Another Sliding Window Problem

Time limit: *500 ms*
Memory limit: *64 MB*

Given a sequence $t_1, t_2, \ldots, t_k$, we are interested in grouping the elements into $\lceil \frac{k}{2} \rceil$ groups — each group will contain exactly 2 elements, and if k is odd there will be exactly one group with just one element. For each group we compute the sum of its elements. We'll call the maximum of these sums the *matching cost*.

Out of all the possible ways of grouping the elements, we are interested in the one that has the smallest *matching cost*. We will call this value the **optimal cost**.

For example, the optimal cost for $(5, 1, 3, 4)$ is 7, and we can get it by grouping the elements in $\{\{5, 1\}, \{3, 4\}\}$. Any other grouping has a larger *matching cost*.

You are given a **sorted** array A and you have to answer Q queries:

- Each query consists of a single integer x.
- For each subarray of A that has an **optimal cost** $\leq x$, add the difference between the rightmost and the leftmost elements of the subarray. Formally, calculate $\sum_{\substack{1 \leq l \leq r \leq N \\ \text{optimal cost}(A_l, \ldots, A_r) \leq x}} (A_r - A_l)$.

Standard input

The first line contains the integers N and Q, the number of elements in the sequence and the number of queries.

The second line contains the sequence of integers $A_1, A_2, \ldots, A_N$ separated by spaces.

The next Q lines, the integer x_i, the value of x for the i-th query, is found.

Standard output

For each query, return the value of the indicated summation.

Constraints and notes

- $1 \leq N \leq 10^5$.
- $1 \leq Q \leq 100$.
- $0 \leq A_i \leq 10^9$ for $1 \leq i \leq N$.
- $0 \leq x_i \leq 2 \times 10^9$ for $1 \leq i \leq Q$