Problem F. Milliarium Aureum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Roman Empire is vast and, as everybody knows, all roads lead to Rome!

Following the trail of the Last Centurion, professor Melody Song found an ancient map of Roman roads. The exact position of Rome was forgotten long ago, and Melody wants to recover it from the map. There are n cities on the map, numbered from 1 to n, and m roads. Each road is marked as either *minor* or *major*.

Major roads were used to travel to Rome and formed a spanning tree, and minor roads were used as alternatives or to travel between other cities. Each road had some fixed width. It is known that major roads were very wide: if we consider two paths from Rome to any other city, an arbitrary path P and a simple path Q using the major roads, the thinnest road on path P could not be wider than the thinnest road on path Q.

The map found by Melody contains information on every road in Roman Empire, namely its type t and width w. Your task is to help her determine which cities may correspond to Rome according to the map.

Input

The first line of input contains two integers n and m ($1 \le n, m \le 10^5$).

Each of the next m lines contains four integers, t_i, u_i, v_i, and w_i, which describe a bidirectional road from city u_i to city v_i with type t_i and width w_i ($1 \le u_i, v_i \le n$, $1 \le w_i \le 10^9$). The type is $t_i = 0$ for minor roads and $t_i = 1$ for major roads.

There may be multiple roads between the same pair cities, and there may be roads connecting a city to itself. It is guaranteed though that there is exactly one simple path via major roads between each pair of cities.

Output

On the first line, output an integer k which is number of cities which may be Rome.

On the second line, output k integers **in ascending order** which are the numbers of those cities.

It is guaranteed that there is at least one such city.

Examples

standard input	standard output
4 6 1 1 2 2 1 1 3 2 1 1 4 2 0 2 3 3 0 3 4 3 0 4 2 3	1 1
3 3 0 2 3 1 1 1 2 2 1 1 3 2	3 1 2 3