

只出现一次的数字 II

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:

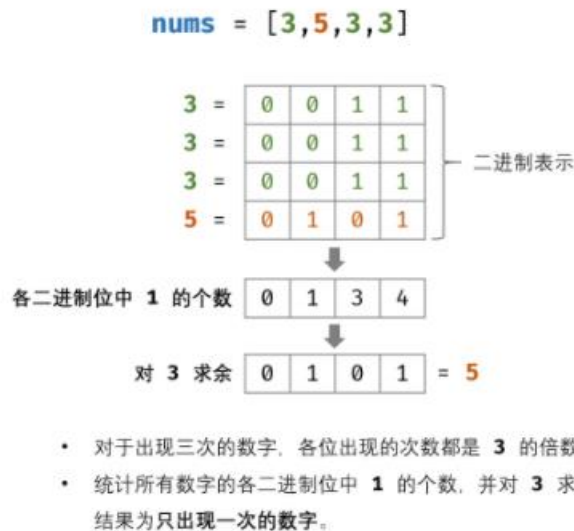
输入：[2,2,3,2]

输出：3

示例 2:

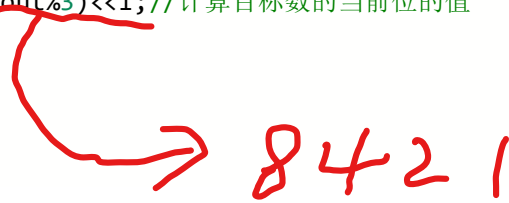
输入：[0,1,0,1,0,1,99]

输出：99



```
class Solution {
public:
    int singleNumber(vector<int>& nums) {
        /*
        统计所有数字中每个位中 1 出现的总数，
        对于某个位 1 出现的次数一定是 3 的倍数+1 或 0，那么对这个数%3 得到的结果就是
        目的数字在该位上的值
        */
        int ans=0;
        for(int i=0;i<32;i++) //int 类型为 32 位，因此只需统计 32 位
        {
```

```
int cout=0;//记录当前位 1 的个数
for(auto& n:nums)
{
    cout+=(n>>i)&1; //获取当前数的当前位的 1 的个数
}
ans+=(cout%3)<<i;//计算目标数的当前位的值
}
return ans;
}
};
```



8421