**41LIDD01 - CM - Complexité - FA - (Serge MIGUET 2025-26)**

Serge MIGUET – 04.78.77.43.77 – LIRIS – C123

Moodle : <https://moodle.univ-lyon2.fr/course/view.php?id=33054>

3 évaluations :

* 08/10/25 (30 minutes, 30 questions)
  + CM1, CM2 et première partie du CM3
* 05/11/25 (15 minutes, 15 questions)
* 26/11/25 (15 minutes, 15 questions)

Complexité : P, NP, Exponentiels

Complexité en temps & complexité en mémoire se basent sur des modèles mathématiques.

Modèle RAM : temps d’accès à une adresse en mémoire = temps d’accès à toute autre adresse

On s’intéresse à l’ordre de grandeur et non pas le temps exact d’exécution.

Ordre de grandeur multiplié par un contant = même ordre de grandeur

Grand-O : mon temps d’exécution n’est pas supérieur à cette borne (, etc.)

* Bornes supérieures

Grand-Omega : mon temps d’exécution = au moins cette borne

* Bornes inférieures
* Borne exacte

Même si ,

RSA : there are algorithms which can generate large prime numbers. If we generate two (*a* and *b*) and multiply them together, we get a number *c* which we know is not prime. However, given just *c*, it is extremely difficult to determine *a* and *b*. The best algorithms currently in existence have exponential run-time. RSA is based on this principle.

Nombre de chiffres!

Quatre mots pour décrire les algorithmes:

1. Affectation / Séquence
2. Conditionnelle
3. Itération
4. Abstraction Procédurale (Appel à fonction)

On n’aura pas besoin des structures de données plus complexes que des tableaux

Opérations élémentaires : opérations qui se font en temps constant

Il faudra bien faire attention à cela !

On s’intéresse au pire des cas

QuickSort : dans de rares cas\*, il se comporte très mal – (temps d’exécution en haut de n^2)

\*tableau presque trié par exemple

Somme des *n* premiers entiers :

Somme des *n* premiers carrés :

Somme des *n* premières puissances de *x* :