Birthday Party

January 2024
C++ — 2 sec — 512 MB

Today is a jubilant day! Today is a wonderful day! Today is the anniversary of the birthday of the matriarch founder of M.O.T.H.E.R. (affectionately known as Mum). All the engineers at M.O.T.H.E.R. are organising a surprise party for Mum. It's all very hush-hush, don't you know.

Because it's being organised by computer engineers, Mum's birthday party is to be held online. Specifically, the engineers need to find a set of **s** interconnected computers on their (sprawling) network – one for each of the **s** guests they're going to invite.

The engineers have decided to call a set of **s** computers that are all *directly* connected to each other a *do* of size **s**. A *do* that cannot be extended by adding another computer (that would be directly connected to all other computers in the *do*) is maximal and has been termed a *social* of size **s**. In order to play many rounds of low-latency pass-the-parcel, the **s** computers must form a *social*.

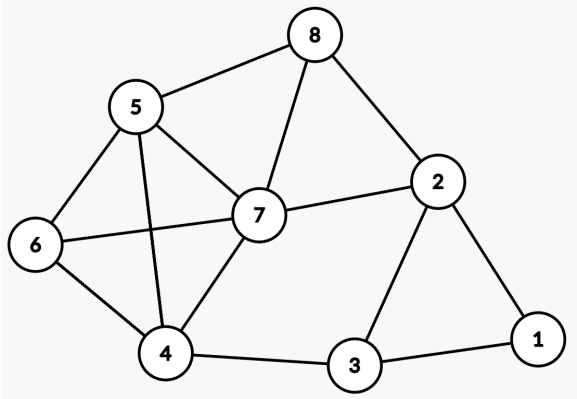
To work out the number of guest they can invite, the engineers want to determine the sizes of all the *socials* in their computer network.

INPUT You should continue to input lines containing two integers, **a** and **b**, until you receive the input -1 -1. For each line, there is a direction connection between computers **a** and **b**. No computer will have a direct connection to itself.

$1 \leq \mathbf{a}, \mathbf{b} \leq 10{,}000$

OUTPUT For each size of *social* (greater than 2) that appears in the computer network, output the size, followed by the number of *socials* of that size, in the network. Sizes should be outputted from smallest to largest and each size should be outputted on a new line. Any sizes that do not appear in the network should not be outputted.

SAMPLE For example, suppose 8 computers are connected in a network, as shown over-leaf. There are 3 *socials* of size 3: (1,2,3), (2,7,8), and (5,7,8); and 1 *social* of size 4: (4,5,6,7). There are no other *socials*.



INPUT	OUTPUT
1 2 1 3 2 3 3 4 4 5 5 6 4 5 4 6 4 7 4 5 4 6 5 8 5 7 6 7 2 8 2 7 7 8 -1 -1	3 3 4 1
1 2 1 3 1 4 1 5 2 3 2 4 2 5 3 4 3 5 4 5 -1 -1	5 1