

A. Meaning Mean

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

Pak Chanek has an array a of n positive integers. Since he is currently learning how to calculate the floored average of two numbers, he wants to practice it on his array a .

While the array a has at least two elements, Pak Chanek will perform the following three-step operation:

1. Pick two different indices i and j ($1 \leq i, j \leq |a|$; $i \neq j$), note that $|a|$ denotes the current size of the array a .
2. Append $\lfloor \frac{a_i + a_j}{2} \rfloor^*$ to the end of the array.
3. Remove elements a_i and a_j from the array and concatenate the remaining parts of the array.

For example, suppose that $a = [5, 4, 3, 2, 1, 1]$. If we choose $i = 1$ and $j = 5$, the resulting array will be $a = [4, 3, 2, 1, 3]$. If we choose $i = 4$ and $j = 3$, the resulting array will be $a = [5, 4, 1, 1, 2]$.

After all operations, the array will consist of a single element x . Find the maximum possible value of x if Pak Chanek performs the operations optimally.

* $\lfloor x \rfloor$ denotes the floor function of x , which is the greatest integer that is less than or equal to x . For example, $\lfloor 6 \rfloor = 6$, $\lfloor 2.5 \rfloor = 2$, $\lfloor -3.6 \rfloor = -4$ and $\lfloor \pi \rfloor = 3$

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 5000$). The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \leq n \leq 50$) — the length of the array a .

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

Do note that the sum of n over all test cases is not bounded.

Output

For each test case, output a single integer: the maximum possible value of x after all numbers have been picked.

Standard Input	Standard Output
3 5 1 7 8 4 5 3 2 6 5 5 5 5 5 5 5	6 4 5

Note

In the first test case, the array is initially $a = [1, 7, 8, 4, 5]$. Pak Chanek will perform the following operations:

1. Pick $i = 1$ and $j = 2$, then $a = [8, 4, 5, 4]$.
2. Pick $i = 3$ and $j = 2$, then $a = [8, 4, 4]$.
3. Pick $i = 2$ and $j = 3$, then $a = [8, 4]$.
4. Pick $i = 1$ and $j = 2$, then $a = [6]$.

After all the operations, the array consists of a single element $x = 6$. It can be proven that there is no series of operations that results in x greater than 6 in the end.

B. Maximize Mex

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

You are given an array a of n positive integers and an integer x . You can do the following two-step operation any (possibly zero) number of times:

1. Choose an index i ($1 \leq i \leq n$).
2. Increase a_i by x , in other words $a_i := a_i + x$.

Find the maximum value of the **MEX** of a if you perform the operations optimally.

The **MEX** (minimum excluded value) of an array is the smallest non-negative integer that is not in the array. For example:

- The **MEX** of $[2, 2, 1]$ is 0 because 0 is not in the array.
- The **MEX** of $[3, 1, 0, 1]$ is 2 because 0 and 1 are in the array but 2 is not.
- The **MEX** of $[0, 3, 1, 2]$ is 4 because 0, 1, 2 and 3 are in the array but 4 is not.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 5000$). The description of the test cases follows.

The first line of each test case contains two integers n and x ($1 \leq n \leq 2 \cdot 10^5$; $1 \leq x \leq 10^9$) — the length of the array and the integer to be used in the operation.

The second line of each test case contains n integers a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^9$) — the given array.

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 maximum **MEX** of a if you perform the operations optimally.

Standard Input	Standard Output
3 6 3 0 3 2 1 5 2 6 2 1 3 4 1 0 2 4 5 2 5 10 3	4 6 0

Note

In the first test case, the **MEX** of a is 4 without performing any operations, which is the maximum.

In the second test case, the **MEX** of a is 5 without performing any operations. If we perform two operations both with $i = 1$, we will have the array $a = [5, 3, 4, 1, 0, 2]$. Then, the **MEX** of a will become 6, which is the maximum.

In the third test case, the **MEX** of a is 0 without performing any operations, which is the maximum.

C1. Adjust The Presentation (Easy Version)

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

This is the easy version of the problem. In the two versions, the constraints on q and the time limit are different. In this version, $q = 0$. You can make hacks only if all the versions of the problem are solved.

A team consisting of n members, numbered from 1 to n , is set to present a slide show at a large meeting. The slide show contains m slides.

There is an array a of length n . Initially, the members are standing in a line in the order of a_1, a_2, \dots, a_n from front to back. The slide show will be presented in order from slide 1 to slide m . Each section will be presented by the member at the front of the line. After each slide is presented, you can move the member at the front of the line to any position in the lineup (without changing the order of the rest of the members). For example, suppose the line of members is $[3, 1, 2, 4]$. After member 3 presents the current slide, you can change the line of members into either $[3, 1, 2, 4]$, $[1, 3, 2, 4]$, $[1, 2, 3, 4]$ or $[1, 2, 4, 3]$.

There is also an array b of length m . The slide show is considered *good* if it is possible to make member b_i present slide i for all i from 1 to m under these constraints.

However, your annoying boss wants to make q updates to the array b . In the i -th update, he will choose a slide s_i and a member t_i and set $b_{s_i} := t_i$. Note that these updates are **persistent**, that is changes made to the array b will apply when processing future updates.

For each of the $q + 1$ states of array b , the initial state and after each of the q updates, determine if the slideshow is good.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains three integers n, m and q ($1 \leq n, m \leq 2 \cdot 10^5$; $q = 0$) — the number of members, the number of sections and the number of updates.

The second line of each test case contains n integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq n$) — the initial order of the members from front to back. It is guaranteed that each integer from 1 to n appears exactly once in a .

The third line of each test case contains m integers b_1, b_2, \dots, b_m ($1 \leq b_i \leq n$) — the members who should present each section.

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

Output

For each test case, output $q + 1$ lines corresponding to the $q + 1$ states of the array b . Output "YA" if the slide show is good, and "TIDAK" otherwise.

You can output the answer in any case (upper or lower). For example, the strings "yA", "Ya", "ya", and "YA" will be recognized as positive responses.

Standard Input	Standard Output
----------------	-----------------

3	YA
4 2 0	YA
1 2 3 4	TIDAK
1 1	
3 6 0	
1 2 3	
1 1 2 3 3 2	
4 6 0	
3 1 4 2	
3 1 1 2 3 4	

Note

For the first test case, you do not need to move the members as both slides are presented by member 1, who is already at the front of the line.

For the second test case, the following is a possible way to move members so that the presentation is good:

1. [1, 2, 3], do not move member 1.
2. [1, 2, 3], move member 1 after member 3.
3. [2, 3, 1], move member 2 after member 3.
4. [3, 2, 1], do not move member 3.
5. [3, 2, 1], move member 3 after member 1.
6. [2, 1, 3], do not move member 2.

C2. Adjust The Presentation (Hard Version)

Input file: standard input
Output file: standard output
Time limit: 5 seconds
Memory limit: 256 megabytes

This is the hard version of the problem. In the two versions, the constraints on q and the time limit are different. In this version, $0 \leq q \leq 2 \cdot 10^5$. You can make hacks only if all the versions of the problem are solved.

A team consisting of n members, numbered from 1 to n , is set to present a slide show at a large meeting. The slide show contains m slides.

There is an array a of length n . Initially, the members are standing in a line in the order of a_1, a_2, \dots, a_n from front to back. The slide show will be presented in order from slide 1 to slide m . Each section will be presented by the member at the front of the line. After each slide is presented, you can move the member at the front of the line to any position in the lineup (without changing the order of the rest of the members). For example, suppose the line of members is $[3, 1, 2, 4]$. After member 3 presents the current slide, you can change the line of members into either $[3, 1, 2, 4]$, $[1, 3, 2, 4]$, $[1, 2, 3, 4]$ or $[1, 2, 4, 3]$.

There is also an array b of length m . The slide show is considered *good* if it is possible to make member b_i present slide i for all i from 1 to m under these constraints.

However, your annoying boss wants to make q updates to the array b . In the i -th update, he will choose a slide s_i and a member t_i and set $b_{s_i} := t_i$. Note that these updates are **persistent**, that is changes made to the array b will apply when processing future updates.

For each of the $q + 1$ states of array b , the initial state and after each of the q updates, determine if the slideshow is good.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains three integers n , m and q ($1 \leq n, m \leq 2 \cdot 10^5$; $0 \leq q \leq 2 \cdot 10^5$) — the number of members and the number of sections.

The second line of each test case contains n integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq n$) — the initial order of the members from front to back. It is guaranteed that each integer from 1 to n appears exactly once in a .

The third line of each test case contains m integers b_1, b_2, \dots, b_m ($1 \leq b_i \leq n$) — the members who should present each section.

Each of the next q lines contains two integers s_i and t_i ($1 \leq s_i \leq m$, $1 \leq t_i \leq n$) — parameters of an update.

It is guaranteed that the sum of n , the sum of m and the sum of q over all test cases do not exceed $2 \cdot 10^5$ respectively.

Output

For each test case, output $q + 1$ lines corresponding to the $q + 1$ states of the array b . Output "YA" if the slide show is good, and "TIDAK" otherwise.

You can output the answer in any case (upper or lower). For example, the strings "yA", "Ya", "ya", and "YA" will be recognized as positive responses.

Standard Input	Standard Output
3	YA
4 2 2	TIDAK
1 2 3 4	YA
1 1	YA
1 2	TIDAK
1 1	YA
3 6 2	TIDAK
1 2 3	YA
1 1 2 3 3 2	YA
3 3	
2 2	
4 6 2	
3 1 4 2	
3 1 1 2 3 4	
3 4	
4 2	

Note

For the first test case, you do not need to move the members as both slides are presented by member 1, who is already at the front of the line. After that, set $b_1 := 2$, now slide 1 must be presented by member 2 which is impossible as member 1 will present slide 1 first. Then, set $b_1 = 1$, the b is the same as the initial b , making a good presentation possible.

D. Boss, Thirsty

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

Pak Chanek has a friend who runs a drink stall in a canteen. His friend will sell drinks for n days, numbered from day 1 to day n . There are also m types of drinks, numbered from 1 to m .

The profit gained from selling a drink on a particular day can vary. On day i , the projected profit from selling drink of type j is $A_{i,j}$. Note that $A_{i,j}$ can be negative, meaning that selling the drink would actually incur a loss.

Pak Chanek wants to help his friend plan the sales over the n days. On day i , Pak Chanek must choose to sell **at least** one type of drink. Furthermore, the types of drinks sold on a single day must form a subarray. In other words, in each day, Pak Chanek will select i and j such that $1 \leq i \leq j \leq m$. Then all types of drinks between i and j (inclusive) will be sold.

However, to ensure that customers from the previous day keep returning, the selection of drink types sold on day i ($i > 1$) must meet the following conditions:

- At least one drink type sold on day i must also have been sold on day $i - 1$.
- At least one drink type sold on day i must **not** have been sold on day $i - 1$.

The daily profit is the sum of the profits from all drink types sold on that day. The total profit from the sales plan is the sum of the profits over n days. What is the maximum total profit that can be achieved if Pak Chanek plans the sales optimally?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 1000$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \leq n \leq 2 \cdot 10^5$; $3 \leq m \leq 2 \cdot 10^5$; $n \cdot m \leq 2 \cdot 10^5$) — the number of rows and columns in a grid.

The next n lines of each test case contain m integers each, where the i -th line contains the integers $A_{i,1} A_{i,2}, \dots, A_{i,m}$ ($-10^9 \leq A_{i,j} \leq 10^9$) — project profits of each drink type on the i -th day.

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


















Output

















For each test case, output a single integer: the maximum profit that Pak Chanek can achieve.


















Standard Input	Standard Output
1 3 6 79 20 49 5 -1000 500 -105 9 109 24 -98 -499 14 47 12 39 23 50	475

Note

Here is Pak Chanek's optimal plan:

		Drinks:					
		1	2	3	4	5	6
Days:	1	 79	 20	 49	 5	 -1000	 200
	2	 -500	 9	 109	 24	 -40	 -400
	3	 14	 47	 12	 39	 23	 50
Profit:		79 + 20 + 49 = 148					

		Drinks:					
		1	2	3	4	5	6
Days:	1	 79	 20	 40	 5	 -1000	 200
	2	 -500	 9	 109	 24	 -40	 -400
	3	 14	 47	 12	 39	 23	 50
Profit:		9 + 109 + 24 = 142					

		Drinks:					
		1	2	3	4	5	6
Days:	1	 79	 20	 40	 5	 -1000	 200
	2	 -500	 9	 109	 24	 -40	 -400
	3	 14	 47	 12	 39	 23	 50
Profit:		14 + 47 + 12 + 39 + 23 + 50 = 185					

- On day 1, Pak Chanek sells drink types 1 to 3. Generating a profit of $79 + 20 + 49 = 148$.
- On day 2, Pak Chanek sells drink types 2 to 4. Generating a profit of $9 + 109 + 24 = 142$
- On day 3, Pak Chanek sells drink types 1 to 6. Generating a profit of 185.

So, the total profit of Pak Chanek's plan is $148 + 142 + 185 = 475$.

E1. Digital Village (Easy Version)

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

This is the easy version of the problem. In the three versions, the constraints on n and m are different. You can make hacks only if all the versions of the problem are solved.

Pak Chanek is setting up internet connections for the village of Khuntien. The village can be represented as a connected simple graph with n houses and m internet cables connecting house u_i and house v_i , each with a latency of w_i .

There are p houses that require internet. Pak Chanek can install servers in at most k of the houses. The houses that need internet will then be connected to one of the servers. However, since each cable has its latency, the latency experienced by house s_i requiring internet will be the **maximum** latency of the cables between that house and the server it is connected to.

For each $k = 1, 2, \dots, n$, help Pak Chanek determine the minimum **total** latency that can be achieved for all the houses requiring internet.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 100$). The description of the test cases follows.

The first line of each test case contains three integers n, m, p ($2 \leq n \leq 400$; $n - 1 \leq m \leq 400$; $1 \leq p \leq n$) — the number of houses, the number of cables and the number of houses that need internet.

The second line of each test case contains p integers s_1, s_2, \dots, s_p ($1 \leq s_i \leq n$) — the houses that need internet. It is guaranteed that all elements of s are distinct.

The i -th of the next m lines of each test case contains three integers u_i, v_i , and w_i ($1 \leq u_i < v_i \leq n$; $1 \leq w_i \leq 10^9$) — the internet cable connecting house u_i and house v_i with latency of w_i . It is guaranteed that the given edges form a connected simple graph.

It is guaranteed that the sum of n^3 and the sum of m^3 do not exceed 10^8 .

Output

For each test case, output n integers: the minimum total latency that can be achieved for all the houses requiring internet **for each** $k = 1, 2, \dots, n$.

Standard Input	Standard Output
2 9 8 5 2 5 6 8 9 1 2 1 1 3 2 3 4 10 4 5 3 4 6 5 1 7 10	34 19 9 4 0 0 0 0 0 2 0 0

7	8	4
7	9	2
3	3	2
3	1	
1	2	1
2	3	3
1	3	2

Note

In the first test case for $k = 3$, a possible optimal solution is to install servers at vertices 2, 6 and 8 and obtain the following latency:

- $\text{latency}(2) = 0$
- $\text{latency}(5) = \max(3, 5) = 5$
- $\text{latency}(6) = 0$
- $\text{latency}(8) = 0$
- $\text{latency}(9) = \max(2, 4) = 4$

So the total latency is 9.

E2. Digital Village (Hard Version)

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

This is the hard version of the problem. In the three versions, the constraints on n and m are different. You can make hacks only if all the versions of the problem are solved.

Pak Chanek is setting up internet connections for the village of Khuntien. The village can be represented as a connected simple graph with n houses and m internet cables connecting house u_i and house v_i , each with a latency of w_i .

There are p houses that require internet. Pak Chanek can install servers in at most k of the houses. The houses that need internet will then be connected to one of the servers. However, since each cable has its latency, the latency experienced by house s_i requiring internet will be the **maximum** latency of the cables between that house and the server it is connected to.

For each $k = 1, 2, \dots, n$, help Pak Chanek determine the minimum **total** latency that can be achieved for all the houses requiring internet.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2000$). The description of the test cases follows.

The first line of each test case contains three integers n, m, p ($2 \leq n \leq 5000$; $n - 1 \leq m \leq 5000$; $1 \leq p \leq n$) — the number of houses, the number of cables, and the number of houses that need internet.

The second line of each test case contains p integers s_1, s_2, \dots, s_p ($1 \leq s_i \leq n$) — the houses that need internet. It is guaranteed that all elements of s are distinct.

The i -th of the next m lines of each test case contains three integers u_i, v_i , and w_i ($1 \leq u_i < v_i \leq n$; $1 \leq w_i \leq 10^9$) — the internet cable connecting house u_i and house v_i with latency of w_i . It is guaranteed that the given edges form a connected simple graph.

It is guaranteed that the sum of n and the sum of m do not exceed 5000.

Output

For each test case, output n integers: the minimum total latency that can be achieved for all the houses requiring internet **for each** $k = 1, 2, \dots, n$.

Standard Input	Standard Output
2 9 8 5 2 5 6 8 9 1 2 1 1 3 2 3 4 10 4 5 3 4 6 5 1 7 10	34 19 9 4 0 0 0 0 0 2 0 0

7	8	4
7	9	2
3	3	2
3	1	
1	2	1
2	3	3
1	3	2

Note

In the first test case for $k = 3$, a possible optimal solution is to install servers at vertices 2, 6 and 8 and obtain the following latency:

- $\text{latency}(2) = 0$
- $\text{latency}(5) = \max(3, 5) = 5$
- $\text{latency}(6) = 0$
- $\text{latency}(8) = 0$
- $\text{latency}(9) = \max(2, 4) = 4$

So the total latency is 9.

E3. Digital Village (Extreme Version)

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

This is the extreme version of the problem. In the three versions, the constraints on n and m are different. You can make hacks only if all the versions of the problem are solved.

Pak Chanek is setting up internet connections for the village of Khuntien. The village can be represented as a connected simple graph with n houses and m internet cables connecting house u_i and house v_i , each with a latency of w_i .

There are p houses that require internet. Pak Chanek can install servers in at most k of the houses. The houses that need internet will then be connected to one of the servers. However, since each cable has its latency, the latency experienced by house s_i requiring internet will be the **maximum** latency of the cables between that house and the server it is connected to.

For each $k = 1, 2, \dots, n$, help Pak Chanek determine the minimum **total** latency that can be achieved for all the houses requiring internet.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains 3 integers n, m, p ($2 \leq n \leq 2 \cdot 10^5$; $n - 1 \leq m \leq 2 \cdot 10^5$; $1 \leq p \leq n$) — the number of houses, the number of cables, and the number of houses that need internet.

The second line of each test case contains p integers s_1, s_2, \dots, s_p ($1 \leq s_i \leq n$) — the houses that need internet. It is guaranteed that all elements of s are distinct.

The i -th of the next m lines of each test case contains three integers u_i, v_i , and w_i ($1 \leq u_i < v_i \leq n$; $1 \leq w_i \leq 10^9$) — the internet cable connecting house u_i and house v_i with latency of w_i . It is guaranteed that the given edges form a connected simple graph.

It is guaranteed that the sum of n and the sum of m do not exceed $2 \cdot 10^5$.

Output

For each test case, output n integers: the minimum total latency that can be achieved for all the houses requiring internet **for each** $k = 1, 2, \dots, n$.

Standard Input	Standard Output
2 9 8 5 2 5 6 8 9 1 2 1 1 3 2 3 4 10 4 5 3 4 6 5 1 7 10	34 19 9 4 0 0 0 0 0 2 0 0

7	8	4
7	9	2
3	3	2
3	1	
1	2	1
2	3	3
1	3	2

Note

In the first test case for $k = 3$, a possible optimal solution is to install servers at vertices 2, 6 and 8 and obtain the following latency:

- $\text{latency}(2) = 0$
- $\text{latency}(5) = \max(3, 5) = 5$
- $\text{latency}(6) = 0$
- $\text{latency}(8) = 0$
- $\text{latency}(9) = \max(2, 4) = 4$

So the total latency is 9.