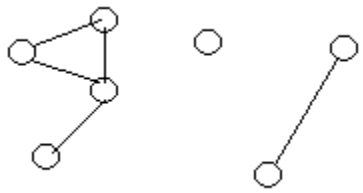
COP 3503 Homework #3: Making Connections

Filename: connect.java
Time Limit: 2 seconds (per input case)
Standard Input, Standard Output

Given that everything is online these days, connectivity is a must. A computer network can be modeled as a graph, where each computer is a vertex and each direct network connection between pairs of computers is an undirected edge.

Consider the process of building a computer network. At the very beginning there will be n computers, with no connections between any of them. Then, as time goes on, pairs of computers are chosen, one pair at a time, and a direct network connection is added. In the middle of such a process, we might get the following graph modeling the connections:



This network currently has 4 computers in one group that can communicate directly or indirectly, 1 computer by itself, and another 2 computers that can communicate with each other.

We can define the average connectivity of a network as the sum of the sizes squared of each of the separate components of this graph, divided by the number of components. For the example graph shown above, the current connectivity equals $(4^2 + 1^2 + 2^2)/3 = 21/3 = 7$.

As a network is being built, the project manager would like to know the average connectivity of the network at that given snapshot of time. Write a program to handle the queries as the network is being constructed!

The Problem:

Given a network of *n* initially separate computers, along with a sequence of steps, where either a pair of computers is connected or a query about the average connectivity is posted, answer each query.

The Input:

The first line of input contains two space separated integers, *n* ($1 \le n \le 10^5$) and *m* ($1 \le m \le 3\text{x}10^5$), where *n* represents the number of computers and *m* represents the total number of connections built and average connectivity queries. The computers are numbered 1 through *n*, inclusive.

The following *m* lines each contain information about one operation, in the order that they occur. Each of these lines will start with a single integer, either 1 or 2. If the first value on one of these lines is 1, it means that a pair of computers is being connected with a direct connection. The value

of 1 will be followed by u and v ($1 \leq u, v \leq n, u \neq v$), representing the pair of computers being connected with a direct connection. If the first value on one of these lines is 2, this is a query and no other information will be on the line. Note: It's possible that the same pair of computers may have more than one direct connection added during the process of connecting the computers. It's also possible that at the end of the process, that not all n computers are connected in the same component.

The Output:

For each query, output the average connectivity of the network at that point in time as a fraction in lowest terms on a line by itself. Specifically, output two integers, x and y with the character '/' in between, indicating that the average connectivity of the network at the time is x divided by y such that x and y share no common factors.

Sample Input **Sample Output**

7 9 2 1 1 2 1 1 3 2 1 3 4 1 2 3 2 1 6 7 2	1/1 13/5 19/4 7/1
4 9 1 1 2 2 1 3 4 2 1 2 3 2 1 1 4 1 2 4 2	2/1 4/1 16/1 16/1

Implementation Requirements

This assignment is testing the use of a disjoint set data structure.

The desired solution includes O(n) disjoint set method calls and O(n) calls to the gcd function (with numbers that fit in a long.) these two sets of method calls will be the bulk of the run time.

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, utilizing objects and the Java API as appropriate for solving the problem.

Note: If you do a careful analysis of the problem, you'll see that some of the output values can exceed the value that can be stored in an int variable. Thus, use the long type when necessary.

What To Submit

For this assignment, please submit a single Java program named **`connect.java`** which solves the posted problem.