Problem A. Three Arrays

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are given three arrays: a containing n_a elements, b containing n_b elements and c containing n_c elements. These arrays are sorted in non-decreasing order: that is, for every i such that $1 \le i < n_a$ we have $a_i \le a_{i+1}$, for every j such that $1 \le j < n_b$ we have $b_j \le b_{j+1}$, and for every k such that $1 \le k < n_c$ we have $c_k \le c_{k+1}$.

Your task is to calculate the number of triples (i, j, k) such that $|a_i - b_j| \le d$, $|a_i - c_k| \le d$, and $|b_j - c_k| \le d$.

Input

The input contains one or more test cases. Each test case consists of four lines.

The first line of each test case contains four integers: d, n_a, n_b, and n_c ($1 \le d \le 10^9$, $1 \le n_a, n_b, n_c \le 5 \cdot 10^5$).

The second line contains n_a integers $a_1, a_2, \ldots, a_{n_a}$: the array a ($-10^9 \le a_i \le 10^9$).

The third line contains n_b integers $b_1, b_2, \ldots, b_{n_b}$: the array b ($-10^9 \le b_i \le 10^9$).

The fourth line contains n_c integers $c_1, c_2, \ldots, c_{n_c}$: the array c ($-10^9 \le c_i \le 10^9$).

All arrays are sorted in non-decreasing order. The total sum of n_a over all testcases does not exceed $5 \cdot 10^5$. The total sum of n_b over all testcases does not exceed $5 \cdot 10^5$. The total sum of all n_c over all testcases does not exceed $5 \cdot 10^5$. The test cases just follow one another without any special separators.

Output

For each test case, print a single integer: the number of triples (i, j, k) such that $|a_i - b_j| \le d$, $|a_i - c_k| \le d$, and $|b_j - c_k| \le d$.

Example

standard input	standard output
1 3 3 3 1 2 3 1 2 3 1 2 3 1 6 6 6 1 1 2 2 3 3 2 2 3 3 4 4 3 3 4 4 5 5	15 56

Problem B. Expected Shopping

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 mebibytes

You want to buy m cans of chips. There are n different shops, and each shop has enough cans of chips to cover your needs. The price of a single can of chips at i-th shop is a_i coins. You don't like to pay more than B coins for a can, so if at some shop j the price of a can is $a_j > B$, then for you this price is *unreasonable*. Otherwise, it is *reasonable*.

You may visit shops in arbitrary order, but each shop can be visited no more than once.

Let us assume that you visit shop j and you still need to buy k cans of chips. If the price at this shop is *reasonable* ($a_j \le B$), then you buy k cans at this shop and go home without visiting any shop afterwards. Otherwise, you buy only one can of chips, and if you still need to buy some cans, you proceed to the next shop.

As soon as you have m cans of chips, you finish your shopping trip. It is guaranteed that there are at least m shops, so this has to happen eventually.

Calculate the expected number of coins you will spend if each possible shopping plan is equiprobable. Formally, this means that each permutation of n numbers denoting the order in which you plan to visit the shops has the same probability of being chosen. The answer must be calculated as a rational fraction $\frac{p}{q}$, where $q > 0$ and $\gcd(p, q) = 1$.

Input

The first line contains three integers: n, m, and B ($1 \le m \le n \le 8 \cdot 10^5$, $1 \le B \le 5 \cdot 10^6$).

The second line contains n integers $a_1, a_2, \ldots, a_n$, where a_i is the price of a single can of chips at i-th shop ($1 \le a_i \le 5 \cdot 10^6$).

Output

Let p and q be the numbers such that $\frac{p}{q}$ is the expected number of coins you will spend, $q > 0$ and $\gcd(p, q) = 1$. Print p on the first line and q on the second line of the output.

Examples

standard input	standard output
3 2 4 2 3 4	6 1
7 3 3 7 6 5 4 3 2 1	47 5
11 5 16 20 21 23 10 6 19 5 5 25 27 14	50329 924

Problem C. Cover the Paths

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You are given an undirected unweighted tree consisting of n vertices labeled by integers 1, 2, …, n. A total of m simple paths are chosen in this tree. Each path is described as a pair of its endpoints (a_i, b_i).

Let V be the set of all vertices of the tree. We say that subset S of V is *good* if for every i such that $1 \leq i \leq m$, the simple path from a_i to b_i contains at least one vertex from S. We say that subset T be *the best* subset if T is a *good* subset and there is no *good* subset X such that $|X| < |T|$.

You have to find *the best* subset of V.

Input

The first line contains an integer n, the number of vertices in the tree ($1 \leq n \leq 10^5$).

Each of the next $n-1$ lines describes an edge of the tree. Edge i is denoted by two integers u_i and v_i, the labels of vertices it connects ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$). It is guaranteed that the given edges form a tree.

The next line contains an integer m, the number of paths ($1 \leq m \leq 10^5$).

Each of the next m lines describes a path in the tree. Path i is denoted by two integers a_i and b_i, the labels of the endpoints ($1 \leq a_i, b_i \leq n$). **For some paths, it may be that $a_i = b_i$.** It is **not** guaranteed that all paths are pairwise distinct.

Output

On the first line, print the size of *the best* subset of V. On the second line, print the labels of vertices belonging to *the best* subset of V in any order.

If there are several possible solutions, print any one of them.

Examples

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

Problem D. Elevator

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You have a very important job: you are responsible for an elevator in the new skyscraper.

There are n persons who will come to the underground parking located on floor 0 and wait for an elevator to bring them to some upper floor. Formally, i-th person comes to the elevator at moment t_i and wants to reach floor a_i. The elevator has **infinite** capacity; that is, there is no limit on the number of people using the elevator at any moment. All numbers t_i are distinct. Passengers always enter the elevator as long as it is at floor 0.

The elevator uses the following algorithm: it stays open on floor 0 until you send it to deliver passengers, then it moves to the highest floor it needs (the maximum a_i among all passengers who are currently in the elevator), distributing the passengers in the process, and returns to the parking. The elevator spends 1 unit of time to move to the next floor (or to the previous floor). The time spent for opening and closing the doors of the elevator, as well as for the passengers entering and leaving the elevator, is negligible. At moment 0, the elevator is at floor 0.

You want to minimize the moment of time when the elevator will return to floor 0 after delivering everyone.

Input

The input contains one or more test cases.

The first line of each test case contains one integer n: the number of passengers ($1 \le n \le 2 \cdot 10^5$).

Each of the following n lines contains two space-separated integers t_i and a_i: the moment of time when i-th passenger comes to the elevator, and the destination floor of i-th passenger ($1 \le t_i, a_i \le 10^9$).

All t_i in one test case are distinct, passengers appear in input in ascending order of t_i.

The sum of the values of n over all test cases does not exceed $2 \cdot 10^5$. The test cases just follow one another without any special separators.

Output

For each test case, print one integer: the minimum possible moment of time when the elevator will return after delivering all passengers.

Example

standard input	standard output
3 1 9 2 6 15 6 3 1 9 2 6 15 8	31 33

Problem E. Code-Cola Plants

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

Berland consists of n cities which are numbered by integers from 1 to n. There are m directed roads connecting some pairs of cities. There is no directed cycle of roads in Berland.

There are two Code-Cola plants in Berland. The first one is a *producing* plant, it is located in the city a. The second one is a *recycling* plant, it is located in the city b.

The Code-Cola Corporation decided to use $n-1$ roads for delivery. Using this set of roads, it must be possible to reach all of the n cities from the production plant (that is, from the city a). Also the Code-Cola Corporation decided to use some **other** $n-1$ roads by recycling trucks which will deliver empty Code-Cola bottles to the recycling plant. Using this second set of roads, it must be possible to reach the recycling plant (that is, the city b) from all of the n cities.

Help the Code-Cola Corporation to find two **disjoint** sets of roads such that:

- each of the two sets contains $n-1$ roads;
- it is possible to get to any city from the city a by moving along the first set of roads;
- it is possible to get from any city to the city b by moving along the second set of roads.

Input

The input contains one or more test cases. The input format for each test case is described below.

Each test case starts with a line containing four integers: n, the number of cities in Berland, m, the number of roads, a, the city with the producing plant, and b, the city with the recycling plant ($2 \le n \le 5 \cdot 10^5$, $1 \le m \le 10^6$, $1 \le a, b \le n$). It is possible that $a = b$.

The following m lines contain descriptions of the roads, one description per line. The i-th description consists of two integers x_i and y_i meaning that there is a directed (one-way) road from x_i to y_i ($1 \le x_i, y_i \le n$). It is guaranteed that there is no directed cycle of roads in Berland. Between a pair of cities, there can be multiple roads in the same direction.

The sum of all values of n over all test cases in a test does not exceed $5 \cdot 10^5$. The sum of all values of m over all test cases in a test does not exceed 10^6. The test cases just follow one another without any special separators.

Output

For each test case, print the answer as follows:

If there is a solution, print "`YES`" on a separate line, followed by two lines containing $n-1$ road indices each. The first line must describe the roads from the first set, the second line must describe the roads from the second set. All $2 \cdot (n-1)$ indices must be distinct. The roads are numbered from 1 to m in order of their appearance in the input. You can print numbers on a line in any order. If there are several possible solutions, print any one of them.

If there is no solution, print "`NO`" on a separate line.

Example

standard input	standard output
4 7 1 4 1 2 1 2 1 4 2 3 2 3 3 4 3 4 4 3 1 2 1 2 2 4 4 3 5 8 3 1 3 2 5 2 3 4 4 5 4 1 2 1 3 5 3 1	YES 2 5 6 3 7 4 NO YES 1 3 4 8 5 6 2 7

Problem F. GCD

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

You are given an array of integers $a_1, a_2, \ldots, a_n$.

Find the maximum possible greatest common divisor of all numbers from the array if you can erase no more than k elements ($k \le \frac{n}{2}$) from this array.

Input

The first line contains two integers: n, the number of elements in the array, and k, the maximum number of elements you can erase ($2 \le n \le 10^5$, $0 \le k \le \frac{n}{2}$).

The second line contains n integers $a_1, a_2, \ldots, a_n$: the array a ($1 \le a_i \le 10^{18}$).

Output

Print the maximum possible greatest common divisor of all elements of the array after erasing no more than k elements.

Examples

standard input	standard output
4 1 6 15 35 14	1
4 2 6 15 35 14	7
3 1 897612484786617600 5828 16027	1457

Problem G. Berland Post

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

Berland Post is the national postal service of Berland. There is exactly one post office in each of n Berland cities. Cities and their respective offices are numbered by integers from 1 to n.

There are m pairs of cities (a, b) such that there is direct post traffic from a to b. For each such pair of cities, the delivery time is known: formally, you are given m triples (a_j, b_j, d_j) meaning that each day, the post office in a_j has to send correspondence to the post office in b_j, and d_j is the time elapsed between sending the correspondence from a_j and receiving it at b_j.

Each day, all offices must be open for the equal consecutive amount of time, which is denoted as T. But opening times may differ. If the opening time of i-th office is o_i, then the closing time is $o_i + T$.

Some values of o_i are known and fixed, but some of them are up to you. Your goal is to find such values $T \geq 0$ and o_i that each office receives all the correspondence no later than at closing time, and T is the minimum possible. It is allowed for an office to receive the correspondence even before opening. Assume that each office sends the correspondence instantly after opening.

Formally, find the minimum possible non-negative T and values o_i such that $o_{a_j} + d_j \leq o_{b_j} + T$ for each of the m given triples (a_j, b_j, d_j).

Input

The input contains one or more test cases.

Each test case starts with a line containing two integers: n, the number of cities, and m, the number of direct traffic paths ($1 \leq n \leq 1000, 0 \leq m \leq 2000$).

The second line of each test case contains n tokens o_i, where o_i is either a question mark ("?") if the opening time of the office i is not given and your task is to define it, or an integer ($-10^5 \leq o_i \leq 10^5$) if the opening time of the office i is known and you can not change it.

The following m lines contain descriptions of direct traffic paths, one per line. Each line contains three integers: a_j, b_j, and d_j, denoting direct post traffic from the city a_j to the city b_j with delivery time d_j ($1 \leq a_j, b_j \leq n$, $a_j \neq b_j$, $1 \leq d_j \leq 100$). It is guaranteed that, for each pair of the cities (a, b), there is at most one direct traffic path from a to b.

The sum of all values n in a test case does not exceed 1000. The sum of all values m in a test case does not exceed 2000. The test cases just follow one another without any special separators.

Output

For each test case, print exactly two lines.

Print the minimum possible non-negative real value of T on the first line and the values $o_1, o_2, \ldots, o_n$ on the second line. The values of o_i must be in the range $[-10^9, 10^9]$. Print T and o_i with absolute error of at most 10^{-4}.

The values $o_i \neq$ "?" in the input must not change. For each of the m given triples (a_j, b_j, d_j), it must be true that $o_{a_j} + d_j \leq o_{b_j} + T$.

Examples

<table>
<tr><th>standard input</th><th>standard output</th></tr>
<tr><td>2 1
5 7
1 2 3</td><td>1
5 7</td></tr>
<tr><td>2 2
? ?
1 2 3
2 1 1
3 0
? ? 3</td><td>2
9 10
0
1 -1 3</td></tr>
</table>

Problem H. Compressed Spanning Subtrees

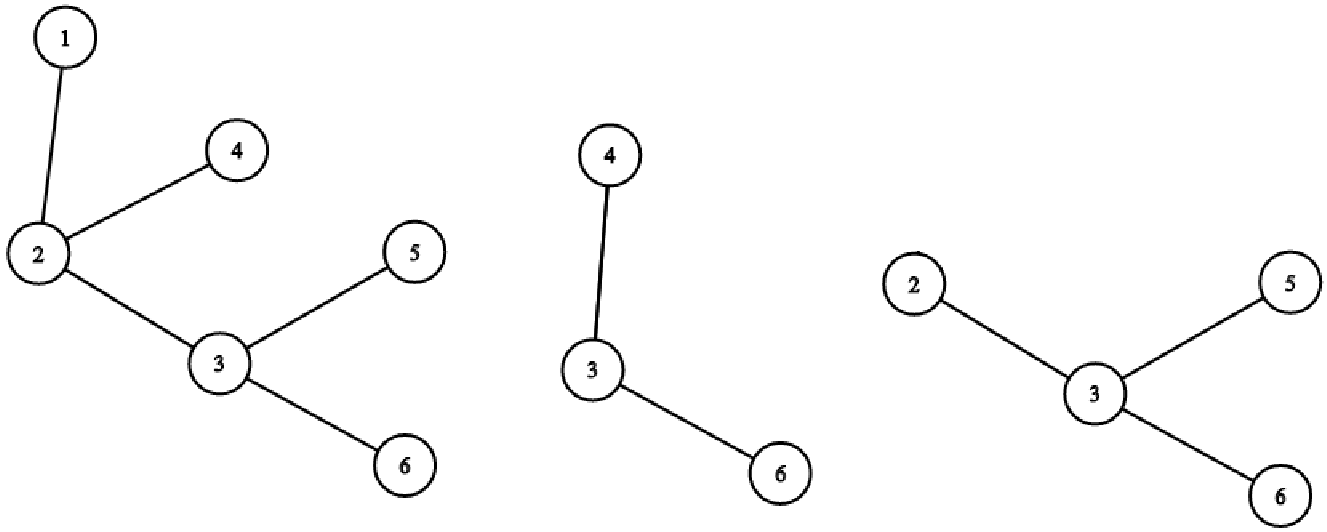
Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

This is an interactive problem. Make sure that your output does not get buffered after each query. Use, for instance, `fflush(stdout)` in `C++`, `System.out.flush()` in `Java`, or `sys.stdout.flush()` in `Python`.

For a tree T consisting of n vertices numbered from 1 to n, the *compressed spanning subtree* $S(X)$ of a set X of *spanned* vertices (vertices that are not in X are called *not spanned*) can be defined by the following algorithm:

1. Assign $S(X) \leftarrow T$;
2. If there is any *not spanned* vertex that has exactly one edge incident to it, remove it along with the edge;
3. Repeat step 2 while its condition stays true;
4. If there is any *not spanned* vertex that has exactly two edges incident to it, remove it along with the edges and add a new edge connecting the two remaining endpoints of the removed edges;
5. Repeat step 4 while its condition stays true.

Formally, $S(X)$ is the smallest subgraph of T containing all vertices in X and then having all other vertices of degree two or less smoothed out.



The tree from test 1*, and its compressed spanning subtrees for* $X = 3, 4, 6$ *and for* $X = 2, 5, 6$.

You are not given the tree T. Instead, your task is to find it. You can ask questions of the following form: "How many vertices does the compressed spanning subtree of X contain?". And since otherwise finding the tree by asking such questions would be impossible, there are no vertices incident to exactly two edges in T.

Interaction Protocol

The first line of input contains a single integer n ($2 \leq n \leq 100$).

Your program can ask a question by printing a line in the format "`?` k x_1 x_2 $\ldots$ x_k" where integer k ($1 \le k \le n$) equals to the number of vertices in X and distinct integers x_i ($1 \le x_i \le n$) represent these vertices in any order. You can ask no more than 2550 questions.

The answer for such question is an integer given on a separate line: the number of vertices in the compressed spanning subtree in question.

After asking sufficient questions, your program must give the answer by printing a line in the format "`!` p_2 p_3 $\ldots$ p_n". Here, considering T as a rooted tree with root at vertex 1, p_i must be the parent vertex of vertex i. After giving the answer, the program must immediately terminate gracefully.

Example

standard input	standard output
6	
	? 3 4 3 6
3	
	? 4 1 2 3 6
4	
	? 3 2 5 6
4	
	! 1 2 2 3 3

Problem I. Prefix-free Queries

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Let $C(s_1, s_2, \ldots, s_k)$ be the number of ways to construct a prefix-free set from the multiset of strings $s_1, s_2, \ldots, s_k$. A prefix-free set is a set of distinct strings in which there are no two strings such that one of these strings is a prefix of another one. In particular, an empty set is a valid prefix-free set. For example, if for any $i \neq j$, s_i is not a prefix of s_j, then $C(s_1, \ldots, s_k) = 2^k$.

Note that we count not the sets themselves, but the ways to construct such sets: the number of ways to choose a subset of indices out of $\{1, 2, \ldots, k\}$ such that the strings with these indices form a prefix-free set. For example, $C(\text{“aa”}, \text{“aa”}, \text{“a”}, \text{“a”}) = 5$: the five ways are to construct an empty set, a set containing the first string, a set containing the second string, a set containing the third string, and a set containing the fourth string.

You are given a string s consisting of n lowercase English letters, and q queries. Let $s[l, r]$ be the substring $s_l s_{l+1} \ldots s_{r-1} s_r$. For each query denoted as “$k\ m\ l_1\ r_1\ l_2\ r_2\ \ldots\ l_k\ r_k$”, print one integer: the value $C(s[l_1, r_1], s[l_2, r_2], \ldots, s[l_k, r_k])$, taken modulo m.

Input

The first line contains two integers: n, the length of s, and q, the number of queries to answer ($1 \le n \le 4 \cdot 10^5$, $1 \le q \le 4 \cdot 10^5$).

The second line contains a string s of length n consisting of lowercase English letters.

Next q lines contain queries, one query per line. Each query has the form “$k\ m\ l_1\ r_1\ l_2\ r_2\ \ldots\ l_k\ r_k$” ($1 \le k \le 4 \cdot 10^5$, $2 \le m \le 10^9$, $1 \le l_j \le r_j \le n$).

The total sum of all k over all queries does not exceed $4 \cdot 10^5$.

Output

For each query, print a line containing a single integer: the value $C(s[l_1, r_1], s[l_2, r_2], \ldots, s[l_k, r_k])$, taken modulo m.

Example

standard input	standard output
10 6	5
aabbaacaba	4
4 30 1 2 5 6 10 10 10 10	0
5 20 1 2 3 4 5 6 7 8 9 10	8
1 2 1 10	16
3 20 9 9 7 7 8 8	9
5 20 6 6 7 7 8 8 9 9 10 10	
4 20 1 1 2 2 3 3 4 4	

Problem J. Subsequence Sum Queries

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You have an array a containing n integers and an integer m. You also have q queries to answer. The i-th query is described as a pair of integers (l_i, r_i). Your task is to calculate the number of such subsequences $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$ that $l_i \le j_1 < j_2 < \ldots < j_k \le r_i$ and $(a_{j_1} + a_{j_2} + \ldots + a_{j_k}) \bmod m = 0$. In other words, you need to calculate the number of subsequences of subarray $[a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}]$ such that the sum of elements in each subsequence is divisible by m.

Input

The first line contains two integers n and m: the number of elements in a and the modulo ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 20$).

The second line contains n integers a_i: the elements of array a ($0 \le a_i \le 10^9$).

The third line contains one integer q: the number of queries ($1 \le q \le 2 \cdot 10^5$).

Then q lines follow. The i-th of these lines contains two integers l_i and r_i that describe the i-th query ($1 \le l_i \le r_i \le n$).

Output

Print q lines. The i-th of them must contain the answer for the i-th query. Queries are indexed in the order they are given in the input. Since the answers can be very large, print them modulo $10^9 + 7$.

Example

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

Problem K. Consistent Occurrences

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

Let us define a *consistent set of occurrences of string t in string s* as a set of occurrences of t in s such that no two occurrences intersect (in other words, no character position in s belongs to two different occurrences).

You are given a string s consisting of n lowercase English letters, and m queries. Each query contains a single string t_i.

For each query, print the maximum size of a consistent set of occurrences of t in s.

Input

The first line contains two space-separated integers n and m: the length of string s and the number of queries ($1 \le n \le 10^5$, $1 \le m \le 10^5$).

The second line contains the string s consisting of n lowercase English letters.

Each of the next m lines contains a single string t_i consisting of lowercase English letters: the i-th query ($1 \le |t_i| \le n$, where $|t_i|$ is the length of the string t_i).

It is guaranteed that the total length of all t_i does not exceed 10^5 characters.

Output

For each query i, print one integer on a separate line: the maximum size of a consistent set of occurrences of t_i in s.

Example

standard input	standard output
6 4 aaaaaa a aa aaa aaaa	6 3 2 1

Problem L. Increasing Costs

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

Berland consists of n cities labeled from 1 to n. The city number 1 is the capital of Berland. There are m two-way roads between some cities. Roads can intersect only in cities. There is no more than one road between each pair of cities, and there is no road that connects a city to itself. If you are moving by j-th road in any direction, you have to pay the tax equal to c_j. **It is possible to reach any city from the capital using only the given m roads.**

You are the CEO of a delivery company, its main office is located in the capital. Your company delivers different goods to every city of Berland, so for each city, you chose some route from the capital to that city which minimized the total sum of taxes of all roads in the route. Let d_k be the total cost of the chosen route from the capital to city k.

The government has decided to choose **exactly one** road (you don't know which one) and increase the tax for using it. So, for each road, you want to know how many cities will be affected if the tax for using this road is increased. City k is affected if, after the tax is increased, you can't choose a route such that the total cost of this route is equal to d_k.

Input

The first line contains two integers n and m: the number of cities and the number roads in Berland ($2 \le n \le 2 \cdot 10^5$, $n - 1 \le m \le 2 \cdot 10^5$).

Each of the next m lines contains three space-separated integers: u_j, v_j, and c_j ($1 \le u_j, v_j \le n$, $1 \le c_j \le 10^9$). These mean that the road number j between cities u_j and v_j initially has tax equal to c_j.

There is no more than one road between each pair of cities, and there is no road that connects a city to itself. It is guaranteed that it is possible to reach every city from the capital using the given roads.

Output

Print m integers, one per line. The j-th integer must be the number of cities affected by increasing the cost of j-th road.

Example

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