15-295 Fall 2018 #9

A. Mouse Hunt

2 seconds, 256 megabytes

Medicine faculty of Berland State University has just finished their admission campaign. As usual, about 80% of applicants are girls and majority of them are going to live in the university dormitory for the next 4 (hopefully) years.

The dormitory consists of n rooms and a single mouse! Girls decided to set mouse traps in some rooms to get rid of the horrible monster. Setting a trap in room number i costs c_i burles. Rooms are numbered from 1 to n.

Mouse doesn't sit in place all the time, it constantly runs. If it is in room i in second t then it will run to room a_i in second $t + 1$ without visiting any other rooms inbetween ($i = a_i$ means that mouse won't leave room i). It's second 0 in the start. If the mouse is in some room with a mouse trap in it, then the mouse get caught into this trap.

That would have been so easy if the girls actually knew where the mouse at. Unfortunately, that's not the case, mouse can be in any room from 1 to n at second 0.

What it the minimal total amount of burles girls can spend to set the traps in order to guarantee that the mouse will eventually be caught no matter the room it started from?

Input

The first line contains as single integers n ($1 \le n \le 2 \cdot 10^5$) — the number of rooms in the dormitory.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^4$) — c_i is the cost of setting the trap in room number i.

The third line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — a_i is the room the mouse will run to the next second after being in room i.

Output

Print a single integer — the minimal total amount of burles girls can spend to set the traps in order to guarantee that the mouse will eventually be caught no matter the room it started from.

input
5 1 2 3 2 10 1 3 4 3 3
output
3

input
4 1 10 2 10 2 4 2 2
output
10

input
7 1 1 1 1 1 1 1 2 2 2 3 6 7 6
output
2

In the first example it is enough to set mouse trap in rooms 1 and 4. If mouse starts in room 1 then it gets caught immideately. If mouse starts in any other room then it eventually comes to room 4.

In the second example it is enough to set mouse trap in room 2. If mouse starts in room 2 then it gets caught immideately. If mouse starts in any other room then it runs to room 2 in second 1.

Here are the paths of the mouse from different starts from the third example:

- $1 \to 2 \to 2 \to \ldots$;
- $2 \to 2 \to \ldots$;
- $3 \to 2 \to 2 \to \ldots$;
- $4 \to 3 \to 2 \to 2 \to \ldots$;
- $5 \to 6 \to 7 \to 6 \to \ldots$;
- $6 \to 7 \to 6 \to \ldots$;
- $7 \to 6 \to 7 \to \ldots$;

So it's enough to set traps in rooms 2 and 6.

B. Explosion Exploit

2.0 s, 256 MB

In a two player card game, you have n minions on the board and the opponent has m minions. Each minion has a health between 1 and 6.

You are contemplating your next move. You want to play an "Explosion" spell which deals d units of damage randomly distributed across all minions. The damage is dealt one unit at a time to some remaining minion on the board. Each living minion (including your own) has the same chance of receiving each unit of damage. When a minion receives a unit of damage, its health is decreased by one. As soon as the health of a minion reaches zero, it is immediately removed from the board, before the next damage is dealt. If there are no minions left on the board, any excess damage caused by the spell is ignored.

Given the current health of all minions, what is the probability that the Explosion will remove all of the opponent's minions? Note that it does not matter if all your own minions die in the process as well, and the damage continues to be dealt even if all your own minions are gone.

Input

The first line of input contains the three integers n, m, and d ($1 \leq n, m \leq 5$, $1 \leq d \leq 100$). Then follows a line containing n integers, the current health of all your minions. Finally, the third line contains m integers, the current health of all the opponent's minions. All healths are between 1 and 6 (inclusive).

Output

Output the probability that the Explosion removes all the opponent's minions, accurate up to an absolute error of 10^{-6}.

input
1 2 2 2 1 1
output
0.33333333

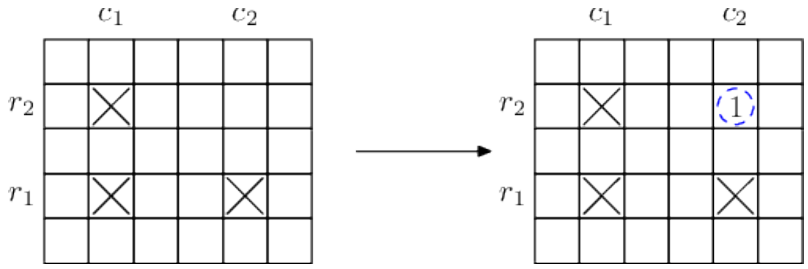
input
2 3 12 3 2 4 2 3
output
0.13773809

C. Chemical table

1 second, 512 megabytes

Innopolis University scientists continue to investigate the periodic table. There are $n \cdot m$ known elements and they form a periodic table: a rectangle with n rows and m columns. Each element can be described by its coordinates (r, c) $(1 \le r \le n, 1 \le c \le m)$ in the table.

Recently scientists discovered that for every four different elements in this table that form a rectangle with sides parallel to the sides of the table, if they have samples of three of the four elements, they can produce a sample of the fourth element using nuclear fusion. So if we have elements in positions (r_1, c_1), (r_1, c_2), (r_2, c_1), where $r_1 \neq r_2$ and $c_1 \neq c_2$, then we can produce element (r_2, c_2).



Samples used in fusion are not wasted and can be used again in future fusions. Newly crafted elements also can be used in future fusions.

Innopolis University scientists already have samples of q elements. They want to obtain samples of all $n \cdot m$ elements. To achieve that, they will purchase some samples from other laboratories and then produce all remaining elements using an arbitrary number of nuclear fusions in some order. Help them to find the minimal number of elements they need to purchase.

Input

The first line contains three integers n, m, q $(1 \le n, m \le 200\,000; 0 \le q \le min(n \cdot m, 200\,000))$, the chemical table dimensions and the number of elements scientists already have.

The following q lines contain two integers r_i, c_i $(1 \le r_i \le n, 1 \le c_i \le m)$, each describes an element that scientists already have. All elements in the input are different.

Output

Print the minimal number of elements to be purchased.

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

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

input
4 3 6 1 2 1 3 2 2 2 3 3 1 3 3
output
1

For each example you have a picture which illustrates it.

The first picture for each example describes the initial set of element samples available. Black crosses represent elements available in the lab initially.

The second picture describes how remaining samples can be obtained. Red dashed circles denote elements that should be purchased from other labs (the optimal solution should minimize the number of red circles). Blue dashed circles are elements that can be produced with nuclear fusion. They are numbered in order in which they can be produced.

Test 1

We can use nuclear fusion and get the element from three other samples, so we don't need to purchase anything.

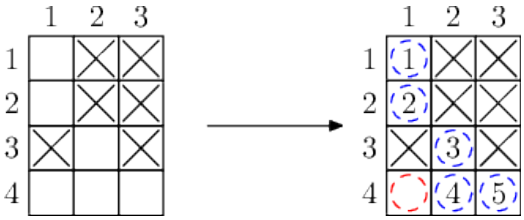
Test 2

We cannot use any nuclear fusion at all as there is only one row, so we have to purchase all missing elements.

Test 3

There are several possible solutions. One of them is illustrated below.

Note that after purchasing one element marked as red it's still not possible to immidiately produce the middle element in the bottom row (marked as 4). So we produce the element in the left-top corner first (marked as 1), and then use it in future fusions.



D. Connected Components?

2 seconds, 256 megabytes

You are given an undirected graph consisting of n vertices and $\frac{n(n-1)}{2} - m$ edges. Instead of giving you the edges that exist in the graph, we give you m unordered pairs (x, y) such that there is no edge between x and y, and if some pair of vertices is not listed in the input, then there is an edge between these vertices.

You have to find the number of connected components in the graph and the size of each component. A connected component is a set of vertices X such that for every two vertices from this set there exists at least one path in the graph connecting these vertices, but adding any other vertex to X violates this rule.

Input

The first line contains two integers n and m ($1 \le n \le 200000, 0 \le m \le min(\frac{n(n-1)}{2}, 200000)$).

Then m lines follow, each containing a pair of integers x and y ($1 \le x, y \le n, x \ne y$) denoting that **there is no edge** between x and y. Each pair is listed at most once; (x, y) and (y, x) are considered the same (so they are never listed in the same test). If some pair of vertices is not listed in the input, then there **exists** an edge between those vertices.

Output

Firstly print k — the number of connected components in this graph.

Then print k integers — the sizes of components. You should output these integers in non-descending order.

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

E. Heaps from Trees

5 seconds, 256 megabytes

You are given a rooted tree with n nodes. The nodes are labeled $1 \ldots n$, and node 1 is the root. Each node has a value v_i.

You would like to turn this tree into a heap. That is, you would like to choose the largest possible subset of nodes that satisfy this Heap Property: For every node pair i, j in the subset, if node i is an ancestor of node j in the tree, then $v_i > v_j$. Note that equality is not allowed.

Figure out the maximum number of nodes you can choose to form such a subset. The subset does not have to form a subtree.

Input

The first line of input will contain a single integer n ($1 \leq n \leq 2 \cdot 10^5$), which is the number of nodes in the tree. The nodes are numbered $1 \ldots n$.

Each of the next n lines will describe the nodes, in order. They will each contain two integers v_i and p_i, where v_i ($0 \leq v_i \leq 10^9$) is the value in the node, and p_i ($0 \leq p_i < i$) is the index of its parent. Every node's index will be strictly greater than its parent node's index. Only node 1, the root, will have $p_1 = 0$, since it has no parent. For all other nodes ($i = 2, \ldots, n$), $1 \leq p_i < i$.

Output

Output a single integer representing the number of nodes in the largest subset satisfying the Heap Property.

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

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

input
11 7 0 8 1 5 1 5 2 4 2 3 2 6 3 6 3 10 4 9 4 11 4
output
7

F. Leaf Sets

3 seconds, 256 megabytes

You are given an undirected tree, consisting of n vertices.

The vertex is called a leaf if it has exactly one vertex adjacent to it.

The distance between some pair of vertices is the number of edges in the shortest path between them.

Let's call some set of leaves *beautiful* if the maximum distance between any pair of leaves in it is less or equal to k.

You want to split **all** leaves into **non-intersecting** beautiful sets. What is the minimal number of sets in such a split?

Input

The first line contains two integers n and k ($3 \le n \le 10^6$, $1 \le k \le 10^6$) — the number of vertices in the tree and the maximum distance between any pair of leaves in each beautiful set.

Each of the next $n - 1$ lines contains two integers v_i and u_i ($1 \le v_i, u_i \le n$) — the description of the i-th edge.

It is guaranteed that the given edges form a tree.

Output

Print a single integer — the minimal number of beautiful sets the split can have.

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

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

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

Here is the graph for the first example: