594 Longest Harmonious Subsequence

2018年4月16日 13:26

We define a harmonious array is an array where the difference between its maximum value and its minimum value is **exactly** 1.

Now, given an integer array, you need to find the length of its longest harmonious subsequence among all its possible subsequences.

Example 1:

Input: [1,3,2,2,5,2,3,7]

Output: 5

Explanation: The longest harmonious subsequence is [3,2,2,2,3].

Note: The length of the input array will not exceed 20,000.

来自 <https://leetcode.com/problems/longest-harmonious-subsequence/description/>

和谐数组是指一个数组里元素的最大值和最小值之间的差别正好是1。

现在，给定一个整数数组，你需要在所有可能的子序列中找到最长的和谐子序列的长度。

示例 1:

输入: [1,3,2,2,5,2,3,7]

输出: 5

原因: 最长的和谐数组是：[3,2,2,2,3].

说明: 输入的数组长度最大不超过20,000.

Solution for Python3:

```
    class Solution1:
1       def findLHS(self, nums):
            """
2           :type nums: List[int]
            :rtype: int
3           """
            keyvalue = collections.Counter(nums)
4           maxV = float('-inf')
            for key in keyvalue:
5               maxV = max(maxV, keyvalue[key] +
    keyvalue.get(key-1, -keyvalue[key]))
6           return maxV

7   class Solution2:
        def findLHS(self, nums):
8           """
            :type nums: List[int]
9           :rtype: int
```

```
            """
            keyvalue = collections.Counter(nums)
            ans = 0
            for key in keyvalue:
                if key + 1 in keyvalue:
                    ans = max(ans, keyvalue[key] +
    keyvalue[key + 1])
            return ans;
```

```
class Solution1 {
public:
    int findLHS(vector<int>& nums) {
        unordered_map<int, int> m;
        for (auto i : nums)
            m[i]++;
        int res = 0;
        for (auto it : m) {
            if (m.count(it.first - 1) > 0)
                res = max(res, it.second +
m[it.first-1]);
        }
        return res;
    }
};

class Solution2 {
public:
    int findLHS(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        int len = 0;
        for (int i = 1, start  = 0, new_start = 0; i
< nums.size(); i++) {
            if (nums[i] - nums[start] > 1)
                start = new_start;
            if (nums[i] != nums[i-1])
                new_start = i;
            if (nums[i] - nums[start] == 1)
                len = max(len, i - start + 1);
        }
```

```
31          return len;
        }
    };
```