DIFFICULTY:

TBD

PREREQUISITES:

Frequency arrays, Basic combinatorics

PROBLEM:

You are given an array A and an integer K. Count the number of subsequences that don't contain any pair whose sum is divisible by K.

EXPLANATION:

First, notice that the condition "$A_i + A_j$ is divisible by K" can be written as $A_i + A_j \equiv 0 \pmod{K}$.

In particular, we can work with all the array elements modulo K so that they're all between 0 and $K-1$.

Now, consider what happens when we have $x + y \equiv 0 \pmod{K}$ when both x and y are less than K. There are three possibilities:

- First, we can have $x = y = 0$
- Second, if K is even we can have $x = y = K/2$
- Finally, if neither of the above hold, we must have $y = K - x$; and in particular $x \neq y$.

Let's leave the first two cases alone for now, and look at the third.
For convenience, let $x < K - x$.
Note that for each x, any good subsequence can have either some occurrences of x, or some occurrences of $K - x$: never both.
In particular, we can take as many x-s as we like, or as many $(K-x)$-s as we like, without affecting any other sums (since $K - (K - x) = x$). Essentially, we 'pair up' x with $K - x$, and then different pairs are completely independent.

So, the choices of which of the x's or $(K-x)$'s we take are completely independent across different x.
This means that any subsequence can be constructed as follows:

- Choose a subset of 1's or a subset of $K-1$'s
- Then, choose a subset of 2's or a subset of $(K-2)$'s
- Then, choose a subset of 3's or a subset of $(K-3)$'s

  $\vdots$

Thus, the total number of subsequences can be found by multiplying the number of choices for different x.

This brings us to the questions: how many choices are there for a fixed x?

▾ Answer

Let $freq(x)$ be the number of occurrences of x in the array.

Note that we can choose any subset of the x's, or any subset of the $(K-x)$'s.
The first one gives us $2^{freq(x)}$ choices, while the second gives us $2^{freq(K-x)}$ choices.

The empty set is counted in both, so we need to subtract 1 to avoid overcounting.
This brings the total to $2^{freq(x)} + 2^{freq(K-x)} - 1$.

The number of subsequences is thus just the product of $(2^{freq(x)} + 2^{freq(K-x)} - 1)$ across all x such that $x < K - x$.

The only exceptions here are $x = 0$ and (if K is even) $x = K/2$, which shouldn't be included in the above product because they behave slightly differently. Do you see how to deal with them?

▾ Answer

$x = 0$ and $x = K/2$ follow a simple rule: there can't be more than one of each in the subsequence.

So, we have $1 + freq(0)$ choices for 0 (choose none of them, or choose exactly one), and similarly $1 + freq(K/2)$ choices for $K/2$.

Multiply these quantities to the previous value to obtain the final answer.

Notice that the value for a given x requires us to compute a power of 2 modulo something.
There are several ways to do this: the simplest is to just precompute the value of $2^x \pmod{MOD}$ for every $0 \leq x \leq 5 \cdot 10^5$ before processing any test cases, after which these can be used in $\mathcal{O}(1)$.
Alternately, you can use binary exponentiation.

TIME COMPLEXITY

$\mathcal{O}(N + K)$ per test case.

```
mod = 10**9 + 7
for _ in range(int(input())):
    n, k = map(int, input().split())
    a = list(map(int, input().split()))
    freq = [0]*k
    for x in a:
        freq[x%k] += 1
    ans = 1
    for i in range(k):
        if i == 0 or 2*i == k:
            ans *= 1 + freq[i]
        else:
            if i > k-i: break
            ans *= pow(2, freq[i], mod) + pow(2, freq[k-i], mod) - 1
        ans %= mod
    print(ans%mod)
```