位运算

题目描述：

给你一个整数数组 `nums` ，除某个元素仅出现 **一次** 外，其余每个元素都恰出现 **三次** 。请你找出并返回那个只出现了一次的元素。

示例 1：

```
输入：nums = [2,2,3,2]
输出：3
```

示例 2：

```
输入：nums = [0,1,0,1,0,1,99]
输出：99
```

题目分析：位运算（逐一确定每一位的二进制数）

方法二：依次确定每一个二进制位

思路与算法

为了方便叙述，我们称「只出现了一次的元素」为「答案」。

由于数组中的元素都在 `int`（即 32 位整数）范围内，因此我们可以依次计算答案的每一个二进制位是 0 还是 1。

具体地，考虑答案的第 i 个二进制位（i 从 0 开始编号），它可能为 0 或 1。对于数组中非答案的元素，每一个元素都出现了 3 次，对应着第 i 个二进制位的 3 个 0 或 3 个 1，无论是哪一种情况，它们的和都是 3 的倍数（即和为 0 或 3）。因此：

> 答案的第 i 个二进制位就是数组中所有元素的第 i 个二进制位之和除以 3 的余数。

这样一来，对于数组中的每一个元素 x，我们使用位运算 `(x >> i) & 1` 得到 x 的第 i 个二进制位，并将它们相加再对 3 取余，得到的结果一定为 0 或 1，即为答案的第 i 个二进制位。

细节

需要注意的是，如果使用的语言对「有符号整数类型」和「无符号整数类型」没有区分，那么可能会得到错误的答案。这是因为「有符号整数类型」（即 `int` 类型）的第 31 个二进制位（即最高位）是补码意义下的符号位，对应着 -2^{31}，而「无符号整数类型」由于没有符号，第 31 个二进制位对应着 2^{31}。因此在某些语言（例如 `Python`）中需要对最高位进行特殊判断。

代码实现

```
class Solution {
public:
    int singleNumber(vector<int>& nums) {
        int ans = 0;
        for (int i = 0; i < 32; ++i) {
            int total = 0;
            for (int num: nums) {
                total += ((num >> i) & 1);
            }
            if (total % 3) {
                ans |= (1 << i);
            }
        }
        return ans;
    }
};

作者：LeetCode-Solution
```