Find the Fake

Problem Solving Technique: Solve an easier version first
It can be easier to reckon with, and might give you insight to the full version of the problem.

Always do the subtasks first. Even then, don't feel limited to needing subtasks—make up your own!

Forget about efficiency for now. Let's try to logic out any solution.

Problem Solving Technique: What does it do?
Have a strange operation? Try rephrasing it in more concrete terms:

- What is its effect on the other things in the scenario?
- What does it enable you to do, in terms of these other things?

It may also be helpful to describe what it *doesn't* do.

What does each query actually tell us?

- If the weight is $2 \cdot \text{size of subset}$, then the fake is among these coins.
- If not, then the fake is **not** among these coins.

Each query simply tells us Yes/No whether the fake is among these coins.

What does *this* knowledge tell a perfect logician like Alice?

Standard Approach: Extreme principle
Trying to grapple with a problem? Look at "extreme" cases and see how they behave.

Consider the extreme case of $q = 0$.

If there are no queries, what would Alice say? She would say "Maybe" on *all n coins*. So evidently, what each piece of information does is turn some of those Maybes into Nos.

Let's connect that with the queries.

- If the fake **is not** among these coins, then the answer for every coin in this selection turns from Maybe to No.

- If the fake **is** among these coins, then the answer for every coin *not* in this selection turns from Maybe to No.

This basically solves the problem. The only question remaining is when Alice says Yes—she does so when there is only one coin left flagged Maybe.

Implement this solution with sets and you will have solved all but the last subtask already.

Implementation

Maintain two sets `maybe` and `no`, the former initially containing all the coins (while the latter starts empty). With each query, we remove coins from `maybe` and add those into `no`.

This is very easy in Python, where `|=` and `&=` and `-=` are supported operations on sets.

For the last subtask, we cannot literally create a set with 10^9 elements—that's too much!

The key is that `maybe` and `no` provide technically redundant information, since they are complements of each other. So, given one, you can recover the other. And the size of either one is always going to be bounded by K.

- If the fake coin is never found, then all we've done is move all the coins from the queries into `no`. So, just shove all K of these coins into `no`.

- If the fake coin is ever found at some point, then we dump all coins into `no` *except* for the ones in this query; those stay in `maybe`. But the number of coins in a single query is $\leq K$, so the size of `maybe` will be small after this move; now we can just use the solution from earlier from this point on.