Kick Start 2019 - Round G

Analysis: The Equation

Test set 1 (Visible)

For the first test set, notice that the maximum value of **k** is 127. This is because each $\mathbf{A_i}$ is at most 100, so the leading digit of $\mathbf{A_i}$ is at most $2^6 = 64$. If **k** ≥ 128, then the leading digit of **k** is at least $2^7 = 128$, meaning that ($\mathbf{A_i}$ xor k) ≥ 128 > **M**.

Hence, we can compute the answer by checking each value of **k** less than 128 and finding the largest one which produces a sum less than **M**.

Test set 2 (Hidden)

For the second test set, the reasoning above tells us that **k** < 2^{50}, which is too big for us to check every value.

Instead, notice that each bit of **k** only affects a single bit of each $\mathbf{A_i}$. We can use this property to compute each bit of **k** separately.

For each $1 \le i \le 50$, define *ones(i)* to be the number of rules $\mathbf{A_i}$ with the *i*-th bit (numbered starting from the least significant bit) equal to 1. Likewise, define *zeroes(i)* to be the number of rules with the *i*-th bit equal to 0. Then we can re-write the sum:

$\Sigma_{1 \le j \le \mathbf{N}} \mathbf{A_j}$ xor **k**

as:

$\Sigma_{i\,:\,i\text{-th bit of } \mathbf{k} \text{ is } 1}\, 2^i \times zeroes(i) + \Sigma_{i\,:\,i\text{-th bit of } \mathbf{k} \text{ is } 0}\, 2^i \times ones(i)$

Note that we can minimize this sum by choosing the *i*-th bit of **k** to be 1 if *ones(i)* ≥ *zeroes(i)*, or 0 otherwise. Define *f(j)* to be the minimum value of the above sum over all bits *i* ≤ *j*. We can use *f(j)* to determine if a feasible value of **k** exists for the lowest *j* bits, which lets us solve this problem greedily. The greedy solution is as follows: starting from the most significant bit *i*, check if we can set it to be one (by adding cost of setting this bit to one and *f(i-1)*). If this value is less than or equal to **m**, there exists a feasible **k** with the *i*-th bit set to one. Since we want to set to maximize **k**, it is optimal for us to set this bit to 1. Otherwise, if the sum is larger than **m**, set the bit to zero. Then iterate by decreasing **m** by the cost at the current bit and checking the next most significant bit (*i*-1). In this way, we are able to find the largest feasible **k**. If *f(i)* is precomputed, the runtime of this algorithm is O(**N**log(max($\mathbf{A_i}$))).

Note that since $\mathbf{A_i} \le 10^{15}$ and **N** ≤ 1000, the maximum sum is a little more than 10^{18}, so using 64-bit integers is sufficient for this problem.