# 题目

给定一个只包含整数的有序数组，每个元素都会出现两次，唯有一个数只会出现一次，找出这个数。

示例 1:

输入: nums = [1,1,2,3,3,4,4,8,8]

输出: 2

示例 2:

输入: nums = [3,3,7,7,10,11,11]

输出: 10

提示:

1 <= nums.length <= 105

0 <= nums[i] <= 105

进阶: 采用的方案可以在O(log n)时间复杂度和O(1)空间复杂度中运行吗？

# 分析

## 方法一：异或运算

**思路：**

如果这道题输入的数组是未排序的，那么这道题就有个通用解法。由于相同的两个数字的异或结果为0，那么将数组内的所有数字都异或后，最后的结果就是单独只出现过一次的数字。完整的代码如下，时间复杂度为O(n)。

**代码：**

class Solution {

public:

int singleNonDuplicate(vector<int>& nums) {

int ret = 0;

for (auto& num : nums) {

ret ^= num;

}

return ret;

}

};

## 方法二：二分查找

代码：

class Solution {

public:

int singleNonDuplicate(vector<int>& nums) {

int left = 0;

int right = nums.size() / 2 - 1;

while (left <= right) {

int mid = left + ((right - left) >> 1);

int i = 2 \* mid;

if (nums[i] != nums[i + 1]) {

if (i == 0 || nums[i - 1] == nums[i - 2]) {

return nums[i];

}

right = mid - 1;

}

else {

left = mid + 1;

}

}

return nums.back();

}

};

**复杂度分析：**

时间复杂度为O(logn)

空间复杂度为 O(1)。