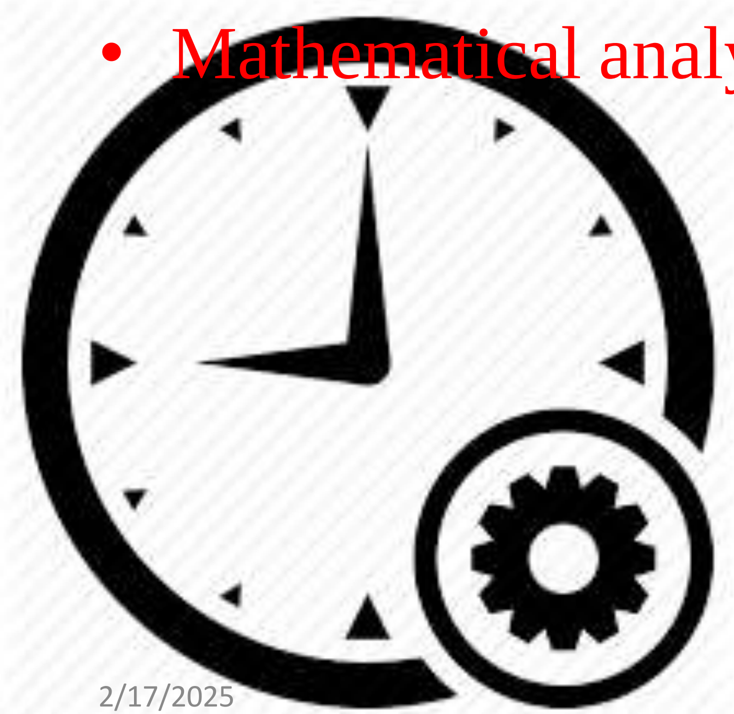
Fundamentals of the Analysis of Algorithm Efficiency

- Analysis Framework
- <u>Asymptotic Notations and its properties</u>
- Mathematical analysis of Recursive algorithms
- Mathematical analysis of Non - Recursive algorithms

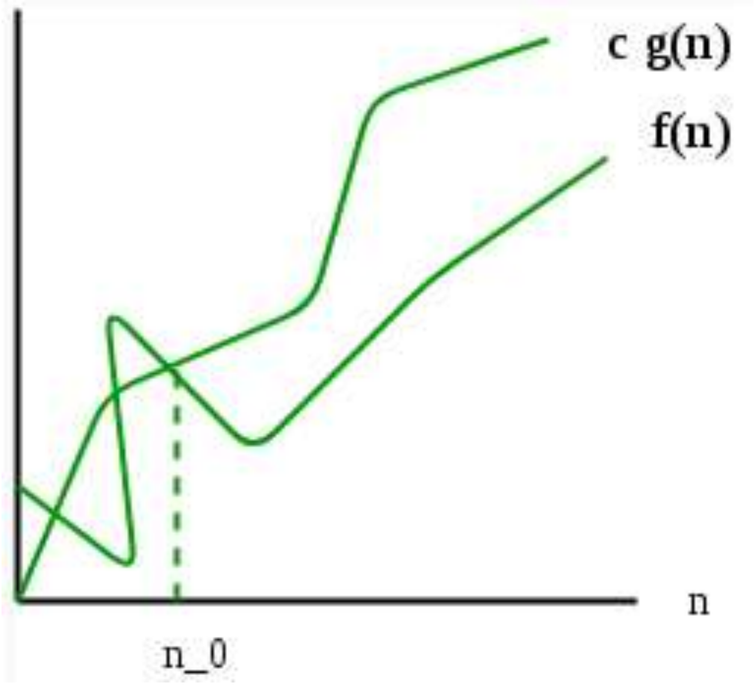


Asymptotic Notations and its properties

- Analysis framework – Efficiency – order of growth
- Order of growth – change in order of input size
- Study of performance changes of algorithm with change in order of input → *Asymptotic Analysis*
- Compare and Rank order of growth → 3 Notations
- Mathematical tool to represent the time complexity of algorithm for Asymptotic Analysis is ***Asymptotic Notation***
- *Notations*
  - Big O Notation (Worst-case efficiency)
  - Big Ω Notation (Best-case efficiency)
  - Big Θ Notation (Average-case efficiency)

Big O Notation (Worst-case efficiency)

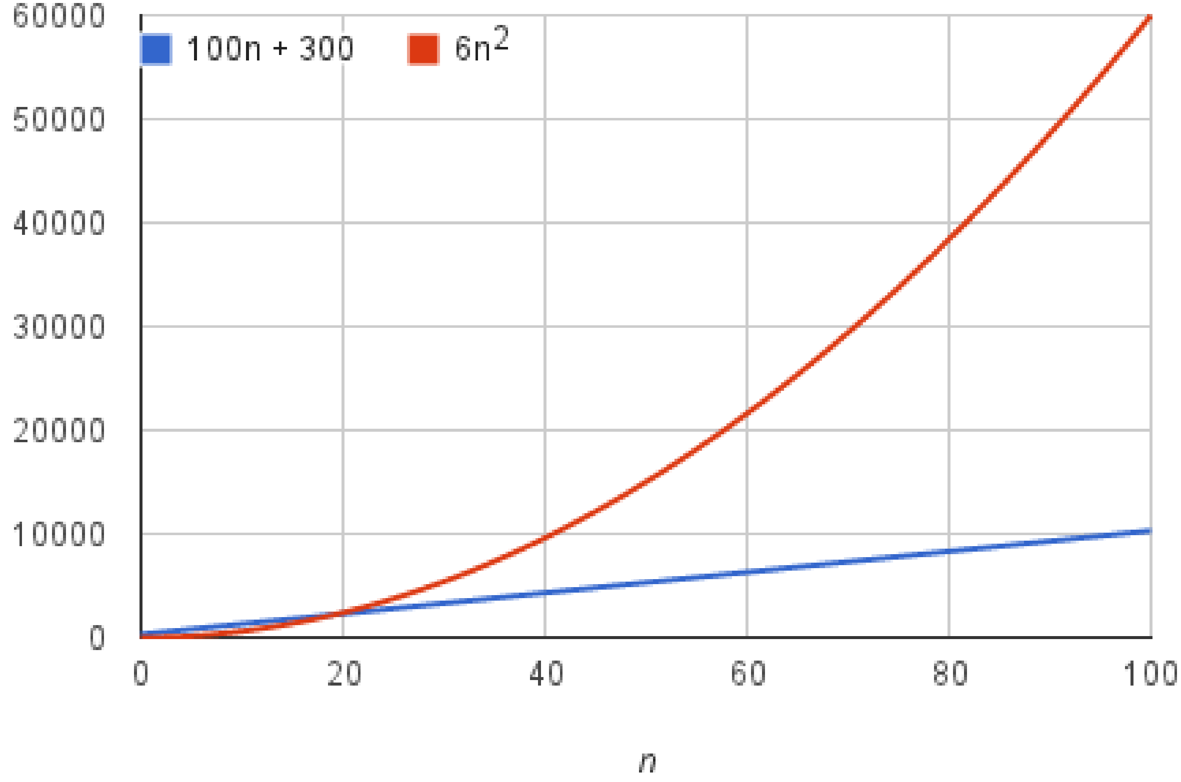
- Upper bound of the running time of an algorithm
- O(g(n)) = { f(n): there exist positive constants c and n0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n0$ }
- $f(n) \in O(g(n))$



Big O Notation (Worst-case efficiency)

n	f(n) = 100n+300	g(n) = $6n^2$
1	400	6
2	500	24
3	600	54
4	700	96
5	800	150
.		
.		
10	1300	600
.		
15	1800	1350
20	2300	2400
21	2400	2646
22	2500	2904
23	2600	3174

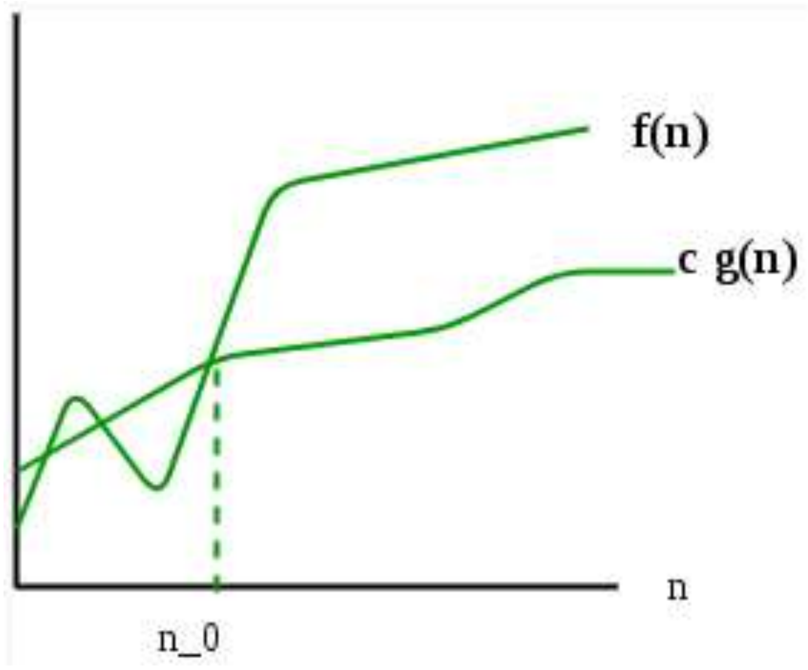
Big O Notation (Worst-case efficiency) - Example



<u>What is n_0 here ?</u>

Big Ω Notation (Best-case efficiency)

- lower bound of the running time of the algorithm
- $\Omega(g(n))$ = { f(n): there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$ }



Big Θ Notation (Average-case efficiency)

- Encloses the function from above and below
- upper and the lower bound of the running time of algorithm
- $\Theta(g(n))$ = { f(n): there exist positive constants c1, c2 and n0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$ }