

ICPC ASIA DHAKA REGIONAL CONTEST 2021

Hosted by



Bangladesh University of Business and Technology

In association with



October 08, 2022

You get 17 Pages, 11 Problems & 300 Minutes

Supported By



Our Sponsors



Media Partner



In Association With





Problem A

Beautiful Blocks Again!



Alice likes playing a game on an $N \times M$ board where each cell contains some points. Rows are numbered from **1** to **N** from top to bottom, and columns are numbered from **1** to **M** from left to right. She starts from the **(1, 1)** cell and ends her journey on the **(N, M)** cell. But on each move, she can only move in the right or down cell. When she is in a cell, she collects all the points in that cell. Alice loves points a lot. So she always moves in a way that the total number of points she can collect is maximized.

Bob has an **empty** $N \times M$ board which is placed **vertically**. He also has $N \times M$ number of **(1 x 1)** blocks and each block has some points. The blocks fall from **above the board one by one** and he can decide which block falls into which columns. **A block falls as long as it does not hit another block or if it hits the bottom of the board. He also does not make a block fall into a full column (a column that already has N blocks).**

Bob fills the empty board with all blocks in the above-mentioned manner and then gives the filled board to Alice. He wants to make Alice happy as much as possible. **So he fills the empty board with the blocks so that Alice can get the maximum number of points.**

Can you find out how Bob will fill the board and how many points Alice will get?

Input

The first line will contain a single integer **T** ($1 \leq T \leq 10^6$) denoting the number of test cases. In each test case, the first line will have two space-separated integers **N, M** ($1 \leq N \times M \leq 10^5$) denoting the number of rows and columns respectively. In the second line, there will be $N \times M$ space-separated integers **P₁, P₂, ..., P_{NxM}** ($0 \leq P_i \leq 10^9$) denoting the points of the blocks. **The blocks fall in the order given in the input.**

The sum of $N \times M$ over all test cases does not exceed 10^6 .

Output

For each test case, in the first line, print the maximum number of points Alice will get. In the second line, print $N \times M$ space-separated integers, **C₁, C₂, ..., C_{NxM}** ($1 \leq C_i \leq M$) where **C_i** denotes the column Bob chooses for the **ith** block to fall. Please see the sample for details.

Note that, there might be multiple optimal ways of falling the blocks. You can output any valid optimal solution.

Sample Input

2 2 2 4 5 5 4 2 1 10 9	14 2 1 1 2 19 1 1
------------------------------------	----------------------------

Explanation: For the 1st sample test, initially 2×2 board is placed vertically. Blocks will fall into it one by one maintaining the given input order. An example of filling the board with blocks so that Alice can maximize her points is shown below.

<p><i>Step - 0 (Board is empty):</i></p> <div style="display: flex; justify-content: space-around; margin: 10px 0;"> <div style="border: 1px solid black; padding: 5px; text-align: center;">4</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">5</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">5</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">4</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td></tr> <tr><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td></tr> </table> </div>					<p><i>Step - 1:</i></p> <div style="display: flex; justify-content: space-around; margin: 10px 0;"> <div style="border: 1px solid black; padding: 5px; text-align: center;">5</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">5</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">4</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td></tr> <tr><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px; background-color: #cccccc;">4</td></tr> </table> </div> <p>$C_1 = 2$</p>				4
	4								

<p><i>Step - 2:</i></p> <div style="display: flex; justify-content: space-around; margin: 10px 0;"> <div style="border: 1px solid black; padding: 5px; text-align: center;">5</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">4</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td></tr> <tr><td style="width: 20px; height: 20px; background-color: #cccccc;">5</td><td style="width: 20px; height: 20px; background-color: #cccccc;">4</td></tr> </table> </div> <p>$C_2 = 1$</p>			5	4	<p><i>Step - 3:</i></p> <div style="display: flex; justify-content: space-around; margin: 10px 0;"> <div style="border: 1px solid black; padding: 5px; text-align: center;">4</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="width: 20px; height: 20px; background-color: #cccccc;">5</td><td style="width: 20px; height: 20px;"></td></tr> <tr><td style="width: 20px; height: 20px; background-color: #cccccc;">5</td><td style="width: 20px; height: 20px; background-color: #cccccc;">4</td></tr> </table> </div> <p>$C_3 = 1$</p>	5		5	4	<p><i>Step - 4:</i></p> <div style="display: flex; justify-content: space-around; margin: 10px 0;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="width: 20px; height: 20px; background-color: #cccccc;">5</td><td style="width: 20px; height: 20px; background-color: #cccccc;">4</td></tr> <tr><td style="width: 20px; height: 20px; background-color: #cccccc;">5</td><td style="width: 20px; height: 20px; background-color: #cccccc;">4</td></tr> </table> </div> <p>$C_3 = 2$</p>	5	4	5	4
5	4													
5														
5	4													
5	4													
5	4													

One possible optimal path visited by Alice is shown below (visited cells in bold).

5	4
5	4

So the maximum number of points obtained by Alice = $5 + 5 + 4 = 14$.



Problem B

Black Bishop



We have an $n \times n$ sized checkerboard where every cell of the board is colored either black or white so that no two cells of the same color are adjacent. The board size, n , will always be **odd** and all the corner cells of the board are black.

We have m identical bishops at the black cells on the board, where m is at most n . We are given the initial and target configuration of the bishops. You have to move the bishops from the initial configuration to the target configuration in at most $4 \times m$ moves.

In a move a bishop can only move diagonally but not over another bishop. Two bishops can never be in the same cell at the same time.

Input

The input begins with the number of cases t ($1 \leq t \leq 100$). Then t test cases follow. Every case starts with n ($1 \leq n \leq 99,999$) and m ($1 \leq m \leq n$) and n is odd. Then follows m lines describing the initial bishop configuration. Next m lines describe the target configuration. Each line of the configuration contains two integers r, c ($1 \leq r, c \leq n$) where they denote the row and column number of the cell a bishop is at.

Sum of n across all cases will not exceed **100,000**.

No two bishops in the initial configurations will be in the same cell. Same goes for the target configuration.

Output

For each case, output the case number and the number of moves denoted by k . Next k lines contain 4 integers r_1, c_1, r_2, c_2 ($1 \leq r_1, c_1, r_2, c_2 \leq n$) meaning a bishop is moved from r_1, c_1 position to r_2, c_2 position (They can be same cell as well). Check the Sample input/output for further clarity. Also please be careful with the output format. Unintended whitespaces might cause a wrong answer.

Sample Input

```
2
5 2
1 3
5 3
3 1
3 5
1 1
1 1
1 1
```

Output for Sample Input

```
Case 1: 2
1 3 3 5
5 3 3 1
Case 2: 0
```



Problem C

Codovid Virus



The world is stunned by the CodoVid virus, which is believed to be one of the deadliest viruses the world has ever seen. Due to its self-reproduction mechanism, the DNA sequence of the CodoVid virus grows at a high speed. The DNA sequence of CodoVid virus is a binary string consisting of **0's** and **1's** only. A codovid virus has an initial DNA sequence S_1 . Starting from day 2, each day codovid virus reproduces a new sequence and adds the new sequence to its DNA sequence.

The sequence of DNA added on the i^{th} day, $S_i = \text{Reverse}(S_{i-1}) + \text{Flip}(S_{i-1}) + \text{Reverse}(S_{i-1})$,

- o The “+” symbol represents *string concatenation* operation.
- o $\text{Reverse}()$ function returns the characters in reverse order. i.e. $\text{Reverse}(1100) = 0011$
- o $\text{Flip}()$ function changes all the 0's to 1's and vice versa. i.e. $\text{Flip}(1001) = 0110$

So, on the first day the Codovid DNA sequence is S_1 . On the second day the sequence is S_1+S_2 . On the third day, the sequence will become $S_1+S_2+S_3$. So, it is an ever-growing sequence, “ $S_1+S_2+S_3+\dots$ ”

An example will make it clear.

Let's assume the Initial CodoVid DNA sequence on Day 1, $S_1 = 100$.

So, $S_2 = \text{Reverse}(100) + \text{Flip}(100) + \text{Reverse}(100) = (001) + (011) + (001) = 001011001$,

$S_3 = \text{Reverse}(001011001) + \text{Flip}(001011001) + \text{Reverse}(001011001) = 100110100110100110100110100$

So, the first few characters of the CodoVid DNA sequence is,

100 001011001 100110100110100110100110100.....

So, the 1st character of the codovid DNA sequence is '1'. Similarly, 2nd, 3rd, 4th, 5th is '0', 6th character is '1' and 7th character is '0' and so on.

In this problem, you are given the initial DNA sequence of the Codovid virus, S_1 and two integers L and R . You have to count the number of 1's from the L^{th} character to the R^{th} character of the Codovid DNA sequence.

Input

The input begins with an integer **T** ($1 \leq T \leq 100$) denoting the number of test cases. T test cases follow. Every case starts with a non-empty string S_1 ($1 \leq |S_1| \leq 10^4$) in a single line. Next line of the input is an integer **Q** ($1 \leq Q \leq 5000$) denoting the number of queries. Then follow Q lines containing two integers **L, R** ($1 \leq L \leq R \leq 10^{17}$). You may safely assume that, **Sum of length(S_1) over all test cases $\leq 5 \times 10^5$** .

Output

For each test case, Print the case number in the format “**Case X:**” in a single line (without quotes), where **X** is the case number. Then, for each query of that test case, print the desired answer in a single line.

Please check the samples for further clarification.

Sample Input

Output for Sample Input

```
1
100
4
1 3
4 12
10 20
1 100
```

```
Case 1:
1
4
5
48
```



Problem D

Der Stern



Let's imagine our earth is at the origin of a 3d cartesian coordinate system and has a radius of 100 units. The most beautiful star of the universe *Der Stern* is at $(0, 0, s)$. You can imagine *Der Stern* to be a tiny but very bright point. Tonight the moon is at $(0, 0, m)$ and the radius of the moon is r . Calculate the probability that a person would be able to see *Der Stern* if the position of the person is randomly chosen on the earth surface. You will be able to see *Der Stern* if there is a line of sight between the person and *Der Stern*.

Please see the sample Input output for more clarity.

Input

The input begins with the number of cases T ($1 \leq T \leq 300$). T test cases follow. Every case consists of 3 integers: s, m, r ($-1,000,000 \leq s, m \leq 1,000,000$; $1 \leq r \leq 1,000,000$).

The earth, moon and the Star would always be disjoint. That is, they won't intersect or touch, also none of them will be inside the other.

Output

For each case, output the case number and the required probability. Answer is expected to be within absolute or relative error of 0.001 of the correct solution.

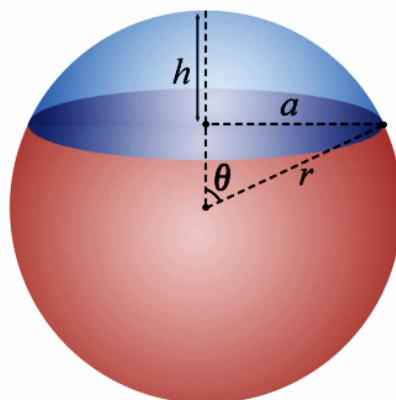
Sample Input

```
1
1000 500 1
```

Output for Sample Input

```
Case 1: 0.4499189902
```

Note



In the picture, the sphere has a radius of r . Suppose we cut a cap of height h ($h \leq r$). The surface area of the cap is $2\pi rh$.



Problem E

Calendars



You will be given a description of several different calendars . You will have to convert the date of one calendar to the other one. If there are p months in a calendar then the calendar is described by a positive integer p followed by p positive integers. These p integers denote the number of days in each month. The months are identified by integers $1, 2, \dots, p$. You do not need to worry about leap years etc as there are no such things associated with these calendars. **Please note that for this problem days, months and years are 1-indexed and these calendars start from the same day (The very first day of all calendars 1-1-1 is the same day).**

Input

First line of the input file contains a positive integer **NC** not exceeding **210**. This denotes the number of calendar descriptions to follow. Each of the next **NC** lines describe a Calendar. These calendars are identified as Calendar **1, 2 ... , NC** respectively.

The description of each calendar is given in a single line. The line starts with an integer **p** ($0 < p \leq 20$) which denotes the number of months in the calendar followed by p positive integers m_1, m_2, \dots, m_p , here m_i denotes the number of days the i -th month has. None of these **p** integers exceed the value **50**.

The next line contains an integer **q** ($0 < q \leq 1010$) which denotes the no of queries. Each of the next **q** lines contain a single query.

Each query has five integers: **cal₁** ($1 \leq \text{cal}_1 \leq \text{NC}$), **cal₂** ($1 \leq \text{cal}_2 \leq \text{NC}$), **day**, **month** and **year** ($1 \leq \text{year} \leq 5000$). Here day, month, year denotes a date in **cal₁** and the task is to convert it to the corresponding date in **cal₂**. You can assume that no invalid date will be given as input.

Output

For each query your program should produce one line of output. This out contains three integers **d₂, m₂, y₂** which denotes the day, month and year of the date when converted to **cal₂**.

Sample Input

Output for Sample Input

2	8 1 2
3 10 20 30	7 2 2
4 20 30 10 20	
2	
1 2 18 2 2	
2 1 17 4 1	



Problem F

Flip



Bitstring is a string consisting of only 0 and 1. Each element in the string is called a bit. A bitstring is called circular when the first and last bit are considered to be adjacent.

Given a circular bitstring of size n and a number k . You want to make all the bits zero. There are two possible operations.

- (i) Take a bit and flip it, the cost of this operation is **1**.
- (ii) Take any k consecutive bits and flip them all, the cost of this operation is **0**.

You are also given q updates. In each update one bit will be flipped. After each update you have to print one line, the minimum cost to make all the bits zero in the current bitstring.

Input

Input starts with **3** numbers, n , k , and q respectively. The following line contains a bitstring of length n . Then q lines each containing an index between **1** to n , denoting which bit to flip.

Constraints

- $2 \leq k \leq n \leq 3 \times 10^5$
- $1 \leq q \leq 3 \times 10^5$

Output

Print one line for each update containing the minimum cost. Please see the sample for details

Sample Input

Output for Sample Input

5 3 2 10010 5 1	0 0
--------------------------	--------

Explanation

Initially the bitstring is: 10010

After the first update it becomes, 10011

We can flip 4th, 5th and 1st bit simultaneously with the 2nd type of operations and make every bit 0 with cost 0.

After the 2nd update it becomes, 00011

Now we do the 2nd type of operations 4 times to achieve all 0 states.

00011 → 01101

01101 → 11110

11110 → 00111

00111 → 00000

Note

Flipping a bit means,

- i) If the bit is 0, change it to 1.
- ii) If the bit is 1, change it to 0.



Problem G

Make GCD Great Again!



You are the president of the United Communities of Contest Programming. In order to maintain the reputation of UCCP, you need to solve a simple, yet interesting problem. You are given an array **A** of **N** positive integers. You can perform the following operation on the given array some number of times (maybe no operation at all), **but not more than once** on a particular element of the array:

- Choose an index **i** of the array. Change **A_i** to some positive number **X > A_i**. The cost of this operation is $|A_i + X| \times |A_i - X|$.

You have to perform the operation to make the GCD (Greatest Common Divisor) of all the elements of the array greater than **1**. But you have to do it at the minimum total cost. You also need to find the maximum GCD you can make with the minimum total cost. Can you make the GCD great again?

Input

The first line will contain a single integer **T**. Each test case will have a single integer **N**, denoting the number of elements of array **A**. The following line will contain **N** space separated integers **A₁, A₂, ..., A_N** denoting the elements of array **A**.

Output

For each case, print one line with “**Case <x>: <y> <z>**”, where **x** is the case number, **y** is the minimum total cost to make the GCD of all the elements of array **A** greater than **1** and **z** is the maximum GCD you can make with the minimum total cost.

Constraints

- $1 \leq T \leq 20$
- $1 \leq N \leq 10^5$
- $1 \leq A_i \leq 10^6$

Sample Input

```
2
5
3 6 12 15 21
7
5 13 17 20 25 33 30
```

Output for Sample Input

```
Case 1: 0 3
Case 2: 191 2
```



Problem H

Vishon Xor



You are given an array **A** of length **N**. You can perform two kinds of operations on this array any number of times (possibly zero).

1. Choose any non-negative integer **X** and then for each **i**, replace **A[i]** with **(A[i] \oplus X)**.
2. Sort the array **A**.

Cost of the array is $\min_{i=1}^{n-1} \text{abs}(A[i] - A[i + 1])$. You need to minimize the cost.

Input

First line will contain a single integer **T** representing the test cases. For each test case there will be a single integer which will represent **N**. In the next line there will be **N** space separated integers where the **i**-th integer will represent **A[i]**.

Constraints

- $1 \leq T \leq 2 \times 10^5$
- $1 \leq N \leq 2 \times 10^5$
- sum of **N** over all test cases $\leq 2 \times 10^5$
- $0 \leq A[i] \leq 10^{18}$

Output

For each test case print a single line representing the minimized cost with test case number (Please check sample output for output format).

Sample Input

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

Output for Sample Input

```
Case 1: 1
Case 2: 0
```



Problem I

Keep Calm, Keep padding



People of Wakanda really love biking. Each city has their own bike hub. You can pick a bike from any hub! After arrival at your destination city, you just need to return the bike to the bike hub.

On a typical morning, each bike hub has K bikes. Whole day people move randomly. So at the end of the day the number of bikes in the hubs gets changed. It can be **greater than**, **less than** or equal to K . We need to re-distribute the bikes so that it is the same as in the morning. So after shuffling each bike hub should have exactly K bikes again.

Wakanda country consists of N cities and there are $N-1$ bi-directional roads between the cities. Every city is reachable from every city by roads.

The Strawhat company is responsible for this bike management system. They have a giant truck of unlimited capacity to move bikes. Their policy is that they will visit all roads exactly twice, but can visit any city multiple times.

Now you need to find **any possible route** so that all conditions are met. You can start from any city. From any bike hub you can collect any numbers of the bikes and similarly give any numbers of bikes from the truck!

Input

The First line of the input contains a single integer T ($1 \leq T \leq 100$) denoting the number of test cases. The description of T test cases as follows:

- The first line of each test case contains two integers N ($1 \leq N \leq 10000$) and K ($1 \leq K \leq 1000$).
- Each of the next $N-1$ lines contains two space-separated integers x and y denote that cities x and y are connected by a road.
- The last line contains N integers denoting the city i has x_i ($0 \leq x_i \leq 10000000$) bikes. It is guaranteed that sum of all x_i will be $N \times K$.

Output

For every test case, you must print the case number, followed by “YES” if there is a possible route, otherwise “NO”.

If there is a possible route, then print $2N-1$ additional lines describing the order of the road traversal. Each line should have 2 integers: u and p .

Where, u indicates the city you will go to and p indicates how many bikes you collect from or give to the bike hub of this city. Non-negative p means you are putting p bikes to the bike hub and negative p means you are taking bikes from bike hub. First line indicates you start from this city.

Please check the samples for further clarification.

Sample Input

```
2
2 100
1 2
200 0
5 20
1 4
2 4
4 5
3 4
31 7 22 23 17
```


Output for Sample Input

```
Case 1: YES
2 0
1 -100
2 100
Case 2: YES
4 -13
2 13
4 -3
5 3
4 0
1 -11
4 11
3 -2
4 2
```

Explanation


In the first case, we start from the bike hub of city 2 and do nothing. Then go to the bike hub of the city 1 and take 100 bikes from there. After this, return to city 2 and put 100 bikes on the bike hub.

Similarly in the second case, we start from the bike hub of the city 4 and take 13 bikes. Then move to city 2 and give 13 bikes and so on.



Problem J

ICPC Standing



Students get the chance to interact with different institutions while showcasing their problem-solving, programming, and teamwork abilities through the International Collegiate Programming Contest (ICPC). The ranking of teams for this contest will be decided based on the following rules.

- The team with the higher number of solved problems will get a higher rank. Rank 1 is considered the highest rank.
- In case of a tie between two teams with the same number of problems solved, the higher rank is determined by penalty time. The team with a lower penalty time will get the higher rank.

The web page that shows all team's ranks in the contest from higher rank to lower rank, is known as the contest standings page. Usually, the contest standing page shows a live rank list consisting of the team name, number of problems solved, total penalty time, problem status (solved or unsolved), number of attempts on each problem, and so on. An example rank list page is shown in the following figure.

STANDINGS													
Rank	Team Name	Solve/ Penalty	A	B	C	D	E	F	G	H	I	J	K
1	Team X (ABC University)	10 1394	✓ 1 (7)	✓ 2 (62)	✗ 1	✓ 3 (130)	✓ 1 (34)	✓ 3 (299)	✓ 1 (63)	✓ 2 (174)	✓ 2 (235)	✓ 1 (140)	✓ 1 (110)
2	Team Y (University of DEF)	9 688	✓ 1 (3)	✓ 1 (28)	✗ 9	✓ 1 (18)	✓ 1 (54)	✗ 4	✓ 1 (105)	✓ 2 (174)	✓ 1 (147)	✓ 1 (67)	✓ 2 (52)
3	Team Z (ABC University)	9 1552	✓ 1 (4)	✓ 1 (92)		✓ 1 (9)	✓ 2 (71)	✗ 1	✓ 4 (166)	✓ 4 (247)	✓ 3 (226)	✓ 1 (169)	✓ 5 (308)
4	Team A (XYZ University)	8 826	✓ 1 (4)	✓ 1 (101)		✓ 1 (72)	✓ 1 (52)	✗ 1	✓ 1 (156)	✓ 2 (212)	✗ 3	✓ 1 (134)	✓ 1 (75)

This year the organizer will not provide any rank list in the contest standing page.

The contest is running. There are still 3 hours remaining in this contest. As a contestant, you are only informed that there are **P** problems in the problem set, your team has already solved **S** problems, and your current team rank is **R**. You have to determine whether there is a chance by any means for your team to become champion after the end of the contest.

Input

The first line will contain a single integer **T** ($1 \leq T \leq 18150$). Each test case will contain three integers **P** ($1 \leq P \leq 10$), **S** ($0 \leq S \leq P$) and **R** ($1 \leq R \leq 165$).

Output

For each test case, Print the case number in the format “**Case X:**” in a single line (without quotes), where **X** is the case number, followed by “**Yes**” if there is a chance for your team, “**No**” otherwise.

Sample Input

Output for Sample Input

<pre>2 10 0 165 10 10 1</pre>	<pre>Case 1: Yes Case 2: Yes</pre>
-------------------------------	------------------------------------



Problem K

Clash of Coprimes



Roh has recently started working on a new project and he really enjoys working on it. The thing that annoys him, however, is that the requirements seem to keep changing continuously. Everyday, Roh's supervisor Zach comes to him with a new list of requirements that Roh must fulfill. The ever-changing requirements have frustrated Roh and he wishes for once that Zach could make up his mind.

Roh's task is simple. He must construct an array $a[1 \dots n]$ of size n , whose elements are positive integers **greater than 1**. Everyday Zach comes to Roh with a new set of requirements. The requirement on the i^{th} day is defined by a subset of indices, S_i . For any two indices x, y such that $x \in S_i$ and $y \notin S_i$, it is required that $\gcd(a[x], a[y]) = 1$.

On the i^{th} day, Roh must construct an array of positive integers $a[1 \dots n]$ of size n , such that,

- $a[i] \geq 2$ for all $1 \leq i \leq n$.
- The array satisfies all requirements from day 1 to day i , ie. it satisfies each of the sets $S_1, S_2, S_3, \dots, S_i$

Of course, there might be many such arrays and Roh must find the array with the minimum sum of elements. Zach doesn't have the time to go through all of Roh's work. So, he asks Roh to report only two numbers, the minimum sum of elements that Roh can achieve, and the number of valid arrays with the minimum sum.

Help Roh complete his task.

Input

The first line contains T , the number of test cases. The first line of each case contains two integers n , and q denoting the number of elements and the number of requirements. It is followed by q lines. The i^{th} line describes the set S_i . It starts with an integer k denoting the number of elements of S_i . Then k distinct integers follow $S_{i,1}, S_{i,2}, S_{i,3}, \dots, S_{i,k}$ the elements of S_i .

Constraints

- $1 \leq T \leq 100$
- $2 \leq n \leq 10^5$
- $1 \leq q \leq 10^5$
- $1 \leq k \leq n$
- $1 \leq S_{i,j} \leq n$ for all $1 \leq j \leq k$
- All elements of a set S_i are distinct, ie. $S_{i,1}, S_{i,2}, \dots, S_{i,k}$ are distinct.
- Sum of n over all cases does not exceed 5×10^5 .
- Sum of q over all cases does not exceed 5×10^5
- The total number of elements over all sets over all test cases does not exceed 10^6 .

Output

For each test case, print **q** lines. The **ith** line for each case should contain two space-separated integers. The first integer is the minimum sum of a valid array that fulfills all requirements upto day **i**. The second integer is the number of valid arrays with the minimum sum. Since the number of valid arrays can be large, print the number of valid arrays modulo **998244353**.

Sample Input

Output for Sample Input

1	10 2
4 2	12 2
2 1 2	
3 1 2 3	

For the first day, **2, 2, 3, 3** and **3, 3, 2, 2** are the interesting sets with the minimum sum.

For the second day, **2, 2, 3, 5** and **2, 2, 5, 3** are the interesting sets with the minimum sum.