Find Neo, ‘the One’

Neo is inside the Matrix and he urgently needs to get back to the Nebuchadnezzar. Tank must find Neo inside the Matrix using messages that Neo is sending. However, Neo can't just say where he is -- otherwise, the agents will get to him first! Instead, Neo uses an ingenious scheme that Morpheus has devised to help Tank find him.

Instead of sending one message, Neo sends a set of messages over the Matrix's communication network. Each router has a unique address and Tank has to figure out which router Neo is closest to.

If we consider the comm. network to be a graph, then the routers in the network are vertices and the links between the routers are edges, and Tank's job is to figure out the id (address) of the Neo's vertex. Each edge in the graph has a weight. As messages are transmitted over these edges, the weight of the link is multiplied to a product carried with the message. Neo always sets the initial value of the product to 1, and as messages traverse through the network, the number carried in the messages is the product of the edge weights of all the edges a message has been forwarded on.

The messages Neo sends are routed randomly throughout the network, and travel arbitrary distances. How-ever, each message can only travel over a single link once (and can only visit a given vertex once). The network topology is already known to Tank. For each message that Neo sends, Tank receives the id of the vertex that the message ends at (i.e. the node from which the message was not forwarded any more) and the product of the weights of edges the message took to get there. Given sufficient number of messages, using just this information, Tank can track Neo and get him back safely.

Morpheus was stumped the problem of the really large number of messages that must be needed in order for this technique to work, Trinity pointed out a very important properties of the Matrix communication graph: All the edges in the graph have prime number weights.

An example graph of this type is shown in Figure 1, consider an example in which Neo is on vertex 16 and sends a message that travels five hops 14, 5, 6, 4 and ends up at node 7. In this case, the information Tank gets is (node 7, product 12350).

Similarly, consider another Neo sends (again starting from vertex 16) that goes through nodes 14, 12, 13, and ends up at node 15. In this case, Tank gets the pair (node 15, product 532).

In both these cases, the only possible node that could have sent that message with those products happens to be node 16. However, consider a message that starts at node 16, and traverses nodes 14, 5, 7, 4, 6, 9, 10, 8, and finally ends up at node 11. In this case, the information Tank gets is (node 11, product 15042300). Tracing back, it turns out this message could have been originated from any of the nodes 16, 17, 18, or 19! So in general, a single message points back to a set of possible originating nodes

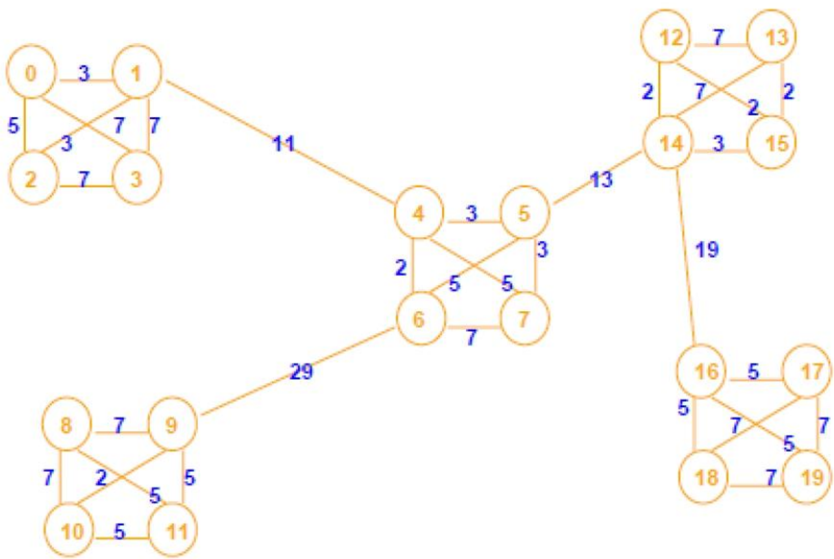

Figure 1: Example Graph

The problem is to find the message‟s actual originating node. The output should consist of list of all possible initiating nodes for each message, followed by the final actual initiating node (“NULL” if no such node exists)

The input contains the following 3 sections separated a commented line (starts with “#”).

1. Section 1: First line. 2 integers (N_V, N_E), number of nodes and edges respectively in the graph.
2. Section 2: Next N_E lines represent the edges, each a 3 number tuple (v1, v2, w) [an edge from node v1 to node v2 with weight w]
3. Section 3: This is followed by received messages; each line consisting of a 2 integer tuple (V_{end}, W_{prod}), the number of the node where the messages ends, and product of all edges the message has travelled through.

Sample Input: Corresponding to the Figure # 1.

#Graph
20 34

#Edge Set
0 3 7
0 2 5
0 1 3
...
5 14 13
14 16 19
#Messages
7 12350
15 532

Output for the Sample Input

16