https://course.acciojob.com/idle?question=a11eac7c-f247-409b-851e-7e5bc94bc2ca

- MEDIUM
- Max Score: 40 Points

Sum of Subarray Minimums

Given an array of integers `arr`, find the sum of the minimums of all contiguous subarrays of the array. Since the answer may be large, print the answer modulo 10^9 + 7.

Your task is to complete the function `minSubarraySum` which receives `arr` and size of array as parameters and returns the sum of the minimums of all contiguous subarrays of the array

Input Format

The first line contains a single integer `n`(size of the array).

The second line contains `n` space integers that denote the elements of the array.

Output Format

Print the sum of minimum of all subarrays.

Example 1

Input

```
4
3 1 2 4
```

Output

```
17
```

Explanation

Subarrays are [3], [1], [2], [4], [3,1], [1,2], [2,4], [3,1,2], [1,2,4], [3,1,2,4].

Minimums are 3, 1, 2, 4, 1, 1, 2, 1, 1, 1.

Sum is 17.

Example 2

Input

```
3
1 2 3
```

Output

```
10
```

Explanation

Subarrays are [1], [2], [3], [1,2], [2,3], [1,2,3].

Minimums are 1, 2, 3, 1, 2, 1.

Sum is 10.

Constraints

1 <= arr.length <= 3 * 10^4

1 <= arr[i] <= 3 * 10^4

Topic Tags

- Stacks

My code

```
// n java
import java.util.*;
import java.lang.*;
import java.io.*;

public class Main
{
	public static void main (String[] args) throws
java.lang.Exception
	{
		//your code here
		Scanner sc= new Scanner(System.in);
		int   n= sc.nextInt();
		int[]arr= new int[n];
		for(int i=0;i<n;i++){
		   arr[i]=sc.nextInt();

		}
		  int ans= sumOfSubarrayMinimums(arr);
		System.out.print(ans);

	}
```

```
        public static int sumOfSubarrayMinimums(int arr[]){

    int sol[]=biggerOnLeft(arr);
    int sor[]=biggerOnRight(arr);
    long ans=0;
    int mod=(int)1e9+7;
    for(int i=0;i<arr.length;i++){
        int noOfSubarrays=(sol[i]+1)*(sor[i]+1);
        ans+=noOfSubarrays*arr[i];
        ans=ans%mod;
    }
    return (int)ans;
}
public static int[] biggerOnLeft(int arr[]){
    int n=arr.length;
    int ans[]=new int[n];
    Stack<Integer>st=new Stack<>();
    for(int i=0;i<n;i++){
        while(st.size()>0&&arr[st.peek()]>arr[i]){
            st.pop();
        }
        if(st.size()==0){
            ans[i]=i;
        }else{
            ans[i]=i-st.peek()-1;
        }
        st.push(i);
    }
    return ans;
}
```

```
    public static int[] biggerOnRight(int arr[]){
        int n=arr.length;
        int ans[]=new int[n];
        Stack<Integer>st=new Stack<>();
        for(int i=n-1;i>=0;i--){
            while(st.size()>0&&arr[st.peek()]>=arr[i]){
                st.pop();
            }
            if(st.size()==0){
                ans[i]=n-i-1;
            }else{

                ans[i]=st.peek()-i-1;
            }
            st.push(i);
        }

        return ans;
    }
}
```