Maratona de Programação da SBC 2021

This problem set is used in simultaneous contests:
Maratona de Programação da SBC 2021
Tercera Fecha Gran Premio de México 2021
Tercera Fecha Gran Premio de Centroamérica 2021
Torneo Argentino de Programación 2021

October 30th, 2021

Problems book

General Information

This problem set contains 14 problems; pages are numbered from 1 to 23, without considering this page. Please, verify your book is complete.

A) Program name
1) Solutions written in C/C++ and Python, the filename of the source code is not significant, can be any name.
2) Solutions written in Java, filename should be: *problem_code*`.java` where *problem_code* is the uppercase letter that identifies the problem. Remember in Java the main class name and the filename must be the same.
3) Solutions written in Kotlin, filename should be: *problem_code*`.kt` where *problem_code* is the uppercase letter that identifies the problem. Remember in Kotlin the main class name and the filename must be the same.

B) Input
1) The input must be read from *standard input*.
2) The input is described using a number of lines that depends on the problem. No extra data appear in the input.
3) When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input.
4) Every line, including the last one, ends with an end-of-line mark.
5) The end of the input matches the end of file.

C) Output
1) The output must be written to *standard output*.
2) When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output.
3) Every line, including the last one, must end with an end-of-line.

Problem A

Assigning Prizes

A programming competition will be held in Nlogonia to determine who is the best Nlogonian programmer of all time.

The competition will have N contestants and there are no ties, in other words, every contestant will be ranked in a position from 1 to N and all ranks are distinct. Lower ranks represent better results.

The event organizers decided that each contestant will receive a prize of at most R rating points, and, to be fair to the competitors that performed better, a contestant will never receive fewer rating points than any other contestant with a worse ranking.

Some competitors, though, are more greedy and want to receive more rating points to be happy. A contestant with rank i needs to receive a prize of at least p_i rating points to be happy.

Ina, a very curious organizer, is wondering in how many ways it is possible to distribute the prizes to the contestants in order to satisfy the organization's conditions and make all contestants happy. Also, since this number can be very large, you should calculate it modulo $10^9 + 7$.

Two ways are different if at least one contestant receives a different prize amount.

Input

The first line contains two integers N and R ($1 \leq N \leq 5000$, $1 \leq R \leq 10^9$), representing the number of contestants and the maximum amount of rating points that each contestant can receive as a prize, respectively.

The second line contains N integers, p_i ($1 \leq p_i \leq 10^9$), representing the minimum rating points the contestant ranked i needs to receive as a prize to be happy.

Output

Print the number of different ways to distribute the prizes modulo $10^9 + 7$.

Input example 1	**Output example 1**
2 5 4 1	9

Input example 2	**Output example 2**
3 10 7 1 10	1

Problem B

Beautiful Words

You are given a string A of length N and a set S, containing M strings.

A cyclic permutation B_i of A, in which i is between 1 and N, is the string

$$B_i = A_i A_{i+1} \cdots A_{N-1} A_N A_1 A_2 \cdots A_{i-2} A_{i-1}$$

and its score is defined as the maximum length of a substring of B_i that is also a substring of some string in S.

A substring is defined as a contiguous sequence of letters. For example, `ab` and `dc` are substrings of `abfdc`, but `ad` and `fc` aren't substrings of `abfdc`.

Your task is to calculate the minimum score over all cyclic permutations of string A.

Input

The first line contains two positive integers N and M, ($1 \leq N \leq 10^5$, $1 \leq M \leq 10^4$), representing the length of the string A and the size of the set S, respectively.

The second line contains the string A.

Each of the next M lines contains one string s_i, representing the i-th string in S.

All strings contain only lowercase English letters and it's guaranteed that the sum of lengths of all strings in S never exceeds 10^5 characters.

Output

Output an integer representing the minimum score over all cyclic permutations of string A.

Input example 1	**Output example 1**
7 3 acmicpc acm icpc maratona	3

Input example 2	**Output example 2**
11 4 competition oncom petition ztxvu fmwper	5

Input example 3	**Output example 3**
12 4 latinamerica zyvu okp wsgh kqpdb	0

Problem C

Creating Multiples

Malba is a very smart kid who loves to perform calculations. He has won several competitions, including the prestigious Tahan competition, in which he got the first prize, representing his country, Logonia.

He created a puzzle, in which he considers a number N, written in a certain base B, and represented by L digits. The objective of the game is to reduce at most one of the digits so that the new number, M, be a multiple of the number $B + 1$. But there is a catch: among the possible solutions you must choose one that renders M the smallest possible value.

For example, suppose that $B = 10$ and $N = 23456$. There are two ways in which M may be obtained: either we reduce the digit 4 to 0 or we reduce the digit 6 to 2. Thus, 4 must be changed to 0, hence $M = 23056$. Sometimes there is no solution, as is the case if $B = 10$ and $N = 102$. In this case, if we change the digit 1 to 9 we get a multiple of 11, but we are not allowed to increase the value of a digit!

Observe that it may be necessary to reduce the first digit to 0. For example, this is the case if $B = 10$ and $N = 322$.

Can you tell which digit should be reduced and what is its new value?

Input

The first line contains two integers B and L ($2 \leq B \leq 10^4$, $1 \leq L \leq 2 \times 10^5$), representing the base and the number of digits of the number N, respectively.

The second line contains L integers $D_1, D_2, \ldots, D_L$ ($0 \leq D_i < B$ for $i = 1, 2, \ldots, L$), representing the digits of the number N. The first digit, D_1, is the most significant and the last, D_L, is the least significant.

Output

Output a line containing two integers, separated by one space. The first integer is the index of the digit to be changed (recall that the index of the first digit, D_1, is 1 and the index of the last digit, D_L, is L). The second integer is the new value of the digit. If there is no solution to the problem, output `-1 -1`. If N is already a multiple of $B + 1$ then output `0 0`.

Input example 1	**Output example 1**
`10 5` `2 3 4 5 6`	`3 0`

Input example 2	**Output example 2**
`10 3` `1 0 2`	`-1 -1`

Input example 3	**Output example 3**
`2 5` `1 0 1 1 1`	`4 0`

Input example 4	**Output example 4**
`17 5` `3 0 0 0 0`	`1 0`

Input example 5	**Output example 5**
16 4 15 0 13 10	1 14

Input example 6	**Output example 6**
16 5 1 15 0 13 10	0 0

Problem D

Dividing the Kingdom

The kingdom of Nlogonia has historically been a very wealthy and quiet place. However, the current circumstances could bring this era of peace and prosperity to an end: The king is a father of two twins, so both of them are heirs to the throne.

The twins don't get along well and are jealous and overly competitive towards each other. Due to this, having both of them rule over the kingdom cooperatively is not a viable option. The kingdom will have to be divided into two independent principalities so that each of them can be given to each prince. Also, the division needs to be totally fair to avoid conflict between the envious brothers.

The kingdom consists of N cities and M roads connecting pairs of cities. The Nlogonians are peculiarly proud of their roads. Each road has an associated positive value which represents its beauty.

The kingdom will be divided in this manner: First, the cities will be partitioned in two sets such that every city is in one and only one set. Then, each principality will consist of the cities in one set and the roads connecting cities of this same set. Roads that connect cities of different principalities will be destroyed, as the princes are not interested in trading with each other, and keeping the roads would only make war more likely.

The beauty of a principality is defined as the maximum beauty of the roads within the principality, or 0 (zero) if the principality has no roads at all. For obvious reasons, the king would like the beauty of both principalities to be the same.

Help the king determine all the possible values of the beauty of the resulting principalities, given that the division is made in such a way that the principalities are equally beautiful.

Input

The first line contains two integers N, M ($1 \leq N, M \leq 5 \times 10^5$), representing the number of cities and the number of roads respectively.

Each of the next M lines contains three integers x_i, y_i, b_i ($1 \leq x_i < y_i \leq N$, $1 \leq b_i \leq 10^9$), representing that there's a road which connects cities x_i and y_i and has beauty b_i. There's no two roads connecting the same pair of cities.

Output

If it's not possible to divide the kingdom so that both principalities have the same beauty, output a line with the string "`IMPOSSIBLE`". Otherwise, output all the possible values for principality beauty resulting from divisions in principalities with equal beauty. Values should be outputted in ascending order, each in its own line.

Input example 1	**Output example 1**
9 7 1 2 3 2 3 3 3 4 3 1 3 2 2 4 2 6 7 1 8 9 1	2 3

Input example 2	**Output example 2**
4 4 1 2 5 2 3 6 1 3 7 3 4 7	IMPOSSIBLE

Input example 3	**Output example 3**
2 1 1 2 10	0

Problem E

Escalator

You have just invented a new type of escalator: the double escalator. Regular escalators take people from one endpoint to another but not in the other direction, while the double escalator can take people from any one of its endpoints to the other one.

It takes 10 seconds for the double escalator to take a person from any of its endpoints to the other one. That is, if a person enters the double escalator from one of the endpoints at moment T, then they will leave at the other endpoint at moment $T + 10$ – this person won't be using the double escalator anymore at moment $T + 10$.

Any time that no one is using the double escalator, it will stop immediately. Thus, it is initially stopped.

When the double escalator is stopped and a person enters it from one of its endpoints, it will turn on automatically and move in the direction that this person wants to go.

If a person arrives at the double escalator and it is already moving in the direction that they want to go, they will enter it immediately. Otherwise, if it's moving in the opposite direction that they want to go, they will wait until it stops, and only then will they enter it. The escalator is so large that it can accommodate many people entering it at the same time.

The double escalator has a very weird effect, probably related to some quantum physics effect (or just chance): no person will ever arrive on the double escalator at the exact moment the escalator stops.

Now that you know how the double escalator works, you will be given the task of simulating it. Given the information about N people, including their time of arrival at the escalator and which direction they want to go, you have to figure out the last moment that the escalator stops.

Input

The first line contains one integer N ($1 \leq N \leq 10^4$), representing the number of people that will use the double escalator.

Each of the next N lines contains two integers t_i and d_i ($1 \leq t_i \leq 10^5$, $0 \leq d_i \leq 1$), representing the time that the i-th person will arrive at the escalator and which direction they want to go. If d_i is equal to 0, they want to go from the left to the right endpoint, and if d_i is equal to 1, they want to go from the right to the left endpoint. All values of t_i are distinct and will be given in ascending order.

Output

Output one line containing the time that the last person will leave the double escalator.

Input example 1	**Output example 1**
3 5 0 8 0 13 0	23

Input example 2	**Output example 2**
3 5 0 7 1 9 0	29

Input example 3	**Output example 3**
3 5 0 10 1 16 0	35

Problem F

Freedom from Prison

Michael and his brother Lincoln are unfairly imprisoned in the same prison, but Michael has a plan to rescue his brother. The prison can be seen as a set of convex polygons in the plane, in which the edges of the polygons are walls. The walls of distinct polygons do not intersect, but polygons may be nested, that is, inside one another. Michael and Lincoln can be seen as two points in the plane. The rescue path will be for Michael to reach his brother and then both of them need to escape the prison.

Walking for them is no problem, but climbing walls is dangerous and difficult, so Michael will try to minimize the total number of walls climbed by him. So Michael first needs to climb some walls to reach his brother if they are not within the same area and then climb some more walls to leave the prison. Leaving the prison means not being within any walls, which can be seen as reaching a point very far away, let's say $(10^{20}, 10^{20})$. Brad is in charge of the placements of the prisoners and is aware of the plan, so he will place both prisoners in two different points in the plane not contained by any segments and such that the minimum amount of walls that need to be climbed by Michael is maximum. What is the minimum amount of walls to be climbed if Brad places the brothers optimally?

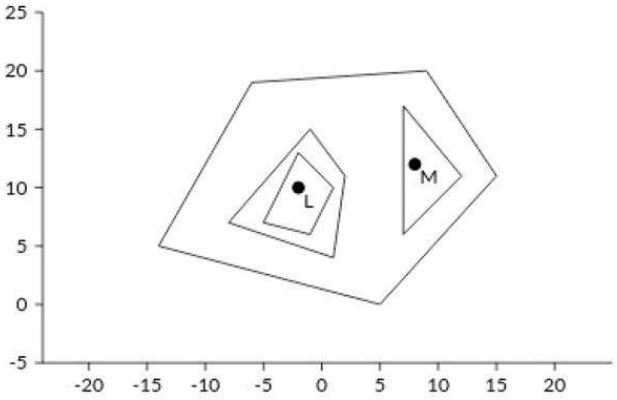

Figure 1: Illustration for valid placement in Example 1. Point M represents Michael and point L represents Lincoln

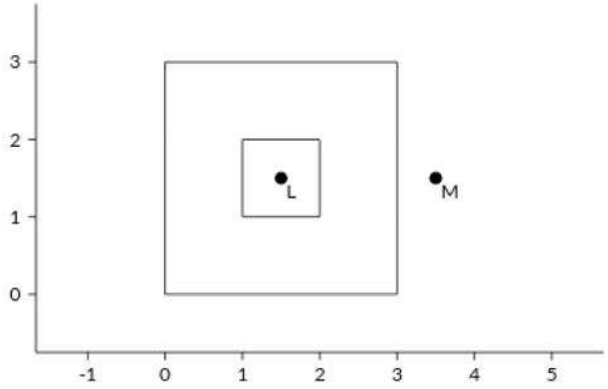

Figure 2: Illustration for valid placement in Example 2

Input

The first line of the input contains one integer N ($1 \leq N \leq 2 \times 10^5$), the number of convex polygons. This line is followed by the descriptions of each polygon. The i-th description starts with an integer k_i ($3 \leq k_i \leq 6 \times 10^5$) followed by k_i lines, each line contains a point (x_j, y_j) ($-10^9 \leq x_j, y_j \leq 10^9$).

The points in the order they are given form a convex polygon in counter-clockwise order and no three consecutive points from the polygon are collinear. No two edges from different polygons intersect. The total number of edges does not exceed 6×10^5, that is $\sum_{i=1}^{N} k_i \leq 6 \times 10^5$.

Output

Print one integer, the minimum number of walls that need to be climbed by Michael to rescue his brother, supposing that Brad assigned the brothers places so that such number of walls is maximum.

Input example 1	**Output example 1**
4 4 1 10 -2 13 -5 7 -1 6 5 15 11 9 20 -6 19 -14 5 5 0 4 -1 15 -8 7 1 4 2 11 3 7 17 7 6 12 11	6

Input example 2	**Output example 2**
2 4 0 0 3 0 3 3 0 3 4 1 1 2 1 2 2 1 2	4

Problem G

Getting in Shape

Juan decided to start working out and is willing to prepare a workout session.

He knows that some days he might not want to do all exercises from his workout session. He decided on some rules to avoid skipping the whole session and not exercising at all, while still allowing him to optionally skip some exercises.

The rules are:

- There will be only two types of exercises: A and B.
- After finishing an exercise of type B he moves to the next exercise, if there is one. Otherwise, the workout session ends.
- After finishing an exercise of type A there are two possibilities: he can move to the next exercise, or he can skip the next exercise and move to the one after that.
- The last exercise in a workout session must always be of type B.

Therefore, there might be different ways in which the workout session can be completed. For example, if the types of exercises in a workout session are `BAAB`, there are 3 ways in which the session can be completed: by doing all exercises, by skipping the 3rd one or by skipping the last one.

Juan wants to prepare his workout session in such a way that there are exactly N different ways in which the workout session can be completed. Can you help him?

Input

One positive integer N ($2 \leq N \leq 10^{15}$), representing the number of ways in which the workout session can be completed.

Output

Output a line containing a string, formed only with characters '`A`' and '`B`', representing the types of the exercises in the workout session. If there are multiple valid answers, output the lexicographically smallest answer. If there is no valid workout sessions, output a line containing the string "`IMPOSSIBLE`" (without quotes).

Input example 1	**Output example 1**
2	AB

Input example 2	**Output example 2**
4	ABAB

Input example 3	**Output example 3**
7	IMPOSSIBLE

Problem H

Handling the Blocks

A friend of yours invented a game and wants to know if you can solve it or if it's impossible.

He assembled a sequence of N blocks. Each block has a number engraved on it and some color. All numbers are distinct and between 1 and N, and different blocks can be of the same color.

The game works as follows: you can play as many turns as you want. In one turn, you choose two different blocks that share the same color and swap them.

You have to tell whether it is possible to get the entire sequence to be sorted into ascending order by numbers engraved on the blocks.

Input

The first line contains two integers N and K ($1 \leq N \leq 10^5$, $1 \leq K \leq N$), representing the number of blocks in the sequence and the number of different colors, respectively.

Each of the next N lines contains two integers n_i and c_i ($1 \leq n_i \leq N$, $1 \leq c_i \leq K$), representing the number and color of the i-th block, respectively.

Output

Output one line containing one character. If the sequence can be arranged in ascending order, write the upper case letter '`Y`'; otherwise write the uppercase letter '`N`'.

Input example 1	**Output example 1**
4 2 3 1 4 2 1 1 2 2	Y

Input example 2	**Output example 2**
4 2 2 1 4 2 1 1 3 2	N

Input example 3	**Output example 3**
3 1 1 1 2 1 3 1	Y

Problem I

Inverting Everything

The government in Nlogonia is bothered by the lack of efficiency of their railroad system. Each pair of cities is connected by a single railroad, but due to budgeting issues, some of these are inactive.

An ideal railroad configuration is one such that for every pair of cities there is exactly one path that connects those cities using only active railroads.

The Nlogonian governor hired you to transform his set of railroads in an ideal configuration. You, unfortunately, don't speak Nlogonian nor can change the activeness of the railroads: only the leader of each city, all of which only speak Nlogonian, can activate or deactivate them.

Nlogonian has some really specific phrases, so you hope you can count on that. You have a friend that speaks Nlogonian, and to help you (and also challenge you a little bit), he taught you a phrase that you can try using: "`lupDujHomwIj luteb gharghmey`". He said that if you say that to a city's leader, then the leader will activate all railroads that are connected to their city and that were previously inactive, and deactivate all railroads that are connected to their city and that were previously active. In other words, the "activation" status of all railroads that connect to that leader's city will be flipped.

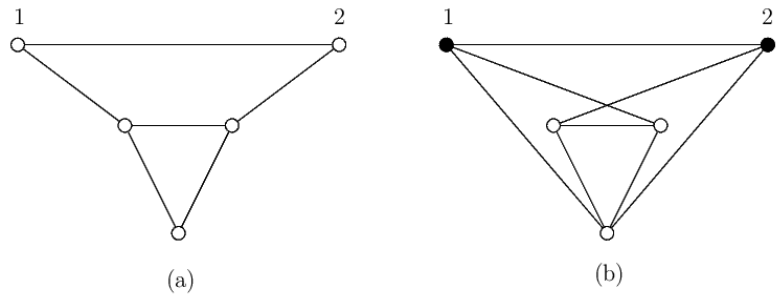

Figure 3: Result of "`lupDujHomwIj luteb gharghmey`" cities 1 and 2

Knowing this phrase, you can call some leaders and make them "`lupDujHomwIj luteb gharghmey`" their cities' railroads.

Your friends doubt that you can make the railroad system ideal by only using this phrase. You really want to prove them wrong. You said you will not only find a solution but tell in how many different ways you can achieve that. More precisely, we say a set of leaders is *good* if by contacting them and only them once, an ideal configuration is achieved. You want to tell your friend how many distinct good sets of leaders there are.

Two sets of leaders are distinct if there is at least one leader that is in one of the sets but not the other.

Since this number may be large, you must output it modulo $10^9 + 7$.

Input

The first line contains two integers, N and M ($1 \leq N \leq 100$, $0 \leq M \leq N \times (N-1)/2$), representing the number of cities and the number of railroads that are initially active.

Each of the next M lines contains two integers, u and v ($1 \leq u < v \leq N$), representing the existence of an initially active railroad between cities u and v. No pair of cities will be listed twice.

Output

Print one line consisting of the amount of *good* sets given the initial railroad configuration, modulo $10^9 + 7$.

Input example 1	**Output example 1**
5 6 1 2 1 3 2 4 3 4 3 5 4 5	8

Input example 2	**Output example 2**
3 2 1 2 2 3	6

Input example 3	**Output example 3**
4 4 1 2 2 3 3 4 1 3	4

Input example 4	**Output example 4**
3 1 1 2	0

Input example 5	**Output example 5**
2 0	2

Input example 6	**Output example 6**
10 15 1 6 1 2 1 5 6 7 6 10 2 3 2 9 7 3 7 8 3 4 8 9 8 5 4 5 4 10 9 10	0

Problem J

Just Bootfall

Football hasn't always been the most popular sport in the Americas. Historians have found records of an ancient sport that was played in many civilizations across the continent. Because of the lack of spoken tradition about it, the original name is unknown, but in modern times it has been very creatively named "bootfall".

We don't know a lot about bootfall, not even the basic rules. However, archeologists have found a lot of notes made by bootfall coaches when trying to assemble their teams, which give us some information about how teams were formed. These notes are filled with numbers and calculations. Bootfall coaches meticulously tried to optimize their team by assigning players to the best positions possible. To facilitate this task, they developed a metric for determining the performance of each arrangement.

There are M positions in a bootfall field, which are distributed in a line. A bootfall team consists of N players, each of which is assigned to some position (all players should be assigned to exactly one position, each position can be occupied by one or more players or can be left unoccupied).

Naturally, players are not equal to each other: Each player can have a different performance when playing in different positions in the field. Concretely, for each player i and each position j, there is a positive value $P_{i,j}$ which represents the performance of player i when playing in position j.

To complicate things further, coaches also consider the aspect of player interaction. Some pairs of players are "best friends". When best friends are far from each other in the field, that harms team performance. There's a positive value C which represents the performance penalty that is paid when moving best friends away from each other.

Once players are assigned to positions, the value of the team performance is calculated as follows: First, we add up the performances of the players when playing in their assigned position. Then, for each pair of players who are best friends, we subtract C times the distance between the two players, where the distance between two players is defined as the difference (in absolute value) between the positions to which the players were assigned.

We want to know how good bootfall coaches were at forming teams. In order to do that, we would like to know what is the maximum possible value of team performance achievable by arranging the players in the optimal positions, given the performances of the players in each position and the pairs of players who are best friends.

Input

The first line contains four integers N, M, K and C ($1 \leq N, M \leq 50$, $0 \leq K \leq 50$, $0 \leq C \leq 10^6$), representing the number of players, the number of positions, the number of pairs of close friends and the penalty for having close friends far from each other.

Each of the next N lines contains M integers. The j-th integer of the i-th line is $P_{i,j}$, representing the performance of player i if playing in position j ($0 \leq P_{i,j} \leq 10^6$).

Each of the next K lines contains 2 integers a_i and b_i ($1 \leq a_i < b_i \leq N$), which represent that players a_i and b_i are close friends. No two pairs of players are repeated in this list.

Output

Output a line containing one integer, representing the maximum team performance possible.

Input example 1	**Output example 1**
3 3 2 5 5 2 1 3 2 8 1 9 3 1 2 1 3	14

(In this case, the optimal solution is to assign players 1 and 3 to position 2, and player 2 to position 3, so the sum of player performances is 2+8+9=19, and we pay a penalty of 5 for players 1 and 2 being at distance 1, and penalty 0 for players 1 and 3 being at the same position).

Problem K

Kathmandu

The pandemic is getting better and you can finally do the thing you've been dreaming of for the past few years: eat at your favorite restaurant! The restaurant happens to be in Kathmandu, but that's fine, you can always take a plane.

The problem is that planes almost always leave you restless. You consider yourself properly rested if you can sleep for T uninterrupted minutes, which means you are never awake from a certain moment t to $t + T$. Also, you're a very easy sleeper: you can fall asleep at the start of any minute and wake up at the end of any minute.

Of course, if you sleep too much you will miss all the airplane meals! That is unacceptable, as no opportunity for free food should go to waste.

Luckily, the airplane company sent you the whole flight schedule: the duration of the flight, D minutes, the number of meals that are going to be served, M, and the exact time they will serve the meals, y_i. You need to be awake at the time the meal is being served to be able to eat it, otherwise, the steward will not serve you. Since you're always hungry, you will devour the meal instantly.

Now you are wondering, for the optimal plane traveling experience, can you get properly rested and still eat all meals during the flight?

Input

The first line of input contains three integers, T, D, M ($1 \leq T, D \leq 10^5$, $0 \leq M \leq 1000$), representing, respectively, the number of minutes you need to sleep without interruption to be properly rested, the duration of the flight and the number of meals that are going to be served during the flight.

Each of the next M lines contains an integer y_i ($0 \leq y_i \leq D$). These integers represent the times at which each meal is going to be served, and are given in chronological order.

Output

Output a line containing one character. If you can get properly rested and still eat all meals during the flight, write the upper case letter '`Y`'; otherwise write the uppercase letter '`N`'.

Input example 1	**Output example 1**

`3 10 3`
`2`
`4`
`7`
	`Y`

Input example 2	**Output example 2**

`4 10 3`
`2`
`4`
`7`
	`N`

Input example 3	**Output example 3**
`5 5 0`	`Y`

Input example 4	**Output example 4**
4 8 2 5 7	Y

Input example 5	**Output example 5**
4 8 2 3 4	Y

Problem L

Listing Passwords

Michael is the manager of a barely known office and inside his room there is a locker with the money to pay the employees. Unfortunately, Michael forgot the password of the locker and it is now Dwight's responsibility to help his boss.

The password is a sequence of N digits, containing only zeroes and ones, and Michael remembers the value at some positions of the sequence, but not the whole password. Michael also remembers M intervals of the password that are palindromes – his mind memorizes palindromes, for some reason.

An interval is a palindrome if, and only if, the first and last digits of the interval are equal, the second and second to last digits are equal, and so on.

Now Dwight wants to know how hard it will be to recover the whole password. You can help Dwight by calculating the number of possible passwords that follow what Michael remembers.

Since the answer may be very large, output it modulo $10^9 + 7$.

Input

The first line contains the two integers N and M ($1 \leq N \leq 3 \times 10^5$, $1 \leq M \leq 3 \times 10^5$), representing how many digits the password has and the number of intervals Michael remembers as palindromes, respectively.

The second line contains N characters s_i, representing what digit Michael remembers about each position of the password. If s_i is '0' or '1', then the i-th digit of the password is 0 or 1, respectively. If s_i is '?', then Michael doesn't remember the i-th digit.

Each of the next M lines contains two integers l_i and r_i ($1 \leq l_i \leq r_i \leq N$), which means that the interval of the password from the digit at position l_i to the digit at position r_i, inclusive, is a palindrome.

Output

Output the number of possible passwords that form a valid password modulo $10^9 + 7$. Since some of Michael's memories may be conflicting, in case there are no passwords that follow all his memories, print '0'.

Input example 1	**Output example 1**
5 2 1??0? 1 3 2 4	2

Input example 2	**Output example 2**
3 2 ??? 1 1 1 3	4

Input example 3	**Output example 3**
5 2 1???0 1 3 3 5	0

Problem M

Monarchy in Vertigo

Monarchy succession can be a tricky topic as it might take into account multiple factors such as descent, sex, legitimacy, and religion. Usually, the Crown is inherited by a sovereign's child or by a childless sovereign's nearest collateral line. Not really straightforward, right? And that's one of the reasons why, all over the world, monarchy is on the edge.

Nlogonia is still ruled by a monarchy though, luckily with simple succession rules. In general, there are only two aspects to take into account: "children comes before siblings" and "older comes before younger".

The reign servants maintain a beautiful and enormous tapestry where the bloodline of Constant, Nlogonia's first ruler, is drawn in the shape of a tree. Whenever a new family member is born a new branch from the parent to the child is drawn in the tapestry. This is such an important event that legend says that when Constant's descendants have a child they will never die before watching their child's name added to the tapestry. When someone dies, a cross is drawn close to that person's name. Whenever the current monarch dies, the tapestry is used by the servants to determine who the next ruler should be. In order to determine who that person is, the servants start from Constant and traverse the tree according to the rules described earlier, "children comes before siblings" and "older comes before younger". They descend the tree starting from Constant, followed by Constant's first child, followed by that child's first child, and so on, until reaching the first person alive or a family member that has no children left to follow, in which case they then go back to that person's parent and move into that parent's next child, repeating the process until the new monarch is determined.

After thousands of years in power Constant's bloodline is huge. To maintain the tapestry and, when the time comes, to determine who the next monarch should be are lengthy processes and Nlogonian servants decided it's time to modernize. They want to create a program to be used to maintain Constant's bloodline, which can also point who the next monarch should be after a ruler's tragic death. Given the importance of this task, the monarchy servants want to test the program by checking that it produces the correct output for all events that happened so far. There is one issue though, none of them is really good with programming and that's why they came after you for help.

More technically, each person in Constant's bloodline will be represented by a unique positive integer identifier. Whenever a new child is born, they take the next lowest unique identifier. Constant's identifier is equal to 1, and initially he's the only person alive. You will be given many events to process, in chronological order. Whenever someone dies, you should help the monarchy servants to figure out who the current monarch is. It's guaranteed that there will always be someone alive to rule.

Input

The first line contains an integer Q ($1 \leq Q \leq 10^5$), representing how many events should be processed. The following Q lines contain two integers t_i and x_i each, representing the i-th event type and argument. If t_i is equal to 1, then it means that person with identifier x_i had a new child. If t_i is equal to 2, then it means that person with identifier x_i died.

Output

For each event in which someone dies, you should print a line with an integer, representing the identifier of the current monarch.

Input example 1	**Output example 1**
8 1 1 1 1 1 2 2 1 2 4 1 2 2 2 2 5	2 2 5 3

Input example 2	**Output example 2**
4 1 1 1 1 2 2 2 1	1 3

Problem N

No Luck

Vini is a very dedicated car painter. Ever since he learned how to paint cars, his dream was to participate in the International Competition for Painting Cars (`ICPC` for short).

Every year Vini's region has a local competition to rank all teams of competitive car painters from this region. Painters in teams that were ranked in the top x spots will then go on to compete in the ICPC. It is a very thrilling competition with lots of new competitors every year, until the noxious fumes from the car paint eventually leads them to permanently retire.

Because of the variance in the national car painting budget and ICPC constraints, this number x may vary between years, which may cause a lot of displeasure in some competitors.

On Vini's last year as a contestant, his team was a single position away from qualifying to the ICPC. How unlucky! To make his "bad luck" feeling worse, in the following year the team who got the same position did qualify! Despite this feeling, after talking to other former contestants, he noticed that many have had the same feeling of being unlucky in one way or another.

Former contestants usually follow the results of the regional competition for a few years after retiring. Hence, a contestant would not feel unlucky for changes in x that occur many years after they retire. More precisely, each former contestant had their last participation in the year a_i, being placed in the position p_i and, after retiring, followed results for the next f_i years.

A contestant that didn't qualify to the ICPC in their last participation has felt unlucky on every year that they followed the results in which they would qualify to the ICPC if they had competed in that year. In other words, for every year up to f_i years after the contestant retired, if they didn't qualify in their last participation, they felt unlucky if the number of teams that qualify for the ICPC in this year was at least p_i.

Given the number of slots per year and the information about the former contestants, we want to know how many years each former contestant felt unlucky.

Input

The first line contains two integers Y and N ($1 \leq Y, N \leq 3 \times 10^5$), representing the number of years of competition and the number of former contestants that Vini had talked to, respectively. (Yes, painting cars is a millenary tradition, also a very popular one!).

The next line contains Y integers $x_1, x_2, \ldots, x_Y$ ($0 \leq x_i \leq 10^5$), representing the how many slots to the ICPC for their region there were for each year.

Each of the next N lines contains three integers a_i, p_i and f_i ($1 \leq a_i \leq Y$, $1 \leq p_i \leq 10^5$, $0 \leq f_i \leq Y - a_i$), representing the year that the i-th former competitor had their last participation, the position the i-th former competitor's team ranked and for how many years after competing the i-th former competitor followed the results after retiring, respectively.

Output

Output N lines, the i-th line should contain how many years the i-th former competitor felt unlucky.

Input example 1	**Output example 1**
5 3 1 2 3 4 5 1 3 4 2 6 3 3 4 1	3 0 1

Input example 2	**Output example 2**
4 1 8 8 8 8 1 7 3	0