Graphs and Complexity for IoT

Master IoT

Mourad Hakem

Contact : mourad.hakem@univ-fcomte.fr
IUT Belfort - Université de Franche-Comté

December 1, 2025

Lecture 2

Let A and B be two algorithms solving the same problem of complexity $100n$ and n^2 respectively. Which is more efficient?

- The ratio of complexities $= n/100$.
- For $n < 100$, B is more efficient, for $n = 100$, A and B have the same efficiency and for $n > 100$, A is more efficient.
- Note that as n becomes larger, A is more efficient than B.
- If the data sizes are "small", most of the algorithms solving the same problem are the same
- It is the behaviour of the complexity of an algorithm when the size of the data becomes large which is important
- We call this behaviour: asymptotic complexity

Definitions: asymptotic notations

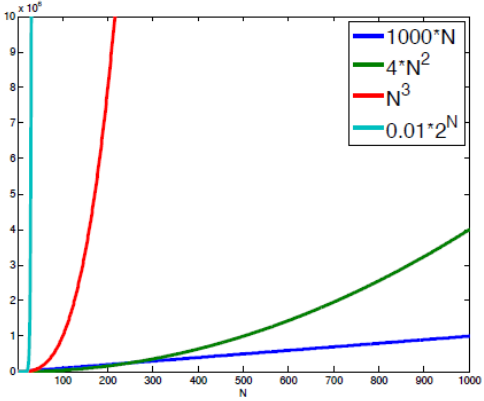
Recall:

The theoretical analysis of the efficiency of an algorithm is carried out with a multiplicative constant to disregard:

- programming language
- compiler and operating system
- computer power

In general, we are not interested by the exact complexity but by its order of magnitude.

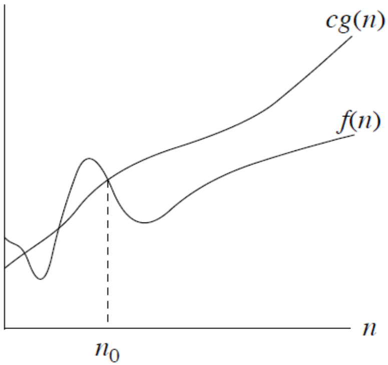
Execution time growth rate: changing the environment affects $T(n)$ by a constant factor, but does not affect the growth rate of $T(n)$.



Asymptotic notations: $\mathcal{O}, \Omega, \Theta$

- **Upper Bound** $\mathcal{O}$: $f(n)$ is in $\mathcal{O}(g(n))$ if :
  $\exists n_0 \geq 1, \exists c > 0, \forall n \geq n_0 : f(n) \leq cg(n)$

  meaning: $\exists$ a threshold from which the function $f(.)$ is always dominated by the function $g(.)$ with a multiplicative constant c.



Asymptotic notations: $\mathcal{O}, \Omega, \Theta$

Example 1 : $f(n) = 5n + 37$ is in $\mathcal{O}(n)$?

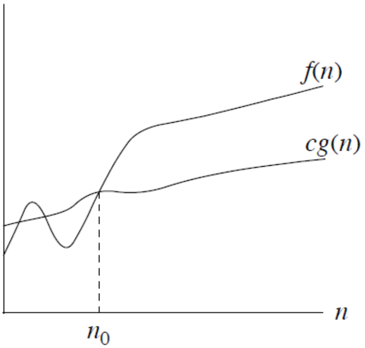
find c and n_0 from which $f(n) \leq cn$?

- $c = 6, n_0 = 37$
- $c = 10, n_0 = 8$

Example 2 : $f(n) = 6n^2 + 2n - 8$ is in $O(n^2)$ $(c = 7, n_0 = 1)$

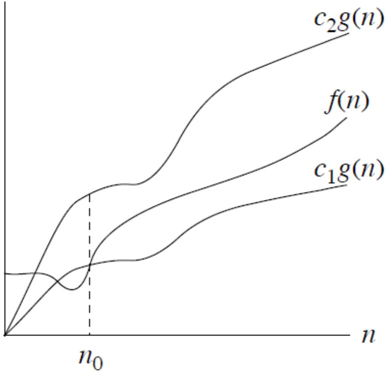
Asymptotic notations: $\mathcal{O}, \Omega, \Theta$

- **Lower Bound** Ω : $f(n)$ is in $\Omega(g(n))$ if :
  $\exists n_0 \geq 1, \exists c > 0, \forall n \geq n_0 : f(n) \geq cg(n)$



Asymptotic notations: $\mathcal{O}, \Omega, \Theta$

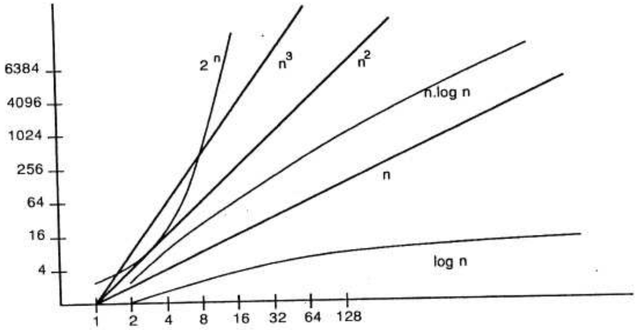
- **Tight Bound Θ** : $f(n)$ est en $\Theta(g(n))$ si :
  $\exists n_0 \geq 1, \exists c_1, c_2 > 0, \forall n \geq n_0 : c_1 g(n) \leq f(n) \leq c2g(n)$



Complexity classes:

- $\mathcal{O}(\log n)$: logarithmic logarithmic.
  example : dichotomic/binary search in an array of size n.
- $\mathcal{O}(n)$: linear
  example: simple search in an unordered array of size n.
- $\mathcal{O}(n \log n)$: quasi-linear (sub-quadratic).
  example : merge sort of an array of size n.
- $\mathcal{O}(n^k)$: polynomial, with $k > 1$.
  example: matrix multiplication
  - quadratic if $k = 2$
  - cubic if k =3
- $\mathcal{O}(a^n)$: exponential, with $a > 1$.
  example: Hanoi Tower
- $\mathcal{O}(n!)$: factorial
  example : TSP

Classes de complexité :



Simplifications rules

After calculating the value of the number of operations, we perform the following simplifications:

1. ignore the multiplicative and additive constants;
2. retain only the dominant terms;

We therefore prefer to have an idea of the execution time of the the algorithm rather than a more precise but unnecessarily complicated expression!

Examples :

1- Let be an algorithm performing $T(n) = 4n^3 - 5n^2 + 2n + 3 \in \mathcal{O}(n^3)$.

2- Example of the factorial: $T(n) = 5n - 1 = \mathcal{O}(n)$

2- Example of data cleaning with shifting version:
$T(n) = \frac{n(n-1)}{2} = \mathcal{O}(n^2)$.