Bubble Cup 12 - Finals [Online Mirror, unrated, Div. 1]

A. BubbleReactor

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are in charge of the BubbleReactor. It consists of N BubbleCores connected with N lines of electrical wiring. Each electrical wiring connects two distinct BubbleCores. There are no BubbleCores connected with more than one line of electrical wiring.

Your task is to start the BubbleReactor by starting each BubbleCore. In order for a BubbleCore to be started it needs to be receiving power from a directly connected BubbleCore which is already started. However, you can kick-start one BubbleCore manually without needing power. It is guaranteed that all BubbleCores can be started.

Before the BubbleCore boot up procedure its potential is calculated as the number of BubbleCores it can power on (the number of inactive BubbleCores which are connected to it directly or with any number of inactive BubbleCores in between, itself included)

Start the BubbleReactor so that the sum of all BubbleCores' potentials is maximum.

Input
First line contains one integer $N(3 \leq N \leq 15.000)$, the number of BubbleCores.

The following N lines contain two integers $U, V(0 \leq U \neq V < N)$ denoting that there exists electrical wiring between BubbleCores U and V.

Output
Single integer, the maximum sum of all BubbleCores' potentials.

Example

input
10 0 1 0 3 0 4 0 9 1 2 2 3 2 7 4 5 4 6 7 8
output
51

Note
If we start by kickstarting BubbleCup 8 and then turning on cores 7, 2, 1, 3, 0, 9, 4, 5, 6 in that order we get potentials 10 + 9 + 8 + 7 + 6 + 5 + 1 + 3 + 1 + 1 = 51

B. Guarding warehouses

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

Bob Bubblestrong just got a new job as security guard. Bob is now responsible for safety of a collection of warehouses, each containing the most valuable Bubble Cup assets - the high-quality bubbles. His task is to detect thieves inside the warehouses and call the police.

Looking from the sky, each warehouse has a shape of a convex polygon. Walls of no two warehouses intersect, and of course, none of the warehouses is built inside of another warehouse.

Little did the Bubble Cup bosses know how lazy Bob is and that he enjoys watching soap operas (he heard they are full of bubbles) from the coziness of his office. Instead of going from one warehouse to another to check if warehouses are secured, the plan Bob has is to monitor all the warehouses from the comfort of his office using the special X-ray goggles. The goggles have an infinite range, so a thief in any of the warehouses could easily be spotted.

However, the goggles promptly broke and the X-rays are now strong only enough to let Bob see through a single wall. Now, Bob would really appreciate if you could help him find out what is the total area inside of the warehouses monitored by the broken

goggles, so that he could know how much area of the warehouses he needs to monitor in person.

Input

The first line contains one integer N ($1 \le N \le 10^4$) – the number of warehouses.

The next N lines describe the warehouses.

The first number of the line is integer c_i ($3 \le c_i \le 10^4$) – the number corners in the i^{th} warehouse, followed by c_i pairs of integers. The j^{th} pair is (x_j, y_j) – the coordinates of the j^{th} corner ($|x_j|, |y_j| \le 3 * 10^4$). The corners are listed in the clockwise order. The total number of corners in all the warehouses is at most $5 * 10^4$.

Bob's office is positioned at the point with coordinates $(0, 0)$. The office is not contained within any of the warehouses.

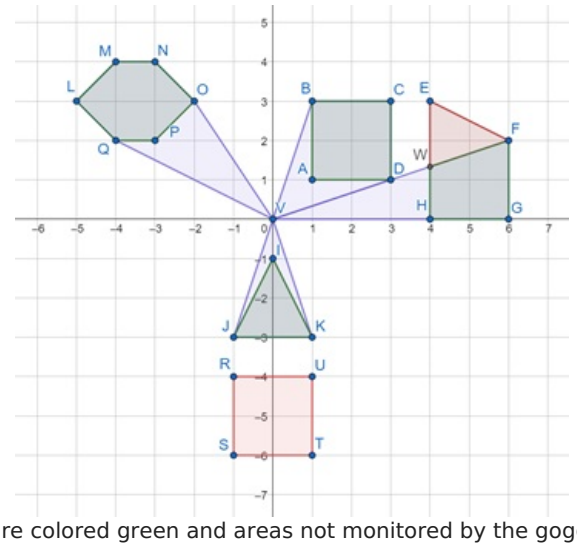
Output

Print a single line containing a single decimal number accurate to at least four decimal places – the total area of the warehouses Bob can monitor using the broken X-ray goggles.

Example

input
5 4 1 1 1 3 3 3 3 1 4 4 3 6 2 6 0 4 0 6 -5 3 -4 4 -3 4 -2 3 -3 2 -4 2 3 0 -1 1 -3 -1 -3 4 1 -4 1 -6 -1 -6 -1 -4
output
13.333333333333

Note



Areas monitored by the X-ray goggles are colored green and areas not monitored by the goggles are colored red.

The warehouses $ABCD$, IJK and $LMNOPQ$ are completely monitored using the googles.

The warehouse $EFGH$ is partially monitored using the goggles: part EFW is not monitored because to monitor each point inside it, the X-rays must go through two walls of warehouse $ABCD$.

The warehouse $RUTS$ is not monitored from the Bob's office, because there are two walls of the warehouse IJK between Bob's office and each point in $RUTS$.

The total area monitored by the goggles is $P = P_{ABCD} + P_{FGHW} + P_{IJK} + P_{LMNOPQ} = 4 + 3.333333333333 + 2 + 4 = 13.333333333333$.

C. Jumping Transformers

time limit per test: 4 seconds
memory limit per test: 128 megabytes
input: standard input
output: standard output

You, the mighty Blackout, are standing in the upper-left $(0, 0)$ corner of NxM matrix. You must move either right or down each second.

There are K transformers jumping around the matrix in the following way. Each transformer starts jumping from position (x, y), at time t, and jumps to the next position each second. The x-axes grows downwards, and y-axes grows to the right. The order of jumping positions is defined as $(x, y), (x + d, y - d), (x + d, y), (x, y + d)$, and is periodic. Before time t transformer is not in the matrix.

You want to arrive to the bottom-right corner $(N - 1, M - 1)$, while slaying transformers and losing the least possible amount of

energy. When you meet the transformer (or more of them) in the matrix field, you must kill them all, and you lose the sum of the energy amounts required to kill each transformer.

After the transformer is killed, he of course stops jumping, falls into the abyss and leaves the matrix world. Output minimum possible amount of energy wasted.

Input
In the first line, integers N,M ($1 \leq N, M \leq 500$), representing size of the matrix, and K ($0 \leq K \leq 5 * 10^5$) , the number of jumping transformers.

In next K lines, for each transformer, numbers x, y, d ($d \geq 1$), t ($0 \leq t \leq N + M - 2$), and e ($0 \leq e \leq 10^9$), representing starting coordinates of transformer, jumping positions distance in pattern described above, time when transformer starts jumping, and energy required to kill it.

It is guaranteed that all 4 of jumping points of the transformers are within matrix coordinates

Output
Print single integer, the minimum possible amount of energy wasted, for Blackout to arrive at bottom-right corner.

Example

input
3 3 5 0 1 1 0 7 1 1 1 0 10 1 1 1 1 2 1 1 1 2 2 0 1 1 2 3
output
9

Note
If Blackout takes the path from (0, 0) to (2, 0), and then from (2, 0) to (2, 2) he will need to kill the first and third transformer for a total energy cost of 9. There exists no path with less energy value.

D. Xor Spanning Tree

time limit per test: 2 seconds
memory limit per test: 128 megabytes
input: standard input
output: standard output

In the galaxy far far away is the ancient interplanetary republic of Bubbleland, consisting of N planets. Between them, there are M bidirectional wormholes, each connecting a pair of planets. Bubbleland is a very centralized republic, having a capital planet Whiteplanet, from which any another planet can be reached using these wormholes. It is also guaranteed that no wormhole connects planet to itself and that no two different wormholes connect same pair of planets.

We call a path that begins at one planet, visits other planets and each of them at most once and returns to starting point a *tour*. Interplanetary Safety Regulations guarantee that each planet belongs to at most one *tour* and that there are at most 42 *tours*.

After many eons of usage, wormholes need to be repaired and each wormhole has the cost W_i which needs to be payed for reparation. Unfortunately, the Senate of Bubbleland is short on budget. Therefore, they have decided only to fix as many wormholes as they need in order to have all planets reachable from capital and to pay as little money as they have to for this repair. However the way in which the Senate calculates the cost is different. Cost of the set of reparations is binary xor of costs of each individual reparation, that is if reparations to be made have costs $A_1, A_2, \ldots, A_k$, the cost of entire set is $A_1 \oplus A_2 \oplus \ldots \oplus A_k$.

Now the Senate would like to know how much money do they have to pay and also the number of different ways to achieve that cost **modulo** 1000000007.

Input
First line of input contains two numbers $N(1 \leq N \leq 100.000)$, the number of planets and $M(1 \leq M \leq 100.041)$, the number of wormholes. Following M lines contain three numbers $U, V(1 \leq U \neq V \leq N)$ and $W(1 \leq W \leq 100.000)$, meaning that there exists a wormhole connecting planets U and V, with repair cost of W.

Output
Output two numbers, the smallest possible cost of entire reparation and the number of different valid reparations with that cost **modulo** 1000000007.

Example

input
6 6 4 1 5 5 2 1 6 3 2 1 2 6 1 3 3 2 3 4

output
1 1

Note
We can repair wormholes 1,2,3,5 and 6, paying $5 \oplus 1 \oplus 2 \oplus 3 \oplus 4 = 1$, one can check that this is the cheapest repair in which all of the planets are connected and the only valid repair with that cost.

E. Product Tuples

time limit per test: 8 seconds
memory limit per test: 128 megabytes
input: standard input
output: standard output

While roaming the mystic areas of Stonefalls, in order to drop legendary loot, an adventurer was given a quest as follows. He was given an array $A = a_1, a_2, \ldots, a_N$ of length N, and a number K.

Define array B as $B(q, A) = \{ q - a_1, q - a_2, \ldots, q - a_N \}$. Define function F as $F(B, K)$ being sum of products of all K-tuples of elements in array B. For example, if the array B is $[2, 3, 4, 5]$, and with $K = 3$, sum of products of all 3-tuples is

$$F(B, 3) = 2 * 3 * 4 + 2 * 3 * 5 + 3 * 4 * 5 + 2 * 4 * 5$$

He was then given a number Q, number of queries of two types:

- Type 1: Given q, i, and d calculate $F(B(q, A), K)$ where we make change to initial array as $A[i] = d$.
- Type 2: Given q, L, R, and d calculate $F(B(q, A), K)$ where we make change to initial array as $A[i] = A[i] + d$ for all i in range $[L, R]$ inclusive.

All changes are temporarily made to initial array, and don't propagate to following queries. Help the adventurer calculate the answer to a quest, and finally get that loot!

Input
In the first two lines, numbers N ($1 \leq N \leq 2 * 10^4$) and K ($1 \leq K \leq N$), the length of initial array A, and tuple size, followed by $a_1, a_2, a_3, \ldots, a_N$ ($0 \leq a_i \leq 10^9$) , elements of array A, in the next line. Then follows number Q ($Q \leq 10$), number of queries. In the next Q lines come queries of the form:

- 1 q i d, for type 1,
- 2 q L R d, for type 2,

as explained above ($0 \leq q, d \leq 10^9, 1 \leq i, L, R \leq N$)

Output
Print Q lines, the answers to queries, modulo 998244353.

Example

input
5 2 1 2 3 4 5 3 1 6 1 1 1 6 5 2 2 6 2 3 1
output
85 127 63

Note
In the first query array A = [1, 2, 3, 4, 5], B = [5, 4, 3, 2, 1], sum of products of 2-tuples = 85.

In second query array A = [1, 2, 3, 4, 2], B = [5, 4, 3, 2, 4], sum of products of 2-tuples = 127

In third query array A = [1, 3, 4, 4, 5], B = [5, 3, 2, 2, 1], sum of products of 2-tuples = 63

F. Workout plan

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

Alan decided to get in shape for the summer, so he created a precise workout plan to follow. His plan is to go to a different gym every day during the next N days and lift $X[i]$ grams on day i. In order to improve his workout performance at the gym, he can buy exactly one pre-workout drink at the gym he is currently in and it will improve his performance by A grams permanently and

immediately. In different gyms these pre-workout drinks can cost different amounts $C[i]$ because of the taste and the gym's location but its permanent workout gains are the same. Before the first day of starting his workout plan, Alan knows he can lift a maximum of K grams. Help Alan spend a minimum total amount of money in order to reach his workout plan. If there is no way for him to complete his workout plan successfully output -1.

Input

The first one contains two integer numbers, integers N $(1 \leq N \leq 10^5)$ and K $(1 \leq K \leq 10^5)$ – representing number of days in the workout plan and how many grams he can lift before starting his workout plan respectively. The second line contains N integer numbers $X[i]$ $(1 \leq X[i] \leq 10^9)$ separated by a single space representing how many grams Alan wants to lift on day i. The third line contains one integer number A $(1 \leq A \leq 10^9)$ representing permanent performance gains from a single drink. The last line contains N integer numbers $C[i]$ $(1 \leq C[i] \leq 10^9)$, representing cost of performance booster drink in the gym he visits on day i.

Output

One integer number representing minimal money spent to finish his workout plan. If he cannot finish his workout plan, output -1.

Examples

input
5 10000 10000 30000 30000 40000 20000 20000 5 2 8 3 6
output
5

input
5 10000 10000 40000 30000 30000 20000 10000 5 2 8 3 6
output
-1

Note

First example: After buying drinks on days 2 and 4 Alan can finish his workout plan. Second example: Alan cannot lift 40000 grams on day 2.

G. Alpha planetary system

time limit per test: 1 second
memory limit per test: 128 megabytes
input: standard input
output: standard output

Three planets X, Y and Z within the Alpha planetary system are inhabited with an advanced civilization. The spaceports of these planets are connected by interplanetary space shuttles. The flight scheduler should decide between 1, 2 and 3 return flights for every existing space shuttle connection. Since the residents of Alpha are strong opponents of the symmetry, there is a strict rule that any two of the spaceports connected by a shuttle must have a different number of flights.

For every pair of connected spaceports, your goal is to propose a number 1, 2 or 3 for each shuttle flight, so that for every two connected spaceports the overall number of flights differs.

You may assume that:

1) Every planet has at least one spaceport

2) There exist only shuttle flights between spaceports of different planets

3) For every two spaceports there is a series of shuttle flights enabling traveling between them

4) Spaceports are not connected by more than one shuttle

Input

The first row of the input is the integer number N $(3 \leq N \leq 100000)$, representing overall number of spaceports. The second row is the integer number M $(2 \leq M \leq 100000)$ representing number of shuttle flight connections.

Third row contains N characters from the set $\{X, Y, Z\}$. Letter on I^{th} position indicates on which planet is situated spaceport I. For example, "XYYXZZ" indicates that the spaceports 0 and 3 are located at planet X, spaceports 1 and 2 are located at Y, and spaceports 4 and 5 are at Z.

Starting from the fourth row, every row contains two integer numbers separated by a whitespace. These numbers are natural numbers smaller than N and indicate the numbers of the spaceports that are connected. For example, "12 15" indicates that there is a shuttle flight between spaceports 12 and 15.

Output

The same representation of shuttle flights in separate rows as in the input, but also containing a third number from the set $\{1, 2, 3\}$ standing for the number of shuttle flights between these spaceports.

Example

input
10 15 XXXXYYYZZZ 0 4 0 5 0 6 4 1 4 8 1 7 1 9 7 2 7 5 5 3 6 2 6 9 8 2 8 3 9 3
output
0 4 2 0 5 2 0 6 2 4 1 1 4 8 1 1 7 2 1 9 3 7 2 2 7 5 1 5 3 1 6 2 1 6 9 1 8 2 3 8 3 1 9 3 1

H. Function Composition

time limit per test: 0.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

We are definitely not going to bother you with another generic story when Alice finds about an array or when Alice and Bob play some stupid game. This time you'll get a simple, plain text.

First, let us define several things. We define function F on the array A such that $F(i, 1) = A[i]$ and $F(i, m) = A[F(i, m - 1)]$ for $m > 1$. In other words, value $F(i, m)$ represents composition $A[\dots A[i]]$ applied m times.

You are given an array of length N with non-negative integers. You are expected to give an answer on Q queries. Each query consists of two numbers – m and y. For each query determine how many x exist such that $F(x, m) = y$.

Input

The first line contains one integer N $(1 \le N \le 2 \cdot 10^5)$ – the size of the array A. The next line contains N non-negative integers – the array A itself $(1 \le A_i \le N)$. The next line contains one integer Q $(1 \le Q \le 10^5)$ – the number of queries. Each of the next Q lines contain two integers m and y $(1 \le m \le 10^{18}, 1 \le y \le N)$.

Output

Output exactly Q lines with a single integer in each that represent the solution. Output the solutions in the order the queries were asked in.

Example

input
10 2 3 1 5 6 4 2 10 7 7 5 10 1 5 7 10 6 1 1 10 8
output
3 0 1 1

0

Note

For the first query we can notice that $F(3,10)=1,\ F(9,10)=1$ and $F(10,10)=1$.

For the second query no x satisfies condition $F(x,5)=7$.

For the third query $F(5,10)=6$ holds.

For the fourth query $F(3,1)=1$.

For the fifth query no x satisfies condition $F(x,10)=8$.

I. The Light Square

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

For her birthday Alice received an interesting gift from her friends – The Light Square. The Light Square game is played on an $N \times N$ lightbulbs square board with a magical lightbulb bar of size $N \times 1$ that has magical properties. At the start of the game some lights on the square board and magical bar are turned on. The goal of the game is to transform the starting light square board pattern into some other pattern using the magical bar without rotating the square board. The magical bar works as follows:

It can be placed on any row or column

The orientation of the magical lightbulb must be left to right or top to bottom for it to keep its magical properties

The entire bar needs to be fully placed on a board

The lights of the magical bar never change

If the light on the magical bar is the same as the light of the square it is placed on it will switch the light on the square board off, otherwise it will switch the light on

The magical bar can be used an infinite number of times

Alice has a hard time transforming her square board into the pattern Bob gave her. Can you help her transform the board or let her know it is impossible? If there are multiple solutions print any.

Input

The first line contains one positive integer number N $(1 \leq N \leq 2000)$ representing the size of the square board.

The next N lines are strings of length N consisting of 1's and 0's representing the initial state of the square board starting from the top row. If the character in a string is 1 it means the light is turned on, otherwise it is off.

The next N lines are strings of length N consisting of 1's and 0's representing the desired state of the square board starting from the top row that was given to Alice by Bob.

The last line is one string of length N consisting of 1's and 0's representing the pattern of the magical bar in a left to right order.

Output

Transform the instructions for Alice in order to transform the square board into the pattern Bob gave her. The first line of the output contains an integer number M $(0 \leq M \leq 10^5)$ representing the number of times Alice will need to apply the magical bar.

The next M lines are of the form "col X" or "row X", where X is 0-based index of the matrix, meaning the magical bar should be applied to either row X or column X. If there is no solution, print only -1. In case of multiple solutions print any correct one.

Examples

input
2 11 11 00 01 11
output
-1

input
2 10 00 00 00 10
output

```
1
row 0
```

input
3 110 011 100 100 011 100 100
output
3 row 0 col 0 col 1

Note

Example 1: It is impossible to transform square board from one format to another

Example 2: Magic bar can be applied on first row or column.