What is Bit Masking?

A number can be represented as a sequence of bits, where the ith bit represents the contribution of 2^i to the number. Bitmasking is a method to see this sequence as an ordered set, where each position denotes an element of the set and its value denotes whether the element is present in the set or not. Let us try to see this by an example.

Consider the following set:
A = {1, 2, 3, 4, 5}

We can represent any subset A of using a bitmask of length 5, with an assumption that if i_th bit is set then it means i_th element is present in the subset. So the bitmask 01010 represents the subset {2, 4}.

Note: We consider the elements from left to right but the bits are numbered right to left. So the zeroth element of the set is the 1^{st} element which is represented by the rightmost bit of the mask.

Operations in Bitmasks

We will now see how operations like addition, deletion and checking for an element in the set are performed using bitmasks.

Addition: We assume that the element is already not present in the set. To add the ith element to the set, we need to set the ith bit of our mask to be one. This can be done by setting the ith bit of the mask to be 1 by taking a bitwise or with (1<<i).
For example:
Consider our set is {1, 2, 3} and our subset given is {1, 3}. If we want to add the number 2 to the set, i.e make the 1st position of our mask to show 1, then we have,

mask = 5 $(101)_2$, k = 1
new_mask = mask | (1<<(1)) = $(101)_2$ | $(010)_2$ = $(111)_2$ = 7

Note: *mask* represents our already present subset and *new_mask* represents our new subset after addition.

Deletion: We assume that the element is already present in the set. To remove the ith element from the set, we need to unset the ith bit of our mask. This can be done by setting the ith bit of the mask to be 0 by taking a bitwise xor with (1<<i).
For example:
Consider our set is {1, 2, 3} and our subset given is {1, 3}. If we want to remove the number 3 from the set, i.e make the 2nd position of our mask to show 0, then we have,

mask = 5 $(101)_2$, k = 2

new_mask = mask ^ (1<<(2)) = $(101)_2$ ^ $(100)_2$ = $(001)_2$ = 1

Checking for an element in the set: To check if an element is present in the given subset or not, we need to check if the ith bit of the bitmask is 1 or zero, where i is the position of the element in the ordered set. We can use bitwise AND operations to check this efficiently by taking bitwise AND with (1<<i).

For example:
Consider our set is {1, 2, 3} and our subset given is {1, 3}. If we want to check if the number 3 is present in the set or not, then we have,

mask = 5 $(101)_2$, k = 2
So, let n = (1 << k) = 2^k = 4 = $(100)_2$
x & n = $(101)_2$ & $(100)_2$ = $(100)_2$ = 4.
Since the result of the bitwise AND is not 0, the 3rd bit of x is set and number 3 is present in the subset.