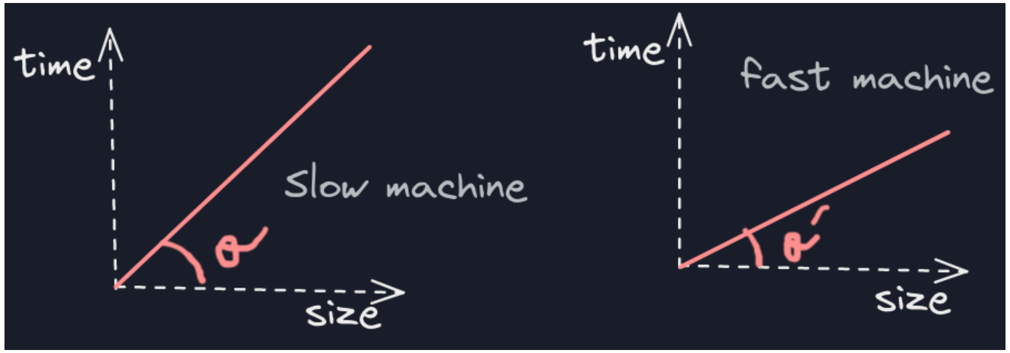
Complexity

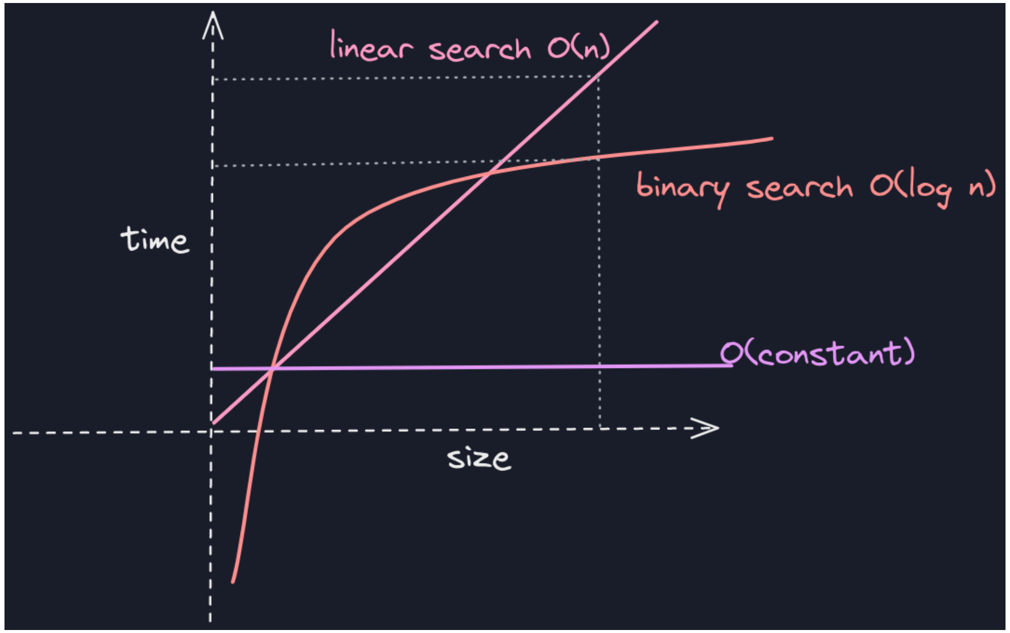
Time Complexity

Time complexity is not the time taken by the machine.



It is a mathematical function that tells us how time grows as the size grows.

Note: While determining time complexity, take size → infinity



Therefore, O(constant) < O(log(n)) < O(n).

Time Complexity tells us about the nature of graph [does not depend on constants or less significant terms]

For example:

$O(3n^3 + 4n^2 + 5n + 6)$

$= O(n^3 + n^2 + n)$ [ignoring coefficients and constants]

$= O(n^3)$ [ignoring less significant terms]

Big O Notation :

Big O Notation means the complexity is less or equal to n^3.

$$\lim_{n \to \text{infinity}} \frac{f(n)}{g(n)} < \text{infinity}$$

let $\underbrace{O(n^3)}_{g(n)} = \underbrace{O(6n^3 + 3n + 5)}_{f(n)}$,

$$\Rightarrow \lim_{n \to \text{infinity}} \frac{6n^3 + 3n + 5}{n^3}$$

$$\Rightarrow \lim_{n \to \text{infinity}} 6 + 3/n^2 + 5/n^3$$

$$\Rightarrow 6 < \text{infinity}$$

Little O Notation :

Little O Notation means the complexity is less than n^3.

Big Omega Notation :

Big Omega Notation means the complexity is more or equal to n^3.

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} > 0$$

let $f(n) = n^4$,

$$\Rightarrow \lim_{n \to \infty} n > 0$$

Little Omega Notation :

Little Omega Notation mean the complexity is more than n^3.

Big Theta Notation :

Big Theta = Big O + Big Omega

$$0 < \lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$$

Space Complexity

Auxiliary Space is the extra or temporary space used by an algorithm.

Space Complexity is the total space taken by the algorithm with respect to the input size.

Space Complexity = Auxiliary Space + Input Size

Note: Auxiliary space is a better criterion than Space Complexity.

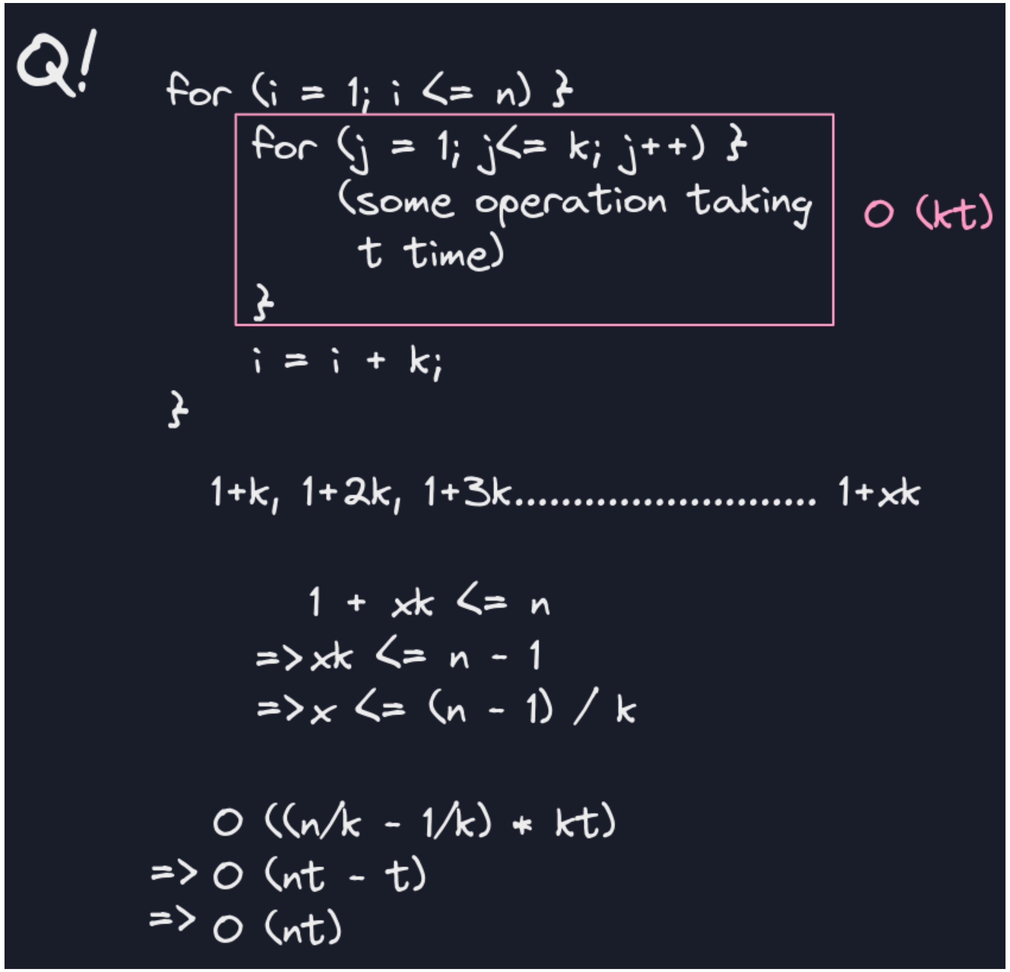
For Example :

Merge Sort → O(n) [Auxiliary Space]

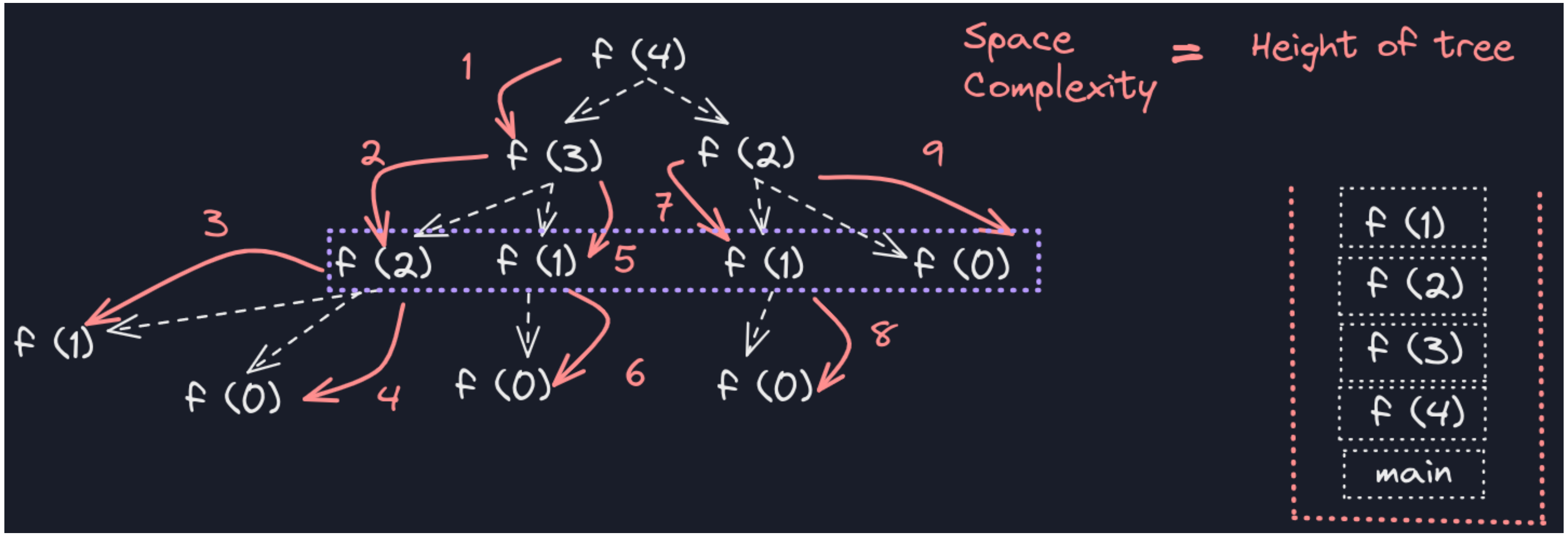
Insertion Sort → O(1) [Auxiliary Space]

Heap Sort → O(1) [Auxiliary Space]

All sorting algorithms use O(n) Space Complexity.



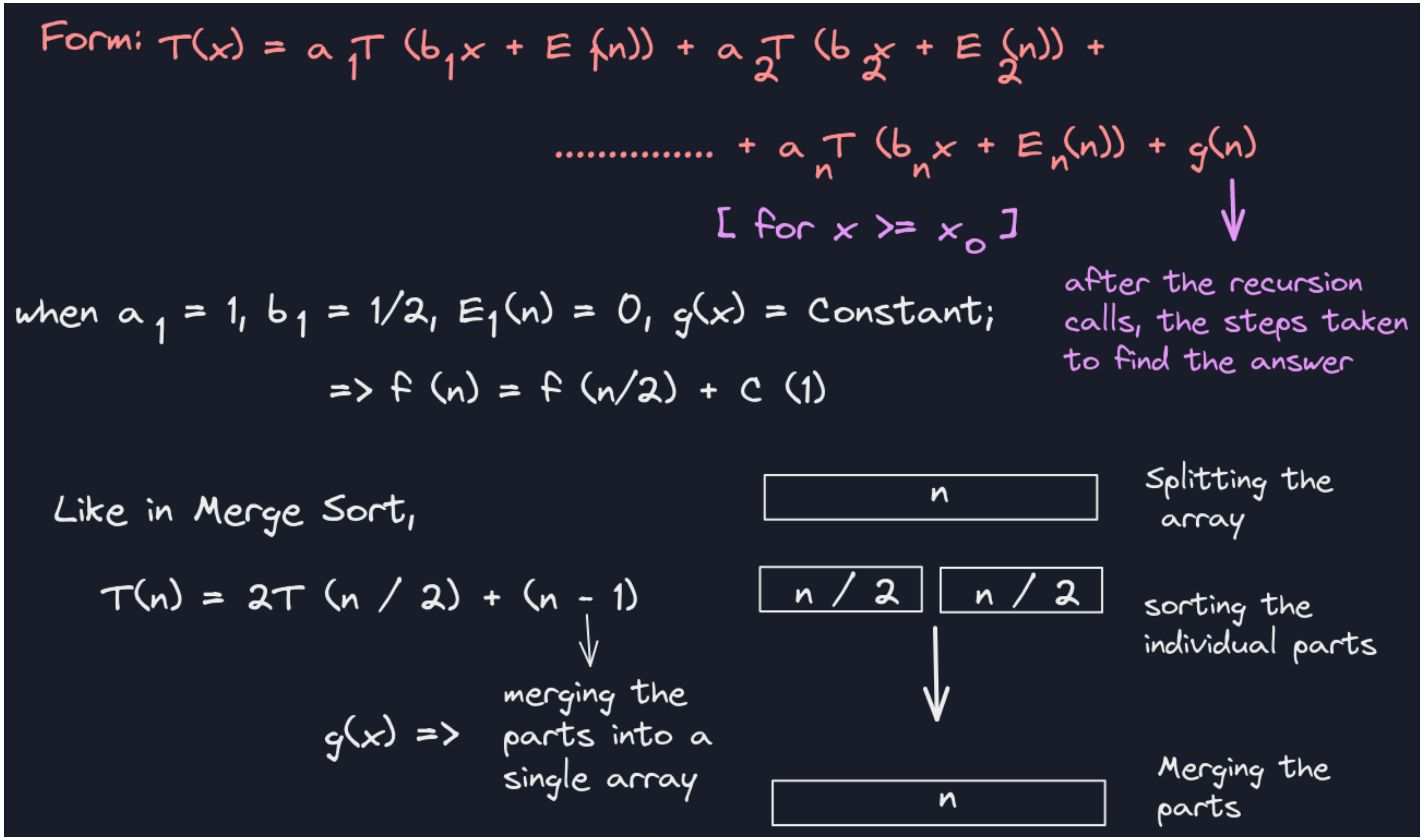
Recursive Tree



At any particular point in time, no two function calls can be in the same level while the call of recursion will not be in the stack at the same time.

Types of Recursion:

Divide & Conquer Recurrence Relation :



$$T(x) = a_1T(b_1x + E_1(n)) + a_2T(b_2x + E_2(n)) + \ldots + a_kT(b_kx + E_k(n)) + g(n)$$

g(n) is the steps taken to find the answer after the completion of recursion.

Example :

when $a_1 = 1, b_1 = 1/2, E_1(n) = 0, g(x) = C$,

we get, $T(n) = T(n/2) + C$

How to actually solve to get time complexity?

1. Plug & Chug
2. Master's Theorem

3. Akra Bazzi Formula [Best Method]

Akra Bazzi Formula :

$$T(x) = \theta(x^p + x^p \int_1^x \frac{g(u)du}{u^{p+1}})$$

What is p ?

- $a_1 b_1^p + a_2 b_2^p + \ldots = 1$
- $$\sum_{i=1}^{k} a_i b_i^p = 1$$
- NOTE: If $p < power\ of\ g(x), \therefore ans = g(x)$

1. Example:

$$T(n) = 2T(\frac{n}{2}) + (n-1)$$

- Here, $a_1 = 2, b_1 = \frac{1}{2}, g(n) = n-1$

  $2(\frac{1}{2})^p = 1 \therefore p = 1$

- By applying Akra Bazzi formula :

$$T(x) = \theta(x^1 + x^1 \int_1^x \frac{u-1}{u^2} du)$$

$$= \theta(x + x(\int_1^x \frac{du}{u} - \int_1^x \frac{du}{u^2}))$$

$$= \theta(x + x(log(u) + \frac{1}{u})_1^x$$

$$= \theta(x + x(logx + \frac{1}{x} - 1))$$

$$= \theta(xlog(x) + 1)$$

$$= \theta(xlog(x))$$

2. Example:

   $T(x) = 2T(\frac{n}{2}) + \frac{8}{9}T(\frac{3n}{4}) + n^2$

   - Here, $a_1 = 2, b_1 = \frac{1}{2}, a_2 = \frac{8}{9}, b_2 = \frac{3}{4}, g(u) = n^2$,

     $2(\frac{1}{2})^p + \frac{8}{9}(\frac{3}{4})^p = 1$

     $\therefore p = 2$

   - By Applying Akra Bazzi formula :

$$T(x) = \theta(x^2 + x^2 \int_1^x \frac{u^2}{u^3} du)$$

$$= \theta(x^2 + x^2 log(x))$$

$$= \theta(x^2 log(x))$$

Linear Recurrence Relation

Solving Homogeneous Linear Recurrences :

The form of a recurrence relationship without any operation after recursion like g(x).

$$f(x) = a_1 f(x-1) + a_2 f(x-2) + \ldots + a_n f(x-n)$$

$$f(x) = \sum_{i=1}^{n} a_1 f(x-i)$$

1. Example [Recursion for Fibonacci] : $f(n) = f(n-1) + f(n-2)$

   - Step 1: Putting $f(n) = \alpha^n$, where $\alpha = constant$

$$\Rightarrow \alpha^n = \alpha^{n-1} + \alpha^{n-2}$$

$$\Rightarrow \alpha^n - \alpha^{n-1} - \alpha^{n-2} = 0$$

     dividing both sides by α^{n-2},

$$\Rightarrow \alpha^2 - \alpha - 1 = 0$$

$$\Rightarrow \alpha = \frac{1 \pm \sqrt{5}}{2}$$

$$\Rightarrow \alpha_1 = \frac{1+\sqrt{5}}{2}, \alpha_1 = \frac{1-\sqrt{5}}{2}$$

- Step 2: If α_1 & α_2 are 2 roots, we can write

  $f(n) = c_1\alpha_1^n + c_2\alpha_2^n$ is a solution,

  [where $c_1\alpha_1^n = f(n-1)$, $c_2\alpha_2^n = f(n-2)$].

$$\therefore f(n) = c_1(\frac{1+\sqrt{5}}{2})^n + c_2(\frac{1-\sqrt{5}}{2})^n$$

- Step 3: number of roots = number of answers

  So, we have 2 answers already

  $\therefore f(0) = 0$ & $f(1) = 1$

  for $f(0) = 0$,

$$c_1(\frac{1+\sqrt{5}}{2})^0 + c_2(\frac{1-\sqrt{5}}{2})^0 = 0$$

$$\Rightarrow c_1 + c_2 = 0$$

$$\Rightarrow c_1 = -c_2$$

  for $f(1) = 1$,

$$c_1(\frac{1+\sqrt{5}}{2}) + c_2(\frac{1-\sqrt{5}}{2}) = 1$$

$$\Rightarrow c_1(\frac{1+\sqrt{5}}{2}) - c_1(\frac{1-\sqrt{5}}{2}) = 1$$

$$\Rightarrow c_1 = \frac{1}{\sqrt{5}} \therefore c_2 = -\frac{1}{\sqrt{5}}$$

- Step 4: Putting c_1 & c_2 in f(n)

$$f(n) = \frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n - \frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^n$$

$$\Rightarrow f(n) = \frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n]$$

when $n \rightarrow \infty$, $(\frac{1-\sqrt{5}}{2})^n$ is less-dominating term.

$$\Rightarrow f(n) = \frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n$$

$\therefore$Time Complexity of Fibonacci with recursive tree is $O(\frac{1+\sqrt{5}}{2})^n = O(1.6180)^n$.

2. Example [Equal Roots]: $f(n) = 2f(n-1) + f(n-2)$

- Step 1 : Putting $f(x) = \alpha^n$, where $\alpha = constant$.

$$\alpha^n = 2\alpha^{n-1} + \alpha^{n-2}$$

$$\Rightarrow \alpha^n - 2\alpha^{n-1} - \alpha^{n-2} = 0$$

Dividing both side by α^{n-2},

$$\alpha^2 - 2\alpha - 1 = 0$$

$$\Rightarrow \alpha = \frac{2 \pm 2\sqrt{2}}{2}$$

$$\Rightarrow \alpha = 1 \pm \sqrt{2}$$

$$\Rightarrow \alpha_1 = 1 + \sqrt{2}, \alpha_2 = 1 - \sqrt{2}$$

- Step 2 : If α_1 & α_2 are two roots, we can write.

$$f(n) = c_1\alpha_1^n + c_2\alpha_2^n$$

$$\Rightarrow f(n) = c_1(1+\sqrt{2})^n + c_2(1-\sqrt{2})^n$$

- Step 3 : number of roots = number of answers.

$\therefore f(0) = 0$ & $f(1) = 1$

for $f(0) = 0$, we get

$$f(0) = c_1 + c_2 = 0$$
$$\Rightarrow c_1 = -c_2$$

for $f(1)$, we get

$$f(1) = c_1(1+\sqrt{2}) + c_2(1-\sqrt{2}) = 1$$
$$\Rightarrow c_1(1+\sqrt{2}) - c_1(1-\sqrt{2}) = 1$$
$$\Rightarrow 2\sqrt{2}c_1 = 1$$
$$\Rightarrow c_1 = \frac{1}{2\sqrt{2}}, c_2 = -\frac{1}{2\sqrt{2}}$$

- Step 4 : putting c_1, c_2 in $f(n)$, we get

$$\Rightarrow f(n) = \frac{1}{2\sqrt{2}}(1+\sqrt{2})^n - \frac{1}{2\sqrt{2}}(1-\sqrt{2})^n$$

when $n \to \infty$, $\frac{1}{2\sqrt{2}}(1-\sqrt{2})^n$ is less-dominating,

$$\Rightarrow f(n) = \frac{1}{2\sqrt{2}}(1+\sqrt{2})^n$$

$\therefore$ Time Complexity of equal roots is $T(n) = O(1+\sqrt{2})^n = O(2.141)^n$.

Solving Non-Homogeneous Linear Recurrences :

$$f(x) = a_1 f(x-1) + a_2 f(x-2) + \ldots + a_n f(n-n) + g(n)$$
$$f(x) = \sum_{i=1}^{n} a_1 f(n-i) + g(n)$$

1. Example : $f(n) = 4f(n-1) + 3^n$, Given $f(1) = 1$

- Step 1 : Homogeneous Solution

$$f(n) = 4f(n-1) + 3^4$$

when $n \to \infty$, 3^4 is less dominating,

$$\Rightarrow f(n) = 4f(n-1)$$

Let $f(n) = \alpha^n$, we get

$$\alpha^n = 4\alpha^{n-1}$$

Dividing both side by α^{n-1},

$$\Rightarrow \alpha = 4$$

$$\therefore f(n) = c_1 4^n$$

- Step 2 : Non-Homogeneous Solution

$$f(n) = 4f(n-1) + 3^n$$

$$\Rightarrow f(n) - 4f(n-1) = 3^n$$

How to find for Non-Homogeneous Linear Recurrences?

- If $g(n)$ is exponential, let $g(n) = 2^n$ try $f(n) = 2^n c$.
- If $f(n) = 2^n c$ does not work, try $f(n) = (an + b)2^n$ or try $f(n) = (an^2 + bn + c)2^n$ & keep increasing the degree.
- If $g(n)$ is polynomial, let $g(n) = n^2 - 1$, try $f(n) = an^2 + bn + c$.
  - For Example : $g(n) = 2^n + n \Rightarrow f(n) = 2^n a + (bn + c)$

  Let $f(n) = c(3)^n$, we get

$$\Rightarrow c(3)^n - 4c(3)^{n-1} = 3^n$$

$$\Rightarrow c - \frac{4}{3}c = 1$$

$$\Rightarrow -\frac{1}{3}c = 1$$

$$\Rightarrow c = -3$$

$$\therefore f(n) = -3^{n-1}$$

- Step 3 : Adding both Homogeneous and Non-Homogeneous solutions

$$f(n) = c_1 4^n - 3^{n-1}$$

$$\Rightarrow f(1) = 4^n c_1 - 3^2 = 1$$

$$\Rightarrow c_1 = \frac{5}{2}$$

$$\therefore f(n) = \frac{5}{2} 4^n - 3^{n+1}$$