A. Identify the Operations

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

We start with a permutation $a_1, a_2, \ldots, a_n$ and with an empty array b. We apply the following operation k times.

On the i-th iteration, we select an index t_i ($1 \le t_i \le n - i + 1$), remove a_{t_i} from the array, and append one of the numbers a_{t_i-1} or a_{t_i+1} (if $t_i - 1$ or $t_i + 1$ are within the array bounds) to the right end of the array b. Then we move elements $a_{t_i+1}, \ldots, a_n$ to the left in order to fill in the empty space.

You are given the initial permutation $a_1, a_2, \ldots, a_n$ and the resulting array $b_1, b_2, \ldots, b_k$. All elements of an array b are **distinct**. Calculate the number of possible sequences of indices $t_1, t_2, \ldots, t_k$ modulo $998\,244\,353$.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 100\,000$), denoting the number of test cases, followed by a description of the test cases.

The first line of each test case contains two integers n, k ($1 \le k < n \le 200\,000$): sizes of arrays a and b.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$): elements of a. All elements of a are **distinct**.

The third line of each test case contains k integers $b_1, b_2, \ldots, b_k$ ($1 \le b_i \le n$): elements of b. All elements of b are **distinct**.

The sum of all n among all test cases is guaranteed to not exceed $200\,000$.

Output

For each test case print one integer: the number of possible sequences modulo $998\,244\,353$.

Example

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

Note

Let's denote as $a_1 a_2 \ldots \cancel{a_i}\, \underline{a_{i+1}} \ldots a_n \to a_1 a_2 \ldots a_{i-1} a_{i+1} \ldots a_{n-1}$ an operation over an element with index i: removal of element a_i from array a and appending element a_{i+1} to array b.

In the first example test, the following two options can be used to produce the given array b:

- $12\underline{3}\cancel{4}5 \to 12\underline{\cancel{3}}5 \to 1\cancel{2}\underline{5} \to 12$; $(t_1, t_2, t_3) = (4, 3, 2)$;
- $12\underline{3}\cancel{4}5 \to \cancel{1}\underline{2}35 \to 2\cancel{3}\underline{5} \to 15$; $(t_1, t_2, t_3) = (4, 1, 2)$.

In the second example test, it is impossible to achieve the given array no matter the operations used. That's because, on the first application, we removed the element next to 4, namely number 3, which means that it couldn't be added to array b on the second step.

In the third example test, there are four options to achieve the given array b:

- $14\cancel{7}\underline{3}625 \to 143\cancel{6}\underline{2}5 \to \cancel{1}\underline{4}325 \to 43\cancel{2}\underline{5} \to 435$;
- $14\cancel{7}\underline{3}625 \to 143\cancel{6}\underline{2}5 \to 14\underline{\cancel{3}}25 \to 14\cancel{2}\underline{5} \to 145$;
- $1473\underline{\cancel{6}}25 \to 147\cancel{3}\underline{2}5 \to \cancel{1}\underline{4}725 \to 47\cancel{2}\underline{5} \to 475$;
- $1473\underline{\cancel{6}}25 \to 147\cancel{3}\underline{2}5 \to 14\underline{\cancel{7}}25 \to 14\cancel{2}\underline{5} \to 145$;

B. Graph Transpositions

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a directed graph of n vertices and m edges. Vertices are numbered from 1 to n. There is a token in vertex 1.

The following actions are allowed:

- Token movement. To move the token from vertex u to vertex v if there is an edge $u \to v$ in the graph. This action takes 1 second.
- Graph transposition. To transpose all the edges in the graph: replace each edge $u \to v$ by an edge $v \to u$. This action takes increasingly more time: k-th transposition takes 2^{k-1} seconds, i.e. the first transposition takes 1 second, the second one takes 2 seconds, the third one takes 4 seconds, and so on.

The goal is to move the token from vertex 1 to vertex n in the shortest possible time. Print this time modulo $998\,244\,353$.

Input

The first line of input contains two integers n, m ($1 \le n, m \le 200\,000$).

The next m lines contain two integers each: u, v ($1 \le u, v \le n; u \ne v$), which represent the edges of the graph. It is guaranteed that all ordered pairs (u, v) are distinct.

It is guaranteed that it is possible to move the token from vertex 1 to vertex n using the actions above.

Output

Print one integer: the minimum required time modulo $998\,244\,353$.

Examples

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

input
4 3 2 1 2 3 4 3
output
10

Note

The first example can be solved by transposing the graph and moving the token to vertex 4, taking 2 seconds.

The best way to solve the second example is the following: transpose the graph, move the token to vertex 2, transpose the graph again, move the token to vertex 3, transpose the graph once more and move the token to vertex 4.

C. Sum

time limit per test: 1.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given n **non-decreasing** arrays of non-negative numbers.

Vasya repeats the following operation k times:

- Selects a non-empty array.
- Puts the first element of the selected array in his pocket.
- Removes the first element from the selected array.

Vasya wants to maximize the sum of the elements in his pocket.

Input

The first line contains two integers n and k ($1 \le n, k \le 3\,000$): the number of arrays and operations.

Each of the next n lines contain an array. The first integer in each line is t_i ($1 \le t_i \le 10^6$): the size of the i-th array. The following t_i

integers $a_{i,j}$ ($0 \le a_{i,1} \le \ldots \le a_{i,t_i} \le 10^8$) are the elements of the i-th array.

It is guaranteed that $k \le \sum_{i=1}^{n} t_i \le 10^6$.

Output
Print one integer: the maximum possible sum of all elements in Vasya's pocket after k operations.

Example

input
3 3 2 5 10 3 1 2 3 2 1 20
output
26

D. Black, White and Grey Tree

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a tree with each vertex coloured white, black or grey. You can remove elements from the tree by selecting a subset of vertices in a single connected component and removing them and their adjacent edges from the graph. The only restriction is that you are not allowed to select a subset containing a white and a black vertex at once.

What is the minimum number of removals necessary to remove all vertices from the tree?

Input
Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 100\,000$), denoting the number of test cases, followed by a description of the test cases.

The first line of each test case contains an integer n ($1 \le n \le 200\,000$): the number of vertices in the tree.

The second line of each test case contains n integers a_v ($0 \le a_v \le 2$): colours of vertices. Gray vertices have $a_v = 0$, white have $a_v = 1$, black have $a_v = 2$.

Each of the next $n - 1$ lines contains two integers u, v ($1 \le u, v \le n$): tree edges.

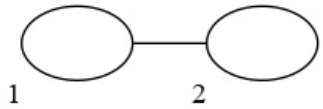
The sum of all n throughout the test is guaranteed to not exceed $200\,000$.

Output
For each test case, print one integer: the minimum number of operations to solve the problem.

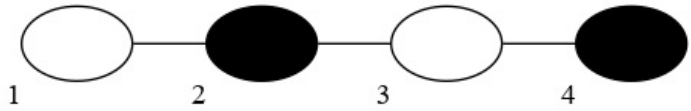
Example

input
4 2 1 1 1 2 4 1 2 1 2 1 2 2 3 3 4 5 1 1 0 1 2 1 2 2 3 3 4 3 5 8 1 2 1 2 2 2 1 2 1 3 2 3 3 4 4 5 5 6 5 7 5 8
output
1 3 2 3

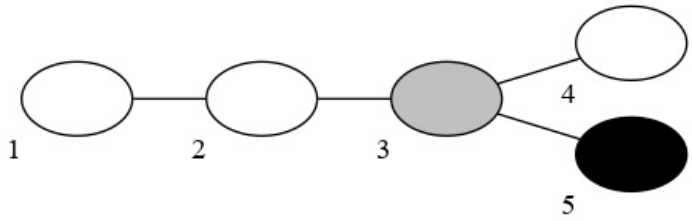
Note



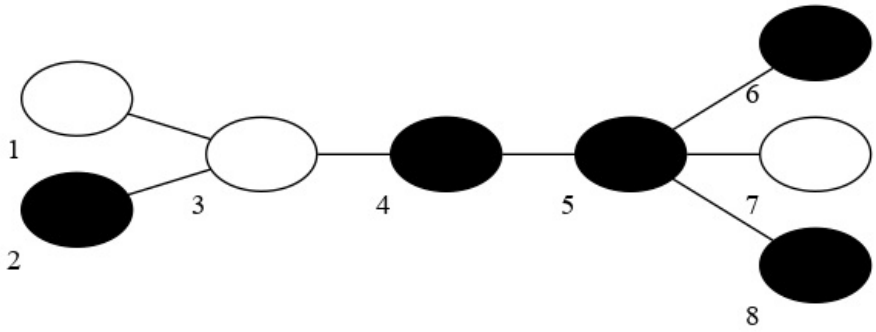
In the first test case, both vertices are white, so you can remove them at the same time.



In the second test case, three operations are enough. First, we need to remove both black vertices (2 and 4), then separately remove vertices 1 and 3. We can't remove them together because they end up in different connectivity components after vertex 2 is removed.



In the third test case, we can remove vertices 1, 2, 3, 4 at the same time, because three of them are white and one is grey. After that, we can remove vertex 5.



In the fourth test case, three operations are enough. One of the ways to solve the problem is to remove all black vertices at once, then remove white vertex 7, and finally, remove connected white vertices 1 and 3.

E. Differentiating Games

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

This is an interactive problem

Ginny is taking an exam on game theory. The professor is tired of hearing the same answers over and over again, so he offered Ginny to play a game instead of a standard exam.

As known from the course, a combinatorial game on a graph with multiple starting positions is a game with a directed graph and multiple starting vertices holding a token each. Two players take turns moving one of the tokens along the graph edges on each turn. The player who can't make a move loses the game. If both players can play an infinitely long game without losing, a draw is called.

For the exam, the professor drew an acyclic directed graph and chose one of its vertices. Ginny needs to guess the vertex the professor chose. To do so, Ginny can choose a multiset of vertices S several times and ask the professor: "If I put one token in each vertex of the given graph for each occurrence of the vertex in the multiset S, and then one more in the selected vertex, what would be the result of the combinatorial game?".

Having given the task, the professor left the room to give Ginny some time to prepare for the game. Ginny thinks that she's being tricked because the problem is impossible to solve. Therefore, while the professor is away, she wants to add or remove several edges from the graph. Even though the original graph was acyclic, edges could be added to the graph to make cycles appear.

Interaction

In this task, interaction consists of several phases.

In the first phase, the interactor gives as an input to your program three integers N ($1 \le N \le 1000$), M ($0 \le M \le 100\,000$), T ($1 \le T \le 2000$): the number of vertices and edges in the initial graph, and the number of times Ginny has to guess the chosen vertex. The next M lines contain pairs of vertices a_i b_i ($1 \le a_i, b_i \le N$): beginning and end of corresponding graph edges. The graph is guaranteed to be acyclic and all of its edges to be distinct.

The solution should print an integer K ($0 \le K \le 4242$): the number of edges to change in the graph. The next K lines should contain either "+ a_i b_i" or "- a_i b_i": the beginning and the end of an edge that Ginny has to add or remove accordingly. You are allowed to add preexisting edges to the graph. Operations are performed in the order of appearance, so Ginny is allowed to remove an edge added by the solution. You can only remove an existing edge. The operations can create cycles in the graph.

The next T phases are dedicated to guessing the chosen vertices. In each phase, the solution can make at most 20 queries and then print the answer. To query a multiset S, the solution should print "? $|S|$ S_1 S_2 ... $S_{|S|}$". The total size of all multisets in a single phase should not exceed 20. The interactor will reply with one of the following words:

- "Win", if the winner of a combinatorial game with tokens in multiset S and the selected vertex is the first player.
- "Lose", if the winner of a combinatorial game with tokens in multiset S and the selected vertex is the second player.
- "Draw", if a combinatorial game with tokens in multiset S and selected vertex ends in a draw.
- "Slow", if the solution made a 21-st request, or the total size of all multisets in a single phase exceeded 20. In this case, the solution should terminate and receive Wrong Answer verdict.

As soon as the selected vertex is guessed, that solution should print "! v". If the chosen vertex is guessed correctly, the interactor will print Correct and the solution should either move on to the next phase of guessing or finish its execution if it's the last phase. Otherwise, the interactor will print Wrong, which means that the solution should terminate and will receive the Wrong Answer verdict.

The interactor can change the chosen vertex based on graph changes and solution actions, but at every given moment of time, at least one vertex that corresponds to all given interactor answers will exist.

Hack format

Hacks have the following extra limitations:

- $T = 1$
- you need to specify a single vertex, chosen by the interactor.

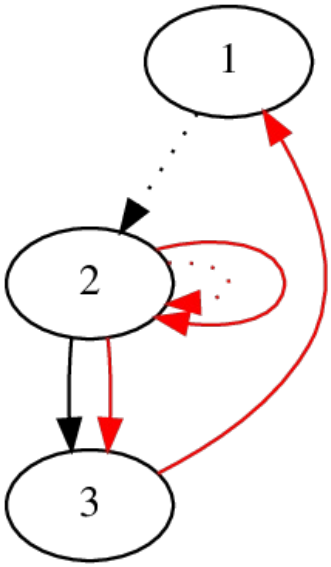
Hack test format. The first line of input contains three integers N M 1. The next M lines on input contain edge description in the same format as in the input. The next line contains a single integer v: the number of the chosen vertex. The hack will be successful even if the solution guesses the vertex right, but the vertex will not be the single one to match all performed queries.

Example

input
3 2 3 1 2 2 3 Lose Correct Win Correct Draw Correct
output
6 + 2 2 - 1 2 + 2 3 - 2 2 + 3 1 + 2 2 ? 0 ! 1 ? 1 2 ! 3 ? 5 1 3 1 3 1 ! 2

Note

In the sample test, the empty lines represent waiting for the input by the other side of the interaction. The real interactor will not print empty lines, and the solution should not print them either.



The image above illustrates the sample test. Added edges are coloured in red, and the removed edges are drawn with a dotted line. Three guessing phases denote different ways of getting the answer.

- If the solution will query just the chosen vertex, the interactor will return the result of the game in that vertex. The first player loses only if the chosen vertex has the number 1.
- If we add a single vertex 2 to the chosen vertex, then if the chosen vertex is either 1 or 2, the game should end in a draw. If vertex number 3 is chosen, then the first player wins.
- If we place three tokens in vertex 1 and two tokens in vertex 3, then the game will end in a draw only if vertex 2 is chosen. If the professor chose vertex 3, the first player will win, if the professor chose vertex 1, then the second player will win.

In the first test, the interactor will behave as if the chosen vertices are the same as those in the example above. However, if you will try to guess the answer before it limits the options to one single vertex, the solution will get "`Wrong Answer`", even if you print the same answers. That's because the interactor is allowed to change the chosen vertex if it's consistent with the previous query answers.

F. Matching

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a connected planar graph and the coordinates of all its vertices. Check if it has exactly one perfect matching.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 100\,000$), denoting the number of test cases, followed by a description of the test cases.

The first line of each test case contains two integers n, m ($2 \le n \le 200\,000; n - 1 \le m \le 3n$): the number of vertices and edges in the graph.

Each of the next n lines contains pairs of integers x_i, y_i ($-10^6 \le x_i, y_i \le 10^6$): coordinates of vertices. All vertices are guaranteed to be distinct. Next m lines of each test case contains two integers u_j, v_j ($1 \le u_j, v_j \le n$): graph edges.

The graph is guaranteed to be connected and has no loops or multi-edges.

The sum of all n among all test cases is guaranteed to not exceed $200\,000$.

Output

For each test case, print 1 if the graph has exactly one perfect matching, or 0 if it does not.

Example

input
4 3 3 0 0 1 1 2 4 1 2 2 3 1 3 4 3 0 1 1 0 2 0 3 1 1 2 2 3

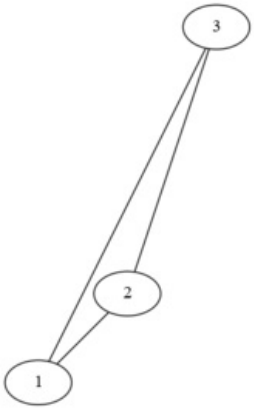
```
3 4
4 4
0 1
1 0
2 0
3 1
1 2
2 3
3 4
1 4
6 7
-2 1
-2 -1
-1 0
1 0
2 1
2 -1
1 2
1 3
2 3
3 4
4 5
4 6
5 6
```

output

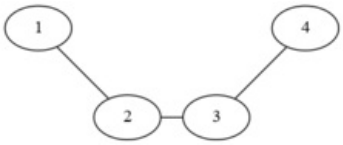
```
0
1
0
1
```

Note

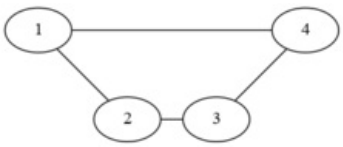
An illustration for the first test case:



An illustration for the second test case:



An illustration for the third test case:



An illustration for the fourth test case: