الموضوع: ch3

التاريخ: / /

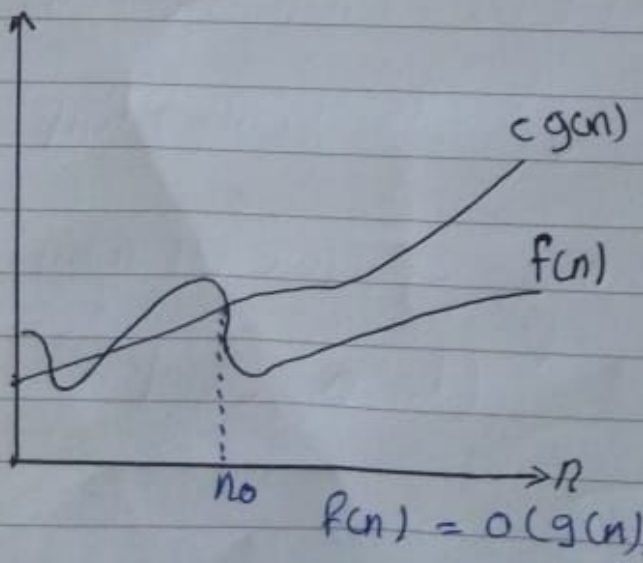
Growth of function

O(n) : is the formal way to express the upper bound of an algorithm's running time.

It measures the worst case time complexity or longest amount of time an algorithm can possibly take to complete.

For example

$O(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \leq f(n) \leq c\, g(n)$ for all $n \geq n_0 \}$.



g(n) is an asymptotic upper bound for f(n). if $f(n) \in O(g(n))$, we write $f(n) = O(g(n))$

التاريخ: / / الموضوع:

Big oh Notation, theorems

theorem 1 Sum of function if $f \in O(g)$, $f' \in O(g')$, then $f + f' \in O(g + g')$ As a corrolary, if $g = g'$, then $f + f' \in O(g)$

theorem 2 Product of functions if $f \in O(g)$, $f' \in O(g')$, then $f \cdot f' \in O(g \cdot g')$

theorem 3 Power of functions if $f \in O(g)$, then for all $\alpha > 0$, $f^{\alpha} \in O(g^{\alpha})$

theorem 4 Composition of function if $f \in O(g)$ and $f' \in O(g')$, then $f \circ f' \in O(g \circ g')$

theorem 5 Constants Contants, c, don't grow at all. That is, for any increasing function f of $\mathbb{N}$, $c \in O(f(n))$

theorem 6 Polynomials versus logs for any $\alpha > 0$, a Polynomial of that degree grows asymptotically faster than algorithm i.e, $\log(n) \in O(n^{\alpha})$

Theorem 7 Polynomials of lower degree Higher-degree polynomials grow more quickly i.e. for any $\alpha, \epsilon > 0$,

$$n^{\alpha} \in o(n^{\alpha+\epsilon})$$

theorem 8 if $f(n)$ is $o(g(n))$ and $g(n)$ is $o(h(n))$ then $f(n)$ is $o(h(n))$ [transitivity]

theorem 9 K is $o(1)$

Big oh Notations Examples

① $f(n) = 7n^4 + 3n^2 + 5n + 1000$ is $o(7n^4) = o(n^4)$

② $f(n) = 7n^2 + 3n\log(n) + 5n + 1000$ is $o(n^2)$

③ $f(n) = 7n^4 + 3^n + 100000$ is $o(3^n)$

④ $f(n) = 7n(n + \log(n))$ is $o(n^2)$

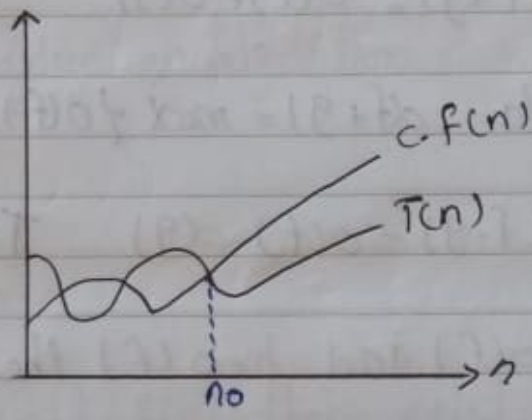
⑤ $f(n) = 7n^4$ is $o(n^4)$

Example :

$T(n) = 32n^2 + 17n + 1$

choose $c = 50$ $(32+17+1)$, $n_0 = 1$

$T(n)$ = is $O(n^2)$.



Example 2 :

A quadratic algorithm with processing time $T(n) = cn^2$ spends $T(N)$ seconds for processing N data items. How much time will be spent for processing $n = 5000$ data items, assuming that $N = 100$ and $T(N) = 1ms$?

The constant factor $c = \frac{T(N)}{N^2}$, therefore $T(n) = cn^2$

$T(n) = T(N)\frac{n^2}{N^2} = \frac{n^2}{1*(100)^2} = \frac{n^2}{10000}$ ms and $(T(5000) = \frac{(5000)^2}{10000} = 2,500$ ms

Example 3 :

Determine whether each statement is true or false and correct the formula in the latter case

① Rule of Sums $O(f+g) = O(f) + O(g)$ FALSE

the correct formula $O(f+g) = \max\{O(f), O(g)\}$?

② Rule of Product $O(f \cdot g) = O(f) \cdot O(g)$ TRUE

③ Transitivity if $g = O(f)$ and $h = O(f)$ then $g = O(h)$

FALSE, the correct formula if $g = O(f)$ and $f = O(h)$ then $g = O(h)$

④ $5n + 8n^2 + 100n^3 = O(n^4)$ TRUE

⑤ $5n + 8n^2 + 100n^3 = O(n^2 \log n)$ FALSE

the correct formula $O(n^3)$

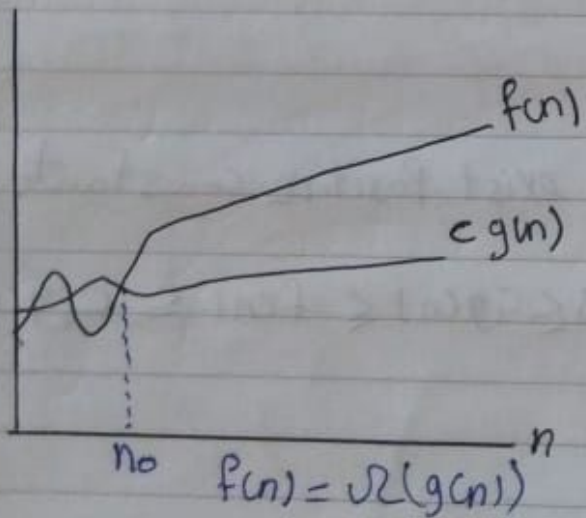
Omega Notation, Ω

The Ω(n) is the formal way to express the lower bound of an algorithm's running time.

It measures the best case time complexity or best amount of time an algorithm can possibly take to complete

for example

Ω(g(n)) = {f(n): there exist positive constants c and no such that $0 \leq cg(n) \leq f(n)$ for all $n \geq no$ }.



g(n) is an asymptotic lower bound for f(n). if f(n) ∈ Ω(g(n)), we write f(n) = Ω(g(n)).

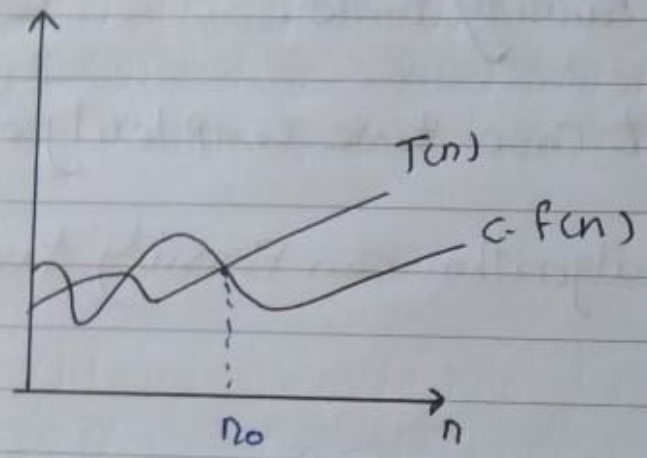
Example:

choose $f(n) = n^2$ $c = 32$

$T(n) = 32n^2 + 17n + 1$

$n_0 = 1$

$T(n)$ is both $\Omega(n)$ and $\Omega(n^2)$

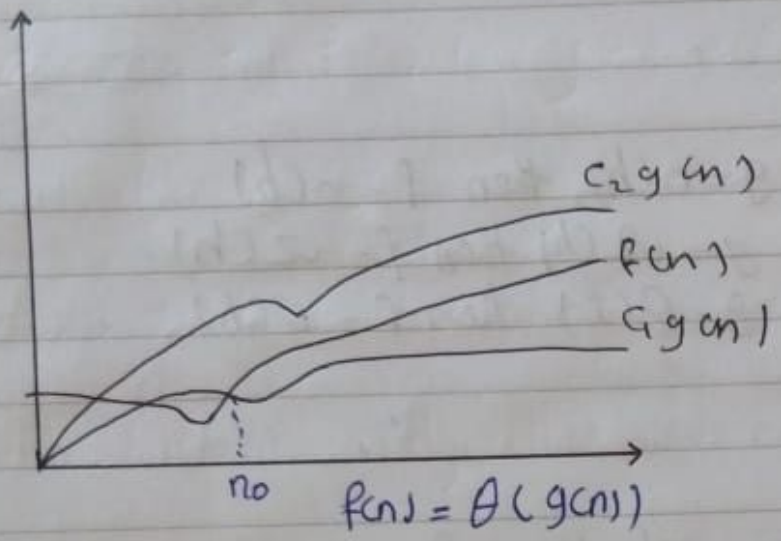


theta Notation, θ

the $\theta(n)$ is the formal way to express both the lower bound and upper bound of an algorithm's running time

for example

$\theta(g(n)) = \{ f(n)$: there exist Positive constants c_1, c_2 and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0 \}$.



$g(n)$ is an asymptotic tight bound for $f(n)$. if $f(n) \in \theta(g(n))$ we write $f(n) = \theta(g(n))$

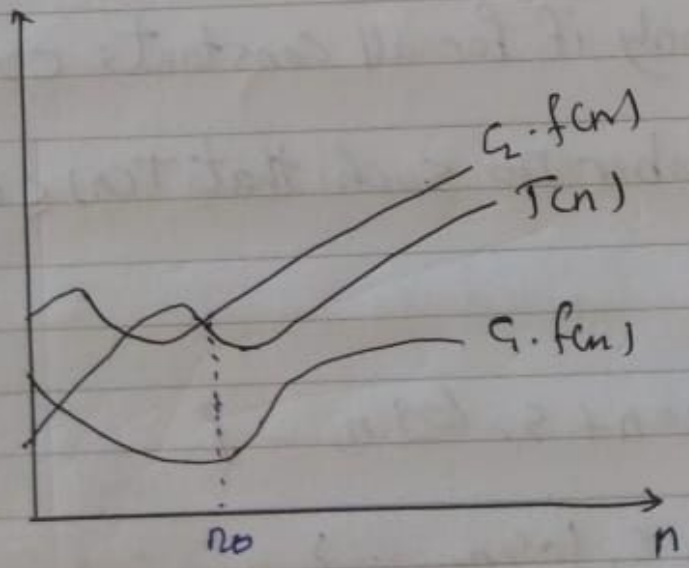
Example:

$T(n) = 32n^2 + 17n + 1$

$T(n)$ is $\theta(n^2)$

choose $f(n) = n^2$

$c_1 = 32$, $c_2 = 50$, $n_0 = 1$



Properties

Transitivity

- If $f = O(g)$ and $g = o(h)$ then $f = o(h)$
- if $f = \Omega(g)$ and $g = \Omega(h)$ then $f = \Omega(h)$
- if $f = \Theta(g)$ and $g = \Theta(h)$ then $f = \Theta(h)$

Additivity

- if $f = o(h)$ and $g = o(h)$ then $f + g = o(h)$.
- if $f = \Omega(h)$ and $g = \Omega(h)$ then $f + g = \Omega(h)$.
- if $f = \Theta(h)$ and $g = \Theta(h)$ then $f + g = \Theta(h)$.

Little oh Notation: o

$T(n) = o(f(n))$ if and only if for all constants $c > 0$, there exist a constant number n_0 such that $T(n) \leq c \cdot f(n)$ for all $n >= n_0$

$O(n^2) = \{n^2, 100n + 5, \log n, \ldots\}$

$o(n^2) = \{100n + 5, \log n, \ldots\}$

EX: $2n = o(n^2)$, but $2n^2 \neq o(n^2)$

Q: Let $T(n) = \frac{1}{2}n^2 + 3n$. which of the following statements are true?

- $T(n) = O(n)$ False
- $T(n) = \Omega(n)$ True choose $c = 0.5$ and $n_0 = 1$
- $T(n) = \Theta(n^2)$ True choose $c_1 = 0.5$, $c_2 = 4$ and $n_0 = 1$
- $T(n) = O(n^3)$ True choose $c = 4$ and $n_0 = 1$