Problem A. Gambling on Choosing Regionals

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

"Participate in large regionals if you want to win gold medals; participate in small regionals if you want to get the qualify to WF..."

There will be n teams participating in the 2024 ICPC Regional Contests numbered from 1 to n. Each team can be described by a positive integer represents its comprehensive strength and a string represents the abbreviation of the university it belongs to. In this problem, we guarantee that the abbreviation of each university is unique and the comprehensive strengths of the teams are pairwise distinct. We also assume that the stronger team will always be ranked ahead of the weaker team in any contest.

This year, k regional contests will be held. For the i-th regional contest, every university can select no more than c_i teams to join the contest. Meanwhile, each team can select no more than 2 regional contests to participate in.

Now, you are asked to calculate the smallest possible rank for each team in the worst case, if they choose the regionals optimally. Note that when a team registers, it does not know the registration strategy of other teams, even if they are from the same university (but the total number of teams in the same university for the i-th regional contest is still limited by the c_i, and we always assume that the team currently under consideration has priority to register).

Please refer to the samples to understand the problem better.

Input

The first line contains two integers n, k ($1 \le n, k \le 10^5$), representing the number of teams and the number of regional contests.

The second line contains k integers $c_1, \cdots, c_k$ ($1 \le c_k \le 10^9$). Where c_i indicates the limit on the number of teams from every university in the i-th regional contest.

For the following n lines, the i-th line indicates the information of the team i. the i-th line contains an integer w_i ($1 \le w_i \le 10^9$) and a string S_i ($1 \le |S_i| \le 10$) consisting of uppercase letters, representing the comprehensive strength and the abbreviation of the corresponding university of the team i.

Output

Output n lines.

For the i-th line, print a single integer representing the smallest possible rank for the team i in the worst case, if they choose the regionals optimally.

Examples

standard input	standard output
5 3 1 2 3 100 THU 110 PKU 95 PKU 105 THU 115 PKU	2 1 2 2 1
5 2 2 3 100 THU 110 PKU 95 PKU 105 THU 115 PKU	4 2 4 3 1

Problem B. Mountain Booking

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

Byte Mountain is a beautiful tourist attraction in Byteland. There are n areas on the Byte Mountain, labeled by $1, 2, \ldots, n$, connected by $n-1$ undirected roads. There is exactly one path between each pair of different areas. In other words, the map of the mountain is a tree with n nodes.

Assume today is day 0. In each of the next m days, the map of the mountain will be changed a little. Formally, in the early morning of day i ($1 \le i \le m$), the k_i-th road will be closed forever, and another new road will be built, but the map of the mountain will always be a tree.

You are given the road construction plans for all the next m days, together with bookings of p tourists. The i-th tourist will visit the b_i-th area in the afternoon of day a_i, and will leave the mountain that night. You are now wondering about some interesting metrics.

Note that each road is weighted. Let $f(u, v)$ be the maximum value of weights among all the roads on the unique single path from u to v ($u \neq v$). You will be given q queries. In the i-th query, you will be given two integers c_i and d_i. You are required to calculate the following metric of the d_i-th area on the c_i-th day:

$$\sum_{1 \le j \le p,\, a_j = c_i,\, b_j \neq d_i} f(b_j, d_i)$$

Input

The first line of the input contains four integers n, m, p and q ($2 \le n \le 10^5, 1 \le m, p, q \le 2 \times 10^5$), denoting the number of areas, the number of incoming days, the number of tourists and the number of queries, respectively.

Each of the next $n-1$ lines contains three integers u_i, v_i and w_i ($1 \le u_i, v_i \le n$, $u_i \neq v_i$, $1 \le w_i \le 10^9$), denoting a two-way road between the u_i-th area and the v_i-th area, whose weight is w_i. The i-th road is labeled by i ($1 \le i < n$). These $n-1$ roads describe the map of the mountain at day 0.

Each of the next m lines contains four integers k_i, u_i, v_i and w_i ($1 \le k_i \le n-2+i$, $1 \le u_i, v_i \le n$, $u_i \neq v_i$, $1 \le w_i \le 10^9$), denoting that at day i the road k_i will be closed forever, and the new road labeled by $n-1+i$ will be built between the u_i-th area and the v_i-th area, whose weight is w_i. It is guaranteed that the map of the mountain will always be a tree, and no road will be closed more than once.

Each of the next p lines contains two integers a_i and b_i ($1 \le a_i \le m$, $1 \le b_i \le n$), describing a tourist's booking.

Each of the next q lines contains two integers c_i and d_i ($1 \le c_i \le m$, $1 \le d_i \le n$), describing a query.

Output

For each query, print a single line containing an integer, denoting the answer.

Example

standard input	standard output
5 3 6 6 1 2 9 1 3 4 1 4 6 4 5 2 3 3 5 3 2 1 5 5 5 3 4 8 1 3 2 1 3 3 2 4 1 5 2 4 1 1 1 3 2 5 2 3 3 1 3 3	8 3 9 11 8 0

Problem C. Prefix of Suffixes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	128 megabytes

Ms. Wozzie has a variable-length array S. Initially S is empty, and the array will be manipulated N times, each time adding a number to the end of S.

She wants to check the current state of the array when she adds the i-th number S_i, so she will give you a pair of A_i, B_i at the same time, and wants you to output the current value of $\sum_{i=1}^{n} \sum_{j=i}^{i+z_i-1} A_j \times B_i$. Define z_i as the longest common prefix of $[S_i, \ldots, S_n]$ and $[S_1, \ldots, S_n]$ where n is the current length.

Recall that the longest common prefix of $[a_1, \ldots, a_n]$ and $[b_1, \ldots, b_m]$ is the largest L satisfying: $a_i = b_i, \forall 1 \leq i \leq L$.

To make sure you answered her question in real time, she will encrypt the number S_i she is currently adding backward through your last answer, the exact decryption is given in the input format.

Input

The first line contains an integer N ($2 \leq N \leq 3 \times 10^5$) indicating the number of operations.

For the next N lines, the i-th line contains three integers S'_i, A_i, B_i ($0 \leq S'_i < N, 1 \leq A_i, B_i \leq 1000$) representing the encrypted S_i and A_i, B_i.

For decryption: the value of S_i is $(S'_i + lastans) \mod N$, where $lastans$ denotes the last output answer. In particular, initially, $lastans = 0$.

Output

Output N lines, with the ith line representing the answer after the i-th operations on S.

Example

standard input	standard output
3 0 1 2 1 2 3 2 3 4	2 12 18

Problem D. Query on Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

Given a rooted tree with n nodes, where the root is node $r = 1$, each node has an associated weight a_i.

The distance between two nodes in the tree is defined as the number of edges on the shortest path between them.

The subtree of a node x is defined as the set of all nodes y such that the shortest path from y to the root r passes through x.

You need to support the following three types of operations:

1. Given x, k, and v, for each node y at a distance exactly k from x, update a_y to $a_y + v$, then output the maximum of a_y.

2. Given x, k, and v, for each node y at a distance of at most k from x, update a_y to $a_y + v$, then output the maximum of a_y.

3. Given x and v, update a_y to $a_y + v$ for every node y in the subtree of x, then output the maximum of a_y.

Input

The first line contains an integer T ($1 \le T \le 10^4$), representing the number of test cases.

For each test case, the first line contains two integers n and q ($1 \le n, q \le 2 \times 10^5$), representing the number of nodes in the tree and the number of operations.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($-10^{14} \le a_i \le 10^{14}$), representing the weight of each node.

The next $n - 1$ lines each contain two integers x and y ($1 \le x, y \le n$, $x \ne y$), representing an edge in the tree.

The next q lines describe the operations. Each line contains three or four integers o, x, v or o, x, k, v ($o \in \{1, 2, 3\}$, $1 \le x \le n$, $0 \le k < 10$, $-10^9 \le v \le 10^9$), representing the operation type and its parameters.

It is guaranteed that the sum of n over all test cases does not exceed 5×10^5, and the sum of q over all test cases does not exceed 2×10^5.

Output

For each test case, output q lines, each containing an integer representing the result of the corresponding operation. If no valid node y exists, output `GG`.

Example

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

Note

The initial node weights are $a = [1, 2, 1, 3, 2]$.

After the 1st operation, the weights remain $a = [1, 2, 1, 3, 2]$.

After the 2nd operation, the weights become $a = [4, 2, 4, 6, 2]$.

After the 3rd operation, the weights become $a = [4, 2, 4, 1, -3]$.

After the 4th operation, the weights become $a = [4, 5, 4, 4, 0]$.

After the 5th operation, the weights become $a = [4, 4, 3, 3, -1]$.

Problem E. Escape

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Sneaker wakes up in a vast maze, and now he wants to escape from it.

Through the map found in every room of the maze, Sneaker learns the structure of the maze. The maze consists of n rooms, with Sneaker starting in room 1, and the exit is in room n. Additionally, there are m two-way passages in the maze, each connecting two different rooms, and the two directions of each passage are isolated from each other.

It is guaranteed that no two different passages connect the same pair of rooms, and there is no passage that connects a room to itself.

Moreover, it is guaranteed that any two rooms can be reached from each other through a series of passages.

Sneaker also discovers that there are k slayer robots in the maze. The i-th slayer robot initially stays in room s_i. Obviously, Sneaker does not want and cannot encounter these slayer robots.

If Sneaker starts moving, each slayer robot will move simultaneously. Specifically, Sneaker will choose a passage connected to the room he is currently in and move to the room on the other side of the passage. At the same time, each slayer robot will randomly choose a passage connected to the room it is in and move to the room on the other side of the passage. Every slayer robot will enter and exit the passage at the same time as Sneaker.

All slayer robots will record the passages they pass through. If a slayer robot travels through a passage that is the same as the last recorded passage, it can choose to erase both occurrences of this passage from its record. In other words, the slayer robot can choose to undo a record.

For example, if a slayer robot currently has a record of passages $(1, 2), (2, 7), (7, 3)$ and is now in room 3, it can choose to move through a new passage, say $(3, 6)$, and its record will become $(1, 2), (2, 7), (7, 3), (3, 6)$, with the robot ending up in room 6. Alternatively, it can move through passage $(3, 7)$ to return to room 7, and its record will become $(1, 2), (2, 7)$.

Furthermore, if the slayer robot then chooses to move through a passage back to room 2, its record will become $(1, 2)$. However, if it moves from room 3 directly to room 2, its record cannot become $(1, 2)$; instead, it will become $(1, 2), (2, 7), (7, 3), (3, 2)$.

Additionally, all slayer robots share a same self-destruct parameter d. If a slayer robot arrives in a room and finds that its record of passages exceeds d, it will immediately self-destruct. If Sneaker enters a room and finds that a slayer robot is in the process of self-destructing or has already self-destructed, it is not considered an encounter with the robot.

Now, Sneaker wants to plan a route with the minimum number of passages and ensure that he will not encounter any slayer robot. Can you help him?

Note: If Sneaker is moving from room x to room y through a passage, and a slayer robot is moving in the opposite direction, from room y to room x through the same passage, Sneaker will not encounter the slayer robot because the two directions of the passage are isolated from each other. The sneaker only knows the initial position of each slayer robot, but does not know their positions at every moment. If both a slayer robot and Sneaker arrive at room n at the same time, then Sneaker is considered to have encountered the slayer robot and cannot escape from the maze.

Input

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

For each test case, the first line contains three integers n, m, and d ($2 \le n \le 2 \times 10^5$, $n - 1 \le m \le \min\{5 \times 10^5, \frac{n(n-1)}{2}\}$, $1 \le d < n$), which represent the number of rooms in the maze,

the number of passages, and the self-destruct parameter, respectively.

The next m lines each contain two integers x_i and y_i ($1 \le x_i < y_i \le n$), indicating that the i-th passage connects room x_i and room y_i.

It is guaranteed that for any $i \neq j$, $x_i \neq x_j$ or $y_i \neq y_j$.

It is guaranteed that any two rooms can be reached from each other through a series of passages.

The $(m+2)$-th line contains several integers. The first integer k ($0 \le k \le n-2$) represents the number of slayer robots, followed by k integers $s_1, s_2, \ldots, s_k$ ($2 \le s_i < n$) which represent the initial room where each slayer robot is.

It is guaranteed that the sum of n over all test cases does not exceed 5×10^5, and the sum of m over all test cases does not exceed 2×10^6.

Output

For each test case, if there exists a valid route, output a single integer p in the first line, representing the minimum number of passages in the valid route.

In the second line, output $p+1$ integers $z_0 = 1, z_1, \ldots, z_p = n$, representing the sequence of rooms that the route passes through.

If no valid route exists, output a single integer -1.

Example

standard input	standard output
2 7 8 2 1 2 2 3 3 7 2 5 5 6 3 6 1 4 4 5 1 4 7 8 2 1 2 2 3 3 7 2 5 5 6 3 6 1 4 4 5 1 5	3 1 2 3 7 -1

Problem F. Tourist

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

On the Codeforces platform, each user has a rating to measure his level. In addition, each user will also have a title based on his rating. Recently, because user Tourist's rating exceeded 4000 points, Codeforces administrators decided to provide a new title, "**Tourist**", to commemorate this feat.

You also want to get this title, because it's cool to be called a "Tourist"! But your current rating is only 1500 points, which is not enough to get this honor. So, you let the GPT predict your rating changes in the next n contests. Specifically, suppose your rating is b points before the i-th contest, then after the i-th contest, the GPT predicts that your rating will become $b + c_i$ points. You want to know after which of those n contests you will get the title of "Tourist" for the **first time**, or there is no such contest.

Note that the rating may become negative during this process, and the "Tourist" title will be awarded to users with a rating of **at least** 4000 points.

Input

The first line contains a single integer n ($1 \le n \le 10^5$), representing the number of contests in the future.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($|c_i| \le 10^9$). Where c_i indicates the change in rating after the i-th contest.

Output

Output a single integer representing the index of the contest after which you will get the title of "Tourist" for the **first time**. If such a contest does not exist, output '-1'.

Examples

standard input	standard output
5 1000 1000 1000 -5000 1000	3
5 20 -100 10 -150 5	-1

Problem G. Game

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

Alice and Bob are playing a game. Initially, Alice has x chips, and Bob has y chips.

The game proceeds in several rounds. In each round, Alice wins with a probability of p_0, Bob wins with a probability of p_1, and there is a probability of $1 - p_0 - p_1$ for a draw.

If there is a draw, the game immediately moves to the next round. Otherwise, if the number of the winner's chips is not smaller than the number of the loser's chips, the winner wins the entire game, and the game ends; otherwise, the loser loses an amount of chips equal to the winner's current chips, and the game moves to the next round.

Note that after each round of the game, no one's chips will increase.

You are asked to find the probability that Alice will ultimately win the entire game.

Input

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

For each test case, the first line contains two integers x and y ($1 \le x, y \le 10^9$), representing the initial number of chips Alice and Bob have, respectively.

The second line contains three non-negative integers a_0, a_1, and b ($1 \le a_0 + a_1 \le b < 998244353$), representing $p_0 = \frac{a_0}{b}$, $p_1 = \frac{a_1}{b}$.

Output

For each test case, output one line containing an integer representing the probability that Alice will win the entire game, modulo 998244353.

Example

standard input	standard output
3 1 1 2 2 6 1 3 2 3 6 3 4 7 3 15	499122177 910398850 220911476

Note

For the first test case, since both players have the same number of chips and the same probability of winning a round, the probability of either Alice or Bob winning the entire game is $\frac{1}{2}$.

For the second test case, Alice must win three rounds before Bob wins a single round to win the entire game. If a round does not end in a draw, Alice wins with a probability of $\frac{2}{5}$, so Alice's final winning probability is $\left(\frac{2}{5}\right)^3 = \frac{8}{125}$.

Problem H. Points Selection

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

There are n points $p_1, p_2, \ldots, p_n$ on the 2D plane. The i-th point p_i is located at (x_i, y_i) with weight w_i.

Consider a query with parameters (a, b, c), let $S(a, b) = \{p_i | x_i \le a \wedge y_i \le b\}$. The query denotes "*Is it possible to select a subset T from S such that* $(\sum_{p_i \in T} w_i) \bmod n = c$*?*" $(T \subseteq S)$

Let query(a, b, c) = True/False be the answer to the query with parameters (a, b, c). You are required to answer all possible queries. Instead of printing $O(n^3)$ lines, you only need to compute the following value:

$$ans = (\sum_{a=1}^{n} \sum_{b=1}^{n} \sum_{c=1}^{n-1} a \cdot b \cdot c \cdot [\text{query}(a, b, c) = \text{True}]) \bmod 2^{64}$$

Input

The first line of the input contains a single integer n $(2 \le n \le 5 \times 10^5)$, denoting the number of points.

In the next n lines, the i-th line contains three integers x_i, y_i and w_i $(1 \le x_i, y_i, w_i \le n)$, describing the i-th point. Note that two points may share the same location.

It is guaranteed that all the values of x_i, y_i and w_i are chosen uniformly at random from integers in their corresponding ranges. The randomness condition does not apply to the sample test case, but your solution must pass the sample as well.

Output

Print a single line containing a single integer, denoting the value of ans.

Examples

standard input	standard output
3 1 2 2 2 3 1 3 1 3	75
5 1 1 4 5 1 4 2 3 3 1 4 1 2 5 2	1935

Problem I. Strange Binary

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A normal binary decomposition can be represented as $n = \overline{a_{31}a_{30}\cdots a_0}$, where $a_i \in \{0, 1\}$.

Now, there is a strange binary decomposition, still represented as $n = \overline{a_{31}a_{30}\cdots a_0}$, but it needs to satisfy:

1. For $i = 0, 1, \cdots, 31$, $a_i \in \{-1, 0, 1\}$.
2. For $i = 0, 1, \cdots, 30$, a_i and a_{i+1} cannot both be 0, i.e., $a_i^2 + a_{i+1}^2 \neq 0$.
3. $$\sum_{i=0}^{31} a_i 2^i = n$$

Given a non-negative integer n, find the above binary decomposition of n, or determine that it cannot be decomposed.

Input

The first line contains an integer T ($1 \leq T \leq 10^4$), representing the number of test cases.

For each test case, there is one line containing a non-negative integer n ($0 \leq n < 2^{30}$), representing the number to be decomposed.

Output

For each test case, output a string `YES` if n can be decomposed, or `NO` if n cannot be decomposed.

If the first line is `YES`, then output 32 integers $a_0, a_1, \ldots, a_{31}$, divided into 4 lines with 8 integers on each line.

Example

standard input	standard output
3 0 3 5	NO YES 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 YES -1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1

Problem J. Stacking of Goods

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	128 megabytes

You are a grocery store owner, and there are n goods in your store. Each good i has an associated weight w_i, initial volume v_i, and compression coefficient c_i.

You stack all the goods into one pile to keep the store tidy. Because goods can be compressed, assuming that the sum of the weights of the items above the goods i is W (Not including itself), then after stacking, the actual volume of the goods i will become $v_i - c_i \times W$.

The space in your store is really limited, so you want to know the minimum possible value of the sum of the actual volumes of the items.

Input

The first line contains a single integer n ($1 \le n \le 10^5$), representing the number of goods.

For the following n lines, each line contains three integers w_i, v_i, c_i ($1 \le w_i \le 10^5, 1 \le v_i \le 10^{12}, 0 \le c_i < \frac{v_i}{\sum w_i}$), representing the weight, volume, and compression coefficient of the i-th good.

It's guaranteed that the actual volume of each good will never be compressed into a negative number or zero.

Output

A single integer represents the answer.

Example

standard input	standard output
3 1 8 1 2 9 2 3 10 2	16

Note

A possible optimal solution is that the order of goods in the pile from bottom to top is $1, 2, 3$. The actual volume of good 1 is $8 - 1 \times (2 + 3) = 3$; the actual volume of good 2 is $9 - 2 \times 3 = 3$; and the actual volume of good 3 is 10 since there are no other goods above it.

Problem K. Match

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

Given two sequences a and b of length n, a bipartite graph is generated in the following manner:

The nodes of the bipartite graph are divided into two parts: the left side and the right side. There are n nodes on both sides. An edge exists between the i-th node on the left side and the j-th node on the right side if and only if $a_i \oplus b_j \geq k$, where $\oplus$ denotes the bitwise XOR operator.

For each x in the range $[1, n]$, output the number of matchings of size x in the bipartite graph, with the result modulo 998244353.

Input

The first line contains two non-negative integers $n, k, (1 \leq n \leq 200, 0 \leq k < 2^{60})$.

The second line contains n non-negative integers, representing the sequence $a, (0 \leq a_i < 2^{60})$.

The third line contains n non-negative integers, representing the sequence $b, (0 \leq b_i < 2^{60})$.

Output

Output n non-negative integers, where the i-th integer represents the number of matchings of size i in the graph, with the result taken modulo 998244353.

Example

standard input	standard output
9 5 1 7 2 8 4 9 2 5 10 1 3 2 4 5 8 8 8 9	51 1034 10768 62195 200965 348924 294444 97344 7200

Problem L. 502 Bad Gateway

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Welcome to the Internet Connection Practice Competition (ICPC) this year! In the competition, you and your teammates will fight against the Obstacle Maker System (OMS) to connect to the server successfully! Be careful with every refresh operation, as the system may assign you unknown obstacles.

To simplify the problem, we model the competition process into the following form:

Each team has a timer during the competition, and the goal is to get the remaining time to 0 as quickly as possible. At the beginning of the 0-th second, the OMS will uniformly and randomly generate an integer t_0 in $[1, T]$ and initialize the remaining time of your timer to t_0 seconds. Next, at the end of each second (starting with the 0-th second), the following events occur in order:

- The remaining time of the timer will be decreased by 1. If the remaining time of the timer is zero at this time, the number of seconds from the start of the competition to this moment is your penalty.
- Otherwise, you can do nothing after this or use the refresh operation on OMS once. If you choose to refresh your timer, OMS will uniformly and randomly generate a new integer t' in $[1, T]$ and set the remaining time of your timer to t'.

Your goal is to minimize your penalty. Please calculate the expected penalty using the optimal strategy. During the contest, you always know the remaining time of the timer, and the value of T.

Input

The first line contains a single integer n ($1 \le n \le 10^6$), representing the number of test cases.

For the following n lines, each line contains a single integer T_i ($1 \le T_i \le 10^9$), representing the interval for generating random numbers is $[1, T_i]$ for the i-th test case.

Output

For each test case, output a single line with two positive integers x_i, y_i where $\gcd(x_i, y_i) = 1$, representing the expected penalty using the optimal strategy is x_i/y_i. It can be proved that the answer can always be described as a fraction.

Example

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