O banco que você trabalha sempre tem problemas para organizar as filas de atendimento dos clientes.

Após uma reunião com a gerência ficou decidido que os clientes ao chegar na agência receberão uma senha numérica em seu aparelho de celular via sms e que a ordem da fila será dada não pela ordem de chegada, mas sim pelo número recebido via sms. A ordem de atendimento será decrescente: aqueles que receberam número maior deverão ser atendidos primeiro.

Então, dada a ordem de chegada dos clientes reordene a fila de acordo com o número recebido via sms, e diga quantos clientes não precisaram trocar de lugar nessa reordenação.

**Entrada**

A primeira linha contém um inteiro **N**, indicando o número de casos de teste a seguir.

Cada caso de teste inicia com um inteiro **M** (1 ≤ **M** ≤ 1000), indicando o número de clientes. Em seguida haverá **M** inteiros distintos **Pi** (1 ≤ **Pi** ≤ 1000), onde o **i**-ésimo inteiro indica o número recebido via sms do **i**-ésimo cliente.

Os inteiros acima são dados em ordem de chegada, ou seja, o primeiro inteiro diz respeito ao primeiro cliente a chegar na fila, o segundo inteiro diz respeito ao segundo cliente, e assim sucessivamente.

**Saída**

Para cada caso de teste imprima uma linha, contendo um inteiro, indicando o número de clientes que não precisaram trocar de lugar mesmo após a fila ser reordenada.

| Exemplo de Entrada | Exemplo de Saída |
| --- | --- |
| 3 3 100 80 90 4 100 120 30 50 4 100 90 30 25 | 1 0 4 |

***Solução***

import java.io.IOException;

import java.util.\*;

public class BankQueue {

private Integer smsNumber;

public BankQueue(int smsNumber) {

this.smsNumber = smsNumber;

}

public int getSmsNumber() {

return this.smsNumber;

}

public static void main(String[] args) throws IOException {

Scanner scanner = new Scanner(System.in);

int N = scanner.nextInt();

scanner.nextLine();

int peopleOnQueue;

ArrayList<Integer> currentQueue = new ArrayList<>();

HashMap<Integer, Integer> numToIndexOriginal = new HashMap<>();

int currentSmsNumber;

int numSwitches;

for(int i = 0; i < N; i++) {

peopleOnQueue = scanner.nextInt();

numSwitches = 0;

for(int j = 0; j < peopleOnQueue; j++) {

currentSmsNumber = scanner.nextInt();

currentQueue.add(currentSmsNumber);

numToIndexOriginal.put(currentSmsNumber, currentQueue.size() - 1);

}

SmsNumberComparator comparator = new SmsNumberComparator();

Collections.sort(currentQueue, comparator);

for(int j = 0; j < currentQueue.size(); j++) {

int originalIndex = numToIndexOriginal.get(currentQueue.get(j));

if(originalIndex != j) {

numSwitches++;

}

}

System.out.println(peopleOnQueue - numSwitches);

currentQueue = new ArrayList<>();

}

scanner.close();

}

static class SmsNumberComparator implements Comparator<Integer> {

public SmsNumberComparator() {}

@Override

public int compare(Integer num0, Integer num1) {

int res = num0.compareTo(num1);

return -res;

}

}

}