→ BIT MASKING

→ Let's say we have four types of fruits : Apple , Orange , Banana , Litchi .

Let's assign these fruits with 0-based indexes.

Apple → 0
Orange → 1
Banana → 2
Litchi → 3

Now, distribute these fruits to 4 person:

1 → 2, 3
2 → 0, 1, 2
3 → 1, 3
4 → 0, 3

Now, let's say we have to find the intersection of 1st and 2ND person's fruits:

Now, if we go through the ordinary approach and if the arrays are sorted. It would take us O(N) time complexity for finding the intersection.
But, we can use Bit-masking for a faster approach.

1	→ 2, 3	→ 1 1 0 0	⇒	12
2	→ 0, 1, 2	→ 0 1 1 1	⇒	7
3	→ 1, 3	→ 1 0 1 0	⇒	10
4	→ 0, 3	→ 1 0 0 1	⇒	9

We have masked every array with a binary number and have set their corresponding Bits.
e.g for 1st person, we have taken a

binary number with 2^{ND} and 3^{RD} Bit as set
Similarly for 2^{ND} person 0^{TH}, 1^{ST} and 2^{ND} Bit as set.
And so on...

Here, Set Bit represents the availability of the particular Bit (Indexed) fruit.

Now, If we have to find the intersection of 1^{ST} and 2^{ND} person's fruits, we will simply take AND (&) and our operation will be done in O(1) ~~of~~ complexity.

e.g: 1 → 2,3 → 1100 ⇒ 12
2 → 0,1,2 → 0111 ⇒ 7

So, if we do, 12 & 7

$$
\begin{array}{r}
1100 \\
\& \ 0111 \\
\hline
0100
\end{array}
$$

→ This represents they have 1 common fruit which is of 2^{ND} type, ~~of fruit in common left then~~, as resultant has 2^{ND} bit as set.

→ Bit Masking - Limitations:

→ We can't take bigger values of n here.
e.g UNSIGNED INT has a capacity of 32 Bits.
So, it can only store data of n = 32.
Similarly UNSIGNED LONG LONG can store data of n = 64.