Code Jam 2019 - Round 1B

Analysis: Draupnir

In this problem, we need to determine the number of each type of X-day ring by querying the total number of rings in existence at the end of a certain day.

Let R_1 be the number of 1-day rings, R_2 be the number of 2-day rings, ... etc. at the end of day 0. The number of X-day rings doubles at the end of every day that is a multiple of X. Thus, the total number of rings on day i is

$$R_1*2^{i} + R_2*2^{floor(i/2)} + R_3*2^{floor(i/3)} + R_4*2^{floor(i/4)} + R_5*2^{floor(i/5)} + R_6*2^{floor(i/6)}.$$

Test set 1

In the first test set, we are allowed six queries to determine the six unknown variables. Note that if we query any day number larger than 63, then the number of 1-day rings on that day will be a multiple of 2^{63} (equivalently, it is 0 modulo 2^{63}). Similarly, if we query any day number larger than X*63, then the number of X-day rings will be 0 modulo 2^{63} on that day.

Thus, on day 315 (=5*63), the total number of rings, modulo 2^{63}, is R_6*2^{52} (since the number of 1-day, 2-day, ..., 5-day rings are all 0 modulo 2^{63}). Since $R_6 \le 100$, we know that R_6*2^{52} does not exceed 2^{63}, so we can directly determine R_6. Then, on day 252 (=4*63), the total number of rings, modulo 2^{63}, is $R_6*2^{42} + R_5*2^{50}$. Since we already know R_6 and we know this sum cannot be more than 2^{63}, we can solve for R_5. We may continue this process by querying days 189 (=3*63), 126 (=2*63), 63 (=1*63) and 1 to determine R_4, R_3, R_2 and R_1, respectively.

Alternative

Note that since we are querying six days and attempting to solve for six variables, we may choose to query (for example) days 1,2, ... , 6. This gives us a system of equations with six equations and six variables. Since these equations are linearly independent, we can solve this system, via Gaussian elimination for example, to get the solution.

Test set 2

The second test set requires another insight. We are only given two queries, so we must be able to solve for multiple R_i at once.

Let's think about what information we get by querying day 189 (=3*63). This gives $R_6*2^{31} + R_5*2^{37} + R_4*2^{47}$, modulo 2^{63}. However, it is possible that there is some overlap between R_6*2^{31} and R_5*2^{37}. For example, all other things being equal, we would not be able to differentiate between the case where R_6=64 and R_5=0 and the case where R_6=0 and R_5=1. Both of these would have 2^{37} total rings.

We must use the fact that $R_i \le 100$. In order for the number of i-day rings to not interfere with the number of (i-1)-day rings on day d, we need $2^{floor(d/(i-1))} > 100 * 2^{floor(d/i)}$ (note that this is equivalent to floor(d/(i-1)) ≥ floor(d/i)+7 since $2^7 > 100$). If we ignore the modulo 2^{63} restriction, then we could solve the question in a single query of (for example) 1000. This would give us

$$R_1*2^{1000} + R_2*2^{500} + R_3*2^{333} + R_4*2^{250} + R_5*2^{200} + R_6*2^{166}.$$

Since $R_i \le 100$, we coud determine R_6 by taking this value modulo 2^{200} to get R_6*2^{166}, then by dividing by 2^{166}. We then could iteratively determine R_5, R_4, ..., R_1. However, this idea does not work since the problem is modulo 2^{63}. There is no single query that can give us all of the information we need, modulo 2^{63}.

We will use one query to determine the values of R_4, R_5, and R_6, followed by a second query to determine the values of R_1, R_2, and R_3. We saw above that a query of 189 (=3*63) will not work. However, a query of (for example) 200 will work:

$$R_4*2^{50} + R_5*2^{40} + R_6*2^{33},$$

and then solve for each in the same way as for the 1000 case. We can then make a second query of (for example) 56:

$$R_1*2^{56} + R_2*2^{28} + R_3*2^{18} + R_4*2^{14} + R_5*2^{11} + R_6*2^{9}.$$

We know the value of R_4, R_5, and R_6 from the first step, so we may substitute those in and then solve for R_1, R_2, and R_3 one-by-one.

Note that the choice of 200 and 56 above were not the only possible options. In this problem, we could either guess-and-check for these values offline or (preferably) write a loop in the program to find two values that satisfy the criteria. We also need to ensure that the term with the largest exponent is not too large. For example, we cannot use a query of 250 in the first step since this gives $R_4*2^{62} + R_5*2^{50} + R_6*2^{41}$, because the first term (R_4*2^{62}) may be at least 2^{63}.