XOR and Gaussian Elimination

md.mottakin.chowdhury

June 2018

1 Maximum XOR Subset

- The idea here is to choose a number which has MSB with maximum value. The max(array) will have this feature. Say, maximum of array is a k-bit number and that number is M.

  Now, there can be multiple numbers which are k-bit numbers and have the same highest value bit as 1. So in that case we will XOR each of them with the M and put them again in the input array and remove M (ideally, we can choose any one of those k-bit numbers as M and put others in array after XORing with it). At the same time we will keep the number M in some other array say **ans**[].

- We will keep doing step 1 until we get 0 as maximum in the array i.e. this loop will run for iterations = number of bits the maximum of array has.

- Now will have an **ans**[] array which will be of size = number of bit the maximum of array contains.

- Now we will initialize the **answer** variable to given 0 and loop for each value in array ans[], if value of **answer** variable is going to increase with the inclusion of ith element, we will update the answer variable with the new value as **answer** XOR **ans[i]**, else we will keep it as it is. Return the answer/ans.

2 A Subset with Given Value

We have a set of numbers S and we want to check if there is a subset with XOR equals X.

We will assume that X and all elements of S are N-digit binary numbers (with leading zeros if necessary).

Start with a number K consisting of N zeros. We will keep updating K at each step, trying to achieve X in the end. The first question is whether the first bit of X is 1. If so, we need to xor K with some element in the subset which has first bit 1 (If such an element does not exist, it is impossible).

So let us assume that we have managed to get the correct value for the first bit of K. How do we move on to the next bit? We have to keep XORing K with elements of S, but we need to ensure that this XORing does not change the first bit itself. For this, we will modify S such that all elements have first bit as 0. Take some element E with first bit 1. We will remove this element from S, but XOR every other element with first bit 1 with E. This will result in all elements in S having first bit as 0. Thus the problem is now similar to the starting situation with N reduced by 1. Keep doing this till K becomes equal to X or till some bit position cannot be set to the right value.

An example: Suppose $S = 1, 2, 3, 4, 5$ and $X = 6$. We then have $N = 3$.

- $K = 000$, $X = 110$, $S = 001, 010, 011, 100, 101$

  First bit needs to be flipped. S contains an element with first bit 1 (We choose 100). We remove 100 from S and XOR 101 with it, turning it into 001. XORing changes K to 100.

- $K = 100$, $X = 110$, $S = 001, 010, 011, 001$

  Note how all elements have first bit 0. Second bit needs to be flipped now. We pick 011 from the set and remove it, XORing changes 010 to 001. XORing changes K to 111.

- $K = 111$, $X = 110$, $S = 001, 001, 001$

  All elements have second bit 0 as well now. Third bit of K needs to be flipped which can be achieved by choosing 001 to XOR with K.