Computational and Asymptotic Complexity

Lesson 3.1

Learning Objectives

LO 3.1.1 **Define** Big O (O), Omega (Ω), Theta (Θ) notations

LO 3.1.2 **Differentiate** computational from asymptotic complexity

LO 3.1.3 **Prove/disprove** asymptotic complexity assumptions of a function/algorithm

Computational Complexity

- Developed by **Juris Hartmanis** and **Richard Stearns**
- Used to measure the degree of difficulty of an algorithm

Computational Complexity

To evaluate an algorithm's efficiency, real-time units such as microseconds and nanoseconds should not be used.

Rather, logical units that express a relationship between the size n of a file or an array and the amount of time t required to process the data should be used.

Computational Complexity

If there is a linear relationship between the size n and time t, i.e., $t_1 = cn_1$, then an increase of data by a factor of 5 results in the increase of the execution time by the same factor; if $n_2 = 5n_1$, then $t_2 = 5t_1$.

Similarly, if $t_1 = \log_2 n$, then doubling n increases t by only one unit of time.

Therefore, if $t_2 = \log_2(2n)$, then $t_2 = t_1 + 1$.

Asymptotic Complexity

A function expressing the relationship between n and t is usually much more complex, and calculating such a function is important only in regard to large bodies of data.

A measure of efficiency called **asymptotic complexity** is used when disregarding certain terms of a function to express the efficiency of an algorithm or when calculating a function is difficult or impossible and only approximations can be found.

Asymptotic Complexity

The growth rate of all terms of function $f(n) = n^2 + 100n + \log_{10} n + 1{,}000$

n	f(n)	n^2		100n		$\log_{10}$ n		1000	
	Value	Value	%	Value	%	Value	%	Value	%
1	1,101	1	0.1	100	9.100	9	0.000	1,000	90.830
10	2,101	100	4.76	1,000	47.600	1	0.050	1,000	47.600
100	21,002	10,000	47.6	10,000	47.600	2	0.001	1,000	4.760
1,000	1,101,003	1,000,000	90.8	100,000	9.100	3	0.000	1,000	0.090
10,000	101,001,004	100,000,000	99.0	1,000,000	0.990	4	0.000	1,000	0.001
100,000	10,010,001,005	10,000,000,000	99.9	10,000,000	0.099	5	0.000	1,000	0.000

Big O Notation

Definition

$f(n)$ is $\mathrm{O}(g(n))$ if there exist positive numbers c and N such that $f(n) \leq c \cdot g(n)$ for all $n \geq N$.

The relationship between f and g can be expressed by stating either

- $c \cdot g(n)$ is an upper bound on the value of $f(n)$
- in the long run, f grows at most as fast as g.

Big O Notation

Example: Is $3n^2 + 5n + 100$, $O(n^2)$?

Definition

$f(n)$ is $O(g(n))$ if there exist positive numbers c and N such that $f(n) \leq c \cdot g(n)$ for all $n \geq N$.

Solution

Let N = 15, c = 4, 5, 6, 7, ...

$3n^2 + 5n + 100 \leq 4n^2$ $\qquad$ $3n^2 + 5n + 100 \leq 6n^2$ $\qquad$ and so on...

$3n^2 + 5n + 100 \leq 5n^2$ $\qquad$ $3n^2 + 5n + 100 \leq 7n^2$

Big O Notation

Example:

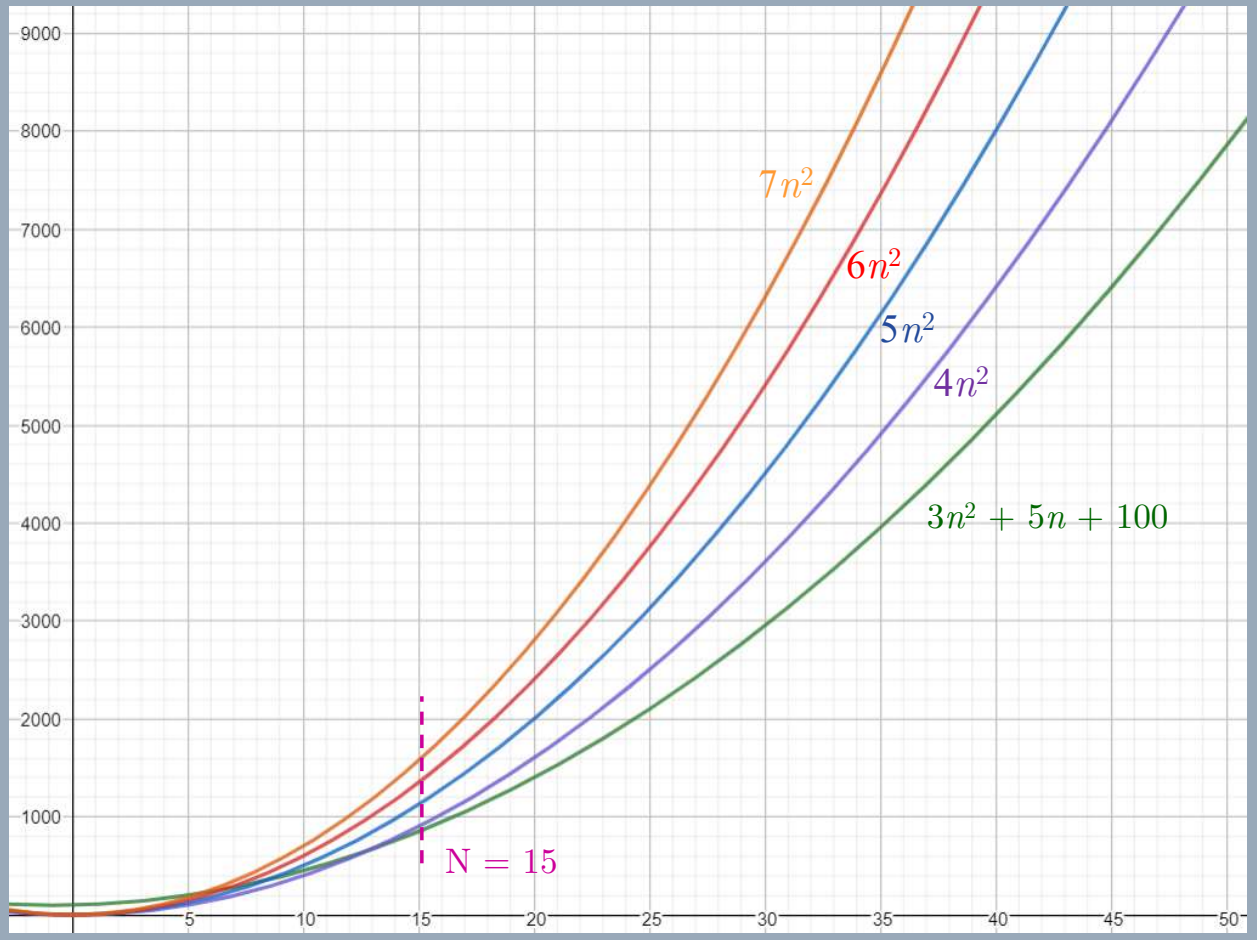
$3n^2 + 5n + 100 \overset{?}{\text{ is }} \mathrm{O}(n^2)$

$f(n) = 3n^2 + 5n + 100$

$g(n) = n^2$

$c \cdot g(n)$ always upper bounds $f(n)$

$\therefore \boldsymbol{f(n) \text{ is } \mathrm{O}(n^2)}$



Big Ω Notation

Definition

$f(n)$ is $\Omega(g(n))$ if there exist positive numbers c and N such that $f(n) \geq c \cdot g(n)$ for all $n \geq N$.

The relationship between f and g can be expressed by stating either

- $c \cdot g(n)$ is a lower bound on the size of $f(n)$
- in the long run, f grows at least the rate of g.

Big Ω Notation

Example: Is $3\log(n)+10$, $\Omega(\log(n))$?

Definition

$f(n)$ is $\Omega(g(n))$ if there exist positive numbers c and N such that $f(n) \geq c \cdot g(n)$ for all $n \geq N$.

Solution

Let N = 20, c = 1, 2, 3

$3\log(n)+10 \geq 3\log(n)$ $\qquad$ $3\log(n)+10 \geq \log(n)$

$3\log(n)+10 \geq 2\log(n)$

Big Ω Notation

Example:

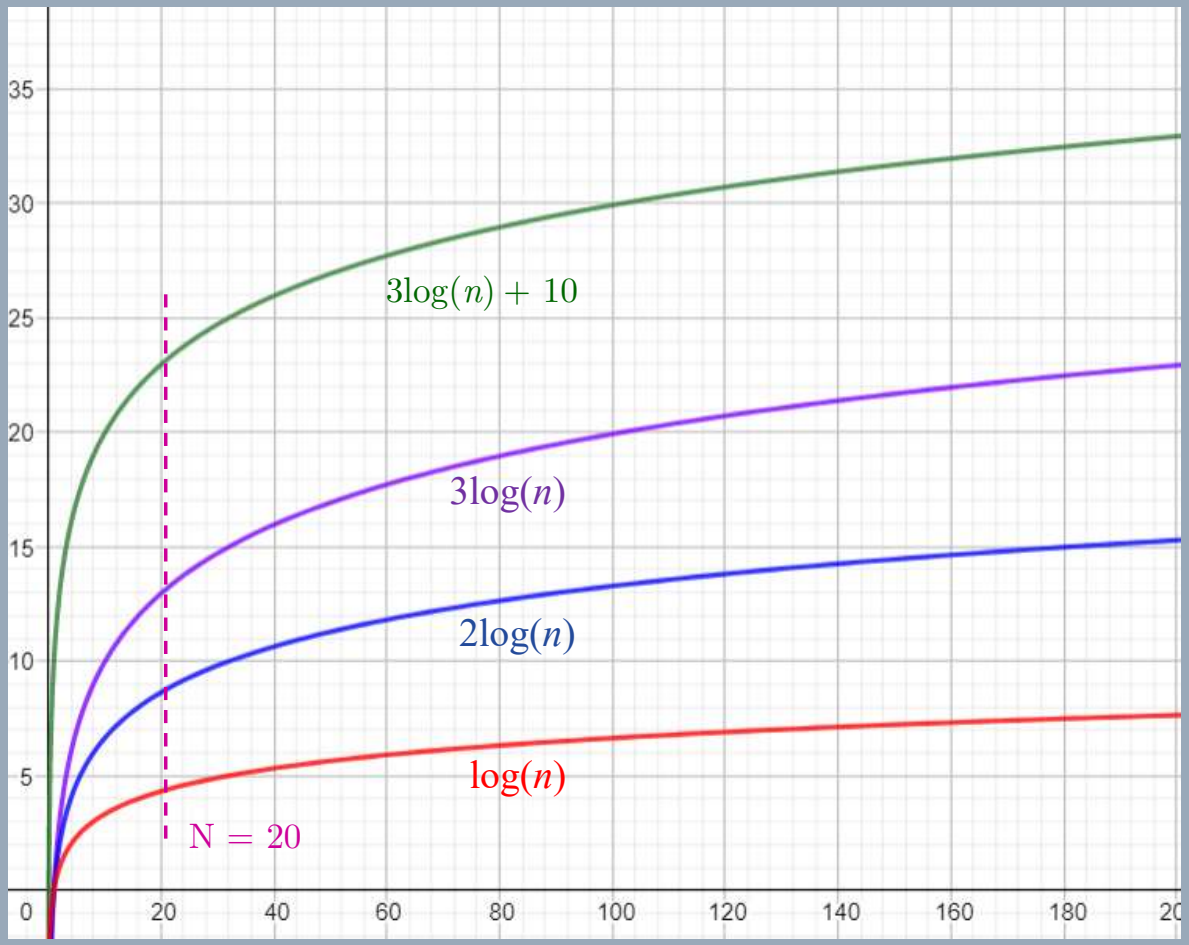
$3\log(n)+10 \overset{?}{\text{is}} \Omega(\log(n))$

$f(n) = 3\log(n)+10$

$g(n) = \log(n)$

$c \cdot g(n)$ lower bounds $f(n)$

$\therefore$ **$f(n)$ is $\Omega(\log(n))$**



Big Θ Notation

Definition

$f(n)$ is $\Theta(g(n))$ if there exist positive numbers c_1, c_2, and N such that $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ for all $n \geq N$.

The relationship between f and g can be expressed by stating either

- f has an order of magnitude g
- in the long run, f grows at the same rate as g.

Big Θ Notation

Example: Is $3{\cdot}2^n + 3n^3 + 5000$, $\Theta(2^n)$? $\Theta(n^3)$?

Definition

$f(n)$ is $\Theta(g(n))$ if there exist positive numbers c_1, c_2, and N such that $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ for all $n \geq N$.

Solution

Let $\mathrm{N} = 14$, $\mathrm{c}_1 = 3$, $\mathrm{c}_2 = 4$

$3{\cdot}2^n \leq 3{\cdot}2^n + 3n^3 + 5000$

$3{\cdot}2^n + 3n^3 + 5000 \leq 4{\cdot}2^n$

$3n^3 \leq 3{\cdot}2^n + 3n^3 + 5000$

$3{\cdot}2^n + 3n^3 + 5000 \nleq 4n^3$

Big Θ Notation

Example:

$3{\cdot}2^n + 3n^3 + 5000 \overset{?}{\text{ is }} \Theta(2^n)$

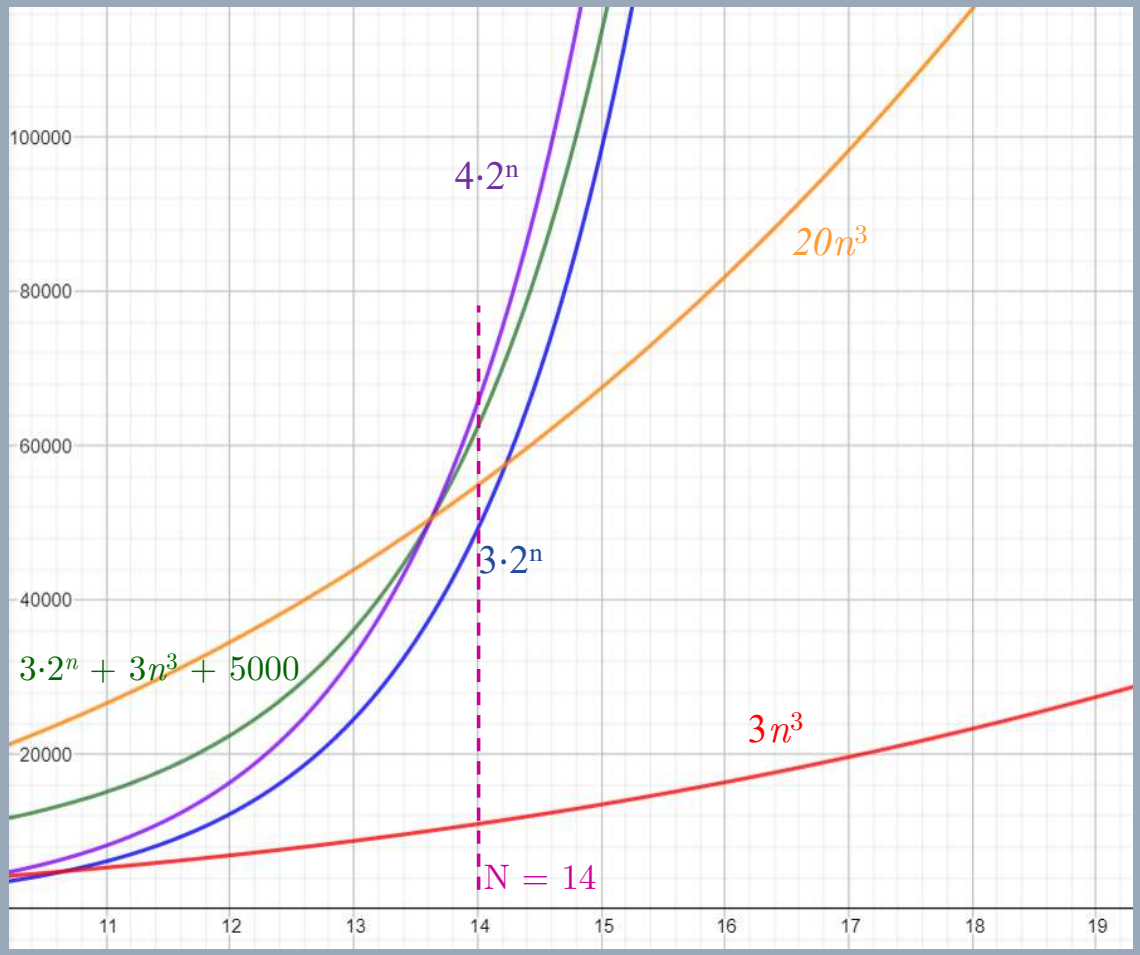
$3{\cdot}2^n + 3n^3 + 5000 \overset{?}{\text{ is }} \Theta(n^3)$

$f(n) = 3n^2 + 5n + 100$

$g(n) = 2^n$

$\therefore$ **$f(n)$ is $\Theta(2^n)$**

$\therefore$ **$f(n)$ is not $\Theta(n^3)$**



Strengthening the Learning Objectives

LO 3.1.1 Define Big O (O), Omega (Ω), Theta (Θ) notations

Match the following in column A with the definitions in column B:

A	B
1. $f(n)$ is $O(g(n))$	if there exist positive numbers c_1, c_2, and N such that $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ for all $n \geq N$.
2. $f(n)$ is $\Omega(g(n))$	if there exist positive numbers c and N such that $f(n) \geq c \cdot g(n)$ for all $n \geq N$.
3. $f(n)$ is $\Theta(g(n))$	if there exist positive numbers c and N such that $f(n) \leq c \cdot g(n)$ for all $n \geq N$.

LO 3.1.2 Differentiate computational from asymptotic complexity

Example:

1. Proving an algorithm runs on linear time in any case of n.
2. Modeling the exact space requirements that needs to be allocated for a procedure processing n items.
3. Finding the worst case time complexity of a function.

LO 3.1.3 Prove/disprove asymptotic complexity assumptions of a function/algorithm

Example:

1. $7\mathrm{n}^2 + 4\mathrm{n}$ is $\mathrm{O}(n)$
2. $n! + 2^n$ is $\Omega(n!)$
3. $8n^2 + 4n + 2$ is $\Theta(n^3)$.
4. $f(n) + g(n)$ is $\Theta(\min(f(n), g(n)))$