Analysis of Algorithms.

(*) To Solve any problem we can have multiple Solutions and to determine the best/optimal Solution among bunch of Solutions we calculate the time complexity of algorithm.

(**) Factor on which Time Complexity of algorithms depends:-

① Machine on which code is running

② Programming language used to Solve

③ System load at that time when code is compiled & run.

(**) We can calculate using asymptotic analysis approach.

(**) We are determining the order of growth of the algorithm

(**) **Best Case** ⇒ It is ideal case in which we want to achieve this but this is practically not possible for every case.

(**) Avg Case ⇒ It is generally not calculated because it totally depends upon the user. So, it is impractical to calculate.

(**) Worst Case ⇒ It is the case we generally calculate for the algorithm. It is practical and give idea about algorithm

Direct way to calculate the order of growth

① Ignore the lower order term

② Ignore the leading term constant.

(**) If the multiple terms are present in expression then

$C < \log\log n < \log n < n^{1/3} < n^{1/2} < n < n^2 < n^3 < n^4 < 2^n < n^n$

$\longrightarrow$

Increasing order of time taken

Example 1 ⇒ $f(n) = 2n^2 + n + 6$.

Soln → $\cancel{2}n^2 + \cancel{n} + \cancel{6}$ ⇒ Order of growth ⇒ n^2.

Example 2 → $f(n) = 1000n + 30$

Soln → $\cancel{1000}n + \cancel{30}$ ⇒ Order of growth ⇒ n

Example 3 → $f(n) = C_1 \log n + C_2$. | $g(n) = C_3 n + C_4 \log\log n + C_5$.

Soln ⇒ $f(n) \Rightarrow \cancel{C_1} \log n + \cancel{C_2}$ | $g(n) = \cancel{C_3} n + \cancel{C_4 \log\log n} + \cancel{C_5}$

$f(n) = \log n$ | $g(n) = n$.

(**) Compare from the chart we can observe that

$f(n)$ is more optimal than $g(n)$.

(##) ASymptotic Notation

① Big O ⇒ Exact order of growth or Upper [Worst]

② Thetha Θ ⇒ Exact order of growth [Avg Case]

③ Omega Ω ⇒ Exact order of growth or Lower [Best]

(**) Some important points about these notations

① Θ (Thetha) is calculated when you know exact order of growth, you know about exact Complexity of algo.

② O (Big) is Calculated when you don't know the exact order of growth, generally it is Calculated in IT-industry, because we can say through Big O refers to threshold Complexity or higher. It is also Called Worst Case Complexity.

③ Ω (Omega) is not generally Calculated because it is not correct way to know about the complexity of algorithm. It is also Called Best Case Complexity