Coding Competitions Farewell Rounds - Round B

Analysis: Intruder Outsmarting

Reachable Numbers on a Wheel

Let us first define the concept of "reachable numbers" on a wheel. These are the set of numbers that can be displayed on a wheel by rotating it in increments of $\mathbf{D}$, starting from the initial number on the wheel as given in the input.

Example: Let us take $\mathbf{N} = 6$ and $\mathbf{D} = 4$, and we start from the number 1 on the wheel. Then the reachable numbers are:

$$1 + 4 = \mathbf{5}$$

$$5 + 4 = 9 \Rightarrow 9 - 6 = \mathbf{3}$$

$$3 + 4 = 7 \Rightarrow 7 - 6 = \mathbf{1}$$

Which brings us back to the beginning. Thus, our set of reachable numbers is $\{1, 3, 5\}$. Note here that we subtract 6 from numbers > 6 because of the circular sequence of numbers on the wheel.

Test Set 1

We can try all combinations of reachable values on the wheels, check for each combination if it forms a palindrome and if it does, calculate the minimum number of operations required to form that combination.

Each wheel can have a maximum of $\mathbf{N}$ reachable values. For $\mathbf{W}$ wheels, the number of possible combinations that can be shown on the security key will be of the order of $\mathbf{N}^{\mathbf{W}}$. A combination can be generated by generating the possible values for each wheel. For a given index i, we can start with $\mathbf{X}_i$, then incrementing the values in steps of $\mathbf{D}$. If the value overflows beyond $\mathbf{N}$, we circle back around to 1 by subtracting $\mathbf{N}$, which then brings us to a value lying in $[1 \cdots \mathbf{N}]$. When we get to a value that we have previously encountered, we then have the complete set of values reachable for that index. In this way, we can generate all combinations by iterating through all possible reachable values for each index.

For each index, the set of reachable positions can be generated in $O(\mathbf{N})$, and for $\mathbf{W}$ wheels, we can do this in $O(\mathbf{N} \times \mathbf{W})$. Then, we can generate all combinations in $O(\mathbf{N}^{\mathbf{W}})$.

For a particular combination A, we can check if it is a palindrome by simply iterating over the array and verifying that $A_i = A_{\mathbf{W}-i+1}$ for all $i = 1$ to $\frac{W}{2}$ in $O(\mathbf{W})$.

If A is verified to be a palindrome, we can find the minimum cost of achieving the combination by summing over the minimum cost of converting each $\mathbf{X}_i$ to A_i. For each i, we can perform a process similar to the combination generation process by starting with $\mathbf{X}_i$ and simulating successive forward operations (that is keep adding $\mathbf{D}$) or by simulating successive backward operations (that is keep subtracting $\mathbf{D}$) till we reach A_i. We take the minimum of the number of forward or backward operations as the minimum number of operations it will take to achieve the conversion. For each index i, we can hence find out the cost in $O(\mathbf{N})$. Therefore, the cost for all $\mathbf{W}$ wheels can be calculated in $O(\mathbf{N} \times \mathbf{W})$ time.

Thus, the overall time complexity: $O(\mathbf{N}^{\mathbf{W}} \times \mathbf{W} \times \mathbf{N})$

Test Set 2

The above brute force of generating all possible combinations approach is not feasible to pass test set 2 constraints. Instead for each pair $(\mathbf{X}_i, \mathbf{X}_{\mathbf{W}-i+1})$ in the array $\mathbf{X}$ (for all $i = 1$ to $\frac{\mathbf{W}}{2}$) that need to be equalized to make the array a palindrome, we calculate the minimum number of operations that can do so.

Before we move forward, we transpose the set of possible values for each wheel from $[1 \cdots \mathbf{N}]$ to $[0 \cdots \mathbf{N}-1]$. This will allow us to visualize the circular nature of operations as modulo N. We do this by simply subtracting 1 from all elements in $\mathbf{X}$.

With that, let us try to find the minimum operations to change $\mathbf{X}_i$ to $\mathbf{X}_{\mathbf{W}-i+1}$. Say we rotate the i-th wheel in the positive direction and try to make it equal to $\mathbf{X}_j$. Let x be the number of operations it takes to do so. With operations being circular in nature, we get:

$$\mathbf{X}_i + \mathbf{D}x \equiv \mathbf{X}_j \pmod{\mathbf{N}}$$

$$\Rightarrow \mathbf{X}_i + \mathbf{D}x + \mathbf{N}y = \mathbf{X}_j$$

$$\Rightarrow \mathbf{D}x + \mathbf{N}y = \mathbf{X}_j - \mathbf{X}_i$$

The above equation is Linear Diophantine Equation. The greatest common divisor (GCD) of two numbers i and j can be written as $GCD_{i,j}$.

From the Extended Euclidean Algorithm, an LDE (Linear Diophantine Equation) $ax + by = c$ has a solution only if c divides $GCD_{a,b}$. Thus, the above equation $\mathbf{D}x + \mathbf{N}y = \mathbf{X}_j - \mathbf{X}_i$ will have a solution for x and y only if $\mathbf{X}_j - \mathbf{X}_i$ is divisible by $GCD_{\mathbf{D},\mathbf{N}}$. Thus, for any pair $(\mathbf{X}_i, \mathbf{X}_{\mathbf{W}-i+1})$, if $\mathbf{X}_{\mathbf{W}-i+1} - \mathbf{X}_i$ is not divisible by $GCD_{\mathbf{D},\mathbf{N}}$ the answer is `IMPOSSIBLE`.

Otherwise, we can use the Extended Euclidean Algorithm to solve for one possible solution for x and y in $O(\log min(\mathbf{D}, \mathbf{N}))$. We need the minimal operations x_{min}, which is $(\mathbf{N} + x) \pmod{\mathbf{N}}$. This is because a general solution for (x, y) will be of the form $(x + k\frac{\mathbf{N}}{GCD_{\mathbf{D},\mathbf{N}}}, y - k\frac{\mathbf{D}}{GCD_{\mathbf{D},\mathbf{N}}})$, where k is an arbitrary integer. From the above general form, we can see that the minimum number of operations x_{min} would be in the range $[0, \frac{\mathbf{N}}{GCD_{\mathbf{D},\mathbf{N}}} - 1]$ (if it is larger than that, we can simply subtract $\frac{\mathbf{N}}{GCD_{\mathbf{D},\mathbf{N}}}$ from it to get a smaller value).

Hence, $x_{min} = x \pmod{\frac{\mathbf{N}}{GCD_{\mathbf{D},\mathbf{N}}}}$. Since x returned from the Extended Euclidean Algorithm can be negative, we add $\frac{\mathbf{N}}{GCD_{\mathbf{D},\mathbf{N}}}$ and take modulo $\frac{\mathbf{N}}{GCD_{\mathbf{D},\mathbf{N}}}$ again to get $x_{min} = (\frac{\mathbf{N}}{GCD_{\mathbf{D},\mathbf{N}}} + x) \pmod{\frac{\mathbf{N}}{GCD_{\mathbf{D},\mathbf{N}}}}$. Let us call this "forward direction x_{min}" as $x_{forward}$.

Similarly, we can find the minimum number of operations if we rotate i-th wheel in the backward direction. We get a similar equation in this case as well:

$$\mathbf{X}_i - \mathbf{D}x \equiv \mathbf{X}_j \pmod{\mathbf{N}}$$

$$\Rightarrow \mathbf{X}_i - \mathbf{D}x + \mathbf{N}y = \mathbf{X}_j$$

$$\Rightarrow \mathbf{D}x - \mathbf{N}y = \mathbf{X}_i - \mathbf{X}_j$$

We solve for x_{min} for this equation in a similar fashion using the Extended Euclidean Algorithm as the forward case. Let us call the x_{min} we get this time as $x_{backward}$.

Thus, the overall minimum operations it takes to change $\mathbf{X}_i$ to $\mathbf{X}_j$ is $min(x_{forward}, x_{backward})$. We add together the minimum costs for all such pairs (that is for all $i = 1$ to $\mathbf{W}/2$) to get the minimum cost to make the array a palindrome.

The overall time complexity: $O(\mathbf{W} \times \log(\mathbf{D}))$.