Asymptotic Notation

- As step count is inconvenient method to compare two or more algorithms .so, we use asymptotic notation.
- Asymptotic notation is the study of how the running time of an algorithm increases with the size of the input in the limit, as the size of the input increases without bound.
- A convenient notation to represent the rate of growth of an algorithm is Asymptotic notation.

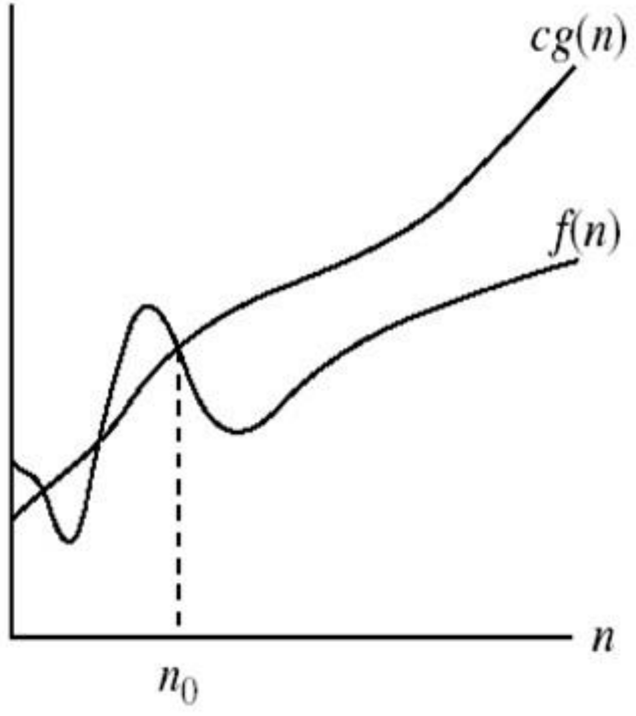
- There are 5 types of Asymptotic notation

  - Big Oh notation (O)
  - Big Omega notation (Ω)
  - Theta notation (θ)
  - Little Oh notation (o)
  - Little omega notation (ω)

Asymptotic notations (cont..)

Big-Oh (O) notation

The function **f(n) = O(g(n))** (read as "f of n is Big oh of g of n") iff (if and only if) there exist positive constants c and n_o Such that **f(n) < =c* g(n)** for all n, n >= n_o



$g(n)$ is an ***asymptotic upper bound*** for $f(n)$.

(or)

We can represent Big-oh as set representation

$$O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0 \} .$$

Example

1. $f(n) = 3n^2+2n$ can be written as $O(n^2)$

Proof : Let $3n^2+2n <= 4n^2$ and $f(n) = 3n^2+2n$ and $c*g(n)= 4*n^2$ find n_0

Let $n_0 = 2$ then $3 * (4) + 2* 2 <= 4*4 => 16 <=16$
$n_0 = 3$ then $3 * (9) + 2* 3 <= 4* 9 => 32 <=36$

i.e., $f(n) <= c*g(n)$ for all $n_0 >=2$. Therefore $3n^2+2n = O(n^2)$

2. $n^4 + 100n^2 + 10n + 50$ is $O(n^4)$

3. $10n^3 + 2n^2$ is $O(n^3)$

4. $n^3 - n^2$ is $O(n^3)$

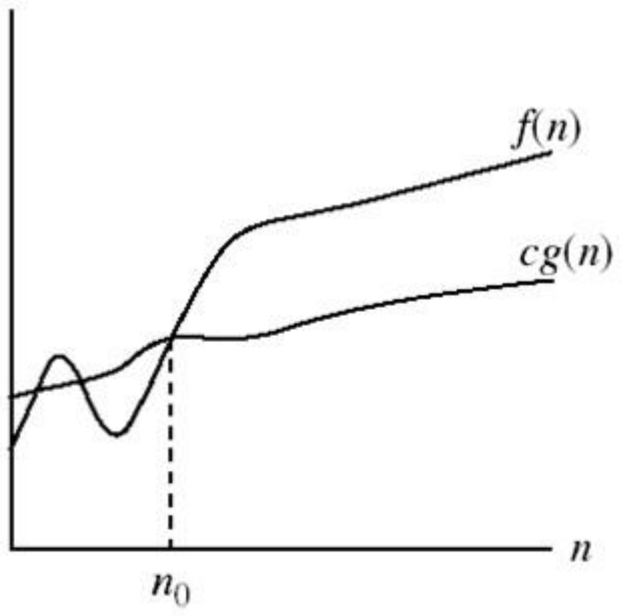
5. constants

- 10 is $O(1)$
- 1273 is $O(1)$

Asymptotic notations (cont..)

Big-Omega (Ω) notation

The function **f(n) = Ω(g(n))** (read as "f of n is omega of g of n") iff (if and only if) there exist positive constants c and n_o Such that **0 < =c* g(n) <=f(n)**for all n, n >= n_o



$g(n)$ is an ***asymptotic lower bound*** for $f(n)$.

(or)

We can represent Big-Omega(Ω)as set representation

$$\Omega(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0 \} .$$

Examples

- $5n^2 = \Omega(n)$ $\exists\, c, n_0$ such that: $0 \le cn \le 5n^2 \Rightarrow cn \le 5n^2 \Rightarrow c = 1$ and $n_0 = 1$

- $100n + 5 \ne \Omega(n^2)$

  $\exists\, c, n_0$ such that: $0 \le cn^2 \le 100n + 5$

  $100n + 5 \le 100n + 5n\ (\forall\ n \ge 1) = 105n$

  $cn^2 \le 105n \Rightarrow n(cn - 105) \le 0$

  Since n is positive $\Rightarrow cn - 105 \le 0 \Rightarrow n \le 105/c$

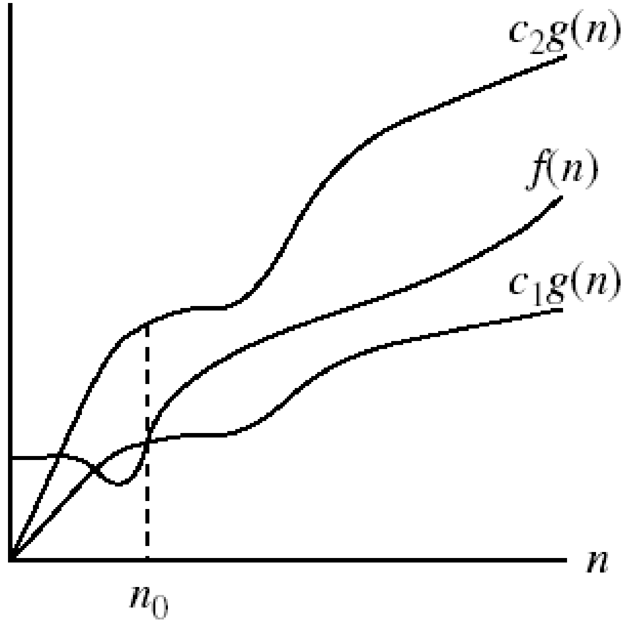
  $\Rightarrow$ contradiction: n cannot be smaller than a constant

- $n = \Omega(2n)$, $n^3 = \Omega(n^2)$, $n = \Omega(\log n)$

Asymptotic notations (cont.)

Theta (Θ) notation

The function **f(n) = Θ(g(n))** (read as "f of n is Theta of g of n") iff (if and only if) there exist positive constants c1,c2, and n_o Such that **0 < =c1* g(n) <=f(n)<=c2*g(n)**for all n, n >= n_o



(or)

We can represent Big- Theta(Θ)as set representation

$$\Theta(g(n)) = \{ f(n) : \text{ there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \} .^1$$

$g(n)$ is an ***asymptotically tight bound*** for $f(n)$.

Small-oh (o) -notation

The asymptotic upper bound provided by O-notation may or may not be asymptotically tight.

The bound $2n^2=O(n^2)$ is asymptotically tight, but the bound $2n=O(n^2)$ is not.

Use o-notation to denote an upper bound that is not asymptotically tight.

$$o(g(n)) = \{f(n) : \text{ for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \le f(n) < cg(n) \text{ for all } n \ge n_0\} .$$

For example, $2n = o(n^2)$, but $2n^2 \ne o(n^2)$.

f(n) becomes insignificant relative to g(n) as n approaches infinity; that is,

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

Small Omega (ω) -notation

By analogy, ω-notation is to Ω-notation as o-notation is to O-notation.

ω-notation is used to denote a lower bound that is not asymptotically tight.

$$\omega(g(n)) = \{f(n) : \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \le cg(n) < f(n) \text{ for all } n \ge n_0\}\ .$$

For example, $n^2/2 = \omega(n)$, but $n^2/2 \ne \omega(n^2)$. The relation $f(n) = \omega(g(n))$ implies that

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty\ ,$$

if the limit exists. That is, $f(n)$ becomes arbitrarily large relative to $g(n)$ as n approaches infinity.

Worst , Best and Average-case efficiencies

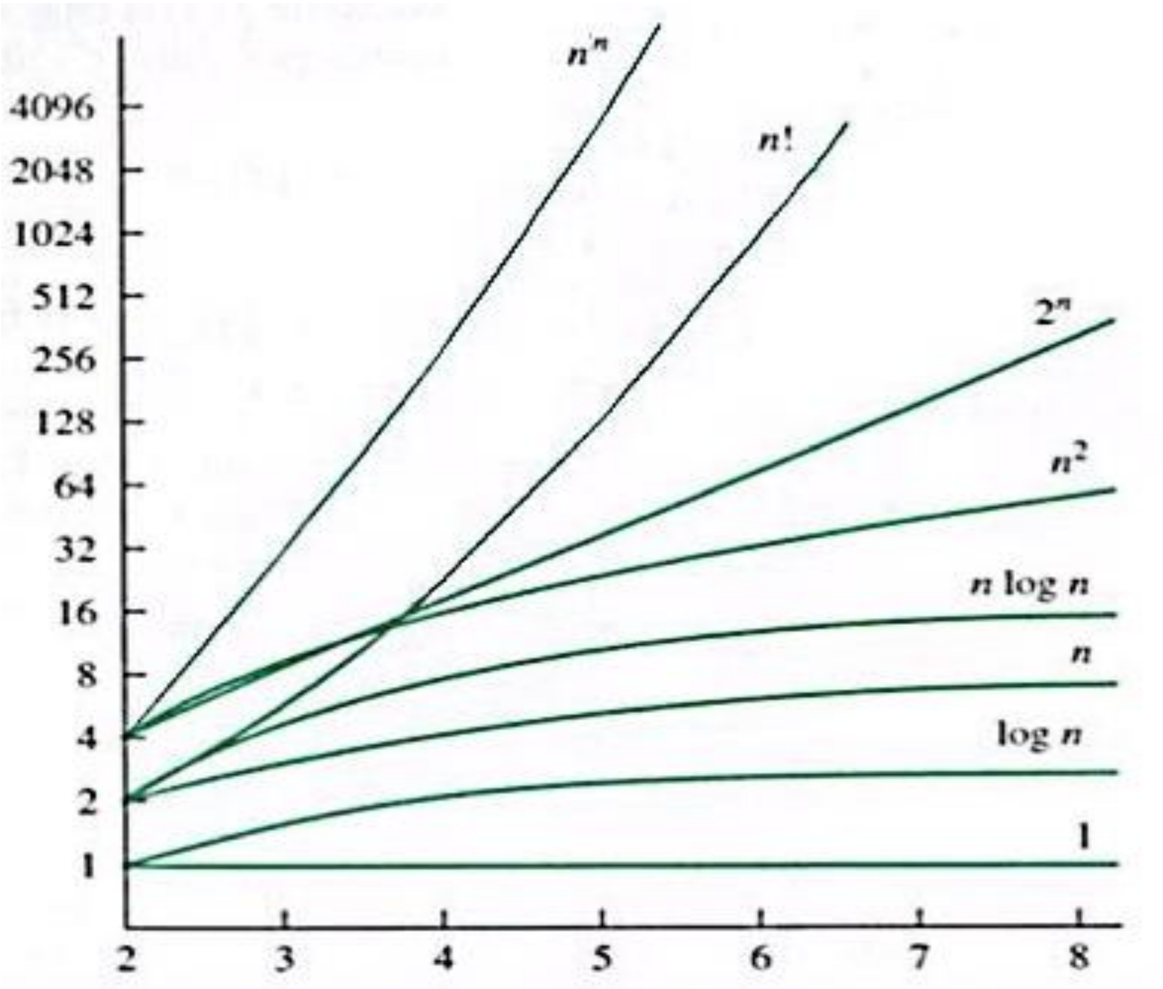
- Worst case
  - Provides a upper bound on running time
  - An absolute guarantee that the algorithm would not run longer, no matter what the inputs are
- Best case
  - Provides a lower bound on running time
  - Input is the one for which the algorithm runs the fastest

$$\textit{Lower Bound} \leq \textit{Running Time} \leq \textit{Upper Bound}$$

- Average case
  - Provides a prediction about the running time
  - Assumes that the input is random

Basic Efficiency Classes

Different types of efficiency classes	
Constant	O(1)
Logarithm	O(log n)
Linear	O(n)
Linear Logarithm	O(n *Log n)
Quadratic	$O(n^2)$
Cubic	$O(n^3)$
Exponential	$O(2^n)$
Factorial	O(n!)
Exponential	$O(n^n)$



Order of growth of the functions

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!) < O(n^n)$$

Properties of Asymptotic notation

1. For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. ■

2. If $f_1(n)$ is in $O(g_1(n))$ and $f_2(n)$ is in $O(g_2(n))$, then $f_1(n) + f_2(n)$ is in $O(\max(g_1(n), g_2(n)))$.

3. If $f_1(n)$ is in $O(g_1(n))$ and $f_2(n)$ is in $O(g_2(n))$, then $f_1(n)f_2(n)$ is in $O(g_1(n)g_2(n))$.

Transitivity:

$$f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n)) \quad \text{imply} \quad f(n) = \Theta(h(n)),$$
$$f(n) = O(g(n)) \text{ and } g(n) = O(h(n)) \quad \text{imply} \quad f(n) = O(h(n)),$$
$$f(n) = \Omega(g(n)) \text{ and } g(n) = \Omega(h(n)) \quad \text{imply} \quad f(n) = \Omega(h(n)),$$
$$f(n) = o(g(n)) \text{ and } g(n) = o(h(n)) \quad \text{imply} \quad f(n) = o(h(n)),$$
$$f(n) = \omega(g(n)) \text{ and } g(n) = \omega(h(n)) \quad \text{imply} \quad f(n) = \omega(h(n)).$$

Contd..

Reflexivity:

$$f(n) = \Theta(f(n)),$$
$$f(n) = O(f(n)),$$
$$f(n) = \Omega(f(n)).$$

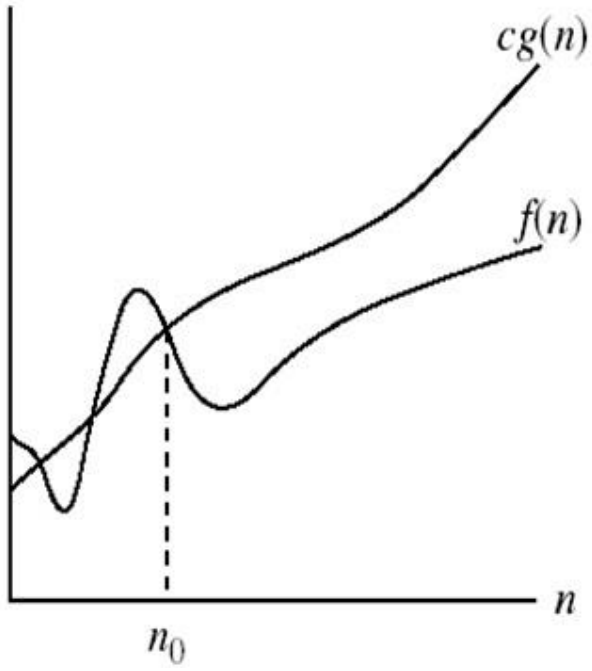
Symmetry:

$$f(n) = \Theta(g(n)) \text{ if and only if } g(n) = \Theta(f(n)).$$

Transpose symmetry:

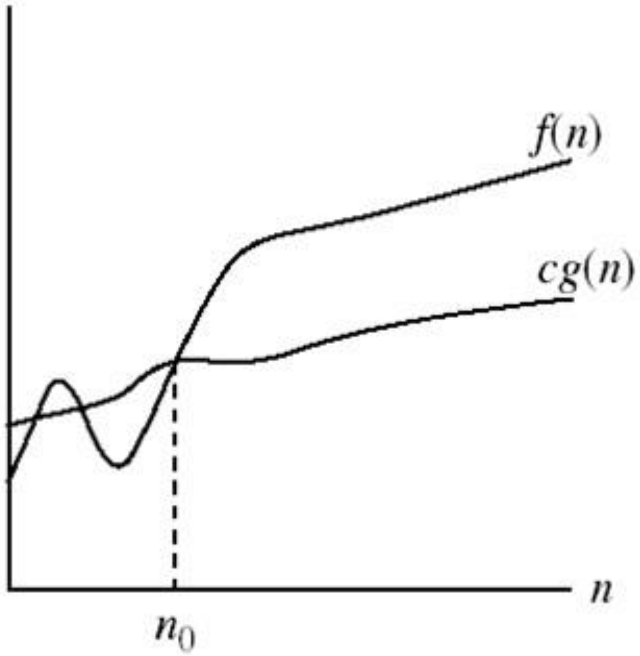
$$f(n) = O(g(n)) \text{ if and only if } g(n) = \Omega(f(n)),$$
$$f(n) = o(g(n)) \text{ if and only if } g(n) = \omega(f(n)).$$

Big-Oh (O) notation



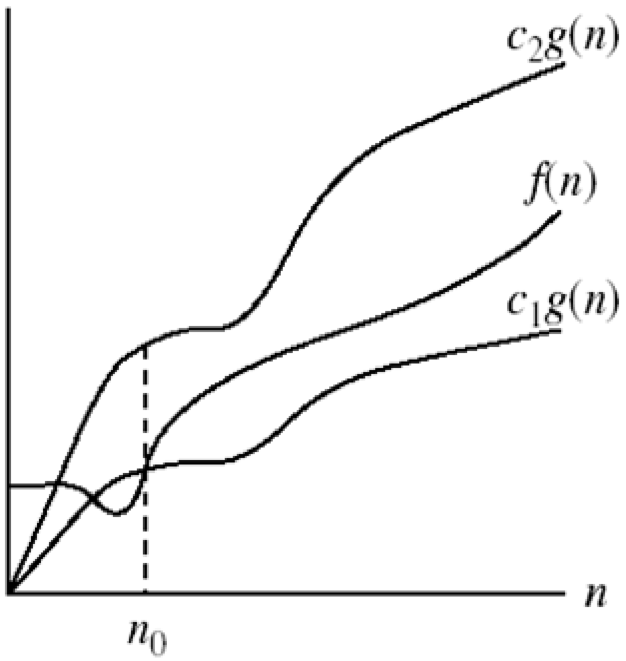
$g(n)$ is an ***asymptotic upper bound*** for $f(n)$.

Big-Omega (Ω) notation



$g(n)$ is an ***asymptotic lower bound*** for $f(n)$.

Theta (Θ) notation



$g(n)$ is an ***asymptotically tight bound*** for $f(n)$.

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}. \tag{2.1}$$

$$\sum_{i=1}^{n} i^2 = \frac{2n^3+3n^2+n}{6} = \frac{n(2n+1)(n+1)}{6}. \tag{2.2}$$

$$\sum_{i=1}^{\log n} n = n \log n. \tag{2.3}$$

$$\sum_{i=0}^{\infty} a^i = \frac{1}{1-a} \text{ for } 0 < a < 1. \tag{2.4}$$

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1} \text{ for } a \neq 1. \tag{2.5}$$

$$\sum_{i=1}^{n} \frac{1}{2^i} = 1 - \frac{1}{2^n}, \tag{2.6}$$

and

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1. \tag{2.7}$$

As a corollary to Equation 2.7,

$$\sum_{i=0}^{\log n} 2^i = 2^{\log n+1} - 1 = 2n - 1. \tag{2.8}$$

Finally,

$$\sum_{i=1}^{n} \frac{i}{2^i} = 2 - \frac{n+2}{2^n}. \tag{2.9}$$

Example 1

```
sum = 0;
for (i=1; i<=n; i++)
        sum += n;
```

Example 2

```
sum = 0;
for (i=1; i<=n; i++)           // First for loop
     for (j=1; j<=i; j++)      // is a double loop
            sum++;
for (k=0; k<n; k++)            // Second for loop
      A[k] = k;
```

Example 3

```
sum1 = 0;
for (k=1; k<=n; k*=2)            // Do log n times
        for (j=1; j<=n; j++)     // Do n times
                sum1++;
sum2 = 0;
for (k=1; k<=n; k*=2)            // Do log n times
        for (j=1; j<=k; j++)     // Do k times
                sum2++;
```

Write the asymptotic notations for the following functions

1. $f(n) = \log n^2$ $g(n) = \log n + 5$
2. $f(n) = sqrt(n)$ $g(n) = \log n^2$
3. $f(n) = (\log n)^2$ $g(n) = \log n$
4. $f(n) = n$ $g(n) = \log^2 n$
5. $f(n) = n \log n + n$ $g(n) = \log n$
6. $f(n) = \log n^2$ $g(n) = (\log n)2$
7. $f(n) = 2^n$ $g(n) = 10 n^2$
8. $f(n) = 2^n$ $g(n) = n \log n$
9. $f(n) = 2^n$ $g(n) = 3^n$
10. $f(n) = 2^n$ $g(n) = n^n$