**1793. Maximum Score of a Good Subarray: -**

Hard Accepted: 42.2K Submissions: 69.3K Acceptance Rate: 60.9%

You are given an array of integers nums **(0-indexed)** and an integer k.

The **score** of a subarray (i, j) is defined as min(nums[i], nums[i+1], ..., nums[j]) \* (j - i + 1). A **good** subarray is a subarray where i <= k <= j.

Return *the maximum possible****score****of a****good****subarray.*

**Example 1:**

**Input:** nums = [1,4,3,7,4,5], k = 3

**Output:** 15

**Explanation:** The optimal subarray is (1, 5) with a score of min(4,3,7,4,5) \* (5-1+1) = 3 \* 5 = 15.

**Example 2:**

**Input:** nums = [5,5,4,5,4,1,1,1], k = 0

**Output:** 20

**Explanation:** The optimal subarray is (0, 4) with a score of min(5,5,4,5,4) \* (4-0+1) = 4 \* 5 = 20.

**Constraints:**

* 1 <= nums.length <= 105
* 1 <= nums[i] <= 2 \* 104
* 0 <= k < nums.length

**Code: -**

class Solution {

public:

    vector<int> leftsmall(vector<int> &nums, int n){

        stack<int> s;

        vector<int> ans;

        for(int i = 0; i < n; ++i){

            while(s.size() and nums[s.top()] >= nums[i])

                s.pop();

            if(s.empty())   ans.push\_back(0);

            else            ans.push\_back(s.top() + 1);

            s.push(i);

        }

        return ans;

    }

    vector<int> rightsmall(vector<int> &nums, int n){

        stack<int> s;

        vector<int> ans(n);

        for(int i = n-1; i >= 0; --i){

            while(s.size() and nums[s.top()] >= nums[i])

                s.pop();

            if(s.empty())   ans[i] = n-1;

            else            ans[i] = s.top() - 1;

            s.push(i);

        }

        return ans;

    }

    int maximumScore(vector<int>& nums, int k) {

        int n = nums.size();

        vector<int> left = leftsmall(nums, n);

        vector<int> right = rightsmall(nums, n);

        int ans = INT\_MIN;

        for(int i = 0; i < n; ++i){

            if(left[i] <= k and k <= right[i])

                ans = max(ans, nums[i] \* (right[i] - left[i] + 1));

        }

        return ans;

    }

};

**T.C: - O(N)**

**S.C: - O(N)**