The 2019 China Collegiate Programming Contest

CCPC 2019

Harbin Site, October 13

Problems

A Artful Paintings
B Binary Numbers
C Competition in Swiss-system
D Driverless Car
E Exchanging Gifts
F Fixing Banners
G Game Store
H Highway Buses
I Interesting Permutation
J Justifying the Conjecture
K Keeping Rabbits
L LRU Algorithm

Do not open before the contest has started.

Prepared by Zhejiang University

Problem A. Artful Paintings

Painting is really a romantic activity. Painting on cubes is even more romantic. And creating artful paintings on cubes is the most romantic thing to do.

N cubes are lined up in a row and they are indexed from 1 to N from left to right. The task for you is to choose some of the cubes and paint them. However, there are some rules that you must not violate, so that the painting will be artful. There are two types of rules, described below:

1. The number of painted cubes whose index belongs to the interval $[L_i, R_i]$ must not be strictly fewer than K_i $(1 \le i \le M_1)$.
2. The number of painted cubes whose index does not belong to the interval $[L_i, R_i]$ must not be strictly fewer than K_i $(1 \le i \le M_2)$.

Painting is also a tiring activity, so the number of cubes you paint should be as small as possible.

Input

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

For each case, the first line of the input contains three integers N, M_1 and M_2 $(1 \le N \le 3\,000$, $0 \le M_1, M_2 \le 3\,000)$, denoting the number of cubes, the number of rules of type 1, and the number of rules of type 2. The next M_1 lines each contains three integers L_i, R_i, K_i $(1 \le i \le M_1, 1 \le L_i \le R_i \le N, 0 \le K_i \le R_i - L_i + 1)$, describing a rule of type 1. The next M_2 lines each contains three integers L_i, R_i, K_i $(1 \le i \le M_2, 1 \le L_i \le R_i \le N, 0 \le K_i \le N - (R_i - L_i + 1))$, describing a rule of type 2.

It is guaranteed that the sum of N over all cases doesn't exceed $3\,000$, the same is true for the sum of M_1 and the sum of M_2.

Output

For each case, print a single integer in a single line, the smallest number of cubes you need to paint.

Example

standard input	standard output
1 3 1 1 1 2 1 2 2 1	1

Problem B. Binary Numbers

ZYB is a smart guy. He is figuring out a problem about binary numbers.

Let a_i be the i-th (0-based indexing) least significant bit in the binary representation of a. For example, if $a = 2$, then $a_0 = 0, a_1 = 1$, and $a_i = 0$ for $i \geq 2$. Define $F_k(a, b)$ as:

$$F_k(a,b) = \begin{cases} F_{k-1}(a,b) + 1 & (a_k = b_k) \\ 0 & (a_k \neq b_k) \end{cases}$$

Specifically, $F_{-1}(a, b) = 0$.

ZYB has N intervals $[L_1, R_1]$, $[L_2, R_2]$, $\ldots$, $[L_n, R_n]$ where $L_1 = 0$, $R_n = 2^m - 1$, and $\forall i \in [1, n-1] : R_i + 1 = L_{i+1}$. He wants to select N integers $A_1, A_2, \ldots, A_n$ so that $A_i \in [L_i, R_i]$, and for each $i \in [1, n]$, the following condition holds:

$$\forall k \in [L_i, R_i] : F_{m-1}(A_i, k) \geq \max_{1 \leq j \leq N} \{F_{m-1}(A_j, k)\}$$

The value of the sequence A is defined as $V(A) = \prod_{i=1}^{n} A_i$.

Of course, there are multiple ways to select the sequence A. Now ZYB wonders what is the sum of $V(A)$ among all different sequences he can select. Two sequences P and Q are different if and only if $\exists i \in [1, n] : P_i \neq Q_i$. Because the answer could be quite large, he only wants to know the answer modulo 100 000 007.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \leq T \leq 100$), the number of cases.

For each case, the first line of the input contains two integers m ($0 \leq m \leq 17$) and N ($1 \leq N \leq 2^m$). The following N lines each contains two integers L_i and R_i ($0 \leq L_i \leq R_i < 2^m$).

It's guaranteed that the sum of 2^m over all cases doesn't exceed 2^{18}.

Output

For each case, print a single integer in a single line, the sum of $V(A)$ modulo 100 000 007.

Example

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

Problem C. Competition in Swiss-system

A Swiss-system tournament is a non-eliminating tournament format that features a set number of rounds of competition, but considerably fewer than that of a round-robin tournament. In a Swiss tournament, each competitor (team or individual) does not necessarily play all other entrants. A Swiss system is used for competitions in which the number of entrants is considered too large for a full round-robin to be feasible, and eliminating any competitors before the end of the tournament is undesirable. Swiss systems are commonly used in chess, bridge and many other tabletop games such as *Magic: the Gathering* and *Yu-Gi-Oh*. The following describes the rules of a certain tournament with Swiss-system.

Tournament structure: The tournament consists of m rounds and is contested by n players. The players are given unique IDs from 1 to n for convenience. In each round, each player either plays a match against another player or receives a bye. A match consists of two or three games, where each game may be a draw or won by one of the players. The third game is not played if and only if either player manages to win both the first and the second game. The player who wins the higher number of games wins the match. If both players win the same number of games, then the match is a draw. A player's rank is determined by four parameters: Match Points (MP), Opponents' match-win percentage (OMW), Game-win percentage (GW) and Opponents' game-win percentage (OGW). You don't need to know how to determine a player's rank since it is not needed for this problem.

Match Points: Players earn 3 match points for each match win, 0 points for each match loss and 1 match point for each match ending in a draw.

Game Points: Game points are similar to match points in that players earn 3 game points for each game they win and 1 point for each game that ends in a draw, and 0 points for any game lost.

Byes: When a player is assigned a bye for a round, they are considered to have won the match 2-0-0 (match results are recorded in this format: Wins–Losses–Draws). Thus, that player earns 3 match points and 6 game points.

Match-win percentage: A player's Match-win percentage is that player's accumulated match points divided by the total match points possible in those rounds (which is 3 times the number of rounds). If this number is lower than 1/3, use 1/3 instead. This limits the effect low performances have when calculating and comparing Opponents' match-win percentage.

Game-win percentage: Similar to the Match-win percentage, a player's Game-win percentage is the total number of game points they earned divided by the total game points possible (3 times the number of games played). Again, use 1/3 if the actual game-win percentage is lower than that. For example, if a player receives a bye in round 1 (which is equivalent to winning 2-0-0), and wins 2-0-1 in round 2, then draws 1-1-1 in round 3, then their Match-win percentage is $(3+3+1)/(3\times 3) = 7/9$, and their Game-win percentage is $(6+7+4)/(6+9+9) = 17/24$.

Opponents' match-win percentage: A player's Opponents' match-win percentage is the average Match-win percentage of each opponent that player faced (ignoring those rounds for which the player received a bye). Use the definition listed above when calculating each opponent's Match-win percentage, and always use each opponents' current Match-win percentage (when calculating the OMW after a round, use each opponents' Match-win percentage after that round). If a player plays against the same opponent multiple times during the tournament, then that opponent's Match-win percentage is used that many times when calculating the average. The same is true when calculating the OGW, which is defined below.

Opponents' game-win percentage: Similar to Opponents' match-win percentage, a player's Opponents' game-win percentage is simply the average game-win percentage of all of that player's opponents (again, byes are ignored). If a player has not played any matches (they receive byes for all the rounds), then both the OMW and the OGW of that player are considered to be 1/3.

Because you are a famous programmer, the tournament organiser asked you to write a program to help him run the tournament. You are given the results of each match of each round of this tournament, and you need to calculate the MP, OMW, GW and OGW of each player after each round so that the tournament organiser will be able to determine the rankings of the players much more quickly.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \le T \le 10^5$), the number of cases.

For each case, the first line of the input contains two integers n ($2 \le n \le 10\,000$) and m ($1 \le m \le 16$), the number of players and the number of rounds respectively. The following line contains m integers $a_1, a_2, \ldots, a_m$ where the i-th integer denotes the number of matches in the i-th round ($1 \le i \le m,\ 0 \le a_i \le \lfloor n/2 \rfloor$).

The following $a_1 + a_2 + \cdots + a_m$ lines each describes a match. The first a_1 lines describe matches in the first round, the next a_2 lines describe matches in the second round, and so on. Each match is described by five integers p_1, p_2, w_1, w_2, d ($1 \le p_1, p_2 \le n, p_1 \ne p_2, 0 \le w_1, w_2 \le 2, 0 \le d \le 3$), where p_1 and p_2 denote the IDs of the two players in the match, and w_1 and w_2 denote the number of games won by the first player and the second player, respectively. Finally, d denotes the number of games that ended in a draw. It is guaranteed that w_1, w_2, d represents a legal match result defined in the tournament structure and that each player appears in no more than one match in any single round. If a player doesn't appear in any matches for a round, then that player is considered to have been assigned a bye for that round.

It is also guaranteed that the sum of $n \cdot m$ over all cases doesn't exceed $3 \cdot 10^5$.

Output

The output for each case should contain $(n+1) \cdot m$ lines, where the first $n+1$ lines describe the results after the first round, the next $n+1$ lines describe the results after the second round, and so on.

For each round, print the string "`Round X`" (without quotes) in the first line, where `X` is the number of the round. Then print n lines, where the i-th line should contain four space-separated numbers, the MP, OMW, GW and OGW of the i-th player after this round. The MP should be represented by an integer, while the OMW, GW and OGW should be represented by a simple irreducible fraction in the form of P/Q, where P, Q are positive integers and the greatest common divisor of P and Q is 1. See the sample output for details.

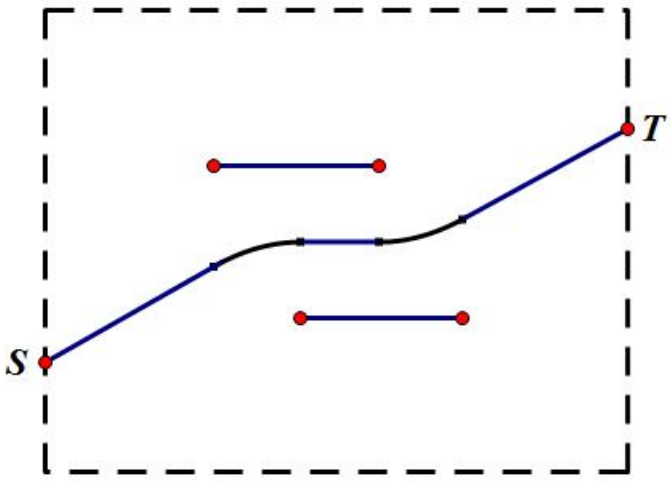
Example

standard input	standard output
2 2 3 0 1 1 1 2 2 0 1 1 2 1 1 1 3 2 1 1 1 2 0 2 0 2 3 2 0 0	Round 1 3 1/3 1/1 1/3 3 1/3 1/1 1/3 Round 2 6 1/2 13/15 7/15 3 1/1 7/15 13/15 Round 3 7 4/9 17/24 11/24 4 7/9 11/24 17/24 Round 1 0 1/1 1/3 1/1 3 1/3 1/1 1/3 3 1/3 1/1 1/3 Round 2 3 1/1 1/2 1/1 6 1/2 1/1 1/2 3 1/1 1/2 1/1

Problem D. Driverless Car

A driverless car company has a rectangular experiment field on a 2D Cartesian plane. Its four sides are all parallel to the coordinate axes. The bottom-left corner of the field is at coordinate (x_l, y_l) while the top-right corner is at coordinate (x_r, y_r).

There are two segments A and B lying strictly inside the rectangle. A and B share no common points. There is also a car on the plane, which can be regarded as a point. It will start at point S on the border of the rectangle, move to somewhere strictly inside the rectangle, and finally stop at another point T on the border of the rectangle. The experiment requires that the distance between the car and the segment A must be equal to the distance between the car and the segment B at all times during the movement (including when the car is at S and T). Also, S and T must be two different points. The distance between a point P and a segment Q is the minimum Euclidean distance from P to any point on Q.



The figure above corresponds to the sample test

Please write a program to help the company find a valid path of movement of the car, such that the length of the path is minimized, or determine that no valid paths exist.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \le T \le 10^5$), the number of cases.

For each case, the first line of the input contains four integers x_l, y_l, x_r, y_r ($-1\,000 \le x_l < x_r \le 1\,000$, $-1\,000 \le y_l < y_r \le 1\,000$), denoting the coordinates of the bottom-left and the top-right corners of the rectangle. Each of the next two lines contains four integers x_1, y_1, x_2, y_2, denoting a segment that connects (x_1, y_1) and (x_2, y_2), where $x_1, x_2 \in [x_l + 1, x_r - 1]$ and $y_1, y_2 \in [y_l + 1, y_r - 1]$.

For each case, it is guaranteed that the two endpoints of each segment do not coincide, and these two segments share no common points.

Output

For each case, if a valid path exists, print a single line containing a single real number, the minimum length of the path. Otherwise, print a single number 0 instead. Your answer will be considered correct if the absolute or relative error does not exceed 10^{-9}.

Formally, if your answer is a and the jury's answer is b, then your answer will be considered correct if and only if $\frac{|a-b|}{\max\{1,|b|\}} \le 10^{-9}$.

Example

standard input	standard output
1 0 0 7 6 2 4 4 4 3 2 5 2	7.552593593868681136

Problem E. Exchanging Gifts

After the dress rehearsal of CCPC Harbin Site 2019, m contestants are still in the contest arena. They are taking photos, discussing the problems, and exchanging gifts.

Initially, everyone has exactly one gift in their hand. Note that some contestants may have the same type of gifts. Specifically, the type of the gift in the i-th contestant's hand can be represented as a positive integer g_i. Two contestants i and j ($1 \le i, j \le m$) share the same type of gifts if and only if $g_i = g_j$ holds.

There can be many rounds of gift exchanging between these m contestants. In each round, two contestants may exchange their gifts with each other. Note that a pair of contestants can exchange gifts multiple times if they like. In the end, there will still be exactly one gift in each contestant's hand.

Let's denote h_i as the type of gift in the i-th contestant's hand in the end. If $g_i \neq h_i$ holds, the i-th contestant will be happy, because they have a different type of gift, otherwise they will be unhappy. Your task is to write a program to help them exchange gifts such that the number of happy contestants is maximized. For example, if $g = [3, 3, 2, 1, 3]$ and $h = [1, 2, 3, 3, 3]$, there will be 4 happy contestants.

Since m can be extremely large, you will be given n sequences $s_1, s_2, \ldots, s_n$, and the sequence g is equal to s_n. The i-th ($1 \le i \le n$) sequence will be given in one of the following two formats:

- "`1 k q[1..k]`" ($1 \le k \le 10^6$, $1 \le q_i \le 10^9$): It means $s_i = [q_1, q_2, \ldots, q_k]$.
- "`2 x y`" ($1 \le x, y \le i - 1$): It means $s_i = s_x + s_y$. Here "+" denotes concatenation of sequences, for example $[3, 3, 2] + [2, 2, 3, 3] = [3, 3, 2, 2, 2, 3, 3]$.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \le T \le 10\,000$), the number of cases.

For each case, the first line of the input contains a single integer n ($1 \le n \le 10^6$), denoting the number of sequences. Each of the next n lines describes a sequence in one of the two formats defined in the problem statement, where the i-th ($1 \le i \le n$) line describes the sequence s_i.

It is guaranteed that the sum of all n over all cases does not exceed 10^6, and the sum of k over all cases does not exceed 10^6. It is also guaranteed that no sequence has a length that exceeds 10^{18}.

Output

For each case, print a single line containing a single integer denoting the maximum number of happy contestants.

Example

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

Problem F. Fixing Banners

Harbin, whose name was originally a Manchu word meaning "a place for drying fishing nets", grew from a small rural settlement on the Songhua River to become one of the largest cities in Northeast China. Founded in 1898 with the coming of the Chinese Eastern Railway, the city first prospered as a region inhabited by an overwhelming majority of the immigrants from the Russian Empire. Now, Harbin is the capital of Heilongjiang province and the largest city in the northeastern region of the People's Republic of China. It serves as a key political, economic, scientific, cultural, and communications hub in Northeast China, as well as an important industrial base of the nation.

This year, a CCPC regional contest is going to be held in this wonderful city, hosted by Northeast Forestry University. To ensure the contest will be a success and enjoyed by programmers around the country, preparations for the event are well underway months before the contest.

You are the leader of a student volunteer group in charge of making banners to decorate the campus during the event. Unfortunately, your group made a mistake and misprinted one of the banners. To be precise, the word "harbin" is missing in that banner. Because you don't have time to reprint it, the only way to fix it is to cut letters from some used old banners and paste them onto the misprinted banner. You have exactly six banners, and for some reason, you must cut exactly one letter from each banner. Then, you can arrange and paste the six letters onto the misprinted banner and try to make the missing word "harbin". However, before you start cutting, you decide to write a program to see if this is possible at all.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \le T \le 50\,000$), the number of cases.

For each case, the input contains six lines. Each line contains a non-empty string consisting only of lowercase English letters, describing the letters on one of the old banners.

The total length of all strings in all cases doesn't exceed $2 \cdot 10^6$.

Output

For each case, print the string "`Yes`" (without quotes) if it is possible to make the word "harbin", otherwise print the string "`No`" (without quotes).

Example

standard input	standard output
2 welcome toparticipate inthe ccpccontest inharbin inoctober harvest belong ninja reset amazing intriguing	No Yes

Problem G. Game Store

Alice and Bob are playing a pretty old Nim game now. In this game, there are several piles of stones, where each pile may contain multiple stones. Two players take turns to remove stones. The player who can't take any legal moves first loses. In each move, the player can take one of the two possible moves below:

- Select a pile of stones, then remove x stones from it, where $x \geq 1$.
- Select two piles of stones, then remove x stones from one pile, and remove y stones from the other pile, where $x, y \geq 1$. Note that x and y can be different.

Alice always takes moves first. Bob thinks it is unfair to him. To make the game look more "fair", Bob can choose any subset of piles and remove those piles before the game starts. Note that Bob can choose to not remove any piles, but he can't remove all the piles, because obviously it will cause Alice to lose.

Both Alice and Bob are masters in this game, so they always play optimally. They will play n games in the next n days, one game per day. There is a game store renting piles of stones. Initially, the store is empty. On the i-th day, the store will put a new set of stones on rent, which costs b_i dollars per day and contains two same piles of stones, where each pile contains exactly a_i stones. A set must be rented in its entirety, Alice can't choose to rent only one pile of stones from a set. On each day, Alice will rent some sets from the store, take all the rented sets back home, then play the game with Bob with all those sets, and finally return all the sets to the store (they will be available for rent again the next day). The player who loses pays for the cost of renting on that day.

You are a master in this game too. You notice that if Alice chooses piles optimally, she will never lose! Alice also discovers this fact, so she wants to make Bob pay as many dollars as possible. However, the number of sets in the store may be extremely large. Please write a program to help Alice rent sets optimally such that Alice can always win the game on each day, and Bob will pay as much as possible.

Input

The input contains only a single case.

The first line of the input contains a single integer n ($1 \leq n \leq 500\,000$), denoting the number of days. Each of the next n lines contains two integers x_i and y_i ($1 \leq i \leq n, 1 \leq x_i, y_i \leq 10^{18}$), describing the new set put on rent on the i-th day. The input is encrypted in order to force online processing. The actual a_i and b_i are $x_i \oplus last$ and $y_i \oplus last$, where $last$ is the answer of the previous day and "$\oplus$" denotes bitwise exclusive-or. Note that for the first day, $last = 0$. It is guaranteed that $1 \leq a_i \leq 10^{18}$ and $1 \leq b_i \leq 10^9$.

Output

Print n lines, where the k-th ($1 \leq k \leq n$) line contains a single integer, the maximum number of dollars that Bob needs to pay on the k-th day.

Example

standard input	standard output
3 1 4 6 7 4 13	4 7 14

Note

In the sample test, $a_1 = 1, b_1 = 4$; $a_2 = 2, b_2 = 3$ and $a_3 = 3, b_3 = 10$.

Problem H. Highway Buses

There are n bus stations in Harbin, connected by m bidirectional highways. It is always possible to reach any bus station using these highways wherever you are. These bus stations are labeled by $1, 2, \ldots, n$.

If you want to take a bus from the i-th station to the j-th ($1 \le i, j \le n$) station, you should first buy such a ticket for your trip and then enter the corresponding bus. The driver will always choose the shortest route to your destination, which is the route consisting of the minimum number of highways. But if your destination is too far away, you will not be able to buy such a ticket. Specifically, you can buy a ticket from the i-th station to the j-th station if and only if you are at the i-th station and $dis(i, j) \le f_i$, where $dis(i, j)$ denotes the number of highways on the shortest route between i and j.

Alice is at the first station, and she wishes to visit her best friend Bob, who is at the k-th ($1 \le k \le n$) station. She needs to take several buses to reach the k-th station **in one day**. Initially, buying a ticket at the i-th station will cost Alice c_i dollars. Buying a ticket on the second day will cost her $c_i + w_i$ dollars. And if she wants to buy a ticket on the third day, she'll need to pay $c_i + 2w_i$ dollars, and so on. In short, the cost changes by w_i every day.

To save money, Alice needs to select the best date T, where $1 \le T \le T_{\max}$, and take several buses all on that day to reach the k-th station, such that the total cost for the tickets is minimized. Note that initially $T = 1$, so she needs to pay $c_i + (T - 1)w_i$ dollars on the T-th day.

Bob hasn't told Alice where he is yet. Please write a program to help Alice find the cheapest way for all possible destinations $k = 1, 2, \ldots, n$.

Input

The input contains only a single case.

The first line of the input contains three integers n, m and $T_{\max}$ ($1 \le n \le 200\,000$, $n - 1 \le m \le n + 50$, $1 \le T_{\max} \le 10^6$), denoting the number of bus stations, the number of highways and the parameter $T_{\max}$.

Each of the next n lines contains three integers f_i, c_i and w_i ($1 \le i \le n, 1 \le f_i \le n$, $1 \le c_i \le 10^9$, $|w_i| \le 10^9$), denoting the parameters of the i-th bus station. It is guaranteed that the cost for each ticket will never be negative and will never exceed $2 \cdot 10^9$.

Each of the next m lines contains two integers u_i and v_i ($1 \le i \le m, 1 \le u_i, v_i \le n$, $u_i \ne v_i$), denoting a bidirectional highway connecting the u_i-th station and the v_i-th station. It is guaranteed that it is always possible to reach any bus station using these highways wherever you are. Note that there may be multiple highways between the same pair of bus stations.

Output

Print n lines, where the k-th ($1 \le k \le n$) line contains a single integer, the minimum number of dollars that Alice needs to pay if Bob is at the k-th station.

Example

standard input	standard output
6 6 2 1 50 -40 1 2 100 2 1 100 2 4 100 3 1 100 1 1 100 1 2 2 3 3 4 4 2 2 5 6 1	0 10 52 52 52 10

Note

In the sample test:

- When $k = 1$, the answer is obviously 0.
- When $k = 2$, the best date T is 2.
- When $k = 3$, the best date T is 1.
- When $k = 4$, the best date T is 1.
- When $k = 5$, the best date T is 1.
- When $k = 6$, the best date T is 2.

Problem I. Interesting Permutation

DreamGrid has an interesting permutation of $1, 2, \ldots, n$ denoted by $a_1, a_2, \ldots, a_n$. He generates three sequences f, g and h, all of length n, according to the permutation a in the way described below:

- For each $1 \le i \le n$, $f_i = \max\{a_1, a_2, \ldots, a_i\}$;
- For each $1 \le i \le n$, $g_i = \min\{a_1, a_2, \ldots, a_i\}$;
- For each $1 \le i \le n$, $h_i = f_i - g_i$.

BaoBao has just found the sequence h DreamGrid generates and decides to restore the original permutation. Given the sequence h, please help BaoBao calculate the number of different permutations that can generate the sequence h. As the answer may be quite large, print the answer modulo $10^9 + 7$.

Input

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

For each case, the first line of the input contains a single integer n ($1 \le n \le 10^5$), the length of the permutation as well as the sequences. The second line contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le i \le n, 0 \le h_i \le 10^9$).

It's guaranteed that the sum of n over all cases does not exceed $2 \cdot 10^6$.

Output

For each case, print a single line containing a single integer, the number of different permutations that can generate the given sequence h. Don't forget to print the answer modulo $10^9 + 7$.

Example

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

Note

For the first sample case, permutations $\{1, 3, 2\}$ and $\{3, 1, 2\}$ can both generate the given sequence.

For the second sample case, permutations $\{1, 2, 3\}$, $\{2, 1, 3\}$, $\{2, 3, 1\}$ and $\{3, 2, 1\}$ can generate the given sequence.

Problem J. Justifying the Conjecture

The great mathematician DreamGrid proposes a conjecture, which states that:

- Every positive integer can be expressed as the sum of a prime number and a composite number.

DreamGrid can't justify his conjecture, so you are invited to write a program to verify it. Given a positive integer n, find a prime number x and a composite number y such that $x + y = n$.

A prime number is a natural number greater than 1 that cannot be formed by multiplying two smaller natural numbers. A natural number greater than 1 that is not prime is called a composite number. Note that 1 is neither a prime number nor a composite number.

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \le T \le 10^5$), the number of cases.

For each case, the only line of the input contains a single integer n ($1 \le n \le 10^9$).

Output

For each case, print two integers x and y in a single line, where $1 \le x, y < n$. If there are multiple valid answers, you may print any of them. If there is no valid answer, print the integer -1 instead.

Example

standard input	standard output
3 4 6 7	-1 2 4 3 4

Problem K. Keeping Rabbits

DreamGrid is the keeper of n rabbits. Initially, the i-th ($1 \le i \le n$) rabbit has a weight of w_i.

Every morning, DreamGrid gives the rabbits a carrot of weight 1 and the rabbits fight for the only carrot. Only one rabbit wins the fight and eats the carrot. After that, the winner's weight increases by 1. The whole process of fighting and eating ends before the next morning.

DreamGrid finds that the heavier a rabbit is, the easier it is to win a fight. Formally, if the weights of the rabbits are $w'_1, w'_2, \ldots, w'_n$ before a fight, the probability that the i-th rabbit wins the fight is

$$\frac{w'_i}{\sum_{j=1}^{n} w'_j}$$

He wants to know the expected weight of every rabbit after k days (k carrots are given and eaten).

Input

The input contains multiple cases. The first line of the input contains a single integer T ($1 \le T \le 10^5$), the number of cases.

For each case, the first line of the input contains two integers n and k ($1 \le n \le 10^5, 1 \le k \le 10^9$). The second line contains n integers $w_1, w_2, \ldots, w_n$ ($1 \le i \le n, 1 \le w_i \le 10^9$).

It's guaranteed that the sum of n over all cases doesn't exceed 10^6.

Output

For each case, print a single line containing n space-separated real numbers, where the i-th ($1 \le i \le n$) number should be equal to the expected weight of the i-th rabbit after k days.

Your answer will be considered correct if the absolute or relative error does not exceed 10^{-4}.

Example

standard input	standard output
3 1 1 2 2 2 1 3 3 2 1 1 1	3.00000000 1.50000000 4.50000000 1.66666667 1.66666667 1.66666667

Problem L. LRU Algorithm

In computing, cache algorithms (also frequently called cache replacement algorithms or cache replacement policies) are optimizing instructions, or algorithms, that a computer program or a hardware-maintained structure can utilize to manage a cache of information stored on the computer. Caching improves performance by keeping recent or often-used data items in memory locations that are faster or computationally cheaper to access than normal memory stores. When the cache is full, the algorithm must choose which items to discard to make room for the new ones.

One of the most famous cache algorithms is called the *Least Recently Used* (LRU) algorithm. In LRU, items in the cache are typically maintained with a linked list. When an item is accessed, it is removed from the linked list (if present), and then inserted to the front of the list. The front of the linked list is the *Most Recently Used* item. Because used items are continually being moved to the front of the linked list, the least used ones naturally end up at the back of the list. When there's not enough room in the cache, the item at the back of the linked list is discarded.

In this problem, each item in the cache is given a unique positive integer identifier for convenience. For example, let's consider the access sequence {4, 3, 4, 2, 3, 1, 4} and assume that the capacity of the cache is 3. The following table shows the content of the linked list.

Step	**Content of the linked list**	**Operation**
1		Acess 4
2	4	Acess 3
3	$3 \to 4$	Acess 4 (already in cache)
4	$4 \to 3$	Acess 2
5	$2 \to 4 \to 3$	Acess 3 (already in cache)
6	$3 \to 2 \to 4$	Acess 1 (pop 4 from back)
7	$1 \to 3 \to 2$	Acess 4 (pop 2 from back)
8	$4 \to 1 \to 3$	

Now, you want to do some memory optimization for your program. To that end, you are going to run some experiments. Assume that the LRU algorithm is used to manage the cache. The access sequence of your program is known and fixed, and you want to run q experiments about the cache. In the i-th ($1 \le i \le q$) experiment, the capacity of the cache is set to m_i and you have a linked list x_i. You want to find out if at some point during the execution of your program, the linked list maintained by the LRU algorithm is exactly the same as x_i.

Just before you start running your program to experiment, one of your friends challenged you to find out the results without actually running the experiment q times. He asks you to write a simple program to find out the answer, can you do it?

Input

The input contains multiple cases. The first line of the input contains a single positive integer T, the number of cases.

For each case, the first line of the input contains two integers n, q ($1 \le n \le 5\,000, 1 \le q \le 2\,000$), the length of the access sequence and the number of experiments. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le i \le n, 1 \le a_i \le n$), where the i-th integer denotes the identifier of the item accessed in the i-th operation.

The following q lines each describes an experiment. The i-th ($1 \le i \le q$) line contains an integer m_i ($1 \le m_i \le n$), the capacity of the cache, followed by m_i integers $x_{i,1}, x_{i,2}, \ldots, x_{i,m_i}$ ($1 \le j \le m_i, 0 \le x_{i,j} \le n$), where the j-th integer denotes the identifier of the j-th item in the linked list x_i. Note that the number of items in x_i may be less than m_i. Let L_i be the length of x_i, then the last $m_i - L_i$ integers in the input will be equal to 0, those zeroes should be ignored, while the other integers will be positive.

It's guaranteed that the sum of n over all cases does not exceed 20 000, the sum of q over all cases does not exceed 20 000 and the sum of m_i over all cases does not exceed $2 \cdot 10^6$.

Output

For each case, print q lines, where the i-th line describes the result of the i-th experiment. If at some point, the linked list maintained by the LRU algorithm is equal to the given list x_i, print the string "`Yes`" (without quotes). Otherwise, print the string "`No`" (without quotes).

Example

standard input	standard output
1 7 5 4 3 4 2 3 1 4 1 4 2 2 3 3 3 2 1 4 4 1 3 2 4 3 4 0 0	Yes No No Yes Yes