Make It Zero 2

- SUBMIT
- ALL SUBMISSIONS

Problem code: LMATRIX2

Aleksandra is the most popular girl in the city. Each boy can only dream about dating her. Other girls want to be like her.

Aleksandra is going to find herself a boyfriend. Of course such a beauty needs someone special. That's why she is going to announce a quiz. Even you can try your chances at this. Apart from boys, the girl also loves math. That's why this quiz is going to be mathematical.

She has two matrixes with **N** rows and **M** columns each. Let $P_{i,j}$ and $A_{i,j}$ be the j^{th} element of the i^{th} row of the first and second matrixes respectively. She likes **0**, that's why she is going to get rid of all non zero numbers in the second matrix. In each turn she may choose five integers:

- $1 \le x_1 \le x_2 \le N$
- $1 \le y_1 \le y_2 \le M$
- $0 \le k \le 10000$

and after this, for all pairs **(i,j)** such that:

- $x_1 \le i \le x_2$
- $y_1 \le j \le y_2$

she does the following:

- $A_{i,j} = (k + A_{i,j})$ **modulo** $P_{i,j}$

It takes her a lot of turns to do this. That's why she gives this problem to all boys in the city and promises to go for a date with the one who will solve this task with the fewest number of moves.

It is your chance to walk with such a wonderful girl. Just use it!

Input

The first line contains two space-separated integers **N** and **M**, denoting the size of the matrix. Then the i^{th} line of the next **N** lines contains **M** space-separated integers $P_{i,1}$, $P_{i,2}$, ..., $P_{i,M}$. After matrix **P**, matrix **A** is given in the same format.

Output

In the first line, print the integer **Q (0 ≤ Q ≤ N * M)**, denoting the number of the moves for erasing all non zero numbers from the second matrix. In the i^{th} line of the next **Q** lines, print **5** space-separated integers x_1, y_1, x_2, y_2, **k**, denoting the information about submatrix chosen at the i^{th} move.

Note: Your solution will be judged as *wrong answer* if **Q** is larger then **N*M**.

Constraints for the official judge data

- **66 ≤ N, M ≤ 99**
- **1 ≤ $P_{i,j}$ ≤ 10**
- **0 ≤ $A_{i,j}$ < $P_{i,j}$**

Example

Input:

2 2
1 2
2 3
0 1
1 1

Output:

2
1 1 2 2 1
2 2 2 2 1

Scoring

For each test file, your score is calculated as **100 × Q / (N × M)**. Then the total your score is defined as the average of your score for the test files (see the next paragraph, for more details). The aim for this problem is to minimize the total your score.

We have **40** official test files. You must correctly solve all test files to receive *accepted*. During the contest, the scores of the last **30** test files are **0**, that is, the total your score is calculated by only the first**10** tests. After the contest, all solutions will be rescored by the average of the scores of the all **40** test files, and it will be the final score.

Test generation

In the official test file, **N** and **M** are choosing uniformly and randomly from **[66, 99]**. Each **$P_{i,j}$** is chosen randomly from **1** to **10** inclusive. Then each **$A_{i,j}$** is chosen randomly from **[0, $P_{i,j}$ − 1]**.

Little Elephant and Movies

- SUBMIT
- ALL SUBMISSIONS

Problem code: LEMOVIE

Read problems statements in Mandarin Chinese and Russian.

Little Elephant from Zoo of Lviv likes to watch movies.

There are **N** different movies (numbered from **0** to **N − 1**) he wants to watch in some order. Of course, he will watch each movie exactly once. The priority of i^{th} movie is P_i.

A watching of a movie is called *exciting* if and only if one of the following two conditions holds:

- This is the first watching.
- The priority of this movie is strictly greater than the maximal priority of the movies watched so far.

Let us call the number of *exciting* watchings the *excitingness* of the order.

Help him to find the number of different watching orders whose *excitingness* does not exceed **K**. Since the answer can be large, print it modulo **1000000007 (10^9+7)**.

Input

The first line of input contains an integer **T**, denoting the number of test cases. Then **T** test cases follow.

The first line of each test case contains two space-separated integers **N** and **K**. The next line contains **N**space-separated integers P_1, P_2, ..., P_N.

Output

For each test case, print the number of different watching orders having at most **K** *excitingness* modulo**1000000007 (10^9+7)**.

Constraints

- **1 ≤ T ≤ 10**
- **1 ≤ K ≤ N ≤ 200**
- **1 ≤ P_i ≤ 200**

Example

Input:

```
2
3 1
3 1 2
4 3
1 2 2 3
```

Output:

2
24

Explanation

In the first case, there are two boring watching orders whose *excitingness* not greater than **K=1**: **[3, 1, 2]**,**[3, 2, 1]**. Both watching orders have one *exciting* watching: the first watching.

In the second case, every watching order has at most **3** *excitingness*.

Chef and Favourite Sequence

- SUBMIT
- ALL SUBMISSIONS

Problem code: CHSEQ22

Read problems statements in Mandarin Chinese and Russian.

Chef has an integer sequence $a_1, a_2, ..., a_N$ of size **N**, where all the elements of the sequence are **0**initially. Chef also has **M** segments, here the i^{th} one is $[L_i, R_i]$. He wants to create new sequences using the following operation:

- In a single operation, he picks a segment from the **M** segments. Let the chosen segment be **[s, t]**.
- Then *flip* the all elements from s^{th} to t^{th} elements, namely, a_i is changed to $1 - a_i$ for all $s \le i \le t$.

Chef can perform the operation as many times as Chef likes. Chef wants to know how many different sequences Chef can make. Since the answer can be very large, please print it modulo **1000000007 (10^9+7)**.

Input

The first line contains two space-separated integers **N** and **M**, denoting the size of the sequence and the number of the segments respectively. Then the i^{th} line of the next **M** lines contains two space-separated integers L_i and R_i, denoting the i^{th} segments.

Output

Print a single line containing one integer, denoting the number of the sequences which can be created by Chef, modulo **1000000007 (10^9+7)**.

Constraints

- $1 \le N, M \le 100000$ (10^5)
- $1 \le L_i \le R_i \le N$

Example

Input:

3 3
1 1
2 2
3 3

Output:

8

Explanation

In the example case, all the sequences **(0,0,0)**, **(0,0,1)**, **(0,1,0)**, **(1,0,0)**, **(0,1,1)**, **(1,0,1)**, **(1,1,0)**, **(1,1,1)** can be created by Chef.

Graph Challenge

- SUBMIT
- ALL SUBMISSIONS

Problem code: DAGCH

Read problems statements in Mandarin Chinese and Russian.

Chef Hawlader and Chef Heickal are two good friends. Apart from cooking they also love algorithms. Chef Hawlader loves graph theory and Chef Heickal loves number theory.

Once again, Chef Hawlader is giving lecture to Chef Heickal about graph theory. "Hey Heickal, you should concentrate more on graph theory research. What remains in number theory? Graph is everything in the world, you can convert every problem to graph and solve them easily. Even your number theory problems can be converted and solved with graph theory, you know?" says Chef Hawlader. Though Chef Heickal does not hate graph, but according to his theory the life should a mixture of dynamic programming, number theory, data structure, ad hoc, graph theory, and so on. To become a good Chef you have to know all of them not just graph theory.

- "So are you master of graph? Do you know about DFS?" asks Chef Heickal.
- "Yes, of course. DFS, in full form Depth First Search, is a traversing or searching algorithm on graph. It's a basic algorithm", Chef Hawlader replied with angriness in his face.
- "Then, for a given graph, can you number each vertex using DFS in pre-order?" asks Chef Heickal again.
- "Too easy! It will be done by the following pseudo code:"

```
int C = 1;
void DFS(int u)
{
        new_number[u] = C;
        C++;

        // initially all value of visited array is set to false
        visited[u] = true;

        // here v can be chosen in an arbitrary order
        for each v such that there is a edge from u to v
            if(visited[v] == false)
                DFS(v);
}
```

- "Okey, I can give you a hard problem", says Chef Heickal with evil smile.
- "What? Come on", replies Chef Hawlader with grinning face.
- "I'll give you a directed graph with **N** vertices and **M** edges. Here each vertex of the graph is numbered by DFS in pre-order. Of course, every vertex other than vertex **1** are reachable from vertex**1**."
- "Okey, so what's problem?"

- "Wait, I give some definitions. A vertex **x** is a *supreme* vertex of **y**, if there is a directed path $x = v_0, v_1, ..., v_k = y$, where $x < y < v_i$ for all $0 < i < k$, that is a path from **x** to **y** with zero or more intermediate vertex v_i and every intermediate vertex should be greater than **x** and **y** and **x** should be less than **y**. A vertex **v** is a *superior* vertex of **w**, if **v** is the minimum numbered vertex among all the *supreme*vertices of **w**. You will be given **Q** queries on the graph. On each query you will be given a vertex **v**, you have to find out how many vertices are there in the graph, which have vertex **v** as superior vertex."
- "Oh!! Duh!!"

Chef Hawlader could not solve the problem, he is asking you to solve it for him.

Input

The first line of the input contains an integer **T**, denoting the number of test cases. Then **T** test cases follow.

The first line of each test contains three space-separated integers **N**, **M**, and **Q**. The i^{th} line of the next **M**lines contains two space-separated integers U_i and V_i denoting the directed edge from vertex U_i to vertexV_i. The next line will contain **Q** space separated integers P_1, P_2, ..., P_Q, denoting the queries defined above.

Output

For each test case, print a single line containing **Q** space-separated integers, denoting the answer of the **Q** queries according to the order of the input. If there is no superior vertex for a query, print **0**. See the sample input and output for detailed format.

Constraints

- $1 \le T \le 10$
- $2 \le N \le 100000 (10^5)$
- $N - 1 \le M \le 200000 (2 \times 10^5)$
- $1 \le Q \le 100000 (10^5)$
- $1 \le U_i, V_i \le N$, $U_i \ne V_i$
- If $i \ne j$, then $(U_i, V_i) \ne (U_j, V_j)$
- $1 \le P_i \le N$
- In the given graph, the vertices **2**, **3**, ..., **N** are reachable from vertex **1**
- The vertices of the graph is numbered by DFS, denoted by the pseudo code in the statement

Example

Input:

```
2
3 3 3
1 2
1 3
```

3 2
1 2 3
8 9 8
1 2
1 7
2 3
2 5
3 4
5 6
7 8
6 4
8 4
1 2 3 4 5 6 7 8

Output:

2 0 0
3 2 0 0 1 0 1 0

Explanation

Example case 1. Vertex **3** has only vertex **1** as *supreme* vertex as well as *superior* vertex because there is a direct edge from **1** to **3** and vertex **2** has a *supreme* vertex **1** (**1** → **3** → **2**). As vertex **2** has only one*supreme* vertex, it is the *superior* vertex also.

Example case 2. Vertex **4** has two *supreme* vertices **1** (**1** → **7** → **8** → **4**) and **2** (**2** → **5** → **6** → **4**). So vertex**1** is the *superior* vertex of **4**.

Count on a Treap

- SUBMIT
- ALL SUBMISSIONS

Problem code: COT5

Read problems statements in Mandarin Chinese and Russian.

In computer science, a treap is a binary search tree according to the keys and meanwhile a heap according to the weights. Follow the link for more details: http://en.wikipedia.org/wiki/Treap

Your task is to maintain a **max-treap** supporting the following operations:

- 0 **k w**: Insert a new node, whose key and weight are **k** and **w**.
- 1 **k**: Delete a node in the treap with key **k**.
- 2 **ku kv**: Return the distance between node **u** whose key is **ku** and node **v** whose key is **kv**.

No two nodes share the same key or the same weight in the tree, and we guarantee the node is indeed in the treap before a delete operation takes place.

Input

The first line contains an integer **N**, the number of operations.
Each of the next **N** lines contains two or three integers "**0 k w**" "**1 k**" or "**2 ku kv**" . These integers describe the current operation.

Output

For each distance query, print the distance between **u** and **v**.

Constraints

- **1 ≤ N ≤ 200000**
- **0 < k, w, ku, kv < 2^{32}**

Example

Input:

12
0 4 17535
0 10 38964
0 2 21626

0 1 61321
2 1 10
2 10 4
1 10
1 1
0 8 42634
2 8 4
2 8 2
2 4 2

Output:
1
2
2
1
1