```
/*
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。

示例 1:

输入: nums1 = [4,1,2], nums2 = [1,3,4,2].
输出: [-1,3,-1]
解释:
    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。
    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。
    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。

示例 2:

输入: nums1 = [2,4], nums2 = [1,2,3,4].
输出: [3,-1]
解释:
    对于num1中的数字2，第二个数组中的下一个较大数字是3。
    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。

注意:

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

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/next-greater-element-i
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。
*/
```

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

```
class Solution
{


    private:
        int __findIndex(int data , vector<int>& nums)
        {
```

```
            for(int i = 0; i < nums.size();i++)
            {
                if(data == nums[i])
                {
                    return i;
                }
            }


            return 2147483647;



        }


    public:
        vector<int> nextGreaterElement(vector<int>& nums1, vector<int>&
nums2)
        {

            vector<int>         ret_val         ;
            int                 i       = 0     ;
            int                 j       = 0     ;
            int                 temp    = 0     ;


            for(i=0;i<nums1.size();i++)
            {
                temp = -1;
                for(j=__findIndex(nums1[i],nums2);j<nums2.size();j++)
                {
                    if(nums2[j] > nums1[i])
                    {
                        temp = nums2[j];
                        break;
                    }
                }
                ret_val.push_back(temp);
            }
            return ret_val;
        }
};


/*
执行结果:
通过
显示详情
执行用时 :20 ms, 在所有 C++ 提交中击败了56.04% 的用户
内存消耗 :9 MB, 在所有 C++ 提交中击败了69.08%的用户
*/
```

方法二:C++_map法

```cpp
class Solution
{


    public:
        vector<int> nextGreaterElement(vector<int>& nums1, vector<int>&
nums2)
        {

            vector<int>         ret_val         ;
            int                 temp    = 0     ;
            int                 i       = 0     ;
            int                 j       = 0     ;
            map<int,int>        mii             ;   /*字典也可以*/

            for (i = 0; i < nums2.size(); i++)
            {
                mii[nums2[i]] = i;
            }

            for(i=0;i<nums1.size();i++)
            {
                temp = -1;
                for(j = mii[nums1[i]];j<nums2.size();j++)
                {
                    if(nums2[j] > nums1[i])
                    {
                        temp = nums2[j];
                        break;
                    }
                }
                ret_val.push_back(temp);
            }
            return ret_val;
        }
};
/*
执行结果：
通过
显示详情
执行用时 :16 ms, 在所有 C++ 提交中击败了81.93% 的用户
内存消耗 :9.3 MB, 在所有 C++ 提交中击败了53.31%的用户
*/
```

AlimyBreak
2019.09.17