HARBOUR.SPACE UNIVERSITY

Educational Codeforces Round 38 (Rated for Div. 2)

A. Word Correction

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Victor tries to write his own text editor, with word correction included. However, the rules of word correction are really strange.

Victor thinks that if a word contains two **consecutive** vowels, then it's kinda weird and it needs to be replaced. So the word corrector works in such a way: as long as there are two consecutive vowels in the word, it deletes the first vowel in a word such that there is **another vowel right before it**. If there are no two consecutive vowels in the word, it is considered to be correct.

You are given a word s. Can you predict what will it become after correction?

In this problem letters `a`, `e`, `i`, `o`, `u` and `y` are considered to be vowels.

Input

The first line contains one integer n $(1 \le n \le 100)$ — the number of letters in word s before the correction.

The second line contains a string s consisting of exactly n lowercase Latin letters — the word before the correction.

Output

Output the word s after the correction.

Examples

input
```
5
weird
```
output
```
werd
```

input
```
4
word
```
output
```
word
```

input
```
5
aaeaa
```
output
```
a
```

Note

Explanations of the examples:

1. There is only one replace: w**ei**rd $\longrightarrow$ werd;
2. No replace needed since there are no two consecutive vowels;
3. **aa**eaa $\longrightarrow$ **ae**aa $\longrightarrow$ **aa**a $\longrightarrow$ **aa** $\longrightarrow$ a.

B. Run For Your Prize

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

You and your friend are participating in a TV show "Run For Your Prize".

At the start of the show n prizes are located on a straight line. i-th prize is located at position a_i. Positions of all prizes are distinct. You start at position 1, your friend — at position 10^6 (and there is no prize in any of these two positions). You have to work as a team and collect all prizes in minimum possible time, in any order.

You know that it takes exactly 1 second to move from position x to position $x + 1$ or $x - 1$, both for you and your friend. You also have trained enough to instantly pick up any prize, if its position is equal to your current position (and the same is true for your friend). Carrying prizes does not affect your speed (or your friend's speed) at all.

Now you may discuss your strategy with your friend and decide who will pick up each prize. Remember that every prize must be picked up, either by you or by your friend.

What is the minimum number of seconds it will take to pick up all the prizes?

Input

The first line contains one integer n $(1 \leq n \leq 10^5)$ — the number of prizes.

The second line contains n integers $a_1, a_2, ..., a_n$ $(2 \leq a_i \leq 10^6 - 1)$ — the positions of the prizes. No two prizes are located at the same position. Positions are given in ascending order.

Output

Print one integer — the minimum number of seconds it will take to collect all prizes.

Examples

input

```
3
2 3 9
```

output

```
8
```

input

```
2
2 999995
```

output

```
5
```

Note

In the first example you take all the prizes: take the first at 1, the second at 2 and the third at 8.

In the second example you take the first prize in 1 second and your friend takes the other in 5 seconds, you do this simultaneously, so the total time is 5.

C. Constructing Tests

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's denote a m-free matrix as a binary (that is, consisting of only 1's and 0's) matrix such that every square submatrix of size $m \times m$ of this matrix contains at least one zero.

Consider the following problem:

*You are given two integers n and m. You have to construct an m-free square matrix of size $n \times n$ such that **the number of 1's in this matrix is maximum possible**. Print the maximum possible number of 1's in such matrix.*

You don't have to solve this problem. Instead, you have to construct a few tests for it.

You will be given t numbers x_1, x_2, ..., x_t. For every $i \in [1, t]$, find two integers n_i and m_i ($n_i \geq m_i$) such that the answer for the aforementioned problem is exactly x_i if we set $n = n_i$ and $m = m_i$.

Input

The first line contains one integer t ($1 \leq t \leq 100$) — the number of tests you have to construct.

Then t lines follow, i-th line containing one integer x_i ($0 \leq x_i \leq 10^9$).

Note that in hacks you have to set $t = 1$.

Output

For each test you have to construct, output two positive numbers n_i and m_i ($1 \leq m_i \leq n_i \leq 10^9$) such that the maximum number of 1's in a m_i-free $n_i \times n_i$ matrix is exactly x_i. If there are multiple solutions, you may output any of them; and if this is impossible to construct a test, output a single integer -1.

Example

input

```
3
21
0
1
```

output

```
5 2
1 1
-1
```

D. Buy a Ticket

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

Musicians of a popular band "Flayer" have announced that they are going to "make their exit" with a world tour. Of course, they will visit Berland as well.

There are n cities in Berland. People can travel between cities using two-directional train routes; there are exactly m routes, i-th route can be used to go from city v_i to city u_i (and from u_i to v_i), and it costs w_i coins to use this route.

Each city will be visited by "Flayer", and the cost of the concert ticket in i-th city is a_i coins.

You have friends in every city of Berland, and they, knowing about your programming skills, asked you to calculate the minimum possible number of coins they have to pay to visit the concert. For every city i you have to compute the minimum number of coins a person from city i has to spend to travel to some city j (or possibly stay in city i), attend a concert there, and return to city i (if $j \neq i$).

Formally, for every $i \in [1, n]$ you have to calculate $\min_{j=1}^{n} 2d(i, j) + a_j$, where $d(i, j)$ is the minimum number of coins you have to spend to travel from city i to city j. If there is no way to reach city j from city i, then we consider $d(i, j)$ to be infinitely large.

Input

The first line contains two integers n and m ($2 \leq n \leq 2 \cdot 10^5$, $1 \leq m \leq 2 \cdot 10^5$).

Then m lines follow, i-th contains three integers v_i, u_i and w_i ($1 \leq v_i, u_i \leq n, v_i \neq u_i$, $1 \leq w_i \leq 10^{12}$) denoting i-th train route. There are no multiple train routes connecting the same pair of cities, that is, for each (v, u) neither extra (v, u) nor (u, v) present in input.

The next line contains n integers $a_1, a_2, \ldots a_k$ ($1 \leq a_i \leq 10^{12}$) — price to attend the concert in i-th city.

Output

Print n integers. i-th of them must be equal to the minimum number of coins a person from city i has to spend to travel to some city j (or possibly stay in city i), attend a concert there, and return to city i (if $j \neq i$).

Examples

input Copy

```
4 2
1 2 4
2 3 7
6 20 1 25
```

output

```
6 14 1 25
```

input Copy

```
3 3
1 2 1
2 3 1
1 3 1
30 10 20
```

output

```
12 10 12
```

E. Max History

time limit per test: 3 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given an array a of length n. We define f_a the following way:

- Initially $f_a = 0$, $M = 1$;
- for every $2 \le i \le n$ if $a_M < a_i$ then we set $f_a = f_a + a_M$ and then set $M = i$.

Calculate the sum of f_a over all $n!$ permutations of the array a modulo $10^9 + 7$.

Note: two elements are considered different if their indices differ, so for every array a there are exactly $n!$ permutations.

Input

The first line contains integer n $(1 \le n \le 1\,000\,000)$ — the size of array a.

Second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$.

Output

Print the only integer, the sum of f_a over all $n!$ permutations of the array a modulo $10^9 + 7$.

Examples

input Copy

```
2
1 3
```

output

```
1
```

input Copy

```
3
1 1 2
```

output

```
4
```

Note

For the second example all the permutations are:

- $p = [1, 2, 3]$: f_a is equal to 1;
- $p = [1, 3, 2]$: f_a is equal to 1;
- $p = [2, 1, 3]$: f_a is equal to 1;
- $p = [2, 3, 1]$: f_a is equal to 1;
- $p = [3, 1, 2]$: f_a is equal to 0;
- $p = [3, 2, 1]$: f_a is equal to 0.

Where p is the array of the indices of initial array a. The sum of f_a is equal to 4.

F. Erasing Substrings

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given a string s, initially consisting of n lowercase Latin letters. After that, you perform k operations with it, where $k = \lfloor \log_2(n) \rfloor$. During i-th operation you **must** erase some substring of length exactly 2^{i-1} from s.

Print the lexicographically minimal string you may obtain after performing k such operations.

Input

The only line contains one string s consisting of n lowercase Latin letters ($1 \le n \le 5000$).

Output

Print the lexicographically minimal string you may obtain after performing k operations.

Examples

input

```
adcbca
```

output

```
aba
```

input

```
abacabadabacaba
```

output

```
aabacaba
```

Note

Possible operations in examples:

1. adcb**ca** $\longrightarrow$ ad**c**ba $\longrightarrow$ aba;
2. ab**a**cabadabacaba $\longrightarrow$ a**bc**abadabacaba $\longrightarrow$ aaba**daba**caba $\longrightarrow$ aabacaba.

G. Shortest Path Queries

time limit per test: 3.5 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

You are given an undirected connected graph with weighted edges. The length of some path between two vertices is the bitwise xor of weights of all edges belonging to this path (if some edge is traversed more than once, then it is included in bitwise xor the same number of times).

There are three types of queries you have to process:

- $1\ x\ y\ d$ — add an edge connecting vertex x to vertex y with weight d. It is guaranteed that there is no edge connecting x to y before this query;
- $2\ x\ y$ — remove an edge connecting vertex x to vertex y. It is guaranteed that there was such edge in the graph, and the graph stays connected after this query;
- $3\ x\ y$ — calculate the length of the shortest path (possibly non-simple) from vertex x to vertex y.

Print the answers for all queries of type 3.

Input

The first line contains two numbers n and m $(1 \le n, m \le 200000)$ — the number of vertices and the number of edges in the graph, respectively.

Then m lines follow denoting the edges of the graph. Each line contains three integers x, y and d $(1 \le x < y \le n, 0 \le d \le 2^{30} - 1)$. Each pair (x, y) is listed at most once. The initial graph is connected.

Then one line follows, containing an integer q $(1 \le q \le 200000)$ — the number of queries you have to process.

Then q lines follow, denoting queries in the following form:

- $1\ x\ y\ d$ $(1 \le x < y \le n, 0 \le d \le 2^{30} - 1)$ — add an edge connecting vertex x to vertex y with weight d. It is guaranteed that there is no edge connecting x to y before this query;
- $2\ x\ y$ $(1 \le x < y \le n)$ — remove an edge connecting vertex x to vertex y. It is guaranteed that there was such edge in the graph, and the graph stays connected after this query;
- $3\ x\ y$ $(1 \le x < y \le n)$ — calculate the length of the shortest path (possibly non-simple) from vertex x to vertex y.

It is guaranteed that at least one query has type 3.

Output

Print the answers for all queries of type 3 in the order they appear in input.

Example

input Copy

```
5 5
1 2 3
2 3 4
3 4 5
4 5 6
1 5 1
5
3 1 5
1 1 3 1
3 1 5
2 1 5
3 1 5
```

output

```
1
1
2
```