Design and Analysis of Algorithms

Muhammad Waqas Sheikh

Asymptotic Notation

- Running time of an algorithm as a function of input size *n* **for large *n***.
- Expressed using only the **highest-order term** in the expression for the exact running time.
- Describes behavior of function in the limit.
- Written using ***Asymptotic Notation.***

Asymptotic Notation

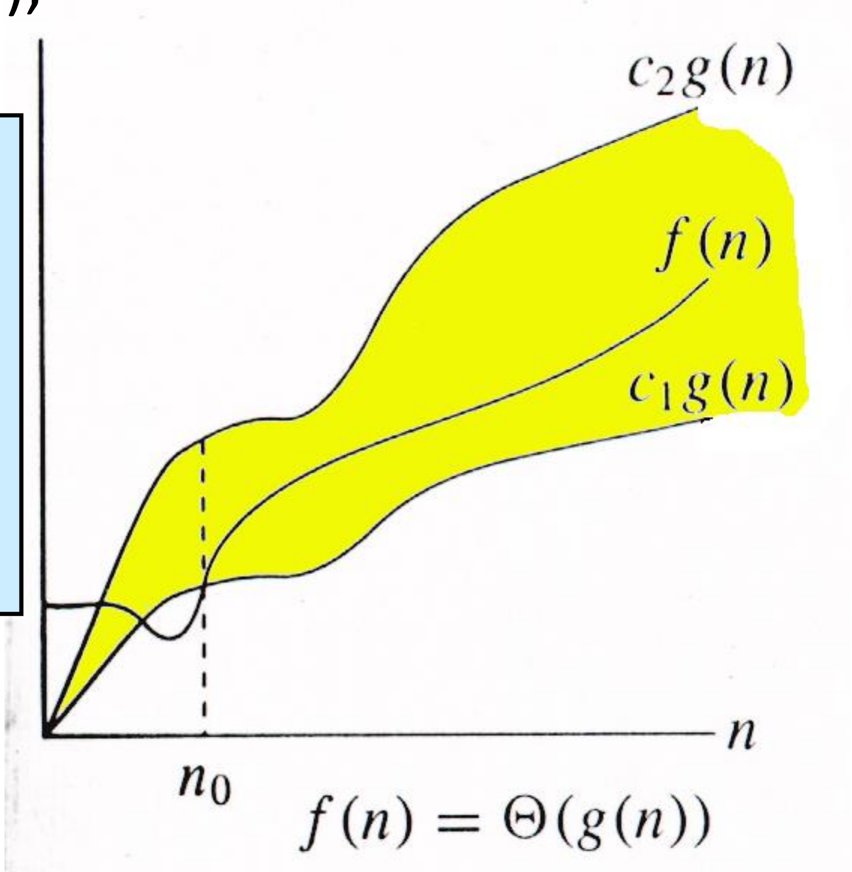
- Put Functions in different classes and think of entire class
- Formal +Systematic
- Asymptotic notation is formal way to speak about functions and their classes
- Asymptotic Analysis is about classification of functions.

- $10n^3 + 5n^2 + 7$ (Cubic)
- $3n^3 + 2n^2 + 5$ (Cubic)
- Constant multiplier should be ignored
- Give more importance to behavior as $n \to \infty$

Θ-notation

For function $g(n)$, we define $\Theta(g(n))$, big-Theta of n, as the set:

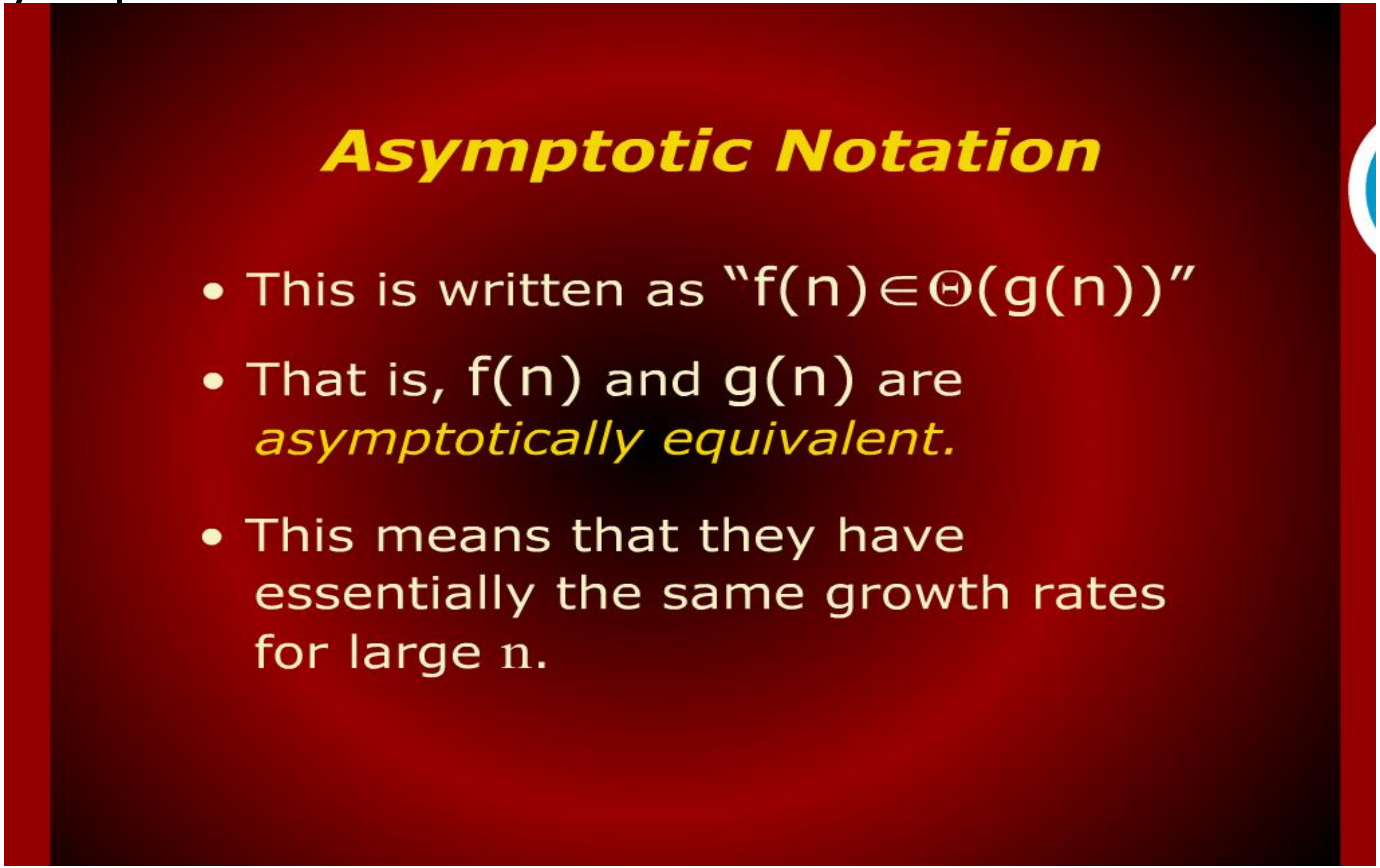
$\Theta(g(n)) = \{f(n)$:
$\exists$ **positive constants c_1, c_2, and n_0,**
such that $\forall n \geq n_0$,
we have $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$
$\}$



Intuitively: Set of all functions that have the same *rate of growth* as $g(n)$.

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

Asymptotic Notation



Asymptotic Notation - Example

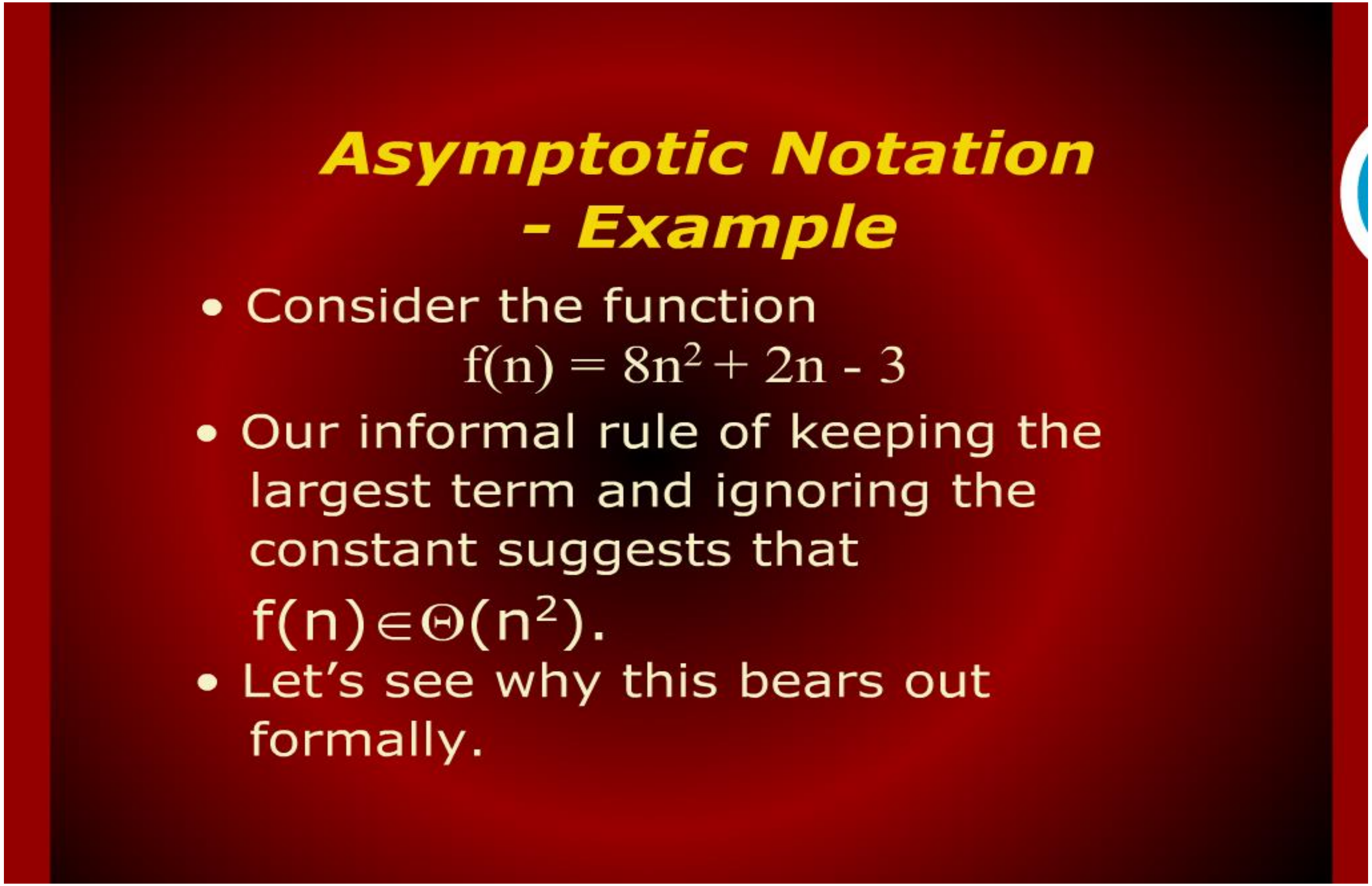
Asymptotic Notation

For example, functions like

- $4n^2$,
- $(8n^2 + 2n - 3)$,
- $(n^2/5 + \sqrt{n} - 10 \log n)$
- $n(n - 3)$

are all asymptotically equivalent. As n becomes large, the dominant (fastest growing) term is some constant times n^2.

Asymptotic Notation - Example



Asymptotic Notation - Example

Asymptotic Notation - Example

We need to show two things for

$$f(n) = 8n^2 + 2n - 3$$

- **Lower bound**: $f(n) = 8n^2 + 2n - 3$ grows asymptotically at least as fast as n^2,
- **Upper bound**: $f(n)$ grows no faster asymptotically than n^2,

Asymptotic Notation - Example

$\leq c_1 g(n) \leq f(n) \leq c_2 g(n$

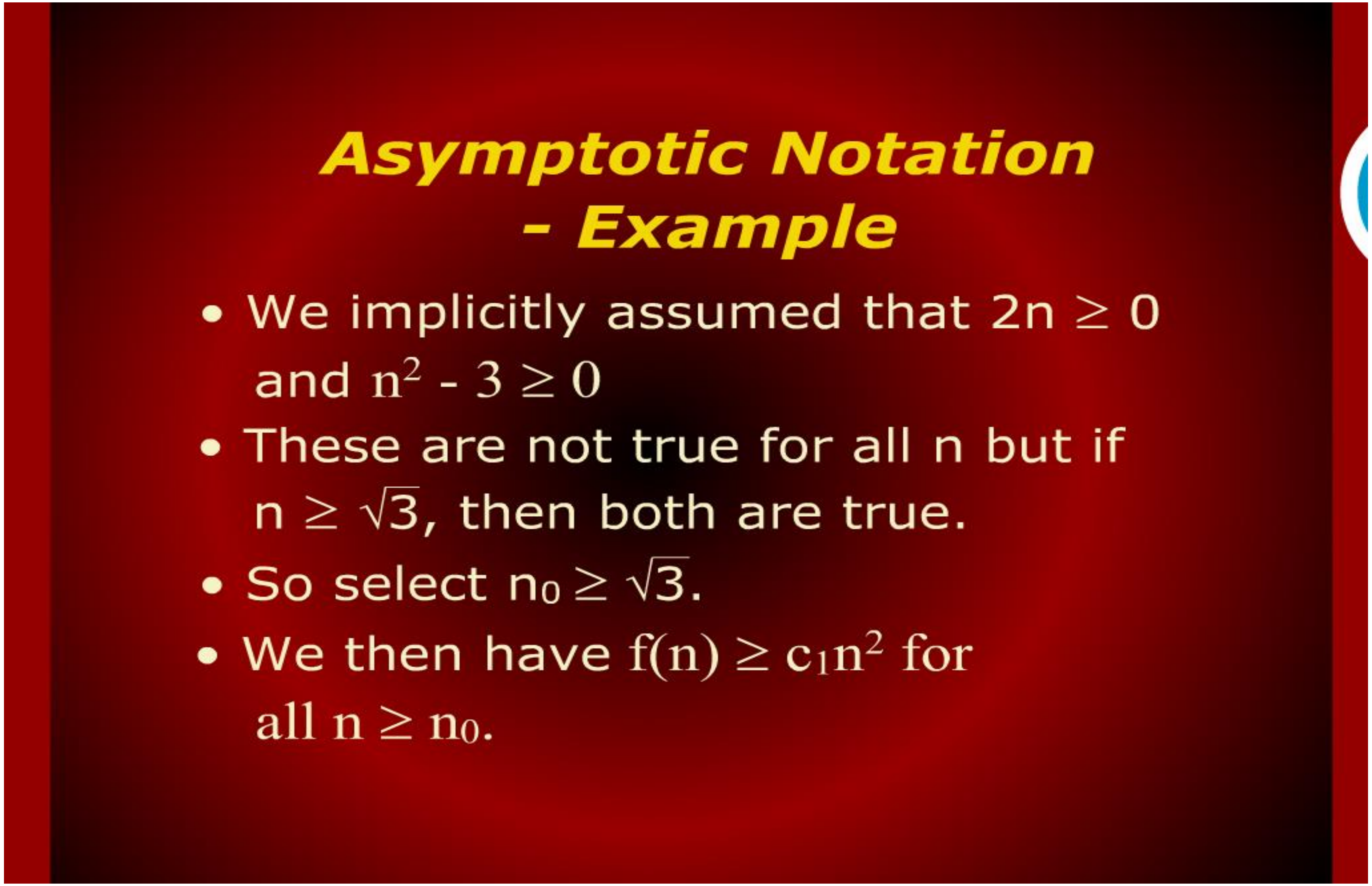
Asymptotic Notation - Example

Lower bound: f(n) grows asymptotically at least as fast as n^2.

- For this, need to show that there exist positive constants c_1 and n_0, such that $f(n) \geq c_1 n^2$ for all $n \geq n_0$.
- Consider the reasoning

$$f(n) = 8n^2+2n-3 \geq 8n^2-3$$
$$= 7n^2+(n^2-3) \geq 7n^2$$

Asymptotic Notation - Example



Asymptotic Notation - Example

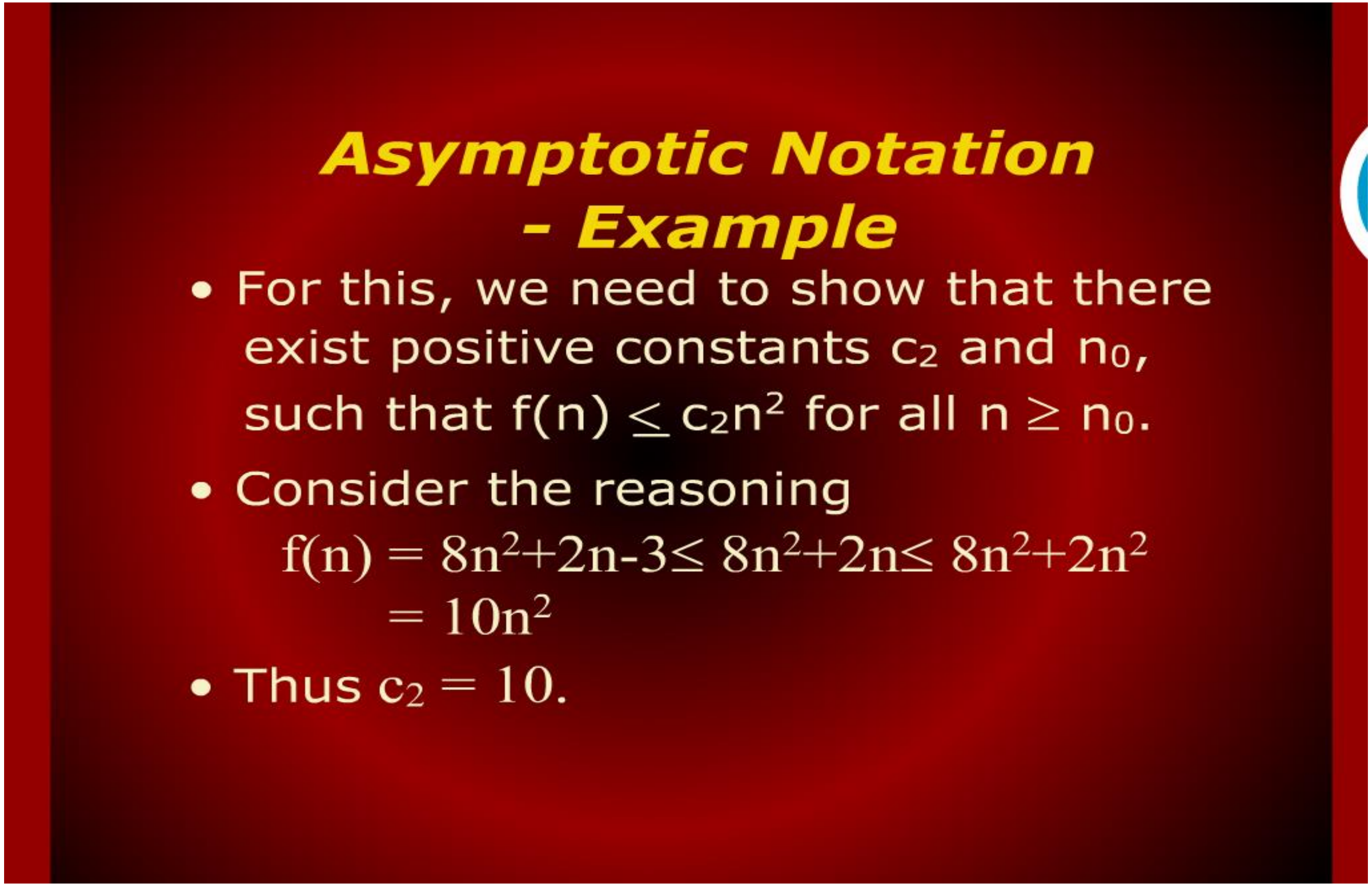
Asymptotic Notation - Example

Upper bound: $f(n)$ grows asymptotically no faster than n^2.

- For this, we need to show that there exist positive constants c_2 and n_0, such that $f(n) \leq c_2 n^2$ for all $n \geq n_0$.
- Consider the reasoning

$$f(n) = 8n^2+2n-3 \leq 8n^2+2n \leq 8n^2+2n^2$$
$$= 10n^2$$

Asymptotic Notation - Example



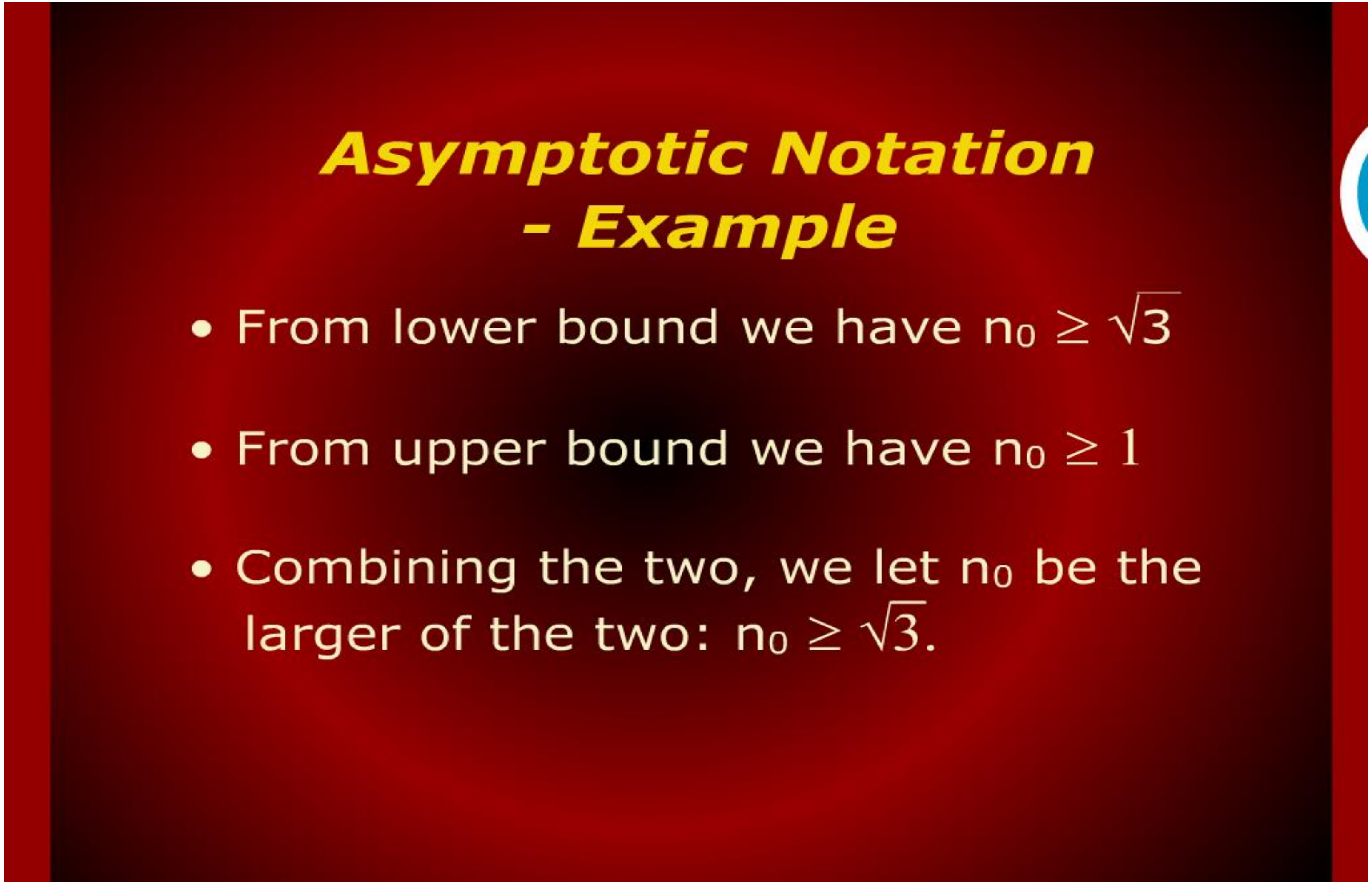
Asymptotic Notation - Example

Asymptotic Notation - Example

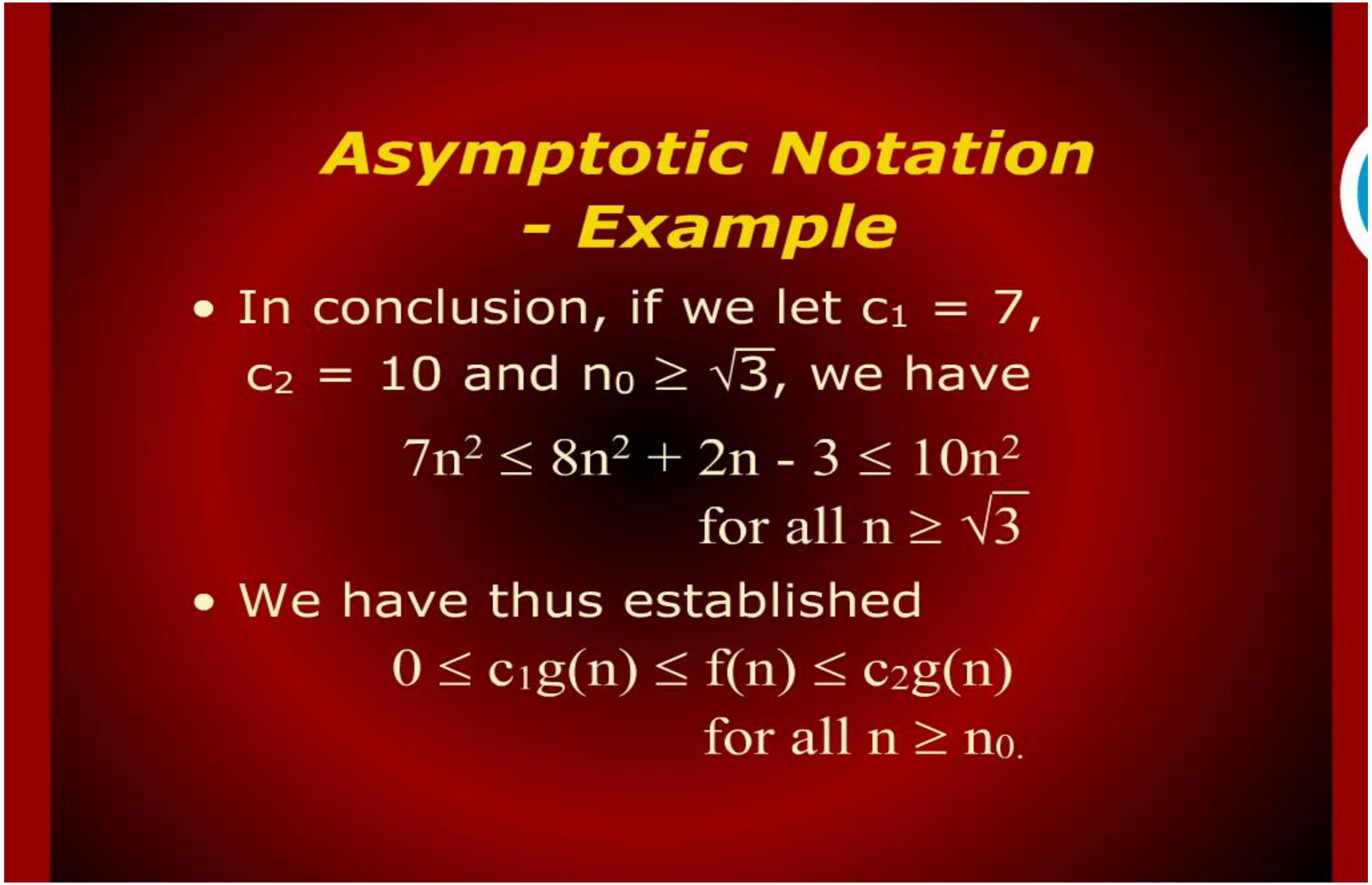
- We implicitly made the assumption that $2n \leq 2n^2$.
- This is not true for all n but it is true for all $n \geq 1$
- So select $n_0 \geq 1$.
- We thus have

$$f(n) \leq c_2 \, n^2 \text{ for all } n \geq n_0 \, .$$

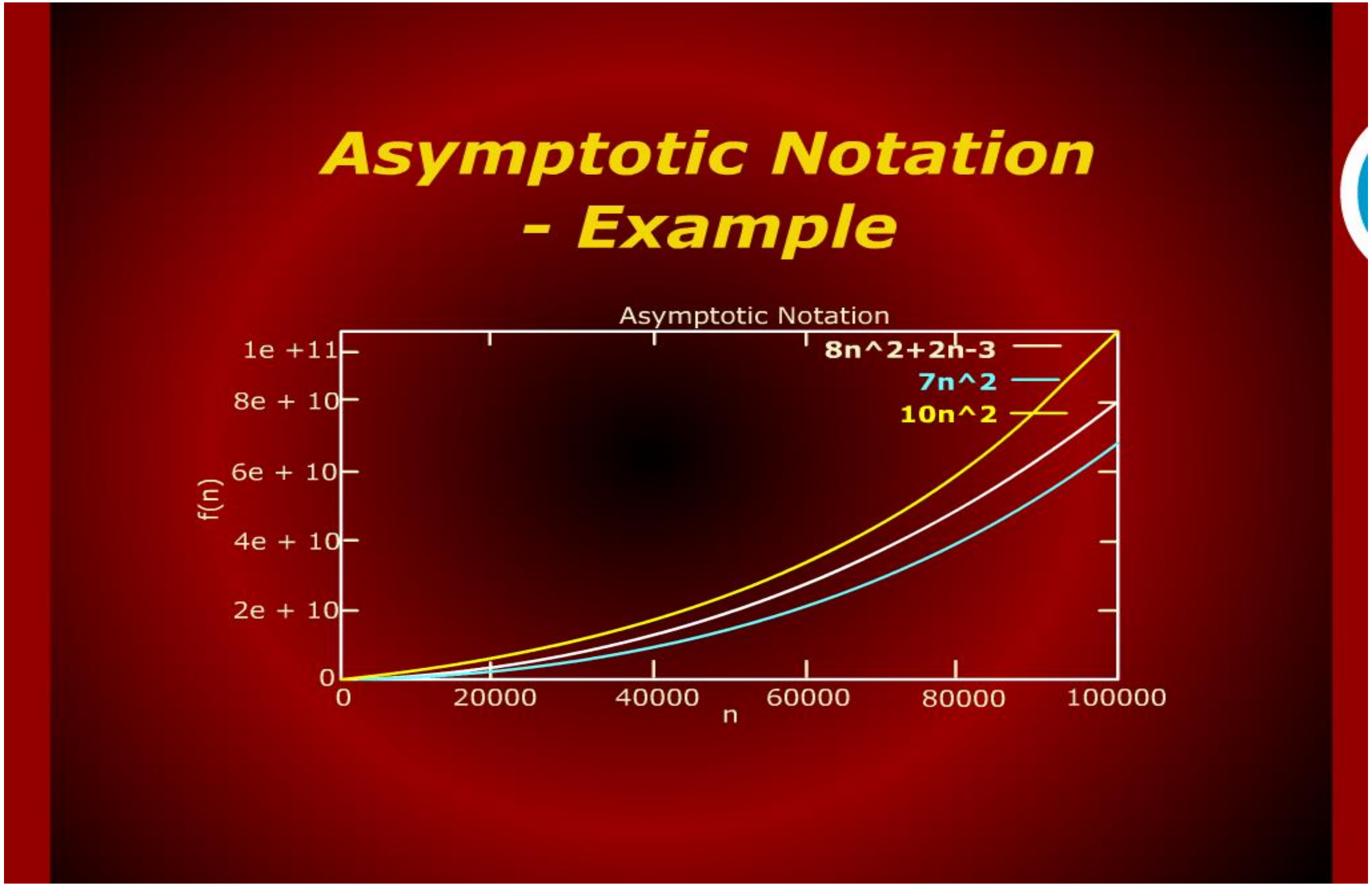
Asymptotic Notation - Example



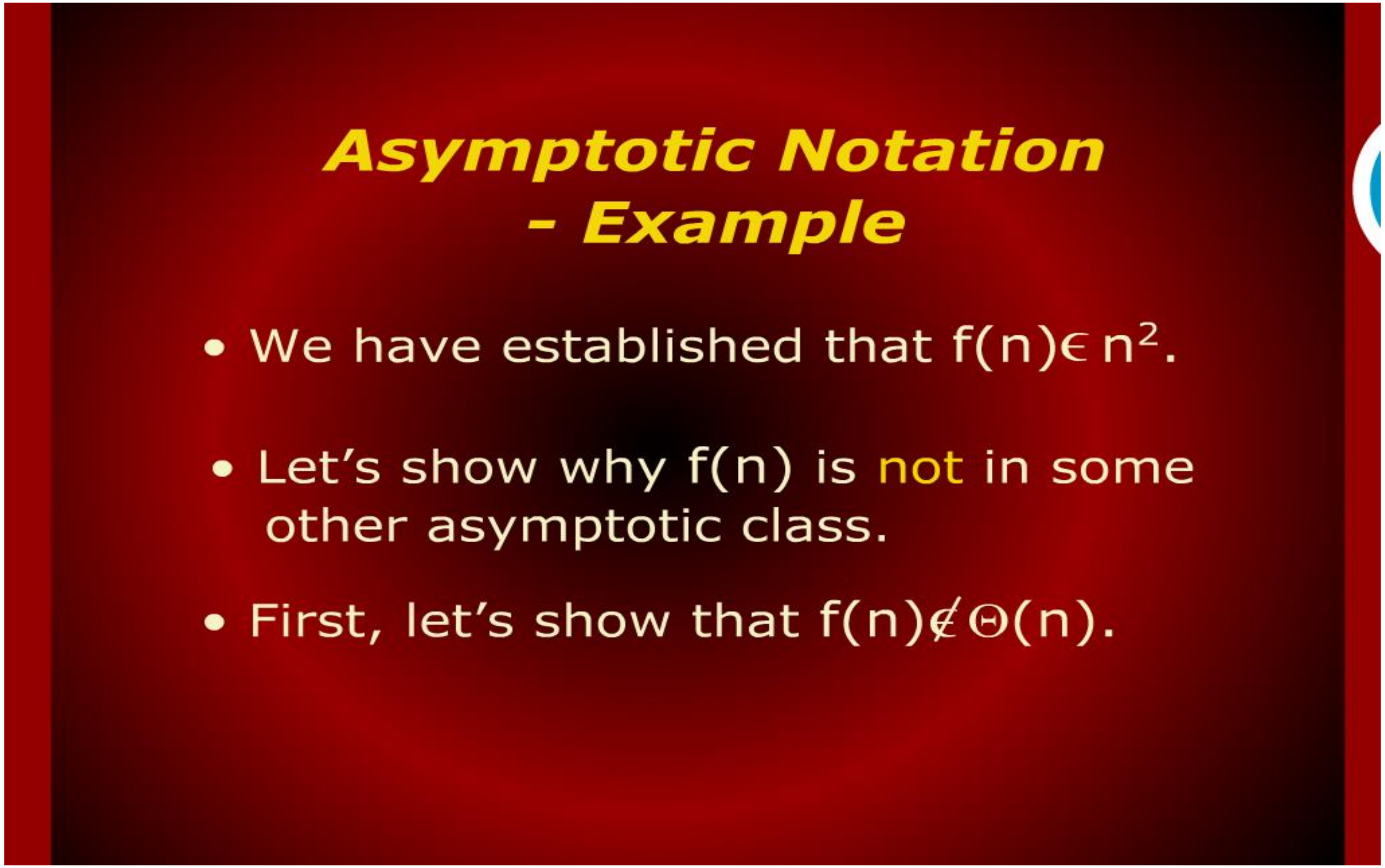
Asymptotic Notation - Example



Asymptotic Notation - Example



Asymptotic Notation - Example



Asymptotic Notation - Example

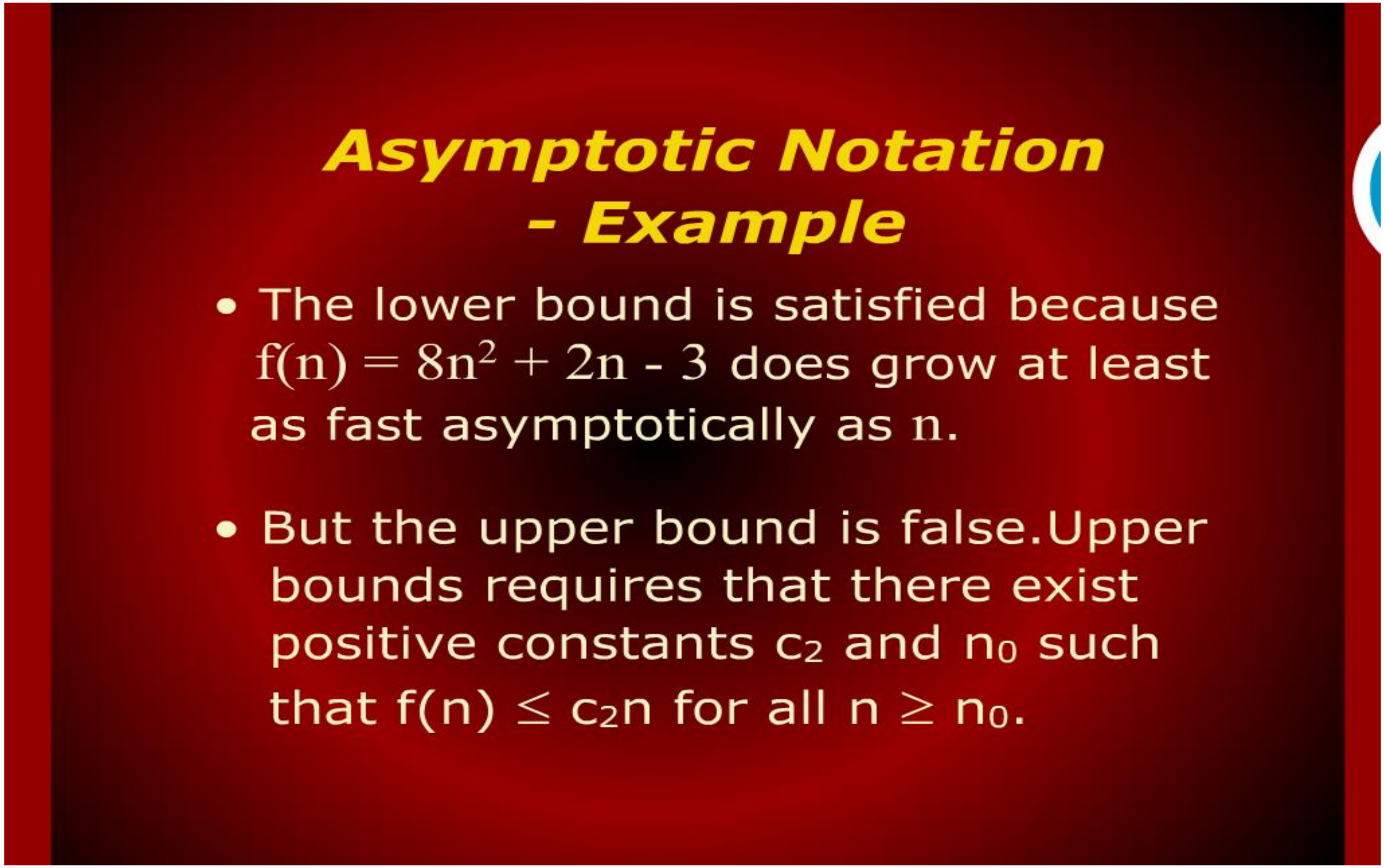
Asymptotic Notation - Example

show that $f(n) \notin \Theta(n)$.

- If this were true, we would have had to satisfy both the upper and lower bounds.

- The lower bound is satisfied because $f(n) = 8n^2 + 2n - 3$ does grow at least as fast asymptotically as n.

Asymptotic Notation - Example



Asymptotic Notation - Example

Asymptotic Notation - Example

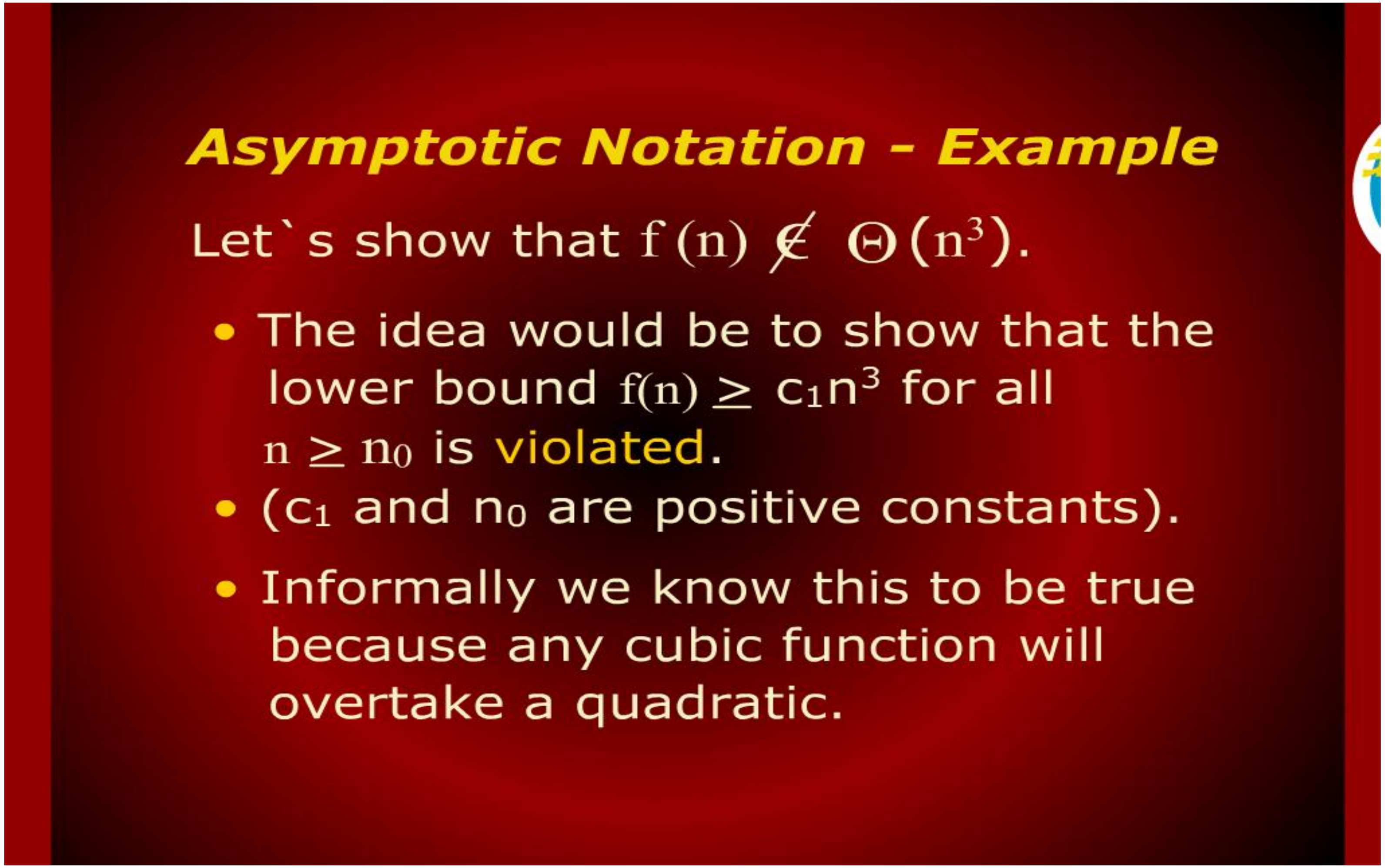
- Informally we know that $f(n) = 8n^2 + 2n - 3$ will eventually exceed $c_2 n$ no matter how large we make c_2.
- To see this, suppose we assume that constants c_2 and n_0 did exist such that $8n^2 + 2n - 3 \le c_2 n$ for all $n \ge n_0$

Asymptotic Notation - Example

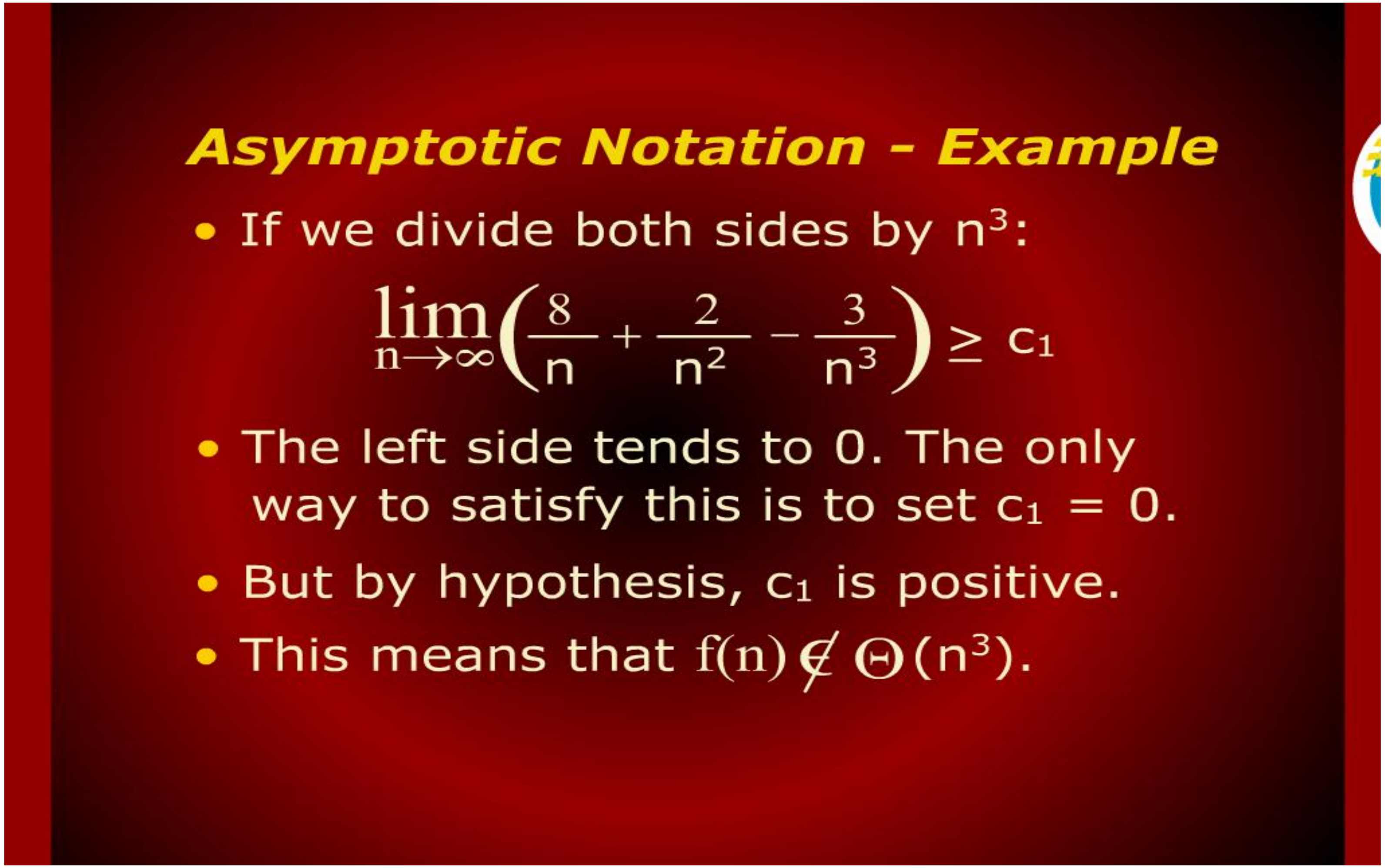
Asymptotic Notation - Example

- If we divide both sides by n, we have $\lim_{n \to \infty} \left(8n + 2 - \frac{3}{n}\right) \leq c_2.$
- It is easy to see that in the limit, the left side tends to ∞.
- So, no matter how large c_2 is, the statement is violated. Thus $f(n) \notin \Theta(n)$.

Asymptotic Notation - Example



Asymptotic Notation - Example



Θ-notation

- $\boldsymbol{f(n) = 10n^3 + 5n^2 + 17 \in \Theta(g(n^3))}$,
- $\boldsymbol{f2(n) = 2n^3 + 3n^2 + 79 \in \Theta(g(n^3))}$,

- Proof :
- $10n^3 \leq f(n) \leq (10 + 5 + 17)n^3$
- $C_1 = 10\,, C_2 = 32$
- $C_1 n^3 \leq f(n) \leq C_1 n^3$ *n0=1*

Θ-notation

- $f(\boldsymbol{n}) = \mathbf{10}\boldsymbol{n}^{\mathbf{3}} + \boldsymbol{n}.\boldsymbol{logn} \in \boldsymbol{\Theta}(\boldsymbol{g}(\boldsymbol{n}^{\mathbf{3}}))$

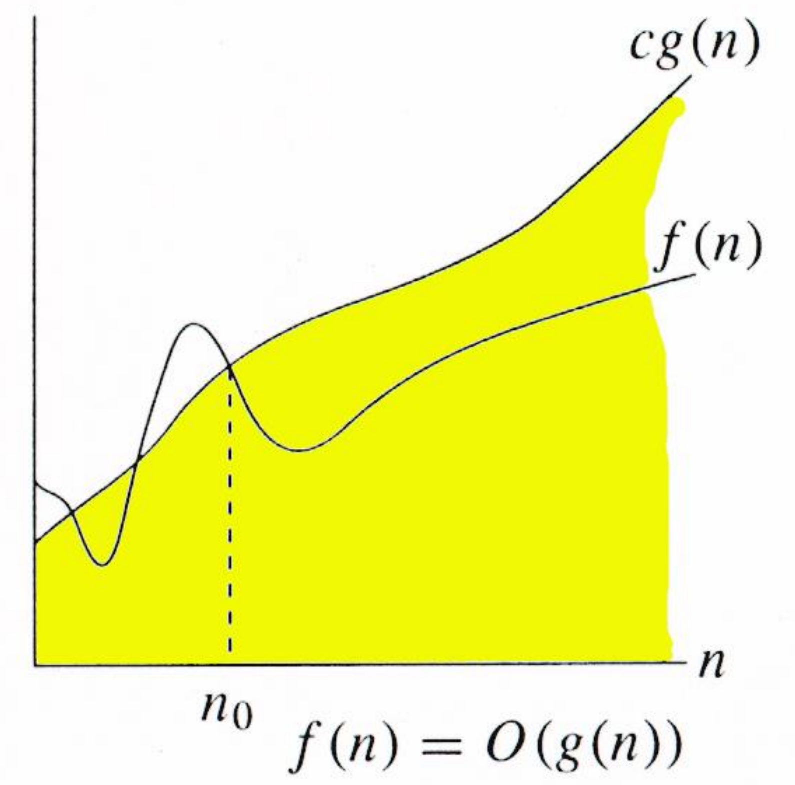
- Proof :
- $10n^3 \leq f(n) \leq (10+1)n^3$
- $C_1 = 10\ , C_2 = 11$
- $C_1 n^3 \leq f(n) \leq C_1 n^3$

O-notation

For function $g(n)$, we define $O(g(n))$, big-O of n, as the set:

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

Intuitively: Set of all functions whose *rate of growth* is the same as or lower than that of $g(n)$.



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

$f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$.
$\Theta(g(n)) \subset O(g(n))$.

Examples

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

- Any linear *function* $an + b$ is in $O(n^2)$. **How?**
- Show that $3n^3 = O(n^4)$ for appropriate c and n_0.

Informally, $O(g(n))$ is the set of all functions with a lower or same order of growth as $g(n)$ (to within a constant multiple, as n goes to infinity). Thus, to give a few examples, the following assertions are all true:

$$n \in O(n^2), \qquad 100n + 5 \in O(n^2), \qquad \frac{1}{2}n(n-1) \in O(n^2).$$

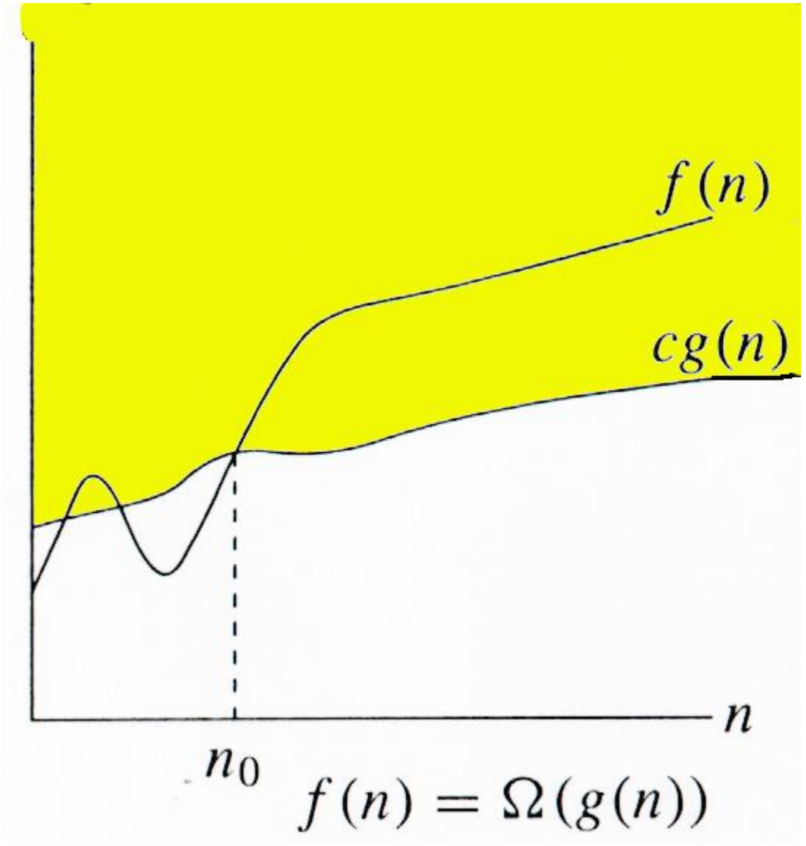
$$n^3 \notin O(n^2), \qquad 0.00001n^3 \notin O(n^2), \qquad n^4 + n + 1 \notin O(n^2).$$

Ω -notation

For function $g(n)$, we define $\Omega(g(n))$, big-Omega of n, as the set:

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

Intuitively: Set of all functions whose *rate of growth* is the same as or higher than that of $g(n)$.



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

$f(n) = \Theta(g(n)) \Rightarrow f(n) = \Omega(g(n))$.
$\Theta(g(n)) \subset \Omega(g(n))$.

The second notation, $\Omega(g(n))$, stands for the set of all functions with a higher or same order of growth as $g(n)$ (to within a constant multiple, as n goes to infinity). For example,

$$n^3 \in \Omega(n^2), \qquad \frac{1}{2}n(n-1) \in \Omega(n^2), \qquad \text{but } 100n + 5 \notin \Omega(n^2).$$

Relations Between Θ, *O*, Ω

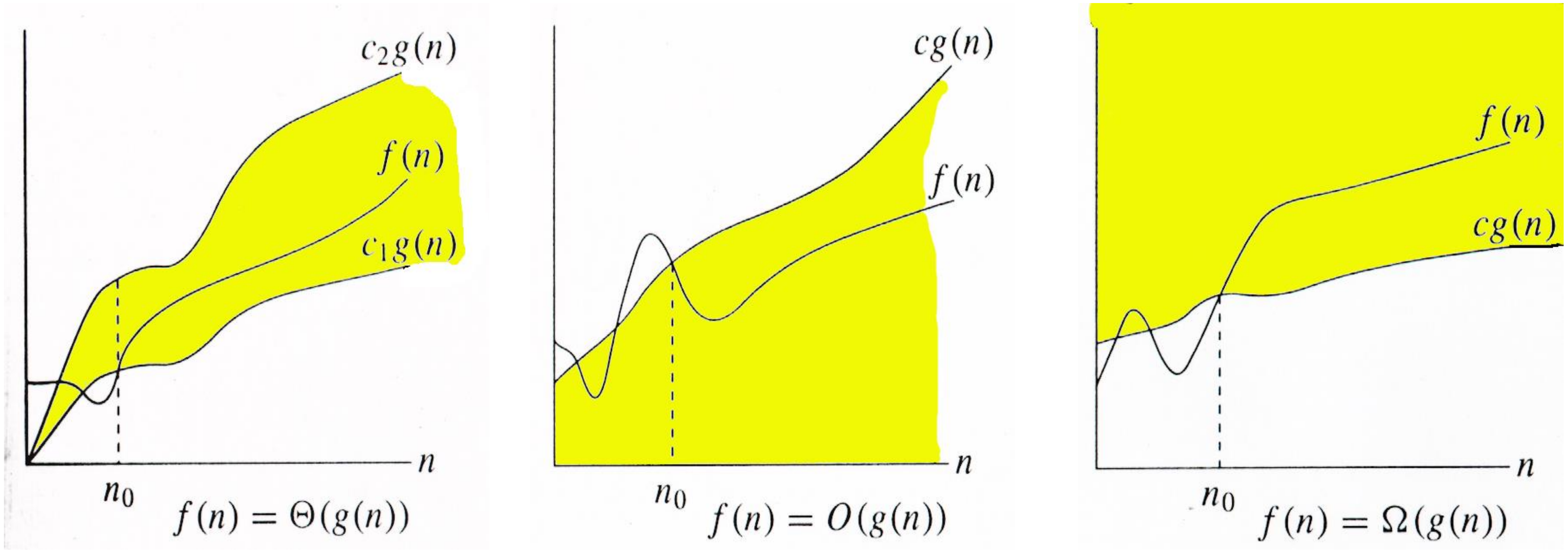


TABLE 2.2 Basic asymptotic efficiency classes

Class	Name	Comments
1	*constant*	Short of best-case efficiencies, very few reasonable examples can be given since an algorithm's running time typically goes to infinity when its input size grows infinitely large.
$\log n$	*logarithmic*	Typically, a result of cutting a problem's size by a constant factor on each iteration of the algorithm (see Section 4.4). Note that a logarithmic algorithm cannot take into account all its input or even a fixed fraction of it: any algorithm that does so will have at least linear running time.
n	*linear*	Algorithms that scan a list of size n (e.g., sequential search) belong to this class.
$n \log n$	*linearithmic*	Many divide-and-conquer algorithms (see Chapter 5), including mergesort and quicksort in the average case, fall into this category.
n^2	*quadratic*	Typically, characterizes efficiency of algorithms with two embedded loops (see the next section). Elementary sorting algorithms and certain operations on $n \times n$ matrices are standard examples.
n^3	*cubic*	Typically, characterizes efficiency of algorithms with three embedded loops (see the next section). Several nontrivial algorithms from linear algebra fall into this class.
2^n	*exponential*	Typical for algorithms that generate all subsets of an n-element set. Often, the term "exponential" is used in a broader sense to include this and larger orders of growth as well.
$n!$	*factorial*	Typical for algorithms that generate all permutations of an n-element set.

o-notation

For a given function $g(n)$, the set little-o:

$o(g(n))$ = $\{f(n)$: $\forall$ $c > 0$, $\exists$ $n_0 > 0$ such that
$\forall$ $n \geq n_0$, we have $0 \leq f(n) < cg(n)\}$.

ω -notation

For a given function $g(n)$, the set little-omega:

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

Relations Between Θ, Ω, O

Theorem : For any two functions $g(n)$ and $f(n)$,

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

$f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

- I.e., $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$
- In practice, asymptotically tight bounds are obtained from asymptotic upper and lower bounds.

Properties

- **Transitivity**

$$f(n) = \Theta(g(n)) \ \& \ g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$$
$$f(n) = O(g(n)) \ \& \ g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$
$$f(n) = \Omega(g(n)) \ \& \ g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$$
$$f(n) = o\,(g(n)) \ \& \ g(n) = o\,(h(n)) \Rightarrow f(n) = o\,(h(n))$$
$$f(n) = \omega(g(n)) \ \& \ g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$$

- **Complementarity**

$$f(n) = O(g(n)) \text{ iff } g(n) = \Omega(f(n))$$

$$f(n) = \ o(g(n)) \text{ iff } g(n) = \omega((f(n))$$

Monotonicity

- $f(n)$ is
  - **monotonically increasing** if $m \leq n \Rightarrow f(m) \leq f(n)$.
  - **monotonically decreasing** if $m \geq n \Rightarrow f(m) \geq f(n)$.
  - **strictly increasing** if $m < n \Rightarrow f(m) < f(n)$.
  - **strictly decreasing** if $m > n \Rightarrow f(m) > f(n)$.

Exponentials

- **Useful Identities:**

$$a^{-1} = \frac{1}{a}$$

$$(a^m)^n = a^{mn}$$

$$a^m a^n = a^{m+n}$$