CSE214 – Analysis of Algorithms

Lecture 2

Asymptotic Notation

Based on Cevdet Aykanat's and Mustafa Ozdal's Lecture Notes - Bilkent

Logaritma Nedir? Logaritma Formülleri Özellikleri

Logaritma Tanımı: a, b ∈ R^+ ve a≠1 olmak üzere $a^x = b$ denklemini sağlayan x sayısına $\log_a b$ denir ve b'nin a tabanında logaritması diye okunur.

1) $\log_a x = b$ ise $x = a^b$ $\quad log_2 8 = 3 \mid 8 = 2^3$

2) $\log_a(A.B) = \log_a A + \log_a B$ $\quad log_2(4*8) = log_2(32) = 5 = log_2(4) + log_2(8) = 2 + 3$

3) $\log_a(A/B) = \log_a A - \log_a B$ $\quad log_2(16/4) = log_2(4) = 2 = log_2(16) - log_2(4) = 4 - 2 = 2$

4) $\log_a A^n = n.\log_a A$ $\quad log_2 8^2 = log_2 64 = 6 = 2 * log_2 8 = 2 * 3 = 6$

5) $\log_{a^m} A^n = \frac{n}{m} \log_a A$ $\quad log_{2^3} 8^2 = log_8 64 = 2 = (2/3) * log_2 8 = 2/3 * 3 = 2$

6) $\log_{(a^n)} x = \frac{1}{n} .\log_a x$ $\quad log_{2^3} 8 = log_8 8 = 1 = (1/3) * log_2 8 = 1/3 * 3 = 1$

7) $\log_a x = (\log_b x)/(\log_b a)$ [taban değiştirme] $\quad log_4 16 = 2 = log_2 16 - log_2 4 = 4 - 2 = 2$

8) $a^{\log_a x} =$ $\quad 2^{log_2 8} = 2^3 = 8 = 8$

9) $\log_a \sqrt[n]{A} = \frac{1}{n} \log_a A$ $\quad log_2 \sqrt[3]{8} = log_2 2 = 1 = \left(\frac{1}{3}\right) * log_2 8 = \frac{1}{3} * 3 = 1$

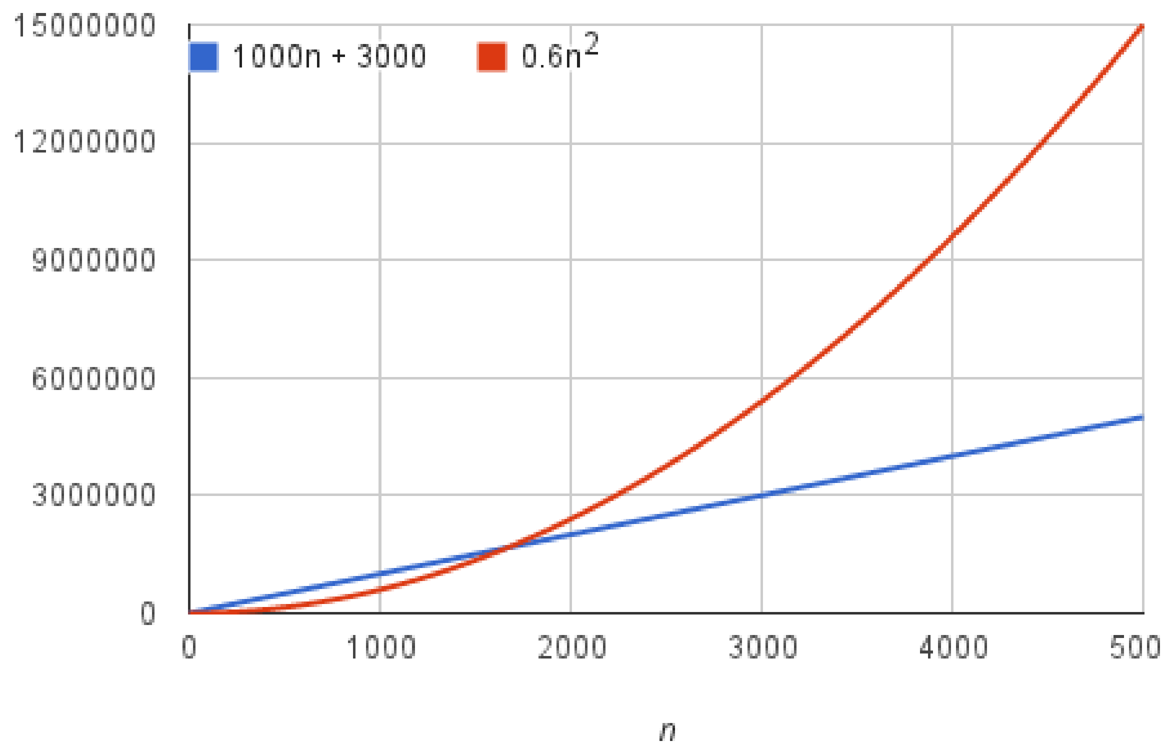
10) $\log_{1/a} x = -\log_a x$ $\quad log_{1/2} 8 = -log_2 8 = -3 \mid 8 = \left(\frac{1}{2}\right)^{-3}$

11) $\log_a b.\log_b c.\log_c d = \log_a d$ $\quad log_2 4 * log_4 16 * log_{16} 256 = 2 * 2 * 2 = 8 = log_2 256$

12) $\log_a b = 1/\log_b a$ veya $\log_a b.\log_b a = 1$

What is Asymptotic notation

By dropping the less significant terms and the constant coefficients, we can focus on the important part of an algorithm's running time—its rate of growth—without getting mired in details that complicate our understanding.



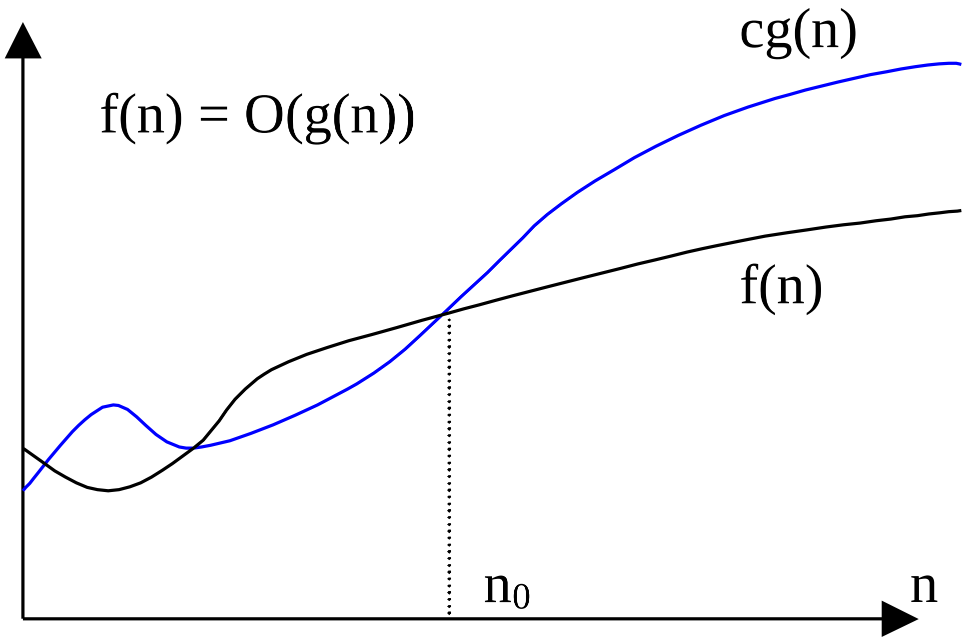
When we drop the constant coefficients and the less significant terms, we use asymptotic notation.

We'll see three forms of it: big-Theta **Θ** notation, big-**O** notation, and big-Omega **Ω** notation.

O-notation: Asymptotic upper bound

$f(n) = O(g(n))$ if $\exists$ positive constants c, n_0 such that

$$0 \le f(n) \le cg(n),\ \forall n \ge n_0$$



Asymptotic running times of algorithms are usually defined by functions whose domain are N={0, 1, 2, …} (natural numbers)

Example

Show that $2n^2 = O(n^3)$

We need to find two positive constants: **c** and $\mathbf{n_0}$ such that:

$$0 \leq 2n^2 \leq cn^3 \quad \text{for all } n \geq n_0$$

Choose $c = 2$ and $n_0 = 1$

$\Rightarrow$ $2n^2 \leq 2n^3$ for all $n \geq 1$

Or, choose $c = 1$ and $n_0 = 2$

$\Rightarrow$ $2n^2 \leq n^3$ for all $n \geq 2$

Example

Show that $2n^2 + n = O(n^2)$

We need to find two positive constants: **c** and **n_0** such that:

$$0 \leq 2n^2 + n \leq cn^2 \text{ for all } n \geq n_0$$

$$2 + (1/n) \leq c \text{ for all } n \geq n_0$$

Choose $c = 3$ and $n_0 = 1$

$\Rightarrow$ $2n^2 + n \leq 3n^2$ for all $n \geq 1$

O-notation

- What does $f(n) = O(g(n))$ really mean?
  - The notation is a little sloppy
  - One-way equation
    - e.g. $n^2 = O(n^3)$, but we cannot say $O(n^3) = n^2$
- $O(g(n))$ is in fact a set of functions:

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

O-notation

- $O(g(n)) = \{f(n): \exists$ positive constants c, n_0 such that

  $0 \le f(n) \le cg(n), \forall n \ge n_0\}$

- In other words: $O(g(n))$ is in fact:

  the set of functions that have asymptotic upper bound g(n)

- e.g. $2n^2 = O(n^3)$ *means* $2n^2 \in O(n^3)$

$2n^2$ is in the set of functions that have asymptotic upper bound n^3

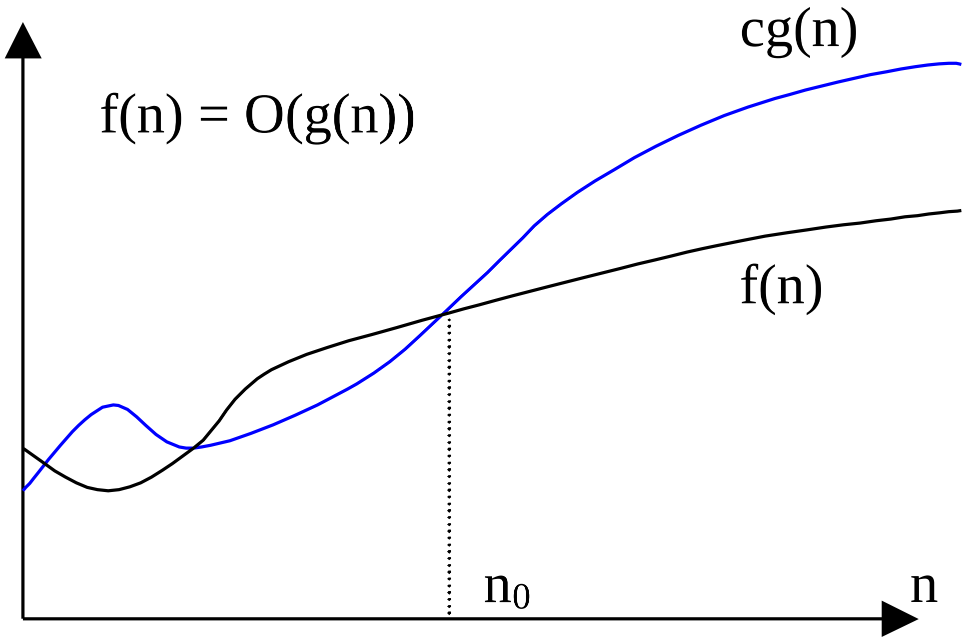
True or False?

$10^9 n^2 = O(n^2)$	True	Choose $c = 10^9$ and $n_0 = 1$ $0 \le 10^9 n^2 \le 10^9 n^2$ for $n \ge 1$
$100 n^{1.9999} = O(n^2)$	True	Choose $c = 100$ and $n_0 = 1$ $0 \le 100 n^{1.9999} \le 100 n^2$ for $n \ge 1$
$\cancel{10^{-9} n^{2.0001} = O(n^2)}$	False	$10^{-9} n^{2.0001} \le c n^2$ for $n \ge n_0$ $10^{-9} n^{0.0001} \le c$ for $n \ge n_0$ Contradiction

Summary: *O*-notation: Asymptotic upper bound

$f(n) \in O(g(n))$ if $\exists$ positive constants c, n_0 such that

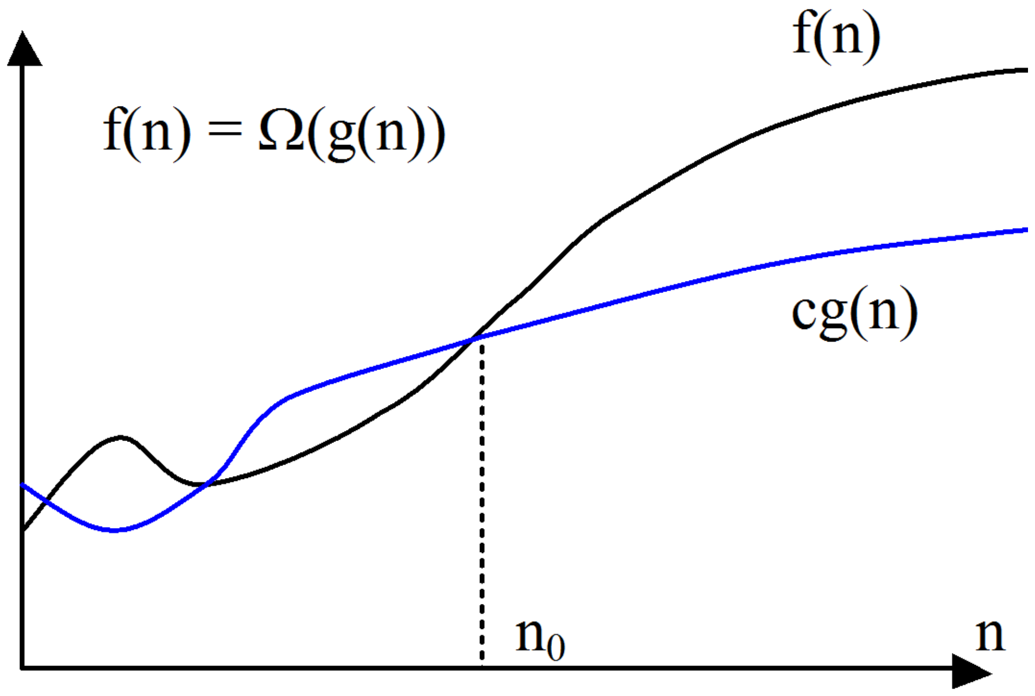
$$0 \le f(n) \le cg(n), \forall n \ge n_0$$



Ω-notation: Asymptotic lower bound

$f(n) = \Omega(g(n))$ if $\exists$ positive constants c, n_0 such that

$$0 \le cg(n) \le f(n),\ \forall n \ge n_0$$

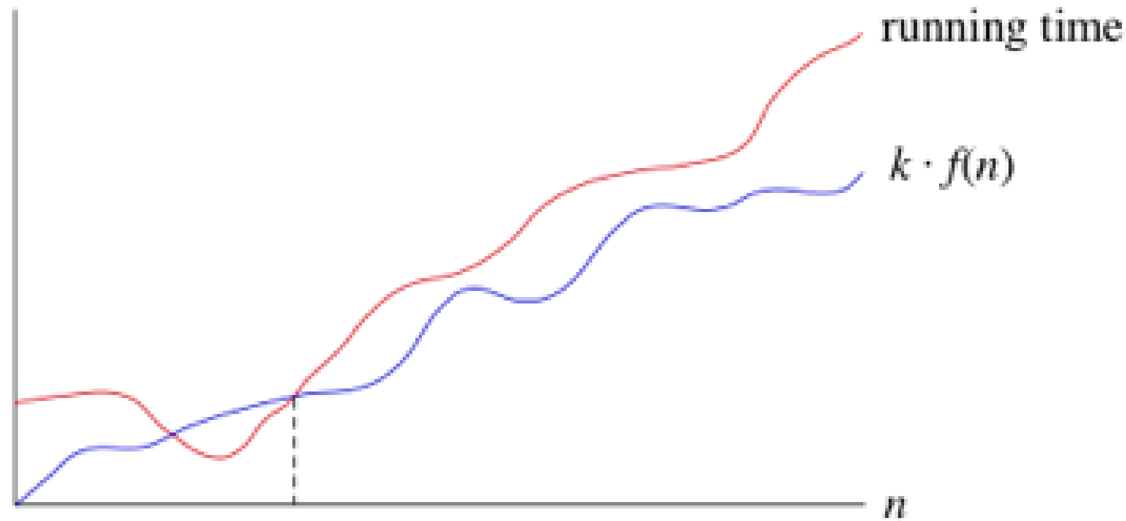


Ω: "big Omega"

What is Asymptotic lower bound

Sometimes, we want to say that an algorithm takes at least a certain amount of time, without providing an upper bound. We use big-Ω notation; that's the Greek letter "omega."

If a running time is $\Omega(f(n))$, then for large enough n, the running time is at least $k \cdot f(n)$, for some constant k. Here's how to think of a running time that is $\Omega(f(n))$:



We say that the running time is "big-Ω of f(n)" We use big-Ω notation for asymptotic lower bounds, since it bounds the growth of the running time from below for large enough input sizes.

Example

Show that $2n^3 = \Omega(n^2)$

We need to find two positive constants: **c** and $\mathbf{n_0}$ such that:

$$0 \le cn^2 \le 2n^3 \quad \text{for all } n \ge n_0$$

Choose $c = 1$ and $n_0 = 1$

$$\Rightarrow n^2 \le 2n^3 \text{ for all } n \ge 1$$

Example

Show that $\sqrt{n} = \Omega(\lg n)$

We need to find two positive constants: **c** and $\mathbf{n_0}$ such that:

$$c \lg n \leq \sqrt{n} \text{ for all } n \geq n_0$$

Choose $c = 1$ and $n_0 = 16$

$$\Rightarrow \lg n \leq \sqrt{n} \text{ for all } n \geq 16$$

Ω-notation: Asymptotic Lower Bound

- $\Omega(g(n)) = \{f(n)$: $\exists$ positive constants c, n_0 such that
  $0 \le cg(n) \le f(n), \forall n \ge n_0\}$

- In other words: $\Omega(g(n))$ is in fact:
  the set of functions that have asymptotic lower bound g(n)

True or False?

$10^9 n^2 = \Omega(n^2)$

True

Choose $c = 10^9$ and $n_0 = 1$

$0 \leq 10^9 n^2 \leq 10^9 n^2$ for $n \geq 1$

$\cancel{100n^{1.9999} = \Omega(n^2)}$

False

$cn^2 \leq 100n^{1.9999}$ for $n \geq n_0$

$n^{0.0001} \leq (100/c)$ for $n \geq n_0$

Contradiction

$10^{-9} n^{2.0001} = \Omega(n^2)$

True

Choose $c = 10^{-9}$ and $n_0 = 1$

$0 \leq 10^{-9} n^2 \leq 10^{-9} n^{2.0001}$ for $n \geq 1$

Summary: O-notation and Ω-notation

- $O(g(n))$: The set of functions with asymptotic upper bound $g(n)$

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

  $f(n) \in O(g(n))$ if $\exists$ positive constants c, n_0 such that

  $$0 \le f(n) \le cg(n),\ \forall\ n \ge n_0$$

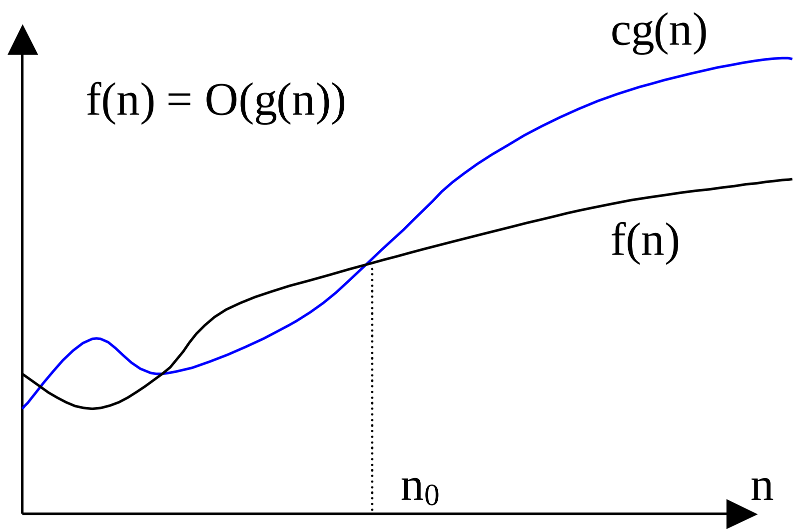
- $\Omega(g(n))$: The set of functions with asymptotic lower bound $g(n)$

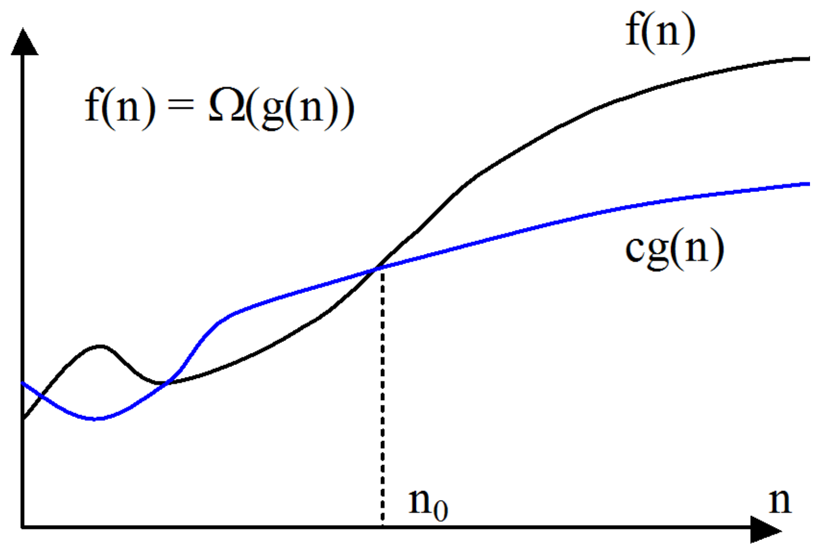
  $f(n) = \Omega(g(n))$

  $f(n) \in \Omega(g(n))$ $\exists$ positive constants c, n_0 such that

  $$0 \le cg(n) \le f(n),\ \forall n \ge n_0$$

Summary: O-notation and Ω-notation

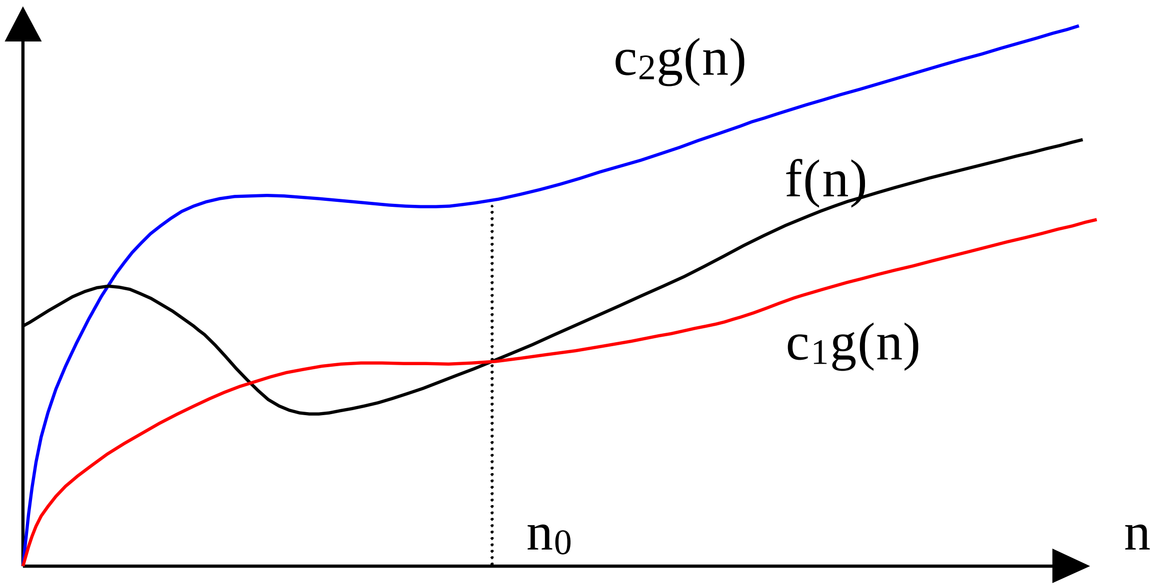




Θ-notation: Asymptotically tight bound

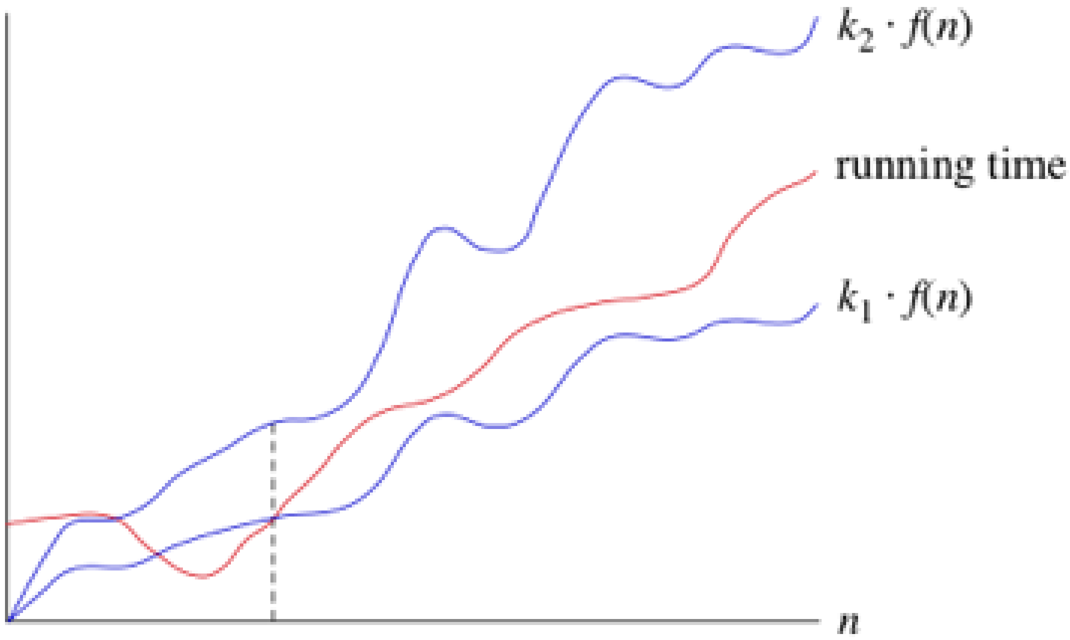
- $f(n)=\Theta(g(n))$ if $\exists$ positive constants c_1, c_2, n_0 such that

$$0 \le c_1 g(n) \le f(n) \le c_2 g(n),\ \forall n \ge n_0$$



What is Asymptotically tight bound

When we say that a particular running time is $\Theta(n)$, we're saying that once n gets large enough, the running time is at least $k1 \cdot f(n)$ and at most $k2 \cdot f(n)$ for some constants $k1$ and $k2$. Here's how to think of $\Theta(n)$:



Once n gets large enough, the running time is between $k1 \cdot f(n)$ and $k2 \cdot f(n)$

Example

Show that $2n^2 + n = \Theta(n^2)$

We need to find 3 positive constants: $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{n_0}$ such that:

$0 \le c_1 n^2 \le 2n^2 + n \le c_2 n^2$ for all $n \ge n_0$

$c_1 \le 2 + (1/n) \le c_2$ for all $n \ge n_0$

Choose $c_1 = 2$, $c_2 = 3$, and $n_0 = 1$

$\Rightarrow$ $2n^2 \le 2n^2 + n \le 3n^2$ for all $n \ge 1$

Example

Show that $\frac{1}{2}n^2 - 2n = \Theta(n^2)$

We need to find 3 positive constants: **c_1, c_2** and **n_0** such that:

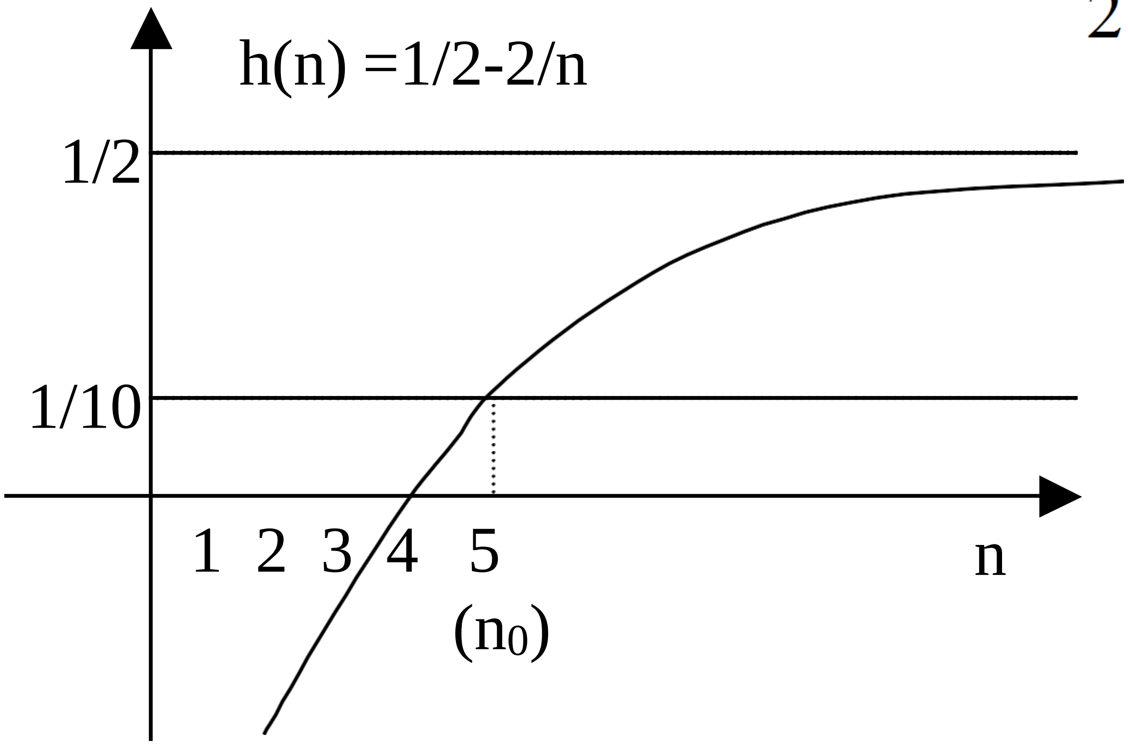
$0 \leq c_1 n^2 \leq \frac{1}{2}n^2 - 2n \leq c_2 n^2$ for all $n \geq n_0$

$$c_1 \leq \frac{1}{2} - \frac{2}{n} \leq c_2$$ for all $n \geq n_0$

Example (cont'd)

- Choose 3 positive constants: c_1, c_2, n_0 that satisfy:

$$c_1 \le \frac{1}{2} - \frac{2}{n} \le c_2 \qquad \text{for all } n \ge n_0$$



$$\frac{1}{10} \le \frac{1}{2} - \frac{2}{n} \qquad \text{for } n \ge 5$$

$$\frac{1}{2} - \frac{2}{n} \le \frac{1}{2} \qquad \text{for } n \ge 0$$

Example (cont'd)

- Choose 3 constants: c_1, c_2, n_0 that satisfy:

$$c_1 \leq \frac{1}{2} - \frac{2}{n} \leq c_2$$ for all $n \geq n_0$

$$\frac{1}{10} \leq \frac{1}{2} - \frac{2}{n}$$ for $n \geq 5$

$$\frac{1}{2} - \frac{2}{n} \leq \frac{1}{2}$$ for $n \geq 0$

Therefore, we can choose:: $c_1 = \frac{1}{10}$ $c_2 = \frac{1}{2}$ $n_0 = 5$

Θ-notation: Asymptotically tight bound

- Theorem: leading constants & low-order terms don’t matter
- Justification: can choose the leading constant large enough to make high-order term dominate other terms

True or False?

$10^9 n^2 = \Theta(n^2)$ True

$\cancel{100n^{1.9999} = \Theta(n^2)}$ False

$\cancel{10^{-9} n^{2.0001} = \Theta(n^2)}$ False

Θ-notation: Asymptotically tight bound

- $\Theta(g(n))=\{f(n)$: $\exists$ positive constants c_1, c_2, n_0 such that

  $0 \le c_1 g(n) \le f(n) \le c_2 g(n), \forall n \ge n_0\}$

- In other words: $\Theta(g(n))$ is in fact:

  the set of functions that have asymptotically tight bound g(n)

Θ-notation: Asymptotically tight bound

- Theorem:

  $f(n) = \Theta(g(n))$ if and only if

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

- In other words:

  Θ is stronger than both O and Ω

Example

- Prove that $10^{-8} n^2 \neq \Theta(n)$

Before proof, note that $10^{-8}n^2 = \Omega(n)$ but $10^{-8}n^2 \neq O(n)$

Proof by contradiction:

Suppose positive constants c_2 and n_0 exist such that:

$10^{-8}n^2 \leq c_2 n$ for all $n \geq n_0$

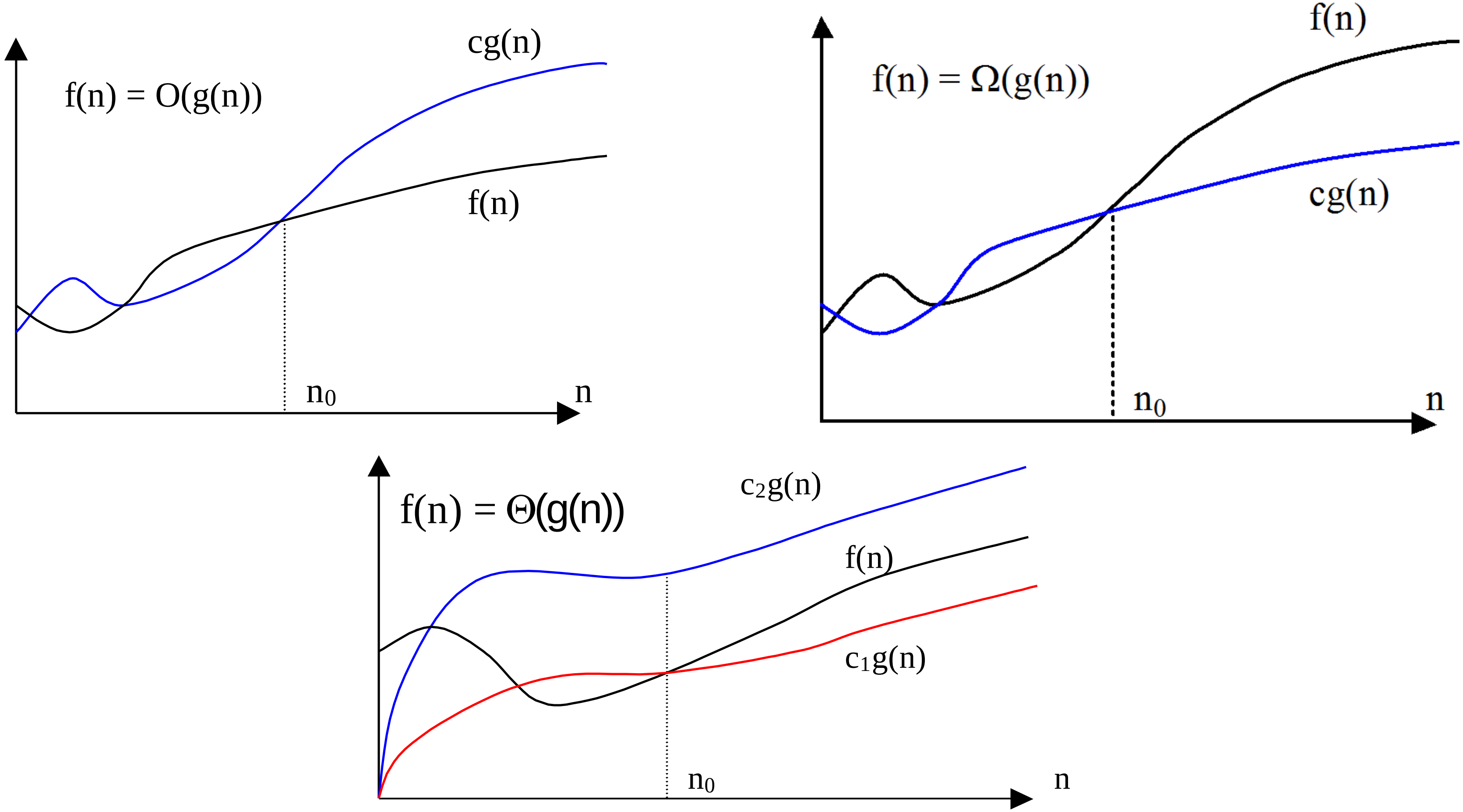
$10^{-8}n \leq c_2$ for all $n \geq n_0$

Contradiction: c_2 is a constant

Summary: O, Ω, and Θ notations

- $O(g(n))$: The set of functions with asymptotic upper bound $g(n)$
- $\Omega(g(n))$: The set of functions with asymptotic lower bound $g(n)$
- $\Theta(g(n))$: The set of functions with asymptotically tight bound $g(n)$
- $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

Summary: O, Ω, and Θ notations



o ("small o") Notation
Asymptotic upper bound that is not tight

Reminder: Upper bound provided by O ("big O") notation can be tight or not tight:

e.g. $2n^2 = O(n^2)$ is asymptotically tight

$2n = O(n^2)$ is not asymptotically tight

both true

o-Notation: An upper bound that is not asymptotically tight

o ("small o") Notation
Asymptotic upper bound that is not tight

- $o(g(n)) = \{f(n)$: for **any** constant $c > 0$,

  $\exists$ a constant $n_0 > 0$, such that

  $0 \le f(n) < cg(n), \forall n \ge n_0\}$

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

  - e.g., $2n = o(n^2)$, any positive c satisfies

    but $2n^2 \ne o(n^2)$, $c = 2$ does not satisfy

ω ("small omega") Notation
Asymptotic lower bound that is not tight

- $\omega(g(n))$ = {$f(n)$: for **any** constant $c > 0$,
  $\exists$ a constant $n_0 > 0$, such that
  $0 \le cg(n) < f(n), \forall n \ge n_0$}

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

  - e.g., $n^2/2 = \omega(n)$, any positive c satisfies

    but $n^2/2 \ne \omega(n^2)$, $c = 1/2$ does not satisfy

Analogy to the comparison of two real numbers

- $f(n) = O(g(n)) \leftrightarrow a \leq b$
- $f(n) = \Omega(g(n)) \leftrightarrow a \geq b$
- $f(n) = \Theta(g(n)) \leftrightarrow a = b$

- $f(n) = o(g(n)) \leftrightarrow a < b$
- $f(n) = \omega(g(n)) \leftrightarrow a > b$

True or False?

$5n^2 = O(n^2)$	True	$n^2 \lg n = O(n^2)$	False
$5n^2 = \Omega(n^2)$	True	$n^2 \lg n = \Omega(n^2)$	True
$5n^2 = \Theta(n^2)$	True	$n^2 \lg n = \Theta(n^2)$	False
$5n^2 = o(n^2)$	False	$n^2 \lg n = o(n^2)$	False
$5n^2 = \omega(n^2)$	False	$n^2 \lg n = \omega(n^2)$	True

$2^n = O(3^n)$	True		
$2^n = \Omega(3^n)$	False	$2^n = o(3^n)$	True
$2^n = \Theta(3^n)$	False	$2^n = \omega(3^n)$	False

Using O-Notation to Describe Running Times

- Used to bound worst-case running times
  - Implies an upper bound runtime for arbitrary inputs as well

- Example:

  "Insertion sort has worst-case runtime of $O(n^2)$"

  Note: This $O(n^2)$ upper bound also applies to its running time on every input.

Using O-Notation to Describe Running Times

- Abuse to say "running time of insertion sort is $O(n^2)$"

- For a given n, the actual running time ***depends on*** *the particular input* of size n
  - i.e., running time is not only a function of n

- However, worst-case running time is only a function of n

Using O-Notation to Describe Running Times

- When we say:

  "*Running time of insertion sort is $O(n^2)$*",

  what we really mean is:

  "*Worst-case running time of insertion sort is $O(n^2)$*"

  or equivalently:

  "No matter what particular input of size n is chosen, the running time on that set of inputs is $O(n^2)$"

Using Ω-Notation to Describe Running Times

- Used to bound best-case running times
  - Implies a lower bound runtime for arbitrary inputs as well

- Example:

  "Insertion sort has best-case runtime of $\Omega(n)$"

  Note: This $\Omega(n)$ lower bound also applies to its running time on every input.

Using Ω-Notation to Describe Running Times

- When we say:

  "*Running time of algorithm A is $\Omega(g(n))$*",

  what we mean is:

  "For any input of size n, the runtime of A is <u>at least</u> a constant times $g(n)$ for sufficiently large n"

Using Ω-Notation to Describe Running Times

- *Note*: It's not contradictory to say:

  "*worst-case running time of insertion sort is* $\Omega(n^2)$"

because there exists an input that causes the algorithm to take $\Omega(n^2)$.

Using Θ-Notation to Describe Running Times

- Consider 2 cases about the runtime of an algorithm:

- Case 1: Worst-case and best-case not asymptotically equal
  - ➔ Use Θ-notation to bound worst-case and best-case runtimes separately

- Case 2: Worst-case and best-case asymptotically equal
  - ➔ Use Θ-notation to bound the runtime for any input

Using Θ-Notation to Describe Running Times
Case 1

- Case 1: Worst-case and best-case not asymptotically equal
  - ➔ Use Θ-notation to bound the worst-case and best-case runtimes separately

- We can say:
  - "The worst-case runtime of insertion sort is $\Theta(n^2)$"
  - "The best-case runtime of insertion sort is $\Theta(n)$"

- But, we can't say:
  - "The runtime of insertion sort is $\Theta(n^2)$ for every input"

Using Θ-Notation to Describe Running Times Case 2

- Case 2: Worst-case and best-case asymptotically equal
  - ➔ Use Θ-notation to bound the runtime for any input

- e.g. For merge-sort, we have:

$$\left.\begin{array}{l} T(n) = O(n\lg n) \\ T(n) = \Omega(n\lg n) \end{array}\right\} \quad T(n) = \Theta(n\lg n)$$

Using Asymptotic Notation to Describe Runtimes Summary

- "The worst case runtime of Insertion Sort is $O(n^2)$"
  - Also implies: "The runtime of Insertion Sort is $O(n^2)$"
- "The best-case runtime of Insertion Sort is $\Omega(n)$"
  - Also implies: "The runtime of Insertion Sort is $\Omega(n)$"
- "The worst case runtime of Insertion Sort is $\Theta(n^2)$"
  - But: "The runtime of Insertion Sort is not $\Theta(n^2)$"
- "The best case runtime of Insertion Sort is $\Theta(n)$"
  - But: "The runtime of Insertion Sort is not $\Theta(n)$"

Using Asymptotic Notation to Describe Runtimes Summary

- “The <u>worst case</u> runtime of Merge Sort is $\Theta(nlgn)$”
- “The <u>best case</u> runtime of Merge Sort is $\Theta(nlgn)$”
- “The runtime of Merge Sort is $\Theta(nlgn)$”
  - *This is true, because the best and worst case runtimes have asymptotically the same tight bound* $\Theta(nlgn)$

Asymptotic Notation in Equations

- Asymptotic notation appears alone on the RHS (right hand side) of an equation:
  - implies set membership

    e.g., $n = O(n^2)$ means $n \in O(n^2)$

- Asymptotic notation appears on the RHS of an equation
  - stands for some anonymous function in the set e.g., $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ means:

    $2n^2 + 3n + 1 = 2n^2 + h(n)$, for some $h(n) \in \Theta(n)$

    i.e., $h(n) = 3n + 1$

Asymptotic Notation in Equations

- Asymptotic notation appears on the LHS of an equation:
  - stands for any anonymous function in the set

    e.g., $2n^2 + \Theta(n) = \Theta(n^2)$ means:

    for any function $g(n) \in \Theta(n)$

    $\exists$ some function $h(n) \in \Theta(n^2)$

    such that $2n^2+g(n) = h(n)$

- RHS provides coarser level of detail than LHS

Source: https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/e/quiz--asymptotic-notation

Quiz: Asymptotic notation

For the functions, n^k and c^n, what is the asymptotic relationship between these functions?

Assume that k >= 1 and c > 1 are constants.

Choose all answers that apply:

- (A) n^k is $O(c^n)$
- (B) n^k is $\Omega(c^n)$
- (C) n^k is $\Theta(c^n)$

Quiz: Asymptotic notation

For the functions, n^k and c^n, what is the asymptotic relationship between these functions?

Assume that k >= 1 and c > 1 are constants.

Choose all answers that apply:

CORRECT (SELECTED)
n^k is $O(c^n)$

INCORRECT
n^k is $\Omega(c^n)$

INCORRECT
n^k is $\Theta(c^n)$

Quiz: Asymptotic notation

For the functions, $\lg n$ and $\log_8 n$, what is the asymptotic relationship between these functions?

Choose all answers that apply:

(A) $\lg n$ is $O(\log_8 n)$

(B) $\lg n$ is $\Omega(\log_8 n)$

(C) $\lg n$ is $\Theta(\log_8 n)$

Quiz: Asymptotic notation

For the functions, $\lg n$ and $\log_8 n$, what is the asymptotic relationship between these functions?

Choose all answers that apply:

CORRECT (SELECTED)

$\lg n$ is $O(\log_8 n)$

CORRECT (SELECTED)

$\lg n$ is $\Omega(\log_8 n)$

CORRECT (SELECTED)

$\lg n$ is $\Theta(\log_8 n)$

Quiz: Asymptotic notation

What is the asymptotic relationship between the functions $n^3 \lg n$ and $3n \log_8 n$?

Choose all answers that apply:

(A) $n^3 \lg n$ is $O(3n \log_8 n)$

(B) $n^3 \lg n$ is $\Omega(3n \log_8 n)$

(C) $n^3 \lg n$ is $\Theta(3n \log_8 n)$

Quiz: Asymptotic notation

What is the asymptotic relationship between the functions $n^3 \lg n$ and $3n \log_8 n$?

Choose all answers that apply:

INCORRECT

$n^3 \lg n$ is $O(3n \log_8 n)$

CORRECT (SELECTED)

$n^3 \lg n$ is $\Omega(3n \log_8 n)$

INCORRECT

$n^3 \lg n$ is $\Theta(3n \log_8 n)$

Quiz: Asymptotic notation

For the functions, 8^n and 4^n, what is the asymptotic relationship between these functions?

Choose all answers that apply:

- (A) 8^n is $O(4^n)$
- (B) 8^n is $\Omega(4^n)$
- (C) 8^n is $\Theta(4^n)$

Quiz: Asymptotic notation

For the functions, 8^n and 4^n, what is the asymptotic relationship between these functions?

Choose all answers that apply:

INCORRECT

8^n is $O(4^n)$

CORRECT (SELECTED)

8^n is $\Omega(4^n)$

INCORRECT

8^n is $\Theta(4^n)$

$$\lg a^b = b \lg a$$

Quiz: Asymptotic notation

For the functions, $\lg n^{\lg 17}$ vs. $\lg 17^{\lg n}$, what is the asymptotic relationship between these functions?

Choose all answers that apply:

- (A) $\lg n^{\lg 17}$ is $O(\lg 17^{\lg n})$
- (B) $\lg n^{\lg 17}$ is $\Omega(\lg 17^{\lg n})$
- (C) $\lg n^{\lg 17}$ is $\Theta(\lg 17^{\lg n})$

Quiz: Asymptotic notation

For the functions, $\lg n^{\lg 17}$ vs. $\lg 17^{\lg n}$, what is the asymptotic relationship between these functions?

Choose all answers that apply:

CORRECT (SELECTED)
$\lg n^{\lg 17}$ is $O(\lg 17^{\lg n})$

CORRECT (SELECTED)
$\lg n^{\lg 17}$ is $\Omega(\lg 17^{\lg n})$

CORRECT (SELECTED)
$\lg n^{\lg 17}$ is $\Theta(\lg 17^{\lg n})$