Topic Stream Mashup: Graphs

A. New Year Transportation

2 s., 256 MB

Problem source: https://codeforces.com/problemset/problem/500/A

New Year is coming in Line World! In this world, there are n cells numbered by integers from 1 to n, as a $1 \times n$ board. People live in cells. However, it was hard to move between distinct cells, because of the difficulty of escaping the cell. People wanted to meet people who live in other cells.

So, user tncks0121 has made a transportation system to move between these cells, to celebrate the New Year. First, he thought of $n - 1$ positive integers $a_1, a_2, \cdots, a_{n-1}$. For every integer i where $1 \leq i \leq n - 1$ the condition $1 \leq a_i \leq n - i$ holds. Next, he made $n - 1$ portals, numbered by integers from 1 to $n - 1$. The i-th ($1 \leq i \leq n - 1$) portal connects cell i and cell $(i + a_i)$, and one can travel from cell i to cell $(i + a_i)$ using the i-th portal. Unfortunately, one cannot use the portal backwards, which means one cannot move from cell $(i + a_i)$ to cell i using the i-th portal. It is easy to see that because of condition $1 \leq a_i \leq n - i$ one can't leave the Line World using portals.

Currently, I am standing at cell 1, and I want to go to cell t. However, I don't know whether it is possible to go there. Please determine whether I can go to cell t by only using the construted transportation system.

Input

The first line contains two space-separated integers n ($3 \leq n \leq 3 \times 10^4$) and t ($2 \leq t \leq n$) — the number of cells, and the index of the cell which I want to go to.

The second line contains $n - 1$ space-separated integers $a_1, a_2, \cdots, a_{n-1}$ ($1 \leq a_i \leq n - i$). It is guaranteed, that using the given transportation system, one cannot leave the Line World.

Output

If I can go to cell t using the transportation system, print "YES". Otherwise, print "NO".

input

```
8 4
1 2 1 2 1 2 1
```

output

```
YES
```

input

```
8 5
1 2 1 2 1 1 1
```

output

```
NO
```

In the first sample, the visited cells are: $1, 2, 4$; so we can successfully visit the cell 4.

In the second sample, the possible cells to visit are: $1, 2, 4, 6, 7, 8$; so we can't visit the cell 5, which we want to visit.

B. Rumor

2 s., 256 MB

Problem source: https://codeforces.com/problemset/problem/893/C

Vova promised himself that he would never play computer games... But recently Firestorm — a well-known game developing company — published their newest game, World of Farcraft, and it became really popular. Of course, Vova started playing it.

Now he tries to solve a quest. The task is to come to a settlement named Overcity and spread a rumor in it.

Vova knows that there are n characters in Overcity. Some characters are friends to each other, and they share information they got. Also Vova knows that he can bribe each character so he or she starts spreading the rumor; i-th character wants c_i gold in exchange for spreading the rumor. When a character hears the rumor, he tells it to all his friends, and they start spreading the rumor to their friends (for free), and so on.

The quest is finished when all n characters know the rumor. What is the minimum amount of gold Vova needs to spend in order to finish the quest?

Take a look at the notes if you think you haven't understood the problem completely.

Input

The first line contains two integer numbers n and m ($1 \leq n \leq 10^5, 0 \leq m \leq 10^5$) — the number of characters in Overcity and the number of pairs of friends.

The second line contains n integer numbers c_i ($0 \leq c_i \leq 10^9$) — the amount of gold i-th character asks to start spreading the rumor.

Then m lines follow, each containing a pair of numbers (x_i, y_i) which represent that characters x_i and y_i are friends ($1 \leq x_i, y_i \leq n$, $x_i \neq y_i$). It is guaranteed that each pair is listed at most once.

Output

Print one number — the minimum amount of gold Vova has to spend in order to finish the quest.

input

```
5 2
2 5 3 4 8
1 4
4 5
```

output

```
10
```

input

```
10 0
1 2 3 4 5 6 7 8 9 10
```

output

```
55
```

input

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

output

```
15
```

In the first example the best decision is to bribe the first character (he will spread the rumor to fourth character, and the fourth one will spread it to fifth). Also Vova has to bribe the second and the third characters, so they know the rumor.

In the second example Vova has to bribe everyone.

In the third example the optimal decision is to bribe the first, the third, the fifth, the seventh and the ninth characters.

C. News Distribution

2 s., 256 MB

Problem source: https://codeforces.com/problemset/problem/1167/C

In some social network, there are n users communicating with each other in m groups of friends. Let's analyze the process of distributing some news between users.

Initially, some user x receives the news from some source. Then he or she sends the news to his or her friends (two users are friends if there is at least one group such that both of them belong to this group). Friends continue sending the news to their friends, and so on. The process ends when there is no pair of friends such that one of them knows the news, and another one doesn't know.

For each user x you have to determine what is the number of users that will know the news if initially only user x starts distributing it.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 5 \cdot 10^5$) — the number of users and the number of groups of friends, respectively.

Then m lines follow, each describing a group of friends. The i-th line begins with integer k_i ($0 \le k_i \le n$) — the number of users in the i-th group. Then k_i **distinct** integers follow, denoting the users belonging to the i-th group.

It is guaranteed that $\sum_{i=1}^{m} k_i \le 5 \cdot 10^5$.

Output

Print n integers. The i-th integer should be equal to the number of users that will know the news if user i starts distributing it.

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

D. NP-Hard Problem

2 s., 256 MB

Problem source: https://codeforces.com/problemset/problem/687/A

Recently, Pari and Arya did some research about NP-Hard problems and they found the minimum vertex cover problem very interesting.

Suppose the graph G is given. Subset A of its vertices is called a vertex cover of this graph, if for each edge uv there is at least one endpoint of it in this set, i.e. $u \in A$ or $v \in A$ (or both).

Pari and Arya have won a great undirected graph as an award in a team contest. Now they have to split it in two parts, but both of them want their parts of the graph to be a vertex cover.

They have agreed to give you their graph and you need to find two **disjoint** subsets of its vertices A and B, such that both A and B are vertex cover or claim it's impossible. Each vertex should be given to no more than one of the friends (or you can even keep it for yourself).

Input

The first line of the input contains two integers n and m ($2 \le n \le 100\,000, 1 \le m \le 100\,000$) — the number of vertices and the number of edges in the prize graph, respectively.

Each of the next m lines contains a pair of integers u_i and v_i ($1 \le u_i,\ v_i \le n$), denoting an undirected edge between u_i and v_i. It's guaranteed the graph won't contain any self-loops or multiple edges.

Output

If it's impossible to split the graph between Pari and Arya as they expect, print "`-1`" (without quotes).

If there are two disjoint sets of vertices, such that both sets are vertex cover, print their descriptions. Each description must contain two lines. The first line contains a single integer k denoting the number of vertices in that vertex cover, and the second line contains k integers — the indices of vertices. Note that because of $m \ge 1$, vertex cover cannot be empty.

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

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

In the first sample, you can give the vertex number 2 to Arya and vertices numbered 1 and 3 to Pari and keep vertex number 4 for yourself (or give it someone, if you wish).

In the second sample, there is no way to satisfy both Pari and Arya.

E. Bear and Friendship Condition

1 s., 256 MB

Problem source: https://codeforces.com/problemset/problem/771/A

Bear Limak examines a social network. Its main functionality is that two members can become friends (then they can talk with each other and share funny pictures).

There are n members, numbered 1 through n. m pairs of members are friends. Of course, a member can't be a friend with themselves.

Let `A-B` denote that members `A` and `B` are friends. Limak thinks that a network is *reasonable* if and only if the following condition is satisfied: For every three **distinct** members (`X`, `Y`, `Z`), if `X-Y` and `Y-Z` then also `X-Z`.

For example: if Alan and Bob are friends, and Bob and Ciri are friends, then Alan and Ciri should be friends as well.

Can you help Limak and check if the network is reasonable? Print "`YES`" or "`NO`" accordingly, without the quotes.

Input

The first line of the input contain two integers n and m ($3 \le n \le 150\,000, 0 \le m \le min(150\,000, \frac{n \cdot (n-1)}{2})$) — the number of members and the number of pairs of members that are friends.

The i-th of the next m lines contains two distinct integers a_i and b_i ($1 \le a_i, b_i \le n, a_i \ne b_i$). Members a_i and b_i are friends with each other. No pair of members will appear more than once in the input.

Output

If the given network is reasonable, print "`YES`" in a single line (without the quotes). Otherwise, print "`NO`" in a single line (without the quotes).

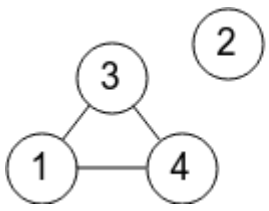
input
4 3 1 3 3 4 1 4
output
YES

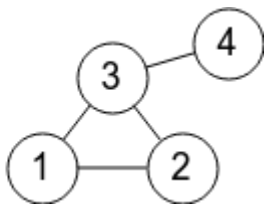
input
4 4 3 1 2 3 3 4 1 2
output
NO

input
10 4 4 3 5 10 8 9 1 2
output
YES

input
3 2 1 2 2 3
output
NO

The drawings below show the situation in the first sample (on the left) and in the second sample (on the right). Each edge represents two members that are friends. The answer is "`NO`" in the second sample because members $(2, 3)$ are friends and members $(3, 4)$ are friends, while members $(2, 4)$ are not.





F. Substring

3 s., 256 MB

Problem source: https://codeforces.com/problemset/problem/919/D

You are given a graph with n nodes and m **directed** edges. One lowercase letter is assigned to each node. We define a path's value as the number of the most frequently occurring letter. For example, if letters on a path are "`abaca`", then the value of that path is 3. Your task is find a path whose value is the largest.

Input

The first line contains two positive integers n, m $(1 \leq n, m \leq 300\,000)$, denoting that the graph has n nodes and m directed edges.

The second line contains a string s with only lowercase English letters. The i-th character is the letter assigned to the i-th node.

Then m lines follow. Each line contains two integers x, y $(1 \leq x, y \leq n)$, describing a directed edge from x to y. Note that x can be equal to y and there can be multiple edges between x and y. Also the graph can be not connected.

Output

Output a single line with a single integer denoting the largest value. If the value can be arbitrarily large, output `-1` instead.

input

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

output

```
3
```

input

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

output

```
-1
```

input

```
10 14
xzyzyzyzqx
1 2
2 4
3 5
4 5
2 6
6 8
6 5
2 10
3 9
10 9
4 6
1 10
2 8
3 7
```

output

```
4
```

In the first sample, the path with largest value is $1 \to 3 \to 4 \to 5$. The value is 3 because the letter '`a`' appears 3 times.

G. Contrived Paths

5 s., 256 MB

Problem source: me (but it's also not very original)

You have an undirected weighted graph with n vertices and m edges, and also an array a with length n. You also have a list b of k (possibly non-distinct) vertices. For each b_i, find the shortest path from b_i to every other vertex with a twist: you can use the input edges normally, but you can also move from any vertex i to any other vertex j for a cost of $a_i + a_j$.

Input

The first line of the input contains three integers, n, m, and k $(1 \leq n, m \leq 10^5; 1 \leq k \leq 10)$.

The next line contains n integers, representing the array a $(1 \leq a_i \leq 10^9)$.

The next m lines contain three integers u_i, v_i, and w_i $(1 \leq u_i, v_i \leq n; 1 \leq w_i \leq 10^9)$, representing an undirected edge between vertices u_i and v_i with cost w_i.

The last line contains k integers, representing the list b $(1 \leq b_i \leq n)$.

Output

Print k lines. On line i, print n integers, where the j-th integer represents the shortest path from b_i to j. Note that it's always possible for every vertex to reach every other vertex.

input

```
3 3 3
2 5 3
1 2 7
2 3 4
3 1 2
1 2 3
```

output

```
0 6 2
6 0 4
2 4 0
```

input

```
5 7 5
1 5 3 4 2
1 2 3
2 3 6
3 5 1
4 2 7
5 2 5
5 1 3
1 4 9
1 2 3 4 5
```

output

```
0 3 4 5 3
3 0 6 7 5
4 6 0 7 1
5 7 7 0 6
3 5 1 6 0
```

input

```
10 15 10
72 12 53 75 23 99 34 20 69 31
1 2 23
2 3 53
3 4 12
4 5 32
5 6 62
6 7 63
7 8 87
5 3 42
2 6 46
7 4 25
7 5 38
7 7 73
8 1 29
2 4 95
5 6 88
1 2 3 4 5 6 7 8 9 10
```

output

```
0 23 76 88 58 69 69 29 104 66
23 0 53 65 35 46 46 32 81 43
76 53 0 12 42 99 37 73 122 84
88 65 12 0 32 88 25 75 124 86
58 35 42 32 0 62 38 43 92 54
69 46 99 88 62 0 63 78 127 89
69 46 37 25 38 63 0 54 103 65
29 32 73 75 43 78 54 0 89 51
104 81 122 124 92 127 103 89 0 100
66 43 84 86 54 89 65 51 100 0
```

It may help to use https://csacademy.com/app/graph_editor/ to visualize the samples.

For the first sample: Vertex 1 can reach vertex 3 via the direct edge, and reach 2 by taking the path $1 \to 3 \to 2$. Vertex 2 can reach vertex 3 via the direct edge, and reach 1 by taking the path $2 \to 3 \to 1$. Vertex 3 can reach both vertices 1 and 2 via direct edges.

H. Maximum Distance

1 s., 256 MB

Problem source: https://codeforces.com/problemset/problem/1081/D

Chouti was tired of the tedious homework, so he opened up an old programming problem he created years ago.

You are given a connected undirected graph with n vertices and m weighted edges. There are k special vertices: $x_1, x_2, \ldots, x_k$.

Let's define the cost of the path as the **maximum** weight of the edges in it. And the *distance* between two vertexes as the **minimum** cost of the paths connecting them.

For each special vertex, find another special vertex which is farthest from it (in terms of the previous paragraph, i.e. the corresponding *distance* is maximum possible) and output the distance between them.

The original constraints are really small so he thought the problem was boring. Now, he raises the constraints and hopes you can solve it for him.

Input

The first line contains three integers n, m and k ($2 \leq k \leq n \leq 10^5$, $n - 1 \leq m \leq 10^5$) — the number of vertices, the number of edges and the number of special vertices.

The second line contains k distinct integers $x_1, x_2, \ldots, x_k$ ($1 \leq x_i \leq n$).

Each of the following m lines contains three integers u, v and w ($1 \leq u, v \leq n, 1 \leq w \leq 10^9$), denoting there is an edge between u and v of weight w. The given graph is undirected, so an edge (u, v) can be used in the both directions.

The graph may have multiple edges and self-loops.

It is guaranteed, that the graph is connected.

Output

The first and only line should contain k integers. The i-th integer is the distance between x_i and the farthest special vertex from it.

input

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

output

```
2 2
```

input

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

output

```
3 3 3
```

In the first example, the distance between vertex 1 and 2 equals to 2 because one can walk through the edge of weight 2 connecting them. So the distance to the farthest node for both 1 and 2 equals to 2.

In the second example, one can find that distance between 1 and 2, distance between 1 and 3 are both 3 and the distance between 2 and 3 is 2.

The graph may have multiple edges between and self-loops, as in the first example.

I. Make It Connected

2 s., 256 MB

Problem source: https://codeforces.com/problemset/problem/1095/F

You are given an undirected graph consisting of n vertices. A number is written on each vertex; the number on vertex i is a_i. Initially there are no edges in the graph.

You may add some edges to this graph, but you have to pay for them. The cost of adding an edge between vertices x and y is $a_x + a_y$ coins. There are also m special offers, each of them is denoted by three numbers x, y and w, and means that you can add an edge connecting vertices x and y and pay w coins for it. You don't have to use special offers: if there is a pair of vertices x and y that has a special offer associated with it, you still may connect these two vertices paying $a_x + a_y$ coins for it.

What is the minimum number of coins you have to spend to make the graph connected? Recall that a graph is connected if it's possible to get from any vertex to any other vertex using only the edges belonging to this graph.

Input

The first line contains two integers n and m ($1 \leq n \leq 2 \cdot 10^5$, $0 \leq m \leq 2 \cdot 10^5$) — the number of vertices in the graph and the number of special offers, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^{12}$) — the numbers written on the vertices.

Then m lines follow, each containing three integers x, y and w ($1 \leq x, y \leq n$, $1 \leq w \leq 10^{12}$, $x \neq y$) denoting a special offer: you may add an edge connecting vertex x and vertex y, and this edge will cost w coins.

Output

Print one integer — the minimum number of coins you have to pay to make the graph connected.

input

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

output

```
5
```

input

```
4 0
1 3 3 7
```

output

```
16
```

input

```
5 4
1 2 3 4 5
1 2 8
1 3 10
1 4 7
1 5 15
```

output

```
18
```

In the first example it is possible to connect 1 to 2 using special offer 2, and then 1 to 3 without using any offers.

In next two examples the optimal answer may be achieved without using special offers.

J. Vladik and Favorite Game

2 s., 256 MB

Problem source: https://codeforces.com/problemset/problem/811/D

This is an interactive problem.

Vladik has favorite game, in which he plays all his free time.

Game field could be represented as $n \times m$ matrix which consists of cells of three types:

- «`.`» — normal cell, player can visit it.
- «`F`» — finish cell, player has to finish his way there to win. There is exactly one cell of this type.
- «`*`» — dangerous cell, if player comes to this cell, he loses.

Initially player is located in the left top cell with coordinates $(1, 1)$.

Player has access to 4 buttons "U", "D", "L", "R", each of them move player up, down, left and right directions respectively.

But it's not that easy! Sometimes friends play game and change functions of buttons. Function of buttons "L" and "R" could have been swapped, also functions of buttons "U" and "D" could have been swapped. Note that functions of buttons can be changed only at the beginning of the game.

Help Vladik win the game!

Input

First line contains two space-separated integers n and m ($1 \le n, m \le 100$) — number of rows and columns respectively.

Each of next n lines contains m characters describing corresponding row of field. Set of characters in field is described above.

Guaranteed that cell with coordinates $(1, 1)$ is normal and there is at least one way from initial cell to finish cell without dangerous cells.

Interaction

You can press buttons no more than $2 \cdot n \cdot m$ times.

To press a button you should print "U", "D", "L", "R" in new line. It's necessary to print newline character and `flush` output. After flushing buffer you should read answer from input data. Answer is the pair of space-separated integers x, y — new position of player. In case, if there is no cell in direction of moving, position will not change. If after any move player lost, in other words player move to dangerous cell, then x and y will be equal to -1.

If after any move player is in finish or dangerous cell, then you should terminate your program.

To finish output buffer (i. e. for operation `flush`) right after printing direction and newline you should do next:

- `fflush(stdout)` in C++
- `System.out.flush()` in Java
- `stdout.flush()` in Python
- `flush(output)` in Pascal
- read documentation for other languages.

Hacks

To perform a hack you should use this format:

```
n m swapLR swapUD
a_1
a_2
...
a_n
```

Where n, m — number of rows and columns in game field. $swapLR$ is equal to 1 in case, when directions "L" and "R" is swapped, and equal to 0 otherwise. $swapUD$ is equal to 1, when directions "U" and "D" is swapped, and equal to 0 otherwise. $a_1, a_2, \ldots, a_n$ — description of corresponding rows of game field.

input

```
4 3
...
**.
F*.
...
1 1
1 2
1 3
1 3
2 3
3 3
4 3
4 2
4 1
3 1
```

output

```
R
L
L
D
U
U
U
R
R
D
```

In first test case all four directions swapped with their opposite directions. Protocol of interaction In more convenient form:

Judge	Contestant
4 3 ... **. F*. ...	
	R
1 1	
	L
1 2	
	L
1 3	
	D
1 3	
	U
2 3	
	U
3 3	
	U
4 3	
	R
4 2	
	R
4 1	
	D
3 1	

This test could be presenter for hack in following way:

```
4 3 1 1
...
**.
F*.
...
```

K. Koala and Notebook

2 s., 512 MB

Problem source: https://codeforces.com/problemset/problem/1209/F

Koala Land consists of m bidirectional roads connecting n cities. The roads are numbered from 1 to m by order in input. It is guaranteed, that one can reach any city from every other city.

Koala starts traveling from city 1. Whenever he travels on a road, he writes its number down in his notebook. He doesn't put spaces between the numbers, so they all get concatenated into a single number.

Before embarking on his trip, Koala is curious about the resulting number for all possible destinations. For each possible destination, what is the smallest number he could have written for it?

Since these numbers may be quite large, print their remainders modulo $10^9 + 7$. Please note, that you need to compute the remainder of the minimum possible number, **not** the minimum possible remainder.

Input

The first line contains two integers n and m ($2 \le n \le 10^5, n - 1 \le m \le 10^5$), the number of cities and the number of roads, respectively.

The i-th of the following m lines contains integers x_i and y_i ($1 \le x_i, y_i \le n, x_i \ne y_i$), representing a bidirectional road between cities x_i and y_i.

It is guaranteed, that for any pair of cities there is at most one road connecting them, and that one can reach any city from every other city.

Output

Print $n - 1$ integers, the answer for every city except for the first city.

The i-th integer should be equal to the smallest number he could have written for destination $i + 1$. Since this number may be large, output its remainder modulo $10^9 + 7$.

input

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

output

```
1
12
123
1234
12345
123456
1234567
12345678
123456789
345678826
```

input

```
12 19
1 2
2 3
2 4
2 5
2 6
2 7
2 8
2 9
2 10
3 11
11 12
1 3
1 4
1 5
1 6
1 7
1 8
1 9
1 10
```

output

```
1
12
13
14
15
16
17
18
19
1210
121011
```

input

```
12 14
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 10
10 11
11 12
1 3
1 4
1 10
```

output

```
1
12
13
134
1345
13456
1498
149
14
1410
141011
```