# 题目

给你三个整数数组 nums1、nums2 和 nums3 ，请你构造并返回一个 元素各不相同的 数组，且由 至少 在 两个 数组中出现的所有值组成。数组中的元素可以按 任意 顺序排列。

示例 1：

输入：nums1 = [1,1,3,2], nums2 = [2,3], nums3 = [3]

输出：[3,2]

解释：至少在两个数组中出现的所有值为：

- 3 ，在全部三个数组中都出现过。

- 2 ，在数组 nums1 和 nums2 中出现过。

示例 2：

输入：nums1 = [3,1], nums2 = [2,3], nums3 = [1,2]

输出：[2,3,1]

解释：至少在两个数组中出现的所有值为：

- 2 ，在数组 nums2 和 nums3 中出现过。

- 3 ，在数组 nums1 和 nums2 中出现过。

- 1 ，在数组 nums1 和 nums3 中出现过。

示例 3：

输入：nums1 = [1,2,2], nums2 = [4,3,3], nums3 = [5]

输出：[]

解释：不存在至少在两个数组中出现的值。

提示：

1 <= nums1.length, nums2.length, nums3.length <= 100

1 <= nums1[i], nums2[j], nums3[k] <= 100

# 分析

## 方法一：哈希表

思路：

题目给出三个整数数组nums1、nums2和nums3。现在我们需要求一个元素各不相同的数组，其中的元素为至少在数组nums1、nums2和nums3中两个数组出现的全部元素。

我们可以用「哈希表」来实现——由于只有三个数组，所以我们一个整数的最低三个二进制位来标记某一个数在哪几个数组中，1表示该数在对应的数组中的，反之0表示不在。最后我们只需要判断每一个数对应的标记数字中二进制位个数是否大于1即可。

代码：

class Solution {

public:

vector<int> twoOutOfThree(vector<int>& nums1, vector<int>& nums2, vector<int>& nums3) {

unordered\_map<int, int> count;

// 因为这里是统计不同数组的个数，单个数组内不需要递增

// 为了区分不同数组是否含有对应的数字，这里使用位运算计数

for (int num : nums1) count[num] |= 0x01;

for (int num : nums2) count[num] |= 0x10;

for (int num : nums3) count[num] |= 0x100;

vector<int> result;

for (auto& [k, v] : count) {

int onesCount = 0;

// 使用位运算和数值判断来计算v中设置为1的位数

if (v & 0x01) onesCount++; // 检查最低位

if (v & 0x10) onesCount++; // 检查次低位

if (v & 0x100) onesCount++; // 检查第三个位

// 如果至少有两个1，则将该数字添加到结果中

if (onesCount >= 2) {

result.push\_back(k);

}

}

return result;

}

};

或：

class Solution {

public:

vector<int> twoOutOfThree(vector<int>& nums1, vector<int>& nums2, vector<int>& nums3) {

unordered\_map<int, int> count;

// 因为这里是统计不同数组的个数，单个数组内不需要递增

// 为了区分不同数组是否含有对应的数字，这里使用位运算计数

for (int num : nums1) count[num] |= 1;

for (int num : nums2) count[num] |= 2;

for (int num : nums3) count[num] |= 4;

vector<int> result;

for (auto& [k, v] : count) {

int onesCount = 0;

// 使用位运算和数值判断来计算v中设置为1的位数

if (v & 1) onesCount++; // 检查最低位

if (v & 2) onesCount++; // 检查次低位

if (v & 4) onesCount++; // 检查第三个位

// 如果至少有两个1，则将该数字添加到结果中

if (onesCount >= 2) {

result.push\_back(k);

}

}

return result;

}

};

或：

class Solution {

public:

vector<int> twoOutOfThree(vector<int>& nums1, vector<int>& nums2, vector<int>& nums3) {

unordered\_map<int, int> count;

// 因为这里是统计不同数组的个数，单个数组内不需要递增

// 为了区分不同数组是否含有对应的数字，这里使用位运算计数

for (int num : nums1) count[num] |= 0x01;

for (int num : nums2) count[num] |= 0x10;

for (int num : nums3) count[num] |= 0x100;

vector<int> result;

for (auto& [k, v] : count) {

if (v & (v - 1)) {

result.push\_back(k);

}

}

return result;

}

};

复杂度分析

时间复杂度：O(n1+n2+n3)，其中n1，n2，n3分别为数组 nums1，nums2，nums3的长度。

空间复杂度：O(n1+n2+n3)，其中 n1，n2，n3分别为数组 nums1，nums2，nums3的长度，主要为哈希表的空间开销。