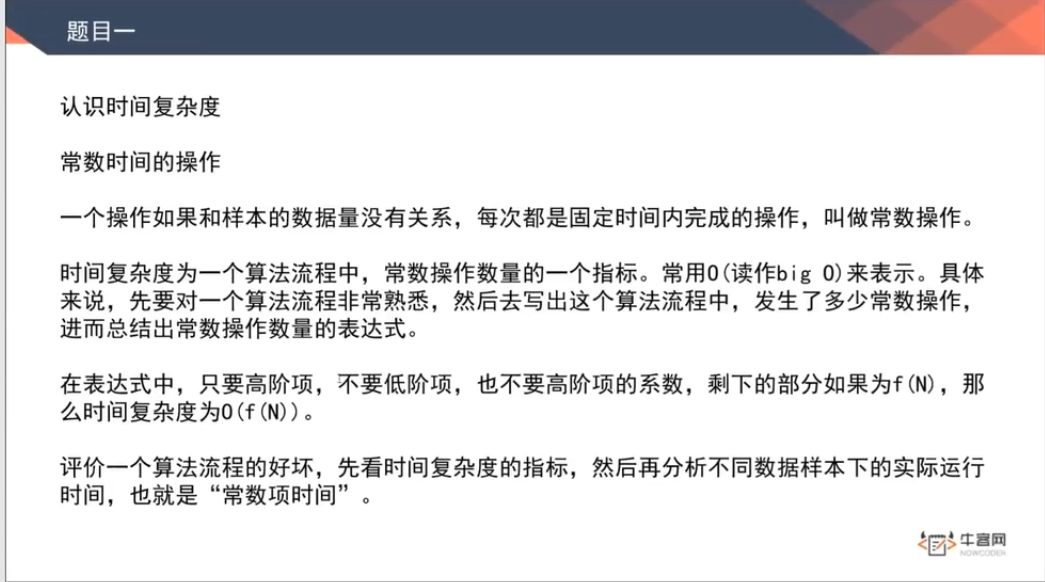
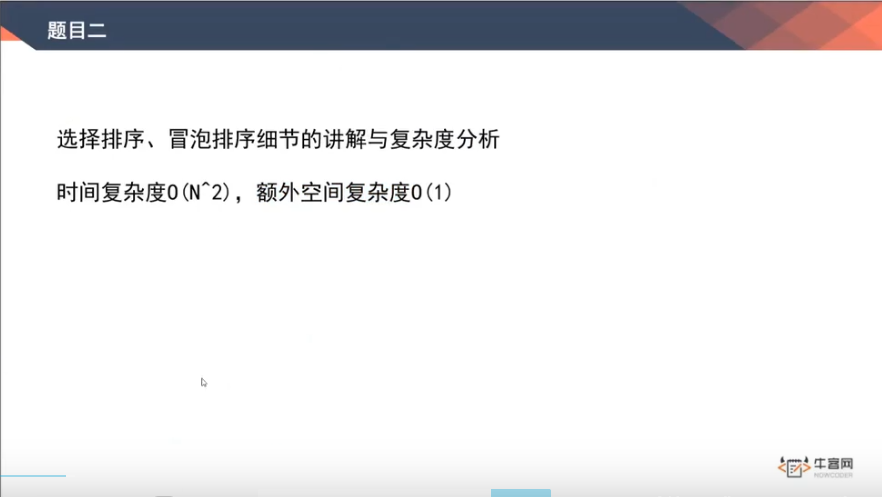
题目1



时间复杂度分析

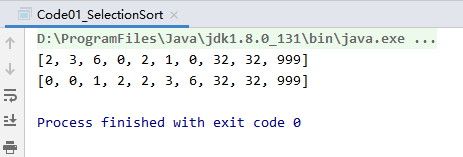
题目2



【1】简单选择排序

**package** com.ac.p1;  
  
**import** java.util.Arrays;  
  
*/\*\*  
 \* 选择排序  
 \*/***public class** Code01\_SelectionSort {  
  
 **public static void** selectionSort(**int** arr[]) {  
 **if** (arr == **null** || arr.**length** < 2)  
 **return**;  
 **for** (**int** i = 0; i < arr.**length** - 1; i++) {  
 **int** minIndex = i; *//最小值下标* **for** (**int** j = i; j < arr.**length** - 1; j++) {  
 minIndex = arr[j] < arr[minIndex] ? j : minIndex;  
 }  
 *swap*(arr, i, minIndex);  
 }  
 }  
  
 **public static void** swap(**int** arr[], **int** i, **int** j) {  
 **int** tmp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = tmp;  
 }  
  
 *//测试代码* **public static void** main(String[] args) {  
 **int**[] arr = {2, 3, 6, 0, 2, 1, 0, 32, 32, 999};  
 System.***out***.println(Arrays.*toString*(arr));  
 *selectionSort*(arr);  
 System.***out***.println(Arrays.*toString*(arr));  
 }  
  
}

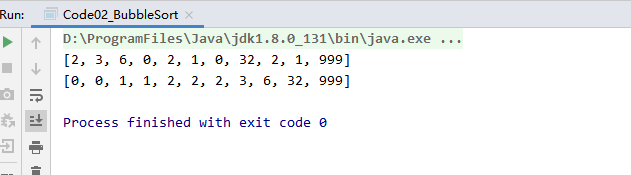
运行结果：



【2】简单冒泡排序

**package** com.ac.p1;  
  
**import** java.util.Arrays;  
  
*/\*\*  
 \* 冒泡排序  
 \*/***public class** Code02\_BubbleSort {  
 **public static void** bubbleSort(**int** arr[]) {  
 **if** (arr == **null** || arr.**length** < 2)  
 **return**;  
*/\*  
 for (int i = 0; i < arr.length - 1; i++) {  
 boolean flag = false; //判断某一次是否发生交换，如果没有交换则结束排序  
 for (int j = 0; j < arr.length - i - 1; j++) {  
 if (arr[j] > arr[j + 1]) {  
 swap(arr, j, j + 1);  
 flag = true;  
 }  
 }  
 if (!flag)  
 return;  
 }  
\*/  
 //倒着写会更好理解* **for** (**int** e = arr.**length** - 1; e > 0; e--) {  
 **boolean** flag = **false**;  
 **for** (**int** j = 0; j < e; j++) {  
 **if** (arr[j] > arr[j + 1]) {  
 *swap*(arr, j, j + 1);  
 flag = **true**;  
 }  
 }  
 **if** (!flag)  
 **return**;  
 }  
  
 }  
  
 **public static void** swap(**int** arr[], **int** i, **int** j) {  
 **int** tmp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = tmp;  
 }  
  
 *//测试代码* **public static void** main(String[] args) {  
 **int**[] arr = {2, 3, 6, 0, 2, 1, 0, 32, 2, 1, 999};  
 System.***out***.println(Arrays.*toString*(arr));  
 *bubbleSort*(arr);  
 System.***out***.println(Arrays.*toString*(arr));  
 }  
  
}

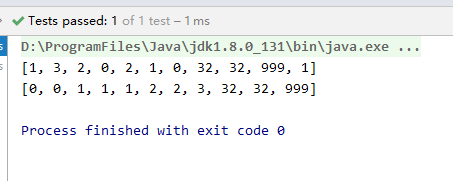
运行结果：



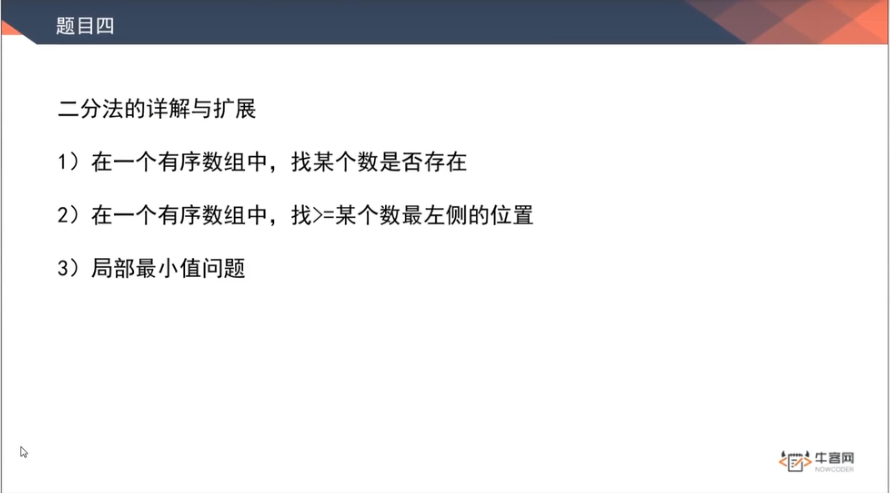
【3】简单插入排序

**package** com.ac.p1;  
  
**import** java.util.Arrays;  
  
*/\*\*  
 \* 简单排序：选择排序  
 \*/***public class** Code01\_SelectionSort {  
  
 **public static void** selectionSort(**int** arr[]) {  
 **if** (arr == **null** || arr.**length** < 2)  
 **return**;  
 **for** (**int** i = 0; i < arr.**length**; i++) {  
 **int** minIndex = i; *//最小值下标* **for** (**int** j = i; j < arr.**length**; j++) {  
 minIndex = arr[j] < arr[minIndex] ? j : minIndex;  
 }  
 *swap*(arr, i, minIndex);  
 }  
 }  
  
 **public static void** swap(**int** arr[], **int** i, **int** j) {  
 **int** tmp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = tmp;  
 }  
  
 *//测试代码* **public static void** main(String[] args) {  
 **int**[] arr = {2, 3, 6, 0, 2, 1, 0, 32, 32, 999,1};  
 System.***out***.println(Arrays.*toString*(arr));  
 *selectionSort*(arr);  
 System.***out***.println(Arrays.*toString*(arr));  
 }  
  
}

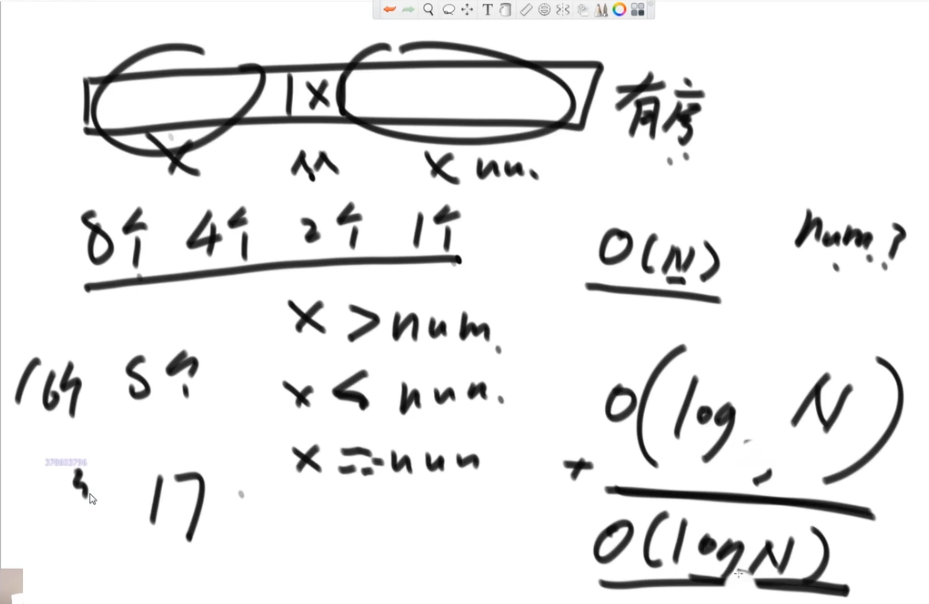
运行结果：



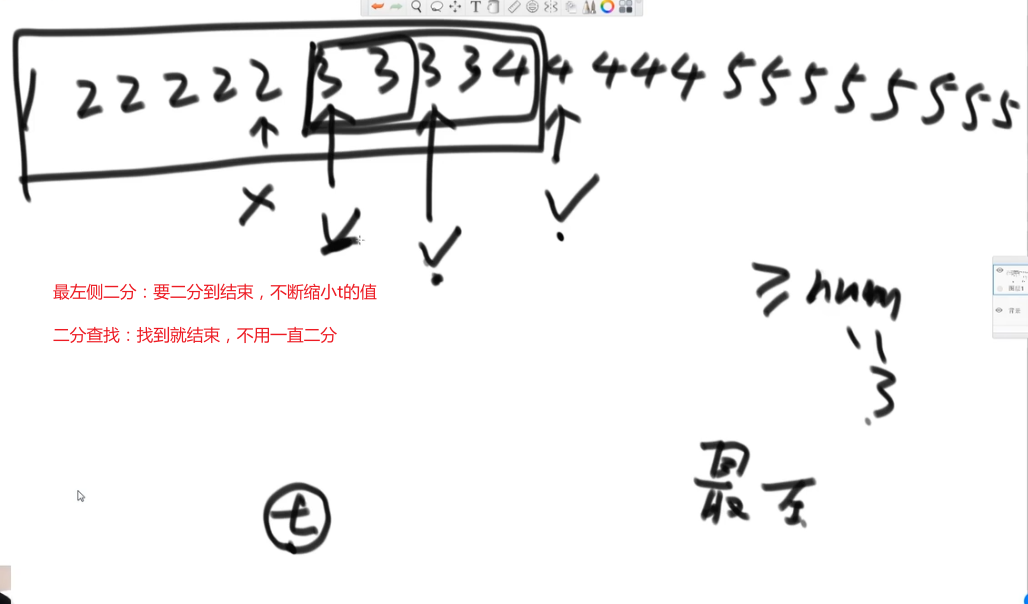
题目4



【1】二分查找 O(logn)



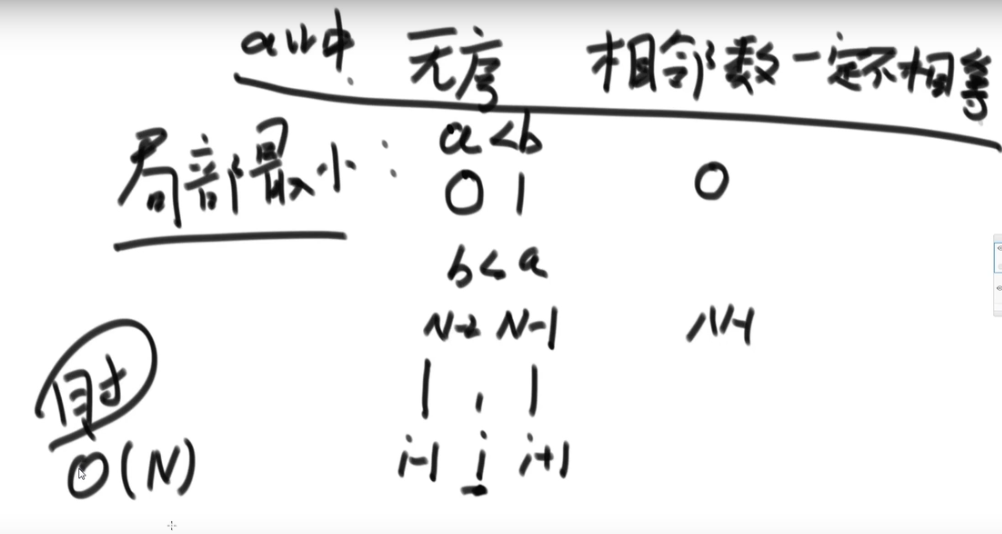
【2】最左侧二分

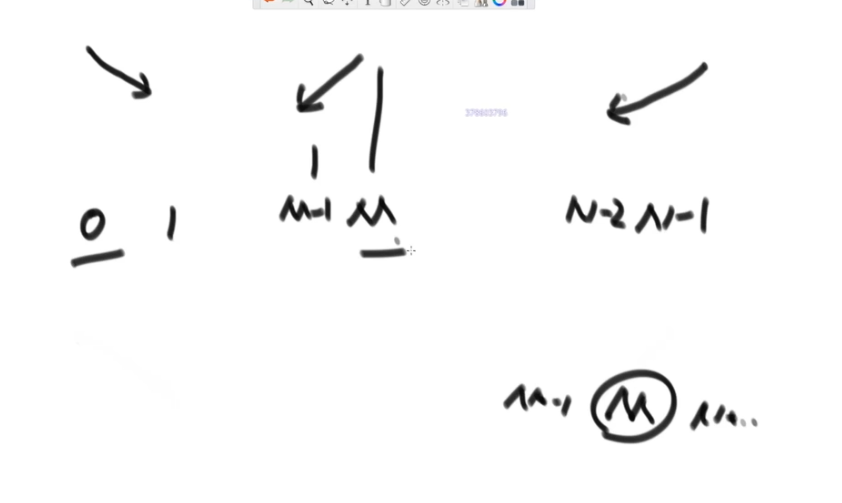


【3】局部最小值问题\*

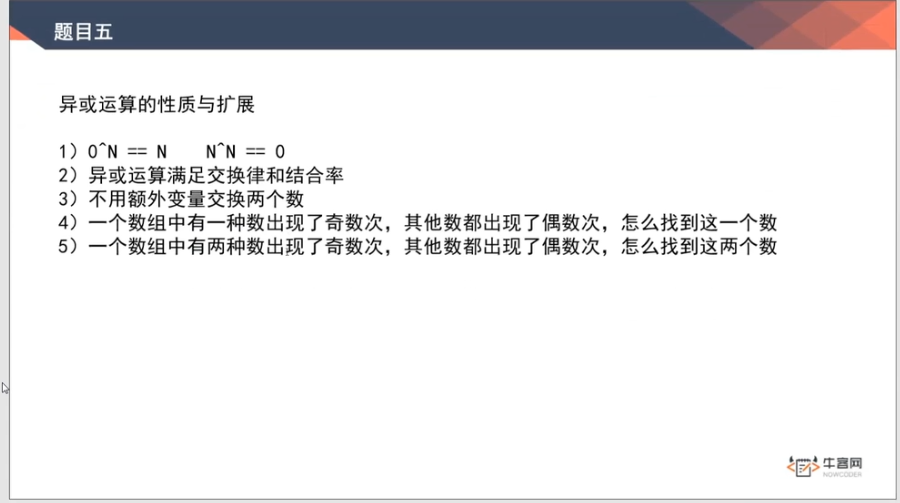
arr无序，相邻数一定不相等。

局部最小





题目5



【1】异或运算性质

不同为1，相同为0。

10001^11111 = 01110

还可以理解为无进位相加。

运算性质：

* 0^N=N 例如：00000 ^ 11111 = 11111
* N^N=0 例如：11111 ^ 11111 = 00000
* 交换律、结合律：a^b=b^a a^b^c = a^(b^c)
* 同样一批数，选择谁先谁后异或结果是一样的。

【2】使用异或运算交换两个数

*//异或运算的交换***public static void** swap\_(**int** arr[], **int** i, **int** j) {  
 arr[i] = arr[i] ^ arr[j];  
 arr[j] = arr[i] ^ arr[j];  
 arr[i] = arr[i] ^ arr[j];  
}

原理：

令X=arr[i] ，Y=arr[j]

X=X^Y

Y=(X^Y)^Y 使用分配律：Y=X^(Y^Y) Y=X^0 Y=X

X=(X^Y)^X 使用交换律：X=(X^X)^Y X=0^Y X=Y

交互完毕

注意：

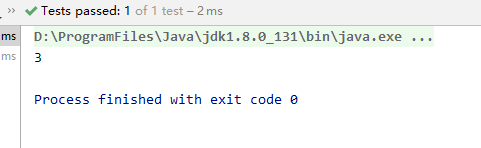
a和b是必须是两块不同的内存。

如果位置一样，会被抹成0。（自己和自己异或）

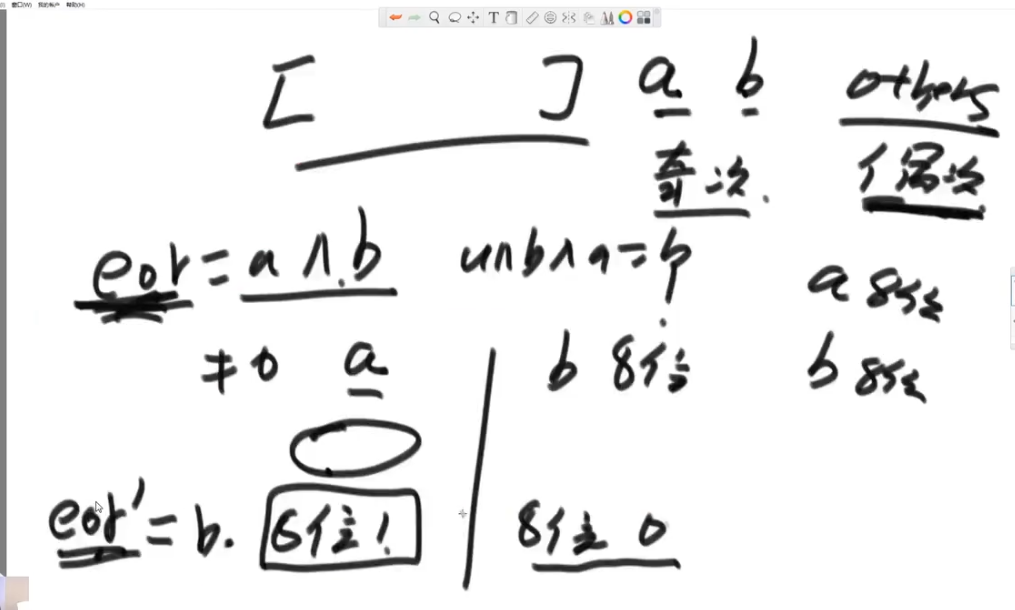
【3】例题1

*/\*\*  
 \* 在一个数组中，一种树出现了奇数次，其他数出现了偶数次，求这个出现了奇数次的数  
 \*/* **public static void** printOddTimesNum1(**int** arr[]) {  
 **int** eor = 0;  
*// for (int i = 0; i < arr.length; i++) {  
// eor = eor ^ arr[i];  
// }* **for** (**int** cur : arr) {  
 eor ^= cur;  
 }  
 System.***out***.println(eor);  
 }  
  
 @Test  
 **public void** test\_f1() {  
 **int**[] arr = {2, 1, 3, 1, 3, 1, 3, 2, 1};  
 *printOddTimesNum1*(arr);  
 }

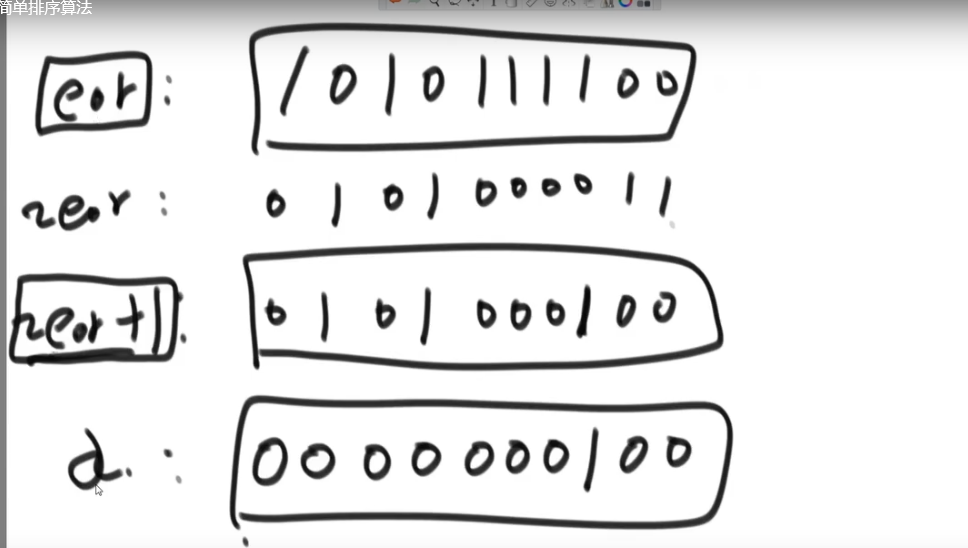
运行结果：



【4】例题2

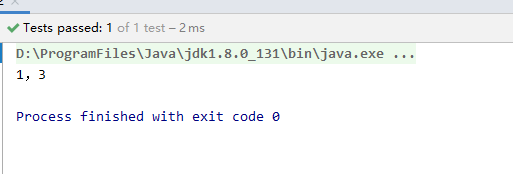


不为0的数，提取最右侧的1：**int rightOne = eor & (~eor + 1)**

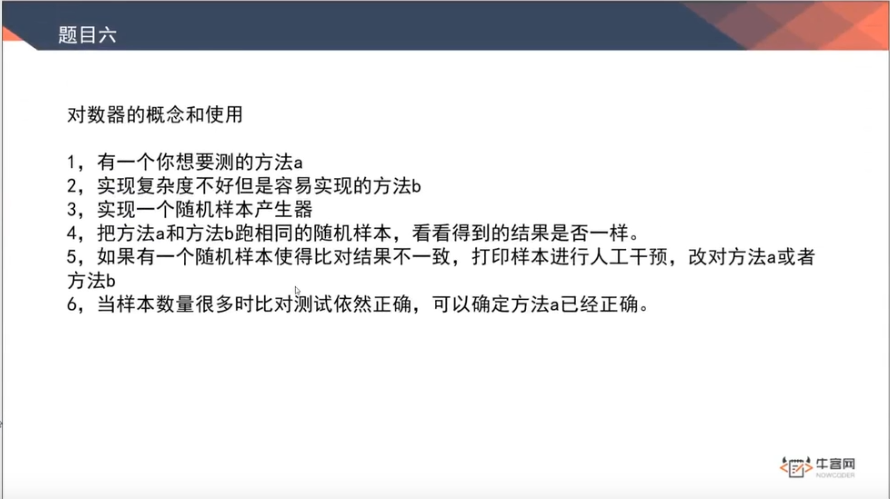


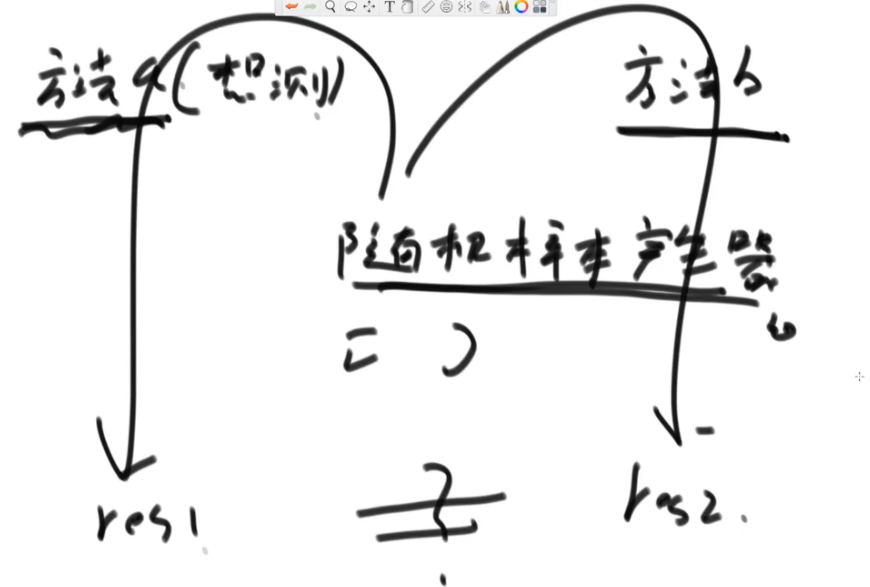
*/\*\*  
 \* 两种数出现了奇数次，其他数出现了偶数次  
 \*/***public static void** printOddTimesNum2(**int** arr[]) {  
 *//设两种奇数次的数为 a 和 b* **int** eor = 0;  
 **for** (**int** cur : arr) {  
 eor ^= cur;  
 }*//eor=a^b  
  
 //取eor最右侧为1的数（固定套路）* **int** rightOne = eor & (~eor + 1);  
  
 **int** onlyOne = 0; *//eor'* **for** (**int** cur : arr) {  
 */\*  
 此处判断可以是：等于rightOne（筛出第i位为1的） ，等于0（筛出第i位为0的），  
 大于等于1（筛出第i位为1的）。就是不能等于1  
 \*/* **if** ((cur & rightOne) == 0) {  
 onlyOne ^= cur;  
 }  
 }*//onlyOne = a 或者 b* System.***out***.println(onlyOne + **"，"** + (eor ^ onlyOne)); *//eor^onlyOne 就可以得到另一个数*}  
  
@Test  
**public void** test\_printOddTimesNum2() {  
 **int**[] arr = {2, 1, 3, 1, 3, 1, 3, 2, 1, 1};  
 *printOddTimesNum2*(arr);  
}

运行结果：



题目6



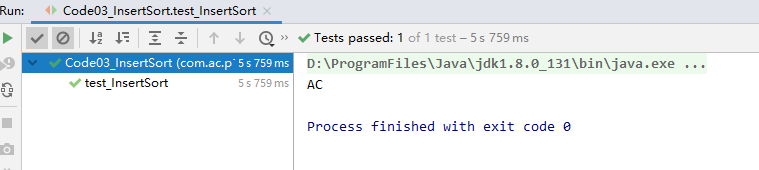


【1】对数器实现类

**package** com.ac.util;  
  
**import** java.util.Arrays;  
  
*/\*\*  
 \* 对数器：  
 \* 随机产生数组  
 \* 检查结果正确性  
 \*/***public class** GA {  
 **private long testTime**;  
 **private long maxSize**;  
 **private long maxValue**;  
  
 **public** GA(**long** maxSize, **long** maxValue) {  
 **this**.**maxSize** = maxSize;  
 **this**.**maxValue** = maxValue;  
 }  
  
 **public int**[] generateRandomIntegerArray() {  
 **int**[] arr = **new int**[(**int**) ((**this**.**maxSize** + 1) \* Math.*random*())]; *//长度随机* **for** (**int** i = 0; i < arr.**length**; i++) {  
 arr[i] = (**int**) ((**maxValue** + 1) \* Math.*random*()) - (**int**) ((**maxValue**) \* Math.*random*());  
 }  
 **return** arr;  
 }  
  
 **public boolean** resultAnalysis(**int** arr1[], **int** arr2[]) {  
 **if** (arr1.**length** != arr2.**length**)  
 **return false**;  
 **for** (**int** i = 0; i < arr1.**length**; i++) {  
 **if** (arr1[i] != arr1[i])  
 **return false**;  
 }  
 **return true**;  
 }  
  
 **public int**[] copy(**int** arr[]) {  
 **if** (arr == **null**)  
 **return null**;  
 **int**[] tmp = Arrays.*copyOf*(arr, arr.**length**);  
 **return** tmp;  
 }  
}

【2】使用对数器测试插入排序

@Test  
 **public void** test\_InsertSort() {  
*// int[] arr = {1, 3, 2, 0, 2, 1, 0, 32, 32, 999, 1};  
// System.out.println(Arrays.toString(arr));  
// InsertSort2(arr);  
// System.out.println(Arrays.toString(arr));  
  
 //使用对数器* GA ga = **new** GA(100000, 100000);  
 **int**[] arr1 = ga.generateRandomIntegerArray();  
 **int**[] arr2 = ga.copy(arr1);  
 *//多线程测试* Thread t1 = **new** Thread(() -> {  
 *InsertSort2*(arr1);  
 }, **"t1"**);  
 t1.start();  
 Thread t2 = **new** Thread(() -> {  
 Arrays.*sort*(arr2);  
 }, **"t2"**);  
 t2.start();  
 Thread t3 = **new** Thread(() -> {  
 **try** {  
 t1.join();  
 t2.join();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.***out***.println(ga.resultAnalysis(arr1, arr2) ? **"AC"** : **"ERROR"**);  
 }, **"t3"**);  
 t3.start();  
 **try** {  
 t3.join();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }



题目7

