Problem A. Arkanoid

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

Arkanoid is a computer game in which the player bounces a ball with a moving paddle (racket). The goal is to remove all bricks from the playing field, where each brick is removed when the ball strikes it (and bounces back). All who played the game know how frustrating and time-consuming striking the last few bricks can be. It is convenient then to have a program that, for a given initial playing field configuration, determines the time required to win the game. For the purpose of this task, we assume that the player plays perfectly, i.e., always bounces the ball off the mid-point of their paddle.

The playing field has a width of m and a height of n, where m is odd and m and n are co-prime[1]. We introduce a Cartesian coordinate system on the playing field such that the bottom left corner has coordinates $(0, 0)$ and the top right corner has coordinates (m, n). For simplicity, we assume that both the ball's size and the paddle's thickness are negligible. The paddle moves along the line $y = 0$, the initial position of the ball is $(\frac{m}{2}, 0)$, and its initial velocity (vector) is $(-\frac{1}{2}, \frac{1}{2})$.

When the ball hits the paddle, an edge of the field, or any brick, it bounces back in an elastic collision. However, any brick that is hit crumbles and is immediately removed from the field. How long until all the bricks are removed?

Input

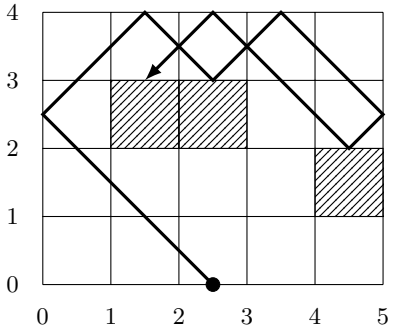
In the first line of the standard input, there are three integers, m, n, and k ($1 \le m, n, k \le 100\,000$, $k \le nm - 1$), separated by single spaces, that specify the playing field's dimensions and the initial number of bricks in it. The k lines that follow describe the bricks: the i-th such line contains a pair of integers x_i and y_i ($1 \le x_i \le m$, $1 \le y_i \le n$), separated by a single space, which signify that there is a square brick in the field whose opposite corners are at points $(x_i - 1, y_i - 1)$ and (x_i, y_i). You may assume that there is no brick in the square corresponding to $x_i = \frac{m+1}{2}$, $y_i = 1$.

Output

In the sole line of the standard output, a single integer equal to the number of time units until all the bricks are removed should be printed.

Example

standard input	standard output
5 4 3 2 3 5 2 3 3	22



[1]Two positive integers are co-prime if their greatest common divisor is 1.

Problem B. Vari-directional Streets

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

Byteasar ponders moving to Bytown and renting an apartment there. Bytown is a beautiful city with many advantages, but it is a driver's nightmare. There are n intersections in the city, interconnected by a more or less erratic network of m streets. The streets ar extremely narrow, necessitating unidirectional traffic. Surprisingly, the urban planners recently came up with a work-around that allows moving in both directions along any street without widening it. Namely, they realized that traffic need not flow both ways concurrently. And so, on odd days the traffic flows as it did since time immemorial, whereas on even days all the street directions are reversed.

Byteasar wants to rent an apartment in a well-connected location. Namely, he is interested in an apartment at such an intersection that every other intersection can be reached from it *in one day* – this may be an odd day for some destinations and even for others. The way back can be ignored, as in the worst case Byteasar can backtrack on the very next day.

Given the Bytown's road network, determine all the intersections that satisfy Byteasar's requirements.

Input

In the first line of the standard input, there are two integers n and m ($2 \leq n \leq 500\,000$, $1 \leq m \leq 1\,000\,000$), separated by a single space, specifying the number of intersections and of streets in Bytown respectively. The intersections are numbered from 1 to n. The m lines that follow describe the streets: the i-th such line contains two integers a_i and b_i ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$), separated by a single space, which indicate that there is a one-way street originally oriented from the intersection no. a_i to the intersection no. b_i (i.e., on odd days the street can be traversed from a_i to b_i, whereas on even days it can be traversed from b_i to a_i). Every ordered pair (a_i, b_i) will appear on input at most once.

Output

In the first line of the standard input a single integer k, equal to the number of intersections satisfying Byteasar's requirements, should be printed. In the second line, an increasing sequence (of length k) of those intersections' numbers, separated by single spaces, should be printed. If $k = 0$, the second line should be empty; your program may either print an empty line or not print it at all.

Example

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

Explanation of the example: From the intersection no. 1, all other intersection can be reached on odd days. From each of the intersection no. 5 and 6, all other intersections can be reached on even days. From the intersection no. 4, the intersections no. 5 and 6 can be reached on odd days, whereas the intersections no. 1, 2, and 3 on even days.

Problem C. Club members

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Byteotian Discussion Club is most extraordinary in its every aspect. Each of its 2^n members has filled out a questionnaire containing n fundamental Yes or No questions. Each member's answers can of course be encoded as a sequence of n bits, which yields an integer in the range from 0 to $2^n - 1$. We are going to ignore the specifics of questions formulations. Instead, we list some extraordinary facts about the club members below.

No two club members have given the same answers, i.e., each number in the aforementioned range is present. Moreover, exactly 2^{n-1} of the club members are men and the remaining 2^{n-1} members are women. If this were not extraordinary enough, they form in fact 2^{n-1} couples. During club sessions, the members sit at a **round** table. We would like to sit them so that every member sits between to their partner and a *nearly agreeing* member, i.e., one who answered only a single question differently.

Input

In the first line of the standard input, there is an integer n ($2 \leq n \leq 19$), specifying the number of fundamental questions. The following 2^{n-1} lines describe the member couples: the i-th such line contains two integers a_i, b_i ($0 \leq a_i, b_i \leq 2^n - 1$), separated by a single space, which indicate that the club members whose questionnaire answers are encoded by a_i and b_i are a couple. Each of the 2^n numbers representing answers is going to appear on input exactly once.

Output

A single line should be printed to the standard output, containing the word `NIE` (Polish for *no*) if no placement of club members satisfies aforementioned requirements, and a valid placement otherwise; the latter should be a sequence of 2^n integers (encoding the answers of successive members along the table), separated by single spaces.

If there is more than one correct answer, print any of those.

Example

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

Problem D. Necklace

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Bytina has n pairwise different beads, numbered from 1 to n, each of a certain known value.

She would like to make a necklace from some of her beads, but has hard time selecting the particular subset of beads and their arrangement. To simplify the choice, she decided to ignore the bead arrangement completely, i.e., deem two necklaces different only if the sets of their constituent beads differ. To make her choice even simpler, she now wants to introduce an order among all possible necklaces.

The total value of the beads is Bytina's single most important criterion. Thus, the higher this value, the later a necklace should appear in the order. However, should there be several necklaces of the same value, these should be ordered lexicographically with respect to the sequences of increasing values of their constituent beads[2].

For example, consider a scenario with four beads of values (in the bead numbers order) 3, 7, 4 and 3. 16 different necklaces can be made of these beads, listed below in Bytina's order.

Necklace no.	Value of chosen beads	Total value of chosen beads	Numbers of chosen beads
1	*none*	0	*none*
2	3	3	1
3	3	3	4
4	4	4	3
5	3 3	6	1 4
6	3 4	7	1 3
7	7	7	2
8	4 3	7	3 4
9	3 7	10	1 2
10	3 4 3	10	1 3 4
11	7 3	10	2 4
12	7 4	11	2 3
13	3 7 3	13	1 2 4
14	3 7 4	14	1 2 3
15	7 4 3	14	2 3 4
16	3 7 4 3	17	1 2 3 4

Finally, Bytina has made up her mind! She wants the k-th necklace in her order. Tell her which one that is!

Input

In the first line of the standard input, there are two integers n and k ($1 \le n, k \le 1\,000\,000$), separated by a single space, specifying the number of beads and the desired necklace number in Bytina's order. In the second line of the input, there is a sequence of n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), separated by single spaces, which specify the successive bead's values.

You may assume that Bytina made no mistake and there are indeed at least k different necklaces.

[2] A bead sequence $i_1, \ldots, i_p$ is lexicographically smaller than the sequence $j_1, \ldots, j_q$ if either the first sequence is a prefix of the second one (i.e., $p < q$, $i_1 = j_1$, $\ldots$, $i_p = j_p$), or at the first position where these two sequences differ, the first sequence has a smaller element than the second one (i.e., there exists $u \in \{1, \ldots, \min(p, q)\}$ such that $i_1 = j_1$, $\ldots$, $i_{u-1} = j_{u-1}$ and $i_u < j_u$).

Output

In the first line of the standard output, a single integer should be printed: the total value of the beads in the desired necklace. In the second line of output, the increasing sequence of values of the constituent beads should be printed, separated by single spaces.

Example

standard input	standard output
4 10 3 7 4 3	10 1 3 4

Problem E. Amusing journeys

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2.5 seconds
Memory limit:	512 mebibytes

Byteasar has discovered the joy of traveling around Bytotia recently. There are n towns (numbered from 1 to n for convenience) in the country, and the Byteotian Railways operate m bidirectional direct train connections between certain pairs of towns. Using those, Byteasar can reach any town in Byteotia, though he may have to change trains.

Our hero has a particular fondness for journeys that start and end in the same city and neither visit any other city nor use any connection more than once. Byteasar calls such journeys *amusing*.

During the latest of his numerous trips, Byteasar has noticed that all his amusing journeys so far used the same number of train connections. He suspects that, rather than a coincidence, this may be a universal property of the railway network in Byteotia, and has asked you to verify this hypothesis. Moreover, should the thesis be true, he would also like to know the number of all possible amusing journeys. For whatever reason, he does not require the exact number of those, but merely its remainder of division by $10^9 + 7$.

A journay can be formally described as a sequence of numbers of successively visited towns. Two journeys of the same length are different if there is an index i such that the i-th towns in the two sequences differ; the length of a journey is the number of train connections it uses.

Input

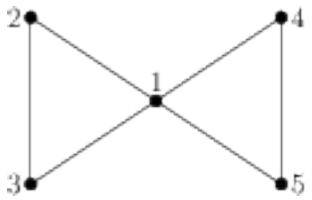
In the first line of the standard input, there are two integers n and m ($1 \leq n \leq 500\,000$, $0 \leq m \leq 1\,000\,000$), separated by a single space, which specify the number of towns and of the train connections in Byteotia respectively. Then m lines follow, describing the train connections. The i-th of these lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$), separated by a single space, which indicate that there is a bidirectional train connection between the towns no. a_i and b_i. There is at most one direct connection linking each pair of towns.

Output

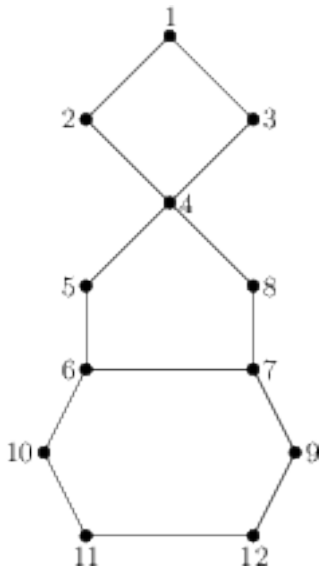
If (unfortunately) there is no amusing journey whatsoever, then the word `BRAK` (Polish for *none*) should be printed to the standard output. If such journeys exist, but they do not all have the same length (refuting Bytesar's hypothesis), then the word `NIE` (Polish for *no*) should be printed to the standard output. Finally, if all the amusing journeys have the same length (i.e., the hypothesis is true), then the word `TAK` (Polish for *yes*) should be printed to the standard output, followed with a new line with two integers separated by a single space: the common length of such journeys and their number modulo $10^9 + 7$.

Example

standard input	standard output
5 6 1 2 2 3 3 1 1 4 4 5 5 1	TAK 3 12
12 14 1 2 2 4 3 1 4 3 4 5 5 6 6 7 7 8 8 4 7 9 9 12 12 11 11 10 10 6	NIE



Explanation of the first example: There are 12 amusing journeys, all of which have length 3. These are: 1-2-3-1, 1-3-2-1, 2-1-3-2, 2-3-1-2, 3-1-2-3, 3-2-1-3, 1-4-5-1, 1-5-4-1, 4-1-5-4, 4-5-1-4, 5-1-4-5, 5-4-1-5.



Explanation of the second example: Not all amusing journeys are of the same length.

Problem F. Nim with a twist

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3.5 seconds
Memory limit:	512 mebibytes

The favorite pastime of Alice and Byteasar is playing *Nim*. The game starts with an arbitrary initial arrangement of tokens into heaps. The two players move alternately, where a single move consists in choosing an arbitrary heap and removing from it any positive number of tokens. The player who cannot make a move loses.

Alice has just proposed another game of Nim. To make it more interesting, this time they have decided to alter the rules slightly. Namely, Alice has distributed the m tokens into heaps of sizes $a_1, a_2, \ldots, a_n$, as usual. Then, before the game commences, Byteasar may remove some of the heaps from play. The number of removed heaps must be divisible by a predetermined number d, and at least one heap must remain. Afterwards, they are going to play a regular game of Nim, with Alice moving first.

Let k denote the number of valid subsets of heaps such that Byteasar can have them removed and be able to win the game afterward, regardless of Alice's moves. Your task is to provide the remainder of k divided by $10^9 + 7$.

Input

The first line of the standard input contains two integers n ($1 \leq n \leq 500\,000$) and d ($1 \leq d \leq 10$), separated by a single space, specifying the number of heaps and the divisor of the number of heaps that Byteasar may remove respectively.

The second line describes the heaps: It contains a sequence of n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 1\,000\,000$), separated by single spaces, such that a_i is the number of tokens in the i-th heap. The sum $a_1 + a_2 + \ldots + a_n$ does not exceed $10\,000\,000$.

Output

The first and only line of the standard output should contain a single integer: the number (modulo $10^9 + 7$) of different subsets of heaps that Byteasar can remove, ensuring his subsequent victory.

Example

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

Explanation of the example: Byteasar can remove either 2 or 4 heaps. He is going to win only if he removes one heap with 1 token and another one with 4 tokens; there are two such sets of heaps.

Problem G. Parade

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Every year, to celebrate the beginning of spring, the Big ByteSpring Parade marches down the streets of Byteburg. This year, his majesty Byteasar XVI will grace it by his presence. The Byteburg's street network consists of n intersections linked by $n - 1$ two-way street segments in such a way that every intersection is reachable from every other intersection.

The exact route of the parade is not determined yet, but it is known that it will start in one intersection, follow a certain number of street segments, and end in a **different** intersection. To keep the paraders entertained, the route will traverse each street segment at most once.

More importantly, to ensure the paraders' safety, every street segment such that exactly one of its endpoints is on the parade's route (including the initial and final intersection) should be closed to traffic. Your goal is to determine the maximum number of street segments that may have to be closed.

Input

In the first line of the standard input, there is a single integer n ($2 \leq n \leq 200\,000$), specifying the number of intersections in Byteburg. The intersections are numbered from 1 to n.

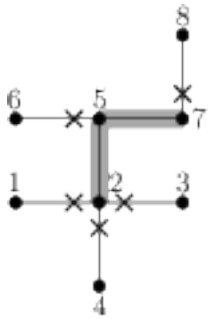
The following $n - 1$ lines describe Byteburg's street network. Each of these lines contains two integers a and b ($1 \leq a, b \leq n$, $a \neq b$), separated by a single space, which indicate that the intersections no. a and b are linked by a two-way street segment.

Output

A single integer should be printed in the first and only line of the standard output: the maximum number of street segments that may have to be closed in order to secure the parade.

Example

standard input	standard output
8 1 2 2 3 4 2 5 2 6 5 5 7 7 8	5



Explanation of the example: If the parade starts in intersection 2 and ends in intersection 7, 5 street segments will have to be closed: (2,1), (2,3), (2,4), (5,6), (7,8), where (a, b) stands for the segment linking intersections no. a and b.

Problem H. Messenger

Input file:	*standard input*
Output file:	*standard output*
Time limit:	9 seconds
Memory limit:	512 mebibytes

After a long rein over the kingdom of Byteotia, Byteasar abdicated, too exhausted to keep on ruling the country. Soon he realized that he missed being up to date on court news, policy, and intrigue. So to stay in touch, he decided to become a royal messenger.

On the very first day at his new job, he was entrusted to deliver an urgent message from one town to another. Rather than make his very first trip as timely as possible, Byteasaur decided to turn it into a tour of the country in order to recover after his years of service as the king. Naturally, he is going to take some precautions so that the new king never finds out the true nature of the messenger's itinerary.

All the roads in Byteotia are one-way only, leading directly from one town to another. Byteasar has declared the exact number of road segments he wants to follow on his trip, regardless of how many are actually required. In order not to arouse the suspicion of the royal officials, he insists on visiting each of the source and destination towns exactly once, but he is willing to visit any other town multiple times, as well as take the same road segment more than once.

Help our hero by writing a program that will determine the number of routes that satisfy his requirements. In other words, the program is to determine the number of different routes of a given length between a given pair of towns that visits each of those exactly once. As this number may be quite large, it is sufficient to return its remainder after division by a number of Byteasar's choice.

Input

In the first line of the standard input, there are three integers n, m, and z ($2 \leq n \leq 100$, $0 \leq m \leq n(n-1)$, $2 \leq z \leq 10^9$), separated by single spaces, that specify, respectively: the number of towns in Byteotia, the number of one-way roads between them, and the divisor chosen by Byteasar. The towns are numbered from 1 to n.

Next, m lines follow, each containing a pair of integers a, b ($1 \leq a, b \leq n$, $a \neq b$), separated by a single space, which indicate that there is a direct one-way road from the town no. a to the town no. b. No road segment appears in the input more than once.

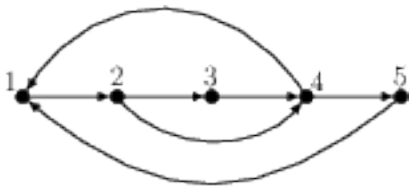
In the next line, there is an integer q ($1 \leq q \leq 500\,000$), specifying the number of Byteasar's queries. Each of the q lines that follow contains a query. The query consists of three integers u_i, v_i, and d_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$, $1 \leq d_i \leq 50$), separated by single spaces, which specify that Byteasar wants to travel from the town no. u_i to the town no. v_i by taking exactly d_i road segments.

Output

Exactly q lines should be printed to the standard output. The i-th of those should contain the remainder after division by z of the number of routes asked for by the i-th query.

Example

standard input	standard output
5 7 10 1 2 2 3 3 4 4 5 5 1 2 4 4 1 2 2 1 3 5 3 6	2 1



Explanation of the example: Two routes satisfy the first query: $2 \to 3 \to 4 \to 1$ oraz $2 \to 4 \to 5 \to 1$; and only one route satisfies the second query: $5 \to 1 \to 2 \to 4 \to 1 \to 2 \to 3$.

Problem I. Diligent Johny

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

It is Little Johny's birthday. And this is serious algorithmic problem, so the poor kid received no toys, games or a computer for his birthday present. Rather, he was presented with long arrays filled with numbers, trees, maps of strange lands rife with roads that lead through numerous tunnels and overpasses, lengthy tapes filled with 1048576 symbols long prefixes of Fibonacci and Thue-Morse words, etc. Of all these educational gifts, he likes an array holding a permutation[3] of the first n positive integers the most. Soon, Johny started wondering what is the lexicographic predecessor permutation[4] of the one he was given. Having figured that out rather quickly, Johny immediately asked himself how could he write this predecessor permutation in his array. The only operation that the array supports is selecting two cells and swapping the contents of those cells. Fortunately, Johny was smart enough to transform the initial permutation into its predecessor in the minimum number of swaps. He found this task so captivating, that he kept on transforming each successive permutation into its predecessor.

In his permutation madness, Johny is ignoring all his birthday party guests, which they find amusing enough but also a little rude. One of them soon realized that Johny will stop once he gets down to the identity permutation $1, 2, \ldots, n$, which is lexicographically smallest. The question is, how long will this take? Help them answer this question, knowing that every swap takes Johny exactly one second. As this might take a while (diligent is Johny's middle name), the guests will be happy enough to know the remainder of division by $10^9 + 7$. After all, they can check back on Johny every $10^9 + 7$ seconds to see if he is finally done.

Input

In the first line of the standard input, there is a single integer n ($1 \le n \le 1\,000\,000$), specifying the length of the permutation that Johny got for his birthday. In the second line, the permutation itself is given, as a sequence of n distinct integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$), separated by single spaces.

Output

Your program should print to the standard output the remainder of division by $10^9 + 7$ of the number of swaps that Johny will make before he stops.

Example

standard input	standard output
3 3 1 2	6

Explanation of the example: The lexicographically decreasing sequence of permutations that Johny will go through is $(2, 3, 1)$, $(2, 1, 3)$, $(1, 3, 2)$, $(1, 2, 3)$. To obtain those, he will make $2 + 1 + 2 + 1 = 6$ swaps in total.

[3] A permutation of the numbers from 1 to n is a sequence of pairwise different integers $p_1, \ldots, p_n$ satisfying $1 \le p_i \le n$ (i.e., every integer from 1 to n appears exactly once in a permutation).

[4] The permutation $P = (p_1, \ldots, p_n)$ is lexicographically smaller than the permutation $Q = (q_1, \ldots, q_n)$ (which we denote $P < Q$) if $p_j < q_j$, where j is the smallest index such that $p_j \ne q_j$. The permutation P is the lexicographic predecessor of Q if $P < Q$ and there exists no permutation R such that $P < R < Q$.

Problem J. Not Nim

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Bythony and his little brother Bytie often play the game Nim. Bythony explained the winning strategy in Nim to his younger brother, but Bytie seems incapable of applying it, as he loses quite often. Therefore, little Bytie keeps suggesting alternative rules, hoping that these would facilitate the game.

His latest proposal is as follows: there are n pairs of heaps, where each heap in the i-th pair initially has a_i pebbles. The players move alternately. Each Bytie's move consists in removing any positive number of pebbles from a heap of his choice. Each Bythony's move on the other hand consists in moving any positive number of pebbles within a pair of heaps of his choice. Bytie moves first. The player who cannot make a move loses.

Bythony was quick to notice that he cannot hope to win this game, but he agreed to play nevertheless, to make his little brother happy. In fact, he intends to delay the inevitable defeat as much as possible, i.e., make as many moves as possible before he loses. Aid him by writing a program that will determine the maximum duration of the game when both players play optimally, i.e., when Bytie aims to win in the fewest number of moves, whereas Bythony aims to make as many moves as possible before losing.

Input

In the first line of the standard input, there is a single integer n ($1 \le n \le 500\,000$), that specifies the number of pairs of heaps. In the second line, there is a sequence of n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), separated by single spaces, that specify the common size of both heaps in successive pairs.

Output

In the only line of the standard output, a single integer should be printed: the number of moves until the game is over when both brothers are playing optimally.

Example

standard input	standard output
2 1 2	7

Explanation of the example: One optimal sequence of moves is as follows:

$$1122 \to 1120 \to 1111 \to 1110 \to 1101 \to 1100 \to 2000 \to 0000$$

Problem K. Stutter (32 MiB ML!)

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	**32** mebibytes

As of recently, Bitie suffers from a strange condition: he keeps stuttering, and, moreover, the only words he utters are numbers. His older brother, Bytie, has noticed a peculiar regularity in Bitie's stutter. He suspects that Bitie is in fact simulating, so that he is excused from attending school, and may spend the time playing computer games. For Bytie, this is rather upsetting, as it prevents him from learning programming. Hence, Bytie is determined to expose his little brother for a fraud, hoping to gain as much time for programming as he desires.

Let us formalize Bytie's suspicions. Suppose we are given a sequence of numbers A.

- A *subsequence* of A is any sequence formed from A by removing arbitrary elements from it, e.g., $1, 1, 7, 5$ is a subsequence of the sequence $1, 3, 1, 7, 6, 6, 5, 5$.
- A *stutter* of A is any subsequence of A that is a concatenation of a number of pairs of equal elements, e.g., $1, 1, 1, 1, 3, 3$ can be obtained by concatenating pairs $(1, 1)$, $(1, 1)$ and $(3, 3)$ and thus is a stutter of the sequence $1, 2, 1, 2, 1, 2, 1, 3, 3$.

Bytie promises a prize for determining, for given two sequences of numbers uttered by Bitie, what is the length of their longest common stutter, i.e., a sequence that is a stutter of both sequences.

Input

The first line of the standard input contains two integers, n and m ($2 \leq n, m \leq 15\,000$), separated by a single space, which are the lengths of the sequences A and B that represent Bitie's utterances. In the second line of input, there are n integers, $a_1, a_2, \ldots, a_n$, separated by single spaces; these are the successive elements of the sequence A ($1 \leq a_i \leq 10^9$). In the third line of input, there are m integers, $b_1, b_2, \ldots, b_m$, separated by single spaces; these are the successive elements of the sequence B ($1 \leq b_i \leq 10^9$).

Output

Your program should print a single nonnegative integer to the standard output: the length of the longest common stutter of A and B. If no common stutter exists (or rather, it is empty), the correct answer is 0.

Example

standard input	standard output
7 9 1 2 2 3 1 1 1 2 4 2 3 1 2 4 1 1	4

Explanation for the example: The longest common stutter is 2, 2, 1, 1.