E. Berland Local Positioning System

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In Berland a bus travels along the main street of the capital. The street begins from the main square and looks like a very long segment. There are n bus stops located along the street, the i-th of them is located at the distance a_i from the central square, all distances are distinct, the stops are numbered in the order of increasing distance from the square, that is, $a_i < a_{i+1}$ for all i from 1 to n - 1. The bus starts its journey from the first stop, it passes stops $2, 3$ and so on. It reaches the stop number n, turns around and goes in the opposite direction to stop 1, passing all the intermediate stops in the reverse order. After that, it again starts to move towards stop n. During the day, the bus runs non-stop on this route.

The bus is equipped with the Berland local positioning system. When the bus passes a stop, the system notes down its number.

One of the key features of the system is that it can respond to the queries about the distance covered by the bus for the parts of its path between some pair of stops. A special module of the system takes the input with the information about a set of stops on a segment of the path, a stop number occurs in the set as many times as the bus drove past it. This module returns the length of the traveled segment of the path (or -1 if it is impossible to determine the length uniquely). The operation of the module is complicated by the fact that *stop numbers occur in the request not in the order they were visited but in the non-decreasing order*.

For example, if the number of stops is 6, and the part of the bus path starts at the bus stop number 5, ends at the stop number 3 and passes the stops as follows: , then the request about this segment of the path will have form: $3, 4, 5, 5, 6$. If the bus on the segment of the path from stop 5 to stop 3 has time to drive past the 1-th stop (i.e., if we consider a segment that ends with the second visit to stop 3 on the way from 5), then the request will have form: $1, 2, 2, 3, 3, 4, 5, 5, 6$.

You will have to repeat the Berland programmers achievement and implement this function.

Input

The first line contains integer n ($2 \le n \le 2 \cdot 10^5$) — the number of stops.

The second line contains n integers ($1 \le a_i \le 10^9$) — the distance from the i-th stop to the central square. The numbers in the second line go in the increasing order.

The third line contains integer m ($1 \le m \le 4 \cdot 10^5$) — the number of stops the bus visited on some segment of the path.

The fourth line contains m integers ($1 \le b_i \le n$) — the sorted list of numbers of the stops visited by the bus on the segment of the path. The number of a stop occurs as many times as it was visited by a bus.

It is guaranteed that the query corresponds to some segment of the path.

Output

In the single line please print the distance covered by a bus. If it is impossible to determine it unambiguously, print -1.

Examples

input
6 2 3 5 7 11 13 5 3 4 5 5 6
output
10

input

```
6
2 3 5 7 11 13
9
1 2 2 3 3 4 5 5 6
```

output

```
16
```

input

```
3
10 200 300
4
1 2 2 3
```

output

```
-1
```

input

```
3
1 2 3
4
1 2 2 3
```

output

```
3
```

Note

The first test from the statement demonstrates the first example shown in the statement of the problem.

The second test from the statement demonstrates the second example shown in the statement of the problem.

In the third sample there are two possible paths that have distinct lengths, consequently, the sought length of the segment isn't defined uniquely.

In the fourth sample, even though two distinct paths correspond to the query, they have the same lengths, so the sought length of the segment is defined uniquely.