ensia The National School of Artificial Intelligence
المدرسة الوطنية العليا للذكاء الاصطناعي

Data Structures and Algorithms 2

Chapter 2
Algorithm Analysis

Dr. Fouzia ANGUEL

2nd year / S3

September 2024 – January 2025

Course Outline

- Algorithms efficiency
  - Machine-dependent vs Machine-independent
- Function ordering
  - Order of growth
  - Weak Order;
  - Landau symbols Big-Oh ; Big-omega ; big theta and Little-oh (asymptotically <).
- Algorithm complexity analysis
  - Rules for complexity analysis
  - Analysis of various types of algorithms
  - Master Theorem

Algorithm Efficiency

Example : Shortest path problem

- A city has *n* view points
- Buses move from one view point to another
- A bus driver wishes to follow the shortest path (travel time wise).
- Every view point is connected to another by a road.
- However, some roads are less congested than others.
- Also, roads are one-way, i.e., the road from view point 1 to 2, is different from that from view point 2 to 1.

Algorithm Efficiency

Example : Shortest path problem

How to find the shortest path between any two pairs?

➔ **Naïve approach**

- List all the paths between a given pair of view points
- Compute the travel time for each.
- Choose the shortest one.

How many paths are there between any two view points (without revisits)?

$$n! \cong (n/e)^n$$

➔ It will be impossible to run the algorithm for n = 30

Algorithm efficiency

- Run time in the computer is Machine dependent

Example : Need to multiply two positive integers a and b

Subroutine 1: Multiply a and b

Subroutine 2: V = a, W = b

While W > 1

V →V + a; W →W-1

Output V

Algorithm efficiency

First subroutine has 1 multiplication.

Second has b additions and subtractions.

For some architectures, 1 multiplication is more expensive than b additions and subtractions.

Ideally, we would like to program all choices and run all of them in the machine we are going to use and find which is efficient!

Machine Independent Analysis

We assume that every basic operation takes constant time

Example **Basic** Operations:

Addition, Subtraction, Multiplication, Memory Access

Non-basic Operations:

Sorting, Searching

Efficiency of an algorithm is the number of basic operations it performs

We do not distinguish between the basic operations.

Subroutine 1 uses 1 basic operation (*)

Subroutine 2 uses 2b basic operations (+, -)

Subroutine **1** is more efficient.

This measure is good for all large input sizes

In fact, we will not worry about the exact values, but will look at "broad classes" of values.

Let there be **n inputs**.

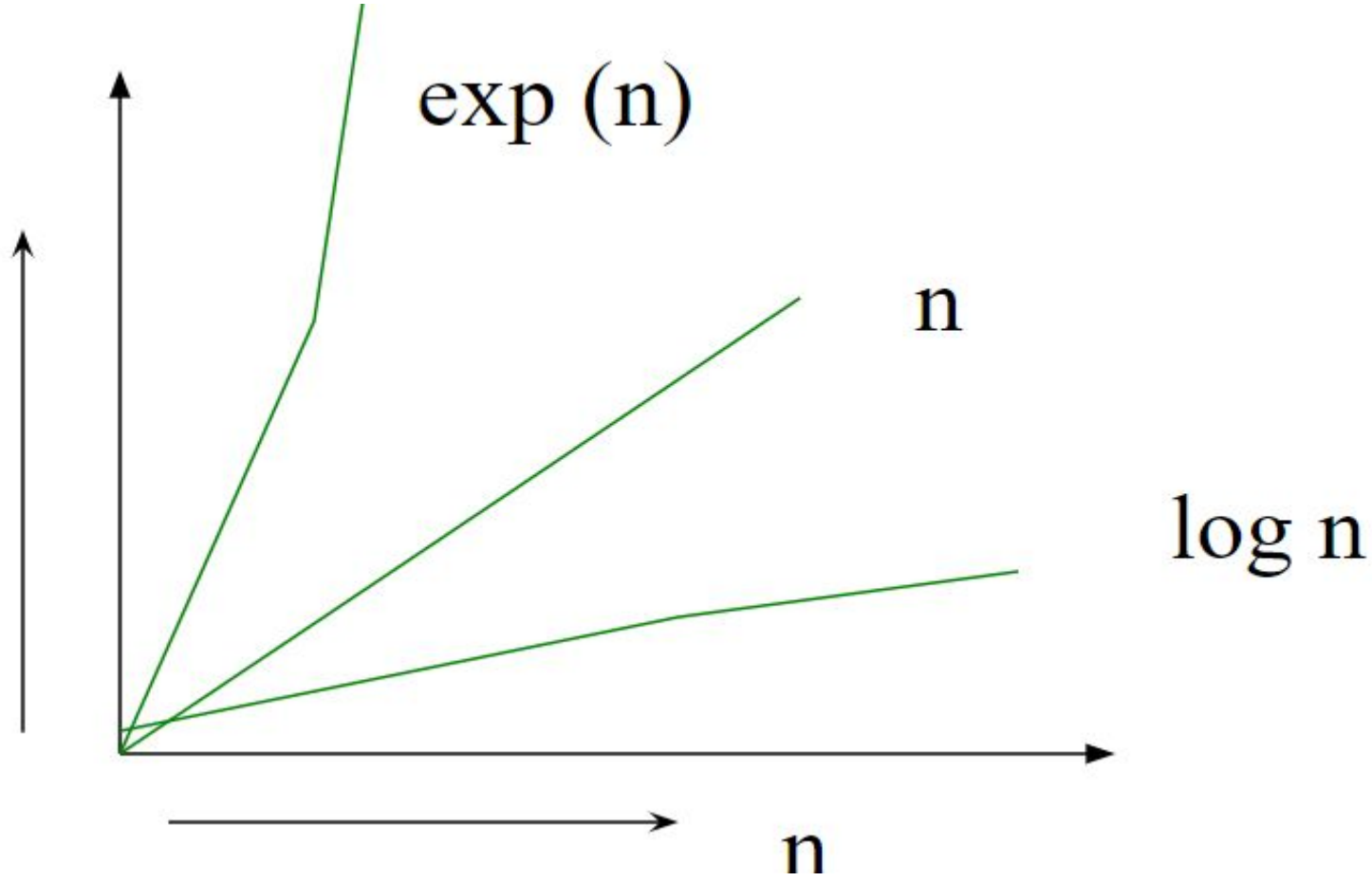
If an algorithm needs **n** basic operations and another needs **2n** basic operations, we will consider them to be in the same efficiency category.

However, we distinguish between exp(n), n, log(n)

Function Ordering

Order of Increase(order of growth)

We worry about the speed of our algorithms for large input sizes.

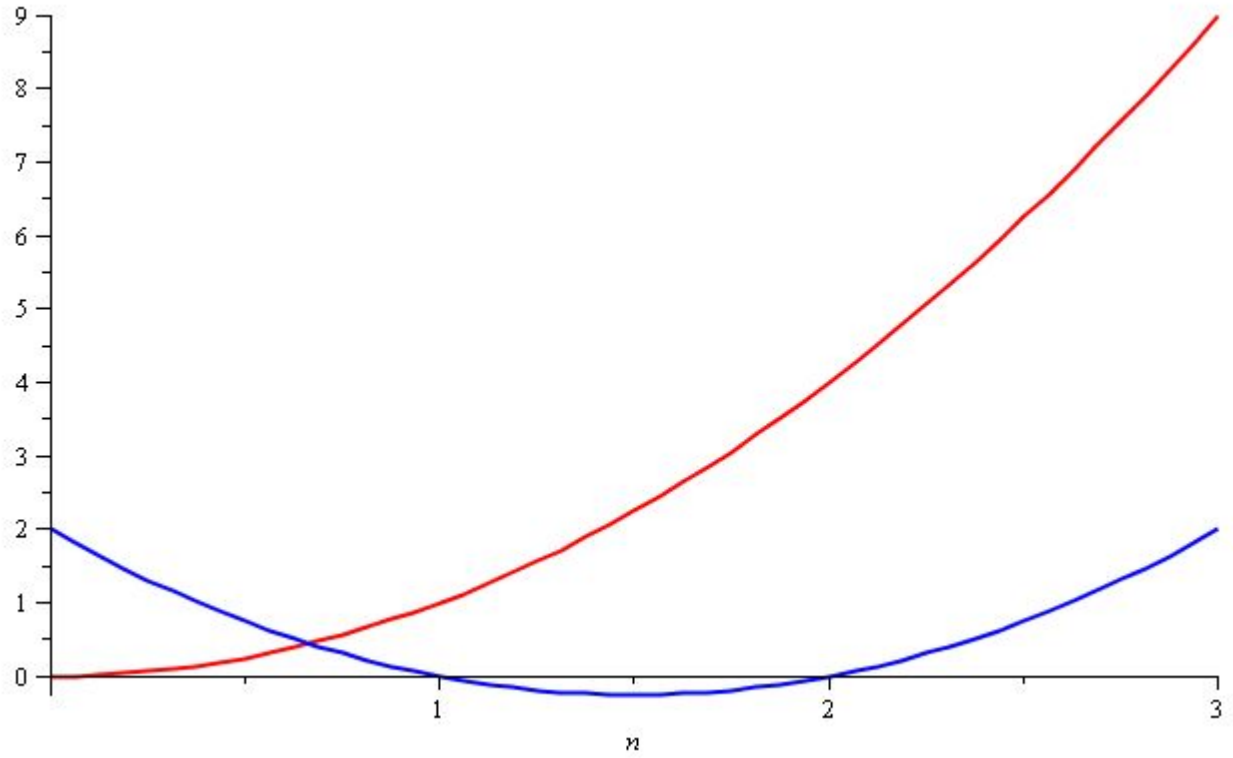


Quadratic Growth

Consider the two functions

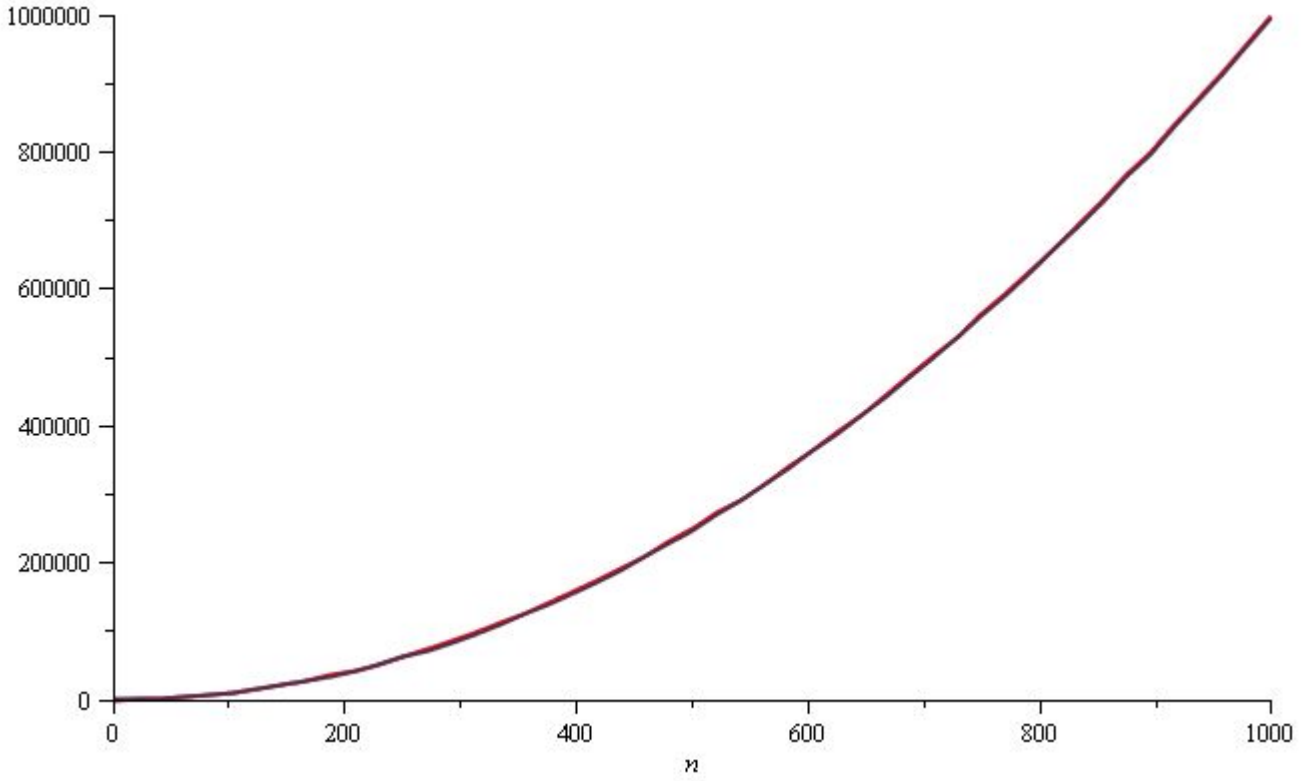
$f(n) = n^2$ and $g(n) = n^2 - 3n + 2$

Around $n = 0$, they look very different



Quadratic Growth

Yet on the range $n = [0, 1000]$, they are (relatively) indistinguishable:



Quadratic Growth

The absolute difference is large, for example,

f(1000) = 1 000 000

g(1000) = 997 002

but the relative difference is very small

$$\left|\frac{f(1000) - g(1000)}{f(1000)}\right| = 0.002998 < 0.3\%$$

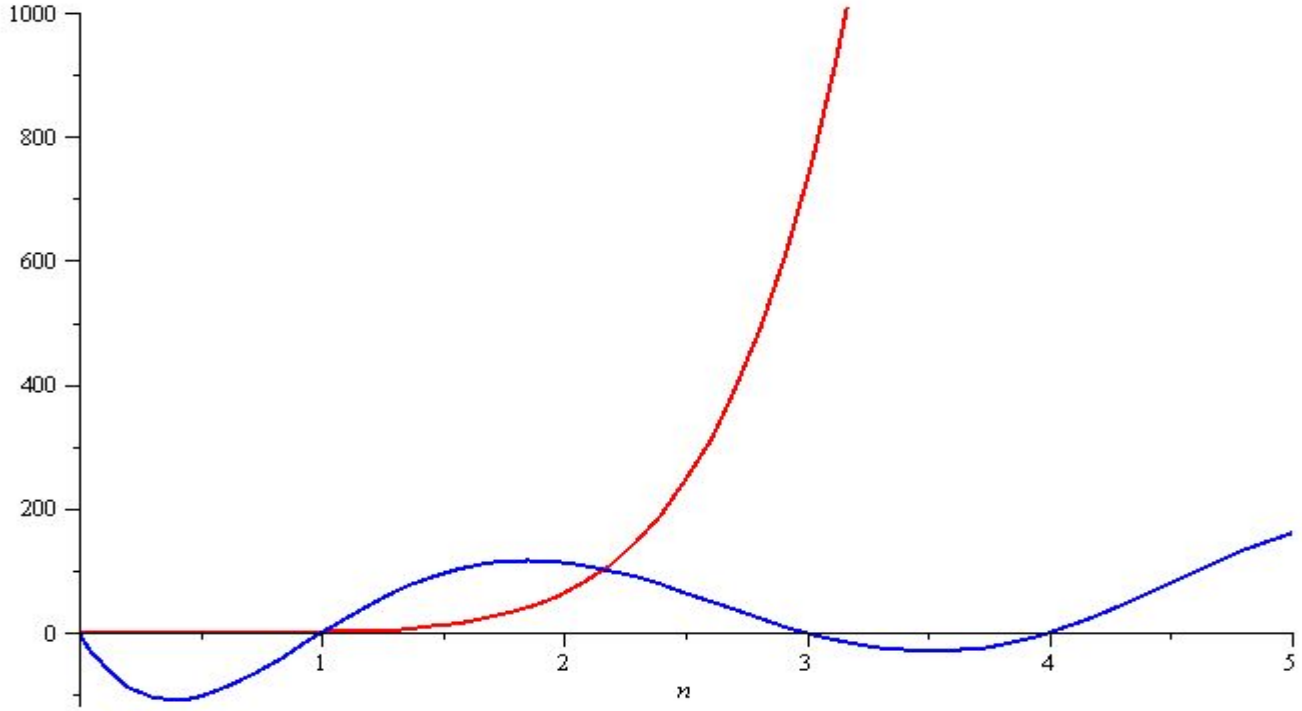
and this difference goes to zero as $n \to \infty$

Polynomial Growth

To demonstrate with another example,

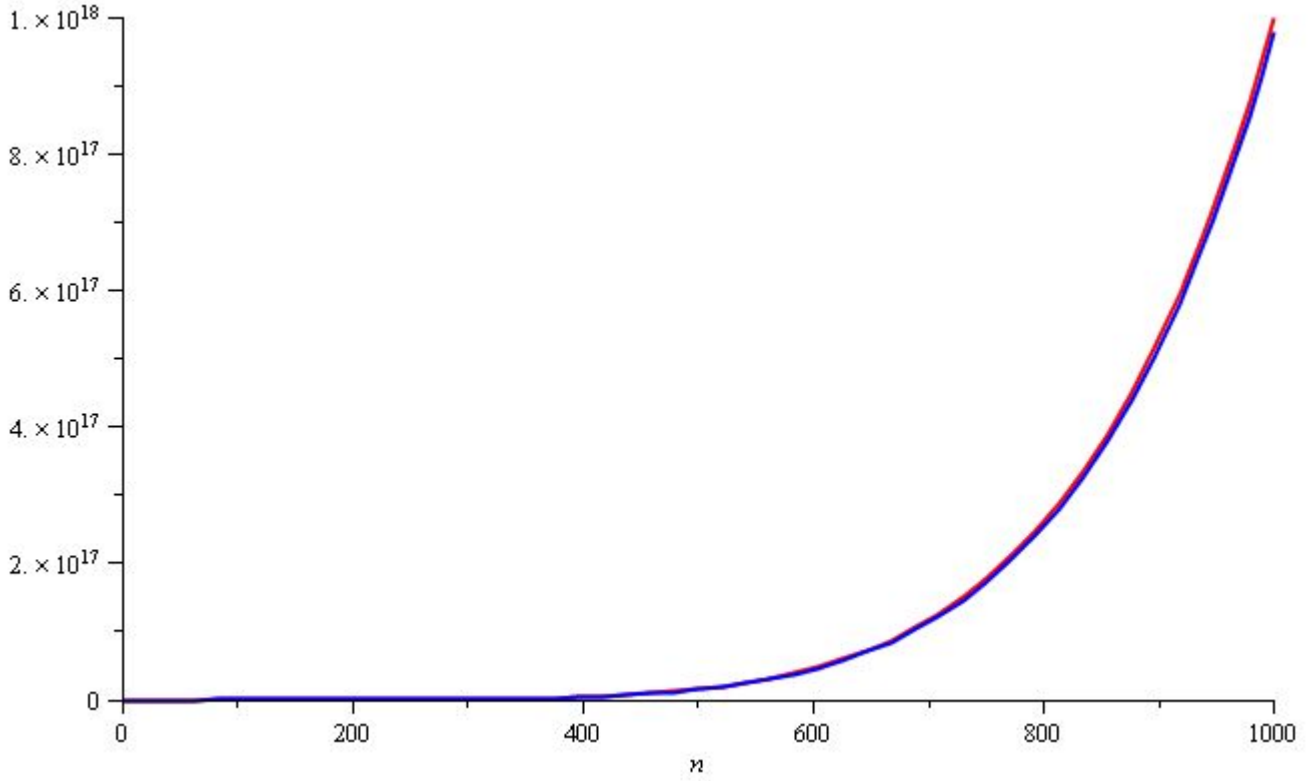
$f(n) = n^6$ and $g(n) = n^6 - 23n^5 + 193n^4 - 729n^3 + 1206n^2 - 648n$

Around $n = 0$, they are very different



Polynomial Growth

Still, around $n = 1000$, the relative difference is less than 3%



Polynomial Growth

The justification for both pairs of polynomials being similar is that, in both cases, they each had the same leading term:

- n^2 in the first case, n^6 in the second

Suppose however, that the coefficients of the leading terms were different

- In this case, both functions would exhibit the same rate of growth, however, one would always be proportionally larger

Weak ordering

Consider the following definitions:

- We will consider two functions to be equivalent, $f \sim g$, if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \qquad \text{where} \qquad 0 < c < \infty$$

- We will state that $\boldsymbol{f < g}$ if $$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

For functions we are interested in, these define a *weak ordering*

Weak ordering

f and g are functions from the set of natural numbers to itself.

Let $f(n)$ and $g(n)$ describe the run-time of two algorithms

- If $\boldsymbol{f(n) \sim g(n)}$, then it is always possible to improve the performance of one function over the other by purchasing a faster computer

- If $\boldsymbol{f(n) < g(n)}$, then you can **never** purchase a computer fast enough so that the second function always runs in less time than the first

Note that for small values of n, it may be reasonable to use an algorithm that is asymptotically more expensive, but we will consider these on a one-by-one basis

Function orders "Landau Symbols"

we will make some assumptions:

- Our functions will describe the time or memory required to solve a problem of size n

- We are restricting to certain functions :
  - They are defined for $n \geq 0$
  - They are strictly positive for all n
    - In fact, $f(n) > c$ for some value $c > 0$
    - That is, any problem requires at least one instruction and byte
  - They are increasing (monotonic increasing)

Function orders "Landau Symbols"

Big Oh Notation

A function **f(n)** is **O(g(n))** if the rate of growth of f(n) is not greater (not faster) than that of g(n).

Definition 1

f(n) = **O(g(n))** if there are a number $\mathbf{n_o}$ and a nonnegative **c** such that

$$\text{for all } n \geq n_o \text{ , } \quad 0 \leq f(n) \leq cg(n).$$

If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists and is finite, then f(n) is O(g(n)) .

Intuitively, (not exactly) f(n) is O(g(n)) means f(n) ≤ g(n) for all n beyond some value n_0; i.e. g(n) is an upper bound for f(n).

Function orders "Landau Symbols"

Example Functions

sqrt(n) , n, 2n, ln n, exp(n), n + sqrt(n) , $n + n^2$

$\lim_{n\to\infty}$ sqrt(n) /n = 0, sqrt(n) is O(n)

$\lim_{n\to\infty}$ n/sqrt(n) = infinity, n is not O(sqrt(n))

$\lim_{n\to\infty}$ n /2n = 1/2, n is O(2n)

$\lim_{n\to\infty}$ 2n /n = 2, 2n is O(n)

$\lim_{n\to\infty} \ln(n)/n = 0$, ln(n) is O(n)

$\lim_{n\to\infty} n/\ln(n) = \text{infinity}$, n is not O(ln(n))

$\lim_{n\to\infty} \exp(n)/n = \text{infinity}$, exp(n) is not O(n)

$\lim_{n\to\infty} n/\exp(n) = 0$, n is O(exp(n))

$\lim_{n\to\infty} (n+\text{sqrt}(n))/n = 1$, n + sqrt(n) is O(n)

$\lim_{n\to\infty} n/(\text{sqrt}(n)+n) = 1$, n is O(n+sqrt(n))

$\lim_{n\to\infty} (n + n^2)/n = \text{infinity}$, $n + n^2$ is not O(n)

$\lim_{n\to\infty} n/(n + n^2) = 0$, n is $O(n + n^2)$

Implication of big-Oh notation

Suppose we know that our algorithm uses at most O(f(n)) basic steps for any n inputs, and n is sufficiently large,

- then we know that our algorithm will terminate after executing at most f(n) basic steps.

- We know that a basic step takes a constant time in a machine.

Hence, our algorithm will terminate in a constant time f(n) units of time, for all large n.

Function orders "Landau Symbols"

Ω "Omega" Notation

Now a lower bound notation, **Ω**

Definition 2

f(n) = Ω(g(n)) if there are a number $\mathbf{n_o}$ and a nonnegative **c** such that

for all $n \geq n_o$, $f(n) \geq cg(n)$.

If $\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists

We say g(n) is a lower bound on f(n),
i.e. no matter what specific inputs we have, the algorithm will not run faster than this lower bound.

Suppose, an algorithm has complexity Ω(f(n)) . This means that there exists a positive constant c such that for all sufficiently large n, there exists at least one input for which the algorithm consumes at least c*f(n) steps.

Function orders "Landau Symbols"

θ "theta " Notation

Definition 3

f(n) = θ(g(n)) if and only if f(n) is O(g(n)) and Ω(g(n))

f(n) = θ(g(n)) if there exist positive n_o, c_1, and c_2 such that

$c_1\, g(n) \le f(n) \le c_2\, g(n)$ whenever $n \ge n_o$

- θ(g(n)) is "asymptotic equality"
- $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ is a finite, positive constant, if it exists

A function f(n) is θ(g(n)) if The function f(n) has a rate of growth **equal** to that of g(n)

Function orders "Landau Symbols"

Little-oh Notation

Definition 4

f(n) = **o(g(n))** if for all positive constant c, there exists an $\mathbf{n_o}$ such that :

$$f(n) < cg(n) \text{ when } n > n_o$$

Less formally , **f(n) = o(g(n))** if f(n) =O(g(n)) and f(n) ≠ θ(g(n)).

"asymptotic strict inequality"

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \quad \text{if} \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} \quad \text{exists}$$

Function orders "Landau Symbols"

Suppose that f(n) and g(n) satisfy $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$

If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ where $0 < c < \infty$, it follows that f(n) = **θ**(g(n))

If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ where $0 \le c < \infty$, it follows that f(n) = **O**(g(n))

If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, it follows that f($n$) = o(g($n$))

Function orders "Landau Symbols"

$$f(n) = \mathbf{o}(g(n)) \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

$$f(n) = \mathbf{O}(g(n)) \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

$$f(n) = \mathbf{\Theta}(g(n)) \qquad 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

$$f(n) = \mathbf{\Omega}(g(n)) \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$$

Function orders "Landau Symbols"

Terminology

Asymptotically less than or equal to	$\mathbf{O}$
Asymptotically greater than or equal to	$\mathbf{\Omega}$
Asymptotically equal to	$\mathbf{\theta}$
Asymptotically strictly less	$\mathbf{o}$

Little-o as a Weak Ordering

We can show that, for example

$$\ln(n) = \mathbf{o}(n^p) \qquad \text{for any } p > 0$$

Proof: Using l'Hôpital's rule.

If you are attempting to determine $\lim_{n\to\infty} \frac{f(n)}{g(n)}$

but both $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$, it follows

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f^{(1)}(n)}{g^{(1)}(n)}$$

Repeat as necessary...

Note: the k^{th} derivative will always be shown as $f^{(k)}(n)$

$$\lim_{n\to\infty} \frac{\ln(n)}{n^p} = \lim_{n\to\infty} \frac{1/n}{pn^{p-1}} = \lim_{n\to\infty} \frac{1}{pn^p} = \frac{1}{p} \lim_{n\to\infty} n^{-p} = 0$$

Big-θ as an Equivalence Relation

If we look at the first relationship, we notice that
$f(n) = \theta(g(n))$ seems to describe an equivalence relation:

1. $f(n) = \theta(g(n))$ if and only if $g(n) = \theta(f(n))$

2. $f(n) = \theta(f(n))$

3. If $f(n) = \theta(g(n))$ and $g(n) = \theta(h(n))$, it follows that $f(n) = \theta(h(n))$

Consequently, we can group all functions into equivalence classes, where all functions within one class are big-theta $\boldsymbol{\theta}$ of each other

Big-θ as an Equivalence Relation

For example, all of

n^2 $\quad 100000\,n^2 - 4\,n + 19$ $\quad n^2 + 1000000$

$323\,n^2 - 4\,n\ln(n) + 43\,n + 10$ $\quad 42n^2 + 32$

$n^2 + 61\,n\ln^2(n) + 7n + 14\ln^3(n) + \ln(n)$

are big-θ of each other

E.g., $42n^2 + 32 = \theta((\ 323\,n^2 - 4\,n\ln(n) + 43\,n + 10\ )$

We will select just one element to represent the entire class of these functions: $\boldsymbol{n^2}$

- We could choose any function, but this is the simplest

Function orders "Landau Symbols"

Terminology

The most common classes are given names:

$\Theta(1)$	constant
$\Theta(\ln(n))$	logarithmic
$\Theta(\log(n))$	
$\Theta(n)$	linear
$\Theta(n \ln(n))$	"$n \log n$"
$\Theta(n^2)$	quadratic
$\Theta(n^3)$	cubic
$2^n, e^n, 4^n, \ldots$	exponential

Recall that all logarithms are scalar multiples of each other

Therefore $\log_b(n) = \theta(\ln(n))$ for any base b

Function orders "Landau Symbols"

Example Functions

sqrt(n) , n, 2n, ln n, exp(n), n + sqrt(n) , $n + n^2$

$\lim_{n\to\infty}$ sqrt(n) /n = 0, sqrt(n) is o(n) and O(n)

$\lim_{n\to\infty}$ n/sqrt(n) = infinity, n is Ω(sqrt(n))

$\lim_{n\to\infty}$ n /2n = 1/2, n is θ(2n)

$\lim_{n\to\infty}$ 2n /n = 2, 2n is θ(n)