Codeforces Round #396 (Div. 2)

A. Mahmoud and Longest Uncommon Subsequence

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

While Mahmoud and Ehab were practicing for IOI, they found a problem which name was Longest common subsequence. They solved it, and then Ehab challenged Mahmoud with another problem.

Given two strings a and b, find the length of their longest uncommon subsequence, which is the longest string that is a subsequence of one of them and not a subsequence of the other.

A subsequence of some string is a sequence of characters that appears in the same order in the string, The appearances don't have to be consecutive, for example, strings "ac", "bc", "abc" and "a" are subsequences of string "abc" while strings "abbc" and "acb" are not. The empty string is a subsequence of any string. Any string is a subsequence of itself.

Input

The first line contains string a, and the second line — string b. Both of these strings are non-empty and consist of lowercase letters of English alphabet. The length of each string is not bigger than 10^5 characters.

Output

If there's no uncommon subsequence, print "-1". Otherwise print the length of the longest uncommon subsequence of a and b.

Examples

input
abcd defgh
output
5

input
a a
output
-1

Note

In the first example: you can choose "defgh" from string b as it is the longest subsequence of string b that doesn't appear as a subsequence of string a.

B. Mahmoud and a Triangle

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

Mahmoud has n line segments, the i-th of them has length a_i. Ehab challenged him to use **exactly 3** line segments to form a non-degenerate triangle. Mahmoud doesn't accept challenges unless he is sure he can win, so he asked you to tell him if he should accept the challenge. Given the lengths of the line segments, check if he can choose exactly 3 of them to form a non-degenerate triangle.

Mahmoud should use exactly 3 line segments, he can't concatenate two line segments or change any length. A non-degenerate triangle is a triangle with positive area.

Input

The first line contains single integer n ($3 \le n \le 10^5$) — the number of line segments Mahmoud has.

The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$) — the lengths of line segments Mahmoud has.

Output

In the only line print "YES" if he can choose exactly three line segments and form a non-degenerate triangle with them, and "NO" otherwise.

Examples

input
5 1 5 3 2 4
output
YES

input
3 4 1 2
output
NO

Note

For the first example, he can use line segments with lengths 2, 4 and 5 to form a non-degenerate triangle.

C. Mahmoud and a Message

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

Mahmoud wrote a message s of length n. He wants to send it as a birthday present to his friend Moaz who likes strings. He wrote it on a magical paper but he was surprised because some characters disappeared while writing the string. That's because this magical paper doesn't allow character number i in the English alphabet to be written on it in a string of length more than a_i. For example, if $a_1 = 2$ he can't write character 'a' on this paper in a string of length 3 or more. String "aa" is allowed while string "aaa" is not.

Mahmoud decided to split the message into some non-empty substrings so that he can write every substring on an independent magical paper and fulfill the condition. The sum of their lengths should be n and they shouldn't overlap. For example, if $a_1 = 2$ and he wants to send string "aaa", he can split it into "a" and "aa" and use 2 magical papers, or into "a", "a" and "a" and use 3 magical papers. He can't split it into "aa" and "aa" because the sum of their lengths is greater than n. He can split the message into single string if it fulfills the conditions.

A substring of string s is a string that consists of some consecutive characters from string s, strings "ab", "abc" and "b" are substrings of string "abc", while strings "acb" and "ac" are not. Any string is a substring of itself.

While Mahmoud was thinking of how to split the message, Ehab told him that there are many ways to split it. After that Mahmoud asked you three questions:

- How many ways are there to split the string into substrings such that every substring fulfills the condition of the magical paper, the sum of their lengths is n and they don't overlap? Compute the answer modulo $10^9 + 7$.
- What is the maximum length of a substring that can appear in some valid splitting?
- What is the minimum number of substrings the message can be spit in?

Two ways are considered different, if the sets of split positions differ. For example, splitting "aa|a" and "a|aa" are considered different splittings of message "aaa".

Input

The first line contains an integer n ($1 \le n \le 10^3$) denoting the length of the message.

The second line contains the message s of length n that consists of lowercase English letters.

The third line contains 26 integers $a_1, a_2, ..., a_{26}$ ($1 \le a_x \le 10^3$) — the maximum lengths of substring each letter can appear in.

Output

Print three lines.

In the first line print the number of ways to split the message into substrings and fulfill the conditions mentioned in the problem modulo $10^9 + 7$.

In the second line print the length of the longest substring over all the ways.

In the third line print the minimum number of substrings over all the ways.

Examples

input
3 aab 2 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
output
3 2 2

input
10 abcdeabcde 5 5 5 5 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
output
401 4 3

Note

In the first example the three ways to split the message are:

- a|a|b
- aa|b
- a|ab

The longest substrings are "aa" and "ab" of length 2.

The minimum number of substrings is 2 in "a|ab" or "aa|b".

Notice that "aab" is not a possible splitting because the letter 'a' appears in a substring of length 3, while $a_1 = 2$.

D. Mahmoud and a Dictionary

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Mahmoud wants to write a new dictionary that contains n words and relations between them. There are two types of relations: synonymy (i. e. the two words mean the same) and antonymy (i. e. the two words mean the opposite). From time to time he discovers a new relation between two words.

He know that if two words have a relation between them, then each of them has relations with the words that has relations with the other. For example, if `like` means `love` and `love` is the opposite of `hate`, then `like` is also the opposite of `hate`. One more example: if `love` is the opposite of `hate` and `hate` is the opposite of `like`, then `love` means `like`, and so on.

Sometimes Mahmoud discovers a wrong relation. A wrong relation is a relation that makes two words equal and opposite at the same time. For example if he knows that `love` means `like` and `like` is the opposite of `hate`, and then he figures out that `hate` means `like`, the last relation is absolutely wrong because it makes `hate` and `like` opposite and have the same meaning at the same time.

After Mahmoud figured out many relations, he was worried that some of them were wrong so that they will make other relations also wrong, so he decided to tell every relation he figured out to his coder friend Ehab and for every relation he wanted to know is it correct or wrong, basing on the previously discovered relations. If it is wrong he ignores it, and doesn't check with following relations.

After adding all relations, Mahmoud asked Ehab about relations between some words based on the information he had given to him. Ehab is busy making a Codeforces round so he asked you for help.

Input

The first line of input contains three integers n, m and q ($2 \le n \le 10^5$, $1 \le m, q \le 10^5$) where n is the number of words in the dictionary, m is the number of relations Mahmoud figured out and q is the number of questions Mahmoud asked after telling all relations.

The second line contains n distinct words $a_1, a_2, ..., a_n$ consisting of small English letters with length not exceeding 20, which are the words in the dictionary.

Then m lines follow, each of them contains an integer t ($1 \le t \le 2$) followed by two different words x_i and y_i which has appeared in the dictionary words. If $t = 1$, that means x_i has a synonymy relation with y_i, otherwise x_i has an antonymy relation with y_i.

Then q lines follow, each of them contains two different words which has appeared in the dictionary. That are the pairs of words Mahmoud wants to know the relation between basing on the relations he had discovered.

All words in input contain only lowercase English letters and their lengths don't exceed 20 characters. In all relations and in all questions the two words are different.

Output

First, print m lines, one per each relation. If some relation is wrong (makes two words opposite and have the same meaning at the same time) you should print "`NO`" (without quotes) and ignore it, otherwise print "`YES`" (without quotes).

After that print q lines, one per each question. If the two words have the same meaning, output 1. If they are opposites, output 2. If there is no relation between them, output 3.

See the samples for better understanding.

Examples

input

```
3 3 4
hate love like
1 love like
2 love hate
1 hate like
love like
love hate
like hate
hate like
```

output

```
YES
YES
NO
1
2
2
2
```

input

```
8 6 5
hi welcome hello ihateyou goaway dog cat rat
1 hi welcome
1 ihateyou goaway
2 hello ihateyou
2 hi goaway
```

```
2 hi hello
1 hi hello
dog cat
dog hi
hi hello
ihateyou goaway
welcome ihateyou
```

output

```
YES
YES
YES
YES
NO
YES
3
3
1
1
2
```

E. Mahmoud and a xor trip

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

Mahmoud and Ehab live in a country with n cities numbered from 1 to n and connected by n - 1 undirected roads. It's guaranteed that you can reach any city from any other using these roads. Each city has a number a_i attached to it.

We define the distance from city x to city y as the xor of numbers attached to the cities on the path from x to y **(including both x and y)**. In other words if values attached to the cities on the path from x to y form an array p of length l then the distance between them is , where is bitwise xor operation.

Mahmoud and Ehab want to choose two cities and make a journey from one to another. The index of the start city is always less than or equal to the index of the finish city (they may start and finish in the same city and in this case the distance equals the number attached to that city). They can't determine the two cities so they try every city as a start and every city with greater index as a finish. They want to know the total distance between all pairs of cities.

Input

The first line contains integer n ($1 \le n \le 10^5$) — the number of cities in Mahmoud and Ehab's country.

Then the second line contains n integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 10^6$) which represent the numbers attached to the cities. Integer a_i is attached to the city i.

Each of the next n - 1 lines contains two integers u and v ($1 \le u, v \le n, u \ne v$), denoting that there is an undirected road between cities u and v. It's guaranteed that you can reach any city from any other using these roads.

Output

Output one number denoting the total distance between all pairs of cities.

Examples

input
3 1 2 3 1 2 2 3
output
10

input
5 1 2 3 4 5 1 2 2 3 3 4 3 5
output
52

input
5 10 9 8 7 6 1 2 2 3 3 4 3 5
output
131

Note

A bitwise xor takes two bit integers of equal length and performs the logical xor operation on each pair of corresponding bits. The result in each position is 1 if only the first bit is 1 or only the second bit is 1, but will be 0 if both are 0 or both are 1. You can read more about bitwise xor operation here: https://en.wikipedia.org/wiki/Bitwise_operation#XOR.

In the first sample the available paths are:

- city 1 to itself with a distance of 1,
- city 2 to itself with a distance of 2,
- city 3 to itself with a distance of 3,
- city 1 to city 2 with a distance of ,
- city 1 to city 3 with a distance of ,

- city 2 to city 3 with a distance of .

The total distance between all pairs of cities equals $1 + 2 + 3 + 3 + 0 + 1 = 10$.
