ASYMPTOTIC NOTATIONS

Outline of Lecture 3:

- ✔ **Formal Definition** of Individual Asymptotic Notation
- ✔ Geometrical Interpretation of each Asymptotic Notation
- ✔ Examples to understand the corresponding notation
- ✔ Properties of Big Oh Asymptotic Notation
- ✔ Calculation of Big Oh of an Algorithm

Contents

- What is an asymptote?
- What do you mean by the run time $T_A(n)$ of an algorithm?
- Why we need to know asymptotic behavior for analyzing an algorithm?
- What are the different types of asymptotic notations?
- Big Oh: An Asymptotic Upper Bound
  - ✔ Definition
  - ✔ Geometrical Interpretation
  - ✔ Examples
  - ✔ Properties
  - ✔ How Big Oh help us to calculate the time complexity of an algorithm?

Contents (Contd...)

- Small oh: An Asymptotic Loose upper bound
  - ✔ Definition
  - ✔ Geometrical Interpretation
  - ✔ Examples
- Big -Omega: An Asymptotic Lower Bound
  - ✔ Definition
  - ✔ Geometrical Interpretation
  - ✔ Examples
- Small -omega: An Asymptotic lose Lower Bound
  - ✔ Definition
  - ✔ Geometrical Interpretation
  - ✔ Examples
- Theta Notation
  - ✔ Definition
  - ✔ Geometrical Interpretation
  - ✔ Examples

What is an asymptote?

- ✓ Definition
- ✓ Consider a curve say $y = 1 / x$

x	1	2	3	4	5	……	100	1000	10000
y	1	0.5	0.33	0.25	0.2	…….	0.01	0.001	0.0001

x=1

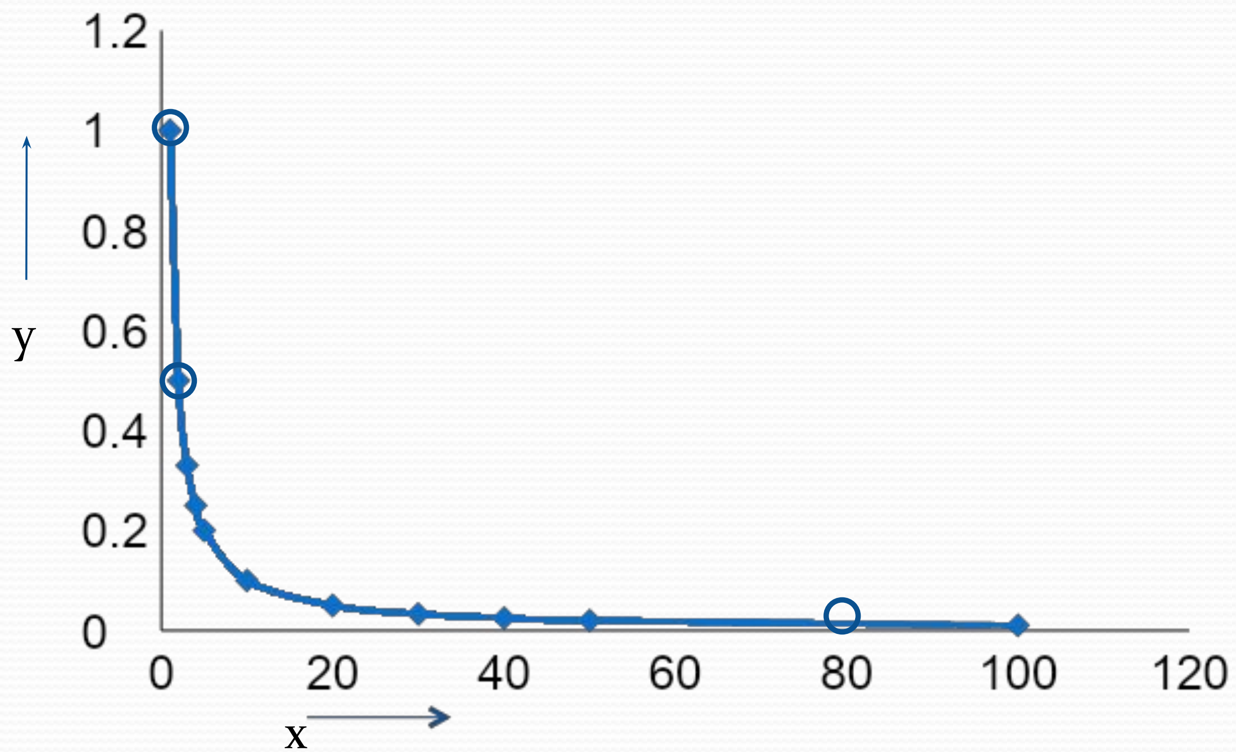
y=1

x=2

y=0.5

x=100

y=0.01



Calculation of Run time $T_A(n)$ of an Algorithm

Selection SelectionSort(A)

```
For I = 1 to n-1 do:
        Smallsub = I
        For J = I + 1 to n do:
            If A(J) < A(Smallsub)
                Smallsub = J
            End-If
        End-For
        Temp = A(I)
        A(I) = A(Smallsub)
        A(Smallsub) = Temp
End-For
```

(n-i) times

3

(n-1) times

$$T_A(n) = \sum\{2 * (n-i) \ + 3\} \ where \ i = 1,2, \ldots\ldots, n-1$$

$$T_A(n) = (\,(2*(n-1)+3) + (2*(n-2)+3) + (2*(n-3)+3) + \cdots \ldots\ldots + (2*1+3))$$

$$= (2*((n-1)*n/2) + 3*(n-1)) = n*(n-1) + 3*(n-1) = \boldsymbol{n^2 + 2n - 3}.$$

Calculation of Run time $T_A(n)$ of an Algorithm: Assumption involved

- All operations have been considered as taking same time.
- Time taken by a particular instruction depends on the architecture.
- For example, multiplication operation takes longer than addition operation.

Why do we need to know asymptotic behavior for analyzing an algorithm?

A given problem

Algorithm A → $T_A(n)$

Algorithm B → $T_B(n)$

- Compare the two functions and determine which algorithm is *the best*.
- $T_A(n_o) < T_B(n_o)$ *where* n_o *denotes the known problem size.*
- *No prior* knowledge of the problem size.

  $T_A(n) <= T_B(n)$, *for all* $n > 0$
- One of the functions is less than or equal to the other over the entire range of problem sizes.

✓ $T_A(n) = 2 * n + 20, T_B(n) = n^2$

✓ *Can we conclude whether* $T_A(n) <= T_B(n) or\ T_A(n) > T_B(n)$?

✓ $T_A(n_0) = 32, T_B(n_0) = 36\ where\ n_0 = 6$

✓ $T_A(n) < T_B(n)\ \forall n \geq 6$

✓ $T_A(n) > T_B\ (n)\ \forall n \leq 5$

✓ So how do we compare these two functions $T_A(n)$ and $T_B(n)$

✓ Algorithm A is better than algorithm B (As we think that the problem input size n as large as possible)

What are the different types of asymptotic notations?

- ✔Big Oh (O)
- ✔ Small oh (o)
- ✔ Big Omega (Ω)
- ✔ Small omega (ω)
- ✔ Theta(Θ)

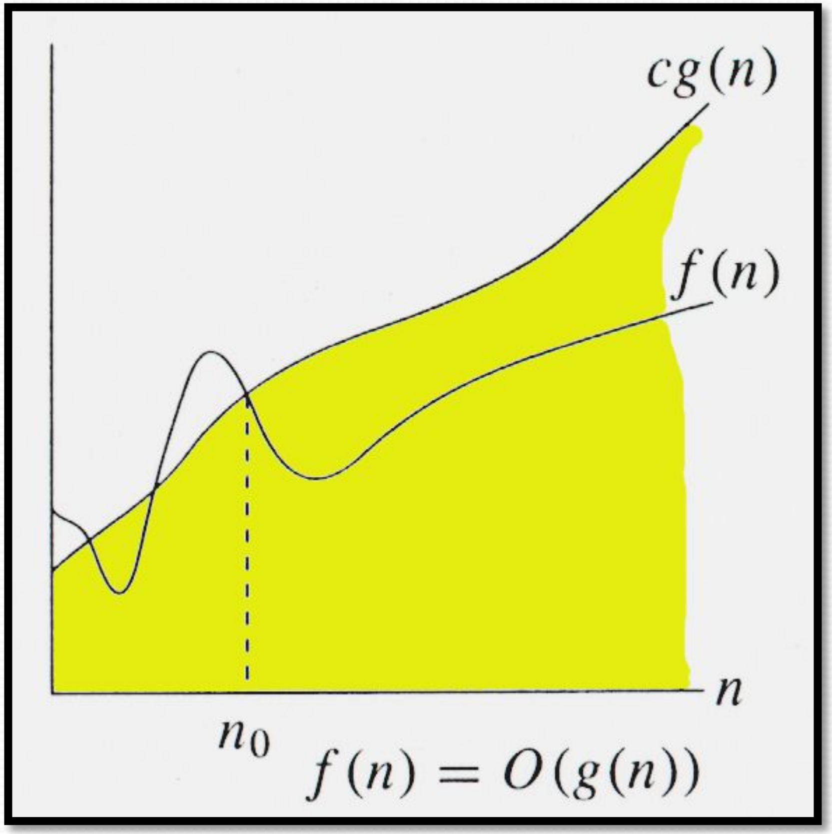
Big Oh Notation

$O(g(n)) = \{f(n) :$
$\exists$ positive constants c and positive integer n_0, such that $\forall n \geq n_0$, we have $0 \leq f(n) \leq cg(n)$ $\}$

- ✔ We write $f(n)=O(g(n))$.
- ✔ $g(n)$ is an ***asymptotic upper bound*** for $f(n)$.
- ✔ $f(n)=O(g(n))$ iff

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

where l is finite.



Example of Asymptotic Upper Bound

$f(n)=3n^2+5$

$4\,g(n) = 4n^2$

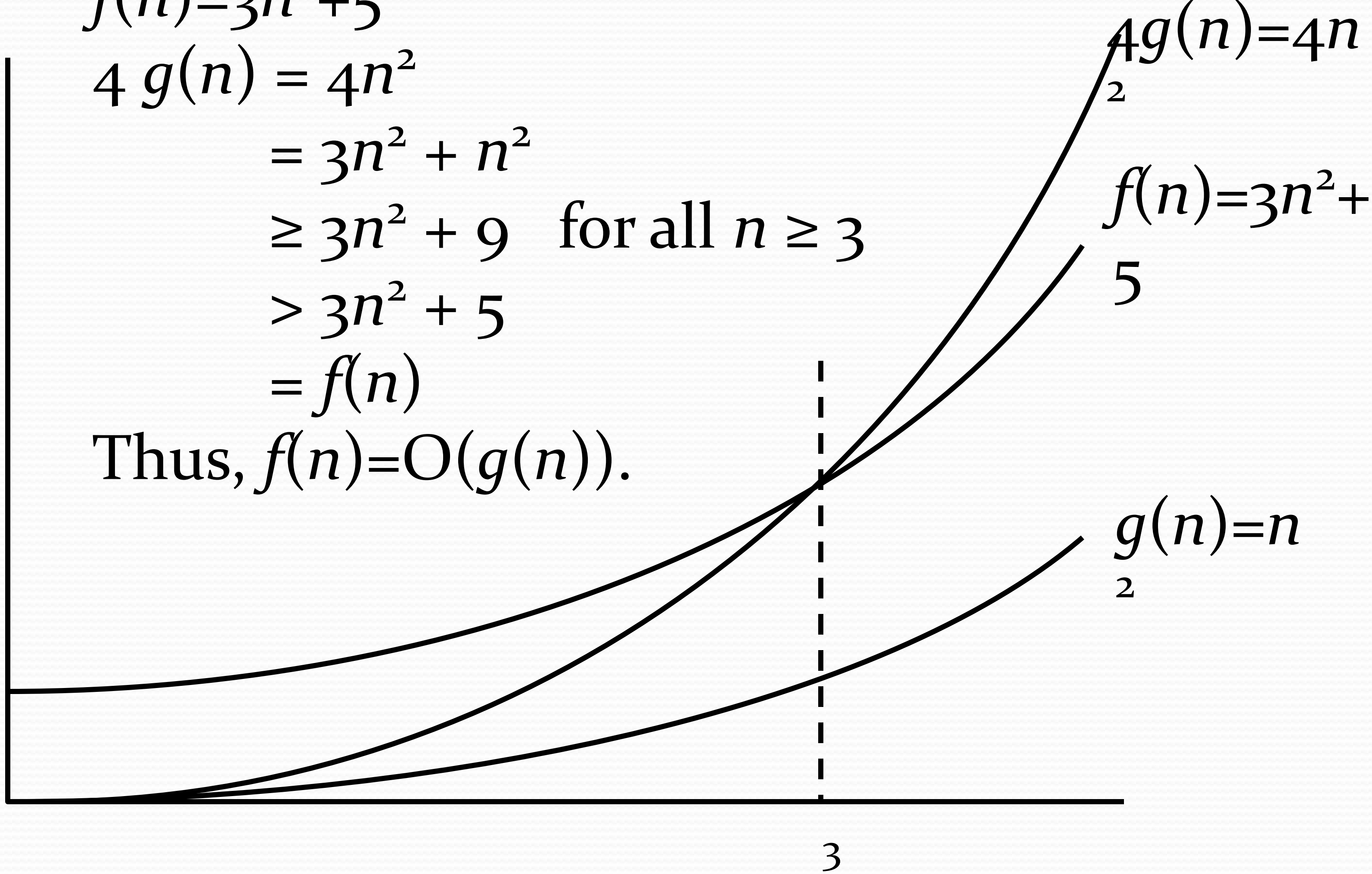
$= 3n^2 + n^2$

$\geq 3n^2 + 9$ for all $n \geq 3$

$> 3n^2 + 5$

$= f(n)$

Thus, $f(n)=O(g(n))$.



Loose and Tight Upper Bounds

The upper bound given by Big Oh may be lose or tight upper bound:

$$f(n) = 3n^2 + 5$$

$g(n) = n^2$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 3$$

Tight Upper bound

$g(n) = n^4$

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

Loose Upper bound

Properties of Big Oh

Property 1

If $f(n) = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a_0$, Then $f(n) = O(n^m)$

Proof: $f(n) = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a_0$

$$\Rightarrow f(n) \le \sum_{i=0}^{m} |a_i| n^i \text{ [since } |x+y| \le |x| + |y| \text{ and } |xy| = |x||y| \text{]}$$

$$\Rightarrow f(n) \le \sum_{i=0}^{m} |a_i| n^m \quad [\text{ Since } n^i \le n^m \text{ as } i = 0,1,2,\ldots.m]$$

$$\Rightarrow f(n) \le n^m \sum_{i=0}^{m} |a_i|$$

$$\Rightarrow f(n) \le c n^m \,\forall\, n \ge 1 \text{ where } c = \sum_{i=0}^{m} |a_i| + 1$$

$$\Rightarrow f(n) = O(n^m)$$

So, the Big Oh of a polynomial function is the highest power of the variable in the function.

Properties of Big Oh (contd.)

Property 2

$$If\ f(n) = c_1 g(n), then\ f(n) = O(g(n))$$

Proof: $f(n) = c_1 g(n)$

$$=> |f(n)| = |c_1 g(n)|$$

$$=> |f(n)| \leq c\,|g(n)|\ \ where\ n \geq 1\ and\ c = (|c_1| + 1)$$

$$=> f(n) = O(g(n))$$

Properties of Big Oh(contd.)

Property 3

$$\begin{gathered} If\ f(n) = f_1(n) + f_2(n) \\ and\ f_1(n) \leq f_2(n)\ for\ all\ values\ of\ n \\ Then\ f(n) = O(f_2(n)) \end{gathered}$$

Proof: $f(n) = f_1(n) + f_2(n)$

$$\Rightarrow f(n) \leq f_2(n) + f_2(n) \quad \forall n$$

$$\Rightarrow f(n) \leq 2f_2(n) \quad \forall n$$

$$\Rightarrow f(n) = O(f_2(n)) \quad \forall n$$

So, we only consider the fastest growing function out of all the functions making up function $f(n)$ in order to find the Big Oh of function $f(n)$.

Properties of Big Oh(contd.)

Property 4

Prove that $O(log_a n) = O(log_b n)$.
Or
Prove that all *log* functions grows in the same fashion in terms of Big Oh.

$$\boldsymbol{Proof}: O(log_a n) = O(log_a b.log_b n) = O(log_b n)$$
$$as\ c = log_a b\ is\ a\ positive\ constant.$$

Example of Calculation of Big Oh using Properties

Sample Code:

Block 1:

$c_1 n \log n$ $\left\{ \begin{array}{l} \{........ \\ \\ ....\} \end{array} \right.$ .....

Block 2:

c_2n^2 $\left\{ \begin{array}{l} \{........ \\ ..... \\ ....\} \end{array} \right.$

$T(n) = c_1 n \log n + c_2 n^2$

Solution:

Considering only the fastest growing term, we get:

$T(n) = c_1 n \log n + c_2 n^2$

$T(n) \leq c_1 n^2 + c_2 n^2$ [since, $n^2 \geq n \log n \;\forall\, n$]

$T(n) \leq cn^2$ where $c = |c_1| + |c_2| + 1$

So, $T(n) = O(n^2)$

Example of Calculation of Big Oh using Properties

Sample Code:

$for(\ i = 1;\ i <= n; i{+}{+})$
$\{$
$\dots .\ c_1\ instructions$
$for(j = 1;\ j <= n;\ j{+}{+})$
$\{$
$\dots . .\ c_2\ instructions$
$for(\ k = 1;\ k <= n;\ k{+}{+})$
$\{$
$\dots . .\ c_3\ instructions$
$\}$
$\}$
$\}$
$\dots \dots .\ c_4\ instructions$
$for(i = 1; i <= n; i{+}{+})$
$\{\dots .\ c_5\ instructions$
$for(j = 1; j \le n; j{+}{+})$
$\{\dots . .\ c_6\ instructions$
$\}$
$\}$

$$T(n) = ((c_3 n + c_2)n + c_1)n + c_4 + (c_6 n + c_5)n$$
$$= c_3 n^3 + c_2 n^2 + c_1 n + c_4 + c_6 n^2 + c_5 n$$
$$= c_3 n^3 + (c_2 + c_6)n^2 + (c_1 + c_5)\,n + c_4$$

Method 1:

Considering only the fastest growing term, we get $T(n) = O(c_3 n^3)$ [using property 3]

So, $T(n) = O(n^3)$ [using property 2]

Method 2:

Considering only the term having highest power of the variable in the polynomial, we get:

$T(n) = O(n^3)$ using property 1

Small Oh: An Asymptotic Loose Upper Bound

$o(g(n)) =$
$\{f(n) : \forall c > o\ \exists n_o > o$ such that $\forall n \geq n_o$
$o \leq f(n) < cg(n)\}$

- We write $\boldsymbol{f(n)=o(g(n))}$
- Small- oh notation is used to denote an upper bound that is not asymptotically tight.
- $f(n)=o(g(n))$ iff

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

Differences between Big Oh and Small oh

Big Oh

- $O(g(n)) = \{f(n) :$ $\exists$ positive constants c and $n_{0,}$ such that $\forall n \geq n_0$,

  $0 \leq f(n) \leq cg(n)$ }
- The bound $0 \leq f(n) \leq cg(n)$ holds for some value of constant c>0

Small oh

- $o(g(n)) =$ $\{f(n) : \forall c > 0\ \exists n_0 > 0$ such that $\forall n \geq n_0$,

  $0 \leq f(n) < cg(n)\}$
- The bound $0 \leq f(n) < cg(n)$ some holds for all value of constants c>0

Example of Small-oh

$$2n = o(n^2)$$

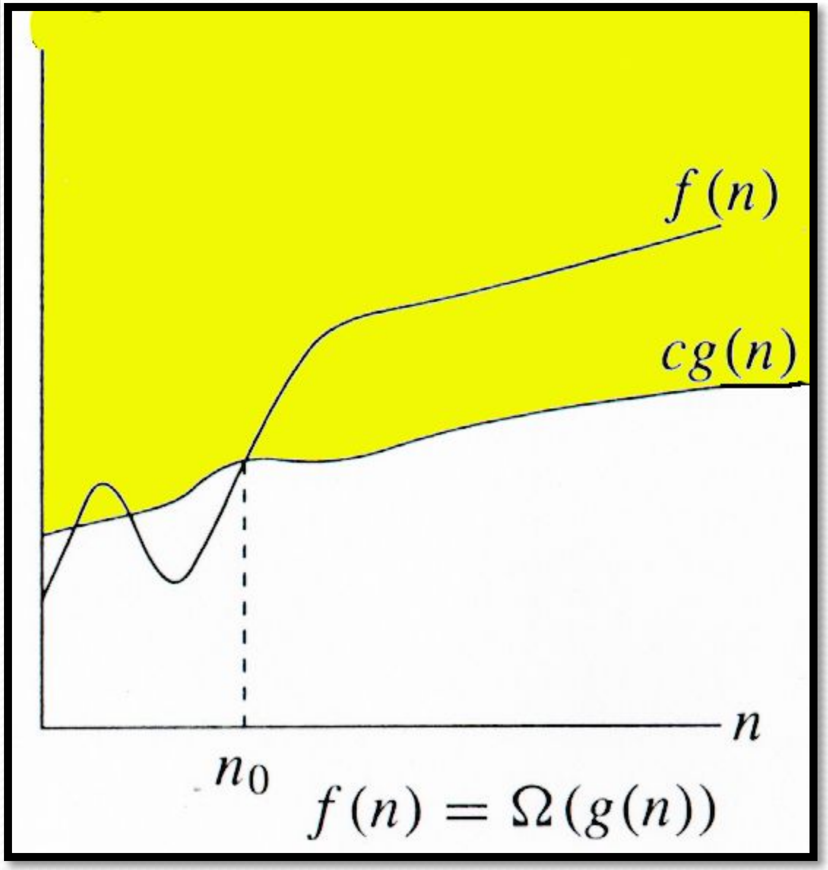
Since, for all values of $c>0$, $cn^2>2n$

$$2n^2 \neq o(n^2)$$

Since, for $c=1$, $cn^2<2n$

Big Omega: An Asymptotic Lower Bound

$\Omega(g(n)) = \{f(n)$:
$\exists$ positive constants c and n_0,
such that $\forall n \geq n_0$,
we have $0 \leq cg(n) \leq f(n)\}$



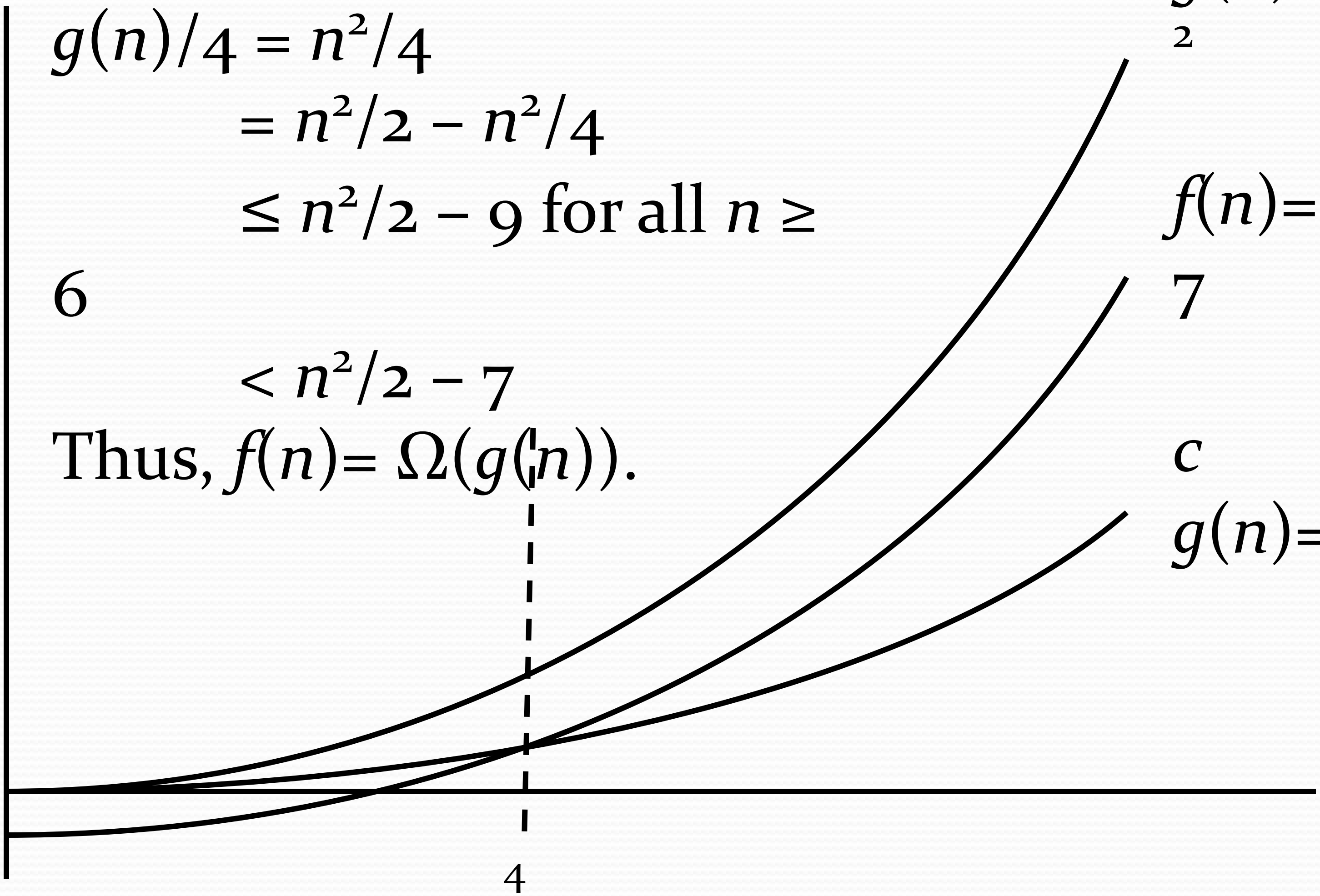
$g(n)$ is an ***asymptotic lower bound*** for $f(n)$.

We write $f(n)=\Omega(g(n))$

Example of Asymptotic Lower Bound (Big Omega)

$$
\begin{aligned}
g(n)/4 &= n^2/4 \\
&= n^2/2 - n^2/4 \\
&\le n^2/2 - 9 \text{ for all } n \ge 6 \\
&< n^2/2 - 7
\end{aligned}
$$

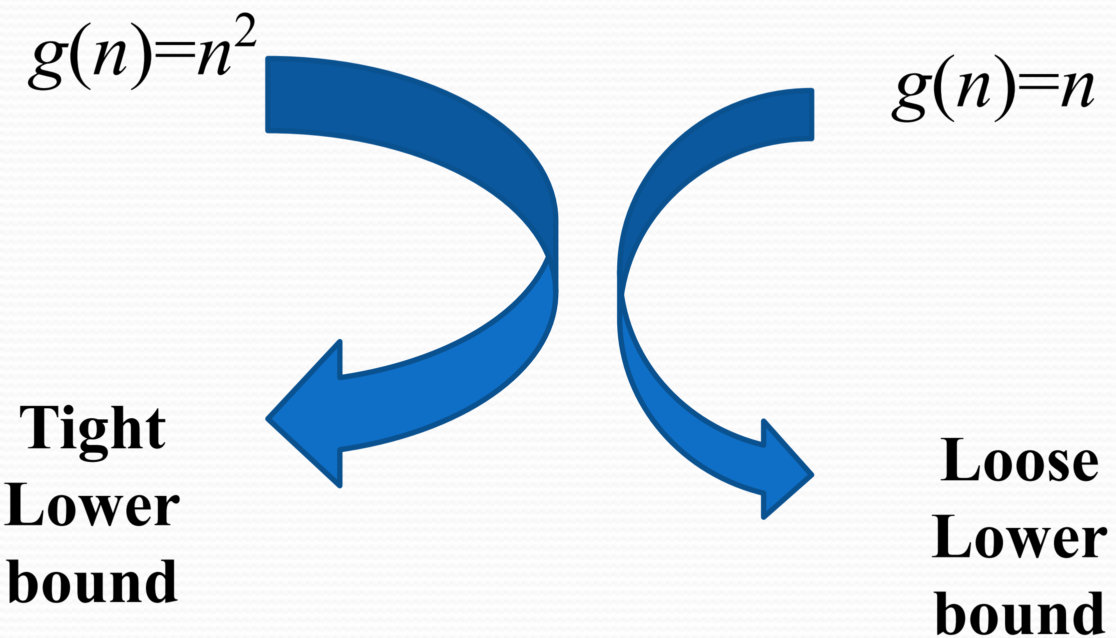
Thus, $f(n) = \Omega(g(n))$.



Loose and Tight Lower Bounds

The lower bound given by Big Oh may be lose or tight lower bound:

$f(n)=n^2/2$



Small Omega: An Asymptotic Loose Lower Bound

$\omega(\boldsymbol{g}(\boldsymbol{n})) = \{ f(n)\colon \forall\ \boldsymbol{c} > \mathbf{0},\ \exists\ \boldsymbol{n}_{\mathbf{0}} > \mathbf{0}$ such that $\forall\ n \geq n_0$, we have $0 \leq cg(n) < f(n)\}$.

- We write $\boldsymbol{f}(\boldsymbol{n}) = \omega\ (\boldsymbol{g}(\boldsymbol{n}))$
- Small- omega notation is used to denote a lower bound that is not asymptotically tight.
- $f(n) = \omega\ (g(n))$ iff

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

Differences between Big Omega and Small Omega

Big Omega

- $\Omega(g(n)) = \{f(n) :$ $\exists$ positive constants c and n_0, such that $\forall n \geq n_0$, we have $0 \leq cg(n) \leq f(n)\}$
- The bound $0 \leq cg(n) \leq f(n)$ holds for some value of constant c>0.

Small omega

- $\omega(g(n)) = \{f(n)\text{: } \forall\ c > 0,\ \exists\ n_0 > 0$ such that $\forall\ n \geq n_0$, we have $0 \leq cg(n) < f(n)\}$.
- The bound $0 \leq cg(n) < f(n)$ some holds for all value of constants c>0.

Example of Small Omega

$$n^2/2 = \omega(n)$$

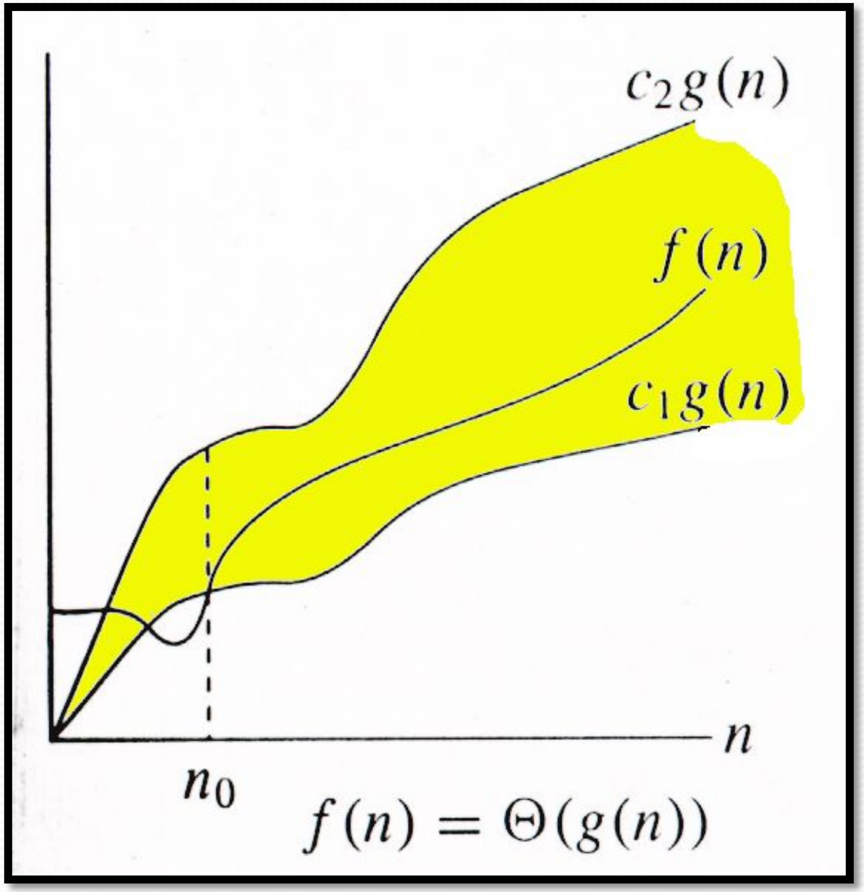
Since, for all values of c>0 $(n^2/2)>cn$

$$n^2/2 \neq \omega(n^2)$$

Since, for c>1/2, $(n^2/2)<cn$

Theta Notation

$\Theta(g(n)) = \{f(n) :$
$\exists$ positive constants c_1, c_2, and n_0, such that $\forall n \geq n_0$,
we have $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$
$\}$



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

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

Example of Theta

Let $\mathrm{f}(n) = \frac{1}{2}n^2 - 3n$

We must find c_1, c_2, n_0 such that:

$$c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2, \forall\, n \ge n_0$$

For right hand inequality:

$$n \ge 1, c_2 = 1/2$$

For left hand inequality:

$$\text{So, taking } n \ge 7, c_1 = 1/14$$

$$n_0 = 7 = \max\{n_1, n_2\} = \{1,7\}$$

$$c_1 = 1/14,$$

$$c_2 = 1/2$$

Hence, $c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2, \forall\, n \ge n_0$

$$\mathrm{f}(n) = \theta(n^2)$$