Problem A. Anatoly Shalyto

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

Median of a multiset of integers is the smallest integer X such that at least half of the elements of the set are less than or equal to X.

Mode of a multiset of integers is the value that occurs the most times in the multiset. If there are multiple such values the mode is the smallest.

Imbalance of a multiset is the absolute difference between the median and the mode.

A multiset T is a **subset** of a multiset S if for every value the number of its occurrences in S isn't less than the number of its occurrences in T.

You are given a multiset of integers. Consider its non-empty subset with the largest imbalance. Print that imbalance.

Input

The first line contains a single integer n ($1 \le n \le 10^5$), size of the multiset.

The second line contains n integers a_i ($0 \le a_i < 10^9$, $a_i \le a_{i+1}$, elements of the multiset.

Output

Print a single integer — the largest imbalance of some subset of the given multiset.

Examples

standard input	standard output
4 1 2 8 8	6
5 2 2 2 8 8	0
5 1 2 3 4 5	3

Problem B. Basirovich Maxim

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

Note that 0-based indexing is used throughout the problem.

You are given an array a of length n and k non-empty sets of integers from 0 to $n-1$. Let S_p denote the p-th set. Each integer from 0 to $n-1$ belongs to exactly one of those sets. It is guaranteed that 0 belongs to S_0.

You choose a nonincreasing array c of nonnegative reals. c_0 must be positive. Let d_p denote $\sum_{i \in S_p} c_i a_i$. Let X denote $\frac{\min_{p=1}^{k-1} d_p}{d_0}$. What is the maximum value of X you can obtain by choosing c accordingly?

Input

The first line contains two integers n and k ($2 \le n \le 4 \cdot 10^4$, $2 \le k \le 4$), the length of a and the number of sets.

The second line contains n integers a_i ($1 \le a_i \le 10^9$), the elements of a.

The third line contains n integers b_i ($b_0 = 0, 0 \le b_i < k$) meaning that i belongs to S_{b_i}.

All sets S_p are non-empty. In other words, all integers from 0 to $k-1$ occur at least once among b_i.

Output

Print a single integer — the maximum value of X you can obtain accurate to absolute or relative error of at most 10^{-4}.

Examples

standard input	standard output
4 4 1 1 1 1 0 1 2 3	1
3 3 3 2 2 0 1 2	0.6666666666666666668
5 4 3 2 1 2 5 0 1 3 0 2	0.29411764686215561816

Problem C. СТАНКЕВ** ANDREW

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

Cyclic shift of a sequence s by k $(0 \le k < |s|)$ positions is the following sequence $s_k s_{k+1} \dots s_{|s|-1} s_0 s_1 \dots s_{k-1}$. Note that 0-indexing is used. For example, cyclic shift by 0 positions is equal to the original sequence.

There is a permutation which is initially equal to the identity permutation $(0, 1, \dots, n-1)$.

Process a number of the following queries. Replace a subarray of the permutation by its cyclic shift by some number of positions. Formally you are given l, r, k, such that $0 \le l < r \le n,\ 1 \le k < r - l$ and replace the subarray between indices l and r (a half-interval, l-th element is included, r-th is not) by its cyclic shift by k positions. For example, applying this operation to the permutation $[1, 0, 3, 4, 2]$ with parameters $l = 1, r = 4, k = 1$ results in a permutation $[1, 3, 4, 0, 2]$.

After each query find the cyclic shift of the permutation which contains the minimal number of inversions.

The changes of the permutation persist and are not reverted between queries.

Input

The first line contains two integers n and q $(2 \le n \le 10^5, 1 \le q \le 10^5)$, length of the permutation and the number of queries, respectively.

q lines follow. i-th of them contains three integers l_i, r_i, k_i $(0 \le l_i < r_i \le n,\ 1 \le k_i < r_i - l_i)$, parameters of the i-th query.

Output

Print q lines. i-th of them should contain an integer k $(0 \le k < n)$, such that after the first i queries cyclic shift of the permutation by k positions contains the minimal number of inversions among cyclic shifts. If there are multiple possible k print the smallest one.

Examples

standard input	standard output
8 1 1 7 3	4
7 5 1 7 2 0 5 1 2 6 3 0 5 4 0 7 5	5 4 5 3 0

Note

In the first example after the modification the permutation is $[0, 4, 5, 6, 1, 2, 3, 7]$. Its cyclic shift with the minimal number of inversions is $[1, 2, 3, 7, 0, 4, 5, 6]$.

Problem D. Dr. Bill Poucher

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

There are n people. Each person sees some of the other people. Each of them will be given a black or white hat. After that each person will simultaneously name a color. Everyone who doesn't guess the color on his hat will die. *Horribly.*

Is there a deterministic strategy which guarantees that at least one person will survive?

Input

The first line contains two integers n and m ($2 \le n \le 3 \cdot 10^5, 1 \le m \le 3 \cdot 10^5$), the number of people and the number of relations of seeing someone (see below), respectively.

m lines follow. i-th of them contains two integers a_i and b_i ($0 \le a_i, b_i < n,\ a_i \ne b_i$) meaning that a_i-th person sees b_i-th person. For all $i \ne j$, $a_i \ne a_j$ or $b_i \ne b_j$ holds.

Output

Print `1` if there exists such strategy and `0` otherwise.

Examples

standard input	standard output
4 3 0 1 1 2 2 3	0
2 2 0 1 1 0	1
6 8 0 2 0 3 2 1 3 1 5 2 1 4 4 5 3 4	1

Problem E. Elena Andreeva

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

This is an interactive problem.

You are given an integer n. Your program chooses a number x between 0 and $n-1$ inclusive. The interactor asks you queries of form: what is the remainder of x modulo y, where y is the parameter of the query.

The interactor isn't stupid and won't ask you a query, such that the answer to it is known beforehand (i.e. it can be deduced from answers to previous queries and the fact that the hidden number is between 0 and $n-1$). For example, if the interactor asks you the remainder of x modulo 10 and you answer 6 the interactor won't query the remainder of x modulo 2 because it can be deduced that it is 0.

You want the interactor to know the number you've chosen in as few queries as possible. Obviously you can answer adaptively and change the chosen number on the fly, but it must remain consistent with previous answers.

At the start of interaction you choose an integer k and implicitly make a claim: ¡¡The interactor will know the chosen number in at most k queries¿¿. You should choose the smallest k, such that you can always make that happen. Precisely, let K denote the smallest possible such k.

If you choose $k > K$ the interactor will tell you that. You can either accept this fact or choose to reverse the roles. You should do it only if you believe that your solution can't be wrong (hint: it is) and there is a problem with authors solution or tests. The interactor will choose the number and you will try to guess it as slowly as possible. If you make k valid queries you will receive Check Failed and prove your point.

If $k \le K$ the guessing starts. The interactor will make queries until it knows the number. If the interactor is able to make $k+1$ queries you didn't fulfill your claim and receive WA. Otherwise you pass the test case. Note that if you choose $k < K$ the interactor will always be able to make at least $k+1$ queries by asking queries optimally (however there is no guarantee that the interactor will do so, refer to the third sample).

Interaction Protocol

You should read an integer n ($2 \le n \le 10^5$).

After that you should make your claim. Print an integer k ($1 \le k \le 10^5$). There are two possibilities:

1. If $k \le K$ the interactor prints `1`. You choose a number x and the interactor starts guessing.

   You should repeatedly read an integer y ($2 \le y \le 10^9$) and print $x \mod y$ until you encounter $y = -1$ or $y = 0$ or the interactor makes the $k+1$-th query. $y = -1$ means your answers are inconsistent. $y = 0$ means that interactor knows the number and you've passed the test. $k+1$-th query means that you didn't fulfill your claim. Your solution should terminate in all 3 cases.

2. If $k > K$ the interactor prints `0`. Repeatedly print an integer y ($2 \le y \le 10^9$) and read an integer r which is equal to $x \mod y$. Your queries should be valid (you don't know the answer beforehand). If some query is not valid the interactor will print `-1` instead of the remainder and your solution should terminate. When you know the hidden number or you simply accept the fact that your solution is wrong print `-1` and terminate instead. If you make k valid queries (this is highly unlikely) you will receive Check Failed or something along the lines.

Note that `-1` and `0` are signaling values and they are out of bounds $[2, 10^9]$ used for ordinary value of y.

Examples

standard input	standard output
3 1 15 -1	1 0
6 1 2 5 -1	2 1 0
100 1 80 -1	1 76

Note

In the first sample the process goes as follows.

- The solution reads n. It is equal to 3.
- It prints $k = 1$.
- It can be shown that $K = 1$.
- The interactor prints `1` and starts guessing.
- It makes a query with $y = 15$.
- The solutions answers 0.
- Interactor knows that the hidden number is 0 and prints `-1`.

In the second sample a similar process happens, except for $k = K = 2$. Therefore the interactor makes two queries instead of one. The hidden number is 5.

In the third example $K > 1$, but the solution prints $k = 1$. However, the interactor makes a query, such that the hidden number (76) is known after the solution answers it and the solution still passes the test. If the interactor asked a query with $y = 10$ instead the solution would've failed test because there will always remain multiple possibilities for the hidden number after the query is answered.

The actual interactor might ask different queries than ones in the samples. Therefore there is no easy way to pass the samples with some placeholder solution

Problem F. Filipp Rukhovich

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

The **palindromicity** of a sequence of characters t of length k is number of indices i, such that $0 \le i < k - i - 1$ and $t_i = t_{k-1-i}$. Note that 0-based indexing is used.

You are given a string s. Count the sum of palindromicities over all its subsequences. Sequences which occur multiple times as a subsequence are counted multiple times (i.e. you sum palindromicities over all $2^{|s|}$ subsequences whether they are distinct or not).

Output the correct answer modulo 998244353. Formally, if the actual answer is y and your answer is x, it will be considered correct if $-2^{63} \le x < 2^{63}$ and $x - y$ is divisible by 998244353.

Input

The only line contains a non-empty string s of lowercase latin letters with length not exceeding 123456.

Output

Output a single integer — sum of palindromicities over all subsequences of s modulo 998244353.

Examples

standard input	standard output
xxx	4
abacaba	80
ypiiouiuiputrhgogghjhp	1084841
dfgfhdgdhjgfdhgdfhgfdgfddfg	190900560
qweqwewqewqeqweqwqwewqrqwrrew	910048289
thosewereactuallykeyboardslaps	405649044

Problem G. Gleb Evstropov

Input file:	*standard input*
Output file:	*standard output*
Time limit:	20 seconds
Memory limit:	512 mebibytes

You are given an array a.

Process two types of queries:

1. You are given x and y. Set a_x to y.
2. You are given l, r and k. Find the largest value of m, such that sequence $k, k+1, \ldots, m$ is a subsequence of $a_{l:r}$.

Input

The first line contains two integers n and q ($1 \le n, q \le 10^6$), the length of a and the number of queries, respectively.

The second line contains n integers a_i ($0 \le a_i < n$), the elements of a.

q lines follow. Each of them has one of the following forms:

- $1\ x\ y$ ($0 \le x, y < n$), describing a query of the first type.
- $2\ l\ r\ k$ ($0 \le l < r \le n$, $0 \le k < n$), describing a query of the second type. Note that half-intervals are used, i.e. $a_{0:3}$ contains elements with indices 0, 1 and 2. It is guaranteed that the given half-interval contains at least one element equal to k.

Output

For each query of the second type print the corresponding m.

Example

standard input	standard output
6 17 0 0 0 1 2 1 2 0 4 0 2 0 5 0 1 3 2 2 0 4 0 2 0 6 0 2 0 4 2 2 5 6 1 1 0 1 2 1 6 1 2 0 5 1 1 0 0 1 5 5 1 2 2 1 4 4 1 3 3 1 1 1 2 0 6 0	1 2 0 1 2 1 1 2 5

Problem H. Hristenko Oleg

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

You are given an $n \times m$ grid of integers.

Consider the following graph. Each cell of the grid is considered as a vertex. Two cells are connected by an edge if they are in the same row or column and the cost of the edge is equal to absolute difference of numbers in its endpoints.

Consider the minimal cost spanning tree of this graph (the cost of the spanning tree is the sum of costs of edges in it). Find its cost.

Input

The first line contains two integers n and m ($1 \le n \cdot m \le 10^5$), the number of the rows and columns in the grid, respectively.

n lines follow. i-th of them contains m integers $a_{i,j}$ ($0 \le a_{i,j} < 10^9$), the elements of the grid.

Output

Print a single integer — the cost of the minimal cost spanning tree.

Examples

standard input	standard output
2 3 0 2 4 3 4 5	7
3 4 1 7 10 2 5 6 8 3 0 5 2 7	16

Problem I. Ivan Smirnov

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

https://stackoverflow.com/questions/45658828/number-of-ways-to-merge-2-parenthesis-sequences. Now you can solve a strictly harder problem in about 10^{18} years. This almost fits in the time limit. All you have to do is some minor optimizations.

Two sequences of parentheses s and t **mergeable** if they can be interleaved to form a balanced parentheses sequence. Formally, if s has length n and t has length m they are mergeable if and only if there exists a balanced parentheses sequence p of length $n+m$, and two disjoint sequences of indices a and b of length n and m respectively, such that:

1. $0 \le a_1 < a_2 < \ldots < a_n < n + m$
2. $0 \le b_1 < b_2 < \ldots < b_m < n + m$
3. For all i $p_{a_i} = s_i$
4. For all i $p_{b_i} = t_i$

The **RLE** (Run Length Encoding) of some string is a list of pairs (c_i, a_i) where c_i is a character and a_i is a positive integer and $\forall_i\ c_i \neq c_{i+1}$, the length of the encoding (i.e. number of pairs in the list) is called the **run length** of the string. The original string of RLE $[(c_1, a_1), (c_2, a_2), \ldots, (c_n, a_n)]$ consists of a_1 repetitions of character c_1, followed by a_2 repetitions of character c_2, and so on, and ends with a_n repetitions of character c_n. It can be shown that RLE is unique.

You are given an RLE of a parentheses sequence s and RLEs of m other parentheses sequences candidates t_i.

For each i find out if t_i and s are mergeable.

Input

The input starts with a description of the sequence s. The first line contains a single integer n ($1 \le n \le 3 \cdot 10^5$), the run length of s.

n lines follow. i-th of them describes a single pair in the RLE of s and contains a character c_i and an integer a_i separated by a single space ($c_i \in \{(,)\}, c_i \neq c_{i+1}, 1 \le a_i \le 10^{12}$).

The next line contains a single integer m ($1 \le n \le 3 \cdot 10^5$), the number of sequences t to check.

The remaining lines describe those sequences one by one, using the same format as the description of s. If this isn't clear you should take a look at the samples.

It is guaranteed that the sum of run lengths of those m sequences doesn't exceed $3 \cdot 10^5$.

Output

Print m lines.

The i-th of those lines should contain `1` if t_i and s are mergeable and `0` if they are not.

Examples

standard input	standard output
4 (1) 3 (3) 1 1 2 (1) 1	0
2 (2) 1 4 1 (1 1) 1 2) 2 (1 2) 4 (3	0 1 1 0
4) 2 (3) 5 (100 3 1) 96 2 (132) 228 4 (2) 3 (5) 100	0 1 1
1) 1000000000000 2 1) 1000000000000 1 (1000000000000	0 1

Problem J. Juke Artem

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

You are given a tree with n vertices enumerated from 0 to $n-1$ inclusive. There is a number in each vertex. Numbers are from 0 to $n-1$ and are all distinct (i.e. they form a permutation). The number x is **in its place** if it is in the x-th vertex.

You are allowed to swap two numbers at the endpoints of some edge of the tree. This costs you 0 if at least one of those numbers was in its place before the swap and 1 otherwise.

What is the minimum cost you have to pay to put all numbers in their places?

Input

The first line contains a single integer n ($2 \le n \le 10^5$), the number of vertices of the tree.

The second line contains $n-1$ integers p_i ($0 \le p_i < i$). i-th of them describes an edge between vertices p_i and i.

The third line contains n integers a_i ($0 \le a_i < n$). i-th of them (in zero based indexing) is equal to the number which is initially in the i-th vertex. a is a permutation.

Output

Print a single integer — the minimum cost you have to pay.

Examples

standard input	standard output
2 0 0 1	0
2 0 1 0	1
3 0 0 0 2 1	2

Problem K. Kunyavskiy Pavel

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

Carol and David are playing a game. There is a rooted binary tree (each vertex has either 0 or 2 children) and an integer k. **Depth** of the vertex is its distance from the root. There is an ordered pair of integers in each leave. Elements of the pairs are between 0 and $k-1$ inclusive.

Let E denote the set of internal (non-leave) vertices with even height and O the set of internal vertices with odd height. For each vertex in E Carol marks one of it children. The set of vertices to mark is Carol's strategy. There are $2^{|E|}$ possible strategies for Carol. David does the same thing with vertices in O. Therefore he has $2^{|O|}$ possible strategies. We consider only pure strategies in this problem.

A token is placed in the root and moved repeatedly from a vertex to its marked child until it arrives at a leave. Let (c, d) be the pair of integers in this leave. Carol receives the payout of c and David receives the payout of d.

A pair of strategies form a **Nash equilibrium** if neither of the players can get a better payout for himself by changing his strategy while the opponents strategy stays the same.

You are given the tree but not given the pairs of integers in the leaves. There are k^{2L} ways to choose the ordered pairs in the leaves, where L is the number of leaves. Count sum of the number of pairs of strategies which form a Nash equilibrium over all possible ways choose the pairs in the leaves.

Output the correct answer modulo 998244353. Formally, if the actual answer is y and your answer is x, it will be considered correct if $-2^{63} \le x < 2^{63}$ and $x - y$ is divisible by 998244353.

Input

The first line contains two integers n and k ($3 \le n < 5000$, $1 \le k \le 20$), the number of vertices in the tree and the upper bound on the elements of the pairs in the leaves.

The second line contains $n-1$ integers p_i ($0 \le p_i < i$). i-th of them (counting from one) describes an edge between vertices p_i and i. Every integer occurs either zero or two times among p_i.

Output

Print a single integer — the answer to the problem modulo 998244353.

Examples

standard input	standard output
3 3 0 0	108
5 1 0 0 1 1	4
17 20 0 1 2 0 1 2 4 7 4 7 3 8 5 3 5 8	465216081

Note

Notes there are 3^4 ways to assign pairs to leaves, 2 possible strategies for Carol and 1 possible strategy for David (he doesn't get to choose anything, therefore this strategy is somewhat degenerate). If the Carol's payout in the leaves are the same both of her strategies form a Nash equilibrium with the David's unique strategy, otherwise only the strategy which marks the leave with the higher payout does. There are $3 \cdot 3^2$ assignments with equal Carol's payouts and $6 \cdot 3^2$ with different payouts. The answer is $27 \cdot 2 + 54 = 108$.

Problem L. Lidia Perovskaya

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

n players are playing in an elimination tournament. The tournament is a sequence of $n-1$ matches. Each match consists of two people playing against each other. One of them loses and is eliminated from the tournament (i.e. he can't participate in any further matches) and the other one wins and is not eliminated. The last match is called the **final**, because it consists of the only two not eliminated players. No two consecutive matches, none of which is the final, may share a participant.

How many different possible tournaments are there? Two tournaments are considered different if there exists a pair of players which played against each other in one of them but didn't in the other.

Output the correct answer modulo a prime number m. Formally, if the actual answer is y and your answer is x, it will be considered correct if $-2^{63} \le x < 2^{63}$ and $x - y$ is divisible by m.

Input

The only line contains two integers n and m ($2 \le n \le 10^6$, $10^6 + 3 \le m \le 10^9 + 9$, m is prime), the number of players and the modulo.

Output

Print a single integer — the number of possible tournaments modulo m.

Examples

standard input	standard output
2 1000003	1
3 756871351	3
4 79415263	12
10 62493391	37074959
228 602495767	489051459
1000 347390201	71907364
3228 172329319	92438468
10000 288002747	214265262
32228 839393021	778284082
100000 625953467	462027594
322228 493329803	424612739
1000000 1000000009	195243062

Problem M. Mikhail Tikhomirov

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

You are given an integer n and a number of non-empty sets of elements from 0 to $n-1$.

Construct a permutation p of length n, such that for every given set S the following holds:

$$\max_{s \in S} p_s - \min_{s \in S} p_s = |S| - 1$$

.

The solution is guaranteed to exist. If there are multiple solutions output any.

Input

The first line contains two integers n and m ($3 \le n \le 100, 1 \le m \le 100$), length of the permutation and the number of sets.

m lines follow. i-th of them contains an integer k_i ($2 \le k_i < n$) followed by k_i integers $a_{i,j}$ ($0 \le a_{i,j} < n$, $a_{i,j} < a_{i,j+1}$), size of the i-th set and its elements themselves.

It is guaranteed that the given sets are pairwise distinct and the answer exists.

Output

Print n integers. i-th of them should be equal to p_i.

Examples

standard input	standard output
3 1 2 0 1	0 1 2
8 5 3 4 5 7 3 2 3 6 3 2 5 7 2 4 5 2 1 6	7 6 3 4 0 1 5 2