Problem A. And to Zero

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

You are given an array of length n, which consists of positive natural numbers. In one move you can choose an ordered pair of its elements and replace first of them with bitwise AND of these two elements. What is the minimum possible number of moves needed to obtain a 0 in the array? Also, how many shortest sequences of moves are there? Two sequences are different if and only if for some k the k-th move is different in the two sequences. Two moves are different if and only if ordered pair of indices of elements are different. As second number can be huge, output it modulo $10^9 + 7$.

Input

The first line of the input contains one integer n ($1 \leq n \leq 10^6$) – the length of the array. The second line contains n integers x_i ($1 \leq x_i < 2^{20}$) – the array.

Output

If there is no way to make some zero appear in the array then output -1. If it's possible, output two integers, where first will be equal to minimal number of moves needed to obtain a 0 in the array, and second will be equal to number of shortest sequences of moves. Output second number taken modulo $10^9 + 7$.

Examples

standard input	standard output

`5`
`8 3 12 7 15`
	`1 6`

`3`
`3 5 6`
	`2 12`

`2`
`3 5`
	`-1`

Problem B. Bitwise-Xor to Zeroes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 mebibytes

You are given an array of length n, which consists of non-negative natural numbers. In one move you can choose a pair of indices i, j ($1 \le i, j \le n$, possibly $i = j$) and perform this pseudocode:

```
$x = a_i \oplus a_j$
$a_i = x$
$a_j = x$
```

What is the minimum possible number of moves needed to make the whole array containing only zeroes? But don't be so fast, answering once for whole array would be too easy. There are many queries, in each of them you should assume that there exists only some interval of input array, and your goal is to make this interval containing only zeroes. Queries are independent – in each query you start from an interval of the initial array. It can be proven that it's always possible to turn an interval into zeroes.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 3 \cdot 10^5$) – the length of the array and number of queries. The second line contains n integers a_i ($0 \le a_i \le 10^9$) – the array. Each of the next m lines contains two integers a and b ($1 \le a \le b \le n$) describing a query about interval $[a, b]$.

Output

Output m lines, where i-th of them should contain answer for i-th query – minimal number of moves to make i-th interval containing only zeroes.

Examples

standard input	standard output

5 3
3 3 3 3 5
1 5
3 3
3 5
	3
	1
	2

Note

In third query we can choose two elements with value 3 and replace them both with value 0. Next, we can choose last index as both i and j and turn its value to 0. Interval will change as follows:
3 3 5 $\rightarrow$ 0 0 5 $\rightarrow$ 0 0 0.

Problem C. Cutting Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

A **tree** is an undirected, connected graph with n vertices and $n-1$ edges for some positive n.

A **component** is a subset of vertices of a tree together with every edge from tree which connects two vertices from this subset. Size of a component is the number of its vertices.

Component is **connected** if we can travel between any two of its vertices using only edges from this component.

You are given a tree with n vertices. For each integer k from interval $[1, n]$ you should compute maximal number of connected components of size exactly k which we can choose in the tree, such that no two components intersect (subsets of vertices are pairwise disjoint).

Note that not every vertex must belong to some chosen component.

Input

The first line of the input contains one integer n ($1 \leq n \leq 2 \cdot 10^5$) – number of vertices in the tree. Each of the next $n-1$ lines contains two integers a and b ($1 \leq a, b \leq n$) describing an edge between vertices a and b.

Output

Output n lines, where i-th of them should contain maximal number of components of size i which we can choose from the tree according to above rules.

Example

standard input	standard output

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

Problem D. Dance Classes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 mebibytes

You are a teacher in a dance school. You are going to teach new group of kids and you want to make them as good as possible. There are n boys and n girls in the group and you should divide them in pairs, in each pair there must be a boy and a girl. As an experienced teacher you know exact formula for *incompatibility* of group of dancers. Every boy can be described with a value a_i and every girl with a value b_j. The *incompatibility* of group is $(\sum_{i,j} a_i \cdot b_j)\ mod\ 10^9 + 7$ where we sum values for all pairs (i, j) such that i is a boy, j is a girl and i dances with j in pair. Of course we want the incompatibility to be as small as possible. Note that we are comparing incompatibility values after applying modulo to them, so best possible incompatibility is 0, and worst is $10^9 + 6$.

You know all values a_i and b_j. As you expected, kids' values are totally random, so both arrays a and b consist of random values from range $[0, 10^9 + 6]$. In particular you can assume that for each test there exist parameters n and *seed* such that this test was generated with this c++ code:

```
#include <bits/stdc++.h>
using namespace std;
const int MOD = 1000000007;

int main() {
    int n, seed;
    cin >> n;
    cin >> seed;
    srand(seed);
    cout << n << endl;
    for (int h = 1; h <= 2; h++) {
        for (int i = 1; i <= n; i++) {
            cout << rand() % MOD << (i == n ? "" : " ");
        }
        cout << endl;
    }
    return 0;
}
```

Your task is to divide kids into pairs such that the incompatibility of group will be minimized.

Input

The first line of the input contains one integer n $(1 \le n \le 10^5)$ – number of boys and number of girls. The second line contains n values $a_1, a_2, \cdots, a_n$ $(0 \le a_i \le 10^9 + 6)$. The third line contains n values $b_1, b_2, \cdots, b_n$ $(0 \le b_i \le 10^9 + 6)$. Both arrays consist of random values.

Output

In first line of output there should be one integer – minimal possible incompatibility of group of dancers. In next line you should print division into pairs. This line should contain a permutation of numbers from 1 to n. If i-th of these numbers is x, it means that in your solution i-th boy dances with x-th girl. If there are many permutations which minimize incompatibility, you can print any of them.

Example

standard input

10
719179442 174703255 999864684 123864197 188093092 732731150 165611498 788451330
141212055 967667550
357207174 51900578 180338702 73463891 778732064 522596279 491461429 23222771
342635672 290781496

standard output
1 8 2 6 10 7 4 3 9 1 5

Problem E. Exact Covering

Input file:	*standard input*
Output file:	*standard output*
Time limit:	9 seconds
Memory limit:	512 mebibytes

You are given a tree with n vertices and with weighted edges. Simple path is a sequence of edges from the tree, such that every two consecutive edges from the sequence have common vertex, to which both of them are incident, no edge appears more than once in this sequence and no vertex is adjacent to more than two edges from the path. Your task is to cover the tree with minimal number of simple paths (choose some multiset of simple paths) such that every edge lies on exactly as many paths as its weight. To make task more difficult, weights are still changing and after every change you have to find the minimum number of paths needed to cover all edges.

Input

The first line of the input contains one integer n ($1 \leq n \leq 5 \cdot 10^5$) – the number of vertices in tree. Each of the next $n-1$ lines contains three integers a, b and c ($1 \leq a, b \leq n;\ 0 \leq c \leq 10^9$) describing an edge between vertices a and b with weight c. Next line contains one integer q ($1 \leq q \leq 5 \cdot 10^5$) – number of changes. Each of the next q lines contains three integers a, b and c ($1 \leq a, b \leq n;\ 0 \leq c \leq 10^9$) which mean that edge between vertices a and b from now will have weight equal to c.

Output

Output $q+1$ lines. i-th line should contain one integer x, being the minimal possible number of paths needed to cover the tree after $i-1$ changes. Note that first number in the output should be equal for answer before any changes.

Example

standard input	standard output

4
1 2 2
2 3 2
2 4 1
2
2 4 2
2 1 100
	3
	3
	100

Problem F. Foreheads Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

Have you ever heard about **Foreheads game**? This game is played by k players. In this game players use only one suit from typical deck of cards (so there are 13 cards). At the beginning all cards are shuffled, everybody picks one card and puts it on his/her forehead, so everybody knows card of every other player, but doesn't know his own. Next, players one by one are guessing the relative rank of their card. For example if first player has a 5 and two other players have a 9 and an ace, then he has the third highest card. After guessing, players are in some way rewarded with points based on how close they were.

In this task we'll look on a little bit modified version of this game. k players are sitting around a table and every of them has a card on his/her forehead. Now, players cyclically are telling if they know their rank. Cyclically means that for example for $k = 3$ they are telling it in order $1, 2, 3, 1, 2, 3, 1, 2, 3...$ if we number them with numbers from 1 to k. If a current player knows his/her rank the game ends instantly. Otherwise next player's turn starts.

Note that there is no guessing in this version of the game, i.e. we are assuming that if player isn't sure about his/her rank, then he/she won't lie and if he can deduce his position, then he will for sure do it, and end the game. In other words, every player is infinitely smart. Your task is to calculate in which turn somebody will tell that he knows his/her rank or tell that game will go on infinitely.

Input

The first line of the input contains two integers k and t ($2 \leq k \leq 5;\ 1 \leq t \leq 30$) – number of players and number of games they play. Games are independent from each other. Each of the next t lines describes one game and contains k different space-separated characters from set $\{2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A\}$, where i-th of them represents i-th player's card.

Output

Print t lines. In i-th of them there should be one one integer, which means in which turn in i-th game somebody will tell that he knows his rank. If i-th game won't end, print -1.

Examples

standard input	standard output

`3 1`
`K 9 2`
	`2`

`5 2`
`4 8 Q 9 7`
`7 A K 2 3`
	`13 1`

Note

In first sample second player can see that third player has a 2, so he deduces that if he'd have a 3 or an ace, then first player would know his rank. First player doesn't know his rank, so second player deduces that he has second highest card and ends the game in second turn.

Problem G. Grid With Mirrors

Input file:	*standard input*
Output file:	*standard output*
Time limit:	9 seconds
Memory limit:	1024 mebibytes

There is a square grid with $n \times n$ cells. Rows are numbered with integers from 1 to n from top to bottom and columns are numbered with integers from 1 to n from left to right. We'll denote cell in x-th column and y-th row as (x, y). Every cell can be either empty or blocked. There are exactly m blocked cells.

There is a beam of light which comes horizontally from left side of the grid to cell $(1, 1)$, so if not obstructed it would then go through cells $(2, 1)$, $(3, 1)$ and so on. Beam of light can go through empty cells, but will stop on blocked cells and on sides of the grid.

You can place mirrors in cells. Every mirror should cover exactly one diagonal of some empty cell and it would change direction of light beam by 90 degrees. In every cell you can put at most one mirror.

Let $f(x, y)$ denote minimum number of mirrors you have to place on the grid to make light beam go through cell (x, y). If it's impossible to make beam go through cell (x, y), then we assume $f(x, y) = 0$. Your task is to calculate $\sum f(x, y)$ where summation goes over all empty cells.

Input

The first line of the input contains two integers n and m ($1 \le n \le 10^5$; $0 \le m \le 10^5$) – the length of one side of the grid and number of blocked cells. Each of the next m lines contains two integers x and y ($1 \le x, y \le n; (x, y) \ne (1, 1)$) which mean that cell (x, y) is blocked. Every cell will appear in the input at most once.

Output

Output one integer – the sum that we are looking for.

Examples

standard input	standard output
3 1 2 2	6

3 2
2 2
2 1
	12

Problem H. Heavier Coins

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

There are n coins. Each of them has a weight which can be represented with some strictly positive real number. Unfortunately you don't know much about exact weights – for every two different coins first one can be lighter than second one, heavier than second one, or even have exactly this same weight. You only know about results of m weightings. In i-th of them a_i-th and b_i-th coin were compared and it turned out that the first one is strictly heavier than the second one.

Now you want to choose sequence of non-empty subsets of coins, such that for every two different subsets from this sequence the one which appears in this sequence later is for sure strictly heavier than the one which appears earlier. Weight of subset of coins is simply sum of weights of its coins. By "for sure" we mean that there cannot exist any set of coins' weights which satisfies weightings and for which later subset isn't heavier than earlier subset. Your task is to find maximum possible number of subsets in this sequence and to print this number modulo $10^9 + 7$.

Input

The first line of the input contains two integers n and m ($1 \le n \le 10^6$; $0 \le m \le 10^6$) – number of coins and number of weightings. Each of the next m lines contain two distinct numbers a_i and b_i ($1 \le a_i, b_i \le n$) which mean that a_i-th coin is strictly heavier than b_i-th coin. It's guaranteed that these weightings don't contradict with themselves, i.e. there exists at least one set of coins' weights which satisfies these weightings.

Output

Output one integer which is equal to maximum possible number of subsets in sequence taken modulo $10^9 + 7$.

Examples

standard input	standard output
`3 2`	
`1 2`	
`1 3`	
`5 0`	`5`

Note

In first test we can choose following sequence of subsets: $\{2\}, \{1\}, \{1, 2\}, \{1, 2, 3\}$. Note that it's forbidden to have in sequence both subsets $\{1\}$ and $\{2, 3\}$, because it's not clear which subset is heavier.

Problem I. Isomorphic Matrices

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

In this task we will consider matrices with n rows and m columns, in which every cell is painted in one of k colours. We say that two matrices are **isomorphic** if and only if we can in first of them firstly permute rows (possibly with identity permutation) and secondly permute columns (again possibly with identity permutation) to get the second matrix. What is maximal possible size of set of matrices, such that every two matrices from this set aren't isomorphic?

Input

The first line of the input contains three integers n, m and k ($1 \le n, m$; $1 \le k \le 10^9$). Matrix won't contain more than 2000 cells, i.e. $n \cdot m \le 2000$ holds.

Output

Output one integer which is equal to maximum possible set of non-isomorphic matrices taken modulo $10^9 + 7$.

Example

standard input	standard output
2 3 2	13

Problem J. Just Xor Partitions

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	512 mebibytes

You are given an array of natural numbers. Your task is to handle two types of queries. In first type of queries you must change value of some element of the array. In second type you are given a small set of numbers (possibly with repetitions, but it doesn't really matter) and you have to tell if it is possible to split the whole array into any number of intervals such that for every interval the XOR of values in this interval is in the given set. Every element of the array must belong to some interval and intervals cannot overlap with each other.

Note that it isn't necessary for every number from the given set to appear as the XOR of some interval. For example, if the XOR of the whole array is in the given set you are allowed to not split the array at all. Also, every element from the set may be equal to the XOR of multiple intervals.

Input

The first line of the input contains two integers n and q ($1 \leq n \leq 10^5$; $1 \leq q$) – length of the array and the number of queries. Second line contains n integers x_i ($0 \leq x_i < 2^{20}$) – the array. Next q lines describe queries. Each query can have one of two types.

Queries of first time are given in form "1 a b" ($1 \leq a \leq n$; $0 \leq b < 2^{20}$) where a is an index of element you should change and b is its new value.

Queries of second type are given in form "2 k a_1 ... a_k" ($1 \leq k \leq 5$; $0 \leq a_i < 2^{20}$) where k is the size of the set and a_i are its elements. Note that numbers a_i don't have to be distinct.

Number of queries of first type won't exceed $4 \cdot 10^5$ and sum of numbers k from queries of second type won't exceed 10^5.

Output

For each query of second type you should output one string, every string in separate line in this same order as order of queries in the input. If it's possible to split array in corresponding query then print **TAK** ("YES" in Polish) and print **NIE** ("NO" in Polish) otherwise.

Example

standard input	standard output

5 10
1 2 0 3 0
2 1 3
2 1 0
1 3 5
2 2 6 3
1 1 8
1 2 5
1 3 3
1 4 1
1 5 1
2 3 2 4 8
	TAK
	TAK
	TAK
	NIE

Note

In first query of second type the array looks like $\{1, 2, 0, 3, 0\}$, so we can split it in following way: $\{1, 2, 0\}$ and $\{3, 0\}$.

In second query of second type the array is the same as in previous query and the XOR of all elements in it equals 0 so we are allowed to split the array into one big interval.
In third query of second type the array looks like $\{1, 2, 5, 3, 0\}$, so we can split it in following way: $\{1, 2, 5\}$ and $\{3, 0\}$.
It's impossible to split the array in last query of second type.

Problem K. KMR

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

KMR is an ACM ICPC team, whose members are dreaming about achieving a bronze medal at the finals. They've found in the internet new training course which can be described as DAG (directed acyclic graph). Vertices are states in training and edges are tasks to solve, vertices are numbered from 1 to n. Every programmer at the beginning of course is in vertex 1 and in any moment he can choose some edge which is outgoing from his vertex and solve problem assigned to it, which will effect in changing actual vertex of this programmer and moving him to end of this edge. Course ends when programmer is in vertex n. Tasks are pairwise different. All three members of KMR want to complete this course independently, but they know that it makes no sense if there is a task solved by all three of them, so they don't want to allow such situation to take place. They are wondering what is the number of ways to complete the course by their team. Your task is to calculate this number and to print it modulo $10^9 + 7$.

Two ways of completing the course by team are considered different if there exists a member and a task such that this member solved this task in the first way, but didn't solve it in the second way.

Input

The first line of the input contains two integers n and m ($2 \le n \le 2000$; $1 \le m \le 10000$) – the number of vertices and the number of edges (tasks). Each of the next m lines contains two integers a and b ($1 \le a, b \le n$) which denote an edge from vertex a to vertex b (representing a task which if solved will move programmer from vertex a to vertex b).

It's guaranteed that described graph is a DAG. It's also guaranteed that by starting in vertex 1 it's possible to achieve every other vertex and by starting in any vertex it's possible to achieve vertex n.

Output

Output one integer which stands for number of different ways of completing the course by KMR taken modulo $10^9 + 7$.

Example

standard input	standard output

4 5
1 2
2 3
3 4
1 3
2 4
	12

3 4
1 2
1 2
2 3
2 3
	36