Agrégation

Xavier Montillet

12 avril 2016

Table des matières

Notations :

➤ todo (pour les oraux blancs)

✔ ok

◕ bof (à changer si possible)

✘ pas ok

Première partie

Informatique

Chapitre 1

Leçons

1.901 ➤Structures de données : exemples et applications.

1.901.1 Rapport du jury

> 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 que d'un algorithme. 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.
>
> (*Rapport du jury.* 2015)

1.902 ➤Diviser pour régner : exemples et applications.

1.902.1 Rapport du jury

> Cette leçon permet au candidat de proposer différents algorithmes utilisant le paradigme *diviser pour régner*. Le jury attend du candidat que ces exemples soient variés et touchent des domaines différents.
>
> Un calcul de complexité ne peut se limiter au cas où la taille du problème est une puissance exacte de 2, ni à une application directe d'un théorème très général recopié approximativement d'un ouvrage de la bibliothèque de l'agrégation. (*Rapport du jury.* 2015)

1.903 ➤Exemples d'algorithmes de tri. Complexité.

1.903.1 Rapport du jury

> Sur un thème aussi classique, le jury attend des candidats la plus grande précision et la plus grande rigueur.
>
> Ainsi, sur l'exemple du tri rapide, il est attendu du candidat qu'il sache décrire avec soin l'algorithme de partition et en prouver la correction et que l'évaluation des complexités dans le cas le pire et en moyenne soit menée avec rigueur.
>
> On attend également du candidat qu'il évoque la question du tri en place, des tris stables, ainsi que la représentation en machine des collections triées.
>
> Le jury ne manquera pas de demander au candidat des applications non triviales du tri. (*Rapport du jury.* 2015)

1.903.2 Remarques

Quand on trie, on trie par rapport à un *préordre* $\preccurlyeq$. On a donc aussi une relation d'équivalence $x \approx y \coloneqq (x \preccurlyeq y) \wedge (x \succcurlyeq y)$ qui n'a aucune raison d'être l'égalité.

Si l'on note $\sqsubseteq$ le préordre associé au tableau donné en entrée d'un tri (donc $T[i] \sqsubseteq T[j] \iff i \leq j$), le fait qu'une procédure de tri soit *stable* veut dire que trier par rapport à $\preccurlyeq$ ou par rapport au préordre lexicographique associé à $(\preccurlyeq, \sqsubseteq)$, que l'on note $\preccurlyeq^{lex}_{\preccurlyeq,\sqsubseteq}$, revient au même. En particulier, si on a une procédure de tri instable, on peut en construire une stable (mais le tri ne peut alors plus être *en place* car pour pouvoir savoir si $x \sqsubseteq y$, on a besoin de stocker de tableau d'entrée et on est donc forcé de travailler sur une copie).

La notion de tri *stable* n'a aucun intérêt si on se restreint aux entiers muni de leur ordre habituel (ou plus généralement à un ensemble muni d'un ordre [par opposition à un préordre]) car si l'on ne peut jamais distinguer $T[i]$ de $T[j]$, s'assurer qu'ils sont restés dans le même ordre n'est pas vraiment utile.

Pour trier selon $\preccurlyeq^{lex}_{\preccurlyeq_1,\preccurlyeq_2}$, on peut d'abord trier selon $\preccurlyeq_2$ et ensuite utiliser une procédure de tri stable pour trier selon $\preccurlyeq_1$ (ce qui peut être plus facile à implémenter que le tri selon $\preccurlyeq^{lex}_{\preccurlyeq_1,\preccurlyeq_2}$ et peut servir si l'on doit trier selon $\preccurlyeq^{lex}_{\preccurlyeq_1,\preccurlyeq_2}$ et $\preccurlyeq^{lex}_{\preccurlyeq'_1,\preccurlyeq_2}$).

1.906 ➤Programmation dynamique : exemples et applications.

1.906.1 Rapport du jury

> Même s'il s'agit d'une leçon d'exemples et d'applications, le jury attend des candidats qu'ils présentent les idées générales de la programmation dynamique et en particulier qu'ils aient compris le caractère générique de la technique de mémoïsation. Le jury appréciera que les exemples choisis par le candidat couvrent des domaines variés, et ne se limitent pas au calcul de la longueur de la plus grande sous-séquence commune à deux chaînes de caractères.

Le jury ne manquera pas d'interroger plus particulièrement le candidat sur la question de la correction des algorithmes proposés et sur la question de leur complexité en espace. (*Rapport du jury.* 2015)

1.907 ➤Algorithmique du texte : exemples et applications.

1.907.1 Rapport du jury

Cette leçon devrait permettre au candidat de présenter une grande variété d'algorithmes et de paradigmes de programmation, et ne devrait pas se limiter au seul problème de la recherche d'un motif dans un texte, surtout si le candidat ne sait présenter que la méthode naïve.

De même, des structures de données plus riches que les tableaux de caractères peuvent montrer leur utilité dans certains algorithmes, qu'il s'agisse d'automates ou d'arbres par exemple.

Cependant, cette leçon ne doit pas être confondue avec la *909 : Langages rationnels. Exemples et applications* ni avec la *910 : Langages algébriques. Exemples et applications.*

La compression de texte peut faire partie de cette leçon si les algorithmes présentés contiennent effectivement des opérations comme les comparaisons de chaînes : la compression LZW, par exemple, ressortit davantage à cette leçon que la compression de Huffman.

(*Rapport du jury.* 2015)

1.909 ➤Langages rationnels. Exemples et applications.

1.909.1 Rapport du jury

Des applications dans le domaine de la compilation entrent naturellement dans le cadre de ces leçons. (*Rapport du jury.* 2015)

1.910 ➤Langages algébriques. Exemples et applications.

TODO

1.912 ➤Fonctions récursives primitives et non primitives. Exemples.

TODO

1.913 ➤Machines de Turing. Applications.

TODO

1.914 Décidabilité et indécidabilité. Exemples.

TODO

1.915 Classes de complexité : exemples.

1.915.1 Rapport du jury

> Le jury attend que le candidat aborde à la fois la complexité en temps et en espace. Il faut naturellement exhiber des exemples de problèmes appartenant aux classes de complexité introduites, et montrer les relations d'inclusion existantes entre ces classes.
>
> Le jury s'attend à ce que le caractère strict ou non de ces inclusions soit abordé, en particulier le candidat doit être capable de montrer la non-appartenance de certains problèmes à certaines classes.
>
> Parler de décidabilité dans cette leçon serait hors sujet.
>
> (*Rapport du jury.* 2015)

1.916 ➤Formules du calcul propositionnel : représentation, formes normales, satisfiabilité. Applications.

1.916.1 Rapport du jury

> Le jury attend des candidats qu'ils abordent les questions de la complexité de la satisfiabilité.
>
> Pour autant, les applications ne sauraient se réduire à la réduction de problèmes NP-complets à SAT.
>
> Une partie significative du plan doit être consacrée à la représentation des formules et à leurs formes normales. (*Rapport du jury.* 2015)

1.917 ➤Logique du premier ordre : syntaxe et sémantique.

1.917.1 Rapport du jury

> La question de la syntaxe dépasse celle de la définition des termes et des formules. Elle comprend aussi celle des règles de la démonstration.
>
> Le jury attend donc du candidat qu'il présente au moins un système de preuve et les liens entre syntaxe et sémantique, en développant en particulier les questions de correction et complétude.
>
> (*Rapport du jury.* 2015)

1.918 Systèmes formels de preuve en logique du premier ordre : exemples.

TODO

1.919 Unification : algorithmes et applications.

TODO

1.920 Réécriture et formes normales. Exemples.

1.920.1 Rapport du jury

> Au-delà des propriétés standards (terminaison, confluence) des systèmes de réécriture, le jury attend notamment du candidat qu'il présente des exemples sur lesquels l'étude des formes normales est pertinente dans des domaines variés : calcul formel, logique, etc.
>
> Un candidat ne doit pas s'étonner que le jury lui demande de calculer des paires critiques sur un exemple concret.
>
> Lorsqu'un résultat classique comme le lemme de Newman est évoqué, le jury attend du candidat qu'il sache le démontrer.
>
> (*Rapport du jury.* 2015)

1.921 ➤Algorithmes de recherche et structures de données associées.

TODO

1.922 ➤Ensembles récursifs, récursivement énumérables. Exemples.

TODO

1.923 ➤Analyses lexicale et syntaxique : applications.

TODO

1.924 Théories et modèles en logique du premier ordre. Exemples.

TODO

1.925 ➤Graphes : représentations et algorithmes.

TODO

1.926 Analyse des algorithmes : complexité. Exemples.

TODO

1.927 ➤Exemples de preuve d'algorithme : correction, terminaison.

1.927.1 Rapport du jury

> Le jury attend du candidat qu'il traite des exemples d'algorithmes récursifs et des exemples d'algorithmes itératifs.
>
> En particulier, le candidat doit présenter des exemples mettant en évidence l'intérêt de la notion d'invariant pour la correction partielle et celle de variant pour la terminaison des segments itératifs.
>
> Une formalisation comme la logique de Hoare pourra utilement être introduite dans cette leçon, à condition toutefois que le candidat en maîtrise le langage. (*Rapport du jury.* 2015)

1.928 Problèmes NP-complets : exemples de réductions

1.928.1 Rapport du jury

> L'objectif ne doit pas être de dresser un catalogue le plus exhaustif possible ; en revanche, pour chaque exemple, il est attendu que le candidat puisse au moins expliquer clairement le problème considéré, et indiquer de quel autre problème une réduction permet de prouver sa NP-complétude.
>
> Les exemples de réduction seront autant que possible choisis dans des domaines variés : graphes, arithmétique, logique, etc. Un exemple de problème NP-complet dans sa généralité qui devient P si on contraint davantage les hypothèses pourra être présenté, ou encore un algorithme P approximant un problème NP-complet.
>
> Si les dessins sont les bienvenus lors du développement, le jury attend une définition claire et concise de la fonction associant, à toute instance du premier problème, une instance du second ainsi que la preuve rigoureuse que cette fonction permet la réduction choisie.
>
> (*Rapport du jury.* 2015)

Chapitre 2

Développements

2.1 Cook

Leçons :
- ➤Fonctions récursives primitives et non primitives. Exemples
- ➤Machines de Turing. Applications
- ➤Formules du calcul propositionnel : représentation, formes normales, satisfiabilité. Applications

2.2 Problèmes indécidables sur les langages algébriques

Leçons :
- ➤Langages algébriques. Exemples et applications
- ➤Machines de Turing. Applications
- ➤Ensembles récursifs, récursivement énumérables. Exemples
- ➤Analyses lexicale et syntaxique : applications

2.3 Tri par Tas

Leçons :
- ➤Structures de données : exemples et applications
- ➤Exemples d'algorithmes de tri. Complexité

2.4 Unification

Leçons :
- ➤Logique du premier ordre : syntaxe et sémantique
- ➤Exemples de preuve d'algorithme : correction, terminaison

Chapitre 3

Références

Livres a priori cools : Kozen [1]

Bibliographie

[1] Dexter C. Kozen. *The Design and Analysis of Algorithms*. New York, NY, USA : Springer-Verlag New York, Inc., 1992. ISBN : 0-387-97687-6.

[2] *Rapport du jury*. 2015. URL : http://cache.media.education.gouv.fr/file/agreg_ext/30/1/math_470301.pdf.

Deuxième partie

Mathématiques