SUM OF SUBARAY MINIMUM

Problem asks us to return the sum of minimum elements of every subarray given an array of integers.

Brute force solution is to obviously run two nested loops where for each element we iterate through all subarrays and just add the minimum element to a sum.

Time Complexity is $O(N^2)$.

Optimal Solution involves use of the logic of previous smaller & next smaller.

What we can do is, we can find the previous smaller and next smaller element for each element. This gives us the range where our element will contribute to the sum.

Eg:

$$1 \quad \underbrace{4 \quad 6 \quad 7}_{4} \quad \underbrace{\overset{\downarrow}{3} \quad 7 \quad 8}_{3} \quad 1$$

Total Subarrays in which 3 is minimum = 12

∴ Contribution of 3 to sum is 36 (12 × 3)

Pseudocode :

```
sumOfSubarrayMinimum(arr, N) {
   nse = findNSE(arr, N);
   pse = findPSE(arr, N);
   total = 0 ;
   mod = (int)(1e9 + 7) ;
   for (int i = 0 → N - 1) {
       left = i - pse[i];
       right = nse[i] - i ;
       total += (left * right * arr[i]) % mod
   }
}
// Returns array
findNSE (arr, N) {
   stack <int> st ;
   ans [N] ;
   for (i = N - 1 → 0) {
      while (!st.empty() && arr[st.top()] >= arr[i]) {
          st.pop();
      }
      ans[i] = st.empty() ? n : st.top();
      st.push(i) ;
   }
   return ans ;
}

findPSE (arr, N) {
   stack <int> st ;
```

```
    ans [N] ;
    for (i = 0 → N - 1)   {
        while (!st.empty () && arr[st.top()]
                                    > arr[i]){
            st.pop() ;
        }
        ans [i] = st.empty() ? -1 : st.top();
        st.push (i) ;
    }
    return  ans ;
}
```

Time Complexity in total is O(5N) and same as space complexit