数据结构实验报告——实验九

学号：__20201060330__ 姓名：__胡诚皓__ 得分：_____

一、实验目的

1. **复习基本查找方法与基本排序方法；**
2. **掌握折半查找方法与快速排序方法；**
3. **了解查找与排序的应用。**

二、实验内容

1. （必做题）快速排序

假设序列中数据元素类型是字符型，对于一个序列，请采用快速排序将序列重新排列为非递减有序序列。

2. （必做题）折半查找

假设序列中数据元素类型是字符型，对于一个非递减有序序列及一个给定关键字，采用折半查找，判断关键字是否在序列中。

三、数据结构及算法描述

1. （必做题）快速排序

数据结构

题目要求以字符型作为元素类型，因此定义了 dataType 为 char 类型。

算法描述

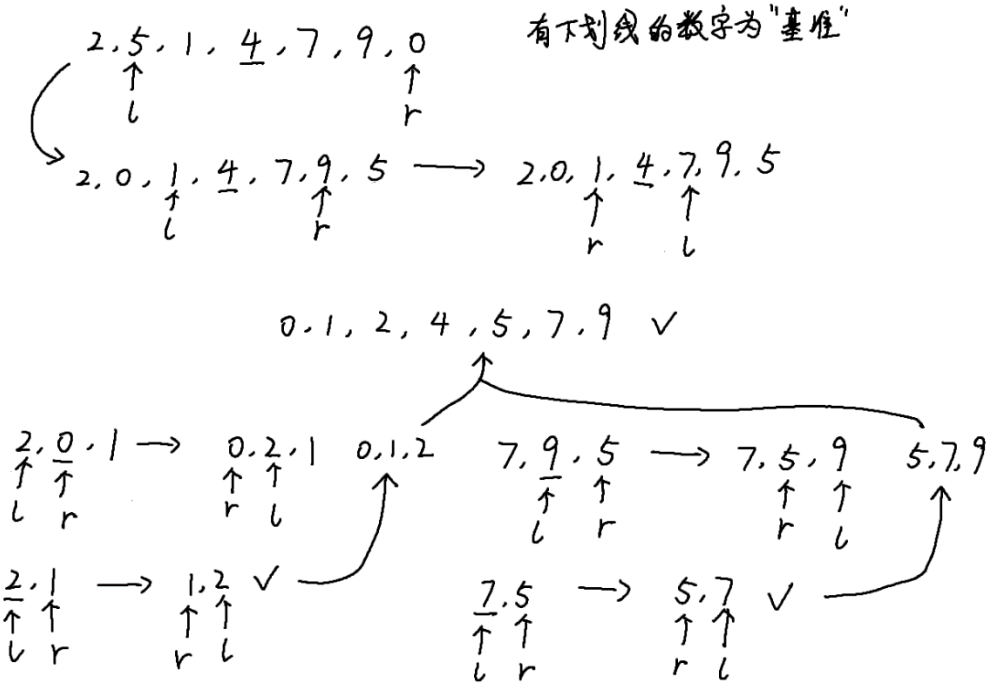
快速排序的平均时间复杂度为 O(NlogN)，是目前已知的在平均效率上最高的排序算法。快速排序采用分治的思想，通过划分、交换元素实现排序。基本步骤为：

1. 从数组中去除一个数作为“基准”

2. 将当前处理的数组分为两个部分——比“基准”大的、比“基准”小的。若为升序排列，则将较小的放到左边、较大的放到右边；降序的反之

3. 对已经分好的两个部分同样执行前两个步骤，进行递归

此处代码中是把当前要排序的部分的中间元素作为“基准”，以下为示例



2. （必做题）折半查找

数据结构

折半查找，即二分查找。时间复杂度为 O(logn)，是非常快的查找算法，但是，二分查找的使用前提是待查找的序列为顺序存储的有序序列。题目要求对字符型数据进行查找，此处使用的是字典序的大小比较，即通过 ASCII 码的大小来比较字符的大小。

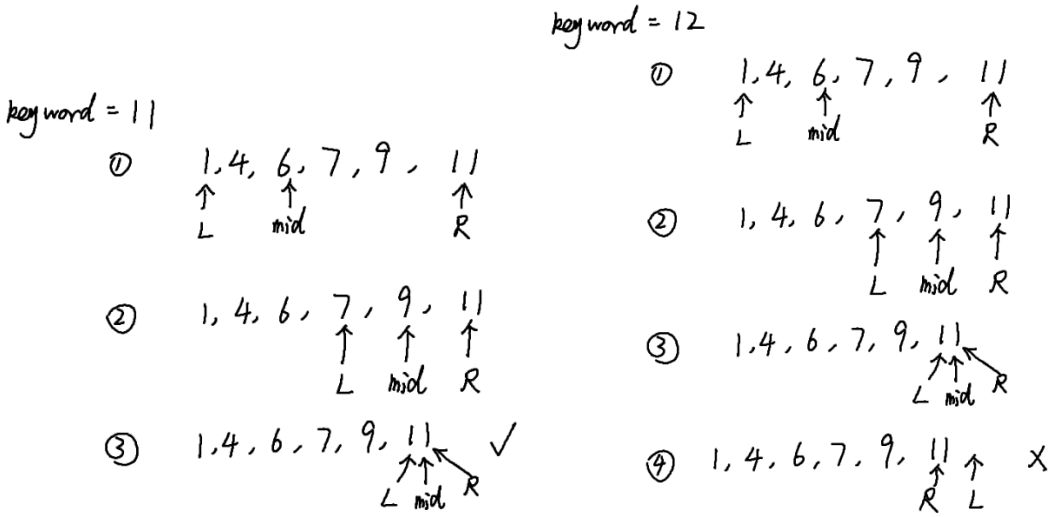
算法描述

二分查找有个的特点在于它不会查找数列的全部元素，正常情况下每次查找的元素都在一半一半地减少，这个效率是非常高的。二分查找一般遵循以下步骤（以序列升序为例，将待查的关键字记为 key）：

①取序列中间的元素与 key 进行比较，若相等则返回“找到”，退出算法

②若中间的元素比 key 小，则对右半边序列执行①；若中间的元素比 key 大，则对左半边序列执行①；若已经取不了一半的序列则返回“找不到”并退出算法。

以下分别为找得到和找不到的示意图。

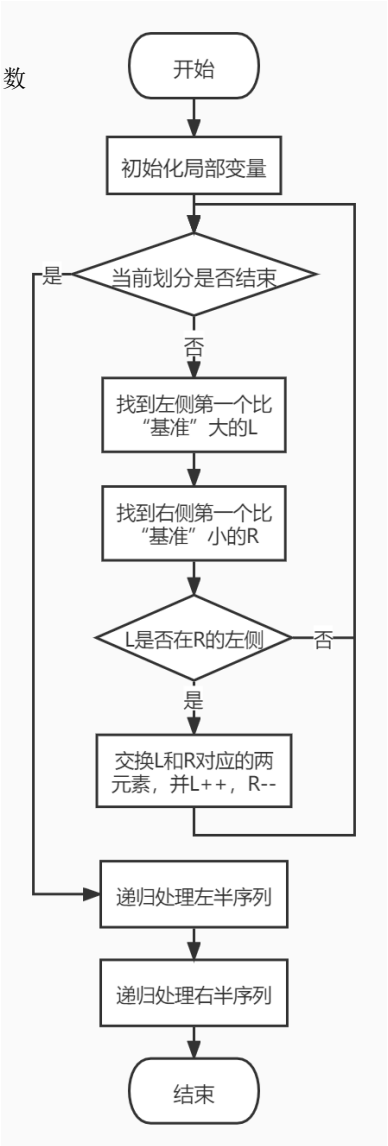


四、详细设计

1. （必做题）快速排序

题 9-1

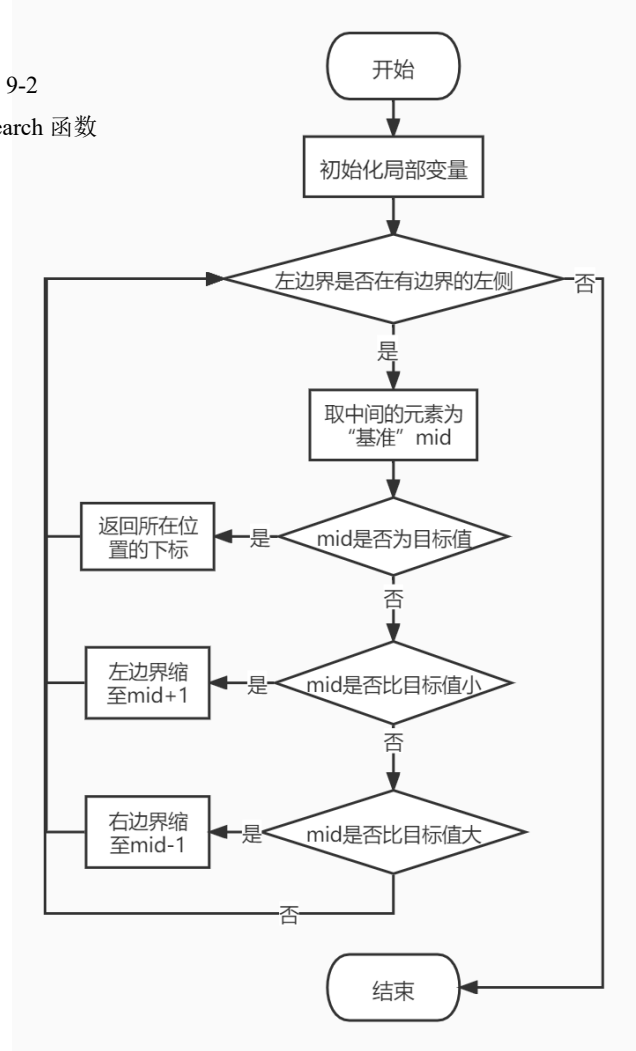
quickSort 函数



2. （必做题）折半查找

题 9-2

binarySearch 函数



五、程序代码

1. （必做题）快速排序

9-1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

typedef char dataType;

void quickSort(dataType [], int, int);

int main() {
    dataType *test, ch;
    int num;
```

```
    //预处理用于存储的空间
    printf("请输入要排序的元素个数：\n");
    scanf("%d", &num);
    test = (char *) malloc(num*sizeof(char));

    printf("请输入要排序的字符元素（以空格分隔）\n");

    for (int i = 0; i < num; i++) {
        //去除空格、换行符
        while((ch = getchar()) && isspace(ch));
        test[i] = ch;
    }

    quickSort(test, 0, num-1);

    printf("排序结果为：\n");
    for (int i = 0; i < num; i++)
        printf("%c ", test[i]);

    printf("\n");
    system("pause");
    return 0;
}

//交换数组元素的函数
void swap(dataType *a, dataType *b) {
    dataType t=*a;
    *a = *b;
    *b = t;
}

//对 arr 序列的[left,right]区间进行快速排序
void quickSort(dataType arr[], int left, int right) {
    int mid=arr[(left+right)/2];
    int l=left, r=right;
    while (l <= r) {
        //找到第一个大于等于 mid 的
        while (arr[l] < mid) l++;
        //找到第一个小于等于 mid 的
        while (arr[r] > mid) r--;
        //将两个元素交换，即分别放入比 mid 小/大的两边
        if (l <= r) {
            swap(arr+l, arr+r);
```

```
            l++;
            r--;
        }
    }
    //继续递归快排每个分段
    if (left < r) quickSort(arr, left, r);
    if (right > l) quickSort(arr, l, right);
}
```

2. （必做题）折半查找

9-2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

typedef int dataType;

int binarySearch(dataType [], int n, dataType);

int main() {
    dataType test[10]={'A','C','D','E','J','L','O','a','d','g'};
    int index;
    dataType ch;
    printf("待查找字符序列为：\n");
    for (int i = 0; i < 10; i++)
        printf("%c ", test[i]);
    printf("\n");

    while (1) {
        printf("请输入要查找的字符（减号-为退出）\n");
        //去除空格、换行符
        while ((ch = getchar()) && isspace(ch));
        if (ch == '-')
            break;
        index = binarySearch(test, 10, ch);
        if (index == -1)
            printf("未找到\n");
        else
            printf("找到了，第一个'%c'的下标为%d\n", ch, index);
    }

    system("pause");
    return 0;
```

```
}

//在非递减序列 arr 中二分查找 keyword，若在 arr 中找到了，则返回下标；若没找到，则返回-1
//需要给入 arr 的长度 n
int binarySearch(dataType arr[], int n, dataType keyword) {
    int left=0, right=n-1;
    int mid;
    while (left <= right) {
        mid = (left+right)/2;
        if (arr[mid]==keyword)
            return mid;
        else if (arr[mid] < keyword)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1;
}
```

六、测试和结果

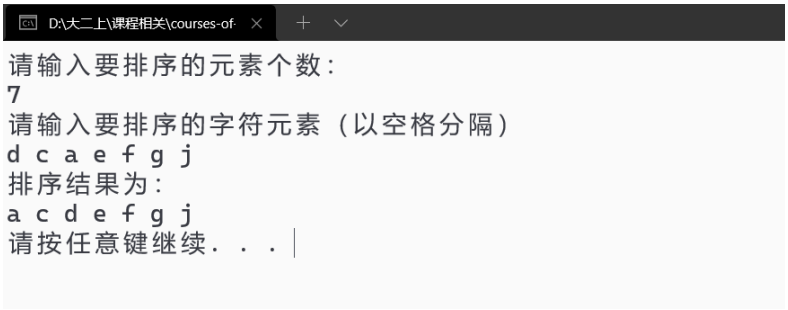
1. （必做题）快速排序

Input:

d c a e f g j

Output:

a c d e f g j

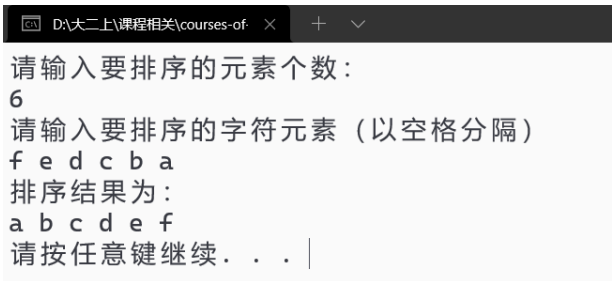


Input:

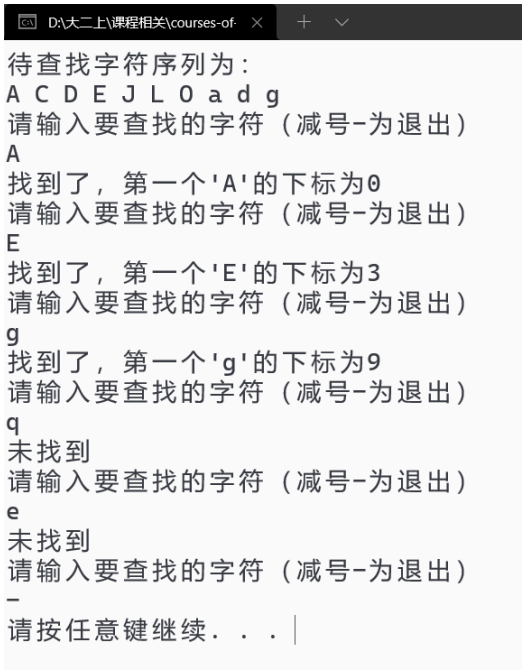
f e d c b a

Output:

a b c d e f



2. （必做题）折半查找



七、用户手册

1. （必做题）快速排序

由于是将字符型进行排序，输入时各个字符之间需要用空格分隔。

2. （必做题）折半查找

此处为了保证被查找的序列是有序的，预先给定了待查找的序列，通过不断输入单个字符进行查找。若找到，会显示下标；若没找到，会显示“未找到”。输入减号“-”以退出查找。