# Leçon 926 - Analyse des algorithmes : complexité. Exemples.

17 juillet 2019

## 1 Extraits du Rapport

**Rapport de jury 2018**

*Il s'agit ici d'une leçon d'exemples. Le candidat prendra soin de proposer l'analyse d'algorithmes portant sur des domaines variés, avec des méthodes d'analyse également variées : approche combinatoire ou probabiliste, analyse en moyenne ou dans le pire cas. Si la complexité en temps est centrale dans la leçon, la complexité en espace ne doit pas être négligée. La notion de complexité amortie a également toute sa place dans cette leçon, sur un exemple bien choisi, comme union find (ce n'est qu'un exemple).*

## 2 Cœur de la leçon

— Complexité en temps (meilleur cas, pire cas, moyenne) et en espace.

— Méthodes d'analyse

— Des exemples.

## 3 À savoir

— Analyse des algorithmes : relations de comparaison $O$, $\theta$ et $\Omega$.

— Exemple d'analyse en moyenne : recherche d'un élément dans un tableau.

— Des algorithmes et leurs complexité dans différent domaine (minimisation d'automate, plus court chemins, unification, recherche... )

— Complexité amortie par méthode de l'agrégat.

## 4 Ouvertures possibles

— Complexité amortie par méthode comptable ou potentiel.

— FFT

— Compromis temps-mémoires, rainbow table.

## 5 Conseils au candidat

— Il ne faut pas se limiter à la complexité en temps, mais aussi parler de complexité en mémoire.

— Connaître la complexité des diverses opérations de base de chacun des algorithmes présentés.

— Il faut trouver un équilibre entre la présentation d'exemples, et la présentation formelle des notions théoriques.

— Si certains points sont présentés de manière informelle, ne pas les appeler proposition ou théorème.

— Faire le lien avec certaines méthodes et certains paradigmes de programmation (e.g, le théorème maître est à relier au paradigme diviser pour régner).

— Bien avoir en tête les mesures pour les tailles des entrées.

# 6 Questions classiques

— Lors d'une analyse de complexité en moyenne, que suppose-t-on sur la distribution des entrées ? Est-ce pertinent ?

— Pourquoi se contente-t-on de $O$, plutôt que de calculs exacts ?

— Pouvez vous dérouler tel algorithme sur tel exemple ?

— Questions de détails d'implémentation/de complexité des opérations élémentaire sur un algorithme présenté.

— Dans tel cas pratique, quelle algorithme vaut-il mieux utiliser ?

— Savez-vous si tel algorithme est optimal ?

# 7 Références


— [*Cor*] Algorithmique - Cormen - à la BU/LSV
*La bible de l'algorithmique, avec toutes les bases. Attention, les calculs avec des probas sont parfois faux.*

— [*Bea*] Éléments d'algorithmique - D. Beauquier, J. Berstel, Ph. Chrétienne - à la BU/LSV
*Bonne référence pour l'algo, pleins de dessins et de preuves. Un peu vieillissant et devenu rare.*


# 8 Dev

\+ Correction totale et/ou complexité de Dijkstra - ([Cor], [Bea], *p. ?*) - 925,926,927
*Long de faire correction et complexité, doit-être adapté selon la leçon.*

++ Complexité du tri rapide - ([Cor], [Bea]) - 903,926,931
*Bien faire attention au proba de [Cor], aller voir [Bea] est pertinent. Avoir une idée de l'écart type des performances.*

++ Analyse amortie dans les arbres 2-4 - ([Bea]) - 901,921,926,(932 ?)
*Développement un peu plus original que les arbres AVL, les B-arbres étant utilisés en pratique dans postgresql pour faire des indexes de bases de données. Dessins et exemples bienvenus.*

++ Hachage Parfait - ([Cor], *p. 258*) - 901,921