Problem A. Agenda

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

Jasper has a very full agenda. He currently has n appointments, where the i'th appointment takes place in the open time interval (x_i, y_i). Since many of them are online, Jasper can attend several appointments at the same time. To make his planning less chaotic, Jasper performs q operations on his current agenda:

- split t: Jasper takes a short break at time t. Every appointment (x, y) with $x < t < y$ is split into two new appointments: (x, t) and (t, y).
- skip t: Jasper decides to skip every appointment that is ongoing at time t. So each appointment (x, y) with $x < t < y$ is removed completely.

Note that the operations are persistent.

The duration of an appointment (x, y) is defined as $y - x$. Determine the total sum of durations of all appointments still in Jasper's agenda after each operation. The input uses the "prevans xor" encoding (see Input and Output).

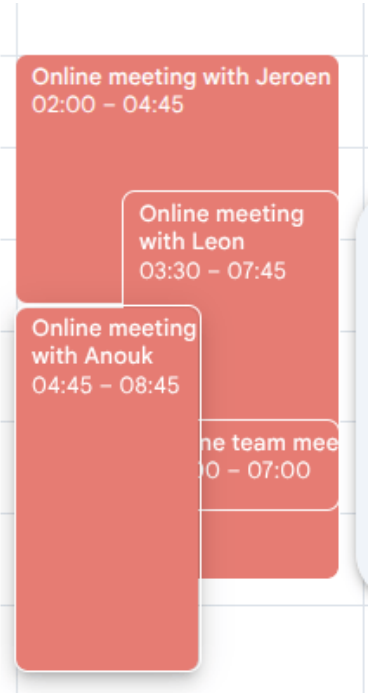


Figure 1: A normal day in the agenda of Jasper.

Input

The first line contains two integers n and q ($1 \le n, q \le 3 \cdot 10^5$), the original number of appointments and the number of operations.

The next n lines (one for each appointment) contain two integers x_i and y_i ($0 \le x_i < y_i \le 10^6$), the start and end time of appointment i.

The last q lines describe the operations. The i'th line consists of a string s_i and an integer t'_i, where s_i either equals "`split`" (denoting a split operation) or "`skip`" (denoting a skip operation). The time t_i of the operation equals $t_i = t'_i \oplus \text{ans}_{i-1}$, the bitwise XOR of the input t'_i and the answer ans_{i-1} after the previous operation. The value ans_0 is the sum of the durations of the initial n appointments.

It is guaranteed that $0 \le t_i \le 10^6$ for all i and that all t_i are distinct.

Output

Output $q+1$ integers $\text{ans}_0, \text{ans}_1, \ldots, \text{ans}_q$, where ans_i is the sum of the durations after the first i operations. The value ans_0 is the sum of the durations of the initial n appointments.

Example

standard input	*standard output*
3 3 1 10 2 5 4 13 split 22 skip 17 skip 0	21 21 12 3

Note

In the example, there are initially 3 appointments ($(1, 10)$, $(2, 5)$, and $(4, 13)$) with total duration 21. After the first operation (with $t = 22 \oplus 21 = 3$), there are 5 appointments ($(1, 3)$, $(3, 10)$, $(2, 3)$, $(3, 5)$, and $(4, 13)$) with total duration 21. The second operation (with $t = 17 \oplus 21 = 4$) will remove appointments $(3, 10)$ and $(3, 5)$, such that the total duration becomes 12. The third operation (with $t = 0 \oplus 12 = 12$) will remove appointment $(4, 13)$ as well, leaving a total duration of 3.

Problem B. Bitterballen Borrel

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

The Dutch bitterbal is a must-have at any pub; the combination of the small, round croquette and a cold glass of beer is hard to beat. Sadly, there are never enough bitterballen to go around. We will therefore have to optimize the bitterbal consumption!



Figure 2: The famous Dutch bitterbal snack, a crispy fried outer layer with a thick meat stew inside, usually served with mustard.

Waiter Leon has a plate, which can carry at most p bitterballen, and he will walk around the pub with the goal of handing the bitterballen out to guests. The evening contains a series of events, where each event is either a new batch of bitterballen that can be picked up in the kitchen, or a group of guests waiting for their deep fried snack delivery.

- If event i is a new batch being available in the kitchen, there will be b_i new bitterballen available. Leon fills up his plate with as many bitterballen as possible. The rest of the bitterballen that don't fit on Leon's plate are thrown away immediately.
- If event i is a group of guests waiting for bitterballen and Leon arrives at the group, he can hand them b_i bitterballen, and the total happiness is increased by h_i. If there are not enough bitterballen on the plate to meet the full demand b_i of the group, they will not accept any bitterballen out of politeness, and the total happiness will not change. Leon is allowed to walk past a group and thus not hand out the bitterballen, even if enough are available.

Leon is not able to be in two places at the same time. We model which events he can attend by a directed graph on the events. A directed edge from event i to event j means that Leon can make it from event i to event j. It is guaranteed that this graph will not contain any cycles, as Leon cannot travel backwards in time.

Leon starts at event 1 with a full plate of p bitterballen, and he moves between the events according to the directed edges. If Leon distributes the bitterballen in the optimal way, what is the maximum possible total happiness that can be gained?

Input

The first line of the input consists of three integers p, n and m ($1 \leq p \leq 100$, $1 \leq n \leq 1000$, $0 \leq m \leq 1000$), the maximum number of bitterballen that Leon can store on his plate, the number of specified events, and the number of movements that are possible between events.

Next, there are n lines, which specify the events. Line i (describing the i'th event) may take one of two forms:

- The character "`k`" followed by one integer b_i ($0 \leq b_i \leq 100$), representing an event to visit the kitchen. In this event, you can add at most b_i bitterballen to the plate.
- The character "`g`" followed by two integers b_i and h_i ($1 \leq b_i \leq 100$, $1 \leq h_i \leq 10^6$), representing an event to visit a group. If served, the group consumes b_i bitterballen, and the total happiness is increased by h_i.

Lastly, there are m lines, each containing two integers i, j ($1 \leq i, j \leq n$, $i \neq j$), meaning that it is possible to move from event i to event j. It is guaranteed that all ordered pairs (i, j) are distinct.

The graph with directed edges (i, j) does not contain any cycles. Note that the events themselves are not sorted in the input, so $i < j$ is not guaranteed.

Output

Output one integer, the maximum total happiness achievable when starting at event 1 with a full plate of p bitterballen.

Examples

standard input	*standard output*
50 3 2 g 20 100 g 20 50 g 20 100 1 2 2 3	200
50 4 4 g 20 25 g 30 50 g 20 75 k 20 1 2 2 3 1 4 4 2	150
50 5 5 g 20 25 g 30 50 g 20 75 k 30 g 10 20 1 2 2 3 1 4 3 5 4 2	150

Note

In the first example, there are 3 events, all corresponding to groups. All groups want 20 bitterballen, and they return 100, 50, and 100 happiness in return. Leon starts with 50 bitterballen, so he cannot serve all 3 groups. It is optimal to serve only groups 1 and 3. This can be done by following the path $1, 2, 3$ and not serving group 2 (even though this would be possible).

In the second example, there are 4 events. The first 3 events correspond to groups, and the fourth event is a visit to the kitchen. Leon starts with 50 bitterballen. He can serve all groups by following the path $1, 4, 2, 3$, and by adding 20 bitterballen to the plate in the kitchen. The maximum total happiness achievable is thus 150.

The third example resembles the second example closely, but there is an extra group reachable from event 3. If Leon would follow the path $1, 4, 2, 3, 5$ and add 30 bitterballen in the kitchen, then he could serve all groups. Unfortunately, the 30 extra bitterballen do not fit (only 50 bitterballen fit on the plate), so he cannot serve all groups. It is optimal to serve 1, 2, and 3, giving total happiness 150.

Problem C. Centenschraper

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Dutchies are known for being stingy and like to save up money. That is why, when you explained to your Dutch friends a game about placing some euro cent coins on a chess board, you could see euro signs light up in their eyes.

The game takes place on an $n \times n$ grid. On some cells of the board, a euro cent coin is placed, while other cells are empty. You can win some money by playing the game: you choose a subrectangle* of the grid and make sure that this subrectangle contains an odd number of coins. You are then allowed to grab all the coins in this subrectangle. What is the largest number of euro cent coins your friend can win after taking one subrectangle? It is guaranteed that there is at least one coin in the grid.



Figure 3: Two traditional Dutch centenschrapers, used to save a few cents worth of yoghurt left in the cup.

Input

The first line contains one integer n ($1 \le n \le 3000$), the side length of the grid.

The next n lines each contain a string of length n consisting of 0s and 1s, denoting the rows of the board, where a 1 means a euro cent coin is placed in that cell, and a 0 means that cell is empty.

It is guaranteed that there is at least one 1 in the grid.

Output

Output the maximum number of euro cent coins of a subrectangle with an odd number of coins in it.

Examples

standard input	*standard output*
3 001 100 011	3
1 1	1
2 11 11	1

*A subrectangle of an $n \times n$ grid is characterized by four numbers l, r, a, b ($1 \le l \le r \le n$, $1 \le a \le b \le n$). The subrectangle then consists of all cells c_{ij} such that $l \le i \le r$ and $a \le j \le b$.

Problem D. DigiD

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

As a Dutch citizen, you and your partner Daan have a DigiD account to do your taxes online. Daan has shared his password with you, and you have pointed out to him that his password is not very strong, so he should change his password to a new, stronger password. He asked you for advice on what makes a good password, and you have decided to prank him a little for his lack of cryptographic knowledge by giving him a ridiculous list of rules that his password should satisfy in order to be secure. You have settled on sharing the following list of ten rules:

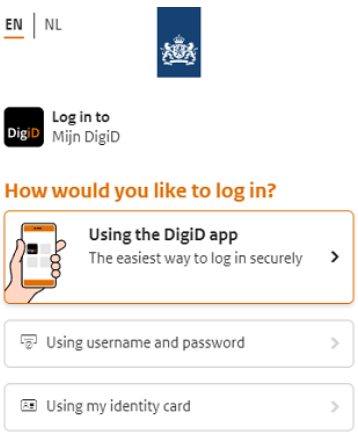


Figure 4: The log-in screen for DigiD.

In the following rules, note the difference between **substring** and **subsequence**. [†]

1. Only non-whitespace printable ASCII characters are used (ASCII codes 33 through 126).
2. At least one character is an uppercase English letter.
3. At least one character is a special ASCII character: not an English letter or a numerical digit.
4. At least one character is a lowercase English letter.
5. There needs to be a palindromic substring of length at least 3.
6. The subsequence formed by all the round bracket characters `()` must form a valid bracket sequence. A valid bracket sequence is defined recursively as either an empty sequence, the concatenation of two valid bracket sequences, or an opening bracket `(` followed by a valid bracket sequence followed by a closing bracket `)`.
7. There must not be a substring of 5 consecutive identical numerical digits.
8. The subsequence formed by all the numerical digit characters must not be monotone: there must be a strict increase somewhere, and there must be a strict decrease somewhere.
9. At least two distinct elements from the periodic table with a two-letter code consisting of one uppercase English letter followed by one lowercase English letter must occur as substrings in the password. In the cheatsheet in the attachments, all the allowed two-letter elements are given.
10. We consider a greedy algorithm for extracting Roman numerals from the password. Each time, take the leftmost character that is one of the uppercase letters `IVXLCDM`. Then, find the longest substring starting at this position that describes a Roman numeral in standard form, and add it to the list of Roman numerals (such a substring is guaranteed to exist, because the character itself is already a Roman numeral). Finally, you delete this substring and repeat until no character from `IVXLCDM` remains. Then the product of these Roman numerals must be 69, and all of the terms must be at least 2.

A complete list of two-letter elements from the periodic table and standard form Roman numerals is in the attached file `cheatsheet.md`.

[†]A substring is a contiguous sequence of characters in the string. A subsequence is obtained by deleting some (or none) of the characters from the string, and concatenating the remaining characters in their original order.

Daan is obviously overwhelmed with your insane list of rules. "Prank succeeded! HAHA!" You think to yourself, but your prank will soon backfire. Daan is very worried that he can not come up with a password that obeys all ten rules, but he will also be able to remember himself. He figured that it should be doable to remember his new password if it is an anagram of his old password, but he struggles to find a permutation that obeys all ten rules. He has given up hope of finding such a permutation, which is a big problem, because he does need to update his password, so you need to help him. You don't want to admit that this was all a prank, because that might diminish the trust in your relationship with Daan. You have thus decided that your only option is to write a program that finds a proper new password for Daan: a permutation of his old password into an anagram that obeys all ten rules.

Input

One string s ($1 \leq |s| \leq 10^5$) describing Daan's old password. It is guaranteed that Daan's old password obeys the first of the ten rules.

Output

First, one string, either `YES` or `NO`, indicating whether a proper new password for Daan is possible. If the first output string is `YES`, then follow it up with another string, describing such a proper new password for Daan.

Examples

standard input	*standard output*
`xX_ILoveSaskia2026_Xx`	`YES` `2026LXIXeSi__aakosvxx`
`password12345`	`NO`
`1234!@#$)(BHIIIIIIXXee`	`YES` `4312!#$()@BeHeIIIXXIII`
`Neapolis2010!`	`NO`

Problem E. Elfstedentocht

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

It is the 25th of January 2027, and it giet oan: the sixteenth Elfstedentocht is finally being held! The Elfstedentocht is a famous ice skating competition across all eleven historic cities in Fryslân. Being a Frisian patriot, you obviously immediately decide to participate. Unfortunately, the form to enter the competition is very unclear. It looks like it has not been used in 30 years (and 21 days)! The text consists of a string s of signs. You can determine that 26 different signs have been used, but you do not know which letter from a to z is meant by each sign (each sign corresponds to a different letter).



Figure 5: There has not actually been an Elfstedentocht in ages, due to insufficient frost.

Of course, you do not want to apply for the wrong event. Luckily, you have a plan: there are k words $w_1, w_2, \ldots, w_k$ that you expect to appear in s. To get a rough idea of how likely it is that the form is the correct one, you want to determine the total number of occurrences of the k words as substrings of s over all 26! possible assignments of signs to letters. Note that the occurrences of words (even the same word) **are** allowed to overlap, and each occurrence needs to be counted separately. As the answer can be huge, output it modulo $10^9 + 7$.

Input

The first line contains a string s ($1 \le |s| \le 10^6$), consisting only of uppercase English letters (they represent the 26 distinct symbols you cannot read).

The second line contains an integer k ($1 \le k \le 10^6$), the number of words that you expect to appear in s.

The next k lines each contain a string w_i ($1 \le |w_i| \le 10^6$), consisting only of lowercase English letters, representing the i'th word.

It is guaranteed that the total sum of lengths of all the words is at most 10^6 ($\sum_{i=1}^{k} |w_i| \le 10^6$).

Output

Output one integer, the total number of occurrences of all the words modulo $10^9 + 7$.

Examples

standard input
AABBCC 1 aa
standard output
322197157

standard input
AABBCCDEF 2 aa abc
standard output
43666263

standard input
DETOCHTFIERTLANSLJOUWERTSNITSDRYLTSSLEATSTARUMHYLPENWARKUMBOALSERTHARNSFRJENTSJERENDOKKUM 11 ljouwert snits drylts sleat starum hylpen warkum boalsert harns frjentsjer dokkum
standard output
537659403

Note

In the first example, we have only one word `aa`. This word can occur in the place of `AA` in s, assuming that `A` gets mapped to `a`. There are 25! possible ways of choosing a mapping where the `A` is already fixed. Similarly, for `BB` and `CC`, we get 25! occurrences each. The substring `AB` of s can never be an occurrence of `aa`, because `A` and `B` are known to be distinct characters, and the same holds for `BC`. In total, the number of occurrences of the word `aa` is $25! \cdot 3 \equiv 322\,197\,157 \pmod{10^9 + 7}$.

Problem F. Fatbike

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

A new epidemic has hit the Netherlands, but not one like COVID. Rather, there is an epidemic of electric bicycles with fat tires, so called "fatbikes", which terrorize the streets! To make matters worse, your good friend Leon also purchased a fatbike, and he enjoys testing out its limits.

After a long, boring shift serving **bitterballen** at the pub, waiter Leon is cycling along a straight road that is L meters long. For simplicity, we model the road as the number line from $x = 0$ to $x = L$. Leon is cycling back from $x = 0$ to his house at position $x = L$.

On the road, there is a speed limit of v $\frac{\text{meters}}{\text{second}}$, but Leon's souped-up fatbike can go way faster.

There are n speed cameras on the road. They are at locations $0 < x_1 < x_2 < \ldots < x_n \leq L$.

Leon starts with speed v at $x = 0$ and can accelerate by at most 1 $\frac{\text{meter}}{\text{second}^2}$[‡]. The fatbike has excellent brakes, so Leon can instantly decelerate to any lower non-negative speed. Note that the fatbike does not have a maximum speed.



Figure 6: Leon riding on his souped-up fatbike.

If Leon passes a camera with a speed larger than the allowed limit v, he will get a fine dropped in his mailbox. Of course, he will not pay the fine, and he will just throw it away in the bin. However, that still costs c seconds of time!

What is the minimum possible time Leon needs to get to his house and throw away all the fines he accumulated in his mailbox if he cycles optimally?

Input

The first line contains four integers n, L, v, c ($1 \leq n \leq 10^6$, $1 \leq L, v, c \leq 10^9$), the number of speed cameras, the length of the road, the speed limit, and the time it takes to throw a fine in the bin. The second line consists of n integers x_i ($0 < x_1 < x_2 < \ldots < x_n \leq L$), the positions of the speed cameras in strictly increasing order.

Output

Output the minimum time in seconds for Leon to get to his house and throw away all the fines.

Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-6}$.

[‡]Starting with speed v_0 at $x = 0$, after t seconds of maximum acceleration, his position will be $v_0 t + \frac{1}{2}t^2$, and his speed will be $v_t = v_0 + t$.

Examples

standard input	*standard output*
1 1 1 1 1	0.732050807568877
5 10 3 2 2 3 7 9 10	2.96745775725644
1 1000000000 1000000000 1000000000 1000000000	0.9999999995
1 10000 1 1 5000	141.424891727022

Problem G. Grachtenrace

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

The Netherlands is famous for its grachten, the city canals. To have some fun with the grachten, you decide to organize a grachtenrace with a group of friends. The start and finish locations are chosen, and each participant must try to go from start to finish by boat in the least amount of time possible.



Figure 7: Two speedboats chasing each other on the canals in Amsterdam.

The network consists of n junctions numbered 1 through n, with junction 1 corresponding to the start and junction n corresponding to the finish. There are m grachten, each connecting a unique unordered pair of junctions and having a given length in meters, which can be traversed in both directions. It is guaranteed that any junction can be reached from any other junction. One might assume there must exist a planar embedding of the network of grachten; this cannot be assumed due to the possibility of traversable sewers and aqueducts at heights different from the rest of the grachten.

There are docks at k junctions of the network: exactly one boat is stored at each dock. To make the race more interesting, the participants can switch boats at all docks. Each boat has a speed given in **seconds per meter**. Traveling with a boat of speed s along a gracht of length w will take $s \cdot w$ seconds. It is guaranteed that there is a dock at junction 1.

You have planned an exciting race and you are almost done with the preparations. Unfortunately, there has been an error in placing the boats: they have been mixed up completely, and as a result, the assignment of boats to docks is uniformly random. It is too late to fix this, so you decide to keep the race as it is. Determine the least amount of time in seconds possible to finish the race.

To recap: the junctions, the grachten, the docks, and the collection of boats and their speeds are all predetermined and not random, but it is guaranteed for each hidden test that the assignment of which boat is placed at which dock is picked uniformly at random. So, to emphasize, **only the assignment of the boats to the docks is random**.

Input

The first line of input contains three space-separated integers n, m, and k ($2 \le n \le 10^5$, $n-1 \le m \le 10^5$, $1 \le k \le n$), representing the number of junctions, the number of grachten, and the number of docks, respectively.

Each of the next k lines contains two space-separated integers b_i and s_i ($1 \le b_i \le n$, $1 \le s_i \le 10^6$), meaning that there is a dock at junction b_i containing a boat with speed of s_i seconds per meter. Each value of b_i is guaranteed to be unique, and exactly one value b_i equals 1.

Each of the next m lines contains three space-separated integers u_i, v_i, and w_i ($1 \le u_i < v_i \le n$, $1 \le w_i \le 10^6$), representing a gracht between junctions u_i and v_i with a length of w_i meters. Each pair (u_i, v_i) is guaranteed to be unique, and it is guaranteed that any junction can be reached from any other junction.

Additionally, it is guaranteed that, at the end of the creation of each hidden test, the values s_i were randomly permuted.

Output

Output a single integer denoting the least amount of time in seconds possible to race from junction 1 to junction n.

Examples

standard input	*standard output*
4 4 2 1 50 2 10 1 2 5 2 3 10 3 4 5 1 4 15	400
4 4 1 1 50 1 2 5 2 3 10 3 4 5 1 4 15	750

Note

In the first example, there are 2 docks at junctions 1 and 2 containing boats with speeds 50 and 10 respectively. You can first travel from junction 1 to junction 2 using the first boat, which takes $50 \cdot 5 = 250$ seconds. Then you can switch to the second boat and move via junction 3 to junction 4, which takes an additional $10 \cdot (10 + 5) = 150$ seconds, so this path will take 400 seconds in total. It can be proven that this is optimal.

Note that the path from junction 1 to junction 4 is shorter (15 meters vs 25 meters), but because it must be traversed using the slower boat with speed 50, this takes $50 \cdot 15 = 750 > 400$ seconds.

Problem H. Heineken

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

In a few weeks, the OWee will be organized, introducing the newly enrolled students to Delft University of Technology. Besides introducing the beautiful city of Delft and the different studies, there is another popular event: the introduction to Heineken, the best beer in the world. Needless to say, this event mainly consists of drinking a lot of beer.



Figure 8: The student association fair on the central market in Delft during the yearly OWee.

To automate the beer tap, the organizers of the OWee have created a network of n junctions (numbered from 1 to n) connected by m pipes to transport the beer. Junction 1 is the storage, which can produce an arbitrary flow of beer. Each junction i from 2 to n has a tap that must emit a constant flow of t_i liters of beer per second (the students will keep drinking continuously, so there is no need to have a tap handle). Each pipe has a given diameter d and can handle any flow of beer in either direction, but transporting L liters of beer per second costs $\frac{L^2}{d}$ energy. The organizers can set the amount of beer transported through each pipe arbitrarily. What is the minimum possible total energy to transport the beer from the storage so that every tap emits the required amount of beer per second?

Input

The first line of the input contains two integers n ($2 \le n \le 5000$) and m ($1 \le m \le 10^4$), the number of junctions and pipes in the network.

The second line contains $n-1$ integers $t_2, t_3, \ldots, t_n$ ($0 \le t_i \le 100$), where t_i denotes the volume of beer in liters per second that must leave the tap at junction i.

The following m lines each contain three integers u, v, and d ($1 \le u < v \le n$ and $1 \le d \le 100$) which mean that there is a pipe between junction u and junction v with diameter d.

It is guaranteed that the network is connected. Note that there can be multiple pipes between two junctions!

Output

Output the minimum possible total energy to transport the beer from the storage so that every tap emits the required amount of beer per second. Your answer should have an absolute or relative error of at most 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer will be accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

standard input	*standard output*
2 1 100 1 2 5	2000
3 3 0 100 1 2 8 2 3 8 1 3 1	2000
3 3 10 50 1 2 5 2 3 10 1 2 100	284.285714

Note

In the first example, 100 liters of beer must be transported per second from junction 1 to junction 2. There is only one pipe to transport this with width 5. The total cost is thus $\frac{100^2}{5} = 2000$.

In the second example, the network must transport 100 liters of beer per second from junction 1 to junction 3. It turns out to be optimal to send 80 liters from junction 1 via junction 2 to junction 3 and send 20 liters from junction 1 directly to junction 3. This costs $2 \cdot \frac{80^2}{8} + 20^2 = 2000$.

Problem I. Inpolderen

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

It may surprise you to learn that the youngest province in the Netherlands was only officially instated in 1986: the province of Flevoland was reclaimed from the Zuiderzee in one of the largest hydraulic projects in the world. It even made it into the Guinness Book of Records as the largest artificial island!



Figure 9: The inpolder projects the Dutch would like to perform in the coming centuries.

Due to the housing shortage, a hydraulic engineering firm has put you in charge of an even larger inpolder project next to Flevoland. There's a large lake in the middle of the Netherlands (het IJsselmeer) in the shape of a strictly convex polygon without parallel edges. The firm has been assigned by the Dutch government to build a polder[§] inside the lake. This polder can be used to build the ten new cities that the elected prime minister has promised to build in the latest elections. The firm is interested in minimizing the perimeter, because the main cost of such a project is reinforcing the perimeter to keep the water out.

Answer multiple queries of the following type: given a surface area A less than the area of the polygonal lake, determine the minimum possible perimeter enclosing a region of area A within the given lake.

Input

The first line contains two integers n and q ($3 \le n \le 50$, $1 \le q \le 50$), indicating that the polygonal lake border consists of n edges, and that there are q queries.

Each of the next n lines contains two integers x_i and y_i ($|x_i| \le 5000$, $|y_i| \le 5000$), describing the vertices of the polygonal lake in counterclockwise order.

Each of the next q lines contains one integer $A_i > 0$, indicating the surface areas for the queries.

It is guaranteed that each A_i is strictly less than the area of the given polygon. It is guaranteed that the polygon is simple and strictly convex. Moreover, it is guaranteed that no two edges of the polygon are parallel.

Output

For each query, output one real number indicating the minimum possible perimeter. Your answer should have a relative or absolute error of at most 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer will be accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

[§]A polder is an artificial land taken from the water by enclosing it and pumping it dry.

Examples

standard input	*standard output*
3 4 0 0 10 0 0 10 1 10 30 49	3.544907702 11.20998243 19.48107299 30.86382236
4 25 0 0 2000 1 2001 101 0 100 20 2000 200000 7600 7700 7800 7900 8000 8100 197700 197800 197900 198000 198100 2000 6000 10000 14000 18000 22000 60000 80000 100000 120000 140000	15.85330919 158.5330919 4187.967611 309.0378887 311.0643884 313.0777712 315.0796329 317.0796334 319.0796339 4111.089587 4113.089597 4115.089607 4117.111941 4119.184218 158.5330919 274.5873699 357.0796445 437.0796728 517.0797091 597.0797534 1357.080573 1757.081294 2157.082215 2557.083336 2957.084657

Problem J. Johan Cruijff

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

This is an interactive problem.

Who does not know Johan Cruijff? He is not only the best footballer the world has ever seen, but also a world renowned philosopher. Some of his historical quotes are "Coincidence is logical" and "If I wanted you to understand that, I would have explained it better". Some people (including himself) even regarded him as the most important export product of the Netherlands. Sadly, he died ten years ago. We shall never see his like again.

Unfortunately, this also means that the Dutch team is terrible at penalty shootouts (the last World Cup was unfortunately a perfect example). There is still hope, though, as there is another young footballer called Johan who is great at shooting penalties! Johan is obsessed with efficiency, which is why he gets bored quickly watching just a single net with a single goalkeeper. His friends decided to do something about that!

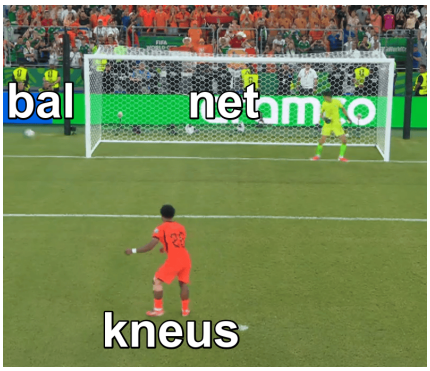


Figure 10: A missed penalty during the World Cup. Johan would also like to miss some penalties.

They play the following game. There are n nets lined up in a row for Johan to shoot penalties at, all initially unclaimed. On move i, his friends line up in front of all nets in the segment $[\ell_i, r_i]$ $(1 \le \ell_i \le r_i \le n)$, which they choose. Then, Johan shoots a penalty once at each net in the segment. Because Johan is a football prodigy, he can choose whether or not he scores as he likes. If he shoots a penalty and scores, he claims the net. If he shoots a penalty and misses, that net becomes unclaimed. The game ends whenever Johan has claimed **every net**.

Johan really likes the game, so he often misses on purpose to continue playing. However, his friends quickly grew tired of the game, so they imposed the following rule: on move i, Johan must score strictly more than $p \cdot (r_i - \ell_i + 1)$ goals, where $p = a/b$ is a rational number $(0 < p < 1)$. Despite the rule, Johan still wants the game to last at least $\lceil (1-p) \cdot n \rceil$ moves.

You will be playing as Johan against the interactor. For each move $i = 1, 2, ...$, you will receive ℓ_i and r_i, and have to decide for each net in the segment $[\ell_i, r_i]$ if you will score or not, but you have to score strictly more than $p \cdot (r_i - \ell_i + 1)$ goals on this move. You then have to choose which nets to claim and unclaim to make the game last at least $\lceil (1-p) \cdot n \rceil$ moves.

Note: An example testing tool has been provided in the attached files.

Input

The first line of the input contains three integers n, a, and b $(1 \le n \le 500,\ 1 \le a < b \le 10^6)$, the number of nets, and the numerator and denominator of the fraction $p = \frac{a}{b}$.

Interaction Protocol

Each move will proceed as follows.

On a single line, you will receive two integers ℓ_i and r_i $(1 \le \ell_i \le r_i \le n)$ representing the endpoints of the current segment.

You have to output $r_i - \ell_i + 1$ space-separated integers, each either 0 or 1, that represent whether Johan scores in the nets, from left to right, where 0 means he missed, and 1 means he scored. You must score strictly more than $p \cdot (r_i - \ell_i + 1)$ goals, or you will receive a Wrong Answer verdict. Do not forget to output the end of line and flush the output. The next segment will only be given after the previous move.

The interactor in this task is **adaptive**. In other words, the given segments may change depending on which nets are currently claimed.

The interaction only ends when all nets are claimed. If the number of moves played at that moment is less than $\lceil (1-p) \cdot n \rceil$, you will receive a Wrong Answer verdict.

It is guaranteed that at most 2000 moves will be played.

Example

standard input	*standard output*
5 2 5 1 3 2 2 4 5	 1 0 1 1 1 1

Problem K. Korfbal

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Mihail and Leon are playing a game of **korfbal**. Korfbal is played as follows: the players have n piles, each consisting of some number of **korfballen** (football-sized balls). The players take turns, with Mihail going first. In each turn, the current player selects one of the piles, chooses a non-empty subset of the korfballen in the chosen pile, and throws them all into the **korf**. The player who scores the last korfbal wins the game.



Figure 11: A game of korfbal. The rules are slightly different from those in this problem.

However, Mihail and Leon have drunk a few cans of Heineken, and they are feeling a bit tipsy. They know that, once they have selected the korfballen they will attempt to throw, there is a 50% chance that they will miss all of them at once, and a 50% chance that they will score all of them at once. If a player misses the thrown korfballen, they have to put them back onto the pile where they came from.

Since Mihail and Leon are very competitive, they will devise an optimal strategy to maximize their respective chances of winning the game. Can you calculate the probability that Mihail wins if both players play optimally? Output this probability, which can be deduced to be a rational number, modulo the prime number $10^9 + 7$.

Input

The first line consists of one integer n ($1 \le n \le 2 \cdot 10^5$), the number of piles of korfballen.

The second line consists of n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$), the number of korfballen in each pile.

Output

You should output one integer p ($0 \le p < 10^9 + 7$), the probability of Mihail winning the game modulo $10^9 + 7$.

It can be proven that the probability can be written as a reduced fraction $\frac{a}{b}$, with $b \not\equiv 0 \pmod{10^9 + 7}$.

Hence, p is the unique integer that satisfies $p \cdot b \equiv a \pmod{10^9 + 7}$.

Examples

standard input	*standard output*
3 419430657 349004802 231564547	0
3 207921793 178655361 113422849	1
1 1	666666672

Problem L. Lactose Lekkernij

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes



Figure 12: KAAAAASSSS

In the Netherlands, there are some unique days, usually during the summer, when the sun actually shines for the whole day. This unique atmospheric event happened to coincide with the famous Urkerdag (Day of Urk). Not wanting to miss this opportunity of a lifetime, Anouk, Bas, and Casper decide to take the bicycle and visit Urk, a beautiful town known for its rich fishing culture and for previously being an island, until a certain inpoldering project connected it to the mainland.

While Anouk and Bas are enjoying their stay at the local festivities, Casper wanders to the marketplace in search of some questionable Dutch cuisine. Eventually, his eyes stumble upon a cheese store selling pieces of gatenkaas (holey cheese) cut into prisms with an equilateral triangular base, and he can't help but buy two such pieces. He keeps one piece for himself and shares the other with his friends. He goes up to the store owner and asks if this design is based on the similar Toblerone chocolate shape. After hearing some defensive and very heated remarks about the originality of the gatenkaas design, Casper decides to go back to his friends while enjoying his piece of gatenkaas.

Anouk and Bas are very excited to see they have gotten a gift, and they tell Casper what a great friend he is! While unwrapping the gift, they barely manage to contain their horrified expressions when seeing the piece of gatenkaas. Anouk and Bas are both lactose intolerant and would thus much rather not eat this gatenkaas, but on the other hand, they do not want to make Casper upset, so they decide to eat the gatenkaas together anyway.

Anouk and Bas decide to turn the task of eating the gatenkaas into a game. They split up the gatenkaas into n rows of cylindrical regions. When looking at the gatenkaas from above, the first row has 1 cylindrical region, the second row has 2 cylindrical regions, the third has 3, and so on. See the artistic interpretation of gatenkaas in the illustration for the second example case below.

Being lactose intolerant, both Anouk and Bas want to eat as little gatenkaas as possible. Using their Dutch senses, they both know how much volume of air in milliliters is trapped within the gatenkaas for each cylindrical region. Starting with Anouk, they take turns choosing one of the three sides of the gatenkaas and biting off the row of circular regions bordering this side. Then, after every bite, the gatenkaas still has an equilateral triangular base. The game ends when the last circular region of cheese has been eaten.

Assuming that Anouk and Bas play optimally with the goal to maximize their own volume of air eaten, what is the difference between the amounts of air eaten by Anouk and Bas in milliliters?

Input

The first line of the input contains a single integer t ($1 \le t \le 50$), the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 50$), the number of rows of cylindrical regions in the gatenkaas.

In the following n lines, the i'th line contains i integers a_{ij} ($0 \le a_{ij} \le 10^6$ for $1 \le j \le i$), the amount of trapped air in milliliters in the j'th cylindrical region of the i'th row.

It is guaranteed that the sum of n over all test cases does not exceed 50.

Output

For each test case, output a single integer: the amount of air in milliliters eaten by Anouk minus the amount of air in milliliters eaten by Bas under optimal play.

Example

standard input	*standard output*
2 2 1 1 0 3 1 2 6 3 4 5	2 7

Note

In the first test case, Anouk chooses to eat from the upper left side, eating $1 + 1 = 2$ milliliters of air. After this, Bas chooses the last remaining region, eating 0 milliliters of air. The difference in total milliliters of air eaten is thus $2 - 0 = 2$.



In the second test case, Anouk chooses to eat from the upper right side, eating $1 + 6 + 5 = 12$ milliliters of air. After this, Bas chooses to eat from the bottom side, eating $3 + 4 = 7$ milliliters of air. Lastly, Anouk eats the last remaining region, eating 2 milliliters of air. The difference in total milliliters of air eaten is thus $12 + 2 - 7 = 7$.

Problem M. Muisjes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

It is a Dutch tradition that, when a baby is born, you invite friends and family to eat beschuit met muisjes. To prepare a beschuit met muisjes, you first apply some butter to a beschuit, which is a specific kind of large round biscuit. You then sprinkle some muisjes onto the beschuit, which are aniseed-flavored round sprinkles that sometimes still have a small tail from the aniseed. The butter makes sure that the muisjes stick to the beschuit, but you still have to make sure the muisjes are not too close to the edge of the beschuit, or they might fall off.



Figure 13: A beschuit met muisjes. The muisjes come in three colors: pink (for girls), blue (for boys), and white.

Mike was invited to eat beschuit met muisjes at Jasper's baby shower. However, since Mike is a picky eater, he only wants to eat a beschuit if it contains **exactly** k muisjes. Unfortunately for Mike, Jasper will place the muisjes. Viewed from above, the muisjes are disks with a diameter of 3 millimeters (or $3 \cdot 10^{-3}$ meters). The muisjes are placed on a comically large table. When viewed from above, the table is a square with corners $(R, R), (R, -R), (-R, -R), (-R, R)$, with $R = 10^7$ meters. However, Jasper is only able to place muisjes at **distinct** coordinates within this square, and the x and y coordinates of their centers, measured in meters, must be integers.

After the muisjes are placed on the table, Mike will try to place his beschuit on the table, so that it covers exactly k muisjes, and the edge of the beschuit does not intersect any of the muisjes. After thinking for a bit, he came up with the following algorithm to place his beschuit:

for $i \leftarrow 1$ **to** 1000 **do**
  sample real numbers $x, y \in [-10^7, 10^7]$ uniformly at random
  dist $\leftarrow$ the sorted list of distances between (x, y) and the centers of the muisjes
  $L \leftarrow k$'th distance in *dist*
  $H \leftarrow (k+1)$'st distance in *dist*
  if $H - L > 3 \cdot 10^{-3}$ **then**
    return a beschuit with center (x, y) and radius $(L + H)/2$
  end if
end for
return FAIL

Jasper wants Mike's algorithm to fail, so he can have more muisjes for himself. Can you help Jasper find an arrangement of n muisjes, and a value $1 \le k < n$, so that Mike's algorithm fails to find a satisfactory beschuit?

A testing tool is given in the attachments, so you can test your solutions locally.

Input

The input is a single dash character, indicating that there is no meaningful input in this problem.

Output

The first line of the output should contain two integers n and k ($1 \le k < n \le 2.5 \cdot 10^5$), denoting the number of muisjes on the table and the number of muisjes Mike should put on his beschuit.

Then follow n lines, each containing two integers $-10^7 \le x_i, y_i \le 10^7$, denoting the x and y coordinates of the i'th muisje in meters. The coordinates should be pairwise distinct.

Example

standard input	*standard output*
-	4 2 -10 10 10 20 234 567 123 345

Note

Mike's algorithm was implemented in C++ using a Mersenne Twister seeded with a microsecond clock as a random number generator. This means that results are non-deterministic. If you feel like a solution is close to passing, submitting multiple times can be a valid strategy. We guarantee there is a solution with a very high chance of passing.

The example is given only to demonstrate the output format, it will not result in an Accepted verdict.