A. Little Elephant and Bits

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

The Little Elephant has an integer a, written in the binary notation. He wants to write this number on a piece of paper.

To make sure that the number a fits on the piece of paper, the Little Elephant **ought** to delete exactly one any digit from number a in the binary record. At that a new number appears. It consists of the remaining binary digits, written in the corresponding order (possible, with leading zeroes).

The Little Elephant wants the number he is going to write on the paper to be as large as possible. Help him find the maximum number that he can obtain after deleting exactly one binary digit and print it in the binary notation.

Input

The single line contains integer a, written in the binary notation without leading zeroes. This number contains more than 1 and at most 10^5 digits.

Output

In the single line print the number that is written without leading zeroes in the binary notation — the answer to the problem.

Sample test(s)

input
```
101
```
output
```
11
```

input
```
110010
```
output
```
11010
```

Note

In the first sample the best strategy is to delete the second digit. That results in number $11_2 = 3_{10}$.

In the second sample the best strategy is to delete the third or fourth digits — that results in number $11010_2 = 26_{10}$.

B. Little Elephant and Elections

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

There have recently been elections in the zoo. Overall there were 7 main political parties: one of them is the Little Elephant Political Party, 6 other parties have less catchy names.

Political parties find their number in the ballot highly important. Overall there are m possible numbers: $1, 2, ..., m$. Each of these 7 parties is going to be assigned in some way to exactly one number, at that, two distinct parties cannot receive the same number.

The Little Elephant Political Party members believe in the lucky digits 4 and 7. They want to evaluate their chances in the elections. For that, they need to find out, how many correct assignments are there, such that the number of lucky digits in the Little Elephant Political Party ballot number is strictly larger than the total number of lucky digits in the ballot numbers of 6 other parties.

Help the Little Elephant Political Party, calculate this number. As the answer can be rather large, print the remainder from dividing it by 1000000007 $(10^9 + 7)$.

Input

A single line contains a single positive integer m $(7 \le m \le 10^9)$ — the number of possible numbers in the ballot.

Output

In a single line print a single integer — the answer to the problem modulo 1000000007 $(10^9 + 7)$.

Sample test(s)

input
```
7
```
output
```
0
```

input
```
8
```
output
```
1440
```

C. Little Elephant and LCM

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

The Little Elephant loves the LCM (least common multiple) operation of a non-empty set of positive integers. The result of the LCM operation of k positive integers $x_1, x_2, ..., x_k$ is the minimum positive integer that is divisible by each of numbers x_i.

Let's assume that there is a sequence of integers $b_1, b_2, ..., b_n$. Let's denote their LCMs as $lcm(b_1, b_2, ..., b_n)$ and the maximum of them as $max(b_1, b_2, ..., b_n)$. The Little Elephant considers a sequence b *good*, if $lcm(b_1, b_2, ..., b_n) = max(b_1, b_2, ..., b_n)$.

The Little Elephant has a sequence of integers $a_1, a_2, ..., a_n$. Help him find the number of good sequences of integers $b_1, b_2, ..., b_n$, such that for all i $(1 \le i \le n)$ the following condition fulfills: $1 \le b_i \le a_i$. As the answer can be rather large, print the remainder from dividing it by 1000000007 $(10^9 + 7)$.

Input

The first line contains a single positive integer n $(1 \le n \le 10^5)$ — the number of integers in the sequence a. The second line contains n space-separated integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 10^5)$ — sequence a.

Output

In the single line print a single integer — the answer to the problem modulo 1000000007 $(10^9 + 7)$.

Sample test(s)

input

```
4
1 4 3 2
```

output

```
15
```

input

```
2
6 3
```

output

```
13
```

D. Little Elephant and Broken Sorting

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

The Little Elephant loves permutations of integers from 1 to n very much. But most of all he loves sorting them. To sort a permutation, the Little Elephant repeatedly swaps some elements. As a result, he must receive a permutation $1, 2, 3, ..., n$.

This time the Little Elephant has permutation $p_1, p_2, ..., p_n$. Its sorting program needs to make exactly m moves, during the i-th move it swaps elements that are at that moment located at the a_i-th and the b_i-th positions. But the Little Elephant's sorting program happened to break down and now on every step it can equiprobably either do nothing or swap the required elements.

Now the Little Elephant doesn't even hope that the program will sort the permutation, but he still wonders: if he runs the program and gets some permutation, how much will the result of sorting resemble the sorted one? For that help the Little Elephant find the mathematical expectation of the number of permutation inversions after all moves of the program are completed.

We'll call a pair of integers i, j $(1 \le i < j \le n)$ an *inversion* in permutatuon $p_1, p_2, ..., p_n$, if the following inequality holds: $p_i > p_j$.

Input

The first line contains two integers n and m $(1 \le n, m \le 1000, n > 1)$ — the permutation size and the number of moves. The second line contains n distinct integers, not exceeding n — the initial permutation. Next m lines each contain two integers: the i-th line contains integers a_i and b_i $(1 \le a_i, b_i \le n, a_i \ne b_i)$ — the positions of elements that were changed during the i-th move.

Output

In the only line print a single real number — the answer to the problem. The answer will be considered correct if its relative or absolute error does not exceed 10^{-6}.

Sample test(s)

input

```
2 1
1 2
1 2
```

output

```
0.500000000
```

input

```
4 3
1 3 2 4
1 2
2 3
1 4
```

output

```
3.000000000
```

E. Little Elephant and Tree

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

The Little Elephant loves trees very much, he especially loves root trees.

He's got a tree consisting of n nodes (the nodes are numbered from 1 to n), with root at node number 1. Each node of the tree contains some list of numbers which initially is empty.

The Little Elephant wants to apply m operations. On the i-th operation $(1 \le i \le m)$ he first adds number i to lists of all nodes of a subtree with the root in node number a_i, and then he adds number i to lists of all nodes of the subtree with root in node b_i.

After applying all operations the Little Elephant wants to count for each node i number c_i — the number of integers j $(1 \le j \le n; j \ne i)$, such that the lists of the i-th and the j-th nodes contain at least one common number.

Help the Little Elephant, count numbers c_i for him.

Input

The first line contains two integers n and m $(1 \le n, m \le 10^5)$ — the number of the tree nodes and the number of operations.

Each of the following n - 1 lines contains two space-separated integers, u_i and v_i $(1 \le u_i, v_i \le n, u_i \ne v_i)$, that mean that there is an edge between nodes number u_i and v_i.

Each of the following m lines contains two space-separated integers, a_i and b_i $(1 \le a_i, b_i \le n, a_i \ne b_i)$, that stand for the indexes of the nodes in the i-th operation.

It is guaranteed that the given graph is an undirected tree.

Output

In a single line print n space-separated integers — $c_1, c_2, ..., c_n$.

Sample test(s)

input

```
5 1
1 2
1 3
3 5
3 4
2 3
```

output

```
0 3 3 3 3
```

input

```
11 3
1 2
2 3
2 4
1 5
5 6
5 7
5 8
6 9
8 10
8 11
2 9
3 6
2 8
```

output

```
0 6 7 6 0 2 0 5 4 5 5
```