# 题目

给定一个非空整数数组，除了某个元素只出现一次以外，其余每个元素均出现了三次。找出那个只出现了一次的元素。

说明：

你的算法应该具有线性时间复杂度。 你可以不使用额外空间来实现吗？

示例 1:

输入: [2,2,3,2]

输出: 3

示例 2:

输入: [0,1,0,1,0,1,99]

输出: 99

**类似题目：**剑指 Offer 56 - II

# 分析

## 方法一：哈希表

**思路：**

对于出现次数的统计类问题，可以采用哈希表（对应数据结构unorded\_map）。

**代码：**

class Solution {

public:

int singleNumber(vector<int>& nums) {

unordered\_map<int,int> mp;

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

mp[nums[i]]++;

unordered\_map<int,int>::iterator it = mp.begin();

for(; it!=mp.end(); it++)

{

if(it->second==1)

return it->first;

}

return 0;

}

};

## 方法二：位运算

**思路：**

使用位运算符可以实现O(1) 的空间复杂度。

∼x表示位运算 NOT

x&y表示位运算 AND

x⊕y表示位运算 XOR

**XOR**

该运算符用于检测出现奇数次的位：1、3、5 等。

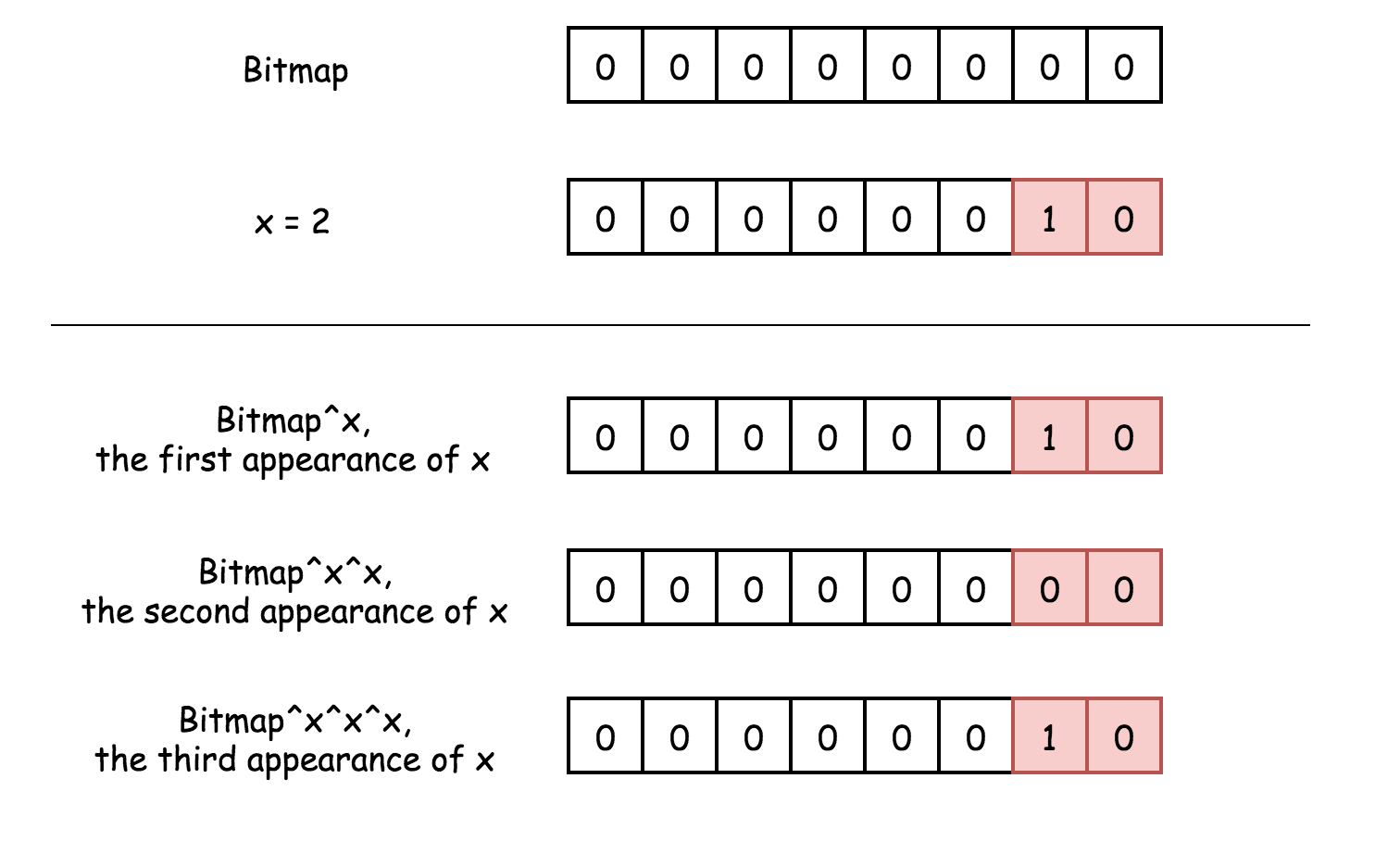
0 与任何数 XOR 结果为该数。

0⊕x=x

两个相同的数 XOR 结果为 0。

x⊕x=0

以此类推，只有某个位置的数字出现奇数次时，该位的掩码才不为 0。



因此，可以检测出出现一次的位和出现三次的位，但是要注意区分这两种情况。

**AND和NOT**

为了区分出现一次的数字和出现三次的数字，使用两个位掩码：seen\_once 和 seen\_twice。

**思路是：**

仅当 seen\_twice 未变时，改变 seen\_once。

仅当 seen\_once 未变时，改变seen\_twice。



位掩码 seen\_once 仅保留出现一次的数字，不保留出现三次的数字。

**代码：**

class Solution {

public int singleNumber(int[] nums) {

int seenOnce = 0, seenTwice = 0;

for (int num : nums) {

// first appearence:

// add num to seen\_once

// don't add to seen\_twice because of presence in seen\_once

// second appearance:

// remove num from seen\_once

// add num to seen\_twice

// third appearance:

// don't add to seen\_once because of presence in seen\_twice

// remove num from seen\_twice

seenOnce = ~seenTwice & (seenOnce ^ num);

seenTwice = ~seenOnce & (seenTwice ^ num);

}

return seenOnce;

}

}