Leçon 915 – Classes de complexité. Exemples

17 juillet 2019

1 Extraits du Rapport

Rapport de jury 2018

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, en abordant le caractère strict ou non de ces inclusions. Le jury s'attend à ce que les notions de réduction polynomiale, de problème complet pour une classe, de robustesse d'une classe vis à vis des modèles de calcul soient abordées. Parler de décidabilité dans cette leçon serait hors sujet.

2 Cœur de la leçon

— Complexité en espace et en temps.
— Les classes usuelles (P, NP, PSPACE) et inclusions.
— Notion de réduction, de problème complet
— SAT et le théorème de Cook

3 À savoir

— Exemples de problèmes complets sur des domaines différents (graphes, automates, grammaires, logique, circuits)
— Exemples de réductions.
— La robustesse vis-à-vis du modèle de calcul (une bande, plusieurs bandes, écriture sur entrée)
— Savitch et les classes en espace
— Modèle des machines RAM
— Les théorèmes de hiérarchie stricte (attention, la preuve concernant l'espace est très difficile, dangereux en développements)

4 Ouvertures possibles

— Des théorèmes de borne inférieure de simulation (une bande vs deux bandes en $O(n)$ vs $O(n^2)$ pour le langage des palindromes).
— Les classes NL, coNL et Immerman-Szelepcsényi
— Développer le côté logique (2SAT, HORNSAT, SAT, QBF, CTLSAT).
— Les classes alternantes, le lien avec PSPACE et EXP et la hiérarchie polynomiale [Car]
— Une machine qui calcule trop rapidement est un automate [Car]
— (vraiment loin du programme) Classes randomisés (tests de primalité, algorithme de Berlekamp, polynomial identity testing, RP, coRP)

5 Conseils au candidat

— Attention à la représentation des données.

— Mettre des exemples, et des exemples.

— Ne pas centrer toute la leçon sur P et NP, mais ne pas les oublier.

— Un petit diagramme qui permet de s'y retrouver en annexe avec les inclusions strictes et les égalités est le bienvenu.

— Si on parle de NL, bien formaliser le type de machines considérées, et bien expliquer la notion de réduction polynomiale, et logarithmique si on parle de NL

— Les différentes ouvertures hors programme peuvent être très dangereuses, le domaine de la complexité est plein de pièges et de subtilités.

6 Questions classiques

— En pratique, que signifie NP-complet ? Quelle est l'intérêt de la théorie de la complexité ?

— Quels sont les impacts de la variation du modèle sur la complexité ?

— Pourquoi utiliser le formalisme des machines de Turing plutôt qu'un autre (RAM, λ-calcul, fonctions récursives) ?

— Citer une classe qui n'est pas P, NP ou $PSPACE$.

— Montrer que tel problème X est C-complet.

— Comment varie la complexité en fonction de la taille de l'alphabet ?

— Discussions sur les variantes de SAT.

7 Références

— [*Per*] Complexité algorithmique - Perifel - BU
La référence parfaite pour la complexité. Formel, clair. C'est essentiellement une traduction formelle du Arora.

— [*Car*] Langages formels, calculabilité et complexité - Carton - à la BU/LSV
Très bonne référence couvrant beaucoup de bases. Se méfier de certaines preuves faites un peu rapidement. La NP-complétude de HamPATH est fausse dedans. Le Kleinberg *la fait bien par contre.*

— [*Aro*] Computational Complexity : A Modern Approach - Arora Barak - au LSV
En anglais. Les premiers chapitres couvrent les notions nécéssaires et donnent souvent une bonne intuition. Manque parfois de formalisme, à croiser avec le Perifel.

— Éviter le Papadimitriou, qui contient de très nombreuses fausses preuves.

8 Dev

++ Théorème de Cook-Levin - ([Car], *p. 191*) - 913,915,916,928
Preuve que SAT est NP-complet. Aller jusqu'à 3-SAT est ambitieux. Bien comprendre la notion de localité du calcul d'une machine de Turing.

++ Preuve de complétude d'un problème au choix.

\+ Immerman Szelepcsényi - ([Per], *p. 121*) - 915
Implique de maitriser parfaitement NL.