Problem A. Minimum Spanning Trees

Input file: standard input
Output file: standard output

One day, Subconscious is facing a problem that reminds him of the horrible experience in a past contest. In that contest, the easiest problem requires to count the number of minimum spanning trees in a randomly generated graph.

Now there is a graph with n vertices labeled from 1 to n. For each pair of vertices u and v ($1 \le u < v \le n$), there will be no edge between them with probability $\frac{p_0}{100}$, or an edge of weight 1 with probability $\frac{p_1}{100}$, or an edge of weight 2 with probability $\frac{p_2}{100}$, ..., or an edge of weight k with probability $\frac{p_k}{100}$, where k is the maximum weight of an edge.

However, your task is not finding the expected weight of the minimum spanning tree of the randomly generated graph since the graph may be disconnected and the task author does not know how to deal with such cases.

Thus, your task becomes, for each integer s between $n-1$ and $k(n-1)$, calculating the probability that the graph is connected and the weight of the minimum spanning tree of the graph is exactly s.

The problem is so hard that even if our talent Subconscious has managed to solve it, he is not sure whether his solution works and wants to check the answers modulo 1 000 000 007 with you.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 200$), the number of test cases.

For each test case, the first line contains two integers n ($2 \le n \le 40$) and k ($1 \le k \le 4$), denoting the number of vertices of the randomly generated graph and the maximum weight of an edge.

The following line contains $k+1$ non-negative integers $p_0, p_1, p_2, \cdots, p_k$ ($\sum_{i=0}^{k} p_i = 100$), describing the probability distribution of an edge between each pair of vertices.

It's guaranteed that no more than 20 test cases satisfy $n > 10$ and no more than 2 test cases satisfy $n > 20$.

Output

For each test case, with given n and k, output a line containing $(k-1)(n-1)+1$ integers, the i-th of which is the probability that the graph is connected and the weight of the minimum spanning tree of the graph is exactly $n-2+i$, modulo 1 000 000 007.

More precisely, if the reduced fraction of the probability is $\frac{p}{q}$, what you should provide is the minimum non-negative integer r such that $qr \equiv p \pmod{1\,000\,000\,007}$. You may safely assume that such r always exists in all test cases.

Example

standard input	standard output
2 3 1 50 50 3 2 0 50 50	500000004 500000004 375000003 125000001

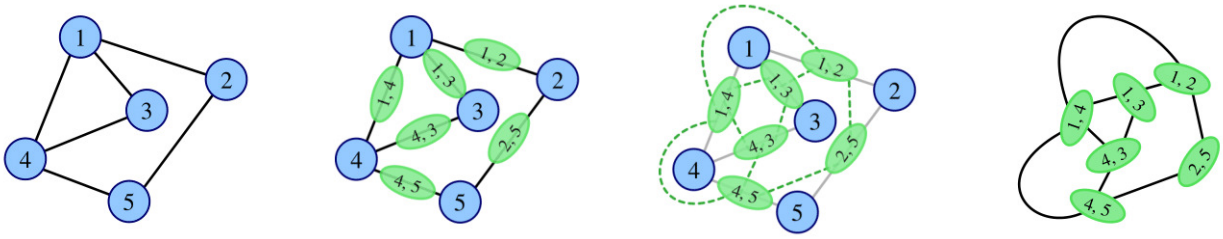
Problem B. Line Graphs

Input file: `standard input`
Output file: `standard output`

Last year, Rounddog participated in a contest with a pretty hard problem set and failed in solving the problem about line graphs. So he decides to make a deep study of line graphs these days.

In the mathematical discipline of graph theory, the line graph of a simple undirected graph G is another simple undirected graph $L(G)$ that represents the adjacency between every two edges in G. Precisely speaking, for an undirected graph G without loops or multiple edges, its line graph $L(G)$ is a graph such that

- each vertex of $L(G)$ represents an edge of G; and
- two vertices of $L(G)$ are adjacent if and only if their corresponding edges share a common endpoint in G.



Given a simple undirected graph G, Rounddog's study aims to find the maximum cliques in its line graph $L(G)$ and he decides to make some critical results of his early study as a challenge for you.

In this problem, you are given a simple undirected graph G and a small positive integer k. After finding all maximum cliques in $L^k(G)$, where $L^0(G) = G$ and $L^s(G) = L(L^{s-1}(G))$ for each positive integer s, you need to show Rounddog the number of vertices in one of the maximum cliques and the number of distinct maximum cliques modulo 1 000 000 007.

Here a subset of vertices of an undirected graph is called clique if and only if there is an edge between each pair of vertices in the subset, and maximum cliques are those cliques with the maximum number of vertices.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 1\,000$), the number of test cases.

For each test case, the first line contains three integers n ($1 \le n \le 100\,000$), m ($0 \le m \le 200\,000$) and k ($1 \le k \le 4$), the number of vertices and edges in the given simple undirected graph G and the number of iterations of the line graph operation.

Then m lines follow, describing all edges of the graph. Each line of them contains two integers u and v ($1 \le u, v \le n, u \ne v$), representing an edge between the u-th vertex and the v-th vertex.

It is guaranteed that the sum of n in all test cases does not exceed 2 000 000, the sum of m does not exceed 3 000 000, and the graph in each test case contains no loops or multiple edges.

Output

For each test case, output a single line with two integers, the number of vertices in one of the maximum cliques and the number of distinct maximum cliques modulo 1 000 000 007.

Example

standard input	standard output
3 5 0 4 5 4 1 1 2 1 3 1 4 1 5 5 4 4 1 2 1 3 1 4 1 5	0 1 4 1 6 12

Problem C. Valentine's Day

Input file: standard input
Output file: standard output

Oipotato loves his girlfriend very much. Since Valentine's Day is coming, he decides to buy some presents for her.

There are n presents in the shop, and Oipotato can choose to buy some of them. We know that his girlfriend will possibly feel extremely happy if she receives a present. Therefore, if Oipotato gives k presents to his girlfriend, she has k chances to feel extremely happy. However, Oipotato doesn't want his girlfriend to feel extremely happy too many times for the gifts.

Formally, for each present i, it has a possibility of P_i to make Oipotato's girlfriend feel extremely happy. Please help Oipotato decide what to buy and maximize the possibility that his girlfriend feels extremely happy for **exactly** one time.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \leq T \leq 100$), indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \leq n \leq 10\,000$), indicating the number of possible presents.

The second line contains n decimals P_i ($0 \leq P_i \leq 1$) with exactly six digits after the decimal point, indicating the possibility that Oipotato's girlfriend feels extremely happy when receiving present i.

It is guaranteed that the sum of n in all test cases does not exceed $450\,000$.

Output

For each test case output one line, indicating the answer. Your answer will be considered correct if and only if the absolute error of your answer is less than 10^{-6}.

Example

standard input	standard output
2 3 0.100000 0.200000 0.900000 3 0.100000 0.300000 0.800000	0.900000000000 0.800000000000

Problem D. Play Games with Rounddog

Input file: `standard input`
Output file: `standard output`

As a typical Nim game, picking stones is very famous all over the world with its quite simple rule. Initially, there are several piles of stones. Two players take turns to remove at least one stone from one of the piles. Whoever cannot make any legal move loses this game.

On August 17th, a very special day, Rounddog and Calabash create another picking stones game of their own. The new rule is as the following.

To start with, Calabash takes out a string S from his right pocket as the keystone of their game, which has m round in total.

At the beginning of each round, their common friend Severus will select a substring T from S. Then before they officially start playing, there are three phases need to be applied.

In Phase 1, Calabash will select several **distinct** substrings from S, satisfying that they all have a suffix T. Taking "`claris`" as an example, one of its suffixes is "`ris`".

Phase 2 requires some magical power. Calabash will turn all strings he selects to stone piles. Specifically, for each string X he chooses, it will become a pile of W_p stones where p is the number of occurrences of X in S. For example, "`aba`" occurs in "`ababa`" for 2 times.

Rounddog will be in charge of Phase 3. After Severus and Calabash's movement, Rounddog chooses some piles from Calabash's selection, and throws them away. But Rounddog can't throw all the piles Calabash selected, because it will lead the game to the end immediately.

With the left piles, Rounddog and Calabash will start playing based on the original rule of Picking Stones. Calabash always moves first.

Now, our beloved Quailty wants to know whether Calabash will win in each round if they both perform the optimal strategy. Furthermore, he also wants you to calculate **the maximum total number of stones** Calabash can achieve in Phase 2 on the premise of his victory.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 3$), the number of test cases.

In each test case, the first line contains an integer n ($1 \le n \le 100\,000$).

The second line contains a string S of length n with only lowercase English letters.

The third line contains n integers, i-th of which is W_i ($1 \le W_i < 2^{58}$).

The fourth line contains an integer m ($1 \le m \le 200\,000$), representing the number of games.

Each of the next m lines contains two integers l and r ($1 \le l \le r \le n$) meaning that in this game, Severus will select $S[l, r]$ as T.

Output

For each test case, output m lines, each of which contains an integer representing the maximum total number of stones Calabash can achieve on the premise of his victory, or -1 if he always loses.

Example

standard input	standard output
1 5 aabab 1 3 5 7 9 5 1 1 1 2 2 2 2 3 3 5	6 1 6 4 1

Problem E. Welcome Party

Input file: `standard input`
Output file: `standard output`

The annual welcome party of the Department of Computer Science and Technology is coming soon! Many students have been applying to show up at the welcome party, and every one of them can choose to sing a song or play crosstalk. This troubles the chief director a lot: how to arrange the program list, such that every student can have a chance to show up on the stage, and the satisfactory value of audiences is maximized?

To cope with this problem, the director proposes a model. In this model, every student has two attributes: the singing ability and crosstalking ability. The satisfactory value of audiences to singings is the maximum singing ability among all students that choose to sing a song; similarly, the satisfactory value to crosstalks is the maximum crosstalking ability among all students that choose play crosstalk. The strange thing is, the overall satisfactory value to the whole party is negatively related to the absolute difference between the satisfactory values to singings and crosstalks. The problem is, what is the minimum possible absolute difference between the satisfactory values of the two types of programs?

Note that:

- every student should choose exactly one type of programs to play;
- at least one student should sing a song, and at least one student should play crosstalk.

Input

The first line of input consists of a single integer T ($1 \leq T \leq 70$), the number of test cases.

Each test case starts with a line of a single integer n ($2 \leq n \leq 100\,000$), denoting the number of students applying to show up on the stage. Then follow n lines, each containing two integers x and y ($0 \leq x, y \leq 10^{18}$), denoting the singing ability and crosstalking ability of a student.

It is guaranteed that the sum of n over all test cases never exceeds $1\,000\,000$.

Output

For each test case, output a single integer, denoting the minimum possible absolute difference between the satisfactory values of the two types of programs.

Example

standard input	standard output
2 5 27 46 89 13 55 8 71 86 22 35 3 3 5 4 7 6 2	3 1

Problem F. Dense Subgraph

Input file: `standard input`
Output file: `standard output`

You have a tree on n vertices, and each vertex has a weight a_v and a degree at most 5.

Let's call the **density** of a subset of vertices S value $\frac{\sum_{v \in S} a_v}{|S|}$, and call the **beauty** of the tree with some vertices turned off the maximum value of the **density** of a subset of at least two turned on vertices with connected induced subgraph, or 0 if no such subset exists.

Now you need to calculate the number of ways to choose such a set of turned off vertices among 2^n ways that the **beauty** of the tree is no larger than x, modulo 1 000 000 007.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 30$), the number of test cases.

The first line of each test case contains two integers n ($2 \le n \le 35\,000$) and x ($0 \le x \le 35\,000$), the number of vertices of a tree and the constraint on the beauty.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 35\,000$), the weights of the tree vertices.

Each of the next $n - 1$ lines contains two integers u and v ($1 \le u, v \le n$), describing an edge connecting vertices u and v in the tree.

It is guaranteed that each vertex of a tree has a degree at most 5.

Output

For each test case, output a line containing a single integer, indicating the number of ways to choose such a set of turned off vertices among 2^n ways that the **beauty** of the tree is no larger than x, modulo 1 000 000 007.

Example

standard input	standard output
2 5 0 1 1 1 1 1 1 2 2 3 3 4 4 5 3 2 2 1 3 1 2 1 3	13 6

Problem G. Closest Pair of Segments

Input file: `standard input`
Output file: `standard output`

The closest pair of points problem is a well-known problem of computational geometry. In this problem, you are given n points in the Euclidean plane and you need to find a pair of points with the smallest distance between them.

Now, Claris, the brilliant one who has participated in programming contests for several years, is trying to solve a harder problem named the closest pair of segments problem, which also has a quite simple description as above.

However, the problem seems even too hard for Claris and she is asking you for help.

Now n segments are lying on the Euclidean plane, you are asked to pick two different segments and then pick a point on the two segments respectively to minimize the distance between these two points.

For simplicity, any two given segments share no common point, and you don't need to show her the two chosen points, but the distance between them instead.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 200$), the number of test cases.

For each test case, the first line contains one integer n ($2 \le n \le 10\,000$), which is the number of segments on the Euclidean plane.

The following n lines describe all the segments lying on the Euclidean plane, the i-th of which contains for integers x_1, y_1, x_2 and y_2 describing a segment that connects (x_1, y_1) and (x_2, y_2), where $-10^9 \le x_1, y_1, x_2, y_2 \le 10^9$.

It's guaranteed that the two endpoints of each segment do not coincide, any two given segments do not intersect with each other in each test case, and no more than 20 test cases satisfy $n > 1\,000$.

Output

For each test case, output a line containing a single real number for the answer to the closest pair of segments problem with an absolute or relative error of at most 10^{-6}.

Precisely speaking, assume that your answer is a and and the jury's answer is b, your answer will be considered correct if and only if $\frac{|a-b|}{\max\{1,|b|\}} \le 10^{-6}$.

Example

standard input	standard output
2 2 0 1 1 2 1 1 2 0 2 0 1 1 2 2 2 3 1	0.707106781187 1.000000000000

Problem H. Coins

Input file: standard input
Output file: standard output

There are n groups of coins, and the i-th group contains two coins valued at a_i and b_i. Now you want to pick exactly k coins out of them. However, there is a restriction - you can not pick the second coin valued at b_i in the i-th group without picking the other one in the same group. In other words, in the i-th group, you can

- pick none of the two coins;
- pick only the first one valued at a_i; or
- pick both of them.

We now want to know the maximum sum of the k picked coins' values, denoted by $f(k)$.

Furthermore, we want to know $f(1), f(2), \cdots, f(2n)$.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \leq T \leq 90$), the number of test cases.

For each test case, the first line contains an integer n ($1 \leq n \leq 100\,000$), indicating the number of coin groups.

Each of the following n lines contains two integers a_i, b_i ($1 \leq a_i, b_i \leq 10\,000$) indicating the coin values in that group.

It is guaranteed that the sum of n in all test cases does not exceed $2\,100\,000$.

Output

For each test group, just output $2n$ integers in a single line representing $f(1), f(2), \cdots, f(2n)$, and adjacent integers should be separated by one space.

Example

standard input	standard output
2 3 1 2 1 4 4 2 2 1 3 3 2	4 6 9 11 12 14 3 5 7 9

Problem I. Block Breaker

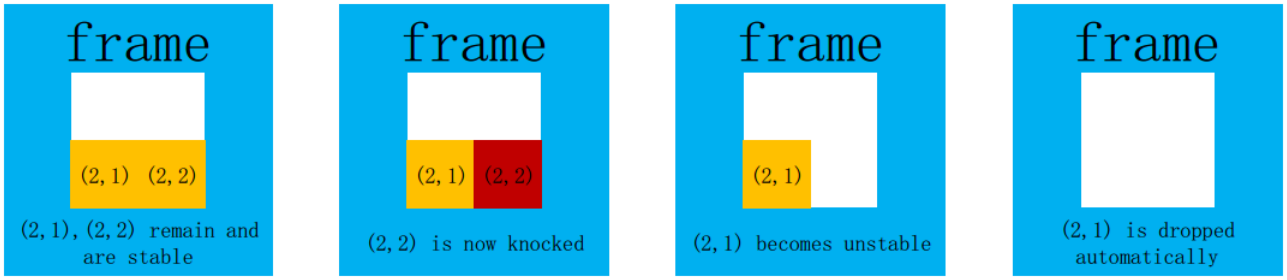
Input file: `standard input`
Output file: `standard output`

Given a rectangle frame of size $n \times m$. Initially, the frame is strewn with $n \times m$ square blocks of size 1×1. Due to the friction with the frame and each other, the blocks are stable and will not drop.

However, the blocks can be knocked down. When a block is knocked down, other remaining blocks may also drop since the friction provided by other remaining blocks may not sustain them anymore. Formally, a block will drop if it is knocked or not stable, which means that at least one of the left block and the right block has been dropped and at least one of the front block and the back block has been dropped. Especially, the frame can be regarded as a huge stable block, which means that if one block's left is the frame, only when its right block has been dropped and at least one of the front block and the back block has been dropped can it drop. The rest situations are similar.

Now you, the block breaker, want to knock down the blocks. Formally, you will do it q times. In each time, you may choose a position (x_i, y_i). If there remains a block at the chosen position, you will knock it down; otherwise, nothing will happen. Moreover, after knocking down the block, you will wait until no unstable blocks are going to drop and then do the next operation.

For example, please look at the following illustration, the frame is of size 2×2 and the block $(1, 1)$ and $(1, 2)$ have been dropped. If we are going to knock the block $(2, 2)$, not only itself but also the block $(2, 1)$ will drop in this knocking operation.



You want to know how many blocks will drop in total in each knocking operation. Specifically, if nothing happens in one operation, the answer should be regarded as 0.

Input

The first line contains one positive integer T ($1 \le T \le 10$), denoting the number of test cases.

For each test case:

The first line contains three positive integers n, m and q ($1 \le n, m \le 2\,000, 1 \le q \le 100\,000$), denoting the sizes in two dimensions of the frame and the number of knocking operations.

Each of the following q lines contains two positive integers x_i and y_i ($1 \le x_i \le n, 1 \le y_i \le m$), describing a knocking operation.

Output

For each test case, output q lines, each of which contains a non-negative integer, denoting the number of dropped blocks in the corresponding knocking operation.

Example

standard input	standard output
2 2 2 3 1 1 1 2 2 2 4 4 6 1 1 1 2 2 1 2 2 4 4 3 3	1 1 2 1 1 2 0 1 11

Problem J. Domino Covering

Input file: standard input
Output file: standard output

Elizur has an empty $n \times m$ grid and he wants to use some 1×2 and 2×1 dominoes to cover the **entire** grid. In the grid, each domino ought to cover exactly two adjacent squares and each square ought to be covered by exactly one domino, where two squares are adjacent if and only if they share a common side.

Obviously, he can achieve that if and only if at least one of n and m is even. In case that both n and m are odd, there is always a square that must be left empty. Hence, he intends to know in how many ways he can cover the entire grid. Two ways are considered different if and only if there exist two dominoes, each selected from one way, satisfying that they cover exactly one square at the same position.

Can you help him determine the answer? The answer may be exceedingly large, so he only requests you to tell him the answer modulo a **prime number** p.

Input

The first line contains a single integer T ($1 \leq T \leq 20\,000$), indicating the number of questions.

In the next T lines, each line contains three integers n ($1 \leq n \leq 35$), m ($1 \leq m \leq 10^{18}$) and p ($2 \leq p \leq 2^{30}$), representing a question he wants to ask you.

It is guaranteed that no more than $1\,000$ cases satisfy $n > 5$ or $m > 10^9$.

Output

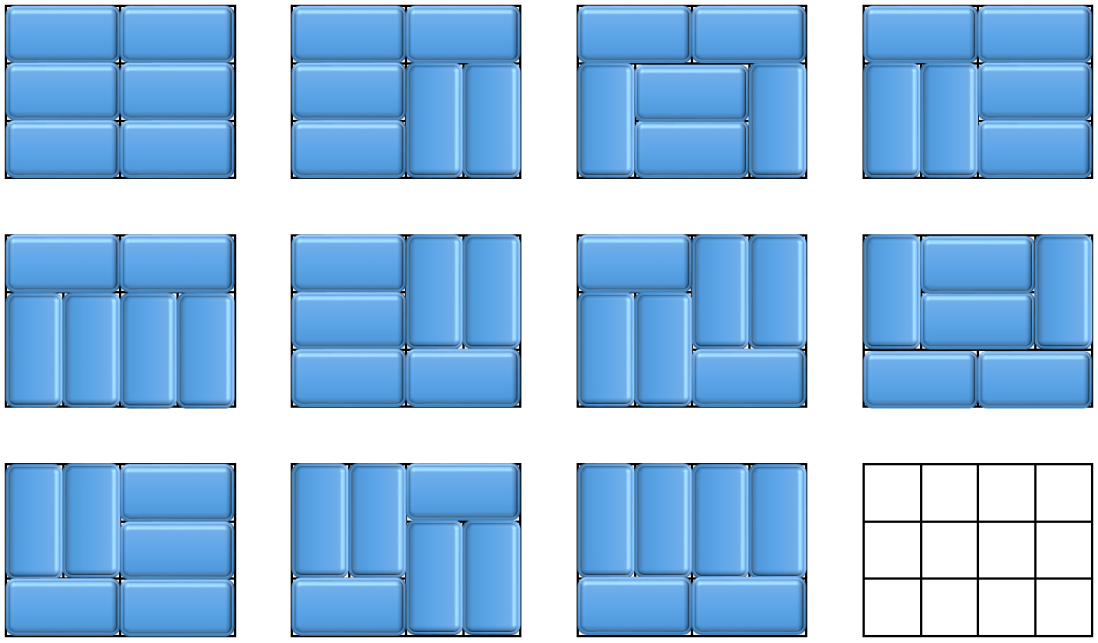
For each question, output a single integer in one line, denoting the answer modulo p.

Example

standard input	standard output
6 2 2 23 2 3 233 3 3 2333 3 4 23333 4 4 2332333 5 251346744251346744 998244353	2 3 0 11 36 295381485

Note

The following image shows all possible ways (11 in total) for the 3×4 grid.



Problem K. Make Rounddog Happy

Input file: `standard input`
Output file: `standard output`

Rounddog always has an array $a_1, a_2, \cdots, a_n$ in his right pocket, satisfying $1 \le a_i \le n$.

A subarray is a non-empty subsegment of the original array. Rounddog defines a good subarray as a subsegment $a_l, a_{l+1}, \cdots, a_r$ that all elements in it are different and $\max(a_l, a_{l+1}, \ldots, a_r) - (r - l + 1) \le k$.

Rounddog is not happy today. As his best friend, you want to find all good subarrays of a to make him happy. In this case, please calculate the total number of good subarrays of a.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 20$), the number of test cases.

The first line of each test case contains two integers n ($1 \le n \le 300\,000$) and k ($1 \le k \le 300\,000$).

The second line contains n integers, the i-th of which is a_i ($1 \le a_i \le n$).

It is guaranteed that the sum of n over all test cases never exceeds $1\,000\,000$.

Output

One integer for each test case, representing the number of subarrays Rounddog likes.

Example

standard input	standard output
2 5 3 2 3 2 2 5 10 4 1 5 4 3 6 2 10 8 4 5	7 31