# 题目

给你一个下标从 0 开始的整数数组 nums 。在一步操作中，你可以执行以下步骤：

从 nums 选出 两个 相等的 整数

从 nums 中移除这两个整数，形成一个 数对

请你在 nums 上多次执行此操作直到无法继续执行。

返回一个下标从 0 开始、长度为 2 的整数数组 answer 作为答案，其中 answer[0] 是形成的数对数目，answer[1] 是对 nums 尽可能执行上述操作后剩下的整数数目。

示例 1：

输入：nums = [1,3,2,1,3,2,2]

输出：[3,1]

解释：

nums[0] 和 nums[3] 形成一个数对，并从 nums 中移除，nums = [3,2,3,2,2] 。

nums[0] 和 nums[2] 形成一个数对，并从 nums 中移除，nums = [2,2,2] 。

nums[0] 和 nums[1] 形成一个数对，并从 nums 中移除，nums = [2] 。

无法形成更多数对。总共形成 3 个数对，nums 中剩下 1 个数字。

示例 2：

输入：nums = [1,1]

输出：[1,0]

解释：nums[0] 和 nums[1] 形成一个数对，并从 nums 中移除，nums = [] 。

无法形成更多数对。总共形成 1 个数对，nums 中剩下 0 个数字。

示例 3：

输入：nums = [0]

输出：[0,1]

解释：无法形成数对，nums 中剩下 1 个数字。

提示：

1 <= nums.length <= 100

0 <= nums[i] <= 100

# 分析

## 方法一：哈希表

思路：

遍历一次数组，用一个哈希表保存元素个数的奇偶性，偶数为false，奇数则为true。每遇到一个元素，则将奇偶性取反，若取反完后为偶数个，则表明在上次偶数个之后又遇到了两个该元素，可以形成一个数对。最后返回一个数组，第一个元素是数对数，第二个元素是数组长度减去数对数的两倍。

代码：

class Solution {

public:

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

unordered\_map<int, bool> cnt;

int res = 0;

for (int num : nums) {

if (cnt.count(num)) {

cnt[num] = !cnt[num];

} else {

cnt[num] = true;

}

if (!cnt[num]) {

res++;

}

}

return {res, (int)nums.size() - 2 \* res};

}

};

复杂度分析

时间复杂度：O(n)，其中n是数组的长度。需要遍历一次数组。

空间复杂度：O(n)。哈希表中最多保存n个元素。