**难度: 简单**

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。

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。

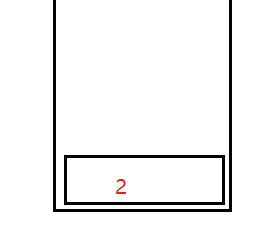
**思路1(使用的是暴力解法):**

1). 先创建存放的结果集数组

2). 获取nums1的值,在nums2的下标并且+1返回

3). 循环判断在nums2+1下标的后的值,有没有大于nums1的值,有就存储对应的值,没用就存放-1

|  |
| --- |
| public class Solution {  public int[] nextGreaterElement(int[] nums1, int[] nums2) {  int[] res = new int[nums1.length];    boolean flag = false;  for (int i = 0; i < nums1.length; i++) {  int index = findNum2Index(nums1[i], nums2);  //比较下标是否越界,以及是否小于  if (index < nums2.length) {  for (int j = index; j < nums2.length; j++) {  if (nums1[i] < nums2[j]) {  //成立就赋值  res[i] = nums2[j];  flag = false;  break;  }  flag = true;  }  if (flag) {  res[i] = -1;  }  }else {  //不成立赋值-1  res[i] = -1;  }  }  return res;  }  /\*\*  \*  \* @param value 数组1的值  \* @param num2 数组2  \* @return 查询到了返回+1后的下标,查询不到直接返回-1  \*/  private int findNum2Index(int value, int[] num2) {  for (int i = 0; i < num2.length; i++) {  //找到对应的值  if (value == num2[i]) {  //下标+1返回  return i+1;  }  }  return -1;  }  } |

**思路2(栈+hashmap):**

**用1, 3, 4, 2来举例子for循环入栈**

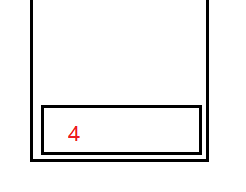
1). 从最后开始2开始

2). 第一此while循环判断栈为空,真,但是取反得到假,不执行

3). 此时获取value,判断是否为空,成立得到value为-1

4). 把值key和value放入到map中

5). **2入栈**

6). 此时4开始

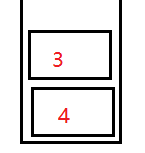
7). while判断栈为空,假,取反得到真,并且判断栈顶的值2 <= 4

8). 此时把**2抛出**

9). 判断栈是否为空,成立,value为-1

10). 把值key和value放入到map中

11). **4入栈**

 12). 此时3开始

13). while判断栈为空,假,取反得到真,并且判断栈顶的值4 <= 3 不成立

14). 判断栈是否为空,不成立,获取栈顶的值4,赋值

15). 把值key和value放入到map中

16). **3入栈**

17). 此时1开始

18). while判断栈为空,假,取反得到真,并且判断栈顶的值3 <= 1 不成立

19). 判断栈是否为空,不成立,获取栈顶的值3,赋值

20). 把值key和value放入到map中

21).  **1入栈**

|  |
| --- |
| public int[] nextGreaterElement2(int[] nums1, int[] nums2) {  //创建一个栈  Deque<Integer> stack = new ArrayDeque<>();  //创建一个hashmap  HashMap<Integer,Integer> map = new HashMap<>();  //创建一个结果集  int[] res = new int[nums1.length];    for (int i = nums2.length -1; i >= 0; i--) {  //栈不为空并且栈的值小于等于的值就进行pop  while(!stack.isEmpty() && stack.peek() <= nums2[i]) {  stack.pop();  }  //判断栈是否存在,不存在-1,存在,把对应的值返回  int value = stack.isEmpty() ? -1 : stack.peek();  //存放到map中  map.put(nums2[i],value);  //把当前值进行push  stack.push(nums2[i]);  }  //循环获取,把值存放到结果集中  for (int i = 0; i < nums1.length; i++) {  res[i] = map.get(nums1[i]);  }  //返回  return res;  } |