1579. Remove Max Number of Edges to Keep Graph Fully Traversable

Description

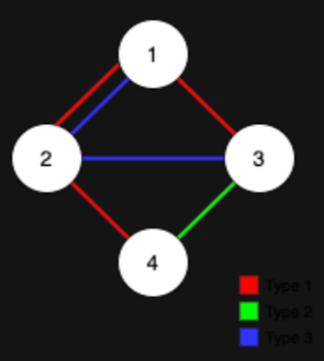
Alice and Bob have an undirected graph of `n` nodes and three types of edges:

- Type 1: Can be traversed by Alice only.
- Type 2: Can be traversed by Bob only.
- Type 3: Can be traversed by both Alice and Bob.

Given an array `edges` where `edges[i] = [type_i, u_i, v_i]` represents a bidirectional edge of type `type_i` between nodes `u_i` and `v_i`, find the maximum number of edges you can remove so that after removing the edges, the graph can still be fully traversed by both Alice and Bob. The graph is fully traversed by Alice and Bob if starting from any node, they can reach all other nodes.

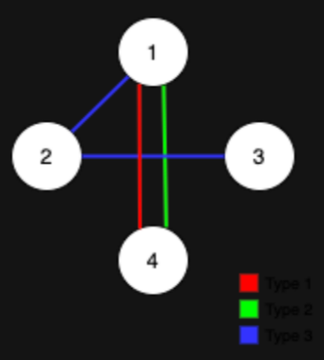
Return *the maximum number of edges you can remove, or return* `-1` *if Alice and Bob cannot fully traverse the graph.*

Example 1:



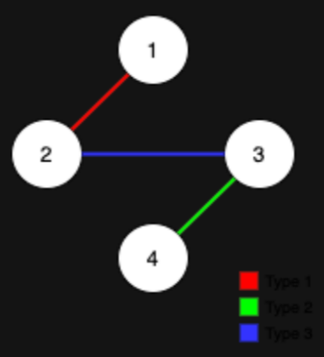

```
Input: n = 4, edges = [[3,1,2],[3,2,3],[1,1,3],[1,2,4],[1,1,2],[2,3,4]]
Output: 2
Explanation: If we remove the 2 edges [1,1,2] and [1,1,3]. The graph will still be fully traversable by Alice and Bob. Removing any additional edge will not make it so. So the maximum number of edges we can remove is 2.
```

Example 2:




```
Input: n = 4, edges = [[3,1,2],[3,2,3],[1,1,4],[2,1,4]]
Output: 0
Explanation: Notice that removing any edge will not make the graph fully traversable by Alice and Bob.
```

Example 3:




```
Input: n = 4, edges = [[3,2,3],[1,1,2],[2,3,4]]
Output: -1
Explanation: In the current graph, Alice cannot reach node 4 from the other nodes. Likewise, Bob cannot reach 1. Therefore it's impossible to make the graph fully traversable.
```

Constraints:

- `1 <= n <= 10^5`
- `1 <= edges.length <= min(10^5, 3 * n * (n - 1) / 2)`
- `edges[i].length == 3`
- `1 <= type_i <= 3`
- `1 <= u_i < v_i <= n`
- All tuples `(type_i, u_i, v_i)` are distinct.