F - Keeping On Track

Source file name: `ontrack.c`, `ontrack.cpp`, `ontrack.java`, *or* `ontrack.py`

Acmar and Ibmar are at war! You are in charge of a rail network that transports important supplies throughout the great state of Acmar during this delicate time. The rail system is made up of a set of rail lines which meet at various junction points. While there is no limit to the number of rail lines that can meet at a junction, the network is set up so that there is only one path between any two junctions. You've tried to argue for some redundancy in the system, i.e., extra rail lines so that there are two or more paths connecting some junctions, but it's wartime and budgets are tight.

However, this may soon change as you've just been given some terrible news from double agents working in Ibmar: within the next month enemy spies plan to blow up one of the junctions! Unfortunately, the exact junction is not known, but knowing your enemy well you are certain that they will undoubtedly strike the *critical junction*, specifically the junction whose removal disconnects the most pairs of other remaining junctions in the system. You don't have much time to act, so the most you can do is add one new line connecting two currently unconnected junctions, thereby reducing the number of disconnected pairs after the critical junction has been destroyed. Your job is to determine how to make the number of disconnected pairs as small as possible by adding in the best possible rail line.

Input

Input starts with a line containing an integer n ($2 \leq n \leq 10000$) indicating the number of rail lines in the system. Following that are n lines of the form i_1 i_2 indicating that a rail line connects junctions i_1 and i_2. Junctions are numbered consecutively starting at 0. All rail lines are two-way and no rail line appears more than once in the input. There is exactly one path between any two junction points given in the input.

The input must be read from standard input.

Output

Display two values n_1 and n_2, where n_1 is the number of pairs of junctions which will be disconnected when the enemy destroys the critical junction, and n_2 is the number of pairs of junctions still disconnected after you add in the best possible rail line. There will never be more than one critical junction.

The output must be written to standard output.

Sample Input 1	Sample Output 1
6 0 1 1 2 2 3 2 4 4 5 4 6	11 5

Sample Input 2	Sample Output 2
2 2 1 0 1	1 0