B. Graph Coloring

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an undirected graph that consists of n vertices and m edges. Initially, each edge is colored either red or blue. Each turn a player picks a single vertex and switches the color of **all** edges incident to it. That is, all red edges with an endpoint in this vertex change the color to blue, while all blue edges with an endpoint in this vertex change the color to red.

Find the minimum possible number of moves required to make the colors of all edges equal.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 100\,000$) — the number of vertices and edges, respectively.

The following m lines provide the description of the edges, as the i-th of them contains two integers u_i and v_i ($1 \le u_i, v_i \le n, u_i \neq v_i$) — the indices of the vertices connected by the i-th edge, and a character c_i () providing the initial color of this edge. If c_i equals 'R', then this edge is initially colored red. Otherwise, c_i is equal to 'B' and this edge is initially colored blue. It's guaranteed that there are no self-loops and multiple edges.

Output

If there is no way to make the colors of all edges equal output -1 in the only line of the output. Otherwise first output k — the minimum number of moves required to achieve the goal, then output k integers $a_1, a_2, ..., a_k$, where a_i is equal to the index of the vertex that should be used at the i-th move.

If there are multiple optimal sequences of moves, output any of them.

Examples

input
3 3 1 2 B 3 1 R 3 2 B

output
1 2

input
6 5 1 3 R 2 3 R 3 4 B 4 5 R 4 6 R

output
2 3 4

input
4 5 1 2 R 1 3 R 2 3 B

```
3 4 B
1 4 B
```

output

```
-1
```