Bubble Sort 2

Bubble sort is an algorithm to sort a sequence. Let's say we are going to sort a sequence $A_0, A_1, \ldots, A_{N-1}$ of length N in non-decreasing order. Bubble sort swaps two adjacent numbers when they are not in the correct order. Swaps are done by repeatedly passing through the sequence. Precisely speaking, in a **pass**, we swap A_i and A_{i+1} if $A_i > A_{i+1}$, for $i = 0, 1, \ldots, N-2$ in this order. It is known that any sequence can be sorted in non-decreasing order by some passes. For a sequence A, we define the **number of passes by bubble sort** as the number of passes needed to sort A using the above algorithm.

JOI-kun has a sequence A of length N. He is going to process Q queries of modifying values of A. To be specific, in the $(j+1)$-th query $(0 \leq j \leq Q-1)$, the value of A_{X_j} is changed into V_j.

JOI-kun wants to know the number of passes by bubble sort for the sequence after processing each query.

Example

Given a sequence $A = \{1, 2, 3, 4\}$ of length $N = 4$ and $Q = 2$ queries: $X = \{0, 2\}$, $V = \{3, 1\}$.

1. For the first query, the value of A_0 is changed into 3. We obtain $A = \{3, 2, 3, 4\}$.
2. For the second query, the value of A_2 is changed into 1. We obtain $A = \{3, 2, 1, 4\}$.

Bubble sort for $A = \{3, 2, 3, 4\}$:

- A is not sorted, so the first pass starts. Since $A_0 > A_1$, we swap them to get $A = \{2, 3, 3, 4\}$. Since $A_1 \leq A_2$, we don't swap them. Since $A_2 \leq A_3$, we don't swap them.
- Now A is sorted, so the bubble sort ends.

Hence, the number of passes by bubble sort is 1 for $A = \{3, 2, 3, 4\}$.

Bubble sort for $A = \{3, 2, 1, 4\}$:

- A is not sorted, so the first pass starts. Since $A_0 > A_1$, we swap them to get $A = \{2, 3, 1, 4\}$. Since $A_1 > A_2$, we swap them to get $A = \{2, 1, 3, 4\}$. Since $A_2 \leq A_3$, we don't swap them.
- A is not sorted yet, so the second pass starts. Since $A_0 > A_1$, we swap them to get $A = \{1, 2, 3, 4\}$. Since $A_1 \leq A_2$, we don't swap them. Since $A_2 \leq A_3$, we don't swap them.
- Now A is sorted, so the bubble sort ends.

Hence, then number of passes by bubble sort is 2 for $A = \{3, 2, 1, 4\}$.

Subtasks

There are 4 subtasks. The score and the constraints for each subtask are as follows:

Subtask	Score	N	Q	A, V
1	17	$1 \leq N \leq 2\,000$	$1 \leq Q \leq 2\,000$	$1 \leq A_i, V_j \leq 1\,000\,000\,000$
2	21	$1 \leq N \leq 8\,000$	$1 \leq Q \leq 8\,000$	$1 \leq A_i, V_j \leq 1\,000\,000\,000$
3	22	$1 \leq N \leq 50\,000$	$1 \leq Q \leq 50\,000$	$1 \leq A_i, V_j \leq 100$
4	40	$1 \leq N \leq 500\,000$	$1 \leq Q \leq 500\,000$	$1 \leq A_i, V_j \leq 1\,000\,000\,000$

Implementation details

You should implement the following function `countScans` to answer Q queries.

- `countScans(A, X, V)`
  - `A`: array of integers of length N representing the initial values of the sequence.
  - `X`, `V`: arrays of integers of length Q representing queries.
  - This function should return an array S of integers of length Q. For each $0 \leq j \leq Q-1$, S_j should be the number of passes by bubble sort for the sequence right after processing $(j+1)$-th query.

Sample grader

The sample grader reads the input in the following format:

- line 1: N Q
- line 2: A_0 A_1 $\ldots$ A_{N-1}
- line $3 + j$ $(0 \leq j \leq Q-1)$: X_j V_j

The sample grader prints the return value of `countScans` in the following format:

- line $1 + j$ $(0 \leq j \leq Q-1)$: S_j