Educational Codeforces Round 36 (Rated for Div. 2)

A. Garden

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Luba thinks about watering her garden. The garden can be represented as a segment of length k. Luba has got n buckets, the i-th bucket allows her to water some continuous subsegment of garden of length **exactly** a_i each hour. **Luba can't water any parts of the garden that were already watered, also she can't water the ground outside the garden.**

Luba has to choose **one** of the buckets in order to water the garden as fast as possible (as mentioned above, each hour she will water some continuous subsegment of length a_i if she chooses the i-th bucket). Help her to determine the minimum number of hours she has to spend watering the garden. It is guaranteed that Luba can always choose a bucket so it is possible water the garden.

See the examples for better understanding.

Input

The first line of input contains two integer numbers n and k ($1 \le n, k \le 100$) — the number of buckets and the length of the garden, respectively.

The second line of input contains n integer numbers a_i ($1 \le a_i \le 100$) — the length of the segment that can be watered by the i-th bucket in one hour.

It is guaranteed that there is at least one bucket such that it is possible to water the garden in integer number of hours using only this bucket.

Output

Print one integer number — the minimum number of hours required to water the garden.

Examples

input	Copy
3 6 2 3 5	

output
2

input	Copy
6 7 1 2 3 4 5 6	

output
7

Note

In the first test the best option is to choose the bucket that allows to water the segment of length 3. We can't choose the bucket that allows to water the segment of length 5 because then we can't water the whole garden.

In the second test we can choose only the bucket that allows us to water the segment of length 1.

B. Browser

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Luba is surfing the Internet. She currently has n opened tabs in her browser, indexed from 1 to n from left to right. The mouse cursor is currently located at the pos-th tab. Luba needs to use the tabs with indices from l to r (inclusive) for her studies, and she wants to close all the tabs that don't belong to this segment as fast as possible.

Each second Luba can either try moving the cursor to the left or to the right (if the cursor is currently at the tab i, then she can move it to the tab $max(i - 1, a)$ or to the tab $min(i + 1, b)$) or try closing all the tabs to the left or to the right of the cursor (if the cursor is currently at the tab i, she can close all the tabs with indices from segment $[a, i - 1]$ or from segment $[i + 1, b]$). In the aforementioned expressions a and b denote the minimum and maximum index of an unclosed tab, respectively. For example, if there were 7 tabs initially and tabs 1, 2 and 7 are closed, then $a = 3$, $b = 6$.

What is the minimum number of seconds Luba has to spend in order to leave **only the tabs with initial indices from l to r inclusive** opened?

Input

The only line of input contains four integer numbers n, pos, l, r ($1 \le n \le 100$, $1 \le pos \le n$, $1 \le l \le r \le n$) — the number of the tabs, the cursor position and the segment which Luba needs to leave opened.

Output

Print one integer equal to the minimum number of seconds required to close all the tabs outside the segment $[l, r]$.

Examples

input	Copy
6 3 2 4	
output	
5	

input	Copy
6 3 1 3	
output	
1	

input	Copy
5 2 1 5	
output	
0	

Note

In the first test Luba can do the following operations: shift the mouse cursor to the tab 2, close all the tabs to the left of it, shift the mouse cursor to the tab 3, then to the tab 4, and then close all the tabs to the right of it.

In the second test she only needs to close all the tabs to the right of the current position of the cursor.

In the third test Luba doesn't need to do anything.

C. Permute Digits

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two positive integer numbers a and b. Permute (change order) of the digits of a to construct maximal number not exceeding b. No number in input and/or output can start with the digit 0.

It is allowed to leave a as it is.

Input

The first line contains integer a ($1 \le a \le 10^{18}$). The second line contains integer b ($1 \le b \le 10^{18}$). Numbers don't have leading zeroes. It is guaranteed that answer exists.

Output

Print the maximum possible number that is a permutation of digits of a and is not greater than b. The answer can't have any leading zeroes. It is guaranteed that the answer exists.

The number in the output should have exactly the same length as number a. It should be a permutation of digits of a.

Examples

input	Copy
123 222	

output
213

input	Copy
3921 10000	

output
9321

input	Copy
4940 5000	

output
4940

D. Almost Acyclic Graph

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a directed graph consisting of n vertices and m edges (each edge is directed, so it can be traversed in only one direction). You are allowed to remove at most one edge from it.

Can you make this graph acyclic by removing at most one edge from it? A directed graph is called acyclic iff it doesn't contain any cycle (a non-empty path that starts and ends in the same vertex).

Input

The first line contains two integers n and m ($2 \le n \le 500$, $1 \le m \le min(n(n - 1), 100000)$) — the number of vertices and the number of edges, respectively.

Then m lines follow. Each line contains two integers u and v denoting a directed edge going from vertex u to vertex v ($1 \le u, v \le n$, $u \ne v$). Each ordered pair (u, v) is listed at most once (there is at most one directed edge from u to v).

Output

If it is possible to make this graph acyclic by removing at most one edge, print `YES`. Otherwise, print `NO`.

Examples

input Copy

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

output

```
YES
```

input Copy

```
5 6
1 2
2 3
3 2
3 1
2 1
4 5
```

output

```
NO
```

Note

In the first example you can remove edge $2 \to 3$, and the graph becomes acyclic.

In the second example you have to remove at least two edges (for example, $2 \to 1$ and $2 \to 3$) in order to make the graph acyclic.

E. Physical Education Lessons

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This year Alex has finished school, and now he is a first-year student of Berland State University. For him it was a total surprise that even though he studies programming, he still has to attend physical education lessons. The end of the term is very soon, but, unfortunately, Alex still hasn't attended a single lesson!

Since Alex doesn't want to get expelled, he wants to know the number of working days left until the end of the term, so he can attend physical education lessons during these days. But in BSU calculating the number of working days is a complicated matter:

There are n days left before the end of the term (numbered from 1 to n), and initially all of them are working days. Then the university staff sequentially publishes q orders, one after another. Each order is characterised by three numbers l, r and k:

- If $k = 1$, then all days from l to r (inclusive) become non-working days. If some of these days are made working days by some previous order, then these days still become non-working days;
- If $k = 2$, then all days from l to r (inclusive) become working days. If some of these days are made non-working days by some previous order, then these days still become working days.

Help Alex to determine the number of working days left after each order!

Input

The first line contains one integer n, and the second line — one integer q ($1 \le n \le 10^9$, $1 \le q \le 3 \cdot 10^5$) — the number of days left before the end of the term, and the number of orders, respectively.

Then q lines follow, i-th line containing three integers l_i, r_i and k_i representing i-th order ($1 \le l_i \le r_i \le n$, $1 \le k_i \le 2$).

Output

Print q integers. i-th of them must be equal to the number of working days left until the end of the term after the first i orders are published.

Example

input Copy

```
4
6
1 2 1
3 4 1
2 3 2
1 3 2
2 4 1
1 4 2
```

output

```
2
0
2
3
1
4
```

F. Imbalance Value of a Tree

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

You are given a tree T consisting of n vertices. A number is written on each vertex; the number written on vertex i is a_i. Let's denote the function $I(x, y)$ as the difference between maximum and minimum value of a_i on a simple path connecting vertices x and y.

Your task is to calculate $\sum_{i=1}^{n}\sum_{j=i}^{n} I(i,j)$.

Input

The first line contains one integer number n ($1 \le n \le 10^6$) — the number of vertices in the tree.

The second line contains n integer numbers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^6$) — the numbers written on the vertices.

Then $n - 1$ lines follow. Each line contains two integers x and y denoting an edge connecting vertex x and vertex y ($1 \le x, y \le n, x \ne y$). It is guaranteed that these edges denote a tree.

Output

Print one number equal to $\sum_{i=1}^{n}\sum_{j=i}^{n} I(i,j)$.

Example

input Copy

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

output

```
6
```

G. Coprime Arrays

time limit per test: 3.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's call an array a of size n *coprime* iff $gcd(a_1, a_2, ..., a_n) = 1$, where gcd is the greatest common divisor of the arguments.

You are given two numbers n and k. For each i $(1 \le i \le k)$ you have to determine the number of *coprime* arrays a of size n such that for every j $(1 \le j \le n)$ $1 \le a_j \le i$. Since the answers can be very large, you have to calculate them modulo $10^9 + 7$.

Input

The first line contains two integers n and k $(1 \le n, k \le 2 \cdot 10^6)$ — the size of the desired arrays and the maximum upper bound on elements, respectively.

Output

Since printing $2 \cdot 10^6$ numbers may take a lot of time, you have to output the answer in such a way:

Let b_i be the number of *coprime* arrays with elements in range $[1, i]$, taken modulo $10^9 + 7$. You have to print $\sum_{i=1}^{k} (b_i \oplus i)$, taken modulo $10^9 + 7$.

Here $\oplus$ denotes bitwise xor operation (^ in C++ or Java, `xor` in Pascal).

Examples

input	Copy
3 4	

output
82

input	Copy
2000000 8	

output
339310063

Note

Explanation of the example:

Since the number of *coprime* arrays is large, we will list the arrays that are non-coprime, but contain only elements in range $[1, i]$:

For $i = 1$, the only array is coprime. $b_1 = 1$.

For $i = 2$, array $[2, 2, 2]$ is not coprime. $b_2 = 7$.

For $i = 3$, arrays $[2, 2, 2]$ and $[3, 3, 3]$ are not coprime. $b_3 = 25$.

For $i = 4$, arrays $[2, 2, 2]$, $[3, 3, 3]$, $[2, 2, 4]$, $[2, 4, 2]$, $[2, 4, 4]$, $[4, 2, 2]$, $[4, 2, 4]$, $[4, 4, 2]$ and $[4, 4, 4]$ are not coprime. $b_4 = 55$.