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。

思路，暴力解法，直接标记开始和结束的位置，每个数出现的次数和度的大小，然后遍历一次数组查找最短的连续子数组，需注意标记开始时，需把结束同时标记，防止数字只出现一次的情况。

```
class Solution {
public:
    int findShortestSubArray(vector<int>& nums) {
        map<int,int> num;
        map<int,int> star;
        map<int,int> end;
        int n=nums.size();
        int max=0;
        int length=50000;
        for(int i=0;i<n;++i){
            ++num[nums[i]];
            max=max>num[nums[i]]?max:num[nums[i]];
            if(star[nums[i]]==0){
```

```
                star[nums[i]]=i+1;
                end[nums[i]]=i+1;
            }else{
                end[nums[i]]=i+1;
            }
        }
        for(int i=0;i<n;++i){
            if(num[nums[i]]==max){
                length=length>(end[nums[i]]-star[nums[i]]+1)?(end[nums[i]]-
star[nums[i]]+1):length;
            }
        }
        return length;
    }
};
```