Problem code: **KGP16A**
Problem name: **Seats**

There are **N** people wanting to seat on a row of **N** seats. The persons and seats are numbered from **1** to **N**.

The **N** people are ordered according to who chooses the seats first. In other words, person **1** is the first to seat, followed by person **2**, then person **3**, etc.

In addition, the $\mathbf{i^{th}}$ person has a preferred seat $\mathbf{S_i}$, a number from **1** to **N**, and a preferred direction $\mathbf{D_i}$, which is either **left** or **right**. This means that on the $\mathbf{i^{th}}$ person's turn, he/she first checks seats $\mathbf{S_i}$ if it is available. If it's available, then he/she sits there, otherwise he walks into the direction specified by $\mathbf{D_i}$ and sits on the first seat available in that direction. It's possible that the $\mathbf{i^{th}}$ person doesn't find an available seat at all; if that happens, that person fails to seat and simply leaves.

You know their preferred seats [$\mathbf{S_1}$, $\mathbf{S_2}$, ..., $\mathbf{S_N}$], but you don't know their preferred directions [$\mathbf{D_1}$, $\mathbf{D_2}$, ..., $\mathbf{D_N}$]. So the problem is: how many assignments of preferred directions are there such that all **N** people successfully seat?

Input

The first line of input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The first line of each test case contains a single integer **N**.

The second line contains **N** integers separated by single spaces: $\mathbf{S_1}$, $\mathbf{S_2}$, ..., $\mathbf{S_N}$.

Output

For each test case, output a single line containing the answer. Since the answer can be very large, only output it modulo $\mathbf{10^9 + 7}$.

Constraints

- **$1 \leq T \leq 200$**
- **$1 \leq N \leq 60$**
- **$1 \leq S_i \leq N$**

Example

Input:
```
1
3
2 2 3
```
Output:
```
6
```

Explanation

The following assignments of preferred directions yield a successful seating:

- **[left, left, left]** which yields the seating **[2, 1, 3]**.
- **[left, left, right]** which yields the seating **[2, 1, 3]**.
- **[left, right, left]** which yields the seating **[3, 1, 2]**.
- **[right, left, left]** which yields the seating **[2, 1, 2]**.
- **[right, left, right]** which yields the seating **[2, 1, 2]**.
- **[right, right, left]** which yields the seating **[3, 1, 2]**.

Problem code: **KGP16B**
Problem name: **Path by a Castle**

Jared and Payton are on the 2D plane, at locations (x_a, y_a) and (x_b, y_b), respectively. They want to meet, but Payton is lazy and doesn't want to move. So Jared wants to walk to Payton's location. But Jared is also lazy, so he wants to do it with the shortest distance possible.

Unfortunately, a castle stands in their way. The castle is an enclosure of walls whose shape is the quadrilateral with vertices (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4), in that order. The walls themselves are infinitely thin, but Jared is not allowed to cross it, or even get near them by a distance of 10^{-100}.

What is the shortest distance that Jared can take to get to Payton's location?

Input

The first line of input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The first line of each test case contains four integers, x_a, y_a, x_b, and y_b, separated by single spaces.

The i^{th} line in the next four lines contains two integers, x_i and y_i, separated by a space.

Output

For each test case, output a single line containing a single real number: the answer to the problem. If there is no path at all, print -1. Your answer will be considered correct if it is within a relative error of $10^{-6}$from the correct answer.

Constraints

- $1 \leq T \leq 10^5$
- The absolute values of the coordinates are at most 10^5.
- Jared and Payton are initially at least a distance of 10^{-100} from the walls.
- The walls form a simple, valid quadrilateral without straight angles.

Example

Input:

```
3
0 -4 0 4
2 2
2 -2
-2 -2
-2 2
0 0 0 4
2 2
-2 2
-2 -2
2 -2
-1 3 1 3
-2 0
0 4
2 0
0 11
```

Output:

```
9.6568542495
-1
2.82842712475
```

Explanation

In the first test case, there are two shortest paths: either go by the left or right of the castle walls.

Problem code: **KGP16C**
Problem name: **Timestamps**

A scientist is performing an experiment. She has to note down the exact time at which **N** interesting events occur. Unfortunately, she is very tired and misses some digits while noting down the time. Also, she just forgets about the date.

So at the end, there are **n** timestamp entries in **hh:mm:ss** format, where **hh** denotes the hour, **mm** minutes and **ss** the seconds. The time is in 24-hour format. Some of the digits are missing and will be denoted by a '?'. The scientist has also forgotten how long back she had started the experiment! So, given these **N** times, what is the minimum amount of time that has passed since the start of the experiment?

The **i**-th event always occurs after or at the same time as the **(i-1)**-th event. The first event always begins at **00:00:00** and that timestamp is correctly noted.

Input

The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The first line of each test case contains a single integer **N** denoting the number of events.

The next **N** lines each contain strings $\mathbf{T_1}$, $\mathbf{T_2}$, ..., $\mathbf{T_N}$ denoting the **N** timestamps.

Each timestamp has 8 characters of the form: "hh:mm:ss" (without quotes) denoting the hour (hh), minute (mm) and seconds (ss). Each part maybe prefixed by a 0 to make it fit in the format. Some of the digits in the timestamp may be absent and will be denoted by '?'.

Output

For each test case, output a single line containing the minimum number of seconds taken for the experiment to be completed.

Constraints

- $\mathbf{1 \leq T \leq 1000}$
- $\mathbf{1 \leq N \leq 10^6}$
- Sum of all **N** over all test cases in a single test file will not exceed $\mathbf{10^6}$

Example

Input:

```
4
5
00:00:00
10:10:10
11:11:11
23:59:59
00:00:00
3
00:00:00
23:59:59
23:59:58
2
00:00:00
2?:??:?1
3
00:00:00
23:59:59
23:59:?8
```

Output:

```
86400
172798
72001
172748
```

Explanation

Example case 1. The first event and last event takes place exactly 24 hours apart and all the other event could have occurred between these 2 events. So the minimum amount of time passed is 24 hours (or 24 * 60 * 60 = 86400 seconds)

Example case 3. Since we have to minimize the amount of time passed, we can fill all ? with 0 and assume the second event occurred at 20:00:01

Problem code: **KGP16D**
Problem name: **Sweet Shops of Kolkata**

Park Street Area market in Kolkata is famous for its mouth watering sweets. There are a total of **N** shops in the market, which are extended over a single street with shops following one another. The shops are enumerated from **1** to **N**, from left to right. The market is unique in the sense that each shop sells only a single type of sweet. Additionally, no two shops sell the same type of sweet. For convenience, the different types of sweets are numbered from **1** to **N**.

Though the famous market has been running smoothly for many years, the city Mayor decided to have it completely overhauled and aims to make this famous market great again. So he has issued a tender for the shops, which you happened to grab by using your good offices with the Mayor. The tender specifies that you should decide which type of sweet each shop should sell. You would like to do so, while keeping in mind that buyers like purchasing sweets if adjacent shops sell sweets of similar types, because then they can purchase similar type sweets without wandering around many shops. Formally, you would like to minimize the maximum absolute value of difference between the types of sweets in adjacent shops. Two shops which are numbered **i** and **i + 1** (**i < N**) are said to be adjacent to each other.

The city Mayor himself is very fond of sweets and has decided to have his own shop in the market too. He has chosen shop numbered **S** in which he will sell sweets of type **Q**. For the remaining **N - 1** shops, you have to assign which sweets should be sold so as to minimize the maximum absolute difference in the type of sweets in adjacent shops.

Input

First line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The only line of each test case contains three space separated integers **N, S, Q**, denoting number of shops in the market, the shop chosen by the Mayor, and the type of sweet the Mayor's shop sells, respectively.

Output

For each test case, output two lines.

In the first line of each test case, output the minimum value of the maximum difference in the types of sweets of adjacent shops.

In the next line, output **N** space separated integers denoting the types of sweets sold in each of the shops. If there are multiple answers, output any one.

Constraints

- **$1 \leq T, N \leq 500$**
- **$1 \leq S, Q \leq N$**

Example

Input:

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

Output:

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

Explanation

Example case 1. The city Mayor has decided to sell sweets of type 3 in the second shop. In the first shop, you can decide to sell sweets of type 1 and that of type 2 in the third shop. In this case, maximum difference in the type of sweets of adjacent shops is 2. There can be one other way of assigning sweets with shops, i.e. sell sweets of type 2 in shop 1 and that of type 1 in shop 3. In this case also, maximum difference in type of adjacent shops is 2. So, you can see that in any case, minimum value of maximum difference in type of adjacent shops will be 2. Hence answer is 2.

Problem code: **KGP16E**
Problem name: **Coal mines of Kharagpur**

Coal Mafias in Kharagpur are notorious for being one of the biggest perpetrators of crimes against officials protecting the mines. There are **n** mines in the area, numbered from **1** to **n**. Government wants to protect the mines by deploying some guards. As you know, governments usually don't have enough resources to assign a single guard for each mine. Instead, they know that some of the mines are quite near each other, and if there is a guard at some mine, he can monitor/guard these nearby mines also. This information about which of the mines are near to each other is given by **m** pair of integers **u, v** denoting the mine numbered **u** and **v** are near to each other.

Earlier government already had all the arrangements in the place. Right after the elections, they decided to remove all the guards. So, now it is the new government's responsibility to allocate guards for the mines. You came to know through a very trusted source that the previous government had less than or equal to **ceil(n / 3)** guards that were protecting all the mines.

Now, you have the responsibility of assigning the guards to the mines. Admit it or not, the general perception of people is that, the current government under which you work is doubly worse and corrupt as compared to the previous one. So, they will cut you slack and will not pay an iota of attention if you put at most **ceil((2 * n) / 3)** guards for protection of all the mines. If you employ more than this number of guards, public will definitely notice it sooner and later, and will think of this as a scam which will make it hard for the government to save its face in parliament. Mind you, heavens will fall if you dare to deploy more than one guard in a mine.

Find out any possible deployments of guards to the mines. If there are multiple possible ways of choosing mines where guards should be present, you can output any.

Note

ceil(x) denotes the smallest integer greater than or equal to x

Input

First line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The first line of each test case contains two space separated integers **n, m** denoting number of mines and the number of proximity relationships among the mines.

Each of the next **m** lines contains two space separated integers **u, v** denoting that mines numbered **u** and **v** are near each other.

Output

For each test case, output two lines.

In the first line, output a single integer corresponding to number of guards that you are deploying. Let this number be **K**.

In the next line, output **K** space separated integers corresponding to the mines where you will be having guards for protection.

Constraints

- $\mathbf{1 \leq T \leq 10^5}$
- $\mathbf{1 \leq n \leq 10^5}$
- $\mathbf{1 \leq u, v \leq n}$
- $\mathbf{0 \leq m \leq min((n * (n - 1)) / 2, 2 * 10^5)}$
- $\mathbf{1} \leq$ Sum of **n** and **m** each, over all the test cases $\leq \mathbf{5 * 10^5}$

Example

Input:

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

Output:

```
1
2
2
1 2
```

Explanation

Example case 1. You can deploy a guard at the mine 2. He will also guard the nearby mines 1 and 3 too.

Example case 2. You can have an guard at mine 1 and one more guard at mine 2. These two guards will also guard the 3rd mine. As, we see that number of guards deployed are 2, which is equal to ceil(6 / 3) = 2, so nobody will say anything about this.

Problem code: **KGP16F**
Problem name: **Optimal Partition**

Competitive Programmers love concise statements. Today, we have one such problem for you!

You're given an array **A** with **N** integers, $\mathbf{A_1, A_2, A_3....A_N}$. You have to partition the elements of array **A** into exactly **K** disjoint subsets such that the following conditions are satisfied:

- Size of each subset must be at least **2**.
- Let the **K** subsets be $\mathbf{S_1, S_2.... S_K}$. Let $\mathbf{D_i = max(S_i) - min(S_i)}$ where $\mathbf{max(S_i)}$ is maximum element in $\mathbf{S_i}$ and $\mathbf{min(S_i)}$ is the minimum element in $\mathbf{S_i}$. We want to minimize value of $\mathbf{D_1 + D_2 + D_3 + ... + D_K}$.

Find the minimum possible sum of $\mathbf{D_1 + D_2 + D_3 + ... + D_K}$ that you can obtain.

Input

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

The first line of each test case will contain two space separated integers **N, K**.

The next line will contain **N** space separated integers $\mathbf{A_1, A_2, A_3....A_N}$

Output

For each test case, output a single integer denoting the minimum sum.

Constraints

- $\mathbf{1 \leq T \leq 10}$
- $\mathbf{2 \leq N \leq 10^5}$
- $\mathbf{2 \leq}$ **Sum of N over all test cases in a single file** $\mathbf{\leq 10^5}$
- $\mathbf{1 \leq K \leq min(500, N / 2)}$
- $\mathbf{1 \leq A_i \leq 10^{12}}$

Example

Input:

```
1
5 2
19 16 5 23 17
```

Output:

```
16
```

Explanation

The best partition that we can obtain involves partitioning **A** into {16, 5, 17} and {19, 23}, incurring a cost of (17 - 5) + (23 - 19) = 16

Problem code: **KGP16G**
Problem name: **Painting a Town**

You are mayor of Kolkata, and you've decided to create a new town in the outskirts of the city!

This new town is very well planned. All the houses in the town are arranged in a rectangle of dimensions **n * m**, i.e. there are **n** rows of houses with each row containing **m** houses. The contract of painting the houses was given to a company **Brute Inc**, which did a rather lousy job. It had to paint all the houses in either red, blue or green color, but they left some of the houses unpainted.

When you saw the shoddy work done by the contractor, you decided to give the task of completing the unfinished painting work to another company, **Entropy Inc**. This company works in a rather weird way; It paints each unpainted house with either red, blue or green color **uniformly at random**.

Consider any square shaped sub-grid of houses in this rectangular town. It can be uniquely represented by the top left house and the bottom right house. This sub-grid of houses will be called **beautiful** if all the houses in it have the same color. The overall beauty of the town will be equal to the total number of beautiful square sub-grids in the town. Your task is to find expected beauty of the town after the painting work is completed.

A sub-grid of houses can be uniquely defined by the coordinates of top left and bottom right houses, i.e. $(\mathbf{x_1}, \mathbf{y_1})$ and $(\mathbf{x_2}, \mathbf{y_2})$, respectively. The houses that lie in the sub-grid will be the ones that have their coordinates **(x, y)** satisfy both $\mathbf{x_1} \leq \mathbf{x} \leq \mathbf{x_2}$ and $\mathbf{y_1} \leq \mathbf{y} \leq \mathbf{y_2}$. This sub-grid will be a square sub-grid if $\mathbf{x_2} - \mathbf{x_1} = \mathbf{y_2} - \mathbf{y_1}$. In particular 1 * 1 cell is also called a square sub-grid. Top left house of the town has coordinates **(1, 1)** and bottom right house has coordinates **(n, m)**.

Input

There is a single test case.

The first line of the input contains two space separated integers **n, m** denoting dimensions of town.

Each of the next **n** lines contains a string consisting of **m** characters, each of them can be **'R'** (denoting red color), **'G'** (denoting green color), **'B'** (denoting blue color) or **'*'** (denoting unpainted house).

Output

Output a single real number corresponding to the expected beauty of the town after the painting of the houses. Your answer will be considered correct if its absolute error is within **1e-2** of the answer.

Constraints

- **$1 \leq n, m \leq 4000$**

Example 1

Input:
```
2 2
RR
RR
```

Output:
```
5
```

Explanation

There are total 5 square sub-grids of houses in the town. All of them painted with same color. So, beauty of the town will be 5.

Example 2

Input:
```
2 2
R*
RR
```

Output:
```
4.33
```

Explanation

The unpainted house can be painted with any of the red, green, blue colors uniformly randomly.

- It is colored red: In this case, beauty of town will be 5.
- It is colored green: In this case, beauty of town will be 4.
- It is colored blue: In this case, beauty of town will be 4.

Hence, expected beauty of the town will be **(5 + 4 + 4) / 3 = 13 / 3 = 4.33.**

Problem code: **KGP16H**
Problem name: **Coal Mafia and Toll Tax**

You are one of the notorious coal mafias. You engage in activities of illegally mining coals. There are **n** villages, enumerated from **1** to **n** from left to right. Each of the village has some coal mines. This information is given to you by an integer $\mathbf{coal_i}$ denoting the amount of coal (in kgs) in village **i**. The cost of coal is fixed at 1 rupees per kg.

There are toll booths in the highway at the middle of the road connecting any two adjacent villages. So, if you want to go from village **i** to village **i + 1**, you might have to pay toll tax for it. This information is provided you by an integer $\mathbf{tax_i}$ which denotes the toll tax to pay on the road joining villages **i** and **i + 1**. Additionally, some of the toll booths allow you to go both to and fro, after paying tax once, others don't. This is provided you by an integer $\mathbf{policy_i}$ which will be either 1 or 2, denoting whether the toll booth allows 1 way taxation or 2 way taxation. If a toll booth has 1 way taxation, you can cross it twice by paying tax for only the first time you cross the road, you don't need to pay tax the second time. On other hand, if it has two way taxation, then you will have to pay toll tax both the times you cross the road.

You brought a big truck and now want to collect coal in turn maximizing your profit. You will start your journey at village 1, and collect coal from some of the villages and then end back at village 1 itself.

Input

First line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The first line of each test case a single integer **n** denoting the number of villages.

The second line of each test case contains **n** space separated integers in which the **i**-th of them denotes $\mathbf{coal_i}$ - amount of coal in village **i**.

The third line of each test case contains **n - 1** space separated integers in which the **i**-th of them denotes $\mathbf{tax_i}$ - amount of toll tax you have to pay each time you cross the road between village **i** and **i + 1**.

The fourth line of each test case contains **n - 1** space separated integers in which the **i**-th of them denotes **policy$_i$** - whether the toll booth charges 1 way taxation or 2 way while using the road between village **i**and **i + 1**.

Output

For each test case, output a single integer in a separate line corresponding to the maximum profit that you can obtain.

Constraints

- **$1 \leq T, n \leq 50$**
- **$0 \leq coal_i, tax_i \leq 10^5$**
- **$1 \leq policy_i \leq 2$**

Example

```
Input:
2
2
1 2
2
2
2
1 3
2
1
Output:
1
2
```

Explanation

Example case 1. There is a toll tax of 2 Rs for crossing the road between village 1 and 2. So, if you decide to collect the coal of 2 units in 2nd village, then you will lose 4 Rs in toll tax. Instead, you just decide to collect coal from first village itself, So your profit will be just 1 Rs.

Example case 2. There is toll tax of 2 Rs for crossing the road between village 1 and 2. So you decide to collect the coal from second village, total coal collected will be 4 units, and you will have to spend 2 rupees in toll tax. So, your profit will be just 2 Rs. Hence answer is 2.

Problem code: **KGP16I**
Problem name: **Monetize Mania**

Recently The Prime Minister of India, Narendra Modi launched the ambitious Gold Monetization Scheme, under which one can deposit their gold reserves to the banks. There are **n** banks in Kharagpur. For simplicity, we enumerate them from **1** to **n** in the order. If you deposit gold of **x** grams in bank **i**, you will have to pay a charge of $\mathbf{a_i} * \mathbf{x} + \mathbf{b_i} * \mathbf{x^2}$ Rs. Government mandates for each bank a limit of gold that it can store, specifically bank **i** can store up to $\mathbf{lim_i}$ grams of gold.

As you know that due to this scheme, people are allowed to split their total gold reserve into various parts (the quantity of gold in parts need not be an integer, it can be any real number) and deposit that into banks. Kejri, being a curious citizen of Kharagpur and a relentless critic of Modi, is interested in knowing answers of following type of **Q** questions that popped into his mind. He believes that the benefits earned through this scheme will not able to outweigh the cost paid for storing the gold. If we consider only the banks numbered from **l** to **r** (inclusive), and one has to store **g** grams of gold in the banks, what's the minimum amount (in Rupees) one has to pay the banks. Note that one can store the amount of gold by breaking it into parts and store the parts into various banks. The weight of these parts can be any real number.

Please note that all the queries are independent to each other, i.e. the money deposited in the previous query won't matter to the limits of gold reserves of the bank in this query.

Can you please help Kejri in this task?

Input

There is a single test case.

The first line contains two space separated integers **n, Q** denoting the number of banks, and the number of questions for which Kejri wants the answers.

The second line contains **n** space separated integers, **i**-th of them denotes $\mathbf{a_i}$.

The third line contains **n** space separated integers, **i**-th of them denotes $\mathbf{b_i}$.

The fourth line contains **n** space separated integers, **i**-th of them denotes $\mathbf{lim_i}$ - maximum amount of gold that bank **i** can store.

In the next **Q** lines, each line will specify a query. There will be three space separated integers, **l, r, g**, denoting that you have to consider only the banks numbered in the range **[l, r]** (inclusive), and want to store **g** grams of gold.

Output

For each query, output a single real number corresponding to the answer of the each query - i.e. the minimum rupees one has to spend. Your answer will be considered correct if the relative error of your answer does not exceed 1e-6.

Constraints

- $\mathbf{1 \leq n, Q \leq 10^5}$
- $\mathbf{1 \leq l \leq r \leq 10^5}$
- $\mathbf{0 \leq a_i, b_i, lim_i \leq 10^3}$
- For each query, it is guaranteed that banks in the range **[l, r]** have enough capacity for **g** grams of gold, i.e. $\mathbf{lim_l + lim_{l+1} + ... + lim_r \geq g}$.

Example

Input:

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

Output:

```
2
3
5
1.6666666
```

Explanation

There are two banks. The first bank will charge $\mathbf{a_1*x + b_1*x^2}$, i.e. $\mathbf{x + x^2}$ Rs for depositing **x** grams gold, whereas the second bank will charge $\mathbf{a_2*x + b_2*x^2}$, i.e. $\mathbf{x + 2*x^2}$ Rs. Both banks can store up to max **1**gram of gold.

Query 1. Kejri wants to store 1 gram gold in Bank 1, he will have to pay 2 Rs for that.

Query 2. Kejri wants to store 1 gram of gold in Bank 2, he will have to pay 3 Rs for that.

Query 3. Kejri wants to store 2 grams of gold in Banks 1 and 2, In both the banks, he will store 1 gram of gold each, so he will have to pay $(1+1^2) + (1+2*1^2) = 5$ Rs for that.

Query 4. Kejri wants to store 1 gram of gold in Banks 1 and 2. It would be optimal for Kejri to store 2/3 grams of gold in the Bank 1 and 1/3 grams of gold in Bank 2, for which he will have to pay $((2/3) + (2/3)^2) + ((1/3) + 2*(1/3)^2) = 1.666666...$ Rs.

Problem code: **KGP16J**
Problem name: **Bus Routes**

The Kingdom of Kharagpur has **N** cities, which are numbered from **1** to **N**. There are many bus routes which go directly from one city to another. A bus route from **City i** to **City j** does not mean that there is a route from **City j** to **City i**.

Yesterday, the King had a dream, and he has decided that for the country to become prosperous, in every city, the number of bus routes coming in should be equal to the number of bus routes going out of that city. He called his transport minister and directed him to implement this. The minister realized that he cannot add any new bus routes, because of financial constraints. So, he can only cancel some of the existing routes. But he also knows that some of these routes are Vital, and hence he cannot cancel them. So, he also wants to cancel as few non-Vital routes as possible. Help the minister find the minimum number of non-Vital routes that he has to cancel, so as to satisfy the King's condition.

Input

The first line of input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The first line of each test case contains three integers **N, V, W**, denoting the number of cities, the number of existing Vital routes and the number of existing non-Vital routes, respectively.

The next **V** lines contain two integers each, separated by single spaces: **i j**, denoting that there is a Vital route from **City i** to **City j**.

The next **W** lines contain two integers each, separated by single spaces: **i j**, denoting that there is a non-Vital route from **City i** to **City j**.

Output

For each test case, output a single line containing the answer, which is the minimum number of non-Vital routes to be cancelled so as to satisfy the King's condition.

Constraints

- **1 ≤ T ≤ 5**
- **1 ≤ N ≤ 100**
- **1 ≤ V, W ≤ 1000**

Example

Input:

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

Output:

```
1
```

Explanation

In the existing network of bus routes:

- The number of routes going out of **City 1** is 2 , and the number of routes coming in is 2.
- The number of routes going out of **City 2** is 1 , and the number of routes coming in is 2.
- The number of routes going out of **City 3** is 2 , and the number of routes coming in is 1.
- The number of routes going out of **City 4** is 1 , and the number of routes coming in is 1.

This does not satisfy the King's condition, because in **Cities 2** and **3**, number of incoming routes does not equal number of outgoing routes.

But we can cancel the route which goes from **City 3** to **City 2** (which is a non-Vital route), and after this change, in the new network:

- The number of routes going out of **City 1** is 2 , and the number of routes coming in is 2.
- The number of routes going out of **City 2** is 1 , and the number of routes coming in is 1.
- The number of routes going out of **City 3** is 1 , and the number of routes coming in is 1.
- The number of routes going out of **City 4** is 1 , and the number of routes coming in is 1.

This satisfy's the King's condition, and 1 is the minimum that we can do. Hence the answer is 1.

Problem code: **KGP16K**
Problem name: **Cookie Flavors**

For Durga Puja, Ramu has baked **N** cookies, and left them in a row. They are of various flavours, and the flavours of each of the cookies is given by the sequence **C = (c_1, c_2, ..., c_N)**, where c_i is the flavour of the i^{th} cookie.

Hanu, is a cookie aficionado and over the ages, has perfected a 'base' sequence of cookie flavours that he prefers to eat in that particular order. It is given by **H = (h_1, h_2, ..., h_M)**, where h_i is the i^{th} flavour that he wants to eat. In addition to this 'base' sequence, he is also happy to eat any 'power' of it, because it leaves the same tastes in his tongue, just intensified. The k^{th} power of **H = (h_1, h_2, ..., h_M)** is H^k **= (h_1, h_1, ... [k** times**], h_2, h_2, ... [k** times**],, ... , h_M, h_M, ... [k** times**])**. ie. the first flavour is repeated **k** times, then the second flavour is repeated **k** times, and so on. He is happy to eat H^k, for any $k \geq 1$.

Hanu also only eats cookies which are kept in a row, one after another. Given these, he wants you to find out the number of ways in which he can eat some of cookies baked by Ramu.

Each flavour is represented by an integer from **1** to 10^6.

Also, Hanu's tastes are complex enough, and it is guaranteed that in his 'base' sequence, there are at least three distinct flavours.

Input

The first line of input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The first line of each test case contains two integers **N, M**, denoting the number of cookies baked, and the number of cookies in Hanu's 'base' sequence, respectively.

The second line contains **N** integers separated by single spaces: c_1, c_2, ..., c_N, denoting the sequence of flavours baked by Ramu.

The third line contains **M** integers separated by single spaces: h_1, h_2, ..., h_M, denoting the 'base' sequence of flavours which Hanu likes to eat.

Output

For each test case, output a single line containing the answer.

Constraints

- $1 \leq T \leq 10$
- $1 \leq M \leq N \leq 10^5$
- $1 \leq c_i, h_i \leq 10^6$

Example

Input:

```
1
16 4
3 2 2 2 5 5 6 6 2 2 5 6 2 5 6 2
2 5 6 2
```

Output:

```
3
```

Explanation

The four cookies starting from the 10^{th} cookie are (2, 5, 6, 2), and these form a 'base' sequence which Hanu likes.

The four cookies starting from the 13^{th} cookie are (2, 5, 6, 2), and these form a 'base' sequence which Hanu likes.

The eight cookies starting from the 3^{rd} cookie are (2, 2, 5, 5, 6, 6, 2, 2), and these form the second power of Hanu's 'base' sequence. ie. H = (2, 5, 6, 2), and H^2 = (2, 2, 5, 5, 6, 6, 2, 2), and hence Hanu would eat these 8 cookies.

These are the only 3 ways in which Hanu can eat cookies. Hence the answer is 3.