Problem A. Sort and &

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

You are given a permutation of size N. You want to sort it. In one operation you can swap the i^{th} and the j^{th} elements ($1 \leq i, j \leq N$) of the array and it will cost you **i&j** (bitwise **AND** of i and j). Total sum of sorting the permutation is the sum of the cost of all the operations you will make. You can't make more than $3N$ operations. Print the minimum cost to sort the permutation and also print one sequence of swap operations which will sort the permutation in the minimum cost.

Input

The first line will contain a single integer $T\,(1 \leq T \leq 10^4)$. Each test case will start with one integer $N\,(1 \leq N \leq 10^4)$, the length of the permutation. The next line will contain N integers $a_1, a_2, \ldots, a_N\,(1 \leq a_i \leq N)$.

It is guaranteed that the sum of N is at most 10^5 across all the test cases.

Output

For each test case, the first line of the output should contain the minimum cost to sort the array. The next line should contain the number of required operations. Then each next line should contain two integers describing the indices being swapped. Please see the sample for details.

Example

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

Note

Permutation of size N is an array of size N where each number from 1 to N appears exactly once.

Problem B. Gona Guni

Input file:	standard input
Output file:	standard output
Time limit:	5 seconds
Memory limit:	1024 megabytes

You are given a tree of N nodes. You can pick a non-empty set of nodes S from the tree. Create a minimal size connected subgraph G from the tree such that it covers all nodes in the set.

Lets define, $cost(S) = size\ of\ minimum\ vertex\ cover\ of\ G$.

Note that, minimum vertex cover of a graph is a set of vertices that includes at least one end point of every edge of the graph and size of the set is as minimum as possible.

You need to find the sum of $cost(S)^M$ for all possible sets S modulo 998244353.

Notes: Minimum vertex cover of a single node graph is 0.

Input

First line will contain a single integer $T(1 \leq T \leq 3000)$ representing the test cases. For each test case, there will be a single line containing two integers $N(1 \leq N \leq 3 \cdot 10^5)$ and $M(0 \leq M \leq 200)$ separated by space. After that N - 1 lines follow. In each line there will be two space separated integers U and V denoting an edge between node U and V $(1 \leq U, V \leq N)$. The sum of N over all test cases is at most $3 \cdot 10^5$.

Output

For each test case print a single line representing the answer modulo 998244353.

Example

standard input	standard output
2 3 1 1 2 1 3 20 200 1 2 1 3 2 4 1 5 5 6 1 7 6 8 6 9 3 10 4 11 6 12 11 13 4 14 13 15 15 16 6 17 13 18 15 19 13 20	4 286430678

Problem C. Packet Transmission

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

You are given a connected network **tree** representing routers and channels. Each node represents a router, and each edge represents a channel connecting two routers. These channels are used to transmit packets from one router to one of its adjacent routers, and the transmission follows the TCP/IP protocol. The edges are labeled with t_i indicating the time (in arbitrary units) it takes to transmit a packet from u_i to v_i or vice versa.

Additionally, each router has enough storage capacity to store any number of packets, they can store packets there indefinitely and they can transmit multiple packets simultaneously but via different channels. This means that you can't send a packet via a channel that is currently occupied by another transmission.

Your task is to send two packets concurrently, one from s_1 router to d_1 router and another from s_2 router to d_2 router. Determine the optimal transmission strategy to minimize the maximum time for delivering both of these packets from source to destination.

Input

Input starts with an integer $T(1 \leq T \leq 1000)$ denoting the number of test cases. Each of the test cases starts with two integers N and $Q(1 \leq N, Q \leq 100000)$. Then there will be $N-1$ lines of input consisting of three integers $u_i, v_i, t_i(1 \leq u_i, v_i \leq N, u_i \neq v_i, 1 \leq t_i \leq 10^9)$ representing the i^{th} edge of the tree.

Then, there will be Q lines each with four integers $s_1, d_1, s_2, d_2(1 \leq s_1, d_1, s_2, d_2 \leq N)$ representing the source and destination routers for two packets. It is guaranteed that the sum of N is at most 500000 and the sum of Q is at most 500000 across all test cases.

Output

For each query, please use the optimal transmission strategy to minimize the maximum time to deliver both packets and print that minimum time in a separate line.

Example

standard input	standard output
2 8 3 1 2 3 2 3 5 2 6 7 3 4 4 4 5 3 6 7 2 1 8 6 1 8 2 6 1 5 2 3 5 8 8 5 4 2 1 2 1 2 3 1 3 4 1 1 2 3 4 1 4 2 3	7 15 24 1 3

Problem D. Qwiksort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

You are given an array containing the integers from 1 to $2n$, each number appearing exactly once. Your goal is to sort the array in increasing order using a special operation called "Qwiksort". In one execution of Qwiksort, you can choose a contiguous range of size n and sort them in increasing order in-place. For example, if $n = 3$ and the array is $[3, 2, 4, 1, 6, 5]$, and you choose to apply Qwiksort to a contiguous range from the second to the fourth position, the array becomes $[3, 1, 2, 4, 6, 5]$.

You can apply the Qwiksort operation zero or more times to sort the array. But due to limitations of our server, you can only do it at most 10 times on a particular array. Please print the operations required to sort the array in increasing order. You do not need to minimize the number of operations. It is guaranteed that it's possible to sort the arrays with at most 10 Qwiksort operations.

Input

First line of input will contain a single integer T ($1 \leq T \leq 40{,}000$) denoting the number of independent test cases. Then the descriptions of T test cases follow. Each test case will start with a line containing an integer n ($2 \leq n \leq 1{,}000$). The next line will contain $2n$ space separated integers, the elements of the array to sort.

Sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, first print the number of Qwiksort operations k ($k \leq 10$) required to sort the array in a line. Then print k lines, each line containing two space separated integers l and r denoting 1-based index specifying the beginning and the end of the range that you applied Qwiksort to.

Note that $1 \leq l \leq r \leq 2n$ and $r - l + 1 = n$ must be ensured.

Example

standard input	standard output
2 5 1 2 3 4 5 10 9 8 7 6 2 1 2 3 4	7 1 5 3 7 5 9 6 10 1 5 3 7 1 5 6 1 2 2 3 3 4 1 2 2 3 1 2

Problem E. Travel on the Grid

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

You are given a grid of a size $N \times M$. You need to travel from cell $(1, 1)$ to (N, M). You can travel to any of the 8 adjacent cells from your position. There are mines in some cells. When any mine explodes, it can kill anyone standing on its cell or anyone standing on the 4 other adjacent cells of the mine (top, bottom, left, right). Stepping into any of these five cells will trigger the mine and it explodes.

You can build and use as many diffusers as you need to avoid the mines. Also, you can use a single diffuser as many times as you need. You need to place the diffuser directly on the cell where a mine is located to neutralize it. Once neutralized, the mine poses no further threat and it will **never explode in the future**. Building a diffuser and reusing a diffuser both have its own cost.

At any time, while in a safe cell (one without a non-diffused mine and not within the blast area of any adjacent non-diffused mines), you can build a diffuser and place that in any of the 8 adjacent cells from your current position.

In order to reuse a diffuser, you will need to travel to the cell where it is located. Then you can take the diffuser and travel with it. When you are traveling with a diffuser, you will not be able to build a new one or pick up another diffuser. At any time, you can deploy a carried diffuser in any of the 8 adjacent cells of your current position if that cell contains a mine. You can't drop a diffuser in a cell if it doesn't have any mines. Each time you reuse the diffuser, there is a fixed cost associated with it. Additionally, each cell on the grid has its own specific cost when you move from that cell to an adjacent one while carrying the diffuser. However, placing a diffuser in a neighboring cell doesn't incur the individual cell cost of carrying the diffuser. Therefore, the total cost of transporting and deploying the diffuser will be the sum of this fixed cost and the individual cell costs incurred during travel.

Now you need to find the cheapest way to reach the cell (N, M). It is guaranteed that no mine in the grid will attack cell $(1, 1)$. Also, **no single cell will be attacked by two different mines**.

Input

The first line will contain a single integer $T(1 \leq T)$. First line of each test case will contain four integers $N, M(2 \leq N, M \leq 1000, N \cdot M \leq 10^5), X(1 \leq X \leq 10^9)$ and $Y(1 \leq Y \leq 10^9)$. Here N and M represent the dimension of the grid $(N \times M)$ while X is the cost for building a diffuser and Y is the fixed cost for reusing a diffuser. The following N lines will contain a string, each of them will be of length M. Each character of the strings will be either '.' (ASCII code 46) or '#' (ASCII code 35). The character '.' denotes a safe cell and the character '#' denotes the cell has a mine on it. Following these, there will be N additional lines. Each line will contain M integers, $V_{ij}(0 \leq V_{ij} \leq 10^9)$ where each V_{ij} represents the associated cost of cell (i, j) for moving to an adjacent cell while carrying a diffuser.

It is guaranteed that the sum of $N \cdot M$ is at most 10^5 across all tests.

Output

For each test case, print the cheapest cost to reach (N, M) from $(1, 1)$ in a single line.

Example

standard input	standard output
2 3 4 10 5 .... .#.. ...# 0 0 0 0 0 1 2 3 0 0 0 0 3 4 10 10 ..#. .... #..# 0 0 1 0 0 0 0 0 1 0 0 0	16 20

Note

Explanation of the example

For the first test case, we start at cell $(1, 1)$ and perform the following steps.

- We build a diffuser (cost 10), place it on cell $(2, 2)$ and neutralize the mine.
- We move to cell $(2, 2)$ and pick up the diffuser (no cost).
- While carrying the diffuser we move to cell $(2, 3)$ (cost 1).
- We place the diffuser on cell $(3, 4)$ to reuse it (cost 5) and neutralize the mine.
- Finally we move to $(3, 4)$ and reach our destination.

Overall cost is $(10 + 1 + 5) = 16$.

For the second test case, we perform the following steps.

- We move to $(2, 2)$.
- We build a diffuser (cost 10), place it on cell $(1, 3)$ and neutralize the mine.
- We move to cell $(2, 3)$.
- We build a diffuser (cost 10), place it on cell $(3, 4)$ and neutralize the mine.
- Finally we move to $(3, 4)$ and reach our destination.

Overall cost is $(10 + 10) = 20$. If we wanted to reuse the diffuser, the cost would have been at least 21 as we would need to travel at least one cell carrying it.

Problem F. Yet Another Crossover Episode

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2.5 seconds
Memory limit:	512 megabytes

Pebae has an integer array $a_1, a_2, \ldots, a_n$. Please help her find the maximum of $\gcd(a_i \oplus a_j, a_i \& a_j)$ over all $1 \le i, j \le n$. Also, count the number of pairs that obtain this maximum value.

Here, $\gcd(x, y)$ is the greatest common divisor of x and y, $\gcd(x, 0) = x$ for $x \ge 0$, $\oplus$ is the bitwise XOR operator, and $\&$ is the bitwise AND operator.

Input

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

The first line of each test case contains an integer n ($1 \le n \le 2^{19}$). The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 16 \cdot n$).

It is guaranteed that the sum of n over all test cases does not exceed 2^{20}.

Output

For each test case, output two integers: the maximum value and the number of pairs that obtain this maximum value.

Example

standard input	standard output
4 4 1 9 1 9 1 0 3 0 0 1 7 12 2 3 0 110 1 69	9 4 0 1 1 5 111 2

Problem G. Picturesque Skyline

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

The city you live in has a lot of high-rise buildings. Due to some city regulations, each building has a unique height. One day you went to the mountains outside the city. After a long hike you looked at the city to appreciate the beauty of it all. There you noticed something weird, actually two very weird things. The first being, from the right spot all the tall buildings look like they are on a single line. The other is that the buildings are forming multiple unusual groups of pyramid-like patterns with a small gap separating two adjacent groups. Two adjacent buildings in the same group don't leave any space.

A group of $2k + 1$ buildings where k is a positive integer may form a pyramid-like pattern If they meet the following condition. Assuming the heights of buildings are $h_1, h_2, ..., h_{2k+1}$ when sorted by position of the building from left to right. Then the group of buildings need to satisfy $h_1 < h_2 < ... < h_k < h_{k+1} > h_{k+2} > ... > h_{2k} > h_{2k+1}$. So k building in increasing heights followed by the tallest building of the group which is then followed by k buildings in decreasing heights. The city is formed of several such pyramid-like patterns (with small gaps between adjacent groups). This makes the skyline of the city quite picturesque and unique. Assume there are no visible gaps between two adjacent buildings in a group.

Below are examples of two cities. The first one has a single pyramid-like pattern of 5 buildings. The second has 6 buildings in total with two pyramid-like patterns.

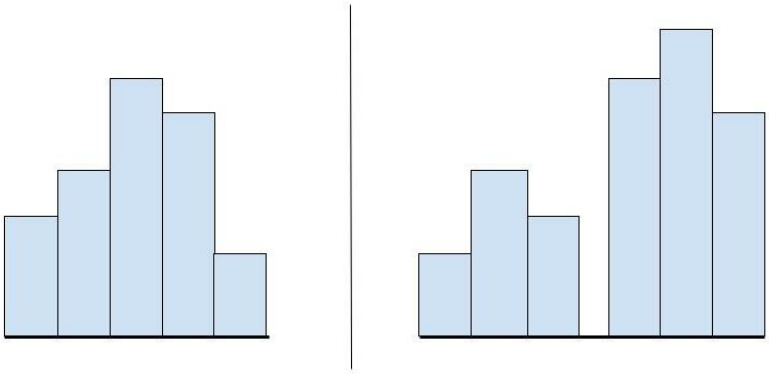


Fig: Example cities.

When you went to a different city as a tourist you noticed that the buildings in that city are also on a straight line without any gaps between adjacent buildings but they may not be grouped in a pyramid-like pattern. Oh the horror!! Now, hypothetically, let's assume you have two machines called Building Swapper 2000 and Group Maker 3000. Building Swapper 2000 can swap two adjacent buildings, but requires one unit of fuel to operate. Similarly Group Maker 3000 can split the buildings into one or more groups so that two adjacent groups have a little bit of space between them but it is solar powered, so it doesn't need any fuel. Now, you have to follow through two steps in order.

— Use the Group Maker 3000 to split the buildings into groups of odd sizes greater than 2. After you have applied this step then there will be one or more groups of buildings (but may not be pyramid-like yet) with a small gap between two adjacent groups.

— Use the Building Swapper 2000 to swap adjacent buildings until each group becomes pyramid-like. You can only swap two adjacent buildings if they are part of the same group.

Finally, you will have some groups of buildings, where all of them are in a pyramid-like pattern and the city will have a picturesque skyline. The question for you is to calculate the minimum cost of fuel required to achieve the goal.

Input

The first line of the input contains an integer T denoting the number of test scenarios. The first line of each test contains a single integer N ($3 \leq N \leq 1000, N \neq 4$), the number of buildings in the city. The next line contains N integers denoting the heights of the building ($1 \leq h_i \leq 10^9$). It is guaranteed that sum of N is at most 10000 over all tests and within each test all building heights are unique.

Output

For each test case print one integer, the minimum amount of fuel required. It can be proven that under the restriction it is always possible to arrange the buildings to achieve the goal.

Example

standard input	standard output
4 3 1 3 2 6 1 2 4 8 16 32 5 1 2 3 4 5 7 6 4 2 1 3 5 7	0 2 3 9

Problem H. Are the nodes reachable?

Input file:	standard input
Output file:	standard output
Time limit:	6 seconds
Memory limit:	1024 megabytes

You are given a directed acyclic graph consisting of N nodes, numbered from 1 to N, and M directed edges. Your task is to answer Q queries about reachability between nodes. For each query, you will be given two nodes, U and V.

- If node V is reachable from node U through existing roads (directed edges), you should output 0 as no additional road is required.
- If V is not reachable from U, you are allowed to build one temporary edge between any two nodes in the graph to make V reachable from U.

After the query is answered, this new road (if any) will be destroyed.

You must determine the minimum cost of constructing this road, if needed. The cost of building a road between node X and node Y is defined as $|X - Y|$, i.e., the absolute difference between their node labels.

Input

The first line will contain a single integer $T(1 \leq T \leq 100)$. First line of each test case will contain two integers $N(1 \leq N \leq 10^4)$ and $M(1 \leq M \leq 8 \cdot 10^4)$ where N denotes the number of nodes in the graph and M denotes the number of directed edges.

Each of the next M lines will contain two integers U and $V(1 \leq U, V \leq N, U \neq V)$ denoting a directed edge from U to V. Next line of the input will contain a single integer $Q(1 \leq Q \leq 10^6)$ denoting the number of queries. Next Q lines will contain two integers U and $V(1 \leq U, V \leq N)$ for which the answer of the query needs to be processed.

It is guaranteed that, the sum of N is at most 10^5, the sum of M is at most 10^5 and the sum of Q is at most 10^6 across all test cases.

Output

For each test case process each of the Q queries. For each query print one integer in a separate line denoting the answer of the query.

Example

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

Problem I. Unhappy Team

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

You are managing a team of N people numbered from 1 to N. In the team everyone perceives everyone else as stronger or weaker than them in work performance. For the year end bonus you need to stack rank the team in some way. A stack rank P is a permutation of the numbers from 1 to N, where P_1 is the best ranked person while P_N is the worst ranked. After the stack ranking is done, it is visible to everyone in the team.

Everyone has an unhappiness score which is equal to the number of people with better rank that they deemed weaker than them. If we find the unhappiness score of everyone and sum up the largest or worst K unhappiness score then we find the score of that stack ranking. For example if the ascendingly sorted unhappiness score of a stack ranking is [0,1,2,2,3] then for $K = 2$ the score is 5(3+2) and for $K = 3$ the score is 7 (3+2+2).

Given N, K and everyone's perception of who is stronger or weaker than them, find the expected unhappiness score of a randomly chosen stack ranking. The answer can be expressed as $\frac{P}{Q}, Q \neq 0$, then answer the value of $(P \cdot Q^{-1})$ modulo 998244353.

Input

The first line of the input contains an integer T($1 \leq T \leq 50$), denoting the number of test scenarios. For each test scenario, the first line contains two integers N and K ($1 \leq K \leq N \leq 16$). The next N lines each contain N uppercase characters. The character j in line i represents what person i thinks of person j. If the character is **'S'** then i thinks j is stronger. If it is **'W'** then i thinks j is weaker. It will be **'X'** if $i = j$. It is guaranteed that the number of test cases where $N > 12$ is at most 3.

Output

For each test scenario print one line, the expected score of a randomly chosen stack ranking modulo 998244353.

Example

standard input	standard output
3 3 2 XSW SXW WSX 4 3 XSWS WXSW SSXS SWWX 1 1 X	499122178 499122179 0

Problem J. Hand Cricket

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	1024 megabytes

Alice and Bob play a game on an integer array A of length N. Alice chooses a secret index i, and Bob chooses a secret index j. If $i \neq j$ then Alice gains A_i points, else she gains 0 points. Then the game ends.

Alice wants to maximize the point. Bob wants to minimize it. Both of them will choose their own strategy in such a way that their strategy cannot be exploited, even if it is leaked to the other person.

Answer Q queries, each with a given tuple $[L, R, K]$. For each query,

- The game will be played within the subarray $A_L, A_{L+1}, ..., A_R$.
- A strategy for a player is a probability distribution $P_L, P_{L+1}, ..., P_R (P_L + P_{L+1} + ... + P_R = 1, 0 \leq P_i \leq 1)$ which implies that the player will choose index i with probability P_i.
- Before the game starts, Alice can apply at most K increment moves. An increment move is to do $A_i = A_i + 1$ for some $i \in [L, R]$. The move can be applied multiple times on the same i. The increment moves only affect the current query and will not persist to subsequent queries.

Print the expected number of points gained by Alice for each query. The answer can be expressed as an irreducible fraction $\frac{X}{Y}, Y \neq 0$. You need to print the value of $(X \cdot Y^{-1})$ modulo 998244353. It is guaranteed that for each of the queries an answer can be calculated (i.e Y^{-1} exists modulo 998244353).

Input

The first line contains an integer $N (1 \leq N \leq 2 \cdot 10^5)$. The second line has N space-separated integers $A_1, A_2, ..., A_N (1 \leq A_i \leq 10^8)$. The third line contains an integer $Q (1 \leq Q \leq 10^5)$, the number of queries. The following Q lines contain 3 integers each, $L_i, R_i, K_i (1 \leq L_i \leq R_i \leq N, 0 \leq K_i \leq 10^8)$ representing the i^{th} query.

Output

Print the answer of each query in a separate line.

Example

standard input	standard output
3 1 2 3 3 1 3 3 1 1 6 1 2 2	2 0 598946613

Problem K. Island Invasion

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

Big Island and Tiny Island are two neighboring islands in the Little Sea. The king of Big Island wants to conquer Tiny island to appease his fragile ego. However, the wind blows strong in Little Sea and the ships are slow and light. Therefore, reaching Tiny island may be difficult and time consuming; in some cases, even impossible. Furthermore, the weather is volatile so the wind direction is hard to predict. Thus the king would like to make plans in advance and find the minimum time to reach Tiny Island from Big Island given various wind velocities.

Big Island and Tiny Island can be described as two **non-intersecting convex polygons** of n and m vertices respectively. The king will ask you q queries. In each query, you will be given a vector $\vec{w} = (w_x, w_y)$ representing the wind velocity and an integer s, the maximum speed of Big Island ships. The ships can choose to depart from any point of Big Island and choose to move in any direction they like. If the ships' own velocity is described by $\vec{v}$, then $|\vec{v}|$ must not be greater than s and the ships' resulting velocity will be $\vec{v} + \vec{w}$.

Your task is to determine for each query, if it is possible to reach any point of Tiny Island. If so, you also need to find the minimum time needed.

Input

The first line contains the number of test cases $T(1 \le T \le 1111)$.

First line of each test case has two integers n and $m(3 \le n, m \le 10^5)$. Each of the next n lines contain two integers x and $y(-10^9 \le x, y \le 10^9)$, the coordinates of a vertex of Big Island. Each of the next m lines contain two integers x and $y(-10^9 \le x, y \le 10^9)$, the coordinates of a vertex of Tiny Island. For each polygon, vertices are given in counter-clockwise order.

The next line contains a single integer $q(1 \le q \le 2 \cdot 10^4)$. Each of the next q lines contain three integers, $w_x, w_y(-10^9 \le w_x, w_y \le 10^9)$ and $s(0 \le s \le 10^9)$.

It is guaranteed that the sum of $n + m$ is at most $2 \cdot 10^5$ and the sum of q is at most $2 \cdot 10^4$ across all test cases.

The input is generated in a way that the answer for any query when it is possible to reach Tiny Island would be at most 10^9.

Output

For each query output a single line.

- If it is not possible to reach any point of Tiny Island, print -1.
- Otherwise print a single real number, the minimum time needed to reach any point of Tiny Island.

Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-6}. Namely, if your answer is a, and the jury's answer is b, then your answer is considered correct, if $\frac{|a-b|}{max(1,|b|)} < 10^{-6}$.

Example

standard input	standard output
1 4 4 0 0 1 0 1 1 0 1 2 2 3 2 3 3 2 3 2 1 1 1 -1 -1 1	0.5857860308 -1

Problem L. Uncle Bob and XOR Sum

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

Uncle Bob, the Mayor of Bytelands, is not only a great leader but also an enthusiastic problem solver. He is so obsessed with problem solving that he hired a team to create new problems for him every day. But no matter how hard the problem is, Uncle Bob solves it within a few minutes. As the days pass by, he feels bored with solving such easy problems. So, he threatens the team to come up with an exciting problem or they have to look for a new job. After hours of brainstorming, the team came up with the following problem:

Given two arrays of integers A and B, determine the number of **non-empty subsets of positions** in A such that the **XOR Sum** of the values corresponding to those positions in A does not contain a submask that is present in B. More formally, if the XOR Sum of the values corresponding to a chosen subset of positions in A is S, and the set of submasks of S is $\{m_1, m_2, \ldots, m_k\}$, then: $m_i \notin B$ for all $1 \leq i \leq k$.

Since the number of valid non-empty subsets can be very large, you are required to provide the answer modulo $1,000,000,007$ $(10^9 + 7)$.

The **XOR Sum** of a subset of positions $\{p_1, p_2, \ldots, p_k\}$ is defined as: $S = A[p_1] \oplus A[p_2] \oplus \cdots \oplus A[p_k]$, where $\oplus$ denotes the bitwise XOR operator.

A **submask** of a number n is any number m such that all bits set in m are also set in n.

Looking at the problem, Uncle Bob was stunned. He never thought in a million years that he would be given a problem that requires any bitwise operations, let alone XOR sum. Now he regrets threatening the team. He is aware that not being able to solve this problem will harm his reputation as a Mayor. So, he hired you to solve it for him. Would you be able to help Uncle Bob and save him from his misery? Also, make sure not to tell anybody that Uncle Bob has hired you!

Input

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

- The first line contains two positive integers N $(1 \leq N \leq 10^5)$ and K $(1 \leq K \leq 10)$ representing the length of the arrays A and B, respectively.
- The second line contains N space-separated integers $a_1, a_2, \ldots, a_n$ $(0 \leq a_i \leq 2^{31} - 1)$.
- The third line contains K space-separated integers $b_1, b_2, \ldots, b_k$ $(0 \leq b_i \leq 2^{31} - 1)$.

You can safely assume that the sum of N over all test cases will not exceed 10^5.

Output

For each test case, print the answer modulo $1,000,000,007$ $(10^9 + 7)$ on a new line.

Example

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

Problem M. Tree Flip

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Alu will be given a rooted tree where each node contains an integer value of 0 or 1. He can perform flip operations at some nodes. This operation will flip the value of the selected node and the values of all the children of the selected node (note, it just flips the values of the children, not of all the descendants). Flipping a value means changing 0 to 1 or 1 to 0. Alu is intelligent so he performs this operation the minimum number of times to make the values of all nodes zero.

However, it's not fun if the tree is static. So here comes Begun into the scene. He has a tree. He can perform following updates on his tree:

Update 1: Begun flips the value of a particular node (note, the value of the children of the selected node are not flipped).

Update 2: Begun makes a particular node the root of the tree.

After each update, you need to output the minimum number of operations Alu requires to perform to make all the values in the tree zero. Note, Alu does not change the Begun's tree. You may imagine that Alu performs his operations on a copy of Begun's tree.

Input

Input begins with the number of test cases, T ($1 \leq T \leq 10^4$).

Each case begins with two positive integers, the number of nodes n and the number of updates q. Next line contains n integers, the i^{th} integer is the value of the i^{th} node. Each of the next $n-1$ lines consists of two integers: u and v ($1 \leq u, v \leq n$). These describe the edges of the tree.

Each of the next q lines contains two integers: $type$ ($type \in \{1, 2\}$) and x ($1 \leq x \leq n$). $type = 1$ denotes **Update 1** and $type = 2$ denotes **Update 2**. x denotes the node on which the update is performed. You may assume that the input tree is valid. Initially 1 is the root of the tree.

It is guaranteed that the sum of n doesn't exceed 10^5 and sum of q doesn't exceed 10^5 across all test cases.

Output

For each test case process the updates and then for each update print the answer in a single line.

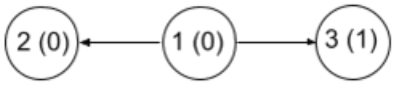
Example

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

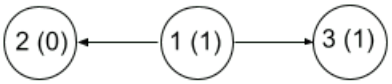
Note

Explanation for the example

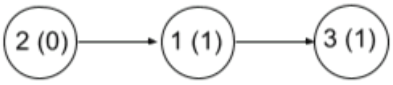
Following image contains the initial tree. Inside the brace, you will find the value of that node. The arrows of the edges go from parent to child.



First update is "1 1". It flips the value of the node 1. Alu needs two operations to make the entire tree zero. Operation on node 2, followed by operation on node 1.



Second update is "2 2". It makes the node 2 as the root of the tree. Alu needs only one operation on node 1 to make the entire tree zero.



Final update is "1 1". It flips the value of the node 1. Alu needs only one operation on node 3 to make the entire tree zero.