Problem A. Qualifiers Ranking Rules

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	128 megabytes

The following is the current ranking rules for the ICPC Asia EC Online Qualifiers, and there will be two online contests.

1. In each contest, only the rank of the top-ranked team from each university will be taken as the score of that university;

2. In each contest, participating universities will be ranked according to their scores;

3. The two rankings of universities are combined using the merge sorting method. For any two universities that obtain the same ranking in different contests, the university that received this ranking in the first contest will be ranked first.

4. Delete duplicate universities and obtain the final ranking of all participating universities (only the highest rankings for each university are retained).

Now assuming that there are n teams in the first contest and m teams in the second contest.

For each contest, given the ranking of each team and the university to which it belongs, please output the final ranking of all participating universities according to the above rules.

You can better understand this process through the sample.

Input

The first line contains two integers n, m $(1 \leq n, m \leq 10^4)$, representing the number of teams participating in the first contest and the second contest.

Then following n lines, the i-th line contains a string s_i $(1 \leq |s_i| \leq 10)$ only consisting of uppercase letters, representing the abbreviation of the university to which the i-th ranked team in the first contest belongs.

Then following m lines, the i-th line contains a string t_i $(1 \leq |t_i| \leq 10)$ only consisting of uppercase letters, representing the abbreviation of the university to which the i-th ranked team in the second contest belongs.

It's guaranteed that each university has only one abbreviation.

Output

Output several lines, the i-th line contains a string, representing the abbreviation of the i-th ranked university in the final ranking.

You should ensure that the abbreviation of any participating universities appears exactly once.

Example

standard input	standard output
14 10 THU THU THU THU XDU THU ZJU THU ZJU THU NJU WHU THU HEU PKU THU PKU PKU ZJU NUPT THU NJU CSU ZJU	THU PKU XDU ZJU NJU NUPT WHU HEU CSU

Note

Sample is part of the results in 2022 ICPC Asia EC Online Contest.

In the first contest, the ranking of the universities is:

THU

XDU

ZJU

NJU

WHU

HEU

In the second contest, the ranking of the universities is:

PKU

THU

ZJU

NUPT

NJU

CSU

By combining these two rankings according to the rules, the rankings of the universities is:

THU
PKU
XDU
THU
ZJU
ZJU
NJU
NUPT
WHU
NJU
HEU
CSU

By deleting duplicate universities we will get the final ranking.

Problem B. String

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

Given two strings S_1 and S_2 of equal length (indexed from 1).

Now you need to answer q queries, with each query consists of a string T.

The query asks how many triplets of integers (i, j, k) $(1 \le i \le j < k \le |S_1|)$ satisfy the condition $S_1[i, j] + S_2[j + 1, k] = T$.

Here $S[l, r]$ denotes the substring of S with index form l to r, and "+" denotes concatenation of strings.

Input

The first line contains a string S_1 .

The second line contains a string S_2 .

It is guaranteed that $1 \le |S_1| = |S_2| \le 10^5$.

The third line contains a positive integer q $(1 \le q \le 2 \times 10^5)$, representing the number of queries.

The next q lines each contain a string T $(1 \le |T| \le 2 \times 10^5)$, representing the query strings.

It is guaranteed that $\sum |T| \le 2 \times 10^5$ and all the strings input only consisting of lowercase letters.

Output

For each query, output a line with a positive integer representing the number of triplets that satisfy the condition.

Example

standard input	standard output
aaababaa aababbca 7 aa abb aab ab abc bb ba	3 1 3 2 2 1 0

Problem C. Multiply Then Plus

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

Given n pairs $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$, you need to perform q operations. Each operation has one of the following forms:

- "`1` k a b" ($1 \le k \le n$, $1 \le a, b \le 10^9$): Modify the k-th pair into (a, b).
- "`2` x l r" ($1 \le x \le 10^9$, $1 \le l \le r \le n$): Find the maximum value of $a_i \times x + b_i$, where $l \le i \le r$.

Input

The first line contains two integers n and q ($1 \le n, q \le 500\,000$) denoting the number of pairs and the number of operations.

Each of the following n lines contains two integers a_i and b_i ($1 \le a_i, b_i \le 10^9$), denoting the i-th pair.

Each of the next q lines describes an operation in the format shown above.

Output

For each query, print a single line containing an integer denoting the answer.

Example

standard input	standard output
3 5 2 3 1 5 3 1 2 1 1 3 2 3 1 2 1 3 1 1 2 3 3 3 2 2 1 3	6 9 4 7

Problem D. Transitivity

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 megabytes

Given a **simple undirected graph** with n vertices and m edges, and it's guaranteed that $m < \frac{n(n-1)}{2}$.

We define an undirected graph to be transitive if and only if for any two **different** vertices u, v :

If there exists a path starting from u and ending at v in the graph, then there should exists an edge connected u and v in the graph.

Now you should add some undirected edges to the graph (**add at least one edge**). You need to ensure that after adding edges, the graph is still a simple undirected graph and is transitive.

The question is, how many edges need to be added at least?

Recall that a simple undirected graph is an undirected graph that does not have more than one edge between any two vertices and no edge starts and ends at the same vertex.

Input

The first line contains two integers n, m ($3 \le n \le 10^6, 1 \le m \le \min(10^6, \frac{n(n-1)}{2} - 1)$), indicating the number of vertices and edges in the given graph.

Then the following m lines, each line contains two integers u, v ($1 \le u, v \le n, u \ne v$) , indicating an edge in the given graph. It's guaranteed that the graph is simple.

Output

A single positive integer, representing the minimum number of edges you need to add.

Example

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

Problem E. Magical Pair

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

For a prime number n, if a pair of positive integers (x, y) satisfies the congruence relation:

$$x^y \equiv y^x \pmod{n}.$$

Then we consider (x, y) to be magical.

We want to know how many ordered pairs of positive integers (x, y) are magical for a given prime number n, where $0 < x, y \leq n^2 - n$. Since the answer could be large, we will output it modulo 998244353.

Input

The first line input a positive integer T ($1 \leq T \leq 10$), which represents the total number of test cases.

Then for each test case, input a single line with a prime number n ($2 \leq n \leq 10^{18}$), and it's guaranteed that $n - 1$ is not a multiple of 998244353.

Output

Output T lines, each containing an integer representing the result modulo 998244353.

Examples

standard input	standard output
5 5 11 67 97 101	104 1550 479886 1614336 1649000
6 998244353 998244853 19260817 1000000007 1000000009 350979772330483783	284789646 90061579 971585925 887008006 527110672 334479293

Problem F. Alice and Bob

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

You are watching Alice and Bob playing a game.

The game is played on an array of length n. Alice and Bob take turns in action, Alice goes first.

In each turn, the current player can select two different numbers a_i and a_j in the array, and then change their values. Assume after changes the values of them are a'_i and a'_j , then an operation is legal if and only if $a_i + a_j = a'_i + a'_j$ and $|a'_i - a'_j| < |a_i - a_j|$. Those who cannot perform legal operations lose.

After playing a few games, You decided to help Alice, because Alice, who was always overloaded when faced with a bunch of numbers, always unable to think in front of the genius Bob.

Before the start of each game, You will help Alice remove several numbers (could be 0) to reduce the burden on her brain, and you always **leave exactly three numbers**. And in order to give Alice a greater chance of winning, You will always leave Alice with a winning state, that is, there must be some kind of operating strategy that makes Alice must win no matter how Bob acts.

Now the question is how many ways there are to remove the numbers.

Two ways for removing numbers are considered different if and only if there exists an integer i $(1 \le i \le n)$ such that a_i is not removed in one way and is removed in the other way.

Input

The first line contains a single integer T , representing the number of test cases.

Then follow the descriptions of each test case.

The first line contains an integer $n(3 \le n \le 5 \times 10^5)$, which represents the number of numbers at the beginning.

The second line contains n integers $a_1, a_2, a_3, \ldots, a_n (0 \le a_i \le 10^{18})$, representing the original n numbers.

It's guaranteed that $\sum n \le 3 \times 10^6$.

Output

For each test case, output one integer in one line, indicating the number of different ways to remove numbers that satisfy the condition.

Example

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

Note

In the first test case, only removing a_2 leaves a loser state.

Problem G. Spanning Tree

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

We generate a spanning tree of n nodes according to the following random process:

Initially, there are no edges connecting the n nodes.

Then process the following $n-1$ operations:

- For the i-th operation, two integers a_i and b_i will be input, and it's guaranteed that the two nodes are not connected before.
- Select a node u_i from the connected block where a_i is located with uniform probability, then select a node v_i from the connected block where b_i is located with uniform probability, and then add an edge to connect u_i and v_i .

It can be proved that no matter which two nodes are selected in each operation, after all operations are processed, a spanning tree of n nodes will be formed.

Now given a spanning tree of n nodes. What is the probability that the spanning tree formed by the random process is exactly this spanning tree?

You only need to output the value of the probability modulo 998244353 .

Please note that the probability could be 0, which means that you can never generate this spanning tree.

Input

The first line contains a single integer n ($1 \le n \le 10^6$) , representing the number of nodes.

For the following $n-1$ lines, each line contains two integers a_i, b_i ($1 \le a_i, b_i \le n, a_i \ne b_i$), representing the i-th operation, and it's guaranteed that the two nodes are not connected before.

For the following $n-1$ lines, each line contains two integers c_i, d_i ($1 \le c_i, d_i \le n, c_i \ne d_i$), representing an edge of the given spanning tree, and it's guaranteed that the given edges forms a spanning tree.

Output

One line containing one integer, representing the value of the probability modulo 998244353 .

Example

standard input	standard output
3 1 2 1 3 1 2 1 3	499122177

Problem H. Range Periodicity Query

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

For a string $w = w_1 w_2 \dots w_{len}$, we say that an integer p is a period of w if $w_i = w_{i+p}$ holds for all i $(1 \le i \le len - p)$ and $1 \le p \le len$.

You will be given a string $d = d_1 d_2 \dots d_n$ to generate $n+1$ strings $S_0, S_1, S_2, \dots, S_n$, where S_0 is an empty string, and for all i $(1 \le i \le n)$:

- When d_i is a lowercase English letter, $S_i = d_i + S_{i-1}$.
- When d_i is an uppercase English letter, assume its lowercase version is c_i, then $S_i = S_{i-1} + c_i$.

Here, "+" denotes concatenation of strings.

You will then be given a sequence of integers $p_1, p_2, \dots, p_m$. You need to answer q queries, in each query, you will be given three integers k, l and r. You need to find the minimum number among $p_l, p_{l+1}, \dots, p_{r-1}, p_r$ such that it is a period of string S_k, or determine there is no answer.

Input

The first line contains a single integer n $(1 \le n \le 500\,000)$ denoting the number of non-empty strings.

The second line contains a string d of length n consists of lowercase and uppercase English letters.

The third line contains a single integer m $(1 \le m \le 500\,000)$ denoting the length of the sequence p.

The fourth line contains m integers $p_1, p_2, \dots, p_m$ $(1 \le p_i \le n)$.

The fifth line contains a single integer q $(1 \le q \le 500\,000)$ denoting the number of queries.

Each of the next q lines contains three integers k, l and r $(1 \le k \le n,\ 1 \le l \le r \le m)$, denoting a query.

Output

For each query, print a single line containing an integer denoting the answer. Note that when there is no answer, please print "-1" instead.

Example

standard input	standard output
7 AABAABa 9 4 3 2 1 7 5 3 6 1 6 1 4 4 2 1 4 2 1 3 3 3 5 5 4 7 7 8 9	1 1 2 -1 3 6

Problem I. Pa?sWorD

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	32 megabytes

You need to log into a mysterious system, but you realize you've forgotten your password. The system does not support resetting password, so the only way to log in is to keep trying.

Fortunately, you still remember some information about the password:

Firstly, you are sure that the length of the password is n .

Then the information you remember can be described by a string s of length n.

let s_i represent the i-th character of s :

- if s_i is an uppercase letter or a digital character, then the i-th character of the password must be s_i;
- if s_i is an lowercase letter, then the i-th character of the password might be s_i or the uppercase form of s_i;
- if s_i is a question mark '?' , then the i-th character of the password might be any uppercase letters, lowercase letters, and digital characters.

It's guaranteed that the string s only contains uppercase letters, lowercase letters, digital characters, and question marks '?'.

In addition, the system also has several requirements for passwords:

- At least one uppercase letter appears in the password;
- At least one lowercase letter appears in the password;
- At least one digital character appears in the password;
- Any two adjacent characters in the password cannot be the same.

Now you want to know, how many possible passwords are there? A possible password should fits both your memory and the system's requirements, and it's guaranteed that there exists at least one possible password.

You know that this answer will be very large, so you just need to calculate the result modulo 998244353 .

Please note the unusual memory limit.

Input

The first line contains a single integer n ($3 \le n \le 10^5$) , representing the length of the password .

The second line contains a string s with length n. It's guaranteed that the string s only contains uppercase letters, lowercase letters, digital characters, and question marks '?'.

It's guaranteed that there exists at least one possible password.

Output

output a single line containing a single integer, representing the answer modulo 998244353.

Example

standard input	standard output
4 a?0B	86

Problem J. Minimum Manhattan Distance

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Recall that on a two-dimensional plane, the Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$. If both coordinates of a point are all integers, then we call this point an integer point.

Given two circles C_1, C_2 on the two-dimensional plane, and guaranteed the x-coordinates of any point in C_1 and any point in C_2 are different, and the y-coordinates of any point in C_1 and any point in C_2 are different.

Each circle is described by two integer points, and the segment connecting the two points represents a diameter of the circle.

Now you need to pick a point (x_0, y_0) inside or on the C_2 such that minimize the expected value of the Manhattan distance from (x_0, y_0) to a point inside C_1 , if we choose this point with uniformly probability among all the points with a real number coordinate inside C_1.

Input

The first line contains a single integer t $(1 \le t \le 10^5)$, representing the number of test cases.

Then follow the descriptions of each test case.

The first line contains 4 integers $x_{1,1}, y_{1,1}, x_{1,2}, y_{1,2}$, representing the segment connecting $(x_{1,1}, y_{1,1})$ and $(x_{1,2}, y_{1,2})$ is a diameter of the circle C_1.

The second line contains 4 integers $x_{2,1}, y_{2,1}, x_{2,2}, y_{2,2}$, representing the segment connecting $(x_{2,1}, y_{2,1})$ and $(x_{2,2}, y_{2,2})$ is a diameter of the circle C_2.

All the coordinates input are integers in the range $[-10^5, 10^5]$.

Output

For each test case, output a single line with a real number - the minimum expected Manhattan distance. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}. That is, if your answer is a, and the jury's answer is b, then the solution will be accepted if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Example

standard input	standard output
1 0 0 2 1 4 5 5 2	4.2639320225

Problem K. Minimum Euclidean Distance

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	128 megabytes

One day you are surviving in the wild. After a period of exploration, you determine a safe area, which is a convex hull with n vertices $P_1, P_2, \ldots, P_n$ in counter-clockwise order and any three of them are not collinear.

Now you are notified that there will be q airdrop supplies, and for the i-th supply, its delivery range is described by a circle C_i , which means the supply will landed with uniformly probability among all the points with a real number coordinate inside C_i.

You need supplies so much that you decide to predetermine a starting point for **each** supply, and the starting point of two different supplies can be different. Every starting point should be inside the safe area and have the smallest expected value of **the square** of the Euclidean distance to the corresponding supply landing point.

Recall that On a two-dimensional plane, the Euclidean distance between two points (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. If both coordinates of a point are all integers, then we call this point an integer point.

Input

The first line contains two integers n, q $(3 \le n, q \le 5000)$, representing the number of vertices of the safe area and the number of airdrop supplies.

The following n lines, each line contains two integers x_i, y_i , representing the coordinates of the i-th vertex of the safe area.

It's garanteed that the vertices are given in counter-clockwise order and any three of them are not collinear.

Then the following q lines, each line contains 4 integers $x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2}$, representing the segment connecting $(x_{i,1}, y_{i,1})$ and $(x_{i,2}, y_{i,2})$ is a diameter of the circle C_i.

All the coordinates input are integers in the range $[-10^9, 10^9]$.

Output

For each airdrop supply, output a single line with a real number - the minimum expected value of **the square** of the Euclidean distance. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-4}. That is, if your answer is a, and the jury's answer is b, then the solution will be accepted if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-4}$.

Example

standard input	standard output
4 3 0 0 1 0 1 1 0 1 0 0 1 1 1 1 2 2 1 1 2 3	0.2500000000 0.7500000000 1.8750000000

Problem L. KaChang!

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	128 megabytes

Setting the time limit for algorithm competition questions is a very skillful task. If you set the time limit too tight, many people will scold you for being too demanding on details. On the other hand, if you set the time limit too loosely and allow an algorithm with unexpected time complexity to pass, then many people will scold you too.

When preparing problems, people usually set the time limit to at least twice the running time of the standard program, but sometimes contestants still feel that the time limit is too tight.

Now you have the standard program for a problem and another n programs considered "should pass the problem". Given the running time of each program, please find the smallest integer $k \geq 2$, so that when the time limit is set to k times the running time of the standard program, all the provided programs can pass. A program can pass if and only if its running time less or equal to the time limit.

Input

The first line contains two integers n, T ($1 \leq n, T \leq 10^5$), representing the number of provided programs (not include the standard program), and the running time of the standard program.

The second line contains n integers $t_1, t_2, \ldots, t_n$ ($1 \leq t_i \leq 10^9$) , representing the running time of the provided programs.

Output

Output a single integer greater or equal to 2, representing the minimin k which could guarantee that all the provided programs can pass.

Example

standard input	standard output
5 1000 998 244 353 1111 2333	3