350 Intersection of Two Arrays II

2018年4月10日 10:17

Question:

Given two arrays, write a function to compute their intersection.

Example:

Given *nums1* = [1, 2, 2, 1], *nums2* = [2, 2], return [2, 2].

Note:

- Each element in the result should appear as many times as it shows in both arrays.
- The result can be in any order.

Follow up:

- What if the given array is already sorted? How would you optimize your algorithm?
- What if *nums1*'s size is small compared to *nums2*'s size? Which algorithm is better?
- What if elements of *nums2* are stored on disk, and the memory is limited such that you cannot load all elements into the memory at once?

来自 <https://leetcode.com/problems/intersection-of-two-arrays-ii/description/>

给定两个数组，写一个方法来计算它们的交集。

例如:

给定 *nums1* = [1, 2, 2, 1], *nums2* = [2, 2], 返回 [2, 2].

注意:

- 输出结果中每个元素出现的次数，应与元素在两个数组中出现的次数一致。
- 我们可以不考虑输出结果的顺序。

跟进:

- 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，内存是有限的，你不能一次加载所有的元素到内存中，你该怎么办？

Solution for Python3:

```
class Solution1:
    def intersect(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: List[int]
        """
        d = {}
        for i in nums1:
           if d.get(i):
               d[i] += 1
           else:
               d[i] = 1
       res = []
        for i in nums2:
           if d.get(i):
               res.append(i)
               d[i] -= 1
               if not d[i]:
                   d.pop(i)
        return res

class Solution2:
    def intersect(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: List[int]
        """
        import collections
        a, b = map(collections.Counter, (nums1, nums2))
        return list((a&b).elements())

class Solution3:
    def intersect(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: List[int]
        """
        import collections
        C = collections.Counter
        return list((C(nums1) & C(nums2)).elements())

class Solution4:
    def intersect(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: List[int]
        """
        import collections
        return list((collections.Counter(nums1) &
collections.Counter(nums2)).elements())
```

Solution for C++:

```
class Solution1 {
public:
    vector<int> intersect(vector<int>& nums1, vector<int>& nums2) {
        unordered_map<int, int> map;
        for (int i : nums1) {
            map[i]++;
        }
        vector<int> res;
        for (int i : nums2) {
            if (map[i]-- > 0)
                res.push_back(i);
        }
        return res;
    }
};

class Solution2 {
public:
    vector<int> intersect(vector<int>& nums1, vector<int>& nums2) {
        sort(nums1.begin(), nums1.end());
        sort(nums2.begin(), nums2.end());
        nums1.erase(set_intersection(nums1.begin(), nums1.end(), nums2.begin(), nums2.end(), nums1.begin()), nums1.end());
        return nums1;
    }
};
```

Appendix:

Python collections.Counter:计数器，统计字符出现个数，dict子类

1) a=collections.Counter(list1)

2) b=collections.Counter(list2)

3) (a & b).elements() 返回a和b中共同出现的元素，包括重复出现，非unique元素集合，再通过list转换。

C++set_intersection （)

1) **set_intersection(a.begin(), a.end(),b.begin(),b.end(),a.begin())**

2) **a,b两个数组前提是有序，然后第5个参数是将ab数组的交集放在哪里**

3) **函数返回取得交集所放位置的最后一个元素的下一个元素。**