1. 排序问题

给一个数组，如A[]={1，3，5，46，11，22，37，45，44，78，67，98}，用除了冒泡排序外的任意一种方法把此数组按降序排列。

思考题：超级水王问题

题目来源：

Tango是微软亚洲研究院的一个试验项目。研究院的员工和实习生们都很喜欢在Tango上面交流灌水。传说，Tango有一大“水王”，他不但喜欢发贴，还会回复其他ID发的每个帖子。坊间风闻该“水王”发帖数目超过了帖子总数的一半。如果你有一个当前论坛上所有帖子（包括回帖）的列表，其中帖子作者的ID也在表中，你能快速找出这个传说中的Tango水王吗？

题目描述：

给你一个数组，需要你确定其中元素的水王，水王的条件为出现的频率超过总数的一半。

如B[]= { 1, 1, 2, 3, 1, 1 };其中水王为“1”

C[]= { 1, 1, 1, 2, 4, 2, 2, 3, 3, 3, 5 } 无水王返回“-1”

示例：求D[]= { 1, 1, 1, 2, 4, 2, 2, 3, 3, 3, 5,2,3,2,2,2,2,2,2 }

package videocode;

import java.util.HashMap;

import java.util.Map.Entry;

/\*

\* 给定一个整型数组arr，长度为N

\* 如果存在某个数，其出现次数大于N/2，返回这个数

\* 如果不存在这样的数，返回-1

\*

\* 要求：时间复杂度O(N)，额外空间复杂度O(1)

\*

\* \*/

public class Code01\_SuperWaterKing {

// 用哈希表对出现的数做词频统计

// 最后遍历哈希表，自然知道有没有水王数，水王数是谁

// 该方法不符合题目的限制，因为使用了哈希表，所以额外空间复杂度O(N)

// 但是该方法功能正确，仅作为对数器使用，用于验证下面的waterKing方法

public static int verify(int[] arr) {

if (arr == null || arr.length == 0) {

return -1;

}

HashMap<Integer, Integer> map = new HashMap<>();

for (int num : arr) {

if (map.containsKey(num)) {

map.put(num, map.get(num) + 1);

} else {

map.put(num, 1);

}

}

int N = arr.length;

for (Entry<Integer, Integer> record : map.entrySet()) {

if (record.getValue() > (N >> 1)) {

return record.getKey();

}

}

return -1;

}

// 真正想实现的方法

public static int waterKing(int[] arr) {

if (arr == null || arr.length == 0) {

return -1;

}

int candidate = 0;

int restHP = 0;

for (int cur : arr) {

if (restHP == 0) { // 如果没有候选

candidate = cur;

restHP = 1;

} else if (cur != candidate) { // 如果有候选，并且当前的数字和候选不一样

restHP--;

} else { // 如果有候选，并且当前的数字和候选一样

restHP++;

}

}

// 如果遍历完成后，没有候选留下来，说明没有水王数

if (restHP == 0) {

return -1;

}

// 如果有候选留下来，再去遍历一遍，得到候选真正出现的次数

int count = 0;

for (int num : arr) {

if (num == candidate) {

count++;

}

}

int N = arr.length;

// 如果候选真正出现的次数大于N/2，返回候选

// 否则返回-1代表没有水王数

return count > (N >> 1) ? candidate : -1;

}

public static int[] randomArray(int maxLen, int range) {

// 随机一个非负数，作为测试数组的长度

int len = (int) (Math.random() \* (maxLen + 1));

int[] arr = new int[len];

if (len == 0) {

return arr;

}

// 50%的概率决定：随机选择一个数，安排其成为测试数据中的水王数

// 剩下50%的概率：不人为干预，完全随机的生成测试数据

boolean hasWaterKing = Math.random() < 0.5;

if (hasWaterKing) { // 人为干预，一定在测试数据中存在某个数是水王数

int waterKing = randomNumber(range);

int index = 0;

// 如果数组长度为N，往数组中人为放入>N/2个水王数

for (; index < ((len + 2) >> 1); index++) {

arr[index] = waterKing;

}

// 数组剩下的位置，填入随机数

for (; index < len; index++) {

arr[index] = randomNumber(range);

}

// 把数字的出现规律打乱

for (index = len - 1; index >= 0; index--) {

// 随机选出一个[0,index]范围上的位置

int randomIndex = (int) (Math.random() \* (index + 1));

// 把inde位置和randomIndex位置上的数字做交换

int tmp = arr[index];

arr[index] = arr[randomIndex];

arr[randomIndex] = tmp;

}

} else { // 不人为干预，完全随机的生成测试数据

for (int i = 0; i < arr.length; i++) {

arr[i] = randomNumber(range);

}

}

return arr;

}

// 如果输入参数是5

// 那么将返回-5~+5范围上的数

// 但在这个范围上返回的数，并不是等概率的

// 因为等不等概率对这个测试来说不是特别重要

public static int randomNumber(int range) {

return (int) (Math.random() \* (range + 1)) - (int) (Math.random() \* (range + 1));

}

// 用两种方法来做验证

// 如果verify方法和waterKing方法每次返回的结果一样

// 说明两个方法都正确

public static void main(String[] args) {

int maxLen = 1000;

int maxValue = 5;

int testTime = 100000;

System.out.println("如果没有打印错误，说明验证通过");

System.out.println("测试开始");

for (int i = 0; i < testTime; i++) {

int[] arr = randomArray(maxLen, maxValue);

int ans1 = verify(arr);

int ans2 = waterKing(arr);

if (ans1 != ans2) {

System.out.println(ans1);

System.out.println("出错了！");

}

}

System.out.println("测试结束");

}

}