XOR RELATED PROBLEM

Ajay Kumar

Date 28Jan 2023

The whole technique can be divided into two main parts, some problems can even be solved by using only the first part

- Represent each given number in it's binary form and consider it as a vector in the Z_2^d vector space, where d is the maximum possible number of bits. Then, xor of some of these numbers is equivalent to addition of the corresponding vectors in the vector space.
- Somehow, relate the answer to the queries of second type with the basis of the vectors found in Part 1.

I'm sure most of you have already made this observation by yourselves at some point. Suppose, we're xor-ing the two numbers Let's do it below

z_2^d

This is a vector space defined as follows:

- V ={0,1,...,2^d-1}.Each number stands for an array of d bits.
- If x , y ∈ V , then x + y is defined to be x ⊕ y . Note that x + y ∈ V is satisfied.

  For any c ∈ Z_2, let cx = x + x + . . . x . So cx = 0 if c is even and cx = x if c is odd.

- z_2^d is indeed closed under both addition and scalar multiplication.
- From now on, we'll do all calculations (mod 2) and substitute + in place of ⊕.

Span

what are span({3,5}),span({3,5,6}), and span ({}) ?

Ans: Note that $3 \oplus 5 = 6.$

span({3,5}) =span({3,5,6})= {0,3,5,6}.

span({})={0}.

Useful properties:

Firstly, $v_{m+1} \in \ span(v_1, \ldots, v_m)$ $implies$ $span(v_1, \ldots, v_m, v_{m+1}) = span(v_1, \ldots, v_m)$.

Basis

- if V=`{0,3,5,6}` then {3,5} is a basis for V, but {3,5,6} is not because $3 + 5 + 6 = 3 \oplus 5 \oplus 6 = 0$. Of course {3,6} and {5,6} are also bases for V.

Que:

- In term of n,how many distinct element are contained within span$(v_1, \ldots, v_n)$, if $v_1, \ldots, v_n$ form a basis?

Ans

if any x could be written as a linear combinations of $v_1, v_2 \ldots, v_n$ in more than one of the 2^n possible ways, this would contradict the definition of basis. It follows that $|v| = 2^n$

- All the vectors here belong to $Z^d{}_2$, so they are representable by a bitmask of length d;
- Suppose at each step,we're taking an input vector $\overrightarrow{v_i}$ and we already have a basis previously taking vectors $\overrightarrow{v_i}, \overrightarrow{v_2}, \ldots, \overrightarrow{v_{i-1}},$ and now we need toupdate the basis such that it can also represent the new vector $\overrightarrow{v_i}$.
- In order to do that ,we first need to check whether $\overrightarrow{v_i}$ is representable using our current basis or not.
- If it is, then this basis is still enough and we don't need to do anything, But if it's not we just add this vect $\overrightarrow{v_i}$ to the sect of basis.
- So the only difficult that remains is ,to efficiently check whether the new vector is representable by the basis or not.
- In order to facilitate this purpose , we use property 1 to sligtly modify any new vectors befor inserting it in the basis, being careful not to breakdown the basis. This way, we can have more control over the form of our basis vetors.

- Let $\overrightarrow{v_i}$ be the first position in the vector's binary representation, where the bit is set
- We make sure that all the basis vector have a different f value.
- Here's how to we do it. Initially , there are no vecotrs in the basis, so we're find, there are no f values to collide with each other. Now suppose we're at the i'th step, we're checking if vector $\overrightarrow{v_i}$ is representable by the basis or not, Since, all of our basis have a different f value, take the one with least f value among them, let's call this basis vector $\overrightarrow{b_1}$.
- if $f(\overrightarrow{v_i}) < f(\overrightarrow{b_1})$ then no matter how we take the linear combination of the basis vectors' can have 1 at positon $f(\overrightarrow{v_i})$. So $\overrightarrow{v_i}$ will be a new basis vector , and since it's fa value is already different from the rest of the basis vector we can insert it into the set as it is and keep a record of it's f value.
- But if $f(\overrightarrow{v_i}) == f(\overrightarrow{b_1})$, then we must subtract $\overrightarrow{b_1}$ for the $\overrightarrow{v_i}$, if we want to represent linear combinations of the basis vectors , since no other basis vector has bit 1 at position $f(\overrightarrow{v_i}) = f(\overrightarrow{b_1})$. So we subtract $\overrightarrow{b_1}$ $from$ $\overrightarrow{v_1}$ and move on to $\overrightarrow{b_2}$.

- Note that by changing the value of $\overrightarrow{v_i}$ we're not causing any problem according to property 1, $\overrightarrow{v_i}$ and $\overrightarrow{v_i} - \overrightarrow{b_1}$ is same use to us. If in some later step we find out $\overrightarrow{v_i}$ is actually not representable the the current basis ,we can still ust in's changed value in the basis , since the set of vectors int he space representable by this new basis would've been the same if we inserting the original $\overrightarrow{v_i}$ instead.
- if, after iterating thorough all the basis vector $\vec{b}'s$ and subtracting the from $\overrightarrow{v_i}$, if needed we still find out that $\overrightarrow{v_i}$ s not null vector. it means that the new changed $\overrightarrow{v_i}$ has a larger value of f that all other basis vectors. So we have to insert it into the basis and keep a record of it's f value.

```
int addelem(int val){
   int i;
   for(i=0;i<maxbits;i++){
      if(val&(1<<i)){
         if(alloted[i]){
            val^=responsible[i];
         }
         else{
            responsible[i]=val;
            alloted[i]=1;
            break;
         }
      }
   }
   return 0;
}
```