Koxia and Sequence (CF 1770F)

Timothy Mou

January 6, 2023

I encountered this Codeforces problem with some really interesting and remarkable ideas, so I decided to write a short note about it. This solution is partly based off an explanation by CF user aryanc403.

Contents

§1 Problem Statement

We are given integers $1 \leq n < 2^{40}$, $0 \leq x < 2^{60}$, and $0 \leq y < 2^{20}$. An array A of n non-negative integers is **good** if $a_1 + a_2 + \cdots + a_n = x$ and $a_1|a_2|\ldots|a_n = y$. We are asked to compute the bitwise xor of all elements of all good arrays.

Formally, let the **score** of an array be $a_1 \oplus a_2 \oplus \cdots \oplus a_n$. We want to compute the **total score** (bitwise xor) of the scores of all good arrays.

§2 High-Level Overview

This can seem like an intimidating problem. One way to get started is to write a brute force and see if there are any patterns in small cases. Alternatively, the fact that we are dealing with xor instead of addition or multiplication indicates that we can utilize some very nice symmetric properties about xor. Here's a simple observation we can make that illustrates this point:

Lemma 2.1

If n is even, the total score is 0.

Proof. First note that in a palindrome of even length n, each element appears an even number of times, so its score is 0, and it does not contribute to the total. Thus let's only consider non-palindromic good arrays. For any good array A, consider its reverse A', which is also a good array. We can compute the total score by pairing up arrays and their reverse (e.g., $(\text{score}(A) \oplus \text{score}(A')) \oplus (\text{score}(B) \oplus \text{score}(B')) \oplus \dots$). For each pair A and A', the two arrays contain the same elements, so $\text{score}(A) \oplus \text{score}(A') = 0$. Thus the total score is also 0. $\square$

Remark 2.2. This simple proof illustrates an important concept that we will use later: if we have a set S and an involution $f : S \to S$ (i.e., $f(f(s)) = s$ for all $s \in s$), we can compute the parity of the size of S by counting the number of elements fixed by f. In other words, we can pair up elements in a set that cancel each other out.

§2.1 Definitions

Let's collect some terminology we will use:

- The **binary representation** of a nonnegative integer x is the unique set of powers of 2 that sum to x.
- If a and b are integers, we say a is a **submask** of b if the powers of 2 in the binary representation of a are also in the binary representation of b.

§2.2 Overview

Our high-level plan will be as follows. Since any element in a good array will be a submask of y, the total score will also be a submask of y. Therefore let's compute the total answer by counting the total contribution of each power of 2 in the binary representation of y. If 2^i appears an odd number of times among all elements of all good arrays, then we add 2^i to the answer; otherwise, we add 0.

So, for each power of 2, we want to count the parity of the number of times it appears among all good arrays. We will need some way of representing the conditions that the sum of a good array is x, and that the bitwise or of a good array is y. However, it turns out that this bitwise or condition is a bit unwieldy, and it is easier to work with the relaxed constraint that the bitwise or is simply a submask of y. We can recover the original constraint by using the inclusion-exclusion principle and considering all submasks of y.

This sounds good so far, but we still haven't dealt with the meat of the problem–counting the number of arrays whose sum is x with some particular constraints. Moreover, by using inclusion-exclusion and considering each power of 2 independently, we have forced ourselves to iterate over $O(y \log y)$ (bit, mask) pairs, so ideally, we would be able to compute this number in constant time or at least $O(\log y)$. Since we are only concerned with the *parity* of the number of arrays, it turns out we can compute this in $O(1)$ with the help of some number theory (Lucas' Theorem) and some clever pairing-up arguments.

§3 Details

§3.1 Matrix Fun

We assume n is odd. Let $B = \begin{bmatrix} b_1 & b_2 & \dots & b_k \end{bmatrix}$ be the powers of 2 in the binary representation of y. For instance, if $y = 13$, then $B = \begin{bmatrix} 2^0 & 2^1 & 2^3 \end{bmatrix}$. If A is a good array, since $a_1|a_2|\dots|a_n = y$, a_i is a submask of y. We can imagine "redistributing" the bits in A to produce a new good array. More precisely, given a good array A, let c_i be the number of elements that have b_i in their binary representation. Since $a_1|a_2|\dots|a_n = y$, $c_i \geq 1$, and since there are n elements in an array, $c_i \leq n$. Let $C = \begin{bmatrix} c_1 & c_2 & \dots & c_k \end{bmatrix}^\top$. Since $a_1 + a_2 + \dots + a_n = x$, we have $BC = c_1 b_1 + c_2 b_2 + \dots + c_k b_k = x$, and there are

$$F(C) = \binom{n}{c_1}\binom{n}{c_2}\cdots\binom{n}{c_k}$$

ways to redistribute bits to obtain new good arrays with the same C vector. Each good array corresponds to some C, and we can compute the total score by computing the scores of each of the $F(C)$ good arrays corresponding to C. If $F(C)$ is even, then since each b_i appears an even number of times, the score of all $F(C)$ arrays is 0. If $F(C)$ is odd, the score of all $F(C)$ arrays is the xor of all b_i, where c_i is odd.

By Lucas' Theorem, $\binom{n}{c_i}$ is odd if and only if c_i is a submask of n. In order for $F(C)$ to be odd, all $\binom{n}{c_i}$ must be odd, so c_i is a submask of n for all i. Therefore, let's rewrite our equation $BC = x$ into a form where we are only considering solutions where c_i is a submask of n. Let $D = \begin{bmatrix} d_1 & d_2 & \dots & d_l \end{bmatrix}^\top$ be the powers of 2 in the binary representation of n (since n is odd, let's assume $d_1 = 1$ for notational convenience). Then since c_i is a submask of n, we can represent c_i with the row vector $E_i = \begin{bmatrix} e_{i,1} & e_{i,2} & \dots & e_{i,l} \end{bmatrix}$, where $e_{i,j} \in \{0, 1\}$ and $c_i = E_i D$. Let E be the matrix of all E_i, so

$$E = \begin{bmatrix} e_{1,1} & e_{1,2} & \dots & e_{1,l} \\ e_{2,1} & e_{2,2} & \dots & e_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ e_{k,1} & e_{k,2} & \dots & e_{k,l} \end{bmatrix} \text{ and } C = ED.$$

Then

$$x = BC = BED = \begin{bmatrix} b_1 & b_2 \dots & b_k \end{bmatrix} \begin{bmatrix} e_{1,1} & e_{1,2} & \dots & e_{1,l} \\ e_{2,1} & e_{2,2} & \dots & e_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ e_{k,1} & e_{k,2} & \dots & e_{k,l} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_l \end{bmatrix}.$$

Each possible binary matrix E corresponds to a choice of C where each c_i is a submask of n. Let's call matrix E **valid** if $BED = x$. For each valid matrix where each row in E has a nonzero entry (i.e., $c_i \geq 1$), we add b_i to the xor-sum for each i such that $e_{i,1} = 1$ (i.e., c_i is odd).

§3.2 Inclusion-Exclusion

Let's compute the total score by computing the contribution of each b_i. We will add b_i to our total score exactly when there is an odd number of valid matrices E with $e_{i,1} = 1$. However, recall that E needs to satisfy the bitwise or condition as well: for each c_i, we must have $1 \leq c_i$, which means that for each i, at least one $e_{i,j}$ in the i'th row must be 1. The problem is that this is a difficult constraint to work with; we've already used this binary matrix representation to encode the constraint that each c_i must be a submask of n, but it's harder to force each row to have a nonzero element.

Let's work around this problem by relaxing this constraint, and then recovering the original answer using the inclusion-exclusion principle over all submasks of y. More precisely, suppose y' is a submask of y. Let $g(i, y')$ be the parity of the number of matrices E where $e_{i,1} = 1$ and the k'th row in E is allowed to have nonzero entries only when d_k is in the binary representation of y'. Importantly, we don't require here that $c_i \geq 1$. Then $g(i, y') = \bigoplus_{y' \subseteq y} g(i, y')$. This is the same idea as regular inclusion-exclusion, except we don't have to worry about signs since we only care about parity.

So now all we have to do is solve this relaxed problem for each i and each submask y'. The issue is that there are $O(y)$ submasks and $O(\log y)$ powers of 2 to handle, so we're already up to $O(y \log y)$ complexity, and we still haven't dealt with how to count the number of solutions to this very large subset sum problem. However, keep in mind that we are only interested in the parity of the number of solutions, and this is again where symmetry comes into play. Consider some valid matrix E. For each nonzero entry $e_{i,j}$, we can think of $e_{i,j}$ as "selecting" a pair of terms $b_i d_j$ to sum to x. Instead of thinking of our solution space as all possible binary matrices E, let's think of it as multisets of terms of the form $b_i d_j$, where we select some subset to sum to x. Then it turns out we have a simple condition to determine the parity of the number of ways to sum to x:

Lemma 3.1

Let T be a multiset of powers of 2 whose sum is S. The number of ways to select a subset of T to sum to x is odd if and only if x is a submask of S.

Proof. Suppose we have two equal numbers in our multiset; let's call them a and b. If we have a subset that includes a but not b, we can pair it up with a nearly identical subset that includes b but not a, and these two subsets have the same sum. Thus if we have two equal numbers in our multiset, we only have to count the number of solutions where we either include both of them or ignore both of them, since by this pairing argument, the number of solutions where we include just one of them is even. However, if we are always either including or ignoring two terms equal to some power 2^k, we might as well replace them with the single term 2^{k+1}. More precisely, if our multiset is T, there is a bijection between

$$\begin{aligned}&[\text{subsets of } T \text{ that either both include or ignore } a \text{ and } b] \\ &\qquad \leftrightarrow [\text{subsets of } T \setminus \{2^k, 2^k\} \cup \{2^{k+1}\}].\end{aligned}$$

Note that these two multisets maintain the same sum.

Moreover, we can iteratively repeat this process until we have no more duplicate numbers in our multiset. At the end of this process, we have some set T' of unique powers of 2 whose sum is still S, and we can form a sum of x exactly when x is a submask of the S. □

Let's apply this observation to solve our subproblem. Given i and a submask y', our multiset T is the set of terms $\{b_i d_j, 1 \leq i \leq k, 1 \leq j \leq l, b_i \in y'\}$ (which are all powers of 2 since b_i and d_j are powers of 2), where b_i is in the binary representation of y'. Since $\sum d_j = n$, T has sum ny'. We can assume that $e_{i,1} = 1$, so let's take this out of our multiset, so now our multiset is $T' = T \setminus \{b_i\}$ with sum $ny' - b_i$. By our Lemma 3.1, we have an odd number of solutions if and only if $x - b_i$ is a submask of $ny' - b_i$.

§3.3 Putting it all together

To summarize:

- Count the contribution of each bit independently.
- Use inclusion-exclusion to sum over all submasks of y.

- We can solve this relaxed subproblem by checking if $x - b_i$ is a submask of $ny' - b_i$.

Despite some of the elaborate reasoning required, the resulting implementation is remarkably simple. I'll leave my submission here for reference.

§4 Vandermonde Solution

The editorial provides a shorter method of arriving at the same implementation of the solution described above using Lucas' theorem and Vandermonde's identity. Again, we'll count the contribution of each power of 2 in the binary representation of y and use inclusion-exclusion to xor-sum over all submasks of y. We wish to count the parity of the number of solutions to

$$a_1 + a_2 + \cdots + a_n = x,$$

where each a_i is a submask of y', and b_i is in the binary representation of a_1. (Since each a_j provides the same contribution to the total score and n is odd, the total score is equal to the contribution of a single a_j.) We can assume we have already added b_i to the sum, so then a_i must be a submask of $y' - b_i$. Using Lucas' theorem, we can encode these submask constraints as the parity of

$$\binom{y' - b_i}{a_1}\binom{y'}{a_2}\cdots\binom{y'}{a_n},$$

since this product will be odd if and only if each binomial coefficient is odd, which requires that a_j is a submask of y' (or $y' - b_i$ in the case of a_1). Thus, we can count the parity of the number of solutions as

$$\sum_{a_1+a_2+\cdots+a_n=x-b_i} \binom{y' - b_i}{a_1}\binom{y'}{a_2}\cdots\binom{y'}{a_n} \pmod 2.$$

By Vandermonde's identity, this is equal to

$$\binom{ny' - b_i}{x - b_i} \pmod 2.$$

Again by Lucas' theorem, we just need to check if $x - b_i$ is a submask of $ny' - b_i$. I find this solution particularly remarkable, since binomial coefficients do not appear in either the problem statement or the final implementation; they are only used an intermediate step.