Yaroslav and Permutations

time limit per test : 2 seconds

memory limit per test : 256 megabytes

Yaroslav has an array that consists of n integers. In one second Yaroslav can swap two neighboring array elements. Now Yaroslav is wondering if he can obtain an array where any two neighboring elements would be distinct in a finite time.

Help Yaroslav.

Input

The first line contains integer n $(1 \le n \le 100)$ — the number of elements in the array. The second line contains n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 1000)$ — the array elements.

Output

In the single line print "YES" (without the quotes) if Yaroslav can obtain the array he needs and "NO" (without the quotes) otherwise.

Sample test(s)

input

```
1
1
```

output

```
YES
```

input

```
3
1 1 2
```

output

```
YES
```

input

```
4
7 7 7 7
```

output

```
NO
```

Note

In the first sample the initial array fits well.

In the second sample Yaroslav can get array: $1, 2, 1$. He can swap the last and the second last elements to obtain it.

In the third sample Yarosav can't get the array he needs.

Greg and Array

time limit per test : 2 seconds

memory limit per test : 256 megabytes

Greg has an array $a = a_1, a_2, ..., a_n$ and m operations. Each operation looks as: $l_i, r_i, d_i, (1 \le l_i \le r_i \le n)$. To apply operation i to the array means to increase all array elements with numbers $l_i, l_i + 1, ..., r_i$ by value d_i.

Greg wrote down k queries on a piece of paper. Each query has the following form: $x_i, y_i, (1 \le x_i \le y_i \le m)$. That means that one should apply operations with numbers $x_i, x_i + 1, ..., y_i$ to the array.

Now Greg is wondering, what the array a will be after all the queries are executed. Help Greg.

Input

The first line contains integers n, m, k $(1 \le n, m, k \le 10^5)$. The second line contains n integers: $a_1, a_2, ..., a_n$ $(0 \le a_i \le 10^5)$ — the initial array.

Next m lines contain operations, the operation number i is written as three integers: $l_i, r_i, d_i, (1 \le l_i \le r_i \le n), (0 \le d_i \le 10^5)$.

Next k lines contain the queries, the query number i is written as two integers: $x_i, y_i, (1 \le x_i \le y_i \le m)$.

The numbers in the lines are separated by single spaces.

Output

On a single line print n integers $a_1, a_2, ..., a_n$ — the array after executing all the queries. Separate the printed numbers by spaces.

Please, do not use the `%lld` specifier to read or write 64-bit integers in *C++*. It is preferred to use the `cin`, `cout` streams of the `%I64d`specifier.

Sample test(s)

input

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

output

```
9 18 17
```

input

```
1 1 1
1
1 1 1
1 1
```

output

```
2
```

input

```
4 3 6
1 2 3 4
1 2 1
2 3 2
3 4 4
1 2
1 3
2 3
1 2
1 3
2 3
```

output

```
5 18 31 20
```

Greg and Graph

time limit per test : 2 seconds

memory limit per test : 256 megabytes

Greg has a weighed directed graph, consisting of n vertices. In this graph any pair of distinct vertices has an edge between them in both directions. Greg loves playing with the graph and now he has invented a new game:

- The game consists of n steps.
- On the i-th step Greg removes vertex number x_i from the graph. As Greg removes a vertex, he also removes all the edges that go in and out of this vertex.
- Before executing each step, Greg wants to know the sum of lengths of the shortest paths between all pairs of the remaining vertices. The shortest path can go through any remaining vertex. In other words, if we assume that $d(i, v, u)$ is the shortest path between vertices v and u in the graph that formed before deleting vertex x_i, then Greg wants to know the value of the following sum: $\sum_{v,u,v \neq u} d(i, v, u)$.

Help Greg, print the value of the required sum before each step.

Input

The first line contains integer n $(1 \le n \le 500)$ — the number of vertices in the graph.
Next n lines contain n integers each — the graph adjacency matrix: the j-th number in the i-th line a_{ij} $(1 \le a_{ij} \le 10^5, a_{ii} = 0)$ represents the weight of the edge that goes from vertex i to vertex j.
The next line contains n distinct integers: $x_1, x_2, ..., x_n$ $(1 \le x_i \le n)$ — the vertices that Greg deletes.

Output

Print n integers — the i-th number equals the required sum before the i-th step.
Please, do not use the `%lld` specifier to read or write 64-bit integers in *C++*. It is preferred to use the `cin`, `cout` streams of the `%I64d`specifier.

Sample test(s)

input

```
1
0
1
```

output

```
0
```

input

```
2
0 5
4 0
1 2
```

output

```
9 0
```

input

```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```

output

```
17 23 404 0
```

TheNumberGameDiv

Manao has recently invented a brand new single-player game called The Number Game.

The player starts with a number **A**. Also, another number **B** is chosen. Note that neither **A** nor **B** contain a zero digit in their base 10 representation.

The player's goal is to obtain **B** from **A**. In each move, the player can either reverse his current number, or he can divide it by 10 (using integer division). For example, if the current number is 12849, the player can either reverse it to obtain 94821, or he can divide it by 10 to obtain 1284. (Note that we always round *down* when using integer division.)

You are given two ints **A** and **B**. If it is possible to obtain **B** from **A**, return the minimum number of moves necessary to reach this goal. Otherwise, return -1.

Sample tests:

0)

```
25
5
Returns: 2
```

Initially, the player has number 25 and needs to obtain 5. He can reverse the number and obtain 52, then divide it by 10 and obtain 5.

1)

```
5162
16
Returns: 4
```

To obtain 16 from 5162 in four moves, the player can perform the following sequence of moves:

- Reverse the number and obtain 2615.
- Divide 2615 by 10 and obtain 261.
- Reverse 261 and obtain 162.
- Divide 162 by 10 and obtain 16.

Note that this is not the only possible sequence of four moves which leads to the goal.

2)

```
334
12
Returns: -1
```

There is no way to obtain 12 from 334.

3)

```
218181918
9181
Returns: 6
```

4)

```
9798147
79817
Returns: -1
```