462

简单选择排序：
第 1 趟排序结果：61 87 512 513 908 170 897 275 653 462
第 2 趟排序结果：61 87 512 513 908 170 897 275 653 462
第 3 趟排序结果：61 87 170 513 908 512 897 275 653 462
第 4 趟排序结果：61 87 170 275 908 512 897 513 653 462
第 5 趟排序结果：61 87 170 275 462 512 897 513 653 908
第 6 趟排序结果：61 87 170 275 462 512 897 513 653 908
第 7 趟排序结果：61 87 170 275 462 512 513 897 653 908
第 8 趟排序结果：61 87 170 275 462 512 513 653 897 908
第 9 趟排序结果：61 87 170 275 462 512 513 653 897 908

直接插入排序：
第 1 趟排序结果：87 513 512 61 908 170 897 275 653 462
第 2 趟排序结果：87 512 513 61 908 170 897 275 653 462
第 3 趟排序结果：61 87 512 513 908 170 897 275 653 462
第 4 趟排序结果：61 87 512 513 908 170 897 275 653 462
第 5 趟排序结果：61 87 170 512 513 908 897 275 653 462
第 6 趟排序结果：61 87 170 512 513 897 908 275 653 462
第 7 趟排序结果：61 87 170 275 512 513 897 908 653 462
第 8 趟排序结果：61 87 170 275 512 513 653 897 908 462
第 9 趟排序结果：61 87 170 275 462 512 513 653 897 908

改进的冒泡排序：
第 1 趟排序结果：87 512 61 513 170 897 275 653 462 908
第 2 趟排序结果：87 61 512 170 513 275 653 462 897 908
第 3 趟排序结果：61 87 170 512 275 513 462 653 897 908
第 4 趟排序结果：61 87 170 275 512 462 513 653 897 908
第 5 趟排序结果：61 87 170 275 462 512 513 653 897 908
第 6 趟排序结果：61 87 170 275 462 512 513 653 897 908

快速排序：
61 87 170 275 462 512 513 653 897 908

归并排序：
61 87 170 275 462 512 513 653 897 908
```

```
原始数据：513 87 512 61 908 170 87 275 653 462

简单选择排序：
第 1 趟排序结果：61 87 512 513 908 170 87 275 653 462
第 2 趟排序结果：61 87 512 513 908 170 87 275 653 462
第 3 趟排序结果：61 87 87 513 908 170 512 275 653 462
第 4 趟排序结果：61 87 87 170 908 513 512 275 653 462
第 5 趟排序结果：61 87 87 170 275 513 512 908 653 462
第 6 趟排序结果：61 87 87 170 275 462 512 908 653 513
第 7 趟排序结果：61 87 87 170 275 462 512 908 653 513
第 8 趟排序结果：61 87 87 170 275 462 512 513 653 908
第 9 趟排序结果：61 87 87 170 275 462 512 513 653 908

直接插入排序：
第 1 趟排序结果：87 513 512 61 908 170 87 275 653 462
第 2 趟排序结果：87 512 513 61 908 170 87 275 653 462
第 3 趟排序结果：61 87 512 513 908 170 87 275 653 462
第 4 趟排序结果：61 87 512 513 908 170 87 275 653 462
第 5 趟排序结果：61 87 170 512 513 908 87 275 653 462
第 6 趟排序结果：61 87 87 170 512 513 908 275 653 462
第 7 趟排序结果：61 87 87 170 275 512 513 908 653 462
第 8 趟排序结果：61 87 87 170 275 512 513 653 908 462
第 9 趟排序结果：61 87 87 170 275 462 512 513 653 908

改进的冒泡排序：
第 1 趟排序结果：87 512 61 513 170 87 275 653 462 908
第 2 趟排序结果：87 61 512 170 87 275 513 462 653 908
第 3 趟排序结果：61 87 170 87 275 512 462 513 653 908
第 4 趟排序结果：61 87 87 170 275 462 512 513 653 908
第 5 趟排序结果：61 87 87 170 275 462 512 513 653 908

快速排序：
61 87 87 170 275 462 512 513 653 908

归并排序：
61 87 87 170 275 462 512 513 653 908
```

3. **上机时遇到的问题**

   无

4. **程序代码**

   ex3.c

   ex3.h

小结

本章主要学习查找和排序。其中，二叉排序树、归并排序等较为复杂，在编写程序时较为容易出错。哈希查找时，要注意处理哈希冲突。