Codeforces Round #227 (Div. 2)

A. George and Sleep

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

George woke up and saw the current time s on the digital clock. Besides, George knows that he has slept for time t.

Help George! Write a program that will, given time s and t, determine the time p when George went to bed. Note that George could have gone to bed yesterday relatively to the current time (see the second test sample).

Input

The first line contains current time s as a string in the format "`hh:mm`". The second line contains time t in the format "`hh:mm`" — the duration of George's sleep. It is guaranteed that the input contains the correct time in the 24-hour format, that is, $00 \le hh \le 23$, $00 \le mm \le 59$.

Output

In the single line print time p — the time George went to bed in the format similar to the format of the time in the input.

Sample test(s)

input
05:50 05:44
output
00:06

input
00:00 01:00
output
23:00

input
00:01 00:00
output
00:01

Note

In the first sample George went to bed at "`00:06`". Note that you should print the time only in the format "`00:06`". That's why answers "`0:06`", "`00:6`" and others will be considered incorrect.

In the second sample, George went to bed yesterday.

In the third sample, George didn't do to bed at all.

B. George and Round

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

George decided to prepare a Codesecrof round, so he has prepared m problems for the round. Let's number the problems with integers 1 through m. George estimates the i-th problem's complexity by integer b_i.

To make the round *good*, he needs to put at least n problems there. Besides, he needs to have at least one problem with complexity exactly a_1, at least one with complexity exactly a_2, ..., and at least one with complexity exactly a_n. Of course, the round can also have problems with other complexities.

George has a poor imagination. It's easier for him to make some already prepared problem simpler than to come up with a new one and prepare it. George is magnificent at simplifying problems. He can simplify any already prepared problem with complexity c to any positive integer complexity d ($c \geq d$), by changing limits on the input data.

However, nothing is so simple. George understood that even if he simplifies some problems, he can run out of problems for a *good* round. That's why he decided to find out the minimum number of problems he needs to come up with in addition to the m he's prepared in order to make a good round. Note that George can come up with a new problem of any complexity.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 3000$) — the minimal number of problems in a good round and the number of problems George's prepared. The second line contains space-separated integers $a_1, a_2, ..., a_n$ ($1 \leq a_1 < a_2 < ... < a_n \leq 10^6$) — the requirements for the complexity of the problems in a good round. The third line contains space-separated integers $b_1, b_2, ..., b_m$ ($1 \leq b_1 \leq b_2 ... \leq b_m \leq 10^6$) — the complexities of the problems prepared by George.

Output

Print a single integer — the answer to the problem.

Sample test(s)

input
3 5 1 2 3 1 2 2 3 3
output
0

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

input
3 1 2 3 4 1
output
3

Note

In the first sample the set of the prepared problems meets the requirements for a good round.

In the second sample, it is enough to come up with and prepare two problems with complexities 2 and 3 to get a good round.

In the third sample it is very easy to get a good round if come up with and prepare extra problems with complexities: $2, 3, 4$.

C. George and Number

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

George is a cat, so he really likes to play. Most of all he likes to play with his array of positive integers b. During the game, George modifies the array by using special changes. Let's mark George's current array as $b_1, b_2, ..., b_{|b|}$ (record $|b|$ denotes the current length of the array). Then one change is a sequence of actions:

- Choose two distinct indexes i and j $(1 \le i, j \le |b|; i \ne j)$, such that $b_i \ge b_j$.
- Get number $v = concat(b_i, b_j)$, where $concat(x, y)$ is a number obtained by adding number y to the end of the decimal record of number x. For example, $concat(500, 10) = 50010$, $concat(2, 2) = 22$.
- Add number v to the end of the array. The length of the array will increase by one.
- Remove from the array numbers with indexes i and j. The length of the array will decrease by two, and elements of the array will become re-numbered from 1 to current length of the array.

George played for a long time with his array b and received from array b an array consisting of exactly one number p. Now George wants to know: what is the maximum number of elements array b could contain originally? Help him find this number. Note that originally the array could contain only **positive** integers.

Input

The first line of the input contains a single integer p $(1 \le p < 10^{100000})$. It is guaranteed that number p doesn't contain any leading zeroes.

Output

Print an integer — the maximum number of elements array b could contain originally.

Sample test(s)

input
9555
output
4

input
10000000005
output
2

input
800101
output
3

input
45
output
1

input
1000000000000001223300003342220044555
output
17

input
19992000
output
1

input
310200
output
2

Note

Let's consider the test examples:

- Originally array b can be equal to $\{5, 9, 5, 5\}$. The sequence of George's changes could have been: $\{5, 9, 5, 5\} \rightarrow \{5, 5, 95\} \rightarrow \{95, 55\} \rightarrow \{9555\}$.
- Originally array b could be equal to $\{1000000000, 5\}$. Please note that the array b cannot contain zeros.
- Originally array b could be equal to $\{800, 10, 1\}$.
- Originally array b could be equal to $\{45\}$. It cannot be equal to $\{4, 5\}$, because George can get only array $\{54\}$ from this array in one operation.

Note that the numbers can be very large.

D. George and Interesting Graph

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

George loves graphs. Most of all, he loves interesting graphs. We will assume that a directed graph is *interesting*, if it meets the following criteria:

- The graph doesn't contain any multiple arcs;
- There is vertex v (we'll call her the *center*), such that for any vertex of graph u, the graph contains arcs (u, v) and (v, u). Please note that the graph also contains loop (v, v).
- The outdegree of all vertexes except for the *center* equals two and the indegree of all vertexes except for the *center* equals two. The outdegree of vertex u is the number of arcs that go out of u, the indegree of vertex u is the number of arcs that go in u. Please note that the graph can contain loops.

However, not everything's that simple. George got a directed graph of n vertices and m arcs as a present. The graph didn't have any multiple arcs. As George loves interesting graphs, he wants to slightly alter the presented graph and transform it into an interesting one. In one alteration he can either remove an arbitrary existing arc from the graph or add an arbitrary arc to the graph.

George wonders: what is the minimum number of changes that he needs to obtain an interesting graph from the graph he's got as a present? Help George and find the answer to the question.

Input

The first line contains two space-separated integers n and m $(2 \le n \le 500, 1 \le m \le 1000)$ — the number of vertices and arcs in the presented graph.

Each of the next m lines contains two space-separated integers a_i, b_i $(1 \le a_i, b_i \le n)$ — the descriptions of the graph's arcs. Pair (a_i, b_i) means that the graph contains an arc from vertex number a_i to vertex number b_i. It is guaranteed that the presented graph doesn't contain multiple arcs.

Assume that the grah vertices are numbered 1 through n.

Output

Print a single integer — the answer to George's question.

Sample test(s)

input
3 7 1 1 2 2 3 1 1 3 3 2 2 3 3 3
output
0

input
3 6 1 1 2 2 3 1 3 2 2 3 3 3
output
1

input
3 1 2 2
output
6

Note

For more information about directed graphs, please visit: `http://en.wikipedia.org/wiki/Directed_graph`

In the first sample the graph already is interesting, its center is vertex 3.

E. George and Cards

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

George is a cat, so he loves playing very much.

Vitaly put n cards in a row in front of George. Each card has one integer written on it. All cards had distinct numbers written on them. Let's number the cards from the left to the right with integers from 1 to n. Then the i-th card from the left contains number p_i ($1 \le p_i \le n$).

Vitaly wants the row to have exactly k cards left. He also wants the i-th card from left to have number b_i written on it. Vitaly gave a task to George, to get the required sequence of cards using the remove operation n - k times.

In one *remove operation* George can choose w ($1 \le w$; w is not greater than the current number of cards in the row) contiguous cards (contiguous subsegment of cards). Let's denote the numbers written on these card as $x_1, x_2, ..., x_w$ (from the left to the right). After that, George can remove the card x_i, such that $x_i \le x_j$ for each j ($1 \le j \le w$). After the described operation George gets w pieces of sausage.

George wondered: what maximum number of pieces of sausage will he get in total if he reaches his goal and acts optimally well? Help George, find an answer to his question!

Input

The first line contains integers n and k ($1 \le k \le n \le 10^6$) — the initial and the final number of cards.

The second line contains n distinct space-separated integers $p_1, p_2, ..., p_n$ ($1 \le p_i \le n$) — the initial row of cards.

The third line contains k space-separated integers $b_1, b_2, ..., b_k$ — the row of cards that you need to get. It is guaranteed that it's possible to obtain the given row by using the remove operation for n - k times.

Output

Print a single integer — the maximum number of pieces of sausage that George can get if he acts optimally well.

Sample test(s)

input
3 2 2 1 3 1 3
output
1

input
10 5 1 2 3 4 5 6 7 8 9 10 2 4 6 8 10
output
30

Codeforces (c) Copyright 2010-2015 Mike Mirzayanov
The only programming contests Web 2.0 platform