Leçon 901 - Structures de données. Exemples et applications.

9 février 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 Coeur de la leçons

— Types abstraits, cas linéaire et non linéaire

— Interface abstraite et implan- tation concrète.

— Complexité des opérations usuelles

3 À savoir

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

— Notion de cout amorti.

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

— Influence de la structure de données sur un algorithme.

— Notion de types persistant et mutable.

4 Ouvertures possibles

— Représentation des entiers et float en machine.

— Structure avancé : Arbre rouge-noir , tas de Fibonnacci, Union-Find

— Structures paresseuses

5 Conseils au candidat

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

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

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

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

— 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 ?

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.

— [*Cor*] 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.

— **TMP** froideveaux **TMP**

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.

— **TMP** dijkstra **TMP**

— **TMP** hachage parfait **TMP**

— **TMP** ABR optimaux **TMP**