# 题目

给定一个大小为n的整数数组，找出其中所有出现超过⌊ n/3 ⌋次的元素。

示例 1：

输入：nums = [3,2,3]

输出：[3]

示例 2：

输入：nums = [1]

输出：[1]

示例 3：

输入：nums = [1,2]

输出：[1,2]

提示：

1 <= nums.length <= 5 \* 104

-109 <= nums[i] <= 109

进阶：尝试设计时间复杂度为O(n)、空间复杂度为O(1)的算法解决此问题。

# 分析

## 方法一：哈希表

思路：

我们用哈希统计数组中每个元素出现的次数，设数组的长度为n，返回所有统计次数超过⌊n/3⌋的元素。

代码：

class Solution {

public:

vector<int> majorityElement(vector<int>& nums) {

int n = nums.size();

vector<int> ans;

unordered\_map<int, int> cnt;

// 记录每个值的出现次数

for (auto &num : nums) {

cnt[num]++;

}

for (auto &num : cnt) {

if (num.second > n / 3) {

ans.push\_back(num.first);

}

}

return ans;

}

};

复杂度分析：

时间复杂度：O(n)，其中n为数组的长度。

空间复杂度：O(n)，其中n为数组的长度，使用哈希表需要开辟额外的空间。