496 Next Greater Element I

2018年4月13日　　11:30

You are given two arrays **(without duplicates)** nums1 and nums2 where nums1’ s elements are subset of nums2. Find all the next greater numbers for nums1's elements in the corresponding places of nums2.
The Next Greater Number of a number **x** in nums1 is the first greater number to its right in nums2. If it does not exist, output -1 for this number.
Example 1:
Input: nums1 = [4,1,2], **nums2** = [1,3,4,2].
Output: [-1,3,-1]
Explanation:
For number 4 in the first array, you cannot find the next greater number for it in the second array, so output -1.
For number 1 in the first array, the next greater number for it in the second array is 3.
For number 2 in the first array, there is no next greater number for it in the second array, so output -1.
Example 2:
Input: nums1 = [2,4], **nums2** = [1,2,3,4].
Output: [3,-1]
Explanation:
For number 2 in the first array, the next greater number for it in the second array is 3.
For number 4 in the first array, there is no next greater number for it in the second array, so output -1.
Note:

1. All elements in nums1 and nums2 are unique.
2. The length of both nums1 and nums2 would not exceed 1000.

来自 <https://leetcode.com/problems/next-greater-element-i/description/>

给定两个**没有重复元素**的数组 nums1 和 nums2 ，其中nums1 是 nums2 的子集。找到 nums1 中每个元素在 nums2 中的下一个比其大的值。
nums1 中数字 **x** 的下一个更大元素是指 **x** 在 nums2 中对应位置的右边的第一个比 **x** 大的元素。如果不存在，对应位置输出-1。
注意:

1. nums1和nums2中所有元素是唯一的。
2. nums1和nums2 的数组大小都不超过1000。

Solution for Python3:

```
class Solution1:
    def nextGreaterElement(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: List[int]
        """
        return [next((y for y in nums2[nums2.index(x):] if y > x), -1) for
x in nums1]

class Solution2:
    def nextGreaterElement(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: List[int]
```

```python
        """
        d = {}
        deq = collections.deque()
        for num in nums2:
            while (deq and deq[-1] < num):
                d[deq.pop()] = num
            deq.append(num)
        for i in range(len(nums1)):
            nums1[i] = d.get(nums1[i], -1);
        return nums1

class Solution3:
    def nextGreaterElement(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: List[int]
        """
        st, d = [], {}
        for num in nums2:
            while len(st) and st[-1] < num:
                d[st.pop()] = num
            st.append(num)
        return list(map(lambda x: d.get(x, -1), nums1))
```

Solution for C++:

```cpp
class Solution {
public:
    vector<int> nextGreaterElement(vector<int>& findNums, vector<int>&
nums) {
        stack<int> s;
        unordered_map<int, int> map;
        for (int num : nums) {
            while (!s.empty() && s.top() < num) {
                map[s.top()] = num;
                s.pop();
            }
            s.push(num);
        }
        for (int i = 0; i < findNums.size(); i++) {
            findNums[i] = map.count(findNums[i]) ? map[findNums[i]] : -1;
        }
        return findNums;
    }
};
```