Maximum product of indexes of next greater on left and right

Given an array a[1..N]. For each element at position i (1 <= i <= N). Where

1. **L(i)** is defined as closest index j such that j < i and a[j] > a[i]. If no such j exists then **L(i) = 0**.
2. **R(i)** is defined as closest index k such that k > i and a[k] > a[i]. If no such k exists then **R(i) = 0**.

LRProduct(i) = L(i)*R(i) .

We need to find an index with maximum LRProduct

Examples:

Input : 1 1 1 1 0 1 1 1 1 1
Output : 24
For {1, 1, 1, 1, 0, 1, 1, 1, 1, 1} all element are same except 0. So only for zero their exist greater element and for others it will be zero. for zero, on left 4th element is closest and greater than zero and on right 6th element is closest and greater. so maximum
*product will be 4*6 = 24.*

Input : 5 4 3 4 5
Output : 8
For {5, 4, 3, 4, 5}, L[] = {0, 1, 2, 1, 0} and R[]

= {0, 5, 4, 5, 0},
LRProduct = {0, 5, 8, 5, 0} and max in this is 8.

Note: Taking starting index as 1 for finding LRproduct.

Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

This problem is based on Next Greater Element.

From the current position, we need to find the closest greater element on its left and right side.
So to find next greater element, we used stack one from left and one from right.simply we are checking which element is greater and storing their index at specified position.
1- if stack is empty, push current index.
2- if stack is not empty
….a) if current element is greater than top element then store the index of current element on index of top element.

Do this, once traversing array element from left and once from right and form the left and right array, then, multiply them to find max product value.

```
// C++ program to find the max
// LRproduct[i] among all i
#include <bits/stdc++.h>
using namespace std;
#define MAX 1000

// function to find just next greater
// element in left side
vector<int> nextGreaterInLeft(int a[], int n)
{
    vector<int> left_index(n, 0);
    stack<int> s;

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

        // checking if current element is greater than top
        while (!s.empty() && a[i] > a[s.top() - 1]) {
            int r = s.top();
            s.pop();

            // on index of top store the current element
```

```
            // index which is just greater than top element
            left_index[r - 1] = i + 1;
        }

        // else push the current element in stack
        s.push(i + 1);
    }
    return left_index;
}

// function to find just next greater element
// in right side
vector<int> nextGreaterInRight(int a[], int n)
{
    vector<int> right_index(n, 0);
    stack<int> s;
    for (int i = 0; i < n; ++i) {

        // checking if current element is greater than top
        while (!s.empty() && a[i] > a[s.top() - 1]) {
            int r = s.top();
            s.pop();

            // on index of top store the current element
            // index which is just greater than top element
            // stored index should be start with 1
            right_index[r - 1] = i + 1;
        }

        // else push the current element in stack
        s.push(i + 1);
    }
    return right_index;
}

// Function to find maximum LR product
int LRProduct(int arr[], int n)
{
    // for each element storing the index of just
    // greater element in left side
    vector<int> left = nextGreaterInLeft(arr, n);

    // for each element storing the index of just
    // greater element in right side
    vector<int> right = nextGreaterInRight(arr, n);
    int ans = -1;
    for (int i = 1; i <= n; i++) {

        // finding the max index product
        ans = max(ans, left[i] * right[i]);
    }

    return ans;
}

// Drivers code
int main()
```

```
{
    int arr[] = {  5, 9, 6, 8, 6, 4, 6, 9, 5, 4, 9 };
    int n = sizeof(arr) / sizeof(arr[1]);

    cout << LRProduct(arr, n);

    return 0;
}
```

```
99
```

Java

```
// Java program to find the
// max LRproduct[i] among all i
import java.io.*;
import java.util.*;

class GFG
{
    static int MAX = 1000;

    // function to find just next
    // greater element in left side
    static int[] nextGreaterInLeft(int []a,
                                   int n)
    {
        int []left_index = new int[MAX];
        Stack<Integer> s = new Stack<Integer>();

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

            // checking if current
            // element is greater than top
            while (s.size() != 0 &&
                     a[i] > a[s.peek() - 1])
            {
                int r = s.peek();
                s.pop();

                // on index of top store
                // the current element
                // index which is just
                // greater than top element
                left_index[r - 1] = i + 1;
            }

            // else push the current
            // element in stack
```