# 题目

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 。

**提示：**

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。

类似题目：

503. 下一个更大元素 II

1856. 子数组最小乘积的最大值

# 分析

## 方法一：暴力破解

**思路：**

对于nums1的每个元素，在nums2中找到它，假设它的下标为i，那么从第i+1位开始搜索是否存在比它大的，找到就将该值加入记录中，否则加入-1。

**代码：**

class Solution {

public:

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

vector<int> res;

for(int n: nums1){

int i = 0;

while(nums2[i] != n) i++;

i++;

for(;i<nums2.size();i++){

if(nums2[i] > n){

res.push\_back(nums2[i]);

break;

}

}

if(i == nums2.size()) res.push\_back(-1);

}

return res;

}

};

## 方法二：单调栈

**思路：**与最小栈类似（都是维护一个满足需求的栈）

对于nums2的元素，维护一个单调不增的栈。

依次扫描nums2的元素，假如扫到第i位，而且当前栈顶元素小于第i位元素，则栈顶元素找到了下一个比它大的元素。将其记录到哈希表中。否则将第i位元素入栈。

当扫描完nums2，留在栈中的元素都是没有比它大的元素存在的。

最后，对于nums1的每个元素，依次查询哈希表即可得到他们的下一个更大元素。

**代码：**

class Solution {

public:

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

vector<int> res;

unordered\_map<int, int> mp;

stack<int> sk;

for(int n: nums2){

while(!sk.empty() && sk.top() < n){

mp[sk.top()] = n;

sk.pop();//这个栈的作用只是为了临时比较

}

sk.push(n);

}

while(!sk.empty()){

mp[sk.top()] = -1;

sk.pop(); //不要遗漏，否则赋值错误

}

for(int n: nums1){

res.push\_back(mp[n]);

//根据key值直接找到map中对应的下一个元素值

}

return res;

}

};