班级：20计算机科学与技术(4) 学号：2020333503081 姓名： 陈伟剑

实验一 递归与分治

**一、实验目的**

1、理解分治算法的概念和基本要素；

2、理解递归的概念；

3、掌握设计有效算法的分治策略。

**二、实验内容和要求**

**实验要求**：通过上机实验进行算法实现，保存和打印出程序的运行结果，并结合程序进行分析，上交实验报告和程序文件。

**实验内容**：

1、归并排序问题：对*n*个元素组成的序列进行排序。

将待排序元素分成大小大致相同的两个子集合，分别对两个集合进行排序，最终将排序好的子集合合并成所要求的排好序的集合。

2、使用二分搜索算法查找任意*n*个有序数列中的指定元素。至少使用两种方法进行编程。

**三、算法思想分析**

①归并排序

先通过递归过程自顶向下将待排序数组arr[n]拆分成n个数，然后自底向上地两两合并拆分的数

②二分查找

递归和非递归的整体思想相同。先给定数组的查找范围[left,right]，计算两者的中值mid=(left+right)>>1，由于数组已经有序，假如下标为中值处的值小于目标值x，说明x在查找范围的右半部分，反之在左半部分。递归通过调用自身实现缩小查找范围，非递归利用left，right两个指针通过循环的方式缩小查找范围。

**四、程序代码**

①归并排序

**#include**<iostream>

**#include**<algorithm>

**#include**<vector>

**using** **namespace** std;

//合并操作

**void** mergeArr(vector<**int**>**&**arr, **int** left, **int** mid, **int** right){

    vector**<int>**temp;

**int** l**=**left,r**=**mid**+**1;

**while**(l**<=**mid **&&** r**<=**right){

**if**(arr**[**l**]<=**arr**[**r**]**){

            temp.push\_back(arr**[**l**]**);

            l**++**;

        }

**else** {

            temp.push\_back(arr**[**r**]**);

            r**++**;

        }

    }

**while**(l**<=**mid)  {temp.push\_back(arr**[**l**]**); l**++**;}

**while**(r**<=**right) {temp.push\_back(arr**[**r**]**); r**++**;}

**for**(**int** i**=**0;i**<=**right**-**left;i**++**){

        //cout<<temp[i]<<" ";

        arr**[**left**+**i**]=**temp**[**i**]**;

    }

    //cout<<endl;

}

// 归并排序

**void** mergeSort(vector<**int**>**&**arr,**int** left,**int** right){

**if**(left**==**right) **return** ;

**int** mid**=**(left**+**right)**>>**1;

    mergeSort(arr,mid**+**1,right);

    mergeSort(arr,left,mid);

    mergeArr(arr,left,mid,right);

}

/\*输入格式：

    数组元素个数

    数组元素

\*/

**int** main(){

**int** x,n,index**=**0;

    vector**<int>**arr;  // arr[0]不存放数据

    cin**>>**n;

**for**(**int** i**=**0;i**<**n;i**++**){

        cin**>>**x;

        arr.push\_back(x);

    }

    mergeSort(arr,0,n**-**1);

**for**(**auto&** num:arr){

        cout**<<**num**<<**" ";

    }

**return** 0;

}

②二分查找

**#include**<iostream>

**#include**<algorithm>

**#include**<vector>

**using** **namespace** std;

// 递归方法

**int** BS1(vector<**int**>**&** arr, **int** left, **int** right, **int** x){

**int** mid**=**(left**+**right)**>>**1;

**if**(left**>**right) **return** 0;

**if**(arr**[**mid**]==**x) **return** mid;  //找到就直接返回下标

**else** **if**(arr**[**mid**]>**x) **return** BS1(arr,left,mid**-**1,x);  //x在左半部分

**else** **return** BS1(arr,mid**+**1,right,x);    //x在右半部分

**return** 0;

}

//非递归方法

**int** BS2(vector<**int**>**&** arr, **int** left, **int** right, **int** x){

**while**(left**<=**right){

**int** mid**=**(left**+**right)**>>**1;

**if**(arr**[**mid**]==**x) **return** mid;

**else** **if**(arr**[**mid**]>**x) right**=**mid**-**1;

**else** left**=**mid**+**1;

    }

**return** 0;

}

/\*输入格式：

    数组元素个数

    数组具体元素

    要查找的数

\*/

**int** main(){

**int** x,n,index**=**0;

    vector**<int>**arr(2000,0);  // arr[0]不存放数据

    cin**>>**n;

**for**(**int** i**=**1;i**<=**n;i**++**){

        cin**>>**x;

        arr**[**i**]=**x;

    }

    //要查询的数

**while**(1){

        cin**>>**x;

        index**=**BS1(arr,1,n,x);

**if**(index**>**0 **&&** index**<=**n **&&** arr**[**index**]==**x) cout**<<**"BS1: index="**<<**index**<<**endl;

**else** cout**<<**"BS1: null"**<<**endl;

        index**=**BS1(arr,1,n,x);

**if**(index**>**0 **&&** index**<=**n **&&** arr**[**index**]==**x) cout**<<**"BS2: index="**<<**index**<<**endl;

**else** cout**<<**"BS2: null"**<<**endl;

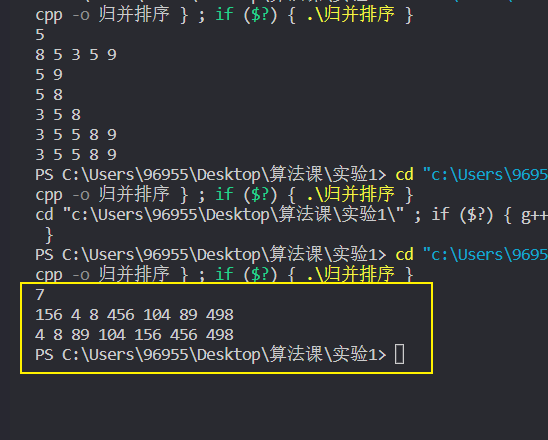
    }

**return** 0;

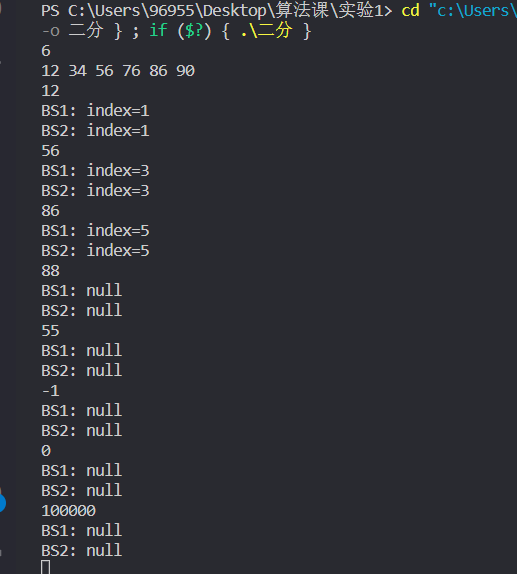
}

**五、结果运行与分析**

①归并排序



②二分查找



**六、心得与体会**

①熟悉了递归算法的大体设计逻辑和实现。

②通过不同的二分查找实现方法更加熟悉了分治算法的思想。

③数据量过大时，堆栈中的递归过程会非常多，很容易消耗空间，所以应当尽可能考虑将相关递归算法转换成非递归形式。

④二分查找的非递归方式其实有多种写法，不过不同写法返回值的意义可能会不同，最好熟练书写其中一种二分查找写法，其他方法了解即可，以免造成混淆。