Next Greater Element I

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 a
rray 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 a
rray 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.

Solution 1

Key observation:
Suppose we have a decreasing sequence followed by a greater number
For example `[5, 4, 3, 2, 1, 6]` then the greater number `6` is the next greater element for all previous numbers in the sequence

We use a stack to keep a **decreasing** sub-sequence, whenever we see a number `x` greater than `stack.peek()` we pop all elements less than `x` and for all the popped ones, their next greater element is `x`
For example `[9, 8, 7, 3, 2, 1, 6]`
The stack will first contain `[9, 8, 7, 3, 2, 1]` and then we see `6` which is greater than `1` so we pop `1 2 3` whose next greater element should be `6`

```
    public int[] nextGreaterElement(int[] findNums, int[] nums) {
        Map<Integer, Integer> map = new HashMap<>(); // map from x to next greate
r element of x
        Stack<Integer> stack = new Stack<>();
        for (int num : nums) {
            while (!stack.isEmpty() && stack.peek() < num)
                map.put(stack.pop(), num);
            stack.push(num);
        }
        for (int i = 0; i < findNums.length; i++)
            findNums[i] = map.getOrDefault(findNums[i], -1);
        return findNums;
    }
```

written by yuxiangmusic original link here

Solution 2

```
public class Solution {
    public int[] nextGreaterElement(int[] findNums, int[] nums) {
        int[] ret = new int[findNums.length];
        ArrayDeque<Integer> stack = new ArrayDeque<>();
        HashMap<Integer, Integer> map = new HashMap<>();
        for(int i = nums.length - 1; i >= 0; i--) {
            while(!stack.isEmpty() && stack.peek() <= nums[i]) {
                stack.pop();
            }
            if(stack.isEmpty()) map.put(nums[i], -1);
            else map.put(nums[i], stack.peek());
            stack.push(nums[i]);
        }
        for(int i = 0; i < findNums.length; i++) {
            ret[i] = map.get(findNums[i]);
        }
        return ret;
    }
}
```

written by zhengyuyu123 original link here

Solution 3

```
def nextGreaterElement(self, findNums, nums):
    return [next((y for y in nums[nums.index(x):] if y > x), -1) for x in findNum
s]
```

written by StefanPochmann original link here

From Leetcoder.