

# Asymptotic Notations

## (Class 6)

From Book's Page No 50 (Chapter 3)

- Many people sloppily use  $O$  when they should use  $\theta$ .
- The book we are following also uses  $\theta$  to express worst-case running time.
- For example, an algorithm analyst may end up with a time function  $T(n) = n^2 + n + 2$  and immediately conclude that  $T(n) = O(n^2)$  which is technically right, but a sharper assertion would be  $T(n) = \theta(n^2)$ .

- We can attribute this oblivious behavior to two reasons. First, many see  $O$  to be more popular and acceptable, possibly because of its long history. Recall that it was introduced more than a century ago, whereas  $\theta$  and  $\Omega$  were introduced only in 1976 (by Donald Knuth).
- Second, it could be because  $O$  is readily available on the keyboard, whereas  $\theta$  is not!

- From a technical point of view, however, the main reason careful analysts prefer to use  $O$  over  $\theta$  is that the  $O$  covers “greater territory” than the  $\theta$ .
- If we take an example of some binary search and want to use  $\theta$  we will have to make two assertions:
  - One for the best case, namely  $\theta(1)$
  - Another for the worst case, namely  $\theta(\log n)$
- With  $O$  we make only one assertion, namely  $O(\log n)$ .

# The need of Asymptotic Notations

- If we are given running times of some algorithm:

$$T(n) = 4n^2 + 2n + 5$$

$$T(n) = n^2 + 10n + 9$$

$$T(n) = 95n^2 + 40n$$

- Which one is better?

- We may get confused by simply watching these polynomial equations.
- But if we only take the dominating term from each equation:

$$T(n) \approx n^2$$

$$T(n) \approx n^2$$

$$T(n) \approx n^2$$

- Now we can easily say that all these algorithms have equal running time.

# Growth of Function (Book's Page Number 32)

- We use Asymptotic Notations to describe the growth rate of the function.
- We know that for the growth of a function, the highest order term matters the most.
- e.g., the term  $c_1n^2$  in the function  $c_1n^2 + c_2n + c_3$  and thus we can neglect the other terms.

# Commonly Used Functions and Their Comparison

- **Constant Functions**

- Whatever is the input size  $n$  these functions take a constant amount of time.

$$f(n) = 1$$

- **Linear Functions**

- These functions grow linearly with the input size  $n$ .

$$f(n) = n$$



- **Quadratic Functions**

- These functions grow faster than the super-linear functions i.e.,  $n \log(n)$ .

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

- **Cubic Functions**

- Faster growing than quadratic but slower than exponential.

$$f(n) = n^3$$

- **Logarithmic Functions**

- These are slower growing than even linear functions.

$$f(n) = \log(n)$$

- **Super-linear Functions**

- Faster growing than linear but slower than quadratic.

$$f(n) = n \log(n)$$

- **Exponential Functions**

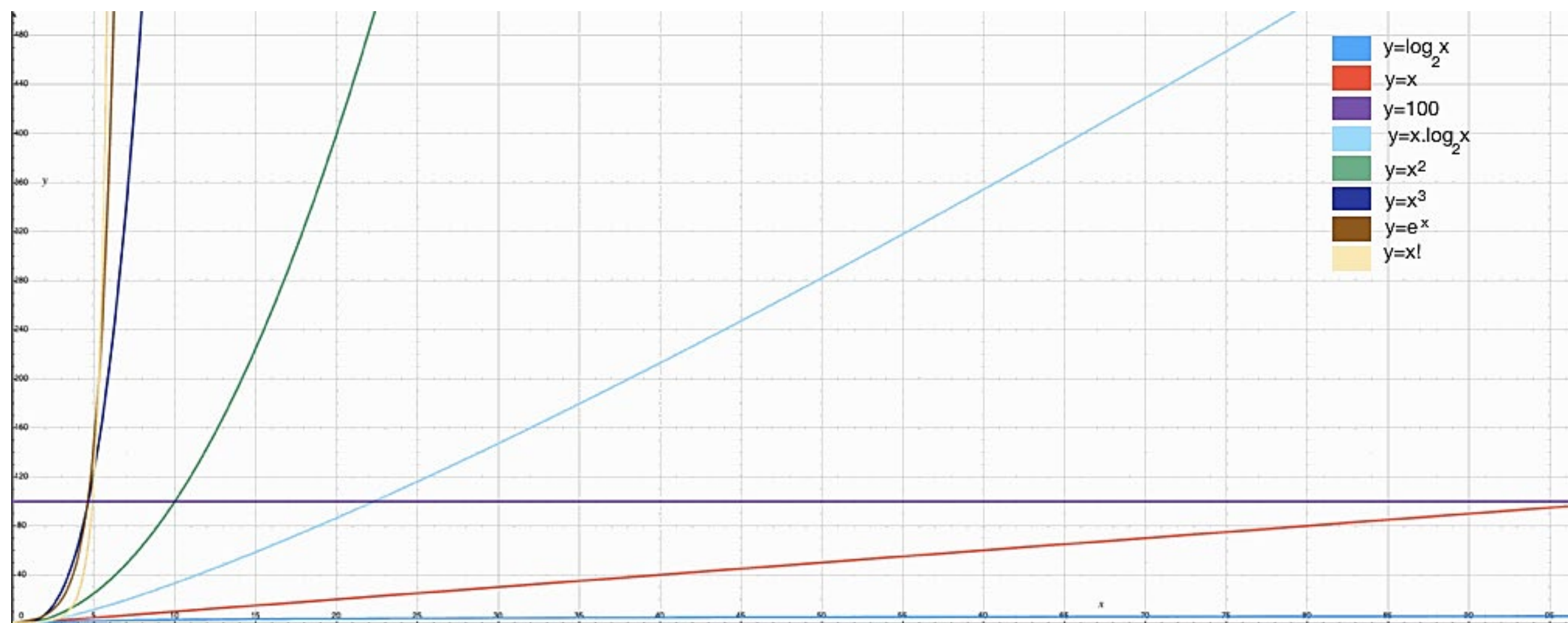
- Faster than all of the functions mentioned here except the factorial functions.

$$f(n) = c^n$$

- **Factorial Functions**

- Fastest growing than all these functions mentioned here.

$$f(n) = n!$$



- From the graph, you can see that for any sufficiently larger  $n$ :

$$n! \geq c^n \geq n^3 \geq n^2 \geq n \log(n) \geq n \geq \log(n) \geq 1$$

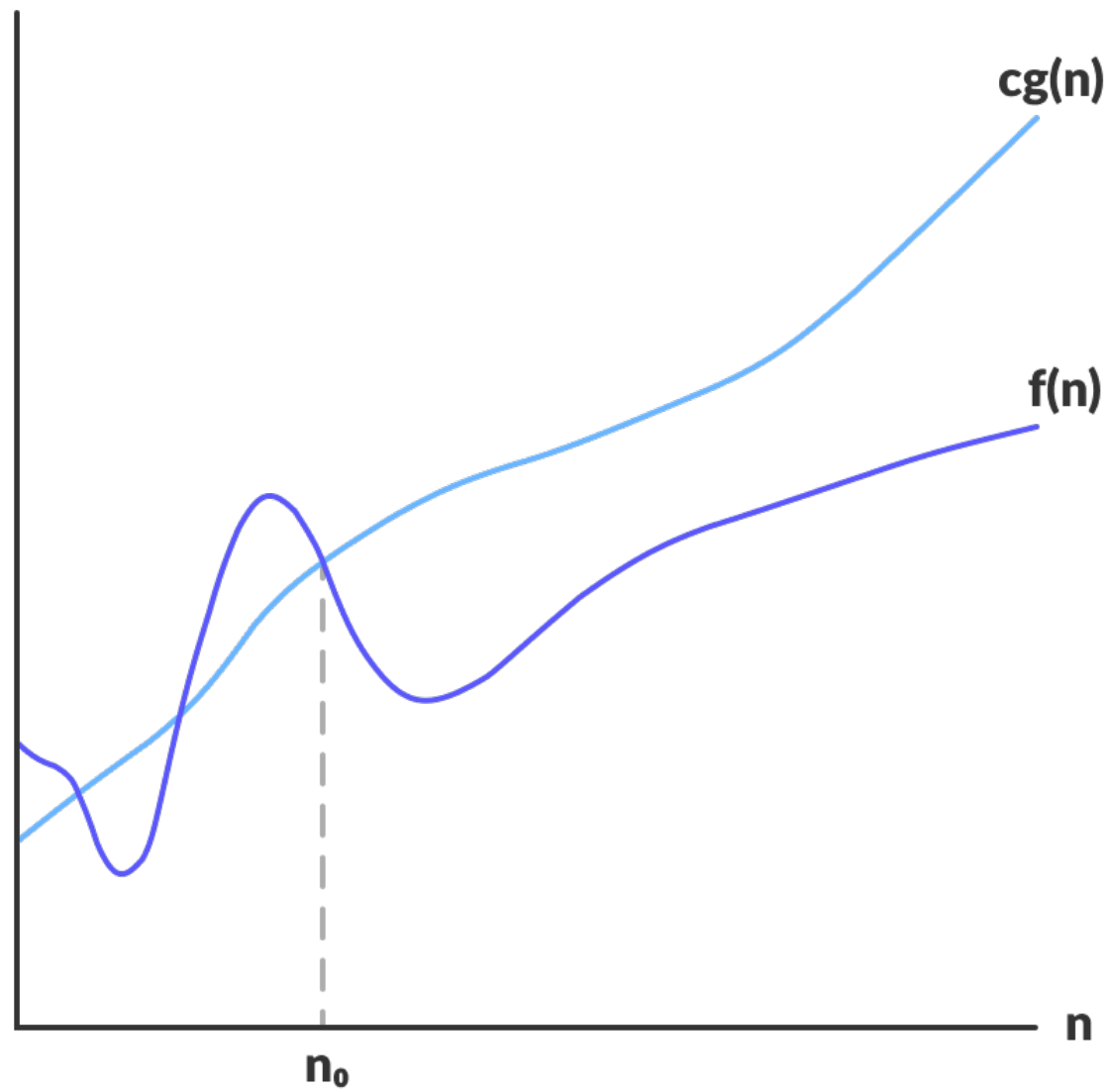
- We always want to keep the rate of the growth as low as possible.
- We try to make an algorithm to follow the function with least growth rate to accomplish a task.

# Asymptotic Notations

- We use some mathematical tools to describe the behavior of the running time.
- There are mainly three asymptotic notations:
  - $O$ -Notation (Big O)
  - $\Omega$ -Notation (Omega)
  - $\theta$ -Notation (Theta)

# $O$ -Notation

- Big-O notation represents the upper bound of the running time of an algorithm.
- Thus, it gives the worst-case complexity of an algorithm.
- $O$ -Notation characterizes an *upper bound* on the asymptotic behavior of a function.



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



- In other words, it says that a function grows no faster than a certain rate, based on the highest-order term.
- Consider, for example, the function  $7n^3 + 100n^2 + 20n + 6$ .
- Its highest-order term is  $7n^3$ , and so we say that this function's rate of growth is  $n^3$ .
- Because this function grows no faster than  $n^3$ , we can write that it is  $O(n^3)$ .
- Also, the restriction is not applied to the region of  $n \leq n_0$ .

# Formal Definition of $O$ -Notation

- For a given function  $g(n)$ , we denote by  $O(g(n))$  the set of functions.

$$O(g(n)) = \{ f(n) :$$

*there exist positive constants  $c$  and  $n_0$  such that*

$$0 \leq f(n) \leq cg(n)$$

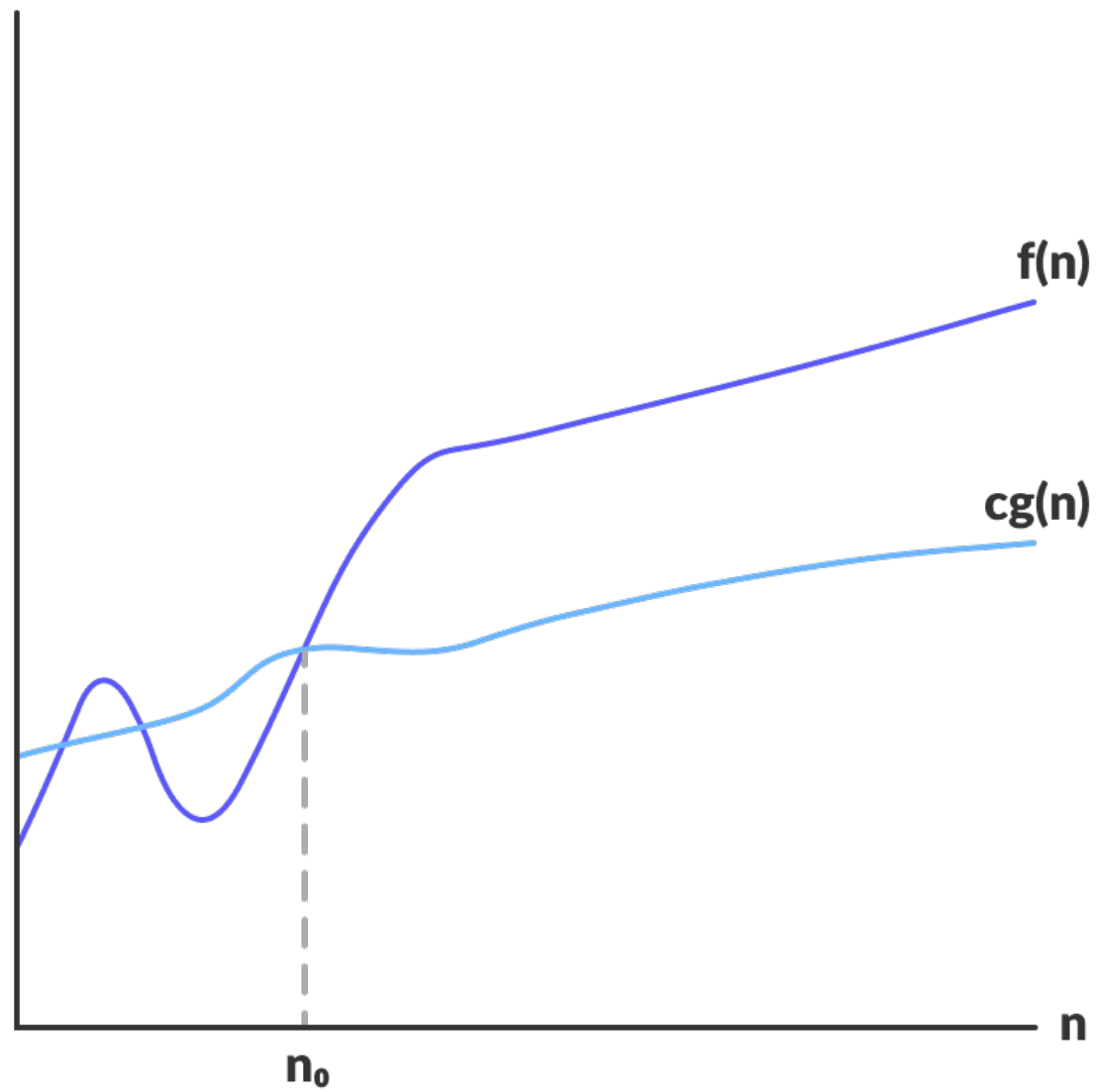
*for all  $n \geq n_0$  }*

$$f(n) \in O(g(n))$$

- A function  $f(n)$  belongs to the set  $O(g(n))$  if there exists a positive constant  $c$  such that  $f(n) \leq cg(n)$  for sufficiently large  $n$ .

# $\Omega$ -Notation

- $\Omega$ -Notation characterizes a *lower bound* on the asymptotic behavior of a function.
- Omega notation represents the lower bound of the running time of an algorithm.
- Thus, it provides the best-case complexity of an algorithm.



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

- In other words, it says that a function grows *at least as fast* as a certain rate.
- Because the highest-order term in the function  $7n^3 + 100n^2 + 20n + 6$  grows at least as fast as  $n^3$ , this function is  $\Omega(n^3)$ .
- This function is also  $\Omega(n^2)$  and  $\Omega(n)$ .
- More generally, it is  $\Omega(n^c)$  for any constant  $c \leq 3$ .

# Formal Definition of $\Omega$ -Notation

- For a given function  $g(n)$ , we denote by  $\Omega(g(n))$  the set of functions.

$$\Omega(g(n)) = \{ f(n) :$$

*there exist positive constants  $c$  and  $n_0$  such that*

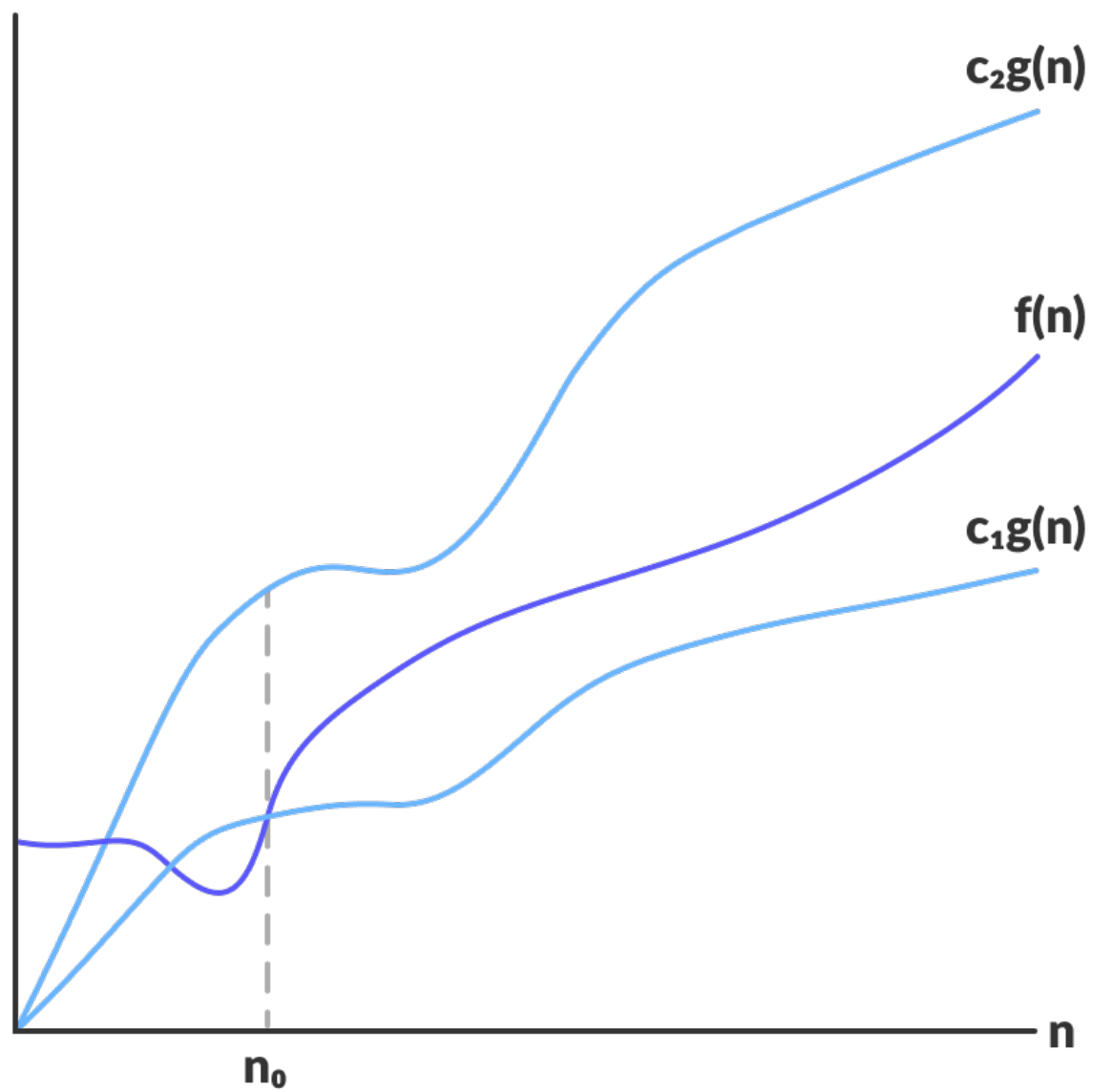
$$0 \leq cg(n) \leq f(n)$$

*for all  $n \geq n_0$  }*

# $\theta$ -Notation

- $\theta$ -Notation characterizes a *tight bound* on the asymptotic behavior of a function.
- Theta notation encloses the function from above and below. Since it represents the *upper* and the *lower bound* of the running time of an algorithm, it is used for analyzing the average-case complexity of an algorithm.
- It says that a function grows *precisely* at a certain rate, based on the highest-order term.





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

# Formal Definition of $\theta$ -Notation

- For a given function  $g(n)$ , we denote by  $\theta(g(n))$  the set of functions.

$$\theta(g(n)) = \{ f(n) :$$

*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  $n \geq n_0$  }*