Given a **non-empty** array of integers, every element appears *twice* except for one. Find that single one.

**Note:**

Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

**Example 1:**

**Input:** [2,2,1]

**Output:** 1

**Example 2:**

**Input:** [4,1,2,1,2]

**Output:** 4

一个数列，只有一个数出现过一次，其他数都出现过两次，额外要求：最好不要用额外的memory

基本方法：最简单的方法当然就是用map(hashmap)，因为map需要key，我们如果没有这个key,就插入，如果有这个key，把这个key删掉，那么最后map只会剩下一个key-value，也就是没消掉的那个

高级方法：异或法xor法

^=代表着xor

**Concept**

* If we take XOR of zero and some bit, it will return that bit //任何数与0xor等于它本身
  + a⊕ 0 =a
* If we take XOR of two same bits, it will return 0 任何数与它本身xor等于0
  + *a*⊕*a*=0
* *a*⊕*b*⊕*a*=(*a*⊕*a*)⊕*b*=0⊕*b*=*b 满足交换律*

So we can XOR all bits together to find the unique number.

class Solution {

public int singleNumber(int[] nums) {

int result=0;

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

result^=nums[i];

}

return result;

}

}

那么我们已开始设置一个结果0，然后和所有数xor，根据交换律，他遇到第一个数会变成那个数，再遇到第二个数会重新变成0，最后没抵消的就是单一的数