# Leçon 901 - Structures de données. Exemples et applications.

7 juin 2019

## 1 Extraits du Rapport

**Rapport de jury 2018**

*Le mot algorithme ne figure pas dans l'intitulé de cette leçon, même si l'utilisation des structures de données est évidemment fortement liée à des questions algorithmiques. La leçon doit donc être orientée plutôt sur la question du choix d'une structure de données. Le jury attend du candidat qu'il présente différents types abstraits de structures de données en donnant quelques exemples de leur usage avant de s'intéresser au choix de la structure concrète. Le candidat ne peut se limiter à des structures linéaires simples comme des tableaux ou des listes, mais doit présenter également quelques structures plus complexes, reposant par exemple sur des implantations à l'aide d'arbres. Les notions de complexité des opérations usuelles sur la structure de données sont bien sûr essentielles dans cette leçon.*

## 2 Cœur de la leçon

— Types abstraits et implantations concrètes.

— Structures linéaires et non linéaires. Exemples de structures complexes.

— Complexité des opérations usuelles selon implantation.

## 3 À savoir

— Tableaux, listes, piles, files, file de priorité, arbres, graphes (orientés et non orientés), ensembles, dictionnaires.

— Influence de la structure de données sur un algorithme choisi.

— Pour un même type de donnée (e.g un graphe), exemples de choix de représentations optimisés selon l'ensemble de données possibles (e.g graphe sparse ou dense)

— Notion de coût amorti.

— Connaître au moins une structure complexe (tas, union-find, arbres approximativement équilibré,...)

## 4 Ouvertures possibles

— Tas de Fibonacci.

— Représentation des graphes.

— Notion de types persistant et mutable.

# 5 Conseils au candidat

— Les algorithmes ne doivent pas être le cœur de la leçon, mais bien les structures de données.

— Les dessins sont souvent très appropriés pour représenter des structures de données et leurs évolutions.

— Attention à ne pas s'éparpiller avec trop de structures différentes et à faire un catalogue. Il n'est pas forcément nécéssaire d'écrire tous les interfaces des types abstraits.

— Bien avoir en tête les abstractions. Par exemple, un dictionnaire ne porte pas nécéssairement sur des strings.

— La complexité, c'est bien, la comparaison des complexités, c'est mieux.

— Ne pas être trop lié à un langage de programmation.

— Les structures avancés introduites doivent être accompagné d'une application.

# 6 Questions classiques

— Pourquoi avoir une approche par type abstraits ?

— Quels structure de donnés sont en pratique les plus utilisés, et intégrés nativement dans les languages de programmations ?

— Connaissez vous l'implémentation des [insérerer ici une structure] en pratique utilisé ?

— Avec les listes chainé/doublement chainé, quelles opérations sont plus ou moins complexe ?

— Le choix du type, par exemple persistent et mutable, est souvent lié à des soucis d'efficacité. Y a t'il d'autre préoccupations lors de la programmation influencées par ce choix ?

— Questions techniques/point de détail sur l'une des structures présenté.

# 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

— Complexité et correction du tri par tas - ([Cor], *3rd edition, p.154*) - 901,903
*Simple dans le fond, mais à bien faire formellement, et pédagogiquement. Dessins et exemples bienvenue.*

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

— Arbre Binaire de Recherche optimaux - ([Cor], *p. 397*) - 901,921