Problem A. Auto-correction

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

Cuber QQ is poor in English writing, and in the process of preparing this contest, he realized that he is making too many grammar mistakes that an auto-correction engine is needed. Instead of using online tools like "Microsoft Aim Writing" or "Grammarly", he was interested in building a new engine on his own.

In particular, he adopted a naive sequence-to-sequence model, that takes a sequence, which is usually a sentence, and predict for each token, which is usually a word or character, whether there is something wrong with it, and if yes, what it should be replaced with. Here are several examples:

- In "Cuber QQ was one of the admirers Quber CC.", "admirers" should be replaced with "admirers of".
- In "Cuber QQ confess his love to Cuber QQ just now.", "confess" should be replaced with "confessed".
- In "Quber CC said that they are being and always will be good friends.", "are being" should be replaced with "are".

You might notice that, in this sequence-to-sequence model, the phrase to replace should be at least one token, and the target should be at least one token too. This is related to the architecture and training approach of his model. We will not go into too many machine learning details here, as it will make the statement tedious. The problem is however, the training data does not conform with such format. In the training data, a sequence with flaws can be annotated with three types of annotations: add, delete and replace. Concretely,

- `A` l s_1 s_2 $\cdots$ s_v: to add sequence s before position l.
- `D` l r: to delete from l-th token to the r-th token, inclusive.
- `R` l r s_1 s_2 $\cdots$ s_v: to replace sub-sequence from l-th token to r-th token, inclusive, with sequence s.

All the annotations are applied directly to the original sequence, i.e., the indices like l and r refers to the original indices, instead of the indices after modification.

As "add" and "delete" are not supported in the model, the preprocessing step needs to rewrite all "add" and "delete" with "replace". Furthermore, as there are many ways to achieve such goal, Cuber QQ wants to find the cheapest way, i.e., after the annotation rewriting, the total number of replaced tokens should be as minimum as possible. If there is a tie, the number of annotation records should be as minimum as possible. In case there is still a tie, any one of them is acceptable.

Input

The input starts with an integer t ($1 \leq T \leq 50\ 000$), denoting the number of test cases.

For each test case, the first line contains two space-separated integers n and q ($1 \leq n, q \leq 2\ 000$), where n is the number of tokens in the original sequence, and q is the number of original annotations.

In the next line, n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$) are presented, denoting the sequence.

The i-th of the following q lines is in one of the 3 formats:

- `A` l_i $s_{i,1}$ $s_{i,2}$ $\cdots$ s_{i,v_i} ($1 \leq l_i \leq n+1$, $1 \leq s_{i,k} \leq n$). Notably, when $l_i = n+1$, it is to add tokens at the end of sequence.
- `D` l_i r_i ($1 \leq l_i \leq r_i \leq n$).
- `R` l_i r_i $s_{i,1}$ $s_{i,2}$ $\cdots$ s_{i,v_i} ($1 \leq l_i \leq r_i \leq n$, $1 \leq s_{i,k} \leq n$).

It is guaranteed that $l_i \leq l_{i+1}$ for all $1 \leq i < n$, and $l_i = l_{i+1}$ only happens when i is `A` and $i+1$ is not, which means, there is at most one "add" at the same position. The annotations are non-overlapping, i.e., $r_i \leq l_{i+1}$ for all $1 \leq i < n$ if r_i is available for i. Furthermore, the corrected sequence after applying all the annotations is not empty.

It is guaranteed that for each test case, the corrected sequence is neither empty nor longer than 4 000. The sum of n and the total length of corrected sequences both do not exceed 50 000.

The numbers and characters in each line are separated with a single space, and there is no extra space at the beginning or end of line.

Output

For each test case, in the first line output two space-separated integers: minimum number of tokens that will be replaced, x, and minimum number of converted annotations, y.

In the following y lines, you can output the annotations in any order. `R` should be omitted as it is the only type that is allowed. The annotations should be non-overlapping, non-empty and follow the exactly same format as input.

Example

standard input	standard output

3
6 2
1 2 5 3 4 6
R 2 3 1
R 4 6 2 3
6 3
1 2 2 3 4 6
R 3 4 4
D 5 5
A 7 5
6 2
1 2 3 4 5 6
A 1 2
A 6 5
	5 1
	2 6 1 2 3
	4 1
	3 6 4 6 5
	2 2
	6 6 5 6
	1 1 2 1

Note

For the first test case, $[2, 3]$ is replaced with `1`, $[4, 6]$ is replaced with `2 3`, and the corrected sequence is `1 1 2 3`. The optimal correction with only `R` is to replace $[2, 6]$ with `1 2 3`.

For the second test case, the corrected sequence is `1 2 4 6 5`. Although `A` and `D` cannot be used, if we merge the consecutive annotations, only 4 tokens need to be replaced.

In the third test case, we show that the corrected sequence can be longer than the original sequence, which is `2 1 2 3 4 5 5 6`.

Problem B. Breaking Down News

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

Cuber QQ, who is the producer of Breaking News, is tired of all the pressures from producing high-quality content and making sponsors happy. The most painful part of this job is, actually, to put all the news clips into units, between which advertisements are broadcast. Recently, after acquiring some knowledge from technology department, he found that he can actually leverage some help from algorithms.

Formally, the sequence of news clips to be broadcast tonight is a sequence a consisting of n integers, where $a = a_1, a_2, \ldots, a_n$ ($a_i \in \{-1, 0, 1\}$), which are the expected quality of those news clips. Being of negative quality does not mean that the clip will not be broadcast. Actually all the contents have already been carefully selected to fit the length of the TV show. Also, the clips should follow the order, which is already well arranged.

To insert ads between clips, Cuber QQ asks you to split the sequence into m ($m \geq 1$) non-empty units, i.e., consecutive subsequences, so that every clip belongs to exactly one unit. Formally, if the k-th unit is $a_{l_k}, a_{l_k+1}, \cdots, a_{r_k}$, then $r_k + 1 = l_{k+1}$ for all $1 \leq k < m$.

Furthermore, each unit should be of moderate length, neither too long nor too short. Formally, $L \leq r_i - l_i + 1 \leq R$ should hold for all $1 \leq k \leq m$.

The quality of a unit, is defined to be 1 if quality sum of its corresponding clips is greater than 0, -1 if the sum is lower than 0, and 0 otherwise. This can be formulated with $v_k = [(\sum_{i=l_k}^{r_k} a_i) > 0] - [(\sum_{i=l_k}^{r_k} a_i) < 0]$.

The problem is to determine the optimal m and the split plan, to maximize the sum of values of all units, i.e., to maximize $\sum_{k=1}^{m} v_k$.

Input

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

For each of the next T cases:

- The first line contains three space-separated integers n, L, R ($1 \leq L \leq R \leq n \leq 10^6$).
- The second line contains n integers $a_1, a_2, \ldots, a_n$ ($a_i \in \{-1, 0, 1\}$).

It is guaranteed that at least one valid split exists.

The sum of n in all test cases doesn't exceed $9 \cdot 10^6$.

Output

For every test case, output one line containing an integer, denoting the maximum value.

Example

standard input	standard output

3
5 1 5
1 -1 0 -1 1
5 5 5
-1 1 -1 1 -1
5 1 1
1 -1 0 -1 1
	1
	-1
	0

Problem C. Clockwise or Counterclockwise

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

After some basic geometric lessons, Cuber QQ has learned that one can draw one and only one circle across three given distinct points, on a 2D plane. Specialized in art, Cuber QQ has shown remarkable skills to draw circle in one stroke, especially when the stroke is done clockwise. He wonder whether he will be able to do that if 3 points has been given.

In particular, he is given three distinct points $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ which lie on a circle centered at $O(0, 0)$. Imagine starting from A, he draws the circle across B and finally gets C. Determine whether he is drawing clockwise or counterclockwise.

Input

The first line contains an integer T $(1 \le T \le 1\ 000)$, denoting the number of test cases.

In the next T lines, each line contains six space-separated integers x_1, y_1, x_2, y_2, x_3, y_3 $(-10^9 \le x_1, y_1, x_2, y_2, x_3, y_3 \le 10^9)$ denoting the coordinate of A, B and C.

It is guaranteed that A, B, C are pairwise distinct and $|AO| = |BO| = |CO| > 0$.

Output

For each test case, output one line containing "Clockwise" or "Counterclockwise".

Example

standard input	standard output

3
1 2 2 1 -1 -2
4 3 -4 3 3 4
4 -3 4 3 3 4
	Clockwise
	Clockwise
	Counterclockwise

Problem D. Discovery of Cycles

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

In response to the 8202 Olympics, Quber City, which was the host of the competition, is planning to build a magnificent stadium. The stadium is nowhere close to traditional stadiums that are typically embedded with a cyclic running trail. According to the plan, there are n service spots in the stadium, and m undirected running trails connecting them. Such adventurous design has brought worldwide attention to the stadium, but also comes with a huge cost.

Therefore the designers are working to simplify the design to cut down the cost. They discovered that the easiest way to do this is to sort the m running trails in some particular order, and keep only some consecutive trails in the list. Since only part of the trails will be built, fund will be saved and everyone will be happy — perhaps except those long-distance runners who needs a cyclic trail for the competition. A cyclic trail is a trail that starts at some service spot, passes through some distinct spots, before getting back to where it starts. All the running trails that have been used are required to be distinct. (This is by definition a *simple cycle* if you are familiar with terms in graph theory.)

Your task is to write a program that can quickly check, whether one can find at least one simple cycle, given that the stadium is built with some particular running trails selected consecutively from the trail list.

Input

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

For each of the next T cases, the first line contains three space-separated integers n, m, q $(1 \le n, m, q \le 3 \cdot 10^5)$, denoting the number of service spots, number of running trails in the initial plan, and the number of queries.

In the next m lines, the i-th line contains two space-separated integers u_i, v_i $(1 \le u_i, v_i \le n$, $u_i \ne v_i)$.

Each of the next q lines contains two space-separated integers l', r' $(1 \le l' \le r' \le m)$. l and r can be calculated via the following procedure.

- Let lastans denote the answer to the last query (if the answer is "`Yes`", lastans $= 1$, otherwise lastans $= 0$) and is initially zero.
- $k_1 = (l' \text{ xor lastans}) \bmod m + 1$.
- $k_2 = (r' \text{ xor lastans}) \bmod m + 1$.
- $l = \min(k_1, k_2)$.
- $r = \max(k_1, k_2)$.

l and r means the edges in $[l, r]$ will be chosen. That means, for this query, only edges $(u_l, v_l), (u_{l+1}, v_{l+1}), \ldots, (u_r, v_r)$ are visible.

It is guaranteed that $\sum n, \sum m \le 1.5 \cdot 10^6$.

Output

For every query, output "`Yes`" in one line if edges that are chosen forms at least one cyclic trail, otherwise output "`No`" in one line.

Example

standard input	standard output

1
3 4 3
1 2
1 2
2 3
3 1
1 4
2 3
2 4
	Yes
	No
	Yes

Problem E. Easy NPC Problem

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

Cuber QQ fell in love with research of NPC problem. He is very passionate in those cutting-edge thinkings, especially in Hamiltonian path problem, which is a well-known typical NPC problem.

In the mathematical field of graph theory, a Hamiltonian path is a path in an undirected or directed graph that visits each vertex exactly once. Mathematicians have spent hundreds of years, trying to find a general yet elegant solution of this problem, but still, the problem is only solved in a limited scope.

Cuber QQ wants to take one step forward, by solving the Hamiltonian path problem in the grid. A grid has $n \times m$ vertices. We use a typical coordinate system in the grid, where each vertex on the grid is labelled by a pair of integers (x, y) $(1 \leq x \leq n, 1 \leq y \leq m)$, and it is connected to adjacent vertices (if they are available), i.e., $(x-1, y)$, $(x+1, y)$, $(x, y-1)$, $(x, y+1)$.

The problem seems too trivial to him, Cuber QQ will take another step forward to find a Hamiltonian path in the grid without visiting the vertex (N_x, N_y) $(1 \leq N_x \leq n, 1 \leq N_y \leq m)$, and the starting vertex must be located at (S_x, S_y) $(1 \leq S_x \leq n, 1 \leq S_y \leq m)$. It is a little too difficult for him now and he asks you for help.

Input

The first line contains an integer T $(T \leq 2\ 500)$, denoting the number of test cases.

Each test case has six space-separated integers n, m, N_x, N_y, S_x, S_y $(1 \leq N_x, S_x \leq n \leq 200, 1 \leq N_y, S_y \leq m \leq 200, N_x \neq S_x \text{ or } N_y \neq S_y)$ in one line.

It is guaranteed that $\sum n + m \leq 10^5$.

Output

For each test case, if there is no solution, print a single integer -1 in one line, otherwise the first line output an integer $nm-1$, which is the length of the path. The next $nm-1$ lines contain the vertices in the visiting order. and each of the n lines contains two space-separated integers x, y $(1 \leq x \leq n, 1 \leq y \leq m)$.

If there are multiple solutions, print any.

Example

standard input	standard output

```
2
2 4 1 2 1 1
2 4 2 2 1 1
```
	```
	7
	1 1
	2 1
	2 2
	2 3
	2 4
	1 4
	1 3
	-1
	```

Problem F. Fluctuation Limit

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

Cuber QQ has signed up for a gambling game, that challenges him to predict the stock price of Quber CC Limited, for the next following n days. He shall make his prediction by filling a table with n intervals, the i-th of which is the predicted interval $[l_i, r_i]$ at the i-th day. If all n prices lie in the corresponding interval, Cuber QQ will win 1 million dollars. Otherwise, he will not earn a single penny.

As is well known, the stock price has a fluctuation limit. For simplicity, we assume the limit up and the limit down are both k, which is an integer. That means, if the stock price at the i-th day is x, the price at the $i+1$-th day is at most $x+k$ and at least $x-k$.

Cuber QQ wants to know whether it is possible to manipulate the stock price, without breaking the limitation above of course, so that he can have the 1 million dollars. Since his table has already been submitted, he cannot modify his predicted intervals any more. It has to be done secretly behind the scenes, and smartly cover it up so that no one will notice.

Input

The input starts with an integer T ($1 \le T \le 10^5$), denoting the number of test cases.

For each test case, the first line contains two space-separated integers n and k ($2 \le n \le 10^5$, $0 \le k \le 10^9$), where n is the number of days and k is the fluctuation limit.

The i-th line of the next n lines contains two space-separated integers l_i and r_i ($0 \le l_i \le r_i \le 10^9$), which is Cuber QQ's predicted interval in the i-th day. A prediction is believed to be correct if the stock price i-th day lies between l_i and r_i, inclusive.

It is guaranteed that the sum of all n does not exceed 10^6.

Output

For each test case, first output a single line `YES` or `NO`, that states whether Cuber QQ will win the 1 million price.

If `YES`, in the next line, output a possible price series, $a_1, a_2, \ldots, a_n$, where $l_i \le a_i \le r_i$ ($1 \le i \le n$) and $|a_i - a_{i+1}| \le k$ ($1 \le i \le n-1$). The integers should be separated with space.

Example

standard input	standard output

2
3 1
1 6
2 5
3 6
2 3
1 5
10 50
	YES
	2 3 3
	NO

Problem G. Gaming of Co-prime Disallowance

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

Cuber QQ and Little Fang are playing a game, called Gaming of Co-prime Disallowance (GCD).

To play GCD, two tables are needed. The game starts with n cards on one table, and players take turns to move the cards to another table. In each turn, a player has to select one and only one card and move it. Let's assume that the players have moved k cards, and there are $n - k$ cards left. The players who cannot make a move any more loses the game.

If one of the following two conditions happen, the players can no longer make a move:

- There is no card on the original table, i.e., $k = n$.
- On each card, there is a number. If the k cards that have been moved have Greatest Common Divisor (GCD)(`https://en.wikipedia.org/wiki/Greatest_common_divisor`) equals 1, i.e., the k numbers are co-prime, the game is over and the player who has made the last valid move wins the game.

To test the fairness of GCD, Cuber QQ and Little Fang plays completely randomly without any strategy at all. Help him calculate how likely he is going to win if Cuber QQ plays first.

Input

The first line of the input contains a single integer T ($1 \leq T \leq 150$), denoting the number of test cases.

Each of the next T cases:

- The first line contains an integer n ($2 \leq n \leq 100$).
- The second line contains n space-separated numbers $a_1, a_2, \cdots a_n$ ($1 \leq a_i \leq 10^5$), the number on n cards, respectively.

It is guaranteed that $\sum n \leq 8\ 000$.

Output

For each test case, output one line contains a real number — the probability that Cuber QQ will win the game.

Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Example

standard input	standard output

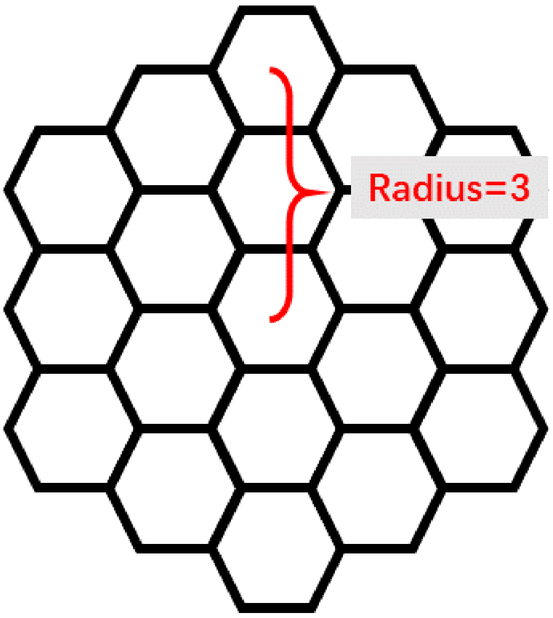
2
4
1 2 3 4
4
1 2 4 8
	0.583333333
	0.500000000

Problem H. Hexagon

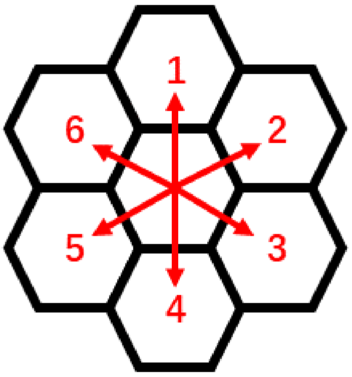
Input file:	`standard input`
Output file:	`standard output`

If the world is a hexagon, I will take as many turns as possible before reaching the end.

Cuber QQ has constructed a hexagon grid with radius n. This will be better if explained in a picture. For example, here is a hexagon grid with radius 3:



He challenges you to take a perfect tour over the hexagon, that is to visit each cell exactly once. Starting from any point in the grid, you can move to any adjacent cell in each step. There are six different directions you can choose from:



Of course, if you are on the boundary, you cannot move outside of the hexagon.

Let $D(x, y)$ denote the direction from cell x to y, and sequence A denotes your route, in which A_i denotes the i-th cell you visit. For index i $(1 < i < |A|)$, if $D(A_{i-1}, A_i) \neq D(A_i, A_{i+1})$, we say there is a *turning* on cell i.

Maximize the number of turning while ensuring that each cell is visited exactly once. Print your route. If there are multiple solution, print any.

Input

The first line of the input contains a single integer T $(1 \leq T \leq 10^4)$, denoting the number of test cases.

Each of the next T cases:

- The first line contains an integer n ($2 \le n \le 500$).

It is guaranteed that the sum of n doesn't exceed $2 \cdot 10^4$.

Output

For each test case, output one line contains a string with $n-1$ characters. The i-th character is $D(A_i, A_{i+1})$.

As you might have guessed, you do not need to print your starting cell, as Cuber QQ will figure out himself according to your planned route.

Example

standard input	standard output
1 2	313456

Problem I. Isomorphic Strings

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

Two strings are called *cyclical isomorphic* if one can rotate one string to get another one. 'Rotate' here means "to take some consecutive chars (maybe none) from the beginning of a string and put them back at the end of the string in the same order". For example, string "abcde" can be rotated to string "deabc".

Now that you know what *cyclical isomorphic* is, Cuber QQ wants to give you a little test.

Here is a string s of length n. Please check if s is a concatenation of k strings, $s_1, s_2, \cdots, s_k$ ($k > 1$), where,

- k is a divisor of n;
- $s_1, s_2, \ldots, s_k$ are of equal length: $\frac{n}{k}$;
- There exists a string t, which is *cyclical isomorphic* with s_i for all $1 \leq i \leq k$.

Print "`Yes`" if the check is positive, or "`No`" otherwise.

Input

The first line contains an integer T ($1 \leq T \leq 1000$), denoting the number of test cases. T cases follow.

- The first line of each test case contains an integer n ($1 \leq n \leq 5 \cdot 10^6$).
- The second line contains a string s of length n consists of lowercase letters only.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^7$.

Output

For each test case, output one line containing "`Yes`" or "`No`" (without quotes).

Example

standard input	standard output
6 1 a 2 aa 3 aab 4 abba 6 abcbcc 8 aaaaaaaa	No Yes No Yes No Yes

Problem J. Jumping on a Cactus

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

Cuber QQ has recently learned jumping on graphs. He now wants to demonstrate his skills on a *cactus*.

Recall that, *cactus* refers to a graph with every edge appearing in **at most one** *cycle*. If you do not know what a *cycle* is, formally, a *cycle* of length k denotes an edge sequence $(u_1, u_2), (u_2, u_3), \ldots, (u_{k-1}, u_k), (u_k, u_1)$.

Assuming you are given an undirected cactus $G = (V, E)$, with n vertices and m edges. A jumping on the cactus can be thought of as a visit to all vertices where every vertex is visited exactly once. That can be represented with a permutation of 1 to n, $p_1, p_2, \ldots, p_n$, and they are visited in order.

Cuber QQ also wishes his jumping to be gradually approaching or distancing away from a particular node e. Concretely, we define $d(a, b)$ to be the distance from a to b (the minimum edges needs to be passed through from a to b). A jumping is defined to be monotonic if,

- for all edges $(u, v) \in E$, $d(u, e) < d(v, e)$ when u is visited before v, or,
- for all edges $(u, v) \in E$, $d(u, e) > d(v, e)$ when u is visited after v.

Count the number of different monotonic jumping plans. Since the answer can be very large, you should print the answer modulo 998 244 353.

Input

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

For each of the next T cases:

- The first line contains three space-separated integers n, m, e $(2 \le n \le 5\ 000,\ 1 \le m \le 2(n-1),\ 1 \le e \le n)$.
- The i-th of the next m lines contains two space-separated integers u_i, v_i $(1 \le u_i, v_i \le n,\ u_i \ne v_i)$. It is guaranteed that $d(u_i, e) \ne d(v_i, e)$.

There are at most 10 test cases where $n \ge 1\ 000$.

Output

For each test case, output one line contains one integer denoting the answer modulo 998 244 353.

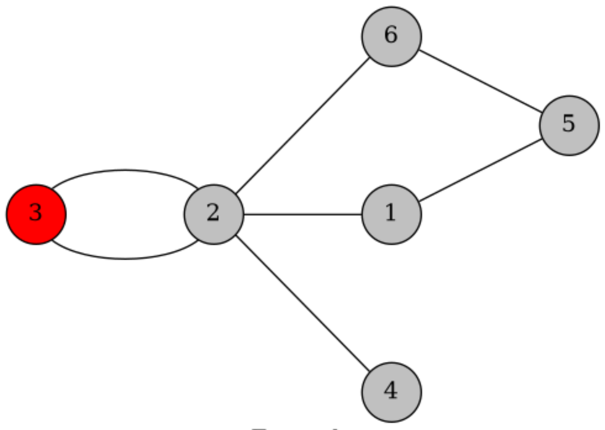
Example

standard input	standard output

1
6 7 3
6 2
5 6
1 5
1 2
3 2
3 2
4 2
	8

Note

For the example, G looks like:



Example

3 is the index of reference vertex.

There are 8 correct permutations:

- $\{3, 2, 4, 1, 6, 5\}$.
- $\{3, 2, 1, 4, 6, 5\}$.
- $\{3, 2, 1, 6, 4, 5\}$.
- $\{3, 2, 1, 6, 5, 4\}$.
- $\{3, 2, 4, 6, 1, 5\}$.
- $\{3, 2, 6, 4, 1, 5\}$.
- $\{3, 2, 6, 1, 4, 5\}$.
- $\{3, 2, 6, 1, 5, 4\}$.

Problem K. Kidnapper's Matching Problem

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

One day when you were sitting in the couch and eating chips, a guy called you, who claimed that his name was Cuber QQ and he had your girlfriend kidnapped. This seems quite unlikely to you because you do not even have a girlfriend. However, to kill the boring time of Sunday afternoon, you asked how much the ransom is. Surprisingly, Cuber QQ is not interested in your money, the ransom is the answer to a matching problem. It turns out that Cuber QQ is a lover of binary operators, xor especially.

First of all, he shall give you a *pair* of sequences a and b, with length n and m respectively.

Given that a and b do not necessarily have the same length, they do not exactly make a *pair*. Therefore consecutive subsequences of a are used to make pairs with b, That makes $n-m+1$ pairs: $a_l, a_{l+1}, \ldots, a_{l+m-1}$ and b for all $1 \le l \le n-m+1$.

For each pair $(a_l, a_{l+1}, \ldots, a_{l+m-1}; b)$, we say they are xor-matched if their *pairwise-xor*'s are all available in a pre-defined set S. Concretely, the pair is a match if and only if $a_{l+i-1} \oplus b_i \in 2^S_\oplus$ of all $1 \le i \le m$. $2^S_\oplus$ is defined as the set of all possible *xor-sum* of S's subsets, i.e.,

$$2^S_\oplus = \{t | t = \bigoplus_{w \in X} w, X \subseteq S\}$$

Note that, since $\{\}$ is always a valid subset of S, 0 is always included in $2^S_\oplus$.

Now Cuber QQ wants you to tell him, for which l's, $a_l, a_{l+1}, \cdots, a_{l+m-1}$ makes a match with b. He is not sending your illusory girlfriend back before you correctly answer his question.

Input

The first line contains an integer T $(1 \le T \le 2 \cdot 10^4)$, denotes the number of test cases.

Each test case begins with three space-separated integers n $(1 \le n \le 2 \cdot 10^5)$, m $(1 \le m \le \min(n, 5 \cdot 10^4))$ and k $(1 \le k \le 100)$, denoting the lengths of a, b and the size of S.

The next line contains n space-separated integers $a_1, a_2, \cdots, a_n$ $(0 \le a_i < 2^{30})$.

The next line contains m space-separated integers $b_1, b_2, \cdots, b_m$ $(0 \le b_i < 2^{30})$.

The last line contains k space-separated integers $S_1, S_2, \cdots, S_k$ $(0 \le S_i < 2^{30})$.

It is guaranteed that $\sum n \le 1.2 \cdot 10^6, \sum m \le 3 \cdot 10^5, \sum k \le 6 \cdot 10^5$.

Output

For each case, you should output:

$$\sum_{i=1}^{n-m+1} [(a_i, a_{i+1}, \cdots, a_{i+m-1}) \text{ matches } b] \cdot 2^{i-1} \bmod (10^9+7)$$

where [condition] equals 1 if condition holds and 0 otherwise.

Example

standard input	standard output

2
4 2 2
2 5 7 6
3 4
5 6
13 3 3
1 1 4 5 1 4 1 9 1 9 8 1 0
2 5 1
1 2 15
	2
	80

Problem L. Linuber File System

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

The Linuber File System, which is designed by Cuber QQ, has a sophisticated privilege management system. Specifically, each file is associated with a confidential score (an integer), and only users with access level greater than or equal to the confidential score of the file can access that file.

LFS also organizes files in a directed tree, like normal file systems (e.g., NTFS, Ext4 and APFS) do. One subtle difference is that, in LFS, everything is file. Some files are linked with some other files as its *subfiles*, but they are considered as files instead of directories as they are associated with data on their own.

One advantage of using LFS is that, it has a really flexible management of privileges. For example, when a file becomes more or less confidential, one can increase/decrease its confidential score. At the same time, the confidential scores of its subfiles, sub-subfiles and so on will be updated accordingly. This can be thought of as an advantage, or a side effect, depending on your need. A mathematical model is, given a directed tree, you can pick any node u on the tree, and add x (positive or negative, possibly zero) to the values for each node in the subtree of u.

Given a LFS with confidential scores at 0 for all files, you are asked to perform the above operation as few times as possible, to make sure the confidential score of file i lies in the interval $[l_i, r_i]$ for all $1 \leq i \leq n$, where n is the number of files in this system. In other words, the file i should not be accessible to users with access level lower than l_i, but people with level at least r_i will have their access guaranteed.

Input

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

Each test case begins with a single integer n ($2 \leq n \leq 2\,000$), denoting the number of files.

In the next $n-1$ lines, each line contains two space-separated integers u_i, v_i ($1 \leq u_i, v_i \leq n$), denoting the endpoints of each linkage. It is not clear whether u_i is a subfile of v_i or v_i is a subfile of u_i. You should infer that on your own, as **it is known that 1 is the root file, i.e., file with no parent file.**

In the next n lines, each line contains two space-separated integers l_i, r_i ($-10^9 \leq l_i \leq r_i \leq 10^9$), denoting the required confidential score interval for file i.

Output

For each test case, print a single line containing an integer, which is the answer.

Example

standard input	standard output

2
3
1 2
1 3
-2 -1
-3 -2
-1 -1
6
1 5
2 4
2 1
1 3
3 6
-2 4
-3 2
4 5
2 2
-5 -1
-1 4
	2
	3