Understanding the problem

18 February 2025 02:08

— Monotonic Stack: —

Q. What is monotonic stack.

Before this use stack to store the element

↳ But not store the element in specific order

- it may be inc
- ——— dec.
- ——— (any other order)

Que. of Stack.

496. Next Greater Element I

Attempted

Easy | Topics | Companies

The **next greater element** of some element `x` in an array is the **first greater** element that is **to the right** of `x` in the same array.

You are given two **distinct 0-indexed** integer arrays `nums1` and `nums2`, where `nums1` is a subset of `nums2`.

For each `0 <= i < nums1.length`, find the index `j` such that `nums1[i] == nums2[j]` and determine the **next greater element** of `nums2[j]` in `nums2`. If there is no next greater element, then the answer for this query is `-1`.

Return *an array* `ans` *of length* `nums1.length` *such that* `ans[i]` *is the* ***next greater element*** *as described above.*

Example

Page No 2 Example.

18 February 2025 02:38

Input: nums1 = [4,1,2], nums2 = [1,3,4,2]
Output: [-1,3,-1]

→ this actual question

But Sir tough us :-

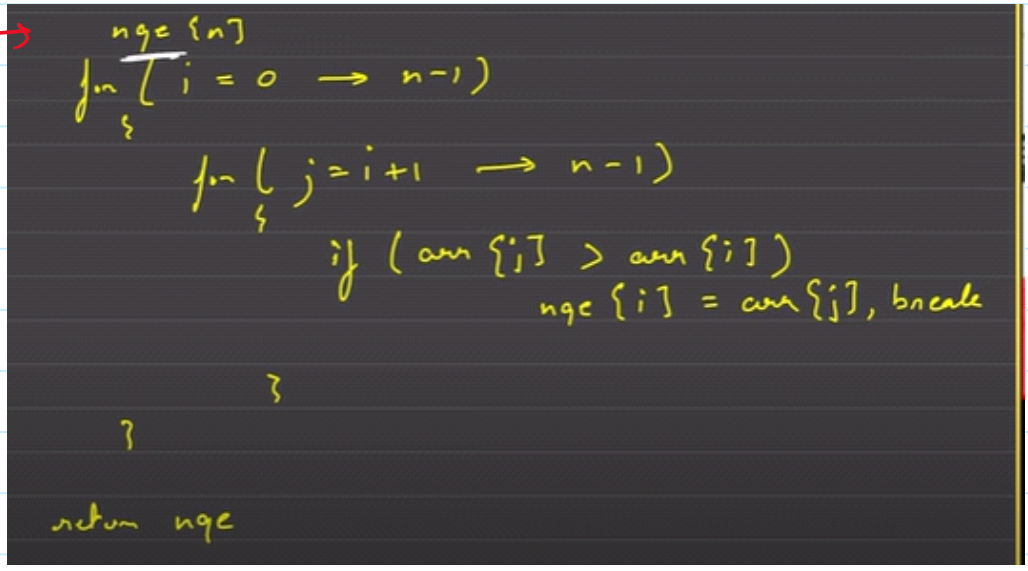
Example - [6, 0, 8, 2, 3]

O/P [8, 8, -1, 3, -1]

O/P

Method 1. Solve this problem

New one.



TC - O(N²)

SC - O(N) → for store Answer.

→ Not happy to reduce time complexity

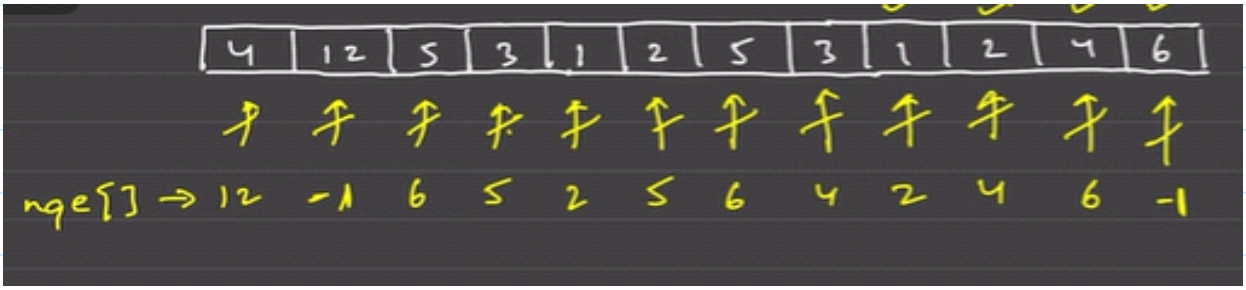
O(n²) → O(N)

18 February 2025 02:44

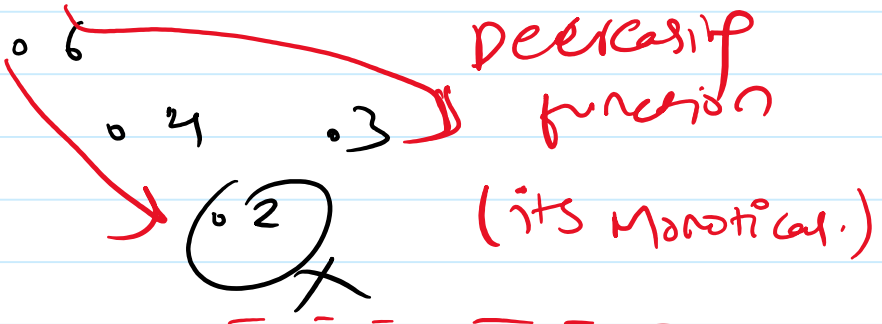
Example to use the monotonic stack

(4, 12, 5, 3, 1, 2, 5, 3, 1, 2, 4, 6)

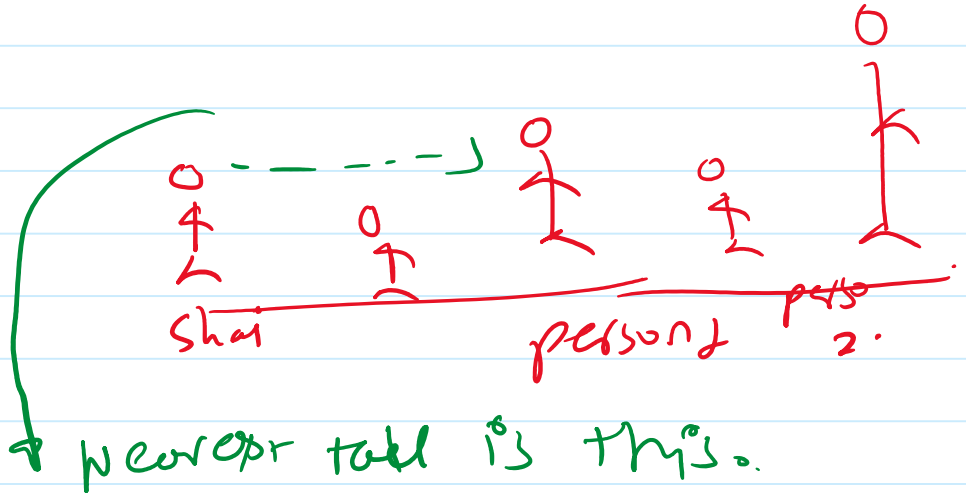
$O(N^2) \rightarrow O(N)$



instead of storing the complete value on stack use monotonic function.



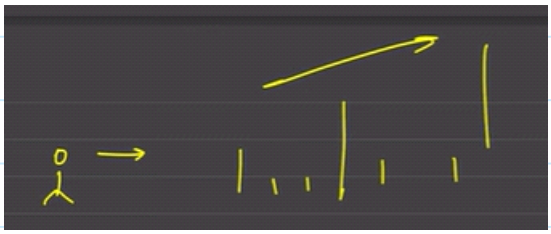
Consider similar to —



Recursor told is this.

Code :-

How to intuition come




```
list<int> findNGE(arr[])                    TC →
{
    nge[n] , stack st

    for(i = n-1 → 0)  ──→ O(N)
    {
        while( !st.empty() && st.top <= arr[i])
            st.pop();  ──→ O(N)

        if(st.empty())  nge[i] = -1
        else            nge[i] = st.top()

        st.push(arr[i])
    }

    return nge;
}
```

TC → O(2N)
↳ worst case

SC → O(N) + O(N)
↳ Ans
↳ Stack

Solution :

Sol

18 February 2025 10:58

Method -1.

```
class Solution {
    public int[] nextGreaterElement(int[] nums1, int[] nums2) {
        int[] ans=new int[nums1.length];
            Arrays.fill(ans,-1);
         for(int i=0;i<nums1.length;i++){

            for(int j=0;j<nums2.length;j++){
                if(nums1[i]==nums2[j]){
                    for(int k=j+1;k<nums2.length;k++){
                        if(nums2[j]<nums2[k]){
                            ans[i]=nums2[k];
                            break;
                        }
                    }



                }
            }
         }

         return ans;
    }
}
```

TC → O(N^3)

Method 2.

```
19      public static int[] nGE(int[] arr){
20          int[] ans=new int[arr.length];
21          Stack<Integer> stack =new Stack<>();
22          for(int i=arr.length-1;i>=0;i--){
23              while(stack.isEmpty()==false && arr[i]>=stack.peek())
24              stack.pop();
25
26              if(stack.isEmpty())
27              ans[i]=-1;
28              else{
29                  ans[i]=stack.peek();
30              }
31              stack.push(arr[i]);
32          }
33
34          return ans;
35      }
```

```
2       public int[] nextGreaterElement(int[] nums1, int[] nums2) {
3          int[] ans=new int[nums1.length];
4
5           int[] arr=nGE(nums2);
6           for(int i=0;i<nums1.length;i++){
7               for(int j=0;j<nums2.length;j++){
8                   if(nums1[i]==nums2[j])
9                   {
10                      ans[i]=arr[j];
11                      break;
12                  }
13              }
14          }
15
16          return ans;
17      }
```