# 题目

给你两个整数数组 nums1 和 nums2 ，请你以数组形式返回两数组的交集。返回结果中每个元素出现的次数，应与元素在两个数组中都出现的次数一致（如果出现次数不一致，则考虑取较小值）。可以不考虑输出结果的顺序。

示例 1：

输入：nums1 = [1,2,2,1], nums2 = [2,2]

输出：[2,2]

示例 2:

输入：nums1 = [4,9,5], nums2 = [9,4,9,8,4]

输出：[4,9]

提示：

1 <= nums1.length, nums2.length <= 1000

0 <= nums1[i], nums2[i] <= 1000

进阶:

如果给定的数组已经排好序呢？你将如何优化你的算法？

如果nums1的大小比nums2小很多，哪种方法更优？

如果nums2的元素存储在磁盘上，磁盘内存是有限的，并且你不能一次加载所有的元素到内存中，你该怎么办？

# 分析

求交集的问题通常采用的方法：排序+哈希表

## 方法一：内置函数

## 方法二：哈希映射

前面的问题349两个数组的交集，我们使用 set 来实现线性时间复杂度。在这里，我们需要使用HashMap来跟踪每个数字出现的次数。

我们先在HashMap 记录一个数组中的存在的数字和对应出现的次数。然后，我们遍历第二个数组，检查数字在HashMap中是否存在，如果存在且计数为正，则将该数字添加到答案并减少HashMap中的计数。

检查数组的大小并对较小的数组进行哈希映射是一个小细节，当其中一个数组较大时，会减少内存的使用。

如果nums1元素个数大于nums2，则交换数组元素。

对于nums1的每个元素，添加到HashMap m中，如果元素已经存在则增加对应的计数。

初始化k = 0，记录当前交集元素个数。

遍历数组nums2：

检查元素在m是否存在，若存在且计数为正：

将元素拷贝到nums1[k]，且k++。

减少m中对应元素的计数。

返回nums1 前k个元素。

## 方法三：排序+双指针

当输入数据是有序的，推荐使用此方法。在这里，我们对两个数组进行排序，并且使用两个指针在一次扫面找出公共的数字。

**算法：**

对数组nums1和nums2排序。

初始化指针i，j和k为 0。

指针i指向nums1，指针j指向nums2：

如果nums1[i] < nums2[j]，则 i++。

如果nums1[i] > nums2[j]，则 j++。

如果nums1[i] == nums2[j]，将元素拷贝到nums1[k]，且i++，j++，k++。

返回数组nums1前k个元素。

**代码：**

class Solution {

public:

vector<int> intersect(vector<int>& nums1, vector<int>& nums2) {

std::sort(nums1.begin(),nums1.end());

std::sort(nums2.begin(),nums2.end());

vector<int>::iterator i=nums1.begin(), j=nums2.begin();

vector<int> s;

while(i!=nums1.end() && j!=nums2.end())

{

if(\*i == \*j)

{

s.push\_back(\*i);

i++;

j++;

}

else if(\*i < \*j)

{

i++;

}

else

{

j++;

}

}

return s;

}

};