# 题目

给你两个数组，arr1和 arr2，

arr2 中的元素各不相同

arr2中的每个元素都出现在 arr1中

对arr1中的元素进行排序，使arr1中项的相对顺序和 arr2中的相对顺序相同。未在 arr2 中出现过的元素需要按照升序放在 arr1 的末尾。

**示例：**

输入：arr1 = [2,3,1,3,2,4,6,7,9,2,19], arr2 = [2,1,4,3,9,6]

输出：[2,2,2,1,4,3,3,9,6,7,19]

**提示：**

arr1.length, arr2.length <= 1000

0 <= arr1[i], arr2[i] <= 1000

arr2 中的元素 arr2[i] 各不相同

arr2 中的每个元素 arr2[i] 都出现在 arr1 中

# 分析

## 方法一：交换+排序

**思路：**

数组一和数组二对比，相等则交换数组一中的数到前面去，最后sort(begin+tmp,end)即可。

**代码：**

vector<int> relativeSortArray(vector<int>& arr1, vector<int>& arr2) {

int tmp = 0;

for(int i = 0;i<arr2.size();++i)

for(int j = 0;j<arr1.size();++j){

if(arr1[j] == arr2[i]){

swap(arr1[j],arr1[tmp]); ++tmp;

}

}

sort(arr1.begin()+tmp,arr1.end());

return arr1;

}

## 方法二：计数排序

**思路：**

题目中明确数组内数字的范围为0 ~ 1000，据此可以考虑使用计数排序。首先使用一个长度为1001的数组counts统计数组arr1内每个数字的出现次数，之后根据题目要求先排序数组arr2内出现的数字，最后排序counts内剩下的数字。

**代码：**

class Solution {

public:

vector<int> relativeSortArray(vector<int>& arr1, vector<int>& arr2) {

vector<int> counts(1001, 0);

for (auto& n : arr1) {

counts[n]++;

}

int i = 0;

// 排序 arr2 内的数字

for (auto& n : arr2) {

while (counts[n] > 0) {

arr1[i++] = n;

counts[n]--;

}

}

// 排序剩下的数字

for (int j = 0; j < counts.size(); ++j) {

while (counts[j] > 0) {

arr1[i++] = j;

counts[j]--;

}

}

return arr1;

}

};

**复杂度：**

由于题目中已经明确辅助数组的长度，所以空间复杂度可以认为是O(1)，若数组arr1和arr2的长度分别为m和n，那么算法的总时间复杂度为O(n+m)。