

Problem J

Jabber Network

Dave, an old Computer Science professor, still maintains a local community computer network even after retirement. Each community member has a computer with three networking cards, and some of these cards may be connected by a cable. They form a connected network, and, following a long resource-saving tradition, the number of cables is kept to the minimum possible.

The habits of all the community members are quite stable: for every two computers the number of packets per second between them is known exactly. However, the network was first assembled a long time ago, so the connections are not necessarily be optimal any more. For two computers numbered i and j we define d_{ij} the shortest path between them, measured in the number of cables, and c_{ij} the number of packets per second that should be transferred from i to j . The *commutation stress* is defined to be the sum of $c_{ij} \cdot d_{ij}$ for all $i < j$, and one would like to minimise it.

Dave realised that it is finally the time to upgrade the cables — after all, they do degrade with time. He wants to take this opportunity to also optimise the network, such that the *commutation stress* becomes smaller. However, he is no longer as quick as in his youth, and his friends may get dissatisfied if too much disruption happens at once. So he decided that he will perform the upgrade using the following scenario. For each of the old cables, he will do the following:

1. Remove the old cable.
2. Connect the network back using a new cable, choosing the computers to connect in such a way that the resulting *commutation stress* is minimum possible.
3. If there are many ways to do this, break ties by choosing the computers with the smallest numbers: if (u_1, u_2) and (v_1, v_2) result in the same *commutation stress*, but $u_1 < v_1$ (or $u_1 = v_1$ and $u_2 < v_2$), then (u_1, u_2) should be chosen.

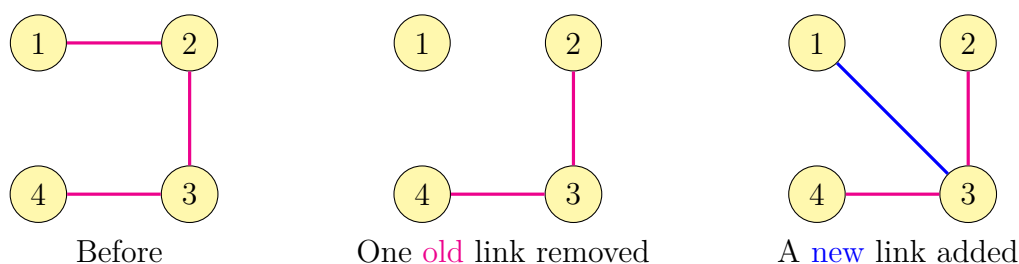


Figure J.1: A single reconnection operation (the first one in the sample input)

Note that, since each computer only has three network cards, Dave cannot connect two arbitrary computers on the second step: if one of them is already connected to three other computers, it is impossible to connect it to yet another computer. Fortunately, it is not hard to show that it is always possible to find two computers to connect: for instance, Dave can choose the two just-disconnected computers.

Unfortunately, the task appeared to be more difficult than it seemed initially. Could you help Dave?

Input

The first line of the input file contains an integer n ($2 \leq n \leq 2 \cdot 10^3$), the number of computers in the network.

The following $n - 1$ lines contain two integers each: a_i, b_i , where ($1 \leq a_i < b_i \leq n$) are the numbers of the computers initially connected by an old cable number i . The cables are to be removed and replaced in the order they are given in the input file. It is guaranteed that it is possible to reach any computer from any other computer using old cables (that is, the network is initially connected), and that no computer is connected with more than three other computers.

The next line contains an integer d ($2 \leq d \leq 10^4$), the number of computer pairs that are known to transmit data to each other.

The following d lines contain three integers each: s_i, t_i and d_i , where s_i and t_i are the numbers of the computers which transmit data to each other ($1 \leq s_i < t_i \leq n$), and d_i ($1 \leq d_i \leq 10^9$) is the number of packets per second to be transmitted.

Output

Output $n - 1$ lines containing two integers each: x_i, y_i , where ($1 \leq x_i < y_i \leq n$), should be the numbers of the computers connected by a new cable at step i .

Note that, due to the tie-breaking rule detailed above, the correct output is unique.

Sample Input 1

```
4
1 2
2 3
3 4
6
1 2 1
1 3 10
1 4 1
2 3 10
2 4 1
3 4 10
```

Sample Output 1

```
1 3
2 3
3 4
```