645 Set Mismatch

2018年4月21日 17:15

The set S originally contains numbers from 1 to n. But unfortunately, due to the data error, one of the numbers in the set got duplicated to **another** number in the set, which results in repetition of one number and loss of another number.
Given an array nums representing the data status of this set after the error. Your task is to firstly find the number occurs twice and then find the number that is missing. Return them in the form of an array.

Example 1:

Input: nums = [1,2,2,4]

Output: [2,3]

Note:

1. The given array size will in the range [2, 10000].
2. The given array's numbers won't have any order.

来自 <https://leetcode.com/problems/set-mismatch/description/>

集合 S 包含从1到 n 的整数。不幸的是，因为数据错误，导致集合里面某一个元素复制了成了集合里面的另外一个元素的值，导致集合丢失了一个整数并且有一个元素重复。
给定一个数组 nums 代表了集合 S 发生错误后的结果。你的任务是首先寻找到重复出现的整数，再找到丢失的整数，将它们以数组的形式返回。

示例 1:

输入: nums = [1,2,2,4]

输出: [2,3]

注意:

1. 给定数组的长度范围是 [2, 10000]。
2. 给定的数组是无序的。

Solution for Python3:

```python
class Solution1:
    def findErrorNums(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """
        dup, miss = -1, -1
        for n in nums:
            if nums[abs(n) - 1] < 0:
                dup = abs(n)
            else:
                nums[abs(n) - 1] *= -1;
        for i in range(len(nums)):
            if nums[i] > 0:
                miss = i + 1
        return [dup, miss]

class Solution2:
    def findErrorNums(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """
        return [sum(nums) - sum(set(nums)), sum(range(1, len(nums) + 1)) - sum(set(nums))]
```

Solution for C++:

```cpp
class Solution1 {
public:
    vector<int> findErrorNums(vector<int>& nums) {
        int n[nums.size()] = {0};
        for (int num : nums) {
```

```
            n[num-1]++;
        }
        int a, b;
        for (int i = 0; i < nums.size(); i++) {
            if (n[i] == 1)
                continue;
            if (n[i] == 0)
                b = i + 1;
            else
                a = i + 1;
        }
        return vector<int>{a, b};
    }
};

class Solution2 {
public:
    vector<int> findErrorNums(vector<int>& nums) {
        int dup = -1, miss = -1;
        for (int n : nums) {
            if (nums[abs(n) - 1] < 0)
                dup = abs(n);
            else
                nums[abs(n) - 1] *= -1;
        }
        for (int i = 0; i < nums.size(); i++)
            if (nums[i] > 0)
                miss = i + 1;
        return vector<int> {dup, miss};
    }
};
```