Asymptotic notation:

The word **Asymptotic** means approaching a value or curve arbitrarily closely (i.e., as some sort of limit is taken).

Asymptotic analysis

It is a technique of representing limiting behavior. The methodology has the applications across science. It can be used to analyze the performance of an algorithm for some large data set.

In computer science in the analysis of algorithms, considering the performance of algorithms when applied to very large input datasets

The simplest example is a function $f\ (n) = n^2+3n$, the term 3n becomes insignificant compared to n^2 when n is very large. The function "$f\ (n)$ is said to be **asymptotically equivalent** to n^2 as $n \rightarrow \infty$", and here is written symbolically as $f\ (n) \sim n^2$.

Asymptotic notations are used to write fastest and slowest possible running time for an algorithm. These are also referred to as 'best case' and 'worst case' scenarios respectively.

"In asymptotic notations, we derive the complexity concerning the size of the input. (Example in terms of n)"

"These notations are important because without expanding the cost of running the algorithm, we can estimate the complexity of the algorithms."

Why is Asymptotic Notation Important?

1. They give simple characteristics of an algorithm's efficiency.
2. They allow the comparisons of the performances of various algorithms.

Asymptotic Notations:

Asymptotic Notation is a way of comparing function that ignores constant factors and small input sizes. Three notations are used to calculate the running time complexity of an algorithm:

1. Big-oh notation:

Big-oh is the formal method of expressing the upper bound of an algorithm's running time. It is the measure of the longest amount of time. The function **f (n) = O (g (n))** [read as "f of n is big-oh of g of n"] if and only if exist positive constant c and such that

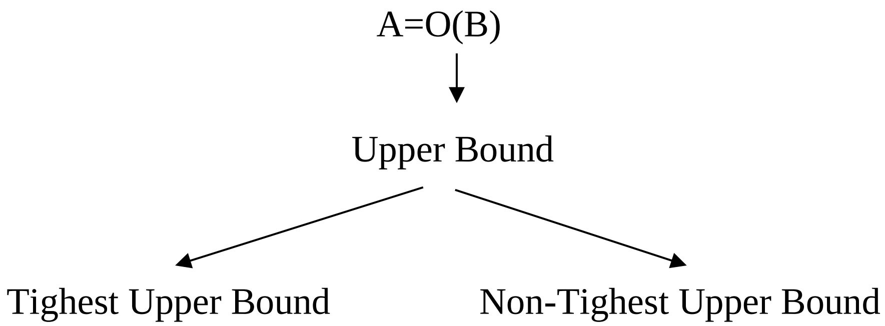
f (n) ⩽ k.g (n)f(n)⩽k.g(n) for n>n0n>n0 in all case
Hence, function g (n) is an upper bound for function f (n), as g (n) grows faster than f (n)

f(n) = 3n+2 & g(n) = n

F(N)=O(G(N))

g(n) can be n^2, n^3….all these functions bound f(n) but always go for least upper bound or tightest upper bound.

Big-Oh Notation (<=) greater or equal		
n^2	$=O(n)$	Not-possible ✗
	$=O(n^2)$	Tighest Upper Bound (Equal)
	$=O(n^3)$	Non-Tighest upper bound Strictly Greater
	$=O(n^{10})$	

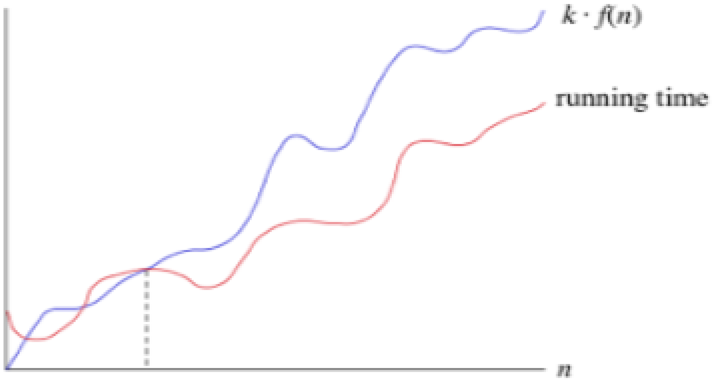


A=O(B)

Upper Bound

Tighest Upper Bound

Non-Tighest Upper Bound



ASYMPTOTIC UPPER BOUND

small-Oh Notation (<) greater than		
n^2	$=O(n^2)$	~~Not-possible~~
	$=O(n^3)$	Non-Tighest upper bound Strictly Greater
	$=O(n^{10})$	

$A=o(B)$

↓

Non-Tighest Upper Bound

Example-1

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \text{----} < 2^n < 3^n < \text{----} < n^n$

$f(n) = 2n + 3$

By the definition of OH: $2n + 3 \leq$ ______ anything > f(n)

$2n + 3 \leq 10n$ say for $n \geq 1$

$5 \leq 10$ True

here, $f(n) = 2n + 3$

$g(n) = n$

$C = 10$

Confusion: Can we right (A) $2n + 3 \le 8n$?????? **Yes!**

(B) $2n + 3 \le$ n^2 ?????? **Yes!**

(C) $2n + 3 \le \log n$?????? **NO!**

Example-2

⦿ $F(n) = 2n^2 + 5n + 1$, find g(n), C and n_o

By the definition of big Oh:

$f(n) = Og(n)$

iff $f(n) \le C.g(n)$ for all $n \ge n_o$

$2n^2 + 5n + 1 \le 4\, g(n)$ say $C = 4$

$2n^2 + 5n + 1 \le 4n^2$ **Highest power of f(n)**

For n =1 : $2 + 5 + 1 \le 4 = 8 \le 4$ **False!**

For n =2 : $2*4 + 5*2 + 1 \le 4*4 = 8 + 10 + 1 \le 16 = 19 \le 16$ **False!**

For n =3 : 2*9 + 5*3 + 1 $\le 4*9 = 18 + 15 + 1 \le 36 = 34 \le 36$ **True!**

WE can say f(n) = O(n^2) for C=4 and n >=3

Example-3

⦿ $F(n) = 2n^2 + 5n + 1$, find g(n), C and n_o

By the definition of big Oh:

$f(n) = Og(n)$

iff $f(n) \le C.g(n)$ for all $n \ge n$

$2n^2 + 5n + 1 \le 4\, g(n)$ say $C = 4$

$2n^2 + 5n + 1 \le 4n^3$

For n =1 : $2 + 5 + 1 \le 4 = 8 \le 4$ **False!**

For n =2 : $2*4 + 5*2 + 1 \le 4*8 = 8 + 10 + 1 \le 32 = 19 \le 32$ **True!**

WE can say f(n) = O(n^3) for C=4 and n >=2

Example-4

◉ $f(n) = 5n^3 + 2n^2 + 10$, find $g(n)$, C and n_o

By the definition of big Oh:

$$f(n) = Og(n)$$

$$\text{iff } f(n) \leq C.g(n) \quad \text{for all } n \geq n_o$$

$5n^3 + 2n^2 + 10 \leq 5n^3 + 2n^2 + 10n$ This is known as variable promotion

$$\leq 5n^3 + 2n^2 + 10n^2$$

$$\leq 5n^3 + 12n^2$$

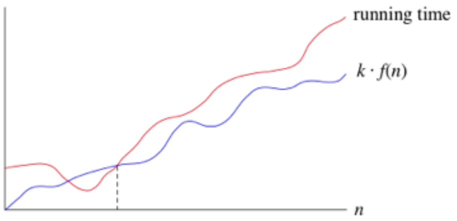
$$\leq 17n^3$$

WE can say $f(n) = O(n^3)$ for $C=17$ and $n >=1$

2. **Omega () Notation:** The function $f(n) = \Omega(g(n))$ [read as "f of n is omega of g of n"] if and only if there exists positive constant c and n_0 such that

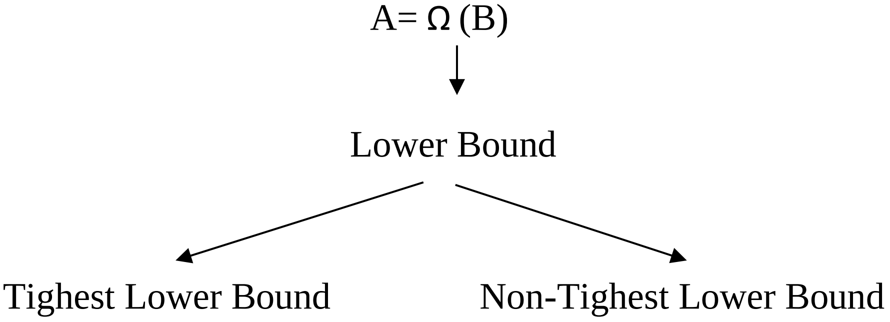
$F(n) \geq k^* g(n)$ for all n, $n \geq n_0$
$f(n) = 3n+2$ & $g(n) = n$

g(n) can be logn, loglogn….all these functions bound f(n) but always go for least upper bound or tightest upper bound.



ASYMPTOTIC LOWER BOUND

Big-Omega Notation (>=) less or equal		
n^3	$= \Omega(n)$	Non-Tighest lower bound Strictly less
	$= \Omega(n^2)$	
	$= \Omega(n^3)$	Tighest lower Bound (Equal)
	$= \Omega(n^4)$	Not-possible ✗



Small-Omega Notation (>) less		
n^3	$= \omega(n)$	Non-Tighest lower bound Strictly less
	$= \omega(n^2)$	
	$= \omega(n^3)$	Not-possible ✗
	$= \omega(n^4)$	

Example-1

◉ $f(n) = 27n^2 + 16n + 25$, find the □- notation

By the definition of big omega:

$$f(n) = □g(n)$$

$$\text{iff } f(n) \geq Cg(n) \text{ for all } C>0, n \geq no$$

Ignore subdominant +ve coefficients

$$27n^2 + 16n + 25 \geq 27n^2$$

For C=27 and $n_o \geq 1$

We can say that $f(n) = □(n^2)$

Example-2

◉ $f(n) = 3^n + 6n^2 + 3n$, find the □- notation

By the definition of big omega:

$$f(n) = □\ g(n)$$

$$\text{iff } f(n) \geq Cg(n) \text{ for all } C>0,\ n \geq no$$

Ignore subdominant +ve coefficients

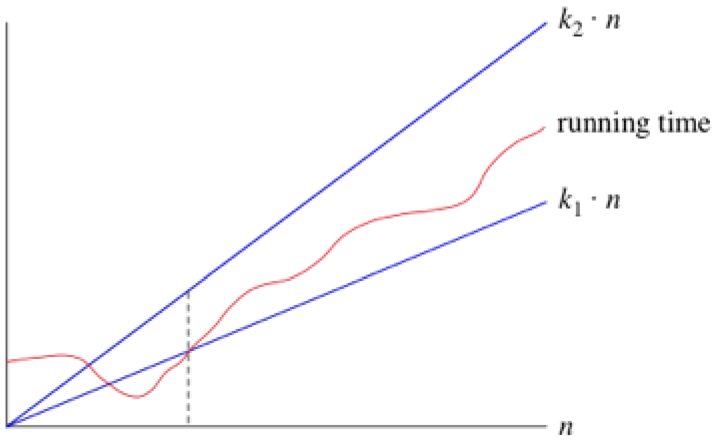
$$3^n + 6n^2 + 3n \geq 3^n$$

For C=1 and $n_o \geq 1$

We can say that $f(n) = □(3^n)$

3. **Theta (θ):** The function f (n) = θ (g (n)) [read as "f is the theta of g of n"] if and only if there exists positive constant k_1, k_2 and k_0 such that

$k_1 * g(n) \leq f(n) \leq k_2\ g(n)$for all n, $n \geq n_0$



ASYMPTOTIC TIGHT BOUND

The Theta Notation is more precise than both the big-oh and Omega notation.

The function f (n) = θ (g (n)) if g(n) is both an upper and lower bound.

$f(n) = 3n+2$ & $g(n) = n^2$

Always the leading term which is n here

$n^3 = O(n^3)$ …………..Tighest Upper Bound

$n^3 = \Omega\ (n^3)$ ……………..Tighest Lower Bound

$n^3 = \theta\ (n^3)$ …………….. Tighest Upper Bound & Tighest Lower Bound

$$A = \theta\ (B)$$

↓

Tighest Upper Bound & Tighest Lower Bound

Example-1

◉ $f(n) = 10n + 5$, find Θ-notation.

By the definition of Theta Notation:

$$f(n) = \Theta(g(n))$$

$$\text{iff } C1g(n) \leq f(n) \leq C2g(n)$$

For $C1g(n) \leq f(n)$:

$$10n \leq 10n + 5$$

Therefore, $f(n) =$ **Ω(n), for** C1=10, n>=1

For $f(n) \leq C2g(n)$:

$$10n + 5 \leq 10n + 5n$$

$$\leq 15n$$

Therefore, $f(n) =$ **O(n), for** C2= 15, n>=1

So we can say that $f(n) = \Theta(n)$

Logarithmic Formulas

1. $\log ab$= $\log a + \log b$

2. $\log \frac{a}{b} = \log a - \log b$

3. $\log a^{b} = b \log a$

4. $a^{\log c^{b}}$ = $b^{\log c^{a}}$

5. $a^{b} = n$ then b = $\log_{a} n$

Comparison of Functions

1. **n^2 n^3**

which is greater ?

Apply log on both sides

Log n^2 log n^3

2*log n 3*log n

Clearly, 3*log n is greater than 2*log n

2. **f(n) = n^2 log n**

g(n) = n $(\log n)^{10}$

which is greater ?

Apply log on both sides

Log (n^2logn)　　　　log[n$(\text{logn})^{10}$]

Log n^2 + loglogn　　logn + log$(\text{logn})^{10}$

2logn + loglogn　>　logn + 10loglogn

Logn is greater than loglogn

Since, logn of LHS is 2 times greater than logn of LHS

3. **f(n) = n ^{logn}**

g(n) = 2 $^{\sqrt{n}}$

Which is greater ?

Apply log on both sides

Logn*logn　　$\sqrt{n}$ *log2

Log^{2}n　　　　$\sqrt{n}$

Again apply log on both sides

2*loglogn < ½*logn

4. **$(n+k)^m$ = θ (n^m) check if it is true or false ?**

Let k=3 & m=2

$(n+3)^2$ = θ (n^2)

This is true.

5. **$2^{n+1} = O(2^n)$ check if it is true or false ?**

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

This is true.

6. **$2^{2n} = O(2^n)$ check if it is true or false ?**

$4^n > 2^n$

THIS IF FALSE

7. **$\sqrt{logn}$ = O(loglogn) check if it is true or false ?**

$\sqrt{logn}$ is greater than loglogn so it cant be big-oh

$This\ is\ false.$

8. **$n^{logn} = O(2^n)$ check if it is true or false ?**

Apply log on both sides

Logn*logn n*log 2

$Log^2 n < n$

This is true.