查找及应用

一 目的

1、掌握查找的基本概念和基本思想；

2、掌握顺序查找和折半查找方法；

3、掌握动态查找及实现方法；

4、理解静态查找与动态查找的思想及应用。

二 内容

1、实现链表上的顺序查找

设有 n 个关键字序列 k={ a1 , a2, …, an }，使用链表作为存储结构，用顺序查找法在序列中查找 key= a2 和 key＝b 的数据元素(b 不存在于 k 中)，如果查找成功，则返回该元素，并在链表中删除它，如果查找不成功，则在链表结尾处插入它作为新的数据元素。

2、娱乐活动中经常要猜商品的价格，请设计高效算法快速猜出商品的价格。

3*、实现基于链式存储的二叉排序树的查找。

三 设计说明

查找就是由一组数据记录成的集合中主动寻找关键字值等于给定值的某个记录，或者是寻找属性值符合特定条件的某些记录。本次实验主要学会静态查找中的顺序查找（题 1）和二分查找（题 2）。

四 功能说明

二分查找中public String guessPrice(int lowPrice, int heightPrice, int price) 判断是否在范围中，不在就不执行后面了，避免死循环； while (guessPrice != price) 判断猜值大小；public int binarySearchRecursively(int low,int high,Comparable key)二分法查找的递归实现；

五 调试分析

1.2 题对比发现，顺序查找虽然操作简单但是不适用于表长过长的查找，例如题 2 经思考采用二分法即折半法，通过 guessPrice(int lowPrice, int heightPrice, int price) 判断是否在范围中，不在就不执行后面了，避免死循环大大缩减了算法的时间复杂度

六 测试结果

```
请输入商品的价格,注意输入一个整数哦!
56
请输入要猜的价格范围，以-分割
0-100
共猜了4次，猜到了价格是：56元。

请输入待排序数组中的第14个元素
7
请输入待排序数组中的第15个元素
7
请输入待排序数组中的第16个元素
8
请输入待排序数组中的第17个元素
8
请输入待排序数组中的第18个元素
9
请输入待排序数组中的第19个元素
6
请输入待排序数组中的第20个元素
5
排序前： 1 2 4 33 45 36 3 55 44 66 77 77 678 47 57 68 78 79 76 65

排序后：冒泡排序:
1 2 4 33 45 36 3 55 44 66 77 77 678 47 57 68 78 79 76 65
```

七 带注释的源程序

```
package sj;
import java.util.Scanner;
public class guess {
    public static void main(String[] args) {
        guess algorithmOne = new guess();

        System.out.println("请输入商品的价格,注意输入一个整数哦！");

        Scanner scanner = new Scanner(System.in);
        int price = scanner.nextInt();

        System.out.println("请输入要猜的价格范围，以-分割");

        String range = scanner.next();
        String nums[] = range.split("-");

        //校验分割后的结果只允许为两个值，否则格式不通过

        if (nums.length != 2) {

            System.out.println("价格范围输入错误");

        }
```

```
        //比较左右两个值，哪个大哪个为 heightPrice，哪个小哪个为 lowPrice
        else {
            int num1 = Integer.parseInt(nums[0]);
            int num2 = Integer.parseInt(nums[1]);
            if (num1 > num2) {
                System.out.println(algorithmOne.guessPrice(num2, num1,
price));
            } else {
                System.out.println(algorithmOne.guessPrice(num1, num2,
price));
            }
        }
    }

    /**
     * 采用二分法
     **/
    public String guessPrice(int lowPrice, int heightPrice, int price) {
        // 判断是否在范围中，不在就不执行后面了，避免死循环
        if (price > heightPrice || price < lowPrice) {
            return "实物价格不在给定范围中";
        } else {
            int height = heightPrice;
            int low = lowPrice;
            int guessPrice = (low + heightPrice) / 2;
            //记录一下执行次数
            int i = 0;
            while (guessPrice != price) {
                // 如果猜大了，那么最大值就是猜的值
                if (guessPrice > price) {
                    height = guessPrice;
                }
                // 如果猜小了，那么最小值就是猜的值
                else if (guessPrice < price) {
                    low = guessPrice;
                }
                guessPrice = (low + height) / 2;
```

```
                i++;
            }
            return "共猜了" + (i + 1) + "次，猜到了价格是：" + guessPrice + "
元。";
        }

    }
}
public int binarySearch(Comparable key) {
    if(length()>0) {
        int low = 0,high = length()-1;
        while(low<=high) {
            int mid = (low+high)/2;
            if(r[mid].key.compareTo(key)==0) {
                return mid;
            }else if(r[mid].key.compareTo(key)>0) {
                high = mid - 1;

            }else {
                low = mid+1;
            }
        }
    }
    return -1;
}
public int binarySearch2(Comparable key) {
    if(length() > 0) {
        int low = 0,high = length() - 1;
        while(low <= high) {
            int mid = (low + high)/2;
        if(key.compareTo(r[mid].key) == 0) {
            System.out.println(key + "的下标值为：" + mid);
            return mid;
        }else if(key.compareTo(r[mid].key) < 0) {
            high = mid - 1;
        }else {
            low = mid + 1;
        }
        }
    }
    return -1;
}
```

```
//二分法查找的递归实现
public int binarySearchRecursively(int low,int high,Comparable key) {
    int mid,result;
    if(low <= high) {
        mid = (low + high)/2;
        if(key.compareTo(r[mid].key) == 0) {
            System.out.println(key + "的下标值为：");
            return mid;
        }else if(key.compareTo(r[mid].key) < 0){
            return binarySearchRecursively(low,mid - 1,key);
        }else {
            return binarySearchRecursively(mid + 1,high,key);
        }
    }
    return -1;
}
package sj;
import java.util.Scanner;
public class Example {


        static SeqList L = null;
        public static void main(String[] args) throws Exception {
            Scanner input = new Scanner(System.in);
            System.out.println("请输入你要建立的顺序表地长度:");
            int maxSize = input.nextInt(); //顺序表空间大小
            KeyType[] d = new KeyType[maxSize];
            for(int i = 0;i < maxSize;i++) {
                System.out.println("请输入待排序数组中的第" + (i+1) + "个元素
");
                d[i] = new KeyType(input.nextInt());
            }

           L = new SeqList(maxSize);       //建立顺序表
            for (int i = 0; i < d.length; i++) {
                RecordNode r = new RecordNode(d[i]);
                L.insert(L.length(), r);
            }

            System.out.print("排序前：");
            L.display();
            System.out.println("");
            System.out.print("排序后：");
            L.bubbleSort();
```

```
            L.display();
            while(true) {
            System.out.println();
                System.out.println("请输入你要查找的关键字: ");
                KeyType key1 = new KeyType(input.nextInt());
                KeyType key2 = new KeyType(input.nextInt());
                L.binarySearch2(key1);
                if(L.binarySearch(key1) == -1) {
                System.out.println("没有找到" + key1 );
                }
                if(L.binarySearchRecursively(0,L.length() - 1,key2) == -1) {
                System.out.println("没有找到" + key2 );
                }
                System.out.println("------------------");
            }

        }


    }
```