剑指Offer\_56\_找出数组中只出现1次的数字

# 题目一：找出只出现一次的1个数字

## 题目介绍：

题目1：数组中只有一个数只出现一次，其他数都出现两次；找出只出现一次的数字。

## 思路介绍

将所有数字**异或**即可。

理论知识：**任何数字与本身以后都是0;任何数字与0异或都是本身**。

## Java代码

/\*\*

\* 找出只出现一次的这个数；

\* 思路：异或即可解决。

\*/

public int getNumberOfOnlyOnce(int[] nums){

int result = 0;

**for(int num :nums) result ^= num;**

return result;

}

# 题目二：找出只出现1次的两个数

## 题目介绍

题目2：找出只出现1次的两个数

数组中有两个数只出现一次，其他数都出现两次；

\* 找出只出现一次的这两个数；

## 思路介绍

思路：异或。首先异或所有，结果中肯定至少有1个Bit为1；根据这个Bit是1和0分成两小组；

\* 分别异或各个小组，各个小组的异或结果即为答案。

## Java代码

public void **FindNumsAppearOnce**(int [] nums,int num1[] , int num2[]) {

int result = 0;

for(int num :nums) result ^= num;

int indexRight = **findIndexOfBitIs1FromRight**(result);//需要右移的位数

for(int num:nums){

if(theBitIs1(num,indexRight)) **num1[0] ^= num;**

else **num2[0] ^= num;**

}

}

/\*\*

\* 需要右移的比特数目

\*/

public int findIndexOfBitIs1FromRight(int num){

int n = 0;//不需要移位

//利用右移，判断正数可以

while((num & 1)==0 && n++<32)

num = num >>1;

return n;

}

/\*\*

\* 判断num的第indexBit位是否为1(从右边数)

\*/

public boolean theBitIs1(int num,int indexBit){

num = num >> indexBit;

return (num&1) == 1;

}

# 题目三：其他数均出现3次，找出只出现1次的这个数

## 题目介绍

题目介绍：一个数组中只有一个数只出现1次，其他数均出现3次，找出只出现1次的这个数。

## 思路介绍

\* \* 思路分析：

\* 将所有数对应比特相加起来，出现三次的肯定是3的倍数，

\* 最终对3求余，得到结果的每一个比特值

## Java代码

@Test

public void test3(){

int[] nums = {3,2,3,3,2,2,5,5,5,89,89,32,89};

int number = findNumberAppearOnlyOnce(nums);

System.out.println("number = " + number);

}

public int **findNumberAppearOnlyOnce**(int[] nums){

int result = 0;

int[] bits = new int[32];

for(int i = 0; i < 32;i++){

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

**bits[i] += theBitIs1(nums[j],i)?1:0;**

}

bits[i] %= 3;

}

//将长度为32的数组转换成一个int值

for(int i = 31;i > 0;i--){

result = (result|bits[i])<<1;

}

return result|bits[0];

}

public boolean **theBitIsOne**(int num,int index){

num = num>>index;

return (num&1) ==1;

}