A. Milya and Two Arrays

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

An array is called *good* if for any element x that appears in this array, it holds that x appears at least twice in this array. For example, the arrays $[1, 2, 1, 1, 2]$, $[3, 3]$, and $[1, 2, 4, 1, 2, 4]$ are good, while the arrays $[1]$, $[1, 2, 1]$, and $[2, 3, 4, 4]$ are not good.

Milya has two **good** arrays a and b of length n. She can rearrange the elements in array a in any way. After that, she obtains an array c of length n, where $c_i = a_i + b_i$ $(1 \le i \le n)$.

Determine whether Milya can rearrange the elements in array a such that there are **at least** 3 distinct elements in array c.

Input

Each test consists of multiple test cases. The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n $(3 \le n \le 50)$ — the length of the arrays a and b.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$ — the elements of the array a.

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ $(1 \le b_i \le 10^9)$ — the elements of the array b.

Output

For each test case, output «YES» (without quotes) if it is possible to obtain at least 3 distinct elements in array c, and «NO» otherwise.

You can output each letter in any case (for example, «YES», «Yes», «yes», «yEs» will be recognized as a positive answer).

Standard Input	Standard Output
5 4 1 2 1 2 1 2 1 2 6 1 2 3 3 2 1 1 1 1 1 1 1 3 1 1 1 1 1 1 6 1 52 52 3 1 3 59 4 3 59 3 4 4 100 1 100 1	YES YES NO YES NO

2 2 2 2	

Note

In the first test case, you can swap the second and third elements. Then the array $a = [1, 1, 2, 2]$, $b = [1, 2, 1, 2]$, and then $c = [2, 3, 3, 4]$.

In the second test case, you can leave the elements unchanged. Then $c = [2, 3, 4, 4, 3, 2]$.

In the third test case, the array a will not change from rearranging the elements in it. Then $c = [2, 2, 2]$, so the answer is «NO».

,

B. Cost of the Array

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

You are given an array a of length n and an **even** integer k ($2 \le k \le n$). You need to split the array a into exactly k non-empty subarrays[†] such that each element of the array a belongs to exactly one subarray.

Next, all subarrays with even indices (second, fourth, ..., k-th) are concatenated into a single array b. After that, 0 is **added** to the end of the array b.

The *cost* of the array b is defined as the minimum index i such that $b_i \neq i$. For example, the cost of the array $b = [1, 2, 4, 5, 0]$ is 3, since $b_1 = 1$, $b_2 = 2$, and $b_3 \neq 3$. Determine **the minimum** cost of the array b that can be obtained with an optimal partitioning of the array a into subarrays.

[†] An array x is a subarray of an array y if x can be obtained from y by the deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and k ($2 \le k \le n \le 2 \cdot 10^5$, k is even) — the length of the array a and the number of subarrays.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the array a.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the minimum cost of the array b that can be obtained.

Standard Input	Standard Output
4 3 2 1 1 1 8 8 1 1 2 2 3 3 4 4 5 4 1 1 1 2 2 5 4 1 1 1000000000 2 2	2 5 2 1

Note

In the first test case, there are only two possible partitionings: $[[1], [1, 1]]$ and $[[1, 1], [1]]$. In either case, $b_1 = 1$, and $b_2 \neq 2$, so the cost is 2.

In the second test case, there is only one possible partitioning, where $b = [1, 2, 3, 4, 0]$, so the cost is 5 ($b_5 = 0 \neq 5$).

In the third test case, the following partitioning works: $[[1], [1, 1], [2], [2]]$. Then $b = [1, 1, 2, 0]$, and the cost is 2.

,

C. Customer Service

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

Nikyr has started working as a queue manager at the company "Black Contour." He needs to choose the order of servicing customers. There are a total of n queues, each initially containing 0 people. In each of the next n moments of time, there are **two sequential** events:

1. New customers arrive in all queues. More formally, at the j-th moment of time, the number of people in the i-th queue increases by a **positive** integer $a_{i,j}$.
2. Nikyr chooses **exactly one** of the n queues to be served at that moment in time. The number of customers in this queue becomes 0.

Let the number of people in the i-th queue after all events be x_i. Nikyr wants MEX[†] of the collection $x_1, x_2, \ldots, x_n$ to be as large as possible. Help him determine the maximum value he can achieve with an optimal order of servicing the queues.

[†] The minimum excluded (MEX) of a collection of integers $c_1, c_2, \ldots, c_k$ is defined as the smallest non-negative integer y which does not occur in the collection c.

For example:

- $\text{MEX}([2, 2, 1]) = 0$, since 0 does not belong to the array.
- $\text{MEX}([3, 1, 0, 1]) = 2$, since 0 and 1 belong to the array, but 2 does not.
- $\text{MEX}([0, 3, 1, 2]) = 4$, since 0, 1, 2, and 3 belong to the array, but 4 does not.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 300$) — the number of queues and moments of time.

The i-th of the next n lines contains n integers $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ ($1 \le a_{i,j} \le 10^9$) — the number of new customers in the i-th queue at each moment of time.

It is guaranteed that the sum of n^2 over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the maximum value of $\text{MEX}([x_1, x_2, \ldots, x_n])$ that can be achieved.

Standard Input	Standard Output
4 2 1 2 2 1 2	2 1 3 3

10 10 10 10 3 2 3 3 4 4 1 2 1 1 4 4 2 2 17 1 9 3 1 5 5 5 11 1 2 1 1	

Note

In the first test case, the second queue can be served at time 1, and the first queue at time 2. There will be $x_1 = 0$ people left in the first queue and $x_2 = 1$ person left in the second queue. Therefore, the answer is $\text{MEX}([0, 1]) = 2$.

In the second test case, the first queue can be served both times. There will be $x_1 = 0$ people left in the first queue and $x_2 = 20$ people left in the second queue. Therefore, the answer is $\text{MEX}([0, 20]) = 1$.

In the third test case, the third queue can be served at time 1, the second queue at time 2, and the first queue at time 3. There will be $x_1 = 0$ people left in the first queue, $x_2 = 1$ person left in the second queue, and $x_3 = 2$ people left in the third queue. Therefore, the answer is $\text{MEX}([0, 1, 2]) = 3$.

,

D. Graph and Graph

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

You are given two connected undirected graphs with the same number of vertices. In both graphs, there is a token located at some vertex. In the first graph, the token is initially at vertex s_1, and in the second graph, the token is initially at vertex s_2. The following operation is repeated an **infinite** number of times:

- Let the token currently be at vertex v_1 in the first graph and at vertex v_2 in the second graph.
- A vertex u_1, adjacent to v_1, is chosen in the first graph.
- A vertex u_2, adjacent to v_2, is chosen in the second graph.
- The tokens are moved to the chosen vertices: in the first graph, the token moves from v_1 to u_1, and in the second graph, from v_2 to u_2.
- The cost of such an operation is equal to $|u_1 - u_2|$.

Determine the minimum possible total cost of all operations or report that this value will be infinitely large.

Input

Each test consists of multiple test cases. The first line contains one integer t ($1 \le t \le 500$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains three integers n, s_1, and s_2 ($2 \le n \le 1000$, $1 \le s_1, s_2 \le n$) — the number of vertices in each graph, the number of the vertex in the first graph where the token is initially located, and the number of the vertex in the second graph where the token is initially located.

The second line of each test case contains one integer m_1 ($1 \le m_1 \le 1000$) — the number of edges in the first graph.

The i-th of the following m_1 lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \neq b_i$) — the numbers of the endpoints of the i-th edge in the first graph.

The next line of each test case contains one integer m_2 ($1 \le m_2 \le 1000$) — the number of edges in the second graph.

The j-th of the following m_2 lines contains two integers c_j and d_j ($1 \le c_j, d_j \le n$, $c_j \neq d_j$) — the numbers of the endpoints of the j-th edge in the second graph.

It is guaranteed that the sum of n, the sum of m_1, and the sum of m_2 over all test cases do not exceed 1000.

It is guaranteed that both graphs are connected.

Output

For each test case, output one integer — the minimum total cost of all operations or -1, if this value will be infinitely large.

Standard Input	Standard Output
3 4 1 1 4 1 2	0 -1 7

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

Note

In the first test case, an infinite sequence of transitions can be constructed to the vertices $2, 3, 4, 1, 2, 3, 4, 1, \ldots$, along which the token can move in both the first and the second graphs.

In the second test case, it can be proven that the cost of any operation will be greater than 0; therefore, the total cost of all operations will be infinitely large.

,

E1. Stop Gaming (Easy Version)

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

This is the easy version of the problem. The difference between the versions is that in this version you only need to find the minimum number of operations. You can hack only if you solved all versions of this problem.

You are given n arrays, each of which has a length of m. Let the j-th element of the i-th array be denoted as $a_{i,j}$. It is guaranteed that all $a_{i,j}$ are **pairwise distinct**. In one operation, you can do the following:

- Choose some integer i ($1 \le i \le n$) and an integer x ($1 \le x \le 2 \cdot n \cdot m$).
- For all integers k from i to n in increasing order, do the following:
    1. Add the element x to the beginning of the k-th array.
    2. Assign x the value of the last element in the k-th array.
    3. Remove the last element from the k-th array.

In other words, you can insert an element at the beginning of any array, after which all elements in this and all following arrays are shifted by one to the right. The last element of the last array is removed.

You are also given a description of the arrays that need to be obtained after all operations. That is, after performing the operations, the j-th element of the i-th array should be equal to $b_{i,j}$. It is guaranteed that all $b_{i,j}$ are **pairwise distinct**.

Determine the minimum number of operations that need to be performed to obtain the desired arrays.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 3 \cdot 10^5$) — the number of arrays and the number of elements in each array.

The i-th of the following n lines contains m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ ($1 \le a_{i,j} \le 2 \cdot n \cdot m$) — the elements of the i-th original array. It is guaranteed that all $a_{i,j}$ are pairwise distinct.

The i-th of the following n lines contains m integers $b_{i,1}, b_{i,2}, \ldots, b_{i,m}$ ($1 \le b_{i,j} \le 2 \cdot n \cdot m$) — the elements of the i-th final array. It is guaranteed that all $b_{i,j}$ are pairwise distinct.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, output a single integer — the minimum number of operations that need to be performed.

Standard Input	Standard Output

4 2 2 2 6 3 4 1 2 7 8 1 5 5 4 1 2 3 5 4 3 2 1 3 3 1 2 3 4 5 6 7 8 9 11 1 2 12 3 4 13 5 6 4 4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 1 2 3 4 18 5 6 7 19 8 20 9 21 22 10	3 5 3 6

Note

In the first test case, the following sequence of 3 operations is suitable:

- Apply the operation to the first array with $x = 1$. Then the element 1 will be added to the beginning of the first array, and the value of x will become 6. The last element will be removed, and the first array will look like $[1, 2]$. Next, the element x is added to the beginning of the second array, and the value of x becomes 4. The last element of the second array is removed, and both arrays look like $[1, 2]$ and $[6, 3]$ respectively after the first operation.
- Apply the operation to the second array with $x = 8$. Then the first array remains unchanged, and both arrays will look like $[1, 2]$ and $[8, 6]$ respectively.
- Apply the operation to the second array with $x = 7$, then both arrays will have the required appearance $[1, 2]$ and $[7, 8]$ respectively.

In the second test case, the desired array can only be achieved in 5 operations.

In the third test case, the following sequence of 3 operations is suitable:

- Apply the operation with $x = 11$ to the first array.
- Apply the operation with $x = 12$ to the second array.
- Apply the operation with $x = 13$ to the third array.

,

E2. Stop Gaming (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

This is the hard version of the problem. The difference between the versions is that in this version you need to output all the operations that need to be performed. You can hack only if you solved all versions of this problem.

You are given n arrays, each of which has a length of m. Let the j-th element of the i-th array be denoted as $a_{i,j}$. It is guaranteed that all $a_{i,j}$ are **pairwise distinct**. In one operation, you can do the following:

- Choose some integer i ($1 \le i \le n$) and an integer x ($1 \le x \le 2 \cdot n \cdot m$).
- For all integers k from i to n in increasing order, do the following:
    1. Add the element x to the beginning of the k-th array.
    2. Assign x the value of the last element in the k-th array.
    3. Remove the last element from the k-th array.

In other words, you can insert an element at the beginning of any array, after which all elements in this and all following arrays are shifted by one to the right. The last element of the last array is removed.

You are also given a description of the arrays that need to be obtained after all operations. That is, after performing the operations, the j-th element of the i-th array should be equal to $b_{i,j}$. It is guaranteed that all $b_{i,j}$ are **pairwise distinct**.

Determine the minimum number of operations that need to be performed to obtain the desired arrays, and also output the sequence of all operations itself.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 3 \cdot 10^5$) — the number of arrays and the number of elements in each array.

The i-th of the following n lines contains m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ ($1 \le a_{i,j} \le 2 \cdot n \cdot m$) — the elements of the i-th original array. It is guaranteed that all $a_{i,j}$ are pairwise distinct.

The i-th of the following n lines contains m integers $b_{i,1}, b_{i,2}, \ldots, b_{i,m}$ ($1 \le b_{i,j} \le 2 \cdot n \cdot m$) — the elements of the i-th final array. It is guaranteed that all $b_{i,j}$ are pairwise distinct.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, output a single integer — the minimum number of operations that need to be performed.

Next, for each operation output two integers i and x ($1 \le i \le n$, $1 \le x \le 2 \cdot n \cdot m$) — the index of the array where the element is inserted and the value of the element, respectively.

If there are multiple possible sequences with the minimum number of operations, output any of them.

Standard Input	Standard Output
4 2 2 2 6 3 4 1 2 7 8 1 5 5 4 1 2 3 5 4 3 2 1 3 3 1 2 3 4 5 6 7 8 9 11 1 2 12 3 4 13 5 6 4 4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 1 2 3 4 18 5 6 7 19 8 20 9 21 22 10	3 1 1 2 8 2 7 5 1 1 1 2 1 3 1 4 1 5 3 1 11 2 12 3 13 6 3 20 2 18 3 19 4 22 4 21 1 17

Note

In the first test case, the following sequence of 3 operations is suitable:

- Apply the operation to the first array with $x = 1$. Then the element 1 will be added to the beginning of the first array, and the value of x will become 6. The last element will be removed, and the first array will look like $[1, 2]$. Next, the element x is added to the beginning of the second array, and the value of x becomes 4. The last element of the second array is removed, and both arrays look like $[1, 2]$ and $[6, 3]$ respectively after the first operation.
- Apply the operation to the second array with $x = 8$. Then the first array remains unchanged, and both arrays will look like $[1, 2]$ and $[8, 6]$ respectively.
- Apply the operation to the second array with $x = 7$, then both arrays will have the required appearance $[1, 2]$ and $[7, 8]$ respectively.

In the second test case, the desired array can only be achieved in 5 operations.

In the third test case, the following sequence of 3 operations is suitable:

- Apply the operation with $x = 11$ to the first array.
- Apply the operation with $x = 12$ to the second array.
- Apply the operation with $x = 13$ to the third array.