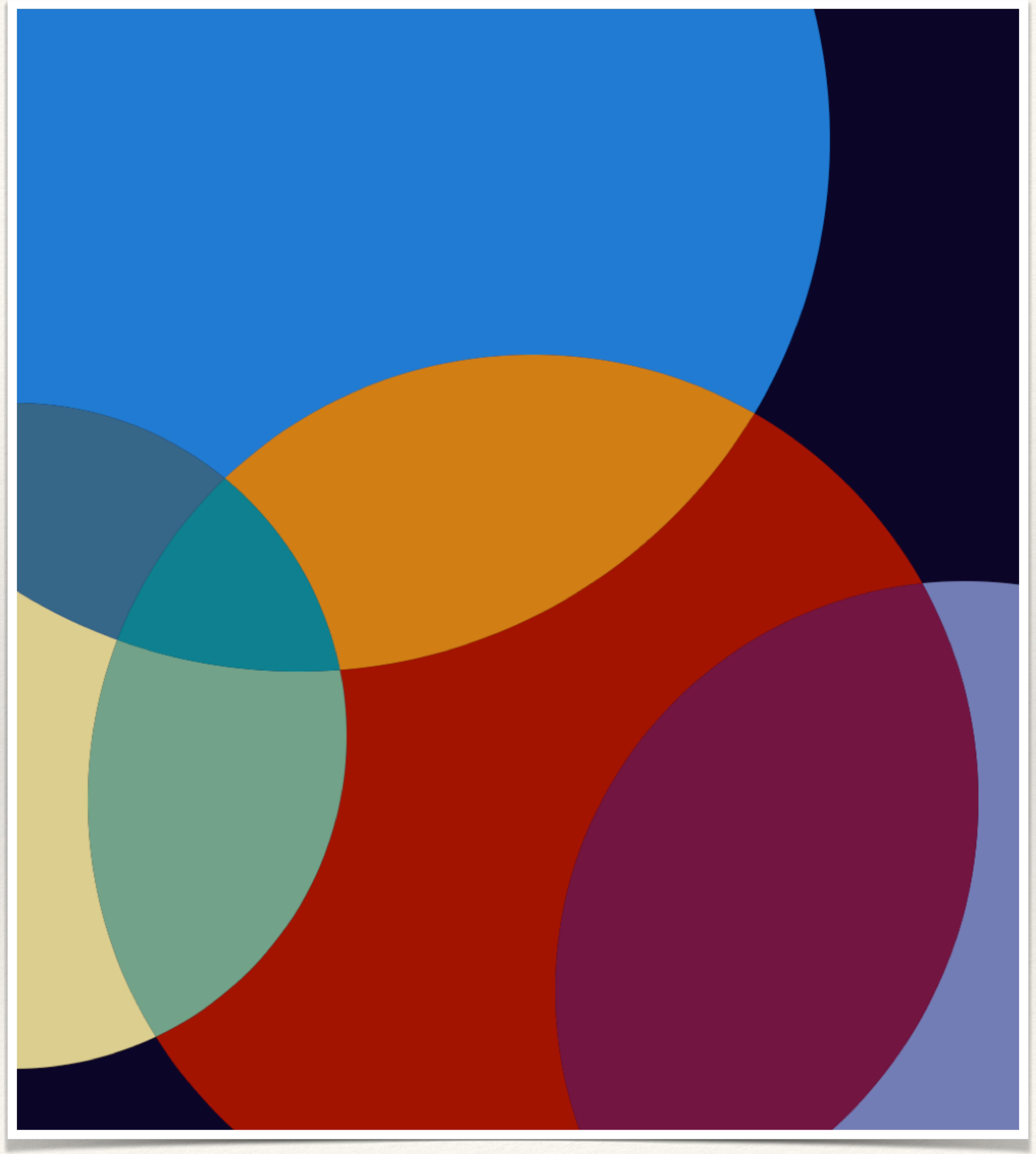
Lecture 13

Growth of Functions

Dr. David Zmiaikou



Why to Study Growth of Functions?

- Ordinarily, there are several algorithms that give a solution to a particular problem.

Why to Study Growth of Functions?

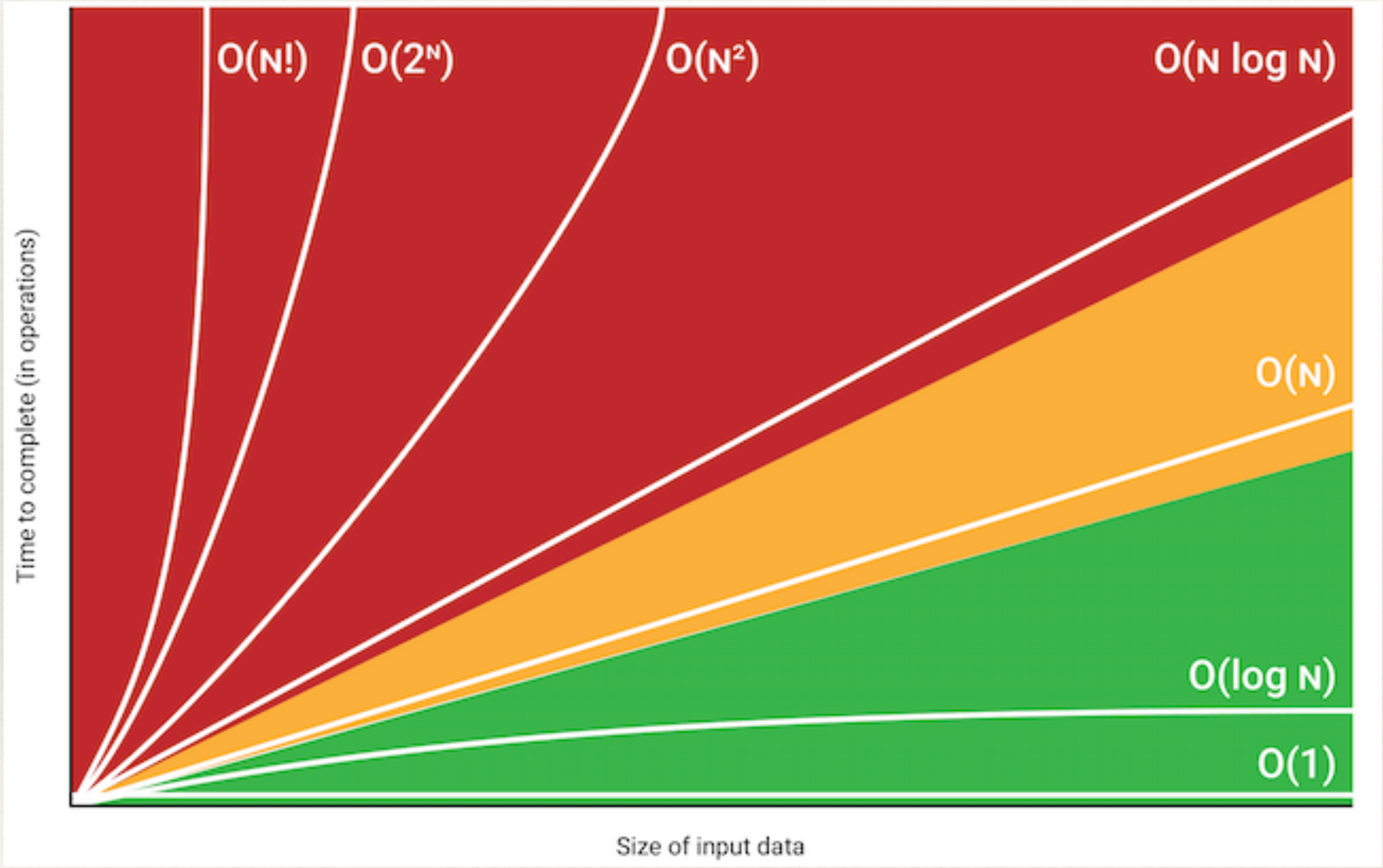
- Ordinarily, there are several algorithms that give a solution to a particular problem.
- Given two algorithms that provide a solution to the same problem, it is natural to ask whether one of these algorithms might be preferred over the other.

Why to Study Growth of Functions?

- Ordinarily, there are several algorithms that give a solution to a particular problem.
- Given two algorithms that provide a solution to the same problem, it is natural to ask whether one of these algorithms might be preferred over the other.
- In order to study this question, we describe a common method of comparing the growth of two functions f and g defined on the set $\mathbb{N}$ of positive integers and whose values are positive real numbers.

Big-O of a Function

$\stackrel{\text{def}}{=}$ What is Big-O?



- A function $f : \mathbb{N} \to \mathbb{R}^+$ is **big-O** (or **big-oh**) of a function $g : \mathbb{N} \to \mathbb{R}^+$,

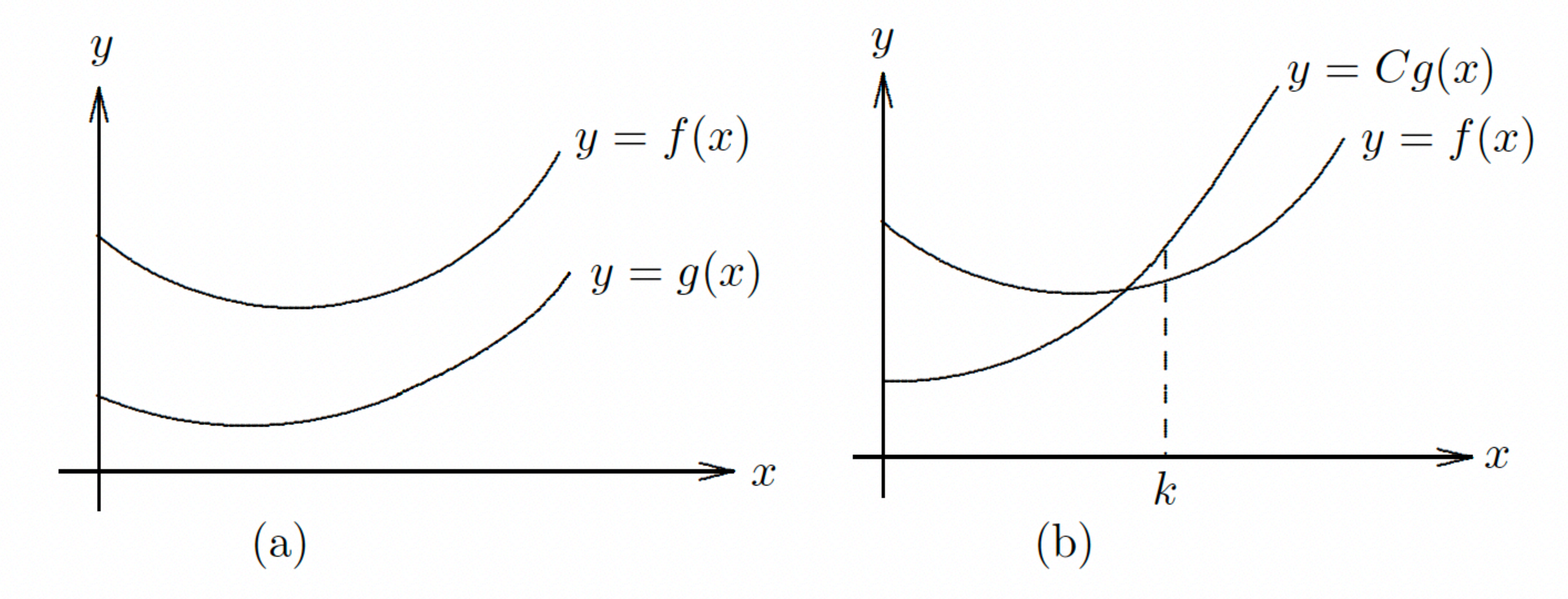
$$\text{written} \quad f = O(g) \quad \text{or} \quad f(n) = O(g(n)),$$

if there exist a positive constant C and a positive integer k such that

$$\boldsymbol{f(n) \leq C \cdot g(n)} \quad \text{for every integer} \quad n \geq k.$$

What is Big-O?

- If $f(n) = O(g(n))$, then for *large values* of n, the function $f(n)$ grows no faster than a constant times the function $g(n)$.



⌛ A bit of history…

- Edmund Georg Hermann Landau (14 February 1877 – 19 February 1938) was a German mathematician.

- Big O is a notation invented by Paul Bachmann, Edmund Landau, and others, collectively called **Bachmann–Landau notation** or **asymptotic notation**. The letter O was chosen by Bachmann to stand for *Ordnung*, meaning the order of approximation.

- **Paul Gustav Heinrich Bachmann** (22 June 1837 – 31 March 1920) was a German mathematician.

Example 1

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = 2n^2 \text{ and } g(n) = n^3 \text{ for all } n \in \mathbb{N}.$$

Show that $f = O(g)$.

Example 1

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = 2n^2 \text{ and } g(n) = n^3 \text{ for all } n \in \mathbb{N}.$$

Show that $f = O(g)$.

Solution. For $n \geq 1$, we have

$$n^2 \leq n^3 \quad \text{and so} \quad 2n^2 \leq 2n^3.$$

Therefore, $2n^2 \leq 2(n^3)$ for all $n \geq 1$. Hence

$$f = O(g), \text{ where } C = 2 \text{ and } k = 1$$

in the definition.

$$2n^2 \leq 1 \cdot n^3 \text{ for all } n \geq 2$$

Example 2

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = 3n^2 + 6 \text{ and } g(n) = n^3 + n \text{ for all } n \in \mathbb{N}.$$

Show that $f = O(g)$.

Example 2

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = 3n^2 + 6 \text{ and } g(n) = n^3 + n \text{ for all } n \in \mathbb{N}.$$

Show that $f = O(g)$.

Solution. For $n \geq 1$, we have $n^2 \leq n^3$ and so $3n^2 \leq 3n^3$.

Also, for $n \geq 2$, we have $6 \leq 3n$.

Therefore,

$$3n^2 + 6 \leq 3(n^3 + n) \quad \text{for all } n \geq 2.$$

Hence

$$f = O(g), \text{ where } C = 3 \text{ and } k = 2$$

in the definition.

Example 3

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = n \text{ and } g(n) = n^2 \text{ for all } n \in \mathbb{N}.$$

Show that $f = O(g)$ but $g \neq O(f)$.

Example 3

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = n \text{ and } g(n) = n^2 \text{ for all } n \in \mathbb{N}.$$

Show that $f = O(g)$ but $g \neq O(f)$.

Solution. For $n \geq 1$, we have $n \leq 1 \cdot n^2$ and so $f = O(g)$ by definition with $C = 1$ and $k = 1$.

On the other hand $g \neq O(f)$. Indeed, suppose, to the contrary, that $g = O(f)$. Then there exist a positive constant C and a positive integer k such that

$$n^2 \leq C \cdot n \text{ for all } n \geq k.$$

Dividing this inequality by n, we have that $n \leq C$ for all $n \geq k$. However, $n \leq C$ is not satisfied if we choose n to be an integer that is both greater than C and greater than k. Hence, $g \neq O(f)$.

Properties of Big-O

- **Theorem 1**. If $f_1 = O(g_1)$ and $f_2 = O(g_2)$, then

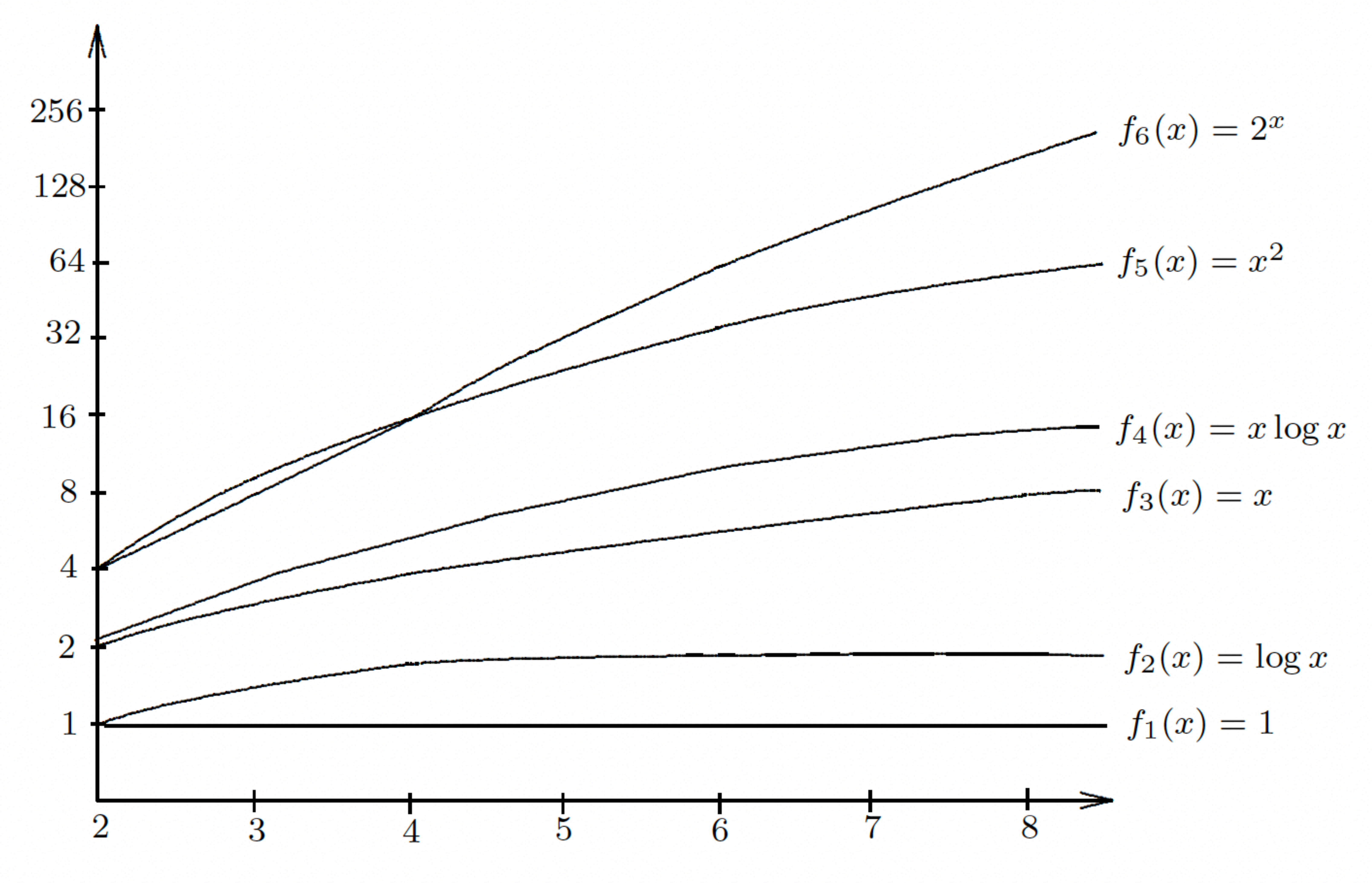
$$f_1 \cdot f_2 = O(g_1 \cdot g_2),$$

$$f_1 + f_2 = O(\max(g_1, g_2)).$$

If $k > 0$ is a constant, then $O(k \cdot g) = O(g)$.

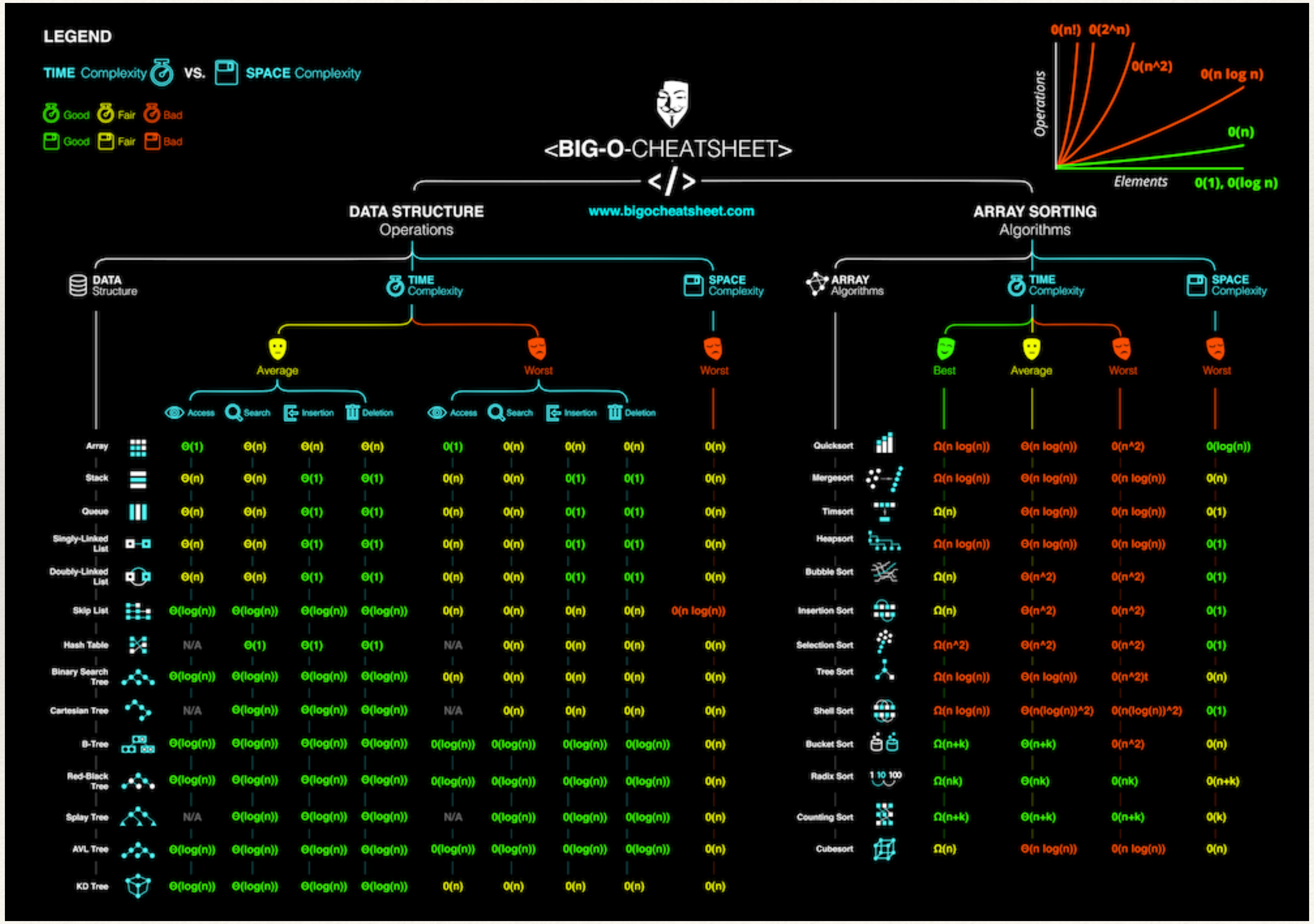
Big-O and Time Complexity

Graphs of Classical Functions



def What is Complexity of an Algorithm?

- The **complexity of an algorithm** is the amount of *space* and *time* needed to execute the algorithm.

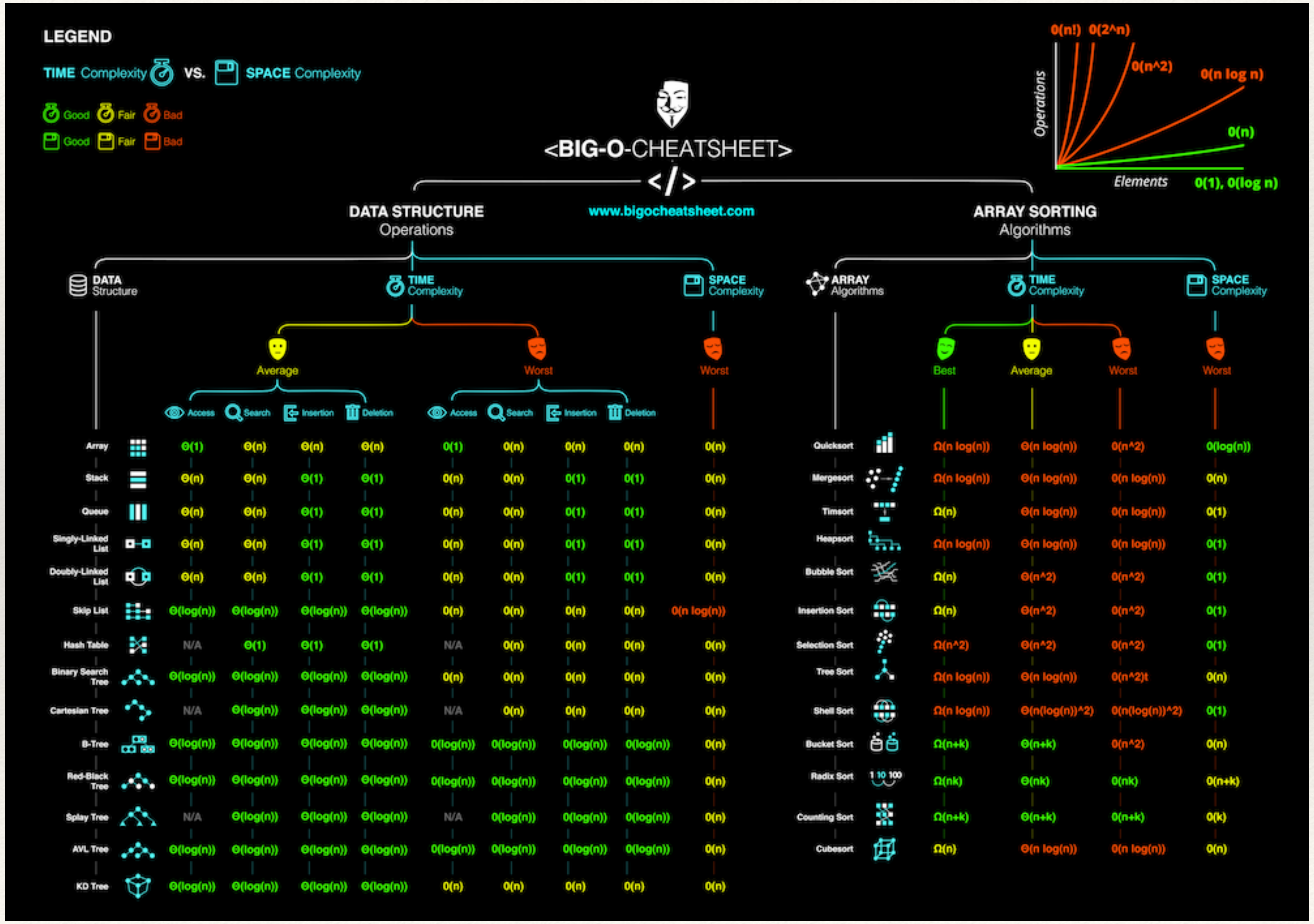


DATA Structure	TIME Complexity Average Access	Average Search	Average Insertion	Average Deletion	Worst Access	Worst Search	Worst Insertion	Worst Deletion	SPACE Complexity Worst
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

ARRAY Algorithms	TIME Complexity Best	Average	Worst	SPACE Complexity Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(1)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)t	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

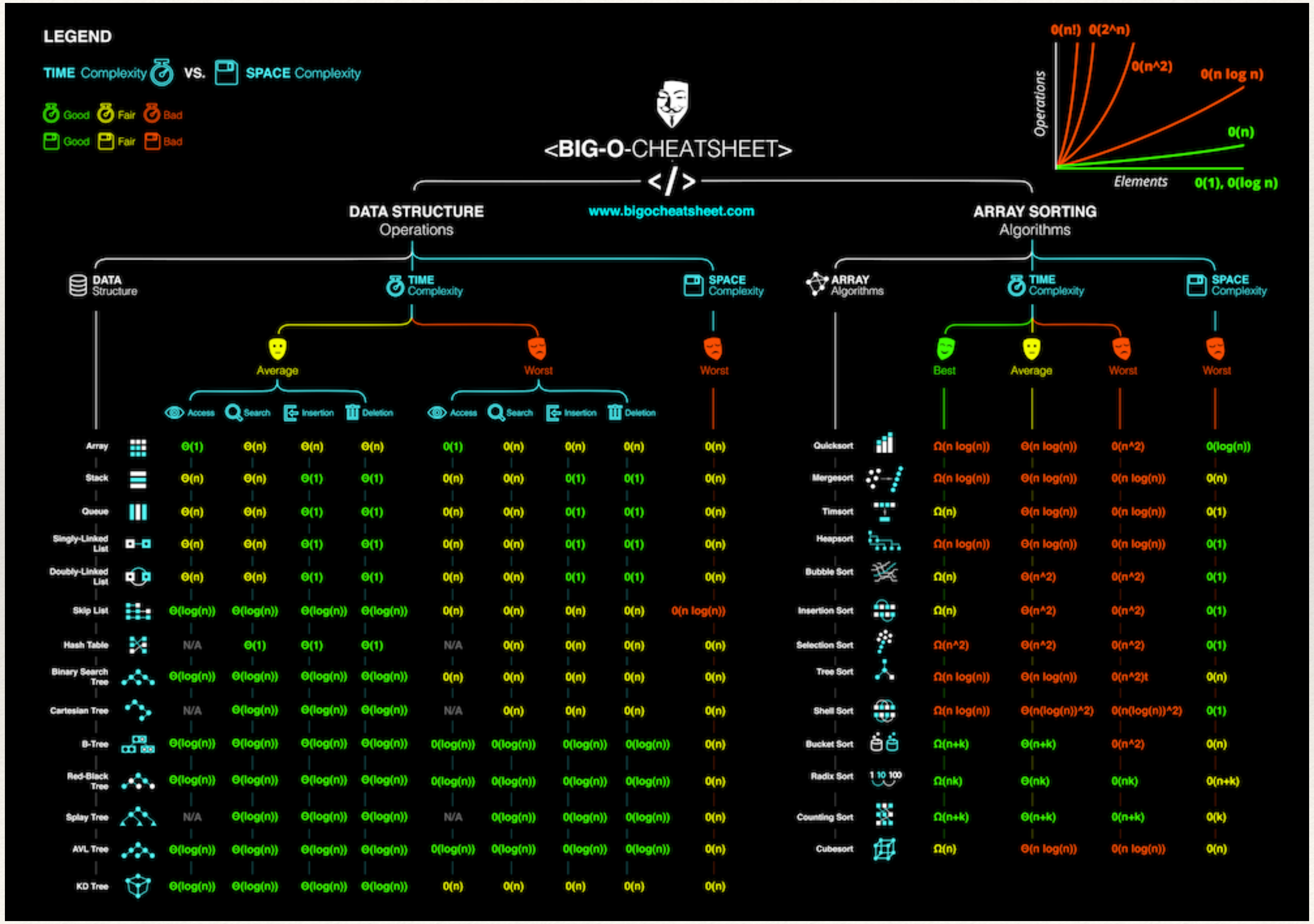
def What is Complexity of an Algorithm?

- The **complexity of an algorithm** is the amount of space and time needed to execute the algorithm.
- The **space complexity** of an algorithm concerns a study of computer memory and the data structures employed.



def What is Complexity of an Algorithm?

- The **complexity of an algorithm** is the amount of space and time needed to execute the algorithm.
- The **space complexity** of an algorithm concerns a study of computer memory and the data structures employed.
- The **time complexity** of an algorithm concerns a study of the time required to solve the problem using the algorithm as a function of the size of the input.

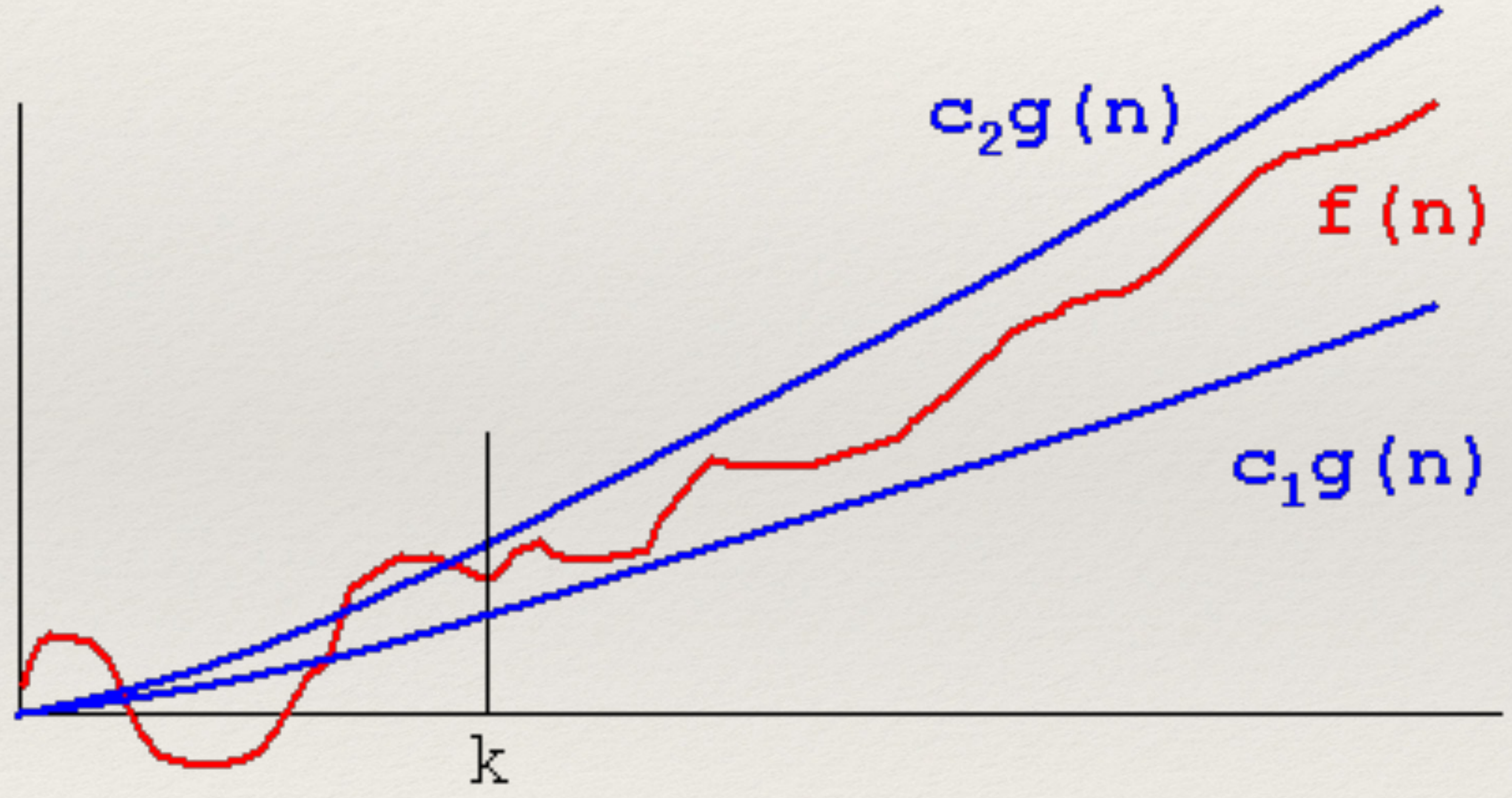


Time Complexity

Name	Notation	Examples
constant	$O(1)$	Determining if a binary number is even or odd.
double logarithmic	$O(\log \log n)$	Average number of comparisons spent finding an item using interpolation search in a sorted array of uniformly distributed values.
logarithmic	$O(\log n)$	Finding an item in a sorted array with a binary search. Balanced search tree.
polylogarithmic	$O((\log n)^c)$, $c > 1$	Matrix chain ordering can be solved in polylogarithmic time on a parallel random-access machine.
linear	$O(n)$	Finding an item in an unsorted list or in an unsorted array.
loglinear, or "n log n"	$O(n \log n)$	Performing a fast Fourier transform. Fastest possible comparison sort. Heapsort and merge sort.
quadratic	$O(n^2)$	Multiplying two n-digit numbers by schoolbook multiplication. Bubble sort, selection sort and insertion sort. Tree sort.
polynomial or algebraic	$O(n^c)$	Maximum matching for bipartite graphs. Finding the determinant of a matrix with LU-decomposition.
exponential	$O(c^n)$, $c > 1$	Finding the solution to the travelling salesman problem using dynamic programming; determining if two logical statements are equivalent using brute-force.
factorial	$O(n!)$	Solving the travelling salesman problem via brute-force search. Finding the determinant with Laplace expansion. Enumerating all partitions of a set.

Big-Θ of a Function

$\stackrel{\text{def}}{=}$ What is Big-Θ?



- A function $f : \mathbb{N} \to \mathbb{R}^+$ is **big-theta** of a function $g : \mathbb{N} \to \mathbb{R}^+$,

  written $f = \Theta(g)$ or $f(n) = \Theta(g(n))$,

  if there exist *positive* constants C_1 and C_2 and a positive integer k such that

  $$\boldsymbol{C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)}$$

  for every integer $n \geq k$.

What is Big-Θ?

- When $f = \Theta(g)$, we say that f and g grow at the same rate.

Example 4

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = 2n^2 + 6 \text{ and } g(n) = 3n^2 + 3n \text{ for all } n \in \mathbb{N}.$$

Show that $f = \Theta(g)$.

Example 4

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = 2n^2 + 6 \text{ and } g(n) = 3n^2 + 3n \text{ for all } n \in \mathbb{N}.$$

Show that $f = \Theta(g)$.

Solution. For $n \geq 3$, we have

$$2n^2 + 6 \leq n^2 + 2n = \frac{2}{3}(3n^2 + 3n).$$

Also, for $n \geq 1$, we have

$$2n^2 + 6 = n^2 + n^2 + 6 \geq n^2 + n = \frac{1}{3}(3n^2 + 3n).$$

Therefore, $\frac{1}{3}(3n^2 + 3n) \leq 2n^2 + 6 \leq \frac{2}{3}(3n^2 + 3n)$ for all $n \geq 3$.

Hence, $f = \Theta(g)$, where $C_1 = \frac{1}{3}$ and $C_2 = \frac{2}{3}$ and $k = 3$ in the definition.

Theorems for Big-Θ

- **Theorem 2**. If f and g are two polynomial functions of the same degree, then $f = \Theta(g)$.

Theorems for Big-Θ

- **Theorem 2**. If f and g are two polynomial functions of the same degree, then $f = \Theta(g)$.

- **Theorem 3**. Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions. Then $f = \Theta(g)$ if and only if $f = O(g)$ and $g = O(f)$.

Example 5

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = n^2 + 3n + 2 \text{ and } g(n) = 5n^2 \text{ for all } n \in \mathbb{N}.$$

Show that $f = \Theta(g)$.

Example 5

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = n^2 + 3n + 2 \text{ and } g(n) = 5n^2 \text{ for all } n \in \mathbb{N}.$$

Show that $f = \Theta(g)$.

First solution. Follows directly from Theorem 2.

Example 5

Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions defined by

$$f(n) = n^2 + 3n + 2 \text{ and } g(n) = 5n^2 \text{ for all } n \in \mathbb{N}.$$

Show that $f = \Theta(g)$.

Second solution. For $n \geq 1$, we have

$$n^2 + 3n + 2 \leq n^2 + 3n^2 + 2n^2 = 6n^2 = \frac{6}{5}(5n^2).$$

Thus $f = O(g)$. Also for $n \geq 1$, we have

$$5n^2 \leq 5 \cdot (n^2 + 3n + 2),$$

and so $g = O(f)$. Hence, $f = \Theta(g)$ according to Theorem 3.

Theorems for Big-Θ

- **Theorem 4**. Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions. If $f = \Theta(g)$, then $g = \Theta(f)$.

Theorems for Big-Θ

- **Theorem 4**. Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions. If $f = \Theta(g)$, then $g = \Theta(f)$.

- **Theorem 5**. Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be two functions. If
$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = c \quad \text{for some positive real number } c,$$
then $f = \Theta(g)$.

Converse for Theorem 5?

The converse of Theorem 5 is not true. Could you construct an example?

Converse for Theorem 5?

The converse of Theorem 5 is not true. Could you construct an example?

Let $f(n) = 1$ for all natural n and $g(n) = 1$ if n is odd and 2 if n is even.

Then $f = \Theta(g)$ by definition,

but $\lim_{n \to +\infty} \frac{f(n)}{g(n)}$ $(= 1, 1/2, 1, 1/2, \ldots)$ doesn't exist.

Thank you!