Codeforces Round #485 (Div. 1)

A. Fair

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Some company is going to hold a fair in Byteland. There are $$$n$$$ towns in Byteland and $$$m$$$ two-way roads between towns. Of course, you can reach any town from any other town using roads.

There are $$$k$$$ types of goods produced in Byteland and every town produces only one type. To hold a fair you have to bring at least $$$s$$$ different types of goods. It costs $$$d(u,v)$$$ coins to bring goods from town $$$u$$$ to town $$$v$$$ where $$$d(u,v)$$$ is the length of the shortest path from $$$u$$$ to $$$v$$$. Length of a path is the number of roads in this path.

The organizers will cover all travel expenses but they can choose the towns to bring goods from. Now they want to calculate minimum expenses to hold a fair in each of $$$n$$$ towns.

Input

There are $$$4$$$ integers $$$n$$$, $$$m$$$, $$$k$$$, $$$s$$$ in the first line of input ($$$1 \le n \le 10^{5}$$$, $$$0 \le m \le 10^{5}$$$, $$$1 \le s \le k \le min(n, 100)$$$) — the number of towns, the number of roads, the number of different types of goods, the number of different types of goods necessary to hold a fair.

In the next line there are $$$n$$$ integers $$$a_1, a_2, \ldots, a_n$$$ ($$$1 \le a_{i} \le k$$$), where $$$a_i$$$ is the type of goods produced in the $$$i$$$-th town. It is guaranteed that all integers between $$$1$$$ and $$$k$$$ occur at least once among integers $$$a_{i}$$$.

In the next $$$m$$$ lines roads are described. Each road is described by two integers $$$u$$$ $$$v$$$ ($$$1 \le u, v \le n$$$, $$$u \ne v$$$) — the towns connected by this road. It is guaranteed that there is no more than one road between every two towns. It is guaranteed that you can go from any town to any other town via roads.

Output

Print $$$n$$$ numbers, the $$$i$$$-th of them is the minimum number of coins you need to spend on travel expenses to hold a fair in town $$$i$$$. Separate numbers with spaces.

Examples

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

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

Note

Let's look at the first sample.

To hold a fair in town $$$1$$$ you can bring goods from towns $$$1$$$ ($$$0$$$ coins), $$$2$$$ ($$$1$$$ coin) and $$$4$$$ ($$$1$$$ coin). Total numbers of coins is $$$2$$$.

Town $$$2$$$: Goods from towns $$$2$$$ ($$$0$$$), $$$1$$$ ($$$1$$$), $$$3$$$ ($$$1$$$). Sum equals $$$2$$$.

Town $$$3$$$: Goods from towns $$$3$$$ ($$$0$$$), $$$2$$$ ($$$1$$$), $$$4$$$ ($$$1$$$). Sum equals $$$2$$$.

Town $$$4$$$: Goods from towns $$$4$$$ ($$$0$$$), $$$1$$$ ($$$1$$$), $$$5$$$ ($$$1$$$). Sum equals $$$2$$$.

Town $$$5$$$: Goods from towns $$$5$$$ ($$$0$$$), $$$4$$$ ($$$1$$$), $$$3$$$ ($$$2$$$). Sum equals $$$3$$$.

B. Petr and Permutations

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petr likes to come up with problems about randomly generated data. This time problem is about random permutation. He decided to generate a random permutation this way: he takes identity permutation of numbers from $$$1$$$ to $$$n$$$ and then $$$3n$$$ times takes a random pair of different elements and swaps them. Alex envies Petr and tries to imitate him in all kind of things. Alex has also come up with a problem about random permutation. He generates a random permutation just like Petr but swaps elements $$$7n+1$$$ times instead of $$$3n$$$ times. Because it is more random, OK?!

You somehow get a test from one of these problems and now you want to know from which one.

Input
In the first line of input there is one integer $$$n$$$ ($$$10^{3} \le n \le 10^{6}$$$).

In the second line there are $$$n$$$ distinct integers between $$$1$$$ and $$$n$$$ — the permutation of size $$$n$$$ from the test.

It is guaranteed that all tests except for sample are generated this way: First we choose $$$n$$$ — the size of the permutation. Then we randomly choose a method to generate a permutation — the one of Petr or the one of Alex. Then we generate a permutation using chosen method.

Output
If the test is generated via Petr's method print "`Petr`" (without quotes). If the test is generated via Alex's method print "`Um_nik`" (without quotes).

Example

input
5 2 4 5 1 3
output
Petr

Note
Please note that the sample is not a valid test (because of limitations for $$$n$$$) and is given only to illustrate input/output format. Your program **still has to print correct answer to this test** to get AC.

Due to randomness of input hacks in this problem are forbidden.

C. AND Graph

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a set of size $$$m$$$ with integer elements between $$$0$$$ and $$$2^{n}-1$$$ inclusive. Let's build an undirected graph on these integers in the following way: connect two integers $$$x$$$ and $$$y$$$ with an edge if and only if $$$x \& y = 0$$$. Here $$$\&$$$ is the bitwise AND operation. Count the number of connected components in that graph.

Input
In the first line of input there are two integers $$$n$$$ and $$$m$$$ ($$$0 \le n \le 22$$$, $$$1 \le m \le 2^{n}$$$).

In the second line there are $$$m$$$ integers $$$a_1, a_2, \ldots, a_m$$$ ($$$0 \le a_{i} < 2^{n}$$$) — the elements of the set. All $$$a_{i}$$$ are distinct.

Output
Print the number of connected components.

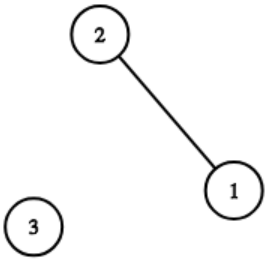
Examples

input
2 3 1 2 3
output
2

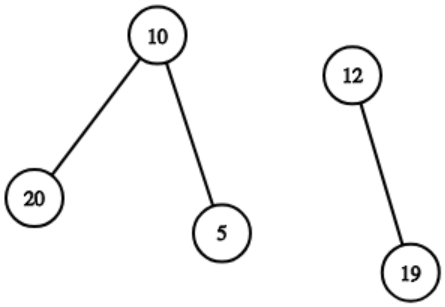
input
5 5 5 19 10 20 12
output

```
2
```

Note
Graph from first sample:



Graph from second sample:



D. Perfect Encoding

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are working as an analyst in a company working on a new system for big data storage. This system will store $$$n$$$ different objects. Each object should have a unique ID.

To create the system, you choose the parameters of the system — integers $$$m \ge 1$$$ and $$$b_{1}, b_{2}, \ldots, b_{m}$$$. With these parameters an ID of some object in the system is an array of integers $$$[a_{1}, a_{2}, \ldots, a_{m}]$$$ where $$$1 \le a_{i} \le b_{i}$$$ holds for every $$$1 \le i \le m$$$.

Developers say that production costs are proportional to $$$\sum_{i=1}^{m} b_{i}$$$. You are asked to choose parameters $$$m$$$ and $$$b_{i}$$$ so that the system will be able to assign unique IDs to $$$n$$$ different objects and production costs are minimized. Note that you don't have to use all available IDs.

Input
In the only line of input there is one positive integer $$$n$$$. The length of the decimal representation of $$$n$$$ is no greater than $$$1.5 \cdot 10^{6}$$$. The integer does not contain leading zeros.

Output
Print one number — minimal value of $$$\sum_{i=1}^{m} b_{i}$$$.

Examples

input
36
output
10

input
37
output
11

input
123456789012345678901234567890
output
177

E. Prince's Problem

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Let the main characters of this problem be personages from some recent movie. New Avengers seem to make a lot of buzz. I didn't watch any part of the franchise and don't know its heroes well, but it won't stop me from using them in this problem statement. So, Thanos and Dr. Strange are doing their superhero and supervillain stuff, but then suddenly they stumble across a regular competitive programming problem.

You are given a tree with $$$n$$$ vertices.

In each vertex $$$v$$$ there is positive integer $$$a_{v}$$$.

You have to answer $$$q$$$ queries.

Each query has a from $$$u$$$ $$$v$$$ $$$x$$$.

You have to calculate $$$\prod_{w \in P} gcd(x, a_{w}) \mod (10^{9} + 7)$$$, where $$$P$$$ is a set of vertices on path from $$$u$$$ to $$$v$$$. In other words, you are to calculate the product of $$$gcd(x, a_{w})$$$ for all vertices $$$w$$$ on the path from $$$u$$$ to $$$v$$$. As it might be large, compute it modulo $$$10^9+7$$$. Here $$$gcd(s, t)$$$ denotes the greatest common divisor of $$$s$$$ and $$$t$$$.

Note that the numbers in vertices **do not** change after queries.

I suppose that you are more interested in superhero business of Thanos and Dr. Strange than in them solving the problem. So you are invited to solve this problem instead of them.

Input

In the first line of input there is one integer $$$n$$$ ($$$1 \le n \le 10^{5}$$$) — the size of the tree.

In the next $$$n-1$$$ lines the edges of the tree are described. The $$$i$$$-th edge is described with two integers $$$u_{i}$$$ and $$$v_{i}$$$ ($$$1 \le u_{i}, v_{i} \le n$$$) and it connects the vertices $$$u_{i}$$$ and $$$v_{i}$$$. It is guaranteed that graph with these edges is a tree.

In the next line there are $$$n$$$ integers $$$a_1, a_2, \ldots, a_n$$$ ($$$1 \le a_{v} \le 10^{7}$$$).

In the next line there is one integer $$$q$$$ ($$$1 \le q \le 10^{5}$$$) — the number of queries.

And in the next $$$q$$$ lines the queries are described. Each query is described with three integers $$$u_{i}$$$, $$$v_{i}$$$ and $$$x_{i}$$$ ($$$1 \le u_{i}, v_{i} \le n$$$, $$$1 \le x_{i} \le 10^{7}$$$).

Output

Print $$$q$$$ numbers — the answers to the queries in the order they are given in the input. Print each answer modulo $$$10^9+7 = 1000000007$$$. Print each number on a separate line.

Examples

input
4 1 2 1 3 1 4 6 4 9 5 3 2 3 6 2 3 2 3 4 7
output
36 4 1

input
6 1 2

```
2 3
2 4
1 5
5 6
100000 200000 500000 40000 800000 250000
3
3 5 10000000
6 2 3500000
4 1 64000
```

output

```
196000
12250
999998215
```

F. Oppa Funcan Style Remastered

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Surely you have seen insane videos by South Korean rapper PSY, such as "Gangnam Style", "Gentleman" and "Daddy". You might also hear that PSY has been recording video "Oppa Funcan Style" two years ago (unfortunately we couldn't find it on the internet). We will remind you what this hit looked like (you can find original description here):

On the ground there are $$$n$$$ platforms, which are numbered with integers from $$$1$$$ to $$$n$$$, on $$$i$$$-th platform there is a dancer with number $$$i$$$. Further, every second all the dancers standing on the platform with number $$$i$$$ jump to the platform with the number $$$f(i)$$$. The moving rule $$$f$$$ is selected in advance and is not changed throughout the clip.

The duration of the clip was $$$k$$$ seconds and the rule $$$f$$$ was chosen in such a way that after $$$k$$$ seconds all dancers were in their initial positions (i.e. the $$$i$$$-th dancer stood on the platform with the number $$$i$$$). That allowed to loop the clip and collect even more likes.

PSY knows that enhanced versions of old artworks become more and more popular every day. So he decided to release a remastered-version of his video.

In his case "enhanced version" means even more insanity, so the number of platforms can be up to $$$10^{18}$$$! But the video director said that if some dancer stays on the same platform all the time, then the viewer will get bored and will turn off the video immediately. Therefore, for all $$$x$$$ from $$$1$$$ to $$$n$$$ $$$f(x) \neq x$$$ must hold.

Big part of classic video's success was in that looping, so in the remastered version all dancers should return to their initial positions in the end of the clip as well.

PSY hasn't decided on the exact number of platforms and video duration yet, so he asks you to check if there is a good rule $$$f$$$ for different options.

Input

In the first line of input there is one integer $$$t$$$ ($$$1 \le t \le 10^{4}$$$) — the number of options for $$$n$$$ and $$$k$$$ to check.

In the next $$$t$$$ lines the options are given: each option is described with two integers $$$n$$$ and $$$k$$$ ($$$1 \le n \le 10^{18}$$$, $$$1 \le k \le 10^{15}$$$) — the number of dancers and the duration in seconds.

It is guaranteed that the number of different values of $$$k$$$ in one test is not greater than $$$50$$$.

Output

Print $$$t$$$ lines. If the $$$i$$$-th option of the video is feasible, print "YES" (without quotes) in $$$i$$$-th line, otherwise print "NO" (without quotes).

Example

input

```
3
7 7
3 8
5 6
```

output

```
YES
NO
YES
```