Problem A. Minimum Spanning Trees

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

One day, Subconscious faced a problem that reminded him of the horrible experience in a past contest. In that contest, the easiest problem required 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 is 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. All edges are generated independently.

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 is this: for each integer s between $n-1$ and $k(n-1)$, calculate 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 won't be 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 $p_0, p_1, p_2, \ldots, p_k$ ($\sum_{i=0}^{k} p_i = 100$), describing the probability distribution of an edge between each pair of vertices.

It is 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 $q \cdot r \equiv p \pmod{1\,000\,000\,007}$. 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

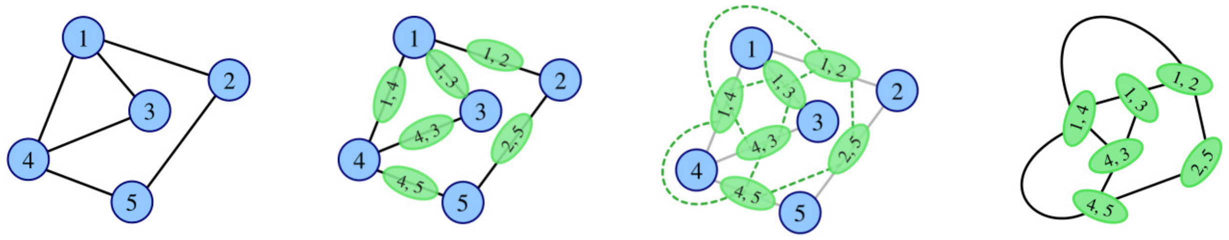
Problem B. Line Graphs

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

Last year, Rounddog participated in a contest with a pretty hard problem set and failed in solving the problem about line graphs. So recently he decided to study them deeper.

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 decided to turn some of his findings into 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 tell Rounddog the number of vertices in a maximum clique in $L^k(G)$, and also the number of distinct maximum cliques modulo 1 000 000 007.

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

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 1000$), 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 of them contains two integers u and v ($1 \le u, v \le n$, $u \ne v$), representing an edge between vertices u and v.

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 and no multiple edges.

Output

For each test case, output a single line with two integers: first the number of vertices in a maximum clique, and then 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*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Oipotato loves his girlfriend very much. Since Valentine's Day is coming, he decided 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, each present i will make Oipotato's girlfriend feel extremely happy with probability P_i. Oipotato now needs to decide what to buy in order to maximize the probability that his girlfriend feels extremely happy **exactly** once. Please help him find that maximum probability.

Input

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

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

The second line contains n real numbers P_i ($0 \le P_i \le 1$) given with exactly six digits after the decimal point, indicating the probability 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 $500\,000$.

Output

For each test case, output one line with 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*
Time limit:	5 seconds
Memory limit:	512 mebibytes

The stone-picking game Nim is very famous all over the world for its quite simple rules. 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 the game.

On August 17th, a very special day, Rounddog and Calabash create another stone-picking game of their own. The new rules are as follows.

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

At the beginning of each round, their common friend Severus will select a substring T from S. Then, before they actually start playing, there are three more preparation phases.

In Phase 1, Calabash will select one or more **distinct** substrings from S such that they all have suffix T. For example, "`ris`" is a suffix of "`claris`". A string is considered to be a suffix of itself.

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, $X =$ "`aba`" occurs in $S =$ "`ababa`" twice, so it will turn into a pile of two stones.

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

With the remaining piles, Rounddog and Calabash will start playing the classic Nim game. Calabash always moves first.

Now, our beloved Quailty wants to know whether Calabash will win in each round if he and Rounddog both play optimally. Furthermore, he also wants you to calculate **the maximum total number of stones** Calabash can create in Phase 2 so that he still wins if both players play optimally.

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 consisting of 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 rounds.

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

Output

For each test case, output m lines, one for each round. On each line, print the maximum total number of stones Calabash can create in Phase 2 of that round so that he still wins if both players play optimally, 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`
Time limit:	2 seconds
Memory limit:	512 mebibytes

The annual welcome party of the Department of Computer Science and Technology is coming soon! Many students have been invited to the party, and every one of them can choose to either sing a song or play crosstalk. This troubles the chief director a lot: how to arrange the performances so that every student shows up on the stage, and the satisfaction for the audience is maximized?

To cope with this problem, the director proposed a model. In this model, every student has two attributes: singing ability and crosstalking ability. The value of audience satisfaction by singing is the maximum singing ability among all students that choose to sing a song. Similarly, the value of audience satisfaction by crosstalking is the maximum crosstalking ability among all students that choose to play crosstalk. The strange thing is, the overall satisfaction value of the whole party is negatively related to the absolute difference between the satisfaction values by singing and crosstalking. The problem is, what is the minimum possible absolute difference between the satisfaction values of the two types of performance?

Note that:

- every student should choose exactly one type of performance 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 ($1 \le T \le 70$), the number of test cases.

Each test case starts with a line containing a single integer n ($2 \le n \le 100\,000$), denoting the number of students invited to the party. Then follow n lines, each containing two integers x and y ($0 \le x, y \le 10^{18}$), 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: the minimum possible absolute difference between the satisfaction values of the two types of performance.

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`
Time limit:	3 seconds
Memory limit:	512 mebibytes

You have a tree on n vertices. Each vertex v has weight a_v, and its degree is at most 5.

The **density** of a subset S of vertices is the value

$$\frac{\sum\limits_{v \in S} a_v}{|S|}.$$

Consider a subset L of the tree vertices. The **beauty** of L is the maximum **density** of S such that it is a subset of L, contains at least two vertices and forms a connected induced subgraph, or 0 if no such S exists.

There are 2^n ways to choose L. How many such L have their **beauty** no larger than x? As the answer can be very large, find it 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 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 the given graph is a tree. **It is also guaranteed that each vertex has degree at most 5.**

Output

For each test case, output a line containing a single integer: the number of ways to choose such a subset L of tree vertices that the **beauty** of L 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*
Time limit:	12 seconds
Memory limit:	512 mebibytes

The closest pair of points problem is a well-known problem in computational geometry. In this problem, you are given n points on 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 too hard, even for Claris, and she is asking you for help.

Now n segments are lying on the Euclidean plane. You have to pick two different segments, and then pick a point on each of them. Do it in such a way that the distance between these two points is the minimum possible.

For simplicity, no two given segments share a common point. Also, you don't need to show her the two points: just find the minimum possible distance between them instead.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 100$): the number of test cases.

For each test case, the first line contains one integer n ($2 \le n \le 100\,000$), 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 these lines contains four 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 is guaranteed that, in each test case, the two endpoints of each segment do not coincide, and no two segments share a common point. It is also guaranteed that the sum of n in all test cases does not exceed 100 000.

Output

For each test case, output a line containing a single real number: 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
	0.707106781185
	1.000000000001

Problem H. Coins

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

There are n groups of coins, and the i-th group contains two coins valued as 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 (the one valued as 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 as a_i; or
- pick both of them.

Let $f(k)$ be the maximum total value if we pick exactly k coins.

Find the values $f(1), f(2), \ldots, f(2n)$.

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 and 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 case, just output $2n$ integers on a single line representing $f(1), f(2), \ldots, f(2n)$. Separate consecutive integers by single spaces.

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

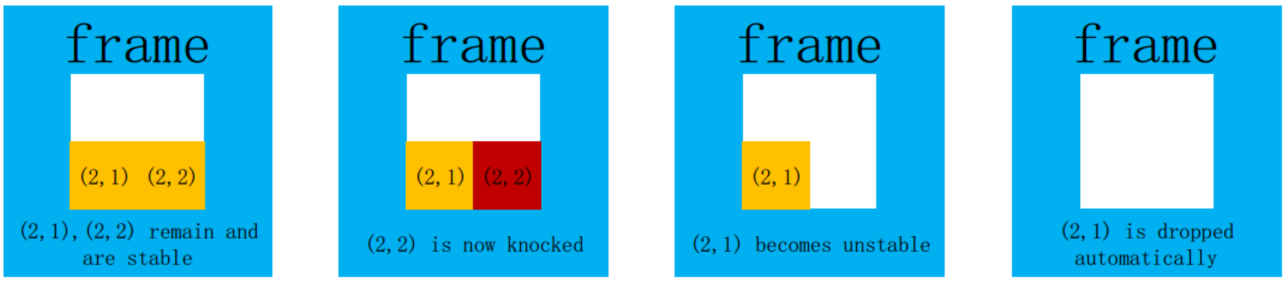
Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider a rectangle frame of size $n \times m$ hanging in the air horizontally. Initially, the frame is filled tightly 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 unstable. A block is unstable when at least one of its left and right neighbors has dropped, and at least one of its front and back neighbors has also dropped. In this definition, the frame can be regarded as a huge block that is always stable.

Now you, the block breaker, want to knock down the blocks. Formally, you are going to make q moves. On i-th move, you choose position (x_i, y_i). If there is still a block at the chosen position, you knock it down; otherwise, nothing happens. After each move, you have to wait until no unstable blocks are going to drop before making the next move.

For example, please look at the following illustration. The frame is of size 2×2, and the blocks $(1, 1)$ and $(1, 2)$ have dropped already. If we knock down the block $(2, 2)$, it will drop, and then the last remaining block $(2, 1)$ will also drop because it will become unstable.



You are given a sequence of moves to make. How many blocks will drop as a result of each move? Specifically, if nothing happens during a move, the answer for that move is 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 $(1 \le n, m \le 2000,\ 1 \le q \le 100\,000)$, denoting the dimensions of the frame and the number of moves.

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 the next move to make.

Output

For each test case, output q lines. Each of them must contain a non-negative integer: the number of blocks that will drop as a result of the corresponding move.

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*
Time limit:	6 seconds
Memory limit:	512 mebibytes

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. 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: otherwise, there is always a square that must be left empty. Hence, he wants 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, one from the first covering and one from the other, such that one of the squares cover is the same but the other is different.

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

Input

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

Each of the next T lines contains three integers, n $(1 \le n \le 35)$, m $(1 \le m \le 10^{18})$, and p $(2 \le p \le 2^{30}$, p is prime), describing one question.

It is guaranteed that no more than 1000 cases satisfy $n > 5$ or $m > 10^9$.

Output

For each question, output a single line with a single integer: the answer modulo p.

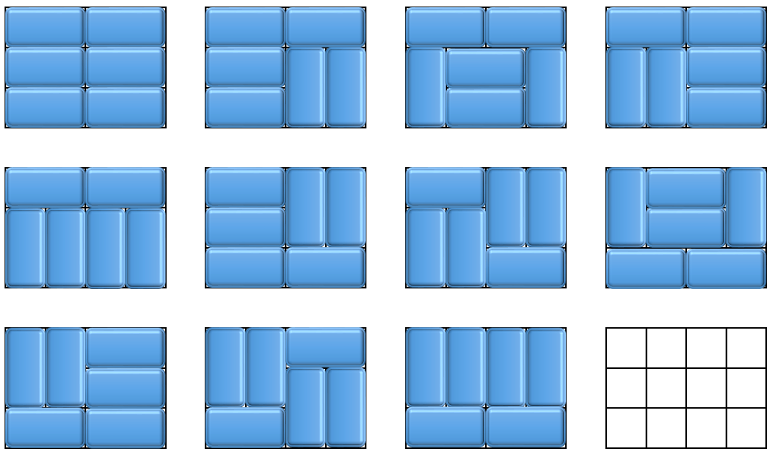
Example

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*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Rounddog always has an array $a_1, a_2, \ldots, a_n$ 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 $a_l, a_{l+1}, \ldots, a_r$ such 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. For this problem, 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

For each test case, print a single line with a single integer: the number of good subarrays in the given array.

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