

```

1  /*
2  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。
3
4  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。
5
6  示例 1:
7
8  输入: nums1 = [4,1,2], nums2 = [1,3,4,2].
9  输出: [-1,3,-1]
10 解释:
11      对于num1中的数字4，你无法在第二个数组中找到下一个更大的数字，因此输出 -1。
12      对于num1中的数字1，第二个数组中数字1右边的下一个较大数字是 3。
13      对于num1中的数字2，第二个数组中没有下一个更大的数字，因此输出 -1。
14
15 示例 2:
16
17 输入: nums1 = [2,4], nums2 = [1,2,3,4].
18 输出: [3,-1]
19 解释:
20      对于num1中的数字2，第二个数组中的下一个较大数字是3。
21      对于num1中的数字4，第二个数组中没有下一个更大的数字，因此输出 -1。
22
23 注意:
24
25      nums1和nums2中所有元素是唯一的。
26      nums1和nums2  的数组大小都不超过1000。
27
28 来源：力扣（LeetCode）
29 链接：https://leetcode-cn.com/problems/next-greater-element-i
30 著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
31 */

```

分析:

- 读完题目,此题的重点在于找到 $num1$ 任意一个元素在 $num2$ 的索引,可以用暴力法或map法.若 $num1$ 的长度为 m , $num2$ 的长度为 n
 - 暴力法:时间复杂度 $O(m * n^2)$,空间复杂度 $O(1)$
 - map法:事件复杂度 $O(m * n)$,空间复杂度为 $O(n)$, 不过结果貌似没改善 , 莫非map的查找操作的时间复杂度不是 $O(1)$

方法一:C++_暴力法

```

1  class solution
2  {
3
4
5      private:
6          int __findIndex(int data , vector<int>& nums)
7          {

```

```

8
9         for(int i = 0; i < nums.size();i++)
10        {
11            if(data == nums[i])
12            {
13                return i;
14            }
15        }
16
17
18        return 2147483647;
19
20
21
22    }
23
24
25    public:
26        vector<int> nextGreaterElement(vector<int>& nums1, vector<int>&
nums2)
27    {
28
29        vector<int>         ret_val         ;
30        int                 i               = 0       ;
31        int                 j               = 0       ;
32        int                 temp            = 0       ;
33
34
35        for(i=0;i<nums1.size();i++)
36        {
37            temp = -1;
38            for(j=__findIndex(nums1[i],nums2);j<nums2.size();j++)
39            {
40                if(nums2[j] > nums1[i])
41                {
42                    temp = nums2[j];
43                    break;
44                }
45            }
46            ret_val.push_back(temp);
47        }
48        return ret_val;
49    }
50 };
51
52
53 /*
54 执行结果:
55 通过
56 显示详情
57 执行用时 :20 ms, 在所有 C++ 提交中击败了56.04% 的用户
58 内存消耗 :9 MB, 在所有 C++ 提交中击败了69.08%的用户
59 */

```

方法二:C++_map法

```

1  class solution
2  {
3
4
5      public:
6      vector<int> nextGreaterElement(vector<int>& nums1, vector<int>&
nums2)
7      {
8
9          vector<int>      ret_val      ;
10         int              temp        = 0    ;
11         int              i          = 0    ;
12         int              j          = 0    ;
13         map<int,int>      mii        ;      /*字典也可以*/
14
15         for (i = 0; i < nums2.size(); i++)
16         {
17             mii[nums2[i]] = i;
18         }
19
20         for(i=0;i<nums1.size();i++)
21         {
22             temp = -1;
23             for(j = mii[nums1[i]];j<nums2.size();j++)
24             {
25                 if(nums2[j] > nums1[i])
26                 {
27                     temp = nums2[j];
28                     break;
29                 }
30             }
31             ret_val.push_back(temp);
32         }
33         return ret_val;
34     }
35 };
36 /*
37 执行结果:
38 通过
39 显示详情
40 执行用时 :16 ms, 在所有 C++ 提交中击败了81.93% 的用户
41 内存消耗 :9.3 MB, 在所有 C++ 提交中击败了53.31%的用户
42 */

```