Problem A. Automorphism

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	512 mebibytes

You are given a rooted tree. Initially, it contains one vertex labeled as 1.

Your task is to process m operations of two types:

- Add a new vertex to the tree.
- Calculate the number of automorphisms of the subtree rooted at vertex u.

As the numbers can be very large, find them modulo 998 244 353.

For a rooted tree, whose root is r and vertex set is S, the automorphism is a bijection $f : S \to S$ such that $f(r) = r$ and $\forall u, v \in S$, $f(u)$ is the parent of $f(v)$ if and only if u is the parent of v.

Input

The first line contains one integer m ($1 \le m \le 3 \cdot 10^5$).

In the following m lines, each line indicates an operation. Each of these lines contains two integers t and x ($0 \le t \le 1$).

If $t = 0$, add a new vertex labeled by the current maximum label plus 1. Add an edge between this new vertex and the vertex x.

If $t = 1$, calculate the number of automorphisms of the subtree of vertex x.

It is guaranteed that, for each operation, the value of x is between 1 and the current maximum label.

Output

For each calculate operation, print a single line with the number of automorphisms modulo 998 244 353.

Example

standard input	standard output
10 0 1 0 1 1 1 0 2 0 3 1 1 0 3 0 3 1 3 1 1	2 2 6 6

Problem B. Border

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

You are given a string $S[1 \dots n]$. We denote its substrings as $S[l \dots r]$, and when $l > r$, such substring is defined to be an empty string. Let

$$f(i,j) = \max\{k \mid 0 \le k \le j-i,\ S[i \dots i+k-1] = S[j-k+1 \dots j]\}.$$

Output $\sum_{1 \le i < j \le n} f(i,j)$.

The string S is generated in the following way. The values n and *seed* are the parameters of the generator.

```
long long seed;
for (int i = 1; i <= n; i++) {
    seed = (seed * 13331 + 23333) % 1000000007;
    s[i] = 'a' + (seed & 1);
}
```

Input

The first line contains two integers: n and *seed* ($1 \le n \le 10^6$, $0 \le seed \le 10^9 + 6$).

Output

Output the answer.

Example

standard input	standard output
10 233333	50

Problem C. Convolution

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

You are given two sequences $a_0, a_1, \ldots, a_n$ and $b_0, b_1, \ldots, b_n$. You want to compute a new sequence $c_0, c_1, \ldots, c_n$ such that

$$c_k = \left(\sum_{i=0}^{k} \binom{k}{i} a_i b_{k-i} \right) \bmod 2^{32}.$$

Here, $\binom{k}{i} = \frac{k!}{i!(k-i)!}$ are binomial coefficients.

Output $c_0, c_1, \ldots, c_n$.

Input

The first line contains an integer n ($1 \le n \le 2 \cdot 10^5$).

The second line contains $n+1$ integers $a_0, a_1, \ldots, a_n$ ($0 \le a_i < 2^{32}$).

The third line contains $n+1$ integers $b_0, b_1, \ldots, b_n$ ($0 \le b_i < 2^{32}$).

Output

Output one line with $n+1$ integers: $c_0, c_1, \ldots, c_n$.

Example

standard input	standard output
5 0 1 2 3 4 5 6 7 8 9 10 11	0 6 26 84 240 640

Problem D. Decomposition

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

You are given a string S. You want to partition it into several (possibly one) nonempty substrings. There are $2^{|S|-1}$ ways of partitions. For example, "aba", "ab"+"a", "a"+"ba", "a"+"b"+"a" are all the partitions of the string "aba".

We define the weight of a substring T as the length of the shortest string x such that $T = xx \ldots x$. For example, the weight of "aaa" is 1, and the weight of "ab" is 2. We define the weight of a partition as the product of the weights of all substrings in this partition.

Output the sum of weights of all partitions. The answer can be large, so output the answer modulo $10^9 + 7$.

Input

The first line contains an integer T ($1 \le T \le 10^5$) indicating the number of test cases.

Each test case is given on a separate one line containing a string S ($1 \le |S| \le 2 \cdot 10^5$) consisting of lowercase English letters.

It is guaranteed that $\sum |S| \le 10^6$.

Output

For each test case, output the answer modulo $10^9 + 7$.

Example

standard input	standard output
1 abaababaabbbaabbbb	5320053

Problem E. Edit

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

You are given two weighted rooted trees. For each vertex, the children are ordered. You can assume they are arranged from left to right. You have four elementary operations: growing, expansion, contraction, and relabeling.

1. **Growing:** For a vertex x, let the children list be $y_1, y_2, \ldots, y_m$. You can add a new vertex z, add an edge between x and z, and insert the vertex z to the k-th position in the children list. The children list of x becomes $y_1, y_2, \ldots, y_{k-1}, z, y_k, y_{k+1}, \ldots, y_m$. The cost of this operation is c_1 times the weight of the new edge (x, z).

2. **Expansion:** For a vertex x, let the children list be $y_1, y_2, \ldots, y_m$. You can choose an interval $[l, r]$ $(1 \le l \le r \le m)$. Add a new vertex z as the parent of vertices $y_l, y_{l+1}, \ldots, y_r$, and an edge between x and z. The children list of x becomes $y_1, y_2, \ldots, y_{l-1}, z, y_{r+1}, \ldots, y_m$. The children list of z becomes $y_l, y_{l+1}, \ldots, y_r$. For all $l \le i \le r$, the weight of the edge (z, y_i) is the same as the edge (x, y_i) in the original tree. The cost of this operation is c_1 times the weight of the new edge (x, z).

3. **Contraction:** For a vertex x, let the children list be $y_1, y_2, \ldots, y_m$. You can choose one of its children y_k. Let the children list of y_k be $z_1, z_2, \ldots, z_p$. You can contract the edge (x, y_k). The vertex y_k is removed after this operation. And the children list of x becomes $y_1, y_2, \ldots, y_{k-1}, z_1, z_2, \ldots, z_p, y_{k+1}, \ldots, y_m$. For all $1 \le i \le p$, the weight of the edge (x, z_i) is the same as the edge (y_k, z_i) in the original tree. The cost of this operation is c_2 times the weight of the edge (x, y_k) in the original tree.

4. **Relabeling:** For a vertex x and one of its children y, change the weight of edge (x, y) from w_1 to w_2. The cost of this operation is $c_3 \cdot |w_1 - w_2|$.

There are also some special rules:

- You can not relabel an edge which is the (x, z) edge added by growing or expansion operation.
- You can not contract an edge which was relabeled.

You want to perform these operations, and change the first tree into the second tree. Output the minimum cost of doing so.

Two trees are considered the same if and only if there is a bijection from the vertices of the first tree to the vertices of the second tree that preserves the root and the order of children, and the weights of corresponding edges are the same.

Input

The first line contains three integers c_1, c_2, and c_3 $(1 \le c_1, c_2, c_3 \le 10^6)$ indicating the cost of growing (or expansion), contraction, and relabeling, respectively.

Next, the two trees are given.

For each tree, the first line contains an integer n indicating the number of vertices. In the following n lines, each line starts an with integer k indicating the number of children, followed by $2k$ integers $c_1, w_1, \ldots, c_k, w_k$ $(0 \le c_i \le 10^6)$ indicating the children list and the weights of edges to children.

The size of the first tree will not be greater than 50. The size of the second tree will not be greater than 2000.

Output

Output the answer.

Example

standard input	standard output
1 1 2 4 2 2 5 4 2 1 3 1 0 0 3 2 2 1 3 2 0 0	5

Problem F. Flow

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

We are considering a maximum flow problem on an infinite network.

You are given a bipartite graph G with n vertices in both parts and m directed edges. Each edge goes from the left part to the right part, and has its capacity specified. We want to construct a family of networks $\{F_k\}$.

Here are the steps to construct the network F_k.

- We first produce k copies of the graph G. We call these copies $G_1, G_2, \ldots, G_k$.
- For all $1 \le i \le k-1$ and $1 \le u \le n$, we add a directed edge from u-th vertex in the right part of G_i to u-th vertex in the left part of G_{i+1} with infinite capacity.
- We add directed edges from the source to all the vertices in the left part of G_1 with infinite capacity.
- We add directed edges from all the vertices in the right part of G_k to the sink with infinite capacity.

Let f_k be the maximum flow in the network F_k.

We want to know what the sequence $\{f_k\}$ looks like when k goes to infinity. If $\{f_k\}$ does not converge to a constant, output -1. Otherwise, output $\lim_{k \to +\infty} f_k$.

Input

The first line contains two integers n and m ($1 \le n \le 2000$, $1 \le m \le 4000$).

Each of the following m lines contains three integers u, v, and w ($1 \le u, v \le n$, $1 \le w \le 10^5$) which indicate that there is a directed edge from the u-th vertex in the left part to the v-th vertex in the right part with capacity w.

Output

If the sequence $\{f_k\}$ does not converge to a constant, output -1. Otherwise, output $\lim_{k \to +\infty} f_k$.

Example

standard input	standard output
5 5 1 2 3 2 3 4 3 1 2 4 5 6 5 4 3	12

Problem G. Good Game

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

You want to walk from $(0, 0, \ldots, 0)$ to $(a_0, a_1, \ldots, a_{n-1})$ in an n-dimensional space. On each step, you can increase one component of your coordinate vector by one. There are m obstacles $p_1, p_2, \ldots, p_m$. You want to find the number of paths that don't pass through the obstacles.

However, this problem is too simple for an ICPC contest in the year 8102. We add one more constraint. For every point $(x_0, x_1, \ldots, x_{n-1})$ on your path, the components of this vector should be non-decreasing: $x_0 \le x_1 \le \ldots \le x_{n-1}$.

Output the number of paths modulo $10^9 + 7$.

Input

The first line contains two integers n and m ($1 \le n \le 50$, $0 \le m \le 50$).

The second line contains n integers $a_0, a_1, \ldots, a_{n-1}$ ($0 \le a_0 \le a_1 \le \ldots \le a_{n-1} \le 10^4$), the coordinate vector of your destination.

The following m lines describe obstacles. The i-th of these lines contains n integers $p_{i,0}, p_{i,1}, \ldots, p_{i,n-1}$ ($0 \le p_{i,0} \le p_{i,1} \le \ldots \le p_{i,n-1} \le 10^4$), the coordinate vector of an obstacle.

The starting point, destination, and all the obstacles are distinct.

Output

Output the answer modulo $10^9 + 7$.

Examples

standard input	standard output
2 0 3 3	5
4 2 1 2 3 4 0 1 2 3 1 1 2 2	312

Problem H. Hamilton Path

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

You are given a directed graph with n vertices and m edges. The vertices are labeled from 1 to n. You need to find all the permutations of vertices $p_1, p_2, \ldots, p_n$ satisfying the following constraint:

- For all $1 \le i < j \le n$, an edge (p_i, p_j) exists if and only if $j = i + 1$.

We define the value of a permutation $p_1, p_2, \ldots, p_n$ as

$$\left(\sum_{i=1}^{n} p_i \cdot 10^{n-i}\right) \bmod (10^9 + 7).$$

Output the number of such permutations modulo $10^9 + 7$. If the number of such permutations is not greater than n, you also need to consider them all in lexicographical order, and output their values in this order.

Input

The first line contains an integer T ($T \le 10^5$) indicating the number of test cases.

For each test case, the first line contains two integers n and m ($n \ge 1$, $m \ge 0$, $1 \le \sum n \le 5 \cdot 10^5$, $1 \le \sum m \le 10^6$).

Each of the following m lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$) indicating that there is a directed edge from u to v in the graph. Note that the graph can contain parallel edges.

Output

For each test case, output the number of the permutations modulo 10^9+7 in the first line. If the number of permutations is not greater than n, print another line with space-separated values of all the permutations, considered in lexicographical order. You **don't need to** output an empty line if the number is greater than n or there is no solution.

Example

standard input	standard output
1 5 6 3 4 2 5 5 3 1 3 4 2 5 1	2 13425 34251

Problem I. IOI Problem Revised

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	512 mebibytes

There was an IOI problem 6102 years ago:

There is a straight highway with villages alongside the highway. The highway is represented as an integer axis, and the position of each village is identified with a single integer coordinate. There are no two villages in the same position. The distance between two positions is the absolute value of the difference of their integer coordinates.

Post offices will be built in some, but not necessarily all of the villages. A village and the post office in it have the same position. For building the post offices, their positions should be chosen so that the total sum of all distances between each village and its nearest post office is minimum.

However, this problem is too simple for an ICPC contest in the year 8102. You are supposed to solve a harder version.

There is a **circular** highway with length L. There are n villages alongside the highway. The position of each village is identified with a single integer coordinate. There **can be** two or more villages in the same position. The distance between two positions is the length of the shortest path along the highway. If there are two villages with coordinates a and b, the distance between them is $\min(|a-b|, L-|a-b|)$.

You want to build k post offices on the highway and minimize the total sum of all distances between each village and its nearest post office. Each post office has to be placed at an integer coordinate.

Input

The first line contains three integers n, k, and L ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le n$, $1 \le L \le 10^{12}$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_1 \le a_2 \le \ldots \le a_n < L$) indicating the coordinates of villages.

Output

On the first line, output the answer. On the second line, output k integers $c_1, c_2, \ldots, c_k$ ($0 \le c_i < L$, $c_i \le c_{i+1}$) indicating the coordinates of the post offices.

Example

standard input	standard output
5 2 10 1 3 4 7 9	5 3 7

Problem J. Jump Jump Jump

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

The rabbit starts at point $(0, 0)$ on the plane. The are k distinct vectors (dx_1, dy_1), (dx_2, dy_2), ..., (dx_k, dy_k). On each step, the rabbit will choose one vector (dx_c, dy_c) randomly with the same probability, and then jump from its current point (x, y) to $(x + dx_c, y + dy_c)$. All choices are independent.

There are traps in all the lattice points (x, x) for all $x \geq 1$. Once the rabbit jumps into a trap, it gets trapped and can not move anymore.

For each x such that $1 \leq x \leq n$, output the probability that the rabbit gets trapped in the lattice point (x, x).

Input

The first line contains two integers n and k ($1 \leq n \leq 10^5$, $1 \leq k \leq 16$).

Each of the following k lines contains two integers dx_i and dy_i ($0 \leq dx_i, dy_i \leq 3$) in each line. All the vectors are distinct.

Output

Print n lines. On line x, print the probability that the rabbit is trapped in the lattice point (x, x). It is guaranteed that the probability can be represented as a fraction A/B where B is coprime to 998 244 353, so output it as $A \cdot B^{-1} \bmod 998\,244\,353$.

Example

standard input	standard output
5 3 0 0 0 1 1 0	499122177 873463809 935854081 959250433 970948609

Note

The probabilities are $\frac{1}{2}, \frac{1}{8}, \frac{1}{16}, \frac{5}{128}, and \frac{7}{256}$.

Problem K. Knight

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

There is a chessboard with n rows and m columns. Some squares on the chessboard are broken. There are two knights on the chessboard, controlled by Alice and Bob. The movement of a knight is determined by two parameters r and c. On each step, Alice or Bob can move their knight to a square which is r squares away horizontally and c squares vertically, or r squares away vertically and c squares horizontally.

Alice and Bob take turns playing, starting with Alice. On each turn, the player moves his or her knight. However, the player can not move the knight to a square which is broken or is occupied by the other knight.

There is an extra constraint. The configuration of the knights can be viewed as an ordered pair (a, b) where a is Alice's square and b is Bob's square. It is forbidden to repeat a configuration which already occurred earlier.

A player loses if he or she can not make a move on his or her turn. Determine the winner if both players play optimally.

Input

The first line contains four integers n, m, r, and c ($1 \le n, m \le 1000$, $0 \le r < n$, $0 \le c < m$).

Each of the following n lines contains a string of length m. Together, these lines describe the chessboard. There are four types for each square:

- "`@`": The square is broken.
- "`.`": The square is not broken.
- "`A`": The square is not broken. It is the start position of Alice's knight.
- "`B`": The square is not broken. It is the start position of Bob's knight.

It is guaranteed that the squares "`A`" and "`B`" both occur exactly once on the chessboard.

Output

Output the name of the winner: "`Alice`" or "`Bob`".

Example

standard input	standard output
2 3 1 2 A@. B@.	Alice

Note

On the first step, Alice moves the knight to the square $(2, 3)$.

On the second step, Bob moves the knight to the square $(1, 3)$.

On the third step, Alice moves the knight back to the square $(1, 1)$.

On the fourth step, Bob can not move the knight back to the square $(2, 1)$, because it will create the ordered pair of squares $(1, 1), (2, 1)$ which is the same as the position in the beginning. Alice wins.

Problem L. Link Cut Digraph

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

After reading the paper *Incremental Topological Ordering and Strong Component Maintenance*, you came up with the following problem.

You are given a graph with n vertices. There are no edges initially. There are m operations. Each operation is first to add a given directed edge to the graph, and then to output the number of pairs (u, v) $(1 \le u < v \le n)$ such that u is reachable from v and v is reachable from u.

Can you implement the algorithm described in the paper in an ICPC contest?

Input

The first line contains two integers n and m ($1 \le n \le 10^5$, $1 \le m \le 2.5 \cdot 10^5$).

Each of the following m lines contains two integers u and v ($1 \le u, v \le n$) indicating a newly added directed edge. Parallel edges and self-loops are allowed.

Output

Output m integers, one per line: the requested number of pairs after adding each given edge.

Example

standard input	standard output
4 6 1 2 2 3 2 1 3 4 4 3 3 2	0 0 1 1 2 6