Mini-Comp 0 Q6 Solution

Alex Tung

February 14, 2017

Simplifying Assumptions

We make several assumptions before explaining the solution.

- There are N positions and each time we move clockwise by K steps. Let $g = gcd(N, K)$. The result is unchanged if we replace (N, K) by $(N/g, K/g)$. From now on, assume $gcd(N, K) = 1$.
- The case $N = 1$ is trivial. From now on, assume $N > 1$.
- Now, we 0-index the presents and choose $A \in [0, N)$ such that $AK \equiv 1 \pmod{N}$. We can remodel the task as one which asks whether the permutation $(0,\ A,\ 2A \bmod N,\ 3A \bmod N,\ ...,\ (N-1)A \bmod N)$ is odd or even. (An **odd permutation** is one that has an odd number of inversions. An **even permutation** is one that has an even number of permutations.)

For example, the first sample test has $N = 10$, $K = 4$.

- $g = gcd(10, 4) = 2$. Now, take $N = 5$ and $K = 2$.
- $A = 3$. We want to know: is the permutation $(0, 3, 1, 4, 2)$ odd or even?

In fact, for the second simplifying step, we need not calculate A; the answer will be the same if we consider $(0,\ K,\ 2K \bmod N,\ 3K \bmod N,\ ...,\ (N-1)K \bmod N)$. Hence, for the example above, it suffices to consider the permutation $(0, 2, 4, 1, 3)$. This fact may seem puzzling at the moment; that it is true follows (though nontrivially) from the solution.

Now, we have a transformed task at hand:

> Given $1 \leq K < N$ with $gcd(N, K) = 1$, is the permutation $(0,\ K,\ 2K \bmod N,\ 3K \bmod N,\ ...,\ (N-1)K \bmod N)$ odd or even?

Propositions will be stated without proof. Most of them are simple facts and it will be cumbersome to put down all the proofs here.

Special Case: N is an odd prime

We focus on the special case where N is an odd prime. First, we use the following proposition:

Proposition 1. An odd permutation becomes even after swapping two (not necessarily adjacent) elements, and vice versa.

Thus, we only need to consider the number of (not necessarily adjacent) swappings needed to sort the elements of the permutation.

An Example

Consider the following example ($N = 11$, $K = 8$). The permutation is:

$$0\ 8\ 5\ 2\ 10\ 7\ 4\ 1\ 9\ 6\ 3$$

Discarding 0 (this will *not* affect the answer), we see that the elements are nicely split into two halves:

$$8\ 5\ 2\ 10\ 7\ 4\ 1\ 9\ 6\ 3$$

Do you see the nice symmetry?

$$\begin{aligned} 8 + 3 &= 11 \\ 5 + 6 &= 11 \\ 2 + 9 &= 11 \\ 10 + 1 &= 11 \\ 7 + 4 &= 11 \end{aligned}$$

Therefore, if we swap the blue elements that are "too large" (i.e. larger than $N/2$) with the corresponding red elements, we obtain the following:

$$\underline{3}\ 5\ 2\ \underline{1}\ \underline{4}\ \underline{7}\ \underline{10}\ 9\ 6\ \underline{8}$$

Three (3) swaps performed in total. Then, to sort the elements, we only need to perform swappings independently within the blue group and the red group. But the two groups are *symmetric*, so, for this stage, the total number of swappings must be even.

Counting Tricks

Making use of this observation, we see that the parity of the number of swappings only depends on the number of "large" elements in the blue group. In other words, we need to count the number of solutions to the following simultaneous equations:

$$\begin{cases} 1 \le r \le \frac{N}{2} \\ rK \bmod N > \frac{N}{2} \end{cases}$$

This is not easy to count, but there is a trick when N is an odd prime and when we are concerned only with the *parity* of the number of solutions.

Let the sought number of solutions be s. We consider the product P_1 of the blue elements.

$$\begin{aligned} P_1 &= K \times (2K) \times ... \times (\tfrac{N-1}{2}K) \\ &= (\tfrac{N-1}{2})! \times K^{\frac{N-1}{2}} \end{aligned}$$

On the other hand, consider the product P_2 of numbers from 1 to $\frac{N-1}{2}$.

$$P_2 = (\tfrac{N-1}{2})!$$

Observe that every time we swap a blue element with the corresponding red element, the value of P_1 (the product of blue elements) modulo N multiplies by (-1). (This is because X is replaced by N - X.) After swapping for s times, the product changes from P_1 to P_2. So we have:

$$\begin{aligned} (-1)^s P_1 &\equiv P_2 \pmod{N} \\ (-1)^s \times (\tfrac{N-1}{2})! \times K^{\frac{N-1}{2}} &\equiv (\tfrac{N-1}{2})! \pmod{N} \\ (-1)^s &\equiv K^{\frac{N-1}{2}} \pmod{N} \end{aligned}$$

To get from the second line to the last line, cancel $(\frac{N-1}{2})!$ from both sides, then multiply $(-1)^s$ on both sides. Cancellation of the term $(\frac{N-1}{2})!$ is valid because N is prime; without this condition, the last line does not follow.

Hence, we can find out whether s is odd or even by calculating $K^{\frac{N-1}{2}}$ mod N. If the value is 1, s is even. Otherwise, s is odd. This can be computed in O(log N) per query, using fast exponentiation.

Case 1: N is odd

Next, we generalise our work to arbitrary odd N. The case where N is even will be dealt with in the last section.

Let $N = pM$, where p is an odd prime and $M > 1$ is odd. Write the N numbers in a rectangular array of size $p \times M$ (we omit mod N from the table so it would not look clumsy):

0	K	$2K$	...	$(M-1)K$
MK	$(M+1)K$	$(M+2)K$	...	$(2M-1)K$
...	...	...	...	...
$((p-1)M)K$	$((p-1)M+1)K$	$((p-1)M+2)K$	...	$(N-1)K$

One can observe that numbers on the same column have the same value modulo M. This provides inspiration for the following swapping algorithm:

- Step 1: sort the elements in each column
- Step 2: sort the columns "as a whole"

We demonstrate this two-step algorithm using the example $N = 15$, $p = 3$, $M = 5$, $K = 7$. First, we draw the 3×5 table:

0	7	14	6	13
5	12	4	11	3
10	2	9	1	8

Next, we sort the elements in each column.

0	2	4	1	3
5	7	9	6	8
10	12	14	11	13

Note that we perform no (0) swaps for column 1 and two (2) swaps for columns 2 through 5. It is not a coincidence that these number of swappings have the same parity; we state it as a proposition here.

Proposition 2. Let u_i be the number of swappings required for column i. Then $u_i \equiv u_j \pmod 2$.

Corollary. Let s be the number of swappings required for step 1. Then $(-1)^s \equiv K^{\frac{p-1}{2}} \pmod p$

From **Proposition 2**, it suffices to consider column 1. Dividing each element by M, we obtain, in order, the elements 0, K mod p, ..., $(p-1)K$ mod p. By previous analysis on the case where N is an odd prime, the corollary follows.

Finally, we move on to step 2 and sort the columns.

0	2	4	1	3
5	7	9	6	8
10	12	14	11	13

0	1	4	2	3
5	6	9	7	8
10	11	14	12	13

0	1	2	4	3
5	6	7	9	8
10	11	12	14	13

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

Each time, we swap p pairs of elements. As p is odd, we may as well consider just the first row, which consists of 0, K mod M, $2K$ mod M, ..., $(M-1)K$ mod M. We have successfully reduced the problem from (N, K) to (M, K mod M).

Hence, we arrive at the following algorithm for the case where N is odd:

```
solve(N, K):
  // P = (0, K, 2K mod N, ..., (N-1)K mod N)
  // returns 0 if P is even, 1 if P is odd
  if N = 1
       return 0
  p := odd prime dividing N
  res := f(K, p),
       where f(a, b) = a^((b-1)/2) mod b (take 1 or -1)
  if res = 1
       return solve(N/p, K mod N/p)
  else
       return 1 - solve(N/p, K mod N/p)
```

By induction, we can rewrite the above algorithm as follows:

```
solve2(N, K):
  // P = (0, K, 2K mod N, ..., (N-1)K mod N)
  // returns 0 if P is even, 1 if P is odd
  factorise N := p1p2p3...pa, pi prime
  res := f(K,p1) f(K,p2) ...  f(K,pa),
       where f(a, b) = a^((b-1)/2) mod b (take 1 or -1)
  if res = 1
       return 0
  else
       return 1
```

This algorithm depends heavily on the efficiency of prime factorisation. Also, we need fast exponentiation for calculating `f(a, b)`. Therefore, the time complexity (per query) is FACT(N) + O(log N), where FACT() is the time complexity of the factorisation algorithm used. For example, if FACT(N) = O($\sqrt{N}$), it will be fast enough to solve all subtasks (at least for the cases where N is odd).

Case 2: N is even

Write $N = 2M$. There are two sub-cases to consider: M is odd and M is even.

Case 2.1: M is odd

Write the $2M$ numbers in a $M \times 2$ table, as follows (we omit mod N):

0	K
$2K$	$3K$
$4K$	$5K$
...	...
$(2M-2)K$	$(2M-1)K$

The numbers on the first column are all even and those on the second column are all odd, so we do not need to swap elements on the first column with those on the second column. Again, as M is odd, the number of swaps required to sort the first column is congruent to that to sort the second column, modulo 2 (recall **Proposition 2**).

Therefore, when $N \equiv 2 \pmod 4$, the permutation is always even.

Case 2.2: M is even

Similar to the sub-case above, we write the numbers in a $M \times 2$ table:

0	K
$2K$	$3K$
$4K$	$5K$
...	...
$(N-2)K$	$(N-1)K$

Our aim to to rearrange the second column so that each element on the second column is exactly (1 + its left element), after which the table will be symmetric and the additional number of swaps will be even.

For example, if $N = 12$ and $K = 5$, we initially have the table on the left and we wish to achieve the table on the right.

0	5
10	3
8	1
6	11
4	9
2	7

0	1
10	11
8	9
6	7
4	5
2	3

Notice that it can be done by cyclic shifting the whole column by a certain number of steps. How many steps, you ask? It will become obvious after we colour some cells.

0	5
10	3
8	1
6	11
4	9
2	7

Each shifting, we swap $(M-1)$ pairs of elements, and the green and red cells interchange positions. In the final configuration, 1, which is green, is on the top-right corner. Notice that originally K is the number on the top-right corner. Therefore, if K is green, the number of swaps is odd; conversely, if K is red, the number of swaps is even.

How exactly are the colours determined? It is deceptively simple:

- we colour a number X green if $X \equiv 1 \pmod 4$
- we colour a number X red if $X \equiv 3 \pmod 4$

Therefore, we have the following: if $K \equiv 1 \pmod 4$, the permutation is even. Otherwise, the permutation is odd.

Combining the two cases, the problem is solved.

Footnote

Now we have solved the problem, but two questions remain:

1. Can we solve it more efficiently?
2. How did the author come up with such a (insert adjective) problem?

The answer to the first question is an emphatic **YES**. In fact, the two questions are somewhat related, so I will answer them both at once.

The author came up with this problem while attending a number theory lecture, which back then was exploring a great result called "quadratic reciprocity" (QR). One proof of QR uses Gauss's lemma, and after a bit googling you will see its connection with this problem, for the case where N is an odd prime. (To be more precise, Gauss's lemma relates the quantity $K^{\frac{N-1}{2}} \bmod N$ to the number of elements in the first part with value larger than $\frac{N-1}{2}$.)

In fact, when N is an odd prime, $K^{\frac{N-1}{2}} \pmod N$ is exactly the Legendre symbol $(\frac{K}{N})$, and Jacobi symbol, a generalisation of the Legendre symbol, corresponds exactly to our sought answer when N is odd (but not necessarily prime)! Since the calculation of the Jacobi symbol $(\frac{a}{b})$ can be done in $O(\log b)$ time (it is, in some sense, similar to the Euclidean algorithm), and that the case when N is even can be solved in $O(1)$ time, our problem can thus be solved in $O(\log N)$ time per query.

Don't worry if you find this problem intimidating. This problem appears in Mini-Comp 0 for the sole reason that it is not suitable for other contests. In particular, we would not test "advanced" number theory/pure mathematics/difficult pattern-finding in the Team Formation Test.