Count no. of sub-array with xor = m in a array of size n where $1 \le n \le 10^6$

a → 1 2 3 4 5 6

If we do $\underbrace{1 \wedge 2 \wedge 3 \wedge 4} \wedge \underbrace{1 \wedge 2} \rightarrow 3 \wedge 4$

Same as pre sum technique pre[i] − pre[j] = pre[j−i]

But how to find if a sub-array has m xor-sum?

$m = 3 \wedge 4$

$\underbrace{1 \wedge 2 \wedge 3 \wedge 4}_{1^{st}} \wedge \underbrace{1 \wedge 2}_{2^{nd}} \rightarrow 3 \wedge 4$ — pre xor between 1^{st} & 2^{nd} type sub-array

curren pre xor

some previous pre xor

$\underbrace{1 \wedge 2 \wedge 3 \wedge 4} \wedge \underbrace{3 \wedge 4} \rightarrow 1 \wedge 2$

If we have encounted any pre xor with this value that mean a subarray exists with xor = m.

$a_i = 3$ $\quad$ $a_j = 4$

$a_i = 011$ $\quad$ $a_j = 100$

$a_i = 011 \,\&\, 100 = 000$

$a_j = 011 \mid 100 = 111$

↳ 1 bit is tronfered

$a_i = 3$ $\quad$ $a_j = 3$

$a_i = 011 \,\&\, 011 = 011$

$a_j = 011 \mid 011 = 011$

No change

So if a_i & a_j are different than there always be a increse of set bit in one number and other will be reduced.

Goal/Aim increas a_i^2

$1^2 - 1$

$2^2 - 4$

$3^2 = 9$

$4^2 = 16$

$5^2 = 25$

So we will be profited by making all a_i as big as possible

eg $\quad 1 \quad 3 \quad 5$

$\quad 001 \quad 011 \quad 101$

$1^2 + 1^2 + 7^2 = 51$

| 1 | 1 | 3 | → 111

| 0 | 0 | 2 | → 001

| 0 | 0 | 1 | → 001

Different base

~~Representation~~ find an integer greater or equal to n in that can be terms of 3's different power

eg $9 = 3^2$ ✓

$2 = \underbrace{3^0 + 3^0}_{\text{same power}}$ ✗

$12 = 3^1 + 3^2$ ✓

ternary representation work same as binary

81	27	9	3	1	
4	3	2	1	0	
0	1	0	0	1	— n

$n = 3^0 \times 1 + 3^1 \times 0 + 3^2 \times 0 + 3^3 \times 1 +$

$= 3 + 27 = 30$

When will the number have same power of 3

↓ → observe

81	27	9	3	1	
0	0	0	2	1	— n

As base is 3 it can have value 0, 1, 2

$1 + 2.3 = 7$ → It will cause problem

81	27	9	3	1	
0	0	0	2	1	— n
0	0	1	0	0	— n'

$n = 7$

$n' = 9$ n' will always be greater than n

if $n = 022 = 8 < n'$

So to remove the problem we set the first 0 encountered after a bit's value 2