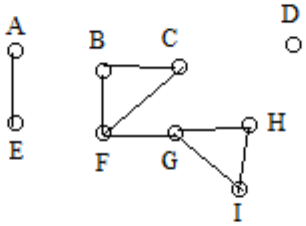
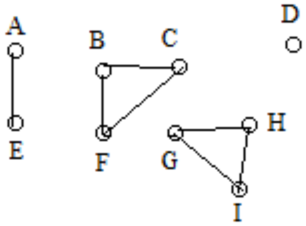
Problem Description: Destroy the connections

You have vowed to defeat your sworn enemy, but warfare in the 21st century isn't fought with guns. Knowledge is power, and knowledge is disseminated through computer networks. Thus, to defeat your sworn enemy, you plan on cutting the wired connections between pairs of computers to reduce the overall connectivity of your enemy's network.

We can model a computer network as pairs of connections between computers, also known as an undirected graph. We can define the connectivity of a network as the sum of the sizes squared of each of the separate components of this graph. For the example graph shown below, the current connectivity equals $2^2 + 6^2 + 1^2 = 41$.



If you were to destroy the connection between computers F and G, then the new network would look like this



and its connectivity would only be $2^2 + 3^2 + 3^2 + 1^2 = 23$.

The Problem:

Given a network of *n* computers, a set of the pairs of computers that are initially connected, and a sequence of steps where connections are destroyed, one by one, calculate the connectivity (as defined in the problem specification above) after each connection is destroyed.

The Input (must be read from standard input (no file i/o. use of file i/o will fail all the test cases and you will get zero)):

The first line of input contains three space separated integers, *n* ($1 \le n \le 10^5$), *m* ($1 \le m \le 3\text{x}10^5$), and *d* ($1 \le d \le m$), representing the number of computers in the enemy network, the number of connections between pairs of computers in the enemy network, and the number of those connections which you will destroy, respectively. The computers are numbered 1 through *n*, inclusive, and the connections are numbered 1 through *m* respectively.

The following *m* lines will each contain a pair of distinct integers, *u* and *v* ($1 \le u, v \le n, u \ne v$), representing that computers *u* and *v* are connected, initially. (Note: we denote the first of these lines as connection 1, the second as connection 2, and so forth.) It is guaranteed that each of the pairs listed will be unique pairs; namely, the same two values of *u* and *v* will never appear on two different lines as separate connections. (Of course, many individual computers will be connected to more than one other computer, so a particular value *u* may appear on more than one of these *m* lines.)

The following *d* lines will each contain a unique integer in between 1 and *m*, inclusive, representing a connection number that gets destroyed. These will appear in the order that they get destroyed.

The Output (standard console output):

Output *d*+1 lines of output. The first line should be the initial connectivity of the network. The following *d* lines should have the connectivity of the network after each connection is destroy, one by one.

Sample Input	Sample Output
9 8 2 1 5 2 3 2 6 3 6 6 7 7 8 7 9 8 9 5 //it means 5th connection from the above list 3	41 23 23

3 3 3
1 2
1 3
2 3
3
1
2
	9
	9
	5
	3

Implementation Requirements

This assignment is testing the use of the disjoint set data structure. You must use a disjoint set to solve the problem even though other ways exist. You will have to modify the disjoint set shown in class to solve the given problem. The point of the assignment is to see if you can figure out how to do that modification on your own.

As always, your code should use good style, including but not limited to: a header comment, reasonable number of internal comments, good modular break down, good variable names, and utilizing objects and the Java API as appropriate for solving the problem,

What To Submit

For this assignment, please submit a single Java program named **Main.java** and **Lastname_testStrategy.txt** with brief explanation of your testing strategy

Hints:

1. The code will not be that straightforward, so you need to plan
2. Draw the sample input first and then see how the output numbers are calculated
3. You can use our existing disjoint set code and update it (make sure to use **path compression and union by rank approach**. Otherwise, your code may fail larger grading test cases due to optimization issue).
4. The union part of the cycle detection concept we have learned in the class could be useful