Problem | Submissions | Comments

Number of NGEs to the right

Easy | Accuracy: 75.68% | Submissions: 2K+ | Points: 2

Given an array of **N** integers and **Q** queries of indices, print the number of next greater elements(NGEs) to the right of the given index element.

Example:

Input: arr = [3, 4, 2, 7, 5, 8, 10, 6]
queries = 2
indices = [0, 5]

Output: 6, 1

Explanation:

The next greater elements to the right of 3(index 0) are 4,7,5,8,10,6.

The next greater elements to the right of 8(index 5) is only 10.

C++ (g++ 5.4) | Average Time: 10m | Start Timer

```cpp
// } Driver Code Ends
//User function Template for C++

class Solution{
public:

    vector<int> count_NGE(int n, vector<int> &arr, int queries, vector<int> &indices){
        //write your code here
        vector<int>ans;
        for(int i = 0 ;  i< queries ; i++){
            int cnt = 0;
            for(int j = indices[i] ; j<n ; j++){
                if(arr[indices[i]] < arr[j]){
                    cnt++;
                }
            }
            ans.push_back(cnt);
        }
        return ans;

    }

};
// } Driver Code Ends
```

Custom Input | Compile & Run | Submit

Remove K Digits

Medium Accuracy: 26.8% Submissions: 16K+ Points: 4

Given a non-negative integer S represented as a string, remove K digits from the number so that the new number is the smallest possible.

Note : The given *num* does not contain any leading zero.

Example 1:

Input:
S = "149811", K = 3
Output: 111
Explanation: Remove the three digits 4, 9, and 8 to form the new number 111 which is smallest.

Example 2:

Input:

C++ (g++ 5.4) | Start Timer

```
// } Driver Code Ends
// User function Template for C++

class Solution {
  public:
    string removeKdigits(string S, int k) {
        string res = "";
        for (char c : S) {

            while (res.length() && res.back() > c && k) {
                res.pop_back();
                k--;
            }

            if (res.length() || c != '0') {
                res.push_back(c);
            }
        }

        while (res.length() && k--) {
            res.pop_back();
        }
        return res.empty() ? "0" : res;
    }
};
// } Driver Code Ends
```

Custom Input | Compile & Run | Submit

Problem | Submissions | Comments

Sum of subarray ranges

Medium | Accuracy: 70.41% | Submissions: 727+ | Points: 4

Given an integer array **arr** of size **N**. The **Range** of a subarray of arr is the difference between the largest and smaller element in the subarray. Return the **sum** of all subarray ranges of arr.

Example 1:

Input:
N = 3
arr[ ] = {1, 2, 3}
Output: 4
Explanation: The 6 subarrays of arr are the following :
{1 } , range = largest - smallest = 1 - 1 = 0
{2 } , range = 2 - 2 = 0
{3 } , range = 3 - 3 = 0
{1, 2}, range = 2 - 1 = 1
{2, 3}, range = 3 - 2 = 1

C++ (g++ 5.4) | Average Time: 30m | Start Timer

```cpp
// } Driver Code Ends
class Solution {
  public:
    long long subarrayRanges(int N, vector<int> &arr) {
        // code here
        int res=0;
        for(int i=0 ; i< N-1 ; i++){
            int mini=arr[i];
            int maxi=arr[i];
            for(int j = i+1; j<N ; j++){
                mini= min(mini,arr[j]);
                maxi= max(maxi,arr[j]);
                res+= (maxi-mini);
            }
        }
        return res;
    }
};
// } Driver Code Ends
```

Custom Input | Compile & Run | Submit