* Types of Recursions :-

1) Linear ; Eg :- fibonacci no. $f(n) = f(n-1) + f(n-2)$

2) Divide & Conquer ; Eg :- binary search $f(n) = f(N/2) + a_1$

Note :-
You can represent recursion in the form of any question.

* Divide and Conquer Recurrences :-

This is actually same as $f(n) = f(N/2) + O(1)$

Form :-

$$T(x) = a_1 T(b_1 x + \varepsilon_1(x)) + a_2 T(b_2 x + \varepsilon_2(x)) + \cdots + a_k T(b_k x + \varepsilon_k(x)) + g(x)$$

for $x \geq x_0$ → same constant

If we relate this with $f(n) = f(N/2) + a_1$ we can say in binary search

Eg :- $T(N) = T(N/2) + C$ → constant.

$\therefore a_1 = 1$
$b_1 = 1/2$ $\quad g(N) = \Theta C$
$\varepsilon_1(x) = 0$

So, both are same & this is the form of divide & conquer recurrences

Other relationship can be something like this

$$T(N) = \underbrace{9}_{a_1}\, T\left(\underbrace{\frac{N}{3}}_{b_1}\right) + \underbrace{\frac{4}{3}}_{a_2}\, T\left(\underbrace{\frac{5}{6}N}_{b_2}\right) + \underbrace{4N^3}_{g(n)}$$

((a)) ((b)).

Q. What is the $g(n)$?

Ans: It means that

let say ((a)) ((b)) ↑

When you get the answer from ((a)) + what you're doing with that answer it takes how much time!

((b)) basically means the ((a)) recursion call is over then what amount of time complexity is required to do actually something with those recursion calls that are over right now. Extra time require at that step!

So, Extra time require at that step is equal to checking whether a number is greater greater than or equal to or less than middle that takes constant amount of time so, that's why we have constant over here.