Disjoint Set Test

Time Limit: 1.4s **Memory Limit:** 256M

Xyene is doing a contest. He comes across the following problem:

You have a weighted graph of N ($1 \leq N \leq 100\,000$) vertices labelled from 1 to N and M ($1 \leq M \leq 1\,000\,000$) edges labelled from 1 to M. Each edge has a unique positive weight — specifically, an edge that shows up earlier in the input has a strictly less weight than an edge that shows up later in the input. The graph will not have multiple edges or self loops. From this information, determine the edges in the minimum spanning tree, or that one does not exist. The minimum spanning tree of this graph is guaranteed to be unique if it exists.

Xyene knows that one fast solution uses Kruskal's algorithm with a Disjoint Set data structure. He practices that data structure every day, but still somehow manages to get it wrong. Will you show him a working example?

Xyene recalls that Kruskal's algorithm goes through all the edges one by one sorted by nondecreasing edge weight. An edge will be added to the minimum spanning tree if the two vertices it connects were not connected by any path so far.

Input Specification

The first line has N and M.

The $i + 1$-th line has u_i and v_i, the two vertices that the i-th edge connects.

Output Specification

If no minimum spanning tree exists, output `Disconnected Graph`.

Otherwise, output $N - 1$ lines: the numbers of the edges that are in the minimum spanning tree in any order.

Sample Input 1

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

Sample Output 1

```
1
2
4
```

Sample Input 2

```
3 1
1 2
```

Sample Output 2

```
Disconnected Graph
```