1   Earliest to become friends

In a social group, there are N people, with unique integer ids from 0 to N-1. There is a list of logs, where each logs[i] = [timestamp, id_A, id_B] contains a non-negative integer timestamp, and the ids of two different people.

Each log represents the time in which two different people became friends. Friendship is symmetric: if A is friends with B, then B is friends with A.

Let's say that person A is acquainted with person B if A is friends with B, or A is a friend of someone acquainted with B.

Return the earliest time for which every person became acquainted with every other person. Return -1 if there is no such earliest time.

Note: Timestamp is in format yyyymmdd

Input/Output

Input	Output	Comments
8 20190101 0 1 20190104 3 4 20190107 2 3 20190211 1 5 20190224 2 4 20190301 0 3 20190312 1 2 20190322 4 5 6	20190301	● First line 8 corresponds to total number of log entries. ● Next eight lines corresponds to *timestamp, id_A, id_B.* ● 10th Line 6 corresponds to the total number of people Explanation: ● The first event occurs at timestamp = *20190101*. 0 and 1 become friends. They form friendship groups [0,1], [2], [3], [4], [5]. ● The second event occurs at timestamp = *20190104*. 3 and 4 become friends. They form friendship groups [0,1], [2], [3,4], [5]. ● The third event occurs at timestamp = *20190107*. 2 and 3 become friends. They form friendship groups [0,1], [2,3,4], [5]. ● The fourth event occurs at timestamp = *20190211*. 1 and 5 become friends. They form friendship groups [0,1,5], [2,3,4]. ● The fifth event occurs at timestamp = *20190224*. 2 and 4 are already friend nothing happens. ● The sixth event occurs at timestamp = *20190301*. 0 and 3 become friends we have that all become friends. ● *No need to process the next two log entries.*
5 20190107 2 3 20190211 1 5 20190224 2 4	-1	● 1 and 5 does not have any other friend (0,2,3,4) ● There is no such time that all 6 people become friends

20190301 0 3 **20190104 3 4** **6**		

2   Delete and form sequence

From any string, we can form a subsequence of that string by deleting some number of characters (possibly no deletions).

Given two strings source and target, return the minimum number of subsequences of source such that their concatenation equals target. If the task is impossible, return -1.

Input/Output

Input	Output	Comments
abc **abcbc**	**2**	• First Line Represents Source • Second Line Represents Target • The target string **abcbc** can be formed by "abc" and "bc", which are subsequences of source "abc".
abc **acdbc**	**-1**	• The target string **acdbc** cannot be constructed from the subsequences of source string due to the character "d" in target string.
xyz **xzyxz**	3	• The target string **xzyxz** can be constructed as follows: "xz" + "y" + "xz"

3   How many can reach me?

There are n servers numbered from *0 to N-1* connected by undirected server-to-server connections forming a network where *connections[i] = [a, b]* represents a connection between servers *A* and *B*. Any server can reach any other server directly or indirectly through the network.

A critical connection is a connection that, if removed, will make some server unable to reach some other server.

Return all critical connections in the network in sorted order.

Input/Output

Input	Output	Comments
4 4 0 1 1 2 2 0 1 3	1 3	• First Line **4** indicates number of Servers • Second line **4** indicates number of Connections • Next **4** lines represents connections between servers. • 1 3 is critical connection if we remove this connection there is no other possibility to connect to **server 3**
6 5 0 1 1 2 5 4 2 0 1 3	1 3 4 5	• 5 4 and 1 3 are critical connections in the given input. • output should be in the sorted order as shown in the output column
4 5 0 1 1 2 2 3 3 1 0 2	-1	• No Critical Connection Found