697 Degree of an Array

2018年5月3日 21:19

Given a non-empty array of non-negative integers nums, the **degree** of this array is defined as the maximum frequency of any one of its elements.
Your task is to find the smallest possible length of a (contiguous) subarray of nums, that has the same degree as nums.
Example 1:
Input: [1, 2, 2, 3, 1]
Output: 2
Explanation:
The input array has a degree of 2 because both elements 1 and 2 appear twice.
Of the subarrays that have the same degree:
[1, 2, 2, 3, 1], [1, 2, 2, 3], [2, 2, 3, 1], [1, 2, 2], [2, 2, 3], [2, 2]
The shortest length is 2. So return 2.
Example 2:
Input: [1,2,2,3,1,4,2]
Output: 6
Note:

- nums.length will be between 1 and 50,000.
- nums[i] will be an integer between 0 and 49,999.

来自 <https://leetcode.com/problems/degree-of-an-array/description/>

给定一个非空且只包含非负数的整数数组 nums, 数组的度的定义是指数组里任一元素出现频数的最大值。
你的任务是找到与 nums 拥有相同大小的度的最短连续子数组，返回其长度。
示例 1:
输入: [1, 2, 2, 3, 1]
输出: 2
解释:
输入数组的度是2，因为元素1和2的出现频数最大，均为2.
连续子数组里面拥有相同度的有如下所示:
[1, 2, 2, 3, 1], [1, 2, 2, 3], [2, 2, 3, 1], [1, 2, 2], [2, 2, 3], [2, 2]
最短连续子数组[2, 2]的长度为2，所以返回2.
示例 2:
输入: [1,2,2,3,1,4,2]
输出: 6
注意:

- nums.length 在1到50,000区间范围内。
- nums[i] 是一个在0到49,999范围内的整数。

Solution for Python3:

```python
class Solution1:
    def findShortestSubArray(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        left, right, cnt = {}, {}, {}
        for i, x in enumerate(nums):
           if x not in left:
               left[x] = i
           right[x] = i
           cnt[x] = cnt.get(x,0) + 1
        ans = len(nums)
        degree = max(cnt.values())
        for x in cnt:
           if cnt[x] == degree:
               ans = min(ans, right[x] - left[x] + 1)
        return ans

class Solution2:
    def findShortestSubArray(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        mp = {}
        for i, x in enumerate(nums):
           mp.setdefault(x, []).append(i)
        ans = len(nums)
        degree = max(len(i) for i in mp.values())
        for x in mp:
           if len(mp[x]) == degree:
               ans = min(ans, mp[x][-1] - mp[x][0] + 1)
        return ans
```

Solution for C++:

```cpp
class Solution1 {
public:
    int findShortestSubArray(vector<int>& nums) {
        unordered_map<int, int> left;
        unordered_map<int, int> right;
        unordered_map<int, int> cnt;
        for (int i = 0; i < nums.size(); i++) {
            int x = nums[i];
            if (left.count(x) == 0)
                left[x] = i;
            right[x] = i;
            cnt[x]++;
        }
        int ans = nums.size();
        int degree = 0;
        for (auto iter = cnt.begin(); iter != cnt.end(); iter++)
            if (iter->second > degree)
                degree = iter->second;
        for (auto iter = cnt.begin(); iter != cnt.end(); iter++)
            if (iter->second == degree)
                ans = min(ans, right[iter->first] - left[iter->first] + 1);
        return ans;
    }
};

class Solution2 {
public:
    int findShortestSubArray(vector<int>& nums) {
        unordered_map<int, vector<int>> mp;
        for (int i = 0; i < nums.size(); i++)
```

```cpp
            mp[nums[i]].push_back(i);
        int degree = 0;
        for (auto it = mp.begin(); it != mp.end(); it++)
            degree = max(degree, int(it->second.size()));
        int ans = nums.size();
        for (auto it = mp.begin(); it != mp.end(); it++) {
            if (it->second.size() == degree) {
                ans = min(ans, it->second.back() - it->second[0] + 1);
            }
        }
        return ans;
    }
};
```