

Java3y Lv6

2018年04月07日 阅读 3896

[2]十道算法题【Java实现】

前言

清明不小心就拖了两天没更了~~

这是十道算法题的第二篇了~上一篇回顾: [十道简单算法题](#)

最近在回顾以前使用C写过的数据结构和算法的东西, 发现自己的算法现在用Java改写一下, 重温一下。

只能说慢慢积累吧~下面的题目难度都是简单的, 算法的大佬可直接算法薄弱的同学可参考一下~

很多与排序相关的小算法(合并数组、获取数字每位值的和), 我都没归并排序(合并数组), 会了桶排序(获取数字每位的值), 这些都不成基础排序的同学可看: [【八大基础排序总结】](#)

由于篇幅问题, 每篇写十道吧~

如果有错的地方, 或者有更好的实现, 更恰当的理解方式希望大家不多交流

关于作者

Java3y Lv6

掘金特邀作者 | ...

获得点赞 18,013

文章被阅读 694,065

掘金小册

< >

Git 原理详解及实用指南

新人价 ¥14.95 ~~¥29.9~~

用 npm script 打造超溜的前端工作流

新人价 ¥9.95 ~~¥19.9~~

新人专享好礼



送你 **45元** 买小册

[立即领取](#)

相关文章

数据库两大神器【索引和锁】

851 47

十道简单算法题

题目的总览

1. 删除下标为k的元素
2. 找出常用的数字
3. 丢失的数字
4. 将0放在数组最后
5. 找出数组的单个数字
6. 画三角形星星
7. 罗马数字倒转成阿拉伯数字
8. 啤酒与饮料
9. 简单凯撒密码
10. 求最大公约数

一、删除下标为k的元素

删除下标为k的元素

思路：数组后一位往前覆盖即可～

java 复制代码

```
/**
 * 删除下标为k的元素
 */
public static void deleteK() {

    //固定的常量(比数组元素的个数要大)
    int N = 10;
```

数据库面试题(开发者必看)

👍 983 💬 31

外行人都能看懂的SpringCloud, 错过了血亏!

👍 749 💬 52

Java集合总结【面试题+脑图】, 将知识点一网打尽!

👍 617 💬 17

我采访了同事, 让他掏出了每天都会浏览的干货网站...

👍 342 💬 65

```
int[] arrays = new int[N];

//对数组进行初始化
for (int i = 0; i < 8; i++) {
    arrays[i] = i;
}

//要删除下标
int k = 7;

for (int i = k; i < N - 1; i++) {
    arrays[i] = arrays[i + 1];
}

System.out.println("公众号: Java3y" + arrays);

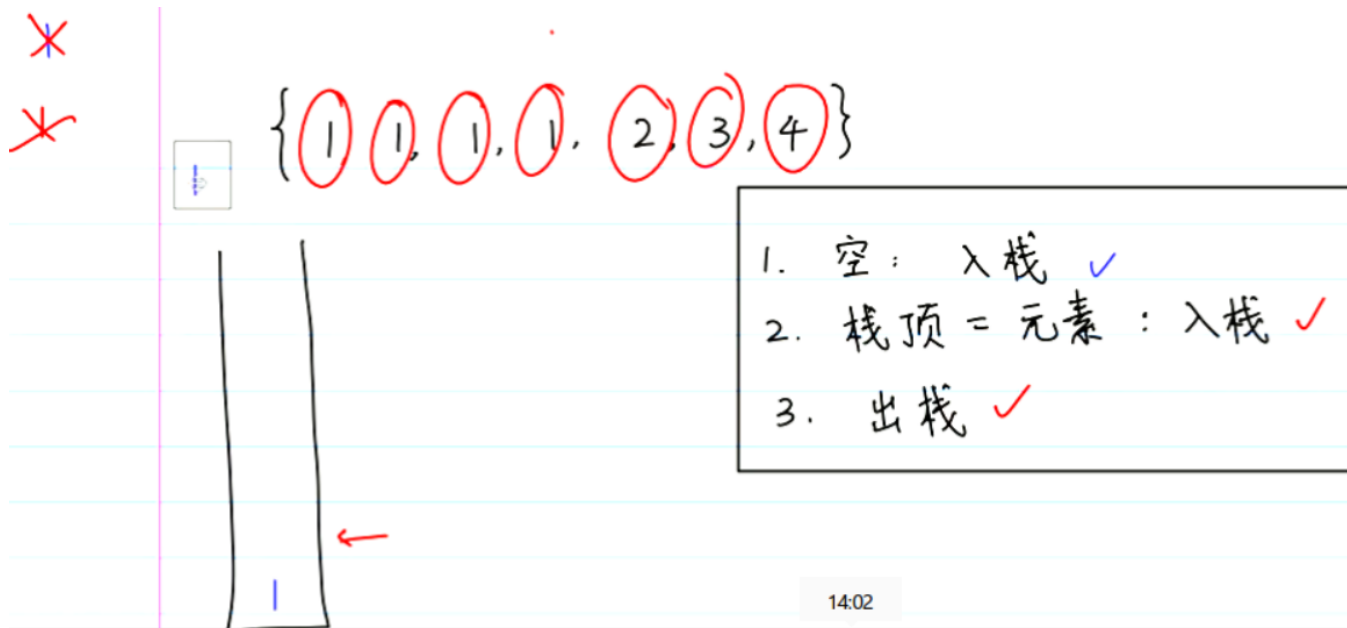
}
```

二、找出常用的数字

给你一个长度为n的数组，其中有一个数字出现的次数至少为 $n/2$ ，找出这个数字

这道题可以用栈的思想来做：

- 如果栈是空的，那么先把数据存进去
- 然后继续遍历其他的数据，只要发现栈中的数据 and 遍历中的数据不一样，那么就出栈
- 如果是相同的，那么就入栈
- 其实就是捉住数字出现的次数多于数组一半的长度这里入手。如果这个数出现的次数是大于这个数组长度的 $2/1$ ，那么最后留下的肯定就是这个数



java 复制代码

/**

* 找出常用的数字:

* 给你一个长度为n的数组，其中有一个数字出现的次数至少为n/2，找出这个数字

*/

```
public static void findMajorityElement(int[] arrays) {
```

```
//构建一个静态栈
```

```
int[] stack = new int[arrays.length];
```

```
// 栈的front指针
```

```
int front = -1;
```

```
// 遍历给出的数组
```

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

```
// 判断该栈为空，那么直接将元素入栈
```

```
if (front == -1) {
```

```
    stack[++front] = arrays[i];
```

```
} else if (stack[front] == arrays[i]) { // 该元素是否与栈的元素一致-->继续入栈
```

```
        stack[++front] = arrays[i];
    } else {
        // 只要不一致, 就出栈
        front--;
    }
}

// 只要该数字出现次数大于数组长度的2/1, 那么留下来的数字肯定在栈顶中
System.out.println("关注公众号: Java3y---->" + stack[0]);
}
```

优化:

- 其实没必要用整个栈来装载数组, 因为我们就使用栈顶元素(出现次数最多的那个), 而栈的大小也可以通过一个变量就可以来确定了
- 只要元素相同->入栈(长度+1)。元素不相同-->出栈(长度-1)
- 最终留下来的肯定是出现最频繁的那个数字!

java 复制代码

```
public static void findMajorityElement2(int[] arrays) {

    // 装载栈的元素
    int candidate = -1;

    // 栈的大小(长度)
    int count = 0;

    // 遍历给出的数组
    for (int i = 0; i < arrays.length; i++) {

        // 判断该栈为空, 那么直接将元素入栈
        if (count == 0) {
```

```
        candidate = arrays[i];
        count++;

    } else if (candidate == arrays[i]) { // 该元素是否与栈的元素一致-->入栈(栈多一个元
        count++;
    } else {
        // 只要不一致-->出栈(栈少一个元素)
        count--;
    }
}

// 只要该数字出现次数大于数组长度的2/1, 那么留下来的数字肯定在栈顶中
System.out.println("关注公众号: Java3y---->" + candidate);
}
```

三、丢失的数字

给你一个数组{0,1,2,3,...n}, 其中有一个数字缺失, 请把缺失的数字找出来

思路:

- 创建一个数组(题目数组的长度+1, 因为题目的数组缺失了一个)
- 创建的数组元素用特殊的符号(数字)来进行填满
- 将题目给出的数组遍历并填充到创建的数组上, 用index(0,1,2,3..)替代
- 最后遍历创建的数组, 哪个还是特殊的符号就是缺失的数字, 返回index(缺失的数字)即可

java 复制代码

```
/**
 * 找到缺失的数字
 *
 * @param arrays
 */
public static void missingNumber(int[] arrays) {
```

```
// 定义要填充到新数组的数字(随意)
int randomNumber = 89898980;

// 创建一个新的数组(比已缺失的数组多一个长度)
int[] newArray = new int[arrays.length + 1];

// 填充特殊的数字进新数组中
for (int i = 0; i < newArray.length; i++) {

    // 随意填充数组到新数组中
    newArray[i] = randomNumber;
}

// 遍历题目的数组并使用index替代掉新数组的元素
for (int i = 0; i < arrays.length; i++) {

    // 题目数组的值[0,1,2,3,4,...n]其中有一个缺失
    int index = arrays[i];

    // 重新填充到新数组上, index对应着题目数组的值
    newArray[index] = 3333333;

}

// 遍历新数组, 只要还有值为89898980, 那么那个就是缺失的数字
for (int i = 0; i < newArray.length; i++) {

    if (newArray[i] == randomNumber) {

        System.out.println("关注公众号: Java3y---->缺失的数字是: " + i);

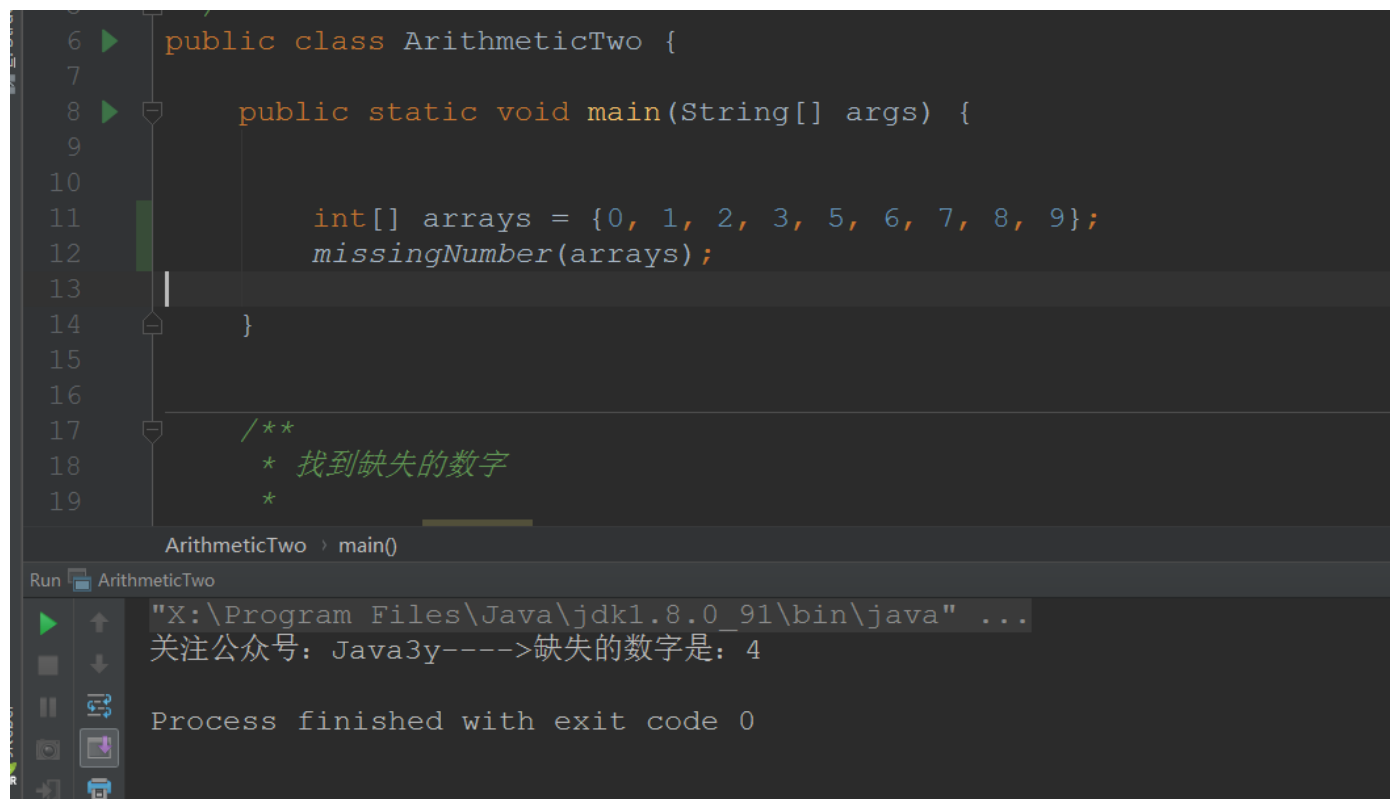
    }

}

}
```

```
}
```

结果：



```
6 public class ArithmeticTwo {
7
8     public static void main(String[] args) {
9
10
11         int[] arrays = {0, 1, 2, 3, 5, 6, 7, 8, 9};
12         missingNumber(arrays);
13
14     }
15
16
17     /**
18      * 找到缺失的数字
19      *
20
21     }
22 }
```

Run ArithmeticTwo

```
"X:\Program Files\Java\jdk1.8.0_91\bin\java" ...
关注公众号：Java3y---->缺失的数字是：4
Process finished with exit code 0
```

优化：

- 题目给出的数组 $\{0, 1, 2, 3, 4, 5, \dots, n\}$ 其中缺失一个数字，要把缺失的数字找出来...我们可以回顾一下高中学过的等差求和公式: $S_n = (a_1 + a_n)n/2$

$$S_n = \frac{(a_1 + a_n)n}{2}$$

- 假设我们没有缺失数字，等差求和公式可以快速得出答案。比如： $\{0, 1, 2, 3\}$ ---> $(0+3)*4/2$ ---> 6，如果此时缺失的是2呢，就是说题目的给出的数组是: $\{0, 1, 3\}$ ，我们利

用等差公式求和之后减去数组每个元素，最后剩下的数就是缺失的数字！ $6-1-3-0 \rightarrow 2$

所以，我们可以写出这样的代码：

java 复制代码

```
/**
 * 利用等差公式找到缺失的数字
 *
 * @param arrays
 */
public static void missingNumber2(int[] arrays) {

    // 套用等差求和公式
    int sum = (arrays[0] + arrays[arrays.length - 1]) * (arrays.length + 1) / 2;

    // 遍历数组，得出的sum减去数组每一位元素，最后即是缺失的数字

    for (int value : arrays) {
        sum = sum - value;
    }

    System.out.println("关注公众号：Java3y---->缺失的数字是: " + sum);

}
```

结果：

```
0
1      int[] arrays = {0, 1, 2, 3, 4, 5, 6, 7, 9};
2      missingNumber2(arrays);
3
4  }
5
6
7  /**
8   * 找到缺失的数字
9   *
10   * @param arrays
11   */
12  @
13  public static void missingNumber(int[] arrays) {
14
15      // 定义要填充到新数组的数字(随意)
```

四、将0放在数组最后

将一个数组的元素，其中是0的，放在数组的最后

思路：

- 使用一个变量zero来记住该数组有多少个0
- 遍历这个数组，如果发现不是0的，就往数组前面移动，如果发现是0就zero++
- 数组移动的位置刚好是 `arr[i-zero]` 的

代码实现：

java 复制代码

```
/**
 * 移动元素0到数组最后
 *
 * @param arrays
 */
public static void moveZero(int[] arrays) {

    // 记录该数组有多少个0元素
    int zero = 0;

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

        // 只要元素不为0，那么就往前面移动
        if (arrays[i] != 0) {
            arrays[i - zero] = arrays[i];
        } else {
            // 如果为0，那么zero ++
            zero++;
        }
    }

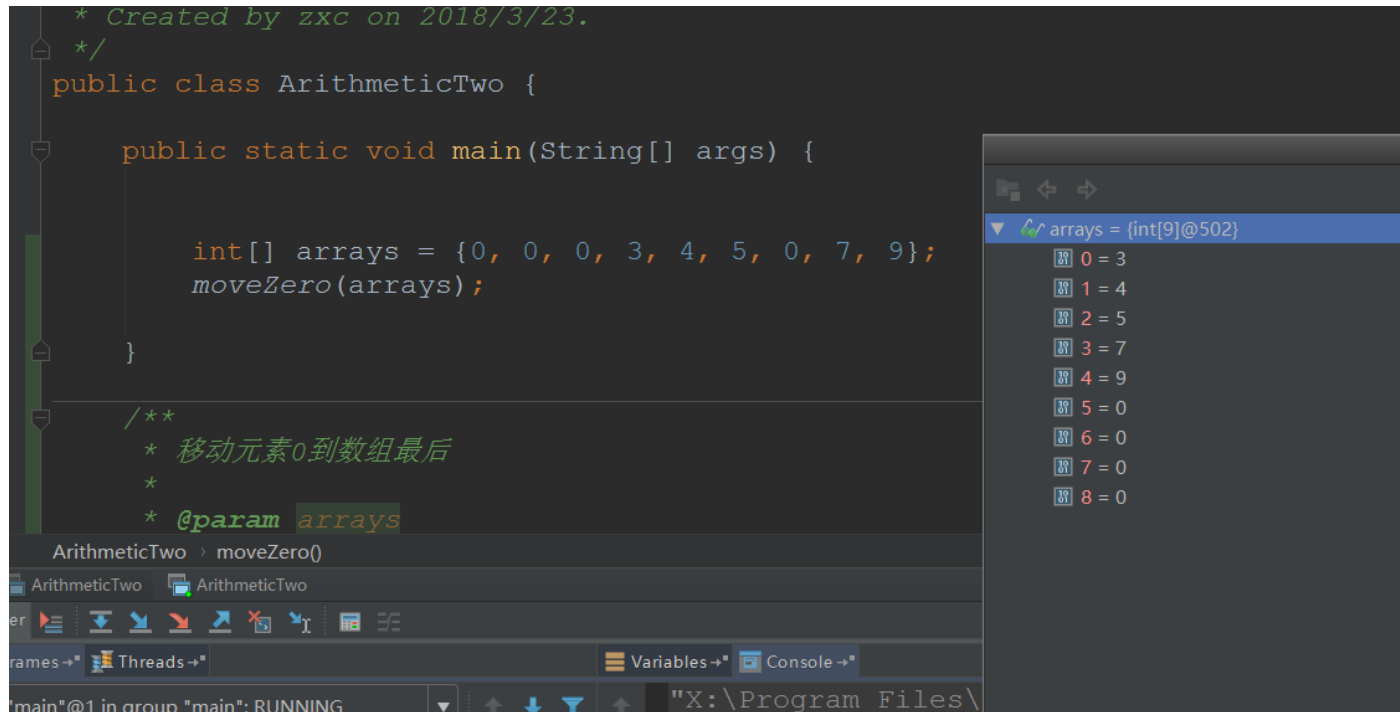
    // 1. 前面已经将非0的元素移动到数组的前面了
    // 2. 将为0的元素填满数组，填充的位置就从length - zero开始

    int j = arrays.length - zero;
    while (j < arrays.length) {
        arrays[j] = 0;
        j++;
    }

    System.out.println("关注公众号: Java3y---->" + arrays);

}
```

结果：



The screenshot shows an IDE with a Java class named `ArithmeticTwo`. The `main` method contains an array `arrays = {0, 0, 0, 3, 4, 5, 0, 7, 9}` and a call to `moveZero(arrays)`. A comment indicates the purpose is to move zeros to the end. The variable viewer on the right shows the state of the `arrays` array after execution: `arrays = (int[9]@502)` with values `0 = 3, 1 = 4, 2 = 5, 3 = 7, 4 = 9, 5 = 0, 6 = 0, 7 = 0, 8 = 0`.

```
* Created by zxc on 2018/3/23.
*/
public class ArithmeticTwo {

    public static void main(String[] args) {

        int[] arrays = {0, 0, 0, 3, 4, 5, 0, 7, 9};
        moveZero(arrays);

    }

    /**
     * 移动元素0到数组最后
     *
     * @param arrays
     */
    public static void moveZero() {

    }
}
```

ArithmeticTwo → moveZero()

ArithmeticTwo

Arrays → Threads → Variables → Console →

main"@1 in group "main": RUNNING

arrays = (int[9]@502)

0	=	3
1	=	4
2	=	5
3	=	7
4	=	9
5	=	0
6	=	0
7	=	0
8	=	0

还可以换种思路(差别不大)：将数组分成几个部分：在j之前的没有0，j到i全是0，i后面还没有遍历

- 如果遍历到的数字不是0，那么跟j进行交换，j++(保证j前面没有0和j到i全是0)
- 直至i遍历完毕后，j前面都不是0，j-i都是0(这就完成我们的任务了)

代码实现：

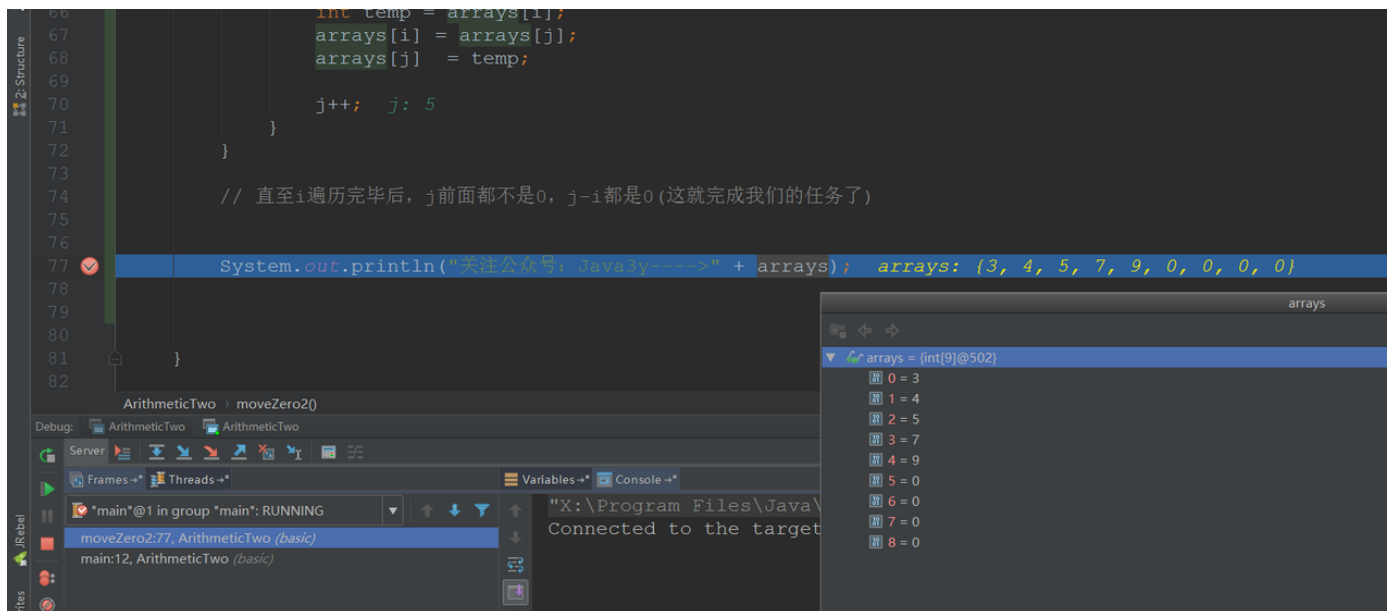
java 复制代码

```
/**
 * 移动元素0到数组最后
 *
 * @param arrays
 */
public static void moveZero2(int[] arrays) {

    // 在j前面的元素都不是0
    int j = 0;
```

```
for (int i = 0; i < arrays.length; i++) {  
  
    if (arrays[i] != 0) {  
  
        // 跟j进行交换, 保证j的前面都不是0  
        int temp = arrays[i];  
        arrays[i] = arrays[j];  
        arrays[j] = temp;  
  
        j++;  
    }  
}  
  
// 直至i遍历完毕后, j前面都不是0, j-i都是0(这就完成我们的任务了)  
  
System.out.println("关注公众号: Java3y----->" + arrays);  
  
}
```

结果还是一样的:



五、找出数组的单个数字

给你一个数组，除了一个数字外，其他的数字都出现了两次，请把这个只出现一次的数字找出来。

思路：

- 将该数组遍历一次，记录每个数字出现的次数
- 如果该数字出现的次数只有1，那么该数字就是单个数字～

java 复制代码

```
/**
 * 找出数组的单个数字
 * @param nums
 * @return
 */
public static void singleNumber(int[] nums) {
```

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

            int count = countNumber(nums, nums[i]);

            // 如果该元素只出现一次，那么就是它了！
            if (count == 1) {
                System.out.println("关注公众号：Java3y--->单一的元素是：" + nums[i]);

                return ;
            }

        }

    }

}

/**
 * 找出每个元素出现的次数
 * @param nums 数组
 * @param value 想知道出现次数的元素
 */

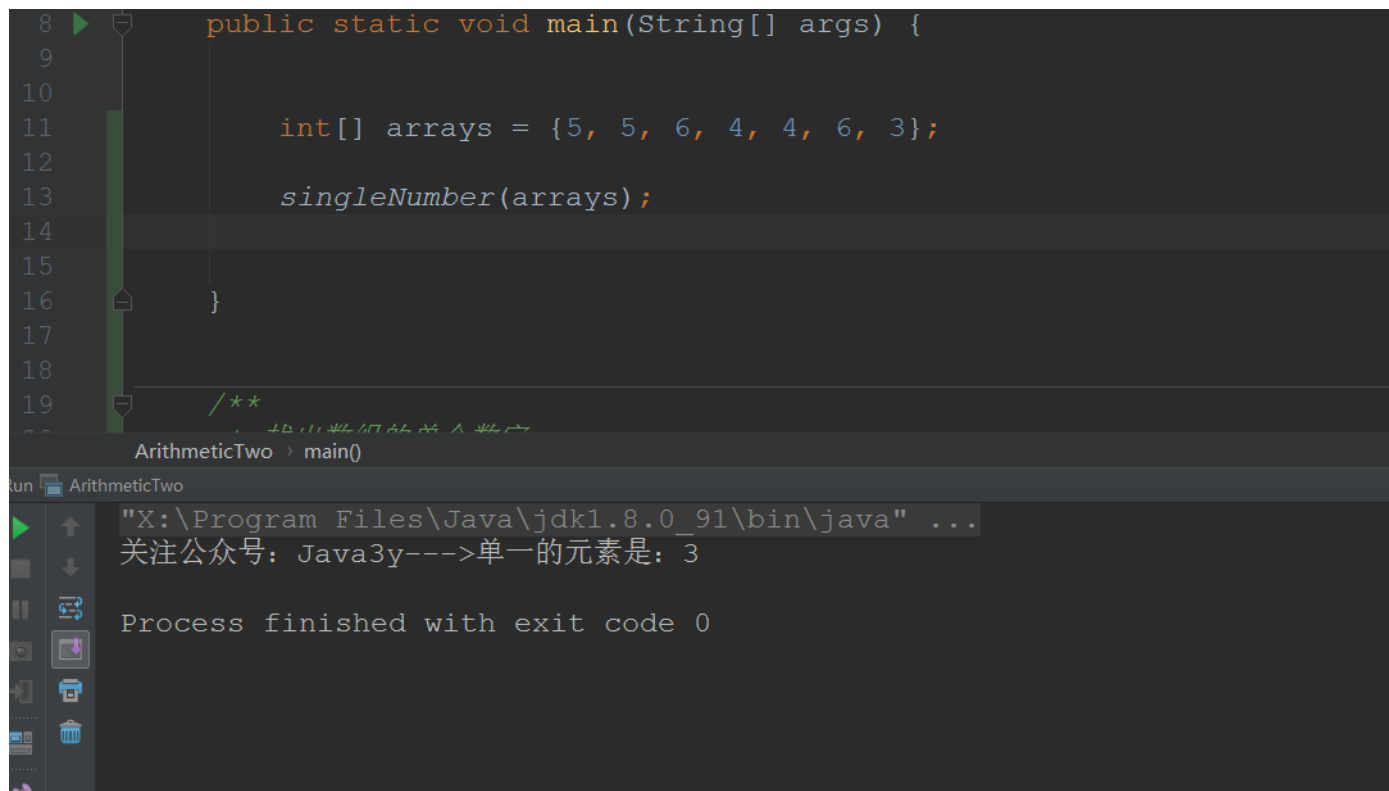
public static int countNumber(int[] nums,int value) {

    int count = 0;

    for (int i = 0; i < nums.length; i++) {
        if (value == nums[i]) {
            count++;
        }
    }

    // 返回该元素出现的次数
    return count;
}
```

结果：



The screenshot shows an IDE with a Java file named `ArithmeticTwo`. The code defines a `main` method that takes a `String[] args` array. Inside, it initializes an `int[] arrays` with the values `{5, 5, 6, 4, 4, 6, 3}` and calls a `singleNumber` method. The IDE's output window shows the command `"X:\Program Files\Java\jdk1.8.0_91\bin\java" ...` and the output `关注公众号：Java3y-->单一的元素是：3`. Below the output, it states `Process finished with exit code 0`.

优化：

这个问题最佳的解法是用到了位运算的异或操作：

- 如果 $5 \oplus 5 = 0$
- 如果 $5 \oplus 7 \oplus 5 = 7$
- 如果 $5 \oplus 6 \oplus 6 \oplus 5 \oplus 7 \oplus 8 \oplus 7 = 8$

从上面的例子可以看出：一堆数字做 异或运算[^]，俩俩相同数字就会被抵消掉～，所以这个特性对于这个题目而言就再适合不过的了：

java 复制代码

```
/**
 * 找出数组的单个数字
 * @param nums
 * @param numsSize
```

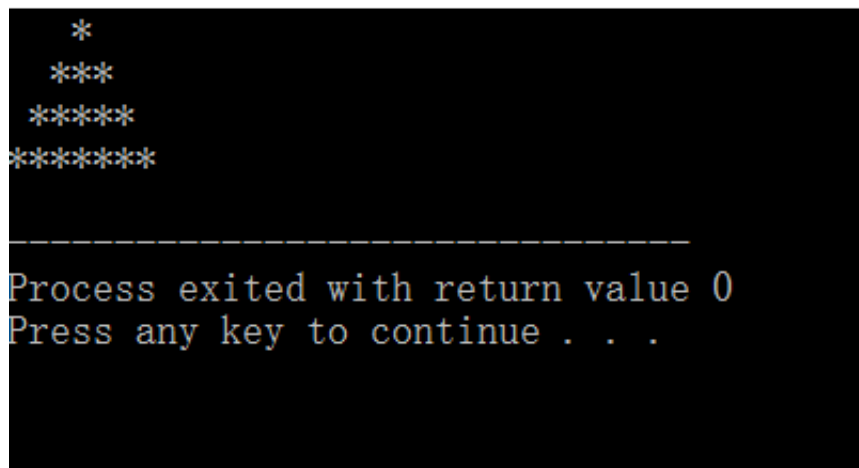


```
* @return
*/
public static int singleNumber(int[] nums, int numsSize) {

    // 第一个数和数组后面的数做^运算，留下的必然是单个数字
    int k = nums[0];
    for (int i = 1; i < numsSize; i++) {
        k = (k ^ nums[i]);
    }
    return k;
}
```

六、画三角形星星

画三角形星星



就是要画上面那种三角形星星，那怎么画呢？？

思路：

- 首先，我们可以发现：每行星星的个数是 $(2 * \text{行数} - 1)$ ，每行的空格数就是最大行数减去第 n 行（最大4行，第4行没有空格，最大4行，第三行1个空格）
- 有了上面的规律，套个for循环即可生成三角形星星～

实现代码：

java 复制代码

```
/**
 * 画星星
 */
public static void drawStar() {

    // 我要画5行的星星
    int row = 5;

    for (int i = 1; i <= 5; i++) {

        // 空格数等于最大行数 - 当前行数
        for (int j = 1; j <= row - i; j++) {
            System.out.print(" ");
        }

        // 星星数等于(当前行数*2-1)
        for (int j = 1; j <= i * 2 - 1; j++) {

            System.out.print("*");

        }

        // 每画一行就换一次行
        System.out.println();
    }
}
```

结果：

```
30 // 空格数等于最大行数 - 当前行数
31 for (int j = 1; j <= row - i; j++) {
32     System.out.print(" ");
33 }
34
35 // 星星数等于(当前行数*2-1)
36 for (int j = 1; j <= i * 2 - 1; j++) {
37     System.out.print("*");
38 }
39
40 // 每画一行就换一次行
41 System.out.println();
42 }
43
44 /**
45  * 找出数组的单个数字
46  */
47 ArithmeticTwo > drawStar()
48
49
```

Run ArithmeticTwo

```
"X:\Program Files\Java\jdk1.8.0_91\bin\java" ...
*
***
*****
*****
*****
```

七、罗马数字倒转成阿拉伯数字

罗马数字倒转成阿拉伯数字

罗马数字我们可能在英语的题目中看得是比较多的，一般常用的我们是阿拉伯数字，那怎么转成阿拉伯数字呢？我们先来了解一下罗马数字：

基本概念

[编辑本段](#)

罗马数字是最早的数字表示方式，比阿拉伯数字早2000多年，起源于**罗马**。

如今我们最常见的罗马数字就是钟表的表盘符号：**I, II, III, IV, V, VI, VII, VIII, IX, X, XI, XII.....**

对应阿拉伯数字（就是现在国际通用的数字），就是1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.....（注：**阿拉伯数字**其实是古代印度人发明的，后来由阿拉伯人传入欧洲，被欧洲人误称为阿拉伯数字。）

记数方法

[编辑本段](#)

基本字符	I	V	X	L	C	D	M
相应的阿拉伯数字表示为	1	5	10	50	100	500	1000

- 1、相同的数字连写，所表示的数等于这些数字相加得到的数，如：**III = 3**；
- 2、小的数字在大的数字的右边，所表示的数等于这些数字相加得到的数，如：**VIII = 8**；**XII = 12**；
- 3、小的数字，（限于**I**、**X**和**C**）在大的数字的左边，所表示的数等于大数减小数得到的数，如：**IV = 4**；**IX = 9**；
- 4、正常使用时，连写的数字重复不得超过三次。（表盘上的四点钟“IIII”例外）
- 5、在一个数的上面画一条横线，表示这个数扩大1000倍。

ps:来源360百科

规则在图上已经说得挺明白的了，我举几个例子：

- 左边的数比右边小，则是用右边的数减去左边的
- 左边的数比右边大，则是用右边的数加上左边的

7 External links

Roman numeric sys

Roman numerals, as used tod

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1,000

Numbers are formed by com
representing multiples of ten
CCVII (two hundreds, a five
Symbols are placed from left
XXXX), subtractive notation

- I placed before V or X in
- X placed before L or C in

50 - 1 = 49

50 + 1 = 51

$$\begin{aligned}
 & \boxed{M} \boxed{CMLIV} \\
 & M + (\boxed{C} \boxed{MLIV}) \\
 & = M + (M - C + (\boxed{LIV})) \\
 & = M + (M - C + (L + \boxed{IV})) \\
 & = M + (M - C + (L + V - I)) \\
 & = 1000 + (1000 - 100 + (50 + 5 - 1))
 \end{aligned}$$

看了上面的例子估计我们会手算将罗马数字转成阿拉伯数字了，那么用程序怎么写呢？？？

思路是这样的：

- 先找到罗马数字最大的那个数字
- 要是左边的数比右边小，则是用右边的数减去左边的
- 左边的数比右边大，则是用右边的数加上左边的
-如此循环则最后获取阿拉伯数字

首先，我们先定义罗马数字和对应的阿拉伯数字(相当于查表)

java 复制代码

```
// 定义罗马数字
char digits[] = {'I', 'V', 'X', 'L', 'C', 'D', 'M'};

// 罗马数字对应的阿拉伯数字
int values[] = { 1, 5, 10, 50, 100, 500, 1000};
```

随后，我们得找到罗马数字当前的最大值，找到最大值之前就先得把罗马数字转成是阿拉伯数字

java 复制代码

```
/**
 * 将罗马数字转成阿拉伯数字，实际上就是一个查表的过程
 *
 * @param roman
 * @return
 */
public static int digitsToValues(char roman) {

    // 定义罗马数字
    char digits[] = {'I', 'V', 'X', 'L', 'C', 'D', 'M'};

    // 罗马数字对应的阿拉伯数字
    int values[] = {1, 5, 10, 50, 100, 500, 1000};

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

        if (digits[i] == roman) {
            return values[i];
        }
    }
}
```

```
    }  
  
    return 0;  
  
}
```

上面的方法已经可以将罗马数字转成阿拉伯数字了，接下来我们要查找出最大值了

java 复制代码

```
/**  
 * 找到当前罗马数字最大值的角标  
 *  
 * @param digits  
 * @return  
 */  
public static int findMaxIndex(String digits, int L, int R) {  
  
    // 假设第一个是最大的  
    int max = digitsToValues(digits.charAt(L));  
    int maxIndex = L;  
  
    for (int i = L; i < R; i++) {  
        // 将罗马数字转成是阿拉伯数字  
        int num = digitsToValues(digits.charAt(i));  
        if (max < num) {  
            max = num;  
            maxIndex = i;  
        }  
    }  
  
    return maxIndex;  
}
```

找到了当前罗马数字的最大值那要怎么做???

- 左边的比右边的要小，则右边的减去左边的值

- 左边的比右边的要大，则右边的加上左边的值
-//实际上是一个递归的过程

于是乎，我们可以写出下面的代码：

java 复制代码

```
/**
 * 将罗马数字转成阿拉伯数字
 *
 * @param romanNumber
 * @param L
 * @param R
 */
public static int romanToNumber(String romanNumber, int L, int R) {

    // 如果只有一个罗马数字，那么可以直接返回了(递归出口)
    if (L == R) {
        return digitsToValues(romanNumber.charAt(L));
    } else if (L > R) { // 如果L和R已经越界了，那么说明没有值了
        return 0;
    } else {

        // 找到当前罗马数字最大值的角标
        int maxIndex = findMaxIndex(romanNumber, L, R);

        // 得到最大值
        int max = digitsToValues(romanNumber.charAt(maxIndex));

        // 在最大值左边的，则用最大值减去左边的
        int left = romanToNumber(romanNumber, L, maxIndex - 1);

        // 在最大值右边的，则用最大值加上右边的
        int right = romanToNumber(romanNumber, maxIndex + 1, R);

        return max - left + right;
    }
}
```


测试一下：

```
6 public class ArithmeticTwo {
7
8     public static void main(String[] args) {
9
10        String roman = "XVIII";
11
12        int num = romanToNumber(roman, L:0, R: roman.length() - 1);
13
14        System.out.println("关注公众号: Java3y----->" + num);
15    }
16
17    ...
18
19    ArithmeticTwo main()
20
21 Run ArithmeticTwo
22 "X:\Program Files\Java\jdk1.8.0_91\bin\java" ...
23 关注公众号: Java3y----->18
24
25 Process finished with exit code 0
```

八、啤酒与饮料

啤酒每罐2.3元，饮料每罐1.9元。小明买了若干啤酒和饮料，一共花了82.3元。我们还知道他买的啤酒比饮料的数量少，请你计算他买了几罐啤酒。

这是蓝桥杯的一道题，我们可以使用暴力搜索即可解出：

- 如果82.3全买啤酒最多能买 $82.3/2.3=35$ 瓶
- 如果82.3全买饮料最多能买 $82.3/1.9=43$ 瓶
- 以此作为控制条件

java 复制代码

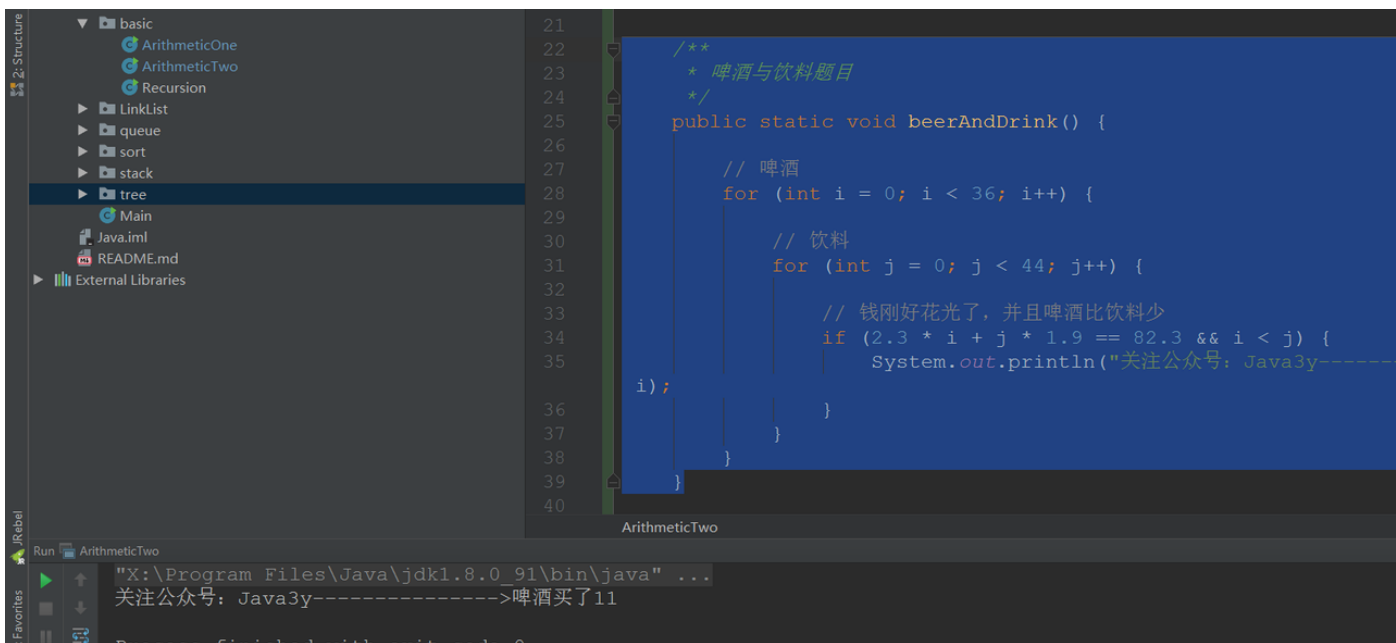
```
/**
 * 啤酒与饮料题目
 */
public static void beerAndDrink() {

    // 啤酒
    for (int i = 0; i < 36; i++) {

        // 饮料
```

```
for (int j = 0; j < 44; j++) {  
  
    // 钱刚好花光了，并且啤酒比饮料少  
    if (2.3 * i + j * 1.9 == 82.3 && i < j) {  
        System.out.println("关注公众号：Java3y----->啤酒买了" + i);  
    }  
}  
}  
}
```

测试：



```
/**  
 * 啤酒与饮料题目  
 */  
public static void beerAndDrink() {  
  
    // 啤酒  
    for (int i = 0; i < 36; i++) {  
  
        // 饮料  
        for (int j = 0; j < 44; j++) {  
  
            // 钱刚好花光了，并且啤酒比饮料少  
            if (2.3 * i + j * 1.9 == 82.3 && i < j) {  
                System.out.println("关注公众号：Java3y----->啤酒买了" + i);  
            }  
        }  
    }  
}
```

Run ArithmeticTwo

"X:\Program Files\Java\jdk1.8.0_91\bin\java" ...
关注公众号：Java3y----->啤酒买了11
Process finished with exit code 0

九、简单凯撒密码

简单凯撒密码

凯撒密码是啥？简单来说：就是通过移位来进行加密

- 比如，A-->B,B-->C,C-->D.....

上面就是最简单的凯撒密码，将所有的字母进行移一位，实现加密

明文字母表: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

密文字母表: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

明文: HELLO WORLD

密文: KHOOR ZRUOG

移了3位



下面我们也来玩一下吧～

左移动和右移动：

java 复制代码

```
/**
 * 右移
 */
public static int rotateRight(int ch) {
    if (ch >= 'A' && ch <= 'Y') {
        return ch + 1;
    } else if (ch >= 'a' && ch <= 'y') {
        return ch + 1;
    } else if (ch == 'Z') {
        return 'A';
    } else if (ch == 'z') {

```

```
        return 'a';
    } else {
        return ch;
    }
}

/**
 * 左移
 */
public static int rotateLeft(int ch) {
    if (ch >= 'B' && ch <= 'Z') {
        return ch - 1;
    } else if (ch >= 'b' && ch <= 'z') {
        return ch - 1;
    } else if (ch == 'A') {
        return 'Z';
    } else if (ch == 'a') {
        return 'z';
    } else {
        return ch;
    }
}
```

加密:

java 复制代码

```
/**
 * 加密
 * @param ch
 * @param shift
 * @return
 */
public static int encode(int ch, int shift) {

    // 如果没有移动, 则直接返回
    if (shift == 0) {
        return ch;
    } else if (shift > 0) {
```

```
        // 如果shift移动的是正数，那么就向右移动
        for (int i = 0; i < shift; i++) {
            ch = rotateRight(ch);
        }
        return ch;
    } else {

        // 如果shift移动的是负数，那么就向左移动
        for (int i = 0; i < -shift; i++) {
            ch = rotateLeft(ch);
        }
        return ch;
    }
}
```

测试：

java 复制代码

```
String s = "HELLO WORLD";
char[] ch = new char[s.length()];

for (int i = 0; i < s.length(); i++) {
    ch[i] = (char) encode(s.charAt(i), 3);
}

System.out.println("关注公众号：Java3y" + ch);
```

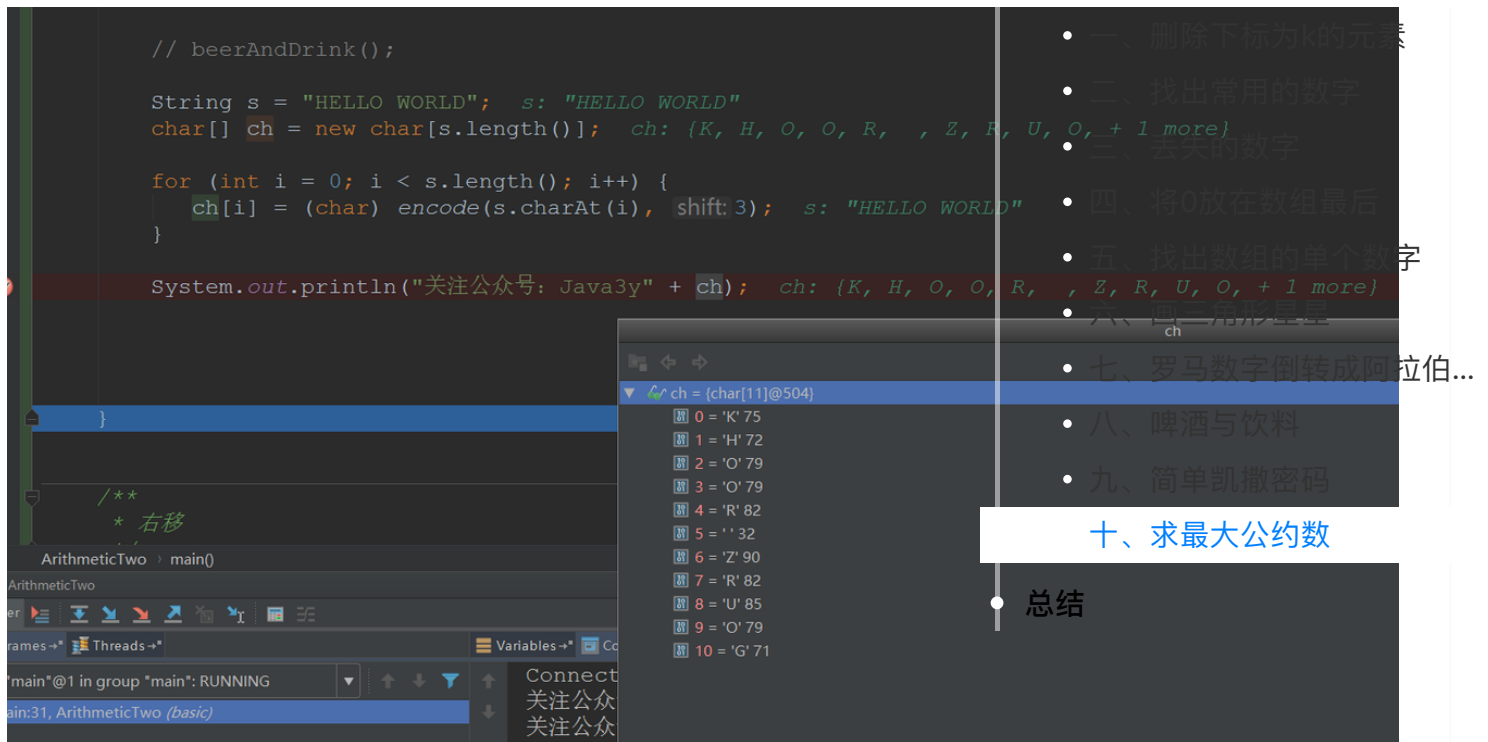
结果：

目录

前言

十道简单算法题

题目的总览



十、求最大公约数

求一个数的最大公约数

算法：是两个数相余，直到余数为0，如果余数不为0，就用除数和余数求余

- 若发现余数为0，那么当前的除数就是最大公约数

java 复制代码

```
/**
 * 求最大公约数
 *
 * @param num1
 * @param num2
 */
public static int gcd(int num1, int num2) {

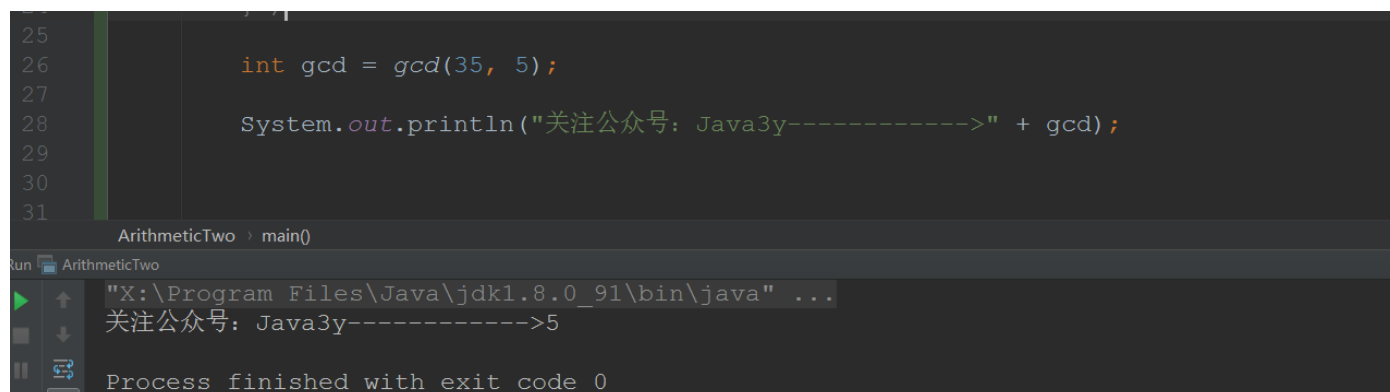
    // 求余数
```

```
    int r = num1 % num2;

    // 如果余数为0，那么除数就是最大公约数
    if (r == 0) {
        return num2;
    } else {

        // 否则，则用除数和余数来进行运算
        return gcd(num2, r);
    }
}
```

结果：



The screenshot shows an IDE with a Java file named ArithmeticTwo.java. The code defines a gcd method and calls it with gcd(35, 5). The output of the program is displayed in the console, showing the result 5. The console output is: "X:\Program Files\Java\jdk1.8.0_91\bin\java" ... 关注公众号: Java3y----->5. The process finished with exit code 0.

```
25
26     int gcd = gcd(35, 5);
27
28     System.out.println("关注公众号: Java3y----->" + gcd);
29
30
31
```

ArithmeticTwo → main()

Run ArithmeticTwo

"X:\Program Files\Java\jdk1.8.0_91\bin\java" ...
关注公众号: Java3y----->5
Process finished with exit code 0

总结

没错，你没看错，简单的小算法也要总结！

其实我觉得这些比较简单的算法是有"套路"可言的，你如果知道它的套路，你就很容易想得出来，如果你不知道它的套路，那么很可能就不会做了(没思路)。

积累了一定的"套路"以后，我们就可以根据经验来推断，揣摩算法题怎么做了。

举个很简单的例子：

- 乘法是在加法的基础之上的，那乘法我们是怎么学的？**背(积累)**出来的，**9*9** 乘法表谁没背过？比如看到 **2+2+2+2+2**，会了乘法(套路)以后，谁还会慢慢加上去。看见了5个2，就直接得出 **2*5** 了

1. 删除下标为k的元素
 - 后一位往前一位覆盖即可
2. 找出常用的数字
 - 利用栈的思想，只要该数组出现的次数大于2分之1，那么他肯定是在栈里面
3. 丢失的数字
 - 实现1：两个数组进行遍历，如果某一个不存在，利用数组的角标就可以找到～
 - 实现2：使用等差求和公式，缺失的数字可以减出来！
4. 将0放在数组最后
 - 实现1：使用变量zero来记住有多少个0，只要不是0就往前面移动，最后将zero补全！
 - 实现2：将数组分成3个部分；在j之前的没有0，j到i全是0，i后面还没有遍历，直至i遍历完毕后，j前面都不是0，j-i都是0(这就完成我们的任务了)
5. 找出数组的单个数字
 - 实现1：遍历数组计算某个元素出现的次数，外层再遍历数组，只要该元素出现的次数是1，那么它就是单个的！
 - 实现2：位运算的异或操作，相同的两个数字会抵消掉！
6. 画三角形星星
 - 找到画星星和空格的规律！星星和空格都与行数有关联！
7. 罗马数字倒转成阿拉伯数字
 - 将罗马数组和阿拉伯数字对应起来，“查表”进行转换！找到最大的值，左边比右边要小，则右减左。反之右加左！
8. 啤酒与饮料
 - 使用暴力查询的方式来将具体的值搜索出来！
9. 简单凯撒密码
 - char本质上就是int，移动时要主要Z，A这些字符～
10. 求最大公约数
 - 如果余数为0，那么除数就是最大公约数，否则就是除数和余数再继续运算！

文章的目录导航: zhongfucheng.bitcron.com/post/shou-j...

如果文章有错的地方欢迎指正，大家互相交流。习惯在微信看技术文章，想要获取更多的Java资源的同学，可以关注微信公众号:Java3y



关注下面的标签，发现更多相似文章

算法

Java

后端

微信

Java3y   掘金特邀作者 | 技术公众号: Java3y
发布了 238 篇专栏 · 获得点赞 18,013 · 获得阅读 694,065

关注

安装掘金浏览器插件

打开新标签页发现好内容，掘金、GitHub、Dribbble、ProductHunt 等站点内容轻松获取。快来安装掘金浏览器插件获取高质量内容吧！

评论

输入评论...

Java3y  (作者)  掘金特邀作者 | 技术公众号...

为了大家方便，刚新建了一下qq群：742919422，大家可以去交流交流。

1年前

 1

 回复

相关推荐

专栏 · 程序员内点事 · 9小时前 · Java

为了不复制粘贴，我被逼着学会了JAVA爬虫

 12  1

专栏 · 风平浪静如码 · 4小时前 · Java

微服务架构：如何用十步解耦你的系统？

 5  1

专栏 · 腾讯云中间件 · 5小时前 · Kubernetes / 后端

浅谈Kubernetes Ingress控制器的技术选型

 6 

专栏 · 程序员内点事 · 1天前 · Java

技术部突然宣布：JAVA开发人员全部要会接口自动化测试框架

 45  1

专栏 · 北海北方 · 2年前 · HTTP

HTTP----HTTP缓存机制

👍 759

💬 47

专栏 · 苡仁 · 1天前 · Java

重学Java并发编程—JMM（Java内存模型）在并发中的原理与应用

👍 7

💬

专栏 · Throwable · 12小时前 · Java

ThreadLocal源码分析-黄金分割数的使用

👍 4

💬

专栏 · HollisChuang · 3天前 · Java

新来个技术总监，禁止我们使用Lombok！

👍 89

💬 72

专栏 · peen · 2天前 · JavaScript / 算法

一个简单的例子提高你的算法能力

👍 93

💬 42