Codeforces Round #142 (Div. 1)

A. Shifts

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

You are given a table consisting of n rows and m columns. Each cell of the table contains a number, 0 or 1. In one move we can choose some row of the table and cyclically shift its values either one cell to the left, or one cell to the right.

To *cyclically shift* a table row one cell to the right means to move the value of each cell, except for the last one, to the right neighboring cell, and to move the value of the last cell to the first cell. A cyclical shift of a row to the left is performed similarly, but in the other direction. For example, if we cyclically shift a row "00110" one cell to the right, we get a row "00011", but if we shift a row "00110" one cell to the left, we get a row "01100".

Determine the minimum number of moves needed to make some table column consist only of numbers 1.

Input

The first line contains two space-separated integers: n ($1 \le n \le 100$) — the number of rows in the table and m ($1 \le m \le 10^4$) — the number of columns in the table. Then n lines follow, each of them contains m characters "0" or "1": the j-th character of the i-th line describes the contents of the cell in the i-th row and in the j-th column of the table.

It is guaranteed that the description of the table contains no other characters besides "0" and "1".

Output

Print a single number: the minimum number of moves needed to get only numbers 1 in some column of the table. If this is impossible, print -1.

Sample test(s)

input
3 6 101010 000100 100000
output
3

input
2 3 111 000
output
-1

Note

In the first sample one way to achieve the goal with the least number of moves is as follows: cyclically shift the second row to the right once, then shift the third row to the left twice. Then the table column before the last one will contain only 1s.

In the second sample one can't shift the rows to get a column containing only 1s.

B. Planets

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

Goa'uld Apophis captured Jack O'Neill's team again! Jack himself was able to escape, but by that time Apophis's ship had already jumped to hyperspace. But Jack knows on what planet will Apophis land. In order to save his friends, Jack must repeatedly go through stargates to get to this planet.

Overall the galaxy has n planets, indexed with numbers from 1 to n. Jack is on the planet with index 1, and Apophis will land on the planet with index n. Jack can move between some pairs of planets through stargates (he can move in both directions); the transfer takes a positive, and, perhaps, for different pairs of planets unequal number of seconds. Jack begins his journey at time 0.

It can be that other travellers are arriving to the planet where Jack is currently located. In this case, Jack has to wait for exactly 1 second before he can use the stargate. That is, if at time t another traveller arrives to the planet, Jack can only pass through the stargate at time $t + 1$, unless there are more travellers arriving at time $t + 1$ to the same planet.

Knowing the information about travel times between the planets, and the times when Jack would not be able to use the stargate on particular planets, determine the minimum time in which he can get to the planet with index n.

Input

The first line contains two space-separated integers: n ($2 \le n \le 10^5$), the number of planets in the galaxy, and m ($0 \le m \le 10^5$) — the number of pairs of planets between which Jack can travel using stargates. Then m lines follow, containing three integers each: the i-th line contains numbers of planets a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), which are connected through stargates, and the integer transfer time (in seconds) c_i ($1 \le c_i \le 10^4$) between these planets. It is guaranteed that between any pair of planets there is at most one stargate connection.

Then n lines follow: the i-th line contains an integer k_i ($0 \le k_i \le 10^5$) that denotes the number of moments of time when other travellers arrive to the planet with index i. Then k_i distinct space-separated integers t_{ij} ($0 \le t_{ij} < 10^9$) follow, sorted in ascending order. An integer t_{ij} means that at time t_{ij} (in seconds) another traveller arrives to the planet i. It is guaranteed that the sum of all k_i does not exceed 10^5.

Output

Print a single number — the least amount of time Jack needs to get from planet 1 to planet n. If Jack can't get to planet n in any amount of time, print number -1.

Sample test(s)

input
4 6 1 2 2 1 3 3 1 4 8 2 3 4 2 4 5 3 4 3 0 1 3 2 3 4 0
output
7

input
3 1 1 2 3 0 1 3 0
output
-1

Note

In the first sample Jack has three ways to go from planet 1. If he moves to planet 4 at once, he spends 8 seconds. If he transfers to planet 3, he spends 3 seconds, but as other travellers arrive to planet 3 at time 3 and 4, he can travel to planet 4 only at time 5, thus spending 8 seconds in total. But if Jack moves to planet 2, and then — to planet 4, then he spends a total of only $2 + 5 = 7$ seconds.

In the second sample one can't get from planet 1 to planet 3 by moving through stargates.

C. Triangles

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

Alice and Bob don't play games anymore. Now they study properties of all sorts of graphs together. Alice invented the following task: she takes a complete undirected graph with n vertices, chooses some m edges and keeps them. Bob gets the $\frac{n(n-1)}{2} - m$ remaining edges.

Alice and Bob are fond of "triangles" in graphs, that is, cycles of length 3. That's why they wonder: what total number of triangles is there in the two graphs formed by Alice and Bob's edges, correspondingly?

Input

The first line contains two space-separated integers n and m ($1 \le n \le 10^6, 0 \le m \le 10^6$) — the number of vertices in the initial complete graph and the number of edges in Alice's graph, correspondingly. Then m lines follow: the i-th line contains two space-separated integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), — the numbers of the two vertices connected by the i-th edge in Alice's graph. It is guaranteed that Alice's graph contains no multiple edges and self-loops. It is guaranteed that the initial complete graph also contains no multiple edges and self-loops.

Consider the graph vertices to be indexed in some way from 1 to n.

Output

Print a single number — the total number of cycles of length 3 in Alice and Bob's graphs together.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is advised to use the `cin`, `cout` streams or the `%I64d` specifier.

Sample test(s)

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

output
3

input
5 3 1 2 2 3 1 3

output
4

Note

In the first sample Alice has 2 triangles: (1, 2, 3) and (2, 3, 4). Bob's graph has only 1 triangle : (1, 4, 5). That's why the two graphs in total contain 3 triangles.

In the second sample Alice's graph has only one triangle: (1, 2, 3). Bob's graph has three triangles: (1, 4, 5), (2, 4, 5) and (3, 4, 5). In this case the answer to the problem is 4.

D. Towers

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

The city of D consists of n towers, built consecutively on a straight line. The height of the tower that goes i-th (from left to right) in the sequence equals h_i. The city mayor decided to rebuild the city to make it *beautiful*. In a *beautiful* city all towers are are arranged in non-descending order of their height from left to right.

The rebuilding consists of performing several (perhaps zero) operations. An operation constitutes using a crane to take any tower and put it altogether on the top of some other neighboring tower. In other words, we can take the tower that stands i-th and put it on the top of either the $(i - 1)$-th tower (if it exists), or the $(i + 1)$-th tower (of it exists). The height of the resulting tower equals the sum of heights of the two towers that were put together. After that the two towers can't be split by any means, but more similar operations can be performed on the resulting tower. Note that after each operation the total number of towers on the straight line decreases by 1.

Help the mayor determine the minimum number of operations required to make the city beautiful.

Input

The first line contains a single integer n $(1 \le n \le 5000)$ — the number of towers in the city. The next line contains n space-separated integers: the i-th number h_i $(1 \le h_i \le 10^5)$ determines the height of the tower that is i-th (from left to right) in the initial tower sequence.

Output

Print a single integer — the minimum number of operations needed to make the city beautiful.

Sample test(s)

input
5 8 2 7 3 1
output
3

input
3 5 2 1
output
2

E. Gifts

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

Once upon a time an old man and his wife lived by the great blue sea. One day the old man went fishing and caught a real live gold fish. The fish said: "Oh ye, old fisherman! Pray set me free to the ocean and I will grant you with n gifts, any gifts you wish!". Then the fish gave the old man a list of gifts and their prices. Some gifts on the list can have the same names but distinct prices. However, there can't be two gifts with the same names and the same prices. Also, there can be gifts with distinct names and the same prices. The old man can ask for n names of items from the list. If the fish's list has p occurrences of the given name, then the old man can't ask for this name of item more than p times.

The old man knows that if he asks for s gifts of the same name, the fish will randomly (i.e. uniformly amongst all possible choices) choose s gifts of distinct prices with such name from the list. The old man wants to please his greedy wife, so he will choose the n names in such a way that he can get n gifts with the maximum price. Besides, he isn't the brightest of fishermen, so if there are several such ways, he chooses one of them uniformly.

The old man wondered, what is the probability that he can get n most expensive gifts. As the old man isn't good at probability theory, he asks you to help him.

Input

The first line contains two space-separated integers n and m $(1 \le n, m \le 1000)$ — the number of the old man's wishes and the number of distinct names in the goldfish's list, correspondingly. Then m lines follow: the i-th line first contains integer k_i $(k_i > 0)$ — the number of distinct prices of gifts with the i-th name, then k_i distinct space-separated integers c_{ij} $(1 \le c_{ij} \le 10^9)$, the gifts' prices.

It is guaranteed that the sum of all k_i doesn't exceed 1000. It is guaranteed that n is not greater than the total number of the gifts.

Output

On a single line print one real number — the probability of getting n most valuable gifts. The answer will be considered correct if its absolute or relative error does not exceed 10^{-9}.

Sample test(s)

input
3 1 3 10 20 30
output
1.000000000

input
3 2 1 40 4 10 20 30 40
output
0.166666667