Références et Développements

17 juillet 2019

1 Références

— [*Han*] Handbook of Theoretical Computer Science - Leeuwen - ?
Contient un bon chapitre par Crochemore *pour l'algorithmique du texte*

— [*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.

— [*Kle*] Algorithm design - Jon Kleinberg, Éva Tardos - à la BU
Bonne réference pour les paradigmes de programmation, très bonne référence pour les algorithmes d'approximation, et les variantes de problèmes NP complets qui deviennent P.

— [*Wol*] Introduction à la calculabilité : cours et exercices corrigés - Wolper - à la BU/LSV
Appréciable pour sa pédagogie, et la compréhension des concepts majeurs.

— [*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.

— [*Cro2*] Algorithms on string - Crochemore - à la BU/LSV

— [*Sip*] Introduction to the Theory of Computation - Michael Sipser - à la BU
En anglais. Intuition de la preuve donnée avant chaque preuve.

— [*Cro*] Jewels of stringology - Crochemore - à la BU/LSV

— [*Sak*] Élèments de théories des Automates - Sakarovitch - à la BU/LSV
Style austère, mais très complet, et bonne source de développements. Beaucoup de hors programme.

— [*Aho*] Compilers : Principles, Techniques, and Tools - Aho et. al - à la BU/LSV
La référence pour la compilation, surnommé le dragon. En réalité, seul deux/trois chapitres sont utiles.

— [*Kri*] Lambda-calcul, types et modèles - Jean-Louis Krivine - ?
Semble mieux que le Barendregt.

— [*Go*] Proof Theory and Automated Deduction - Jean Goubault-Larrecq - à la BU/LSV
Le must pour la logique

— [*Sch*] Compilation : analyse lexicale et syntaxique - Du texte à sa structure en informatique - R. Legendre, F. Schwarzentruber - à la BU
Écrit spécialement pour la leçon 923 de l'agrégation, donne plein d'exemples. Fait le lien entre les clauses de Horn et les algorithmes de saturation en analyse syntaxique en particulier.

— [*Da*] Introduction à la logique - R. David, K. Nour, C. Raffalli - à la BU/LSV
L'autre must pour la logique

— [*Abi*] Foundations of databases - Serge Abiteboul, Richard Hull, Victor Vianu - LSV
Bien prendre la version en anglais, la traduction française possédant pas mal d'erreurs. La référence pour la leçon bases de données.

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

— [*Win*] The formal semantics of programming langages - Winskel - à la BU/LSV

— [*Meh*] Algorithms and Data Structures - K. Mehlhorn et P. Sanders - Au LSV
Pas mal pour les algo et structures liées aux probabilités (hachage par exemple).

— [*Ull*] Principles of database systems - Jeffrey D. Ullman - BU
En anglais. Référence un peu vieille mais évoque tout ce qui était connu en 1982 sur les fondements théoriques des bases de données. Largement suffisant pour l'agrégation.

— [*Dow*] Les démonstrations et les algorithmes - Introduction à la logique et à la calculabilité - Gilles Dowek - à la BU ?
Très bien pour prendre du recul.

— [*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.

— [*Cori*] Logique mathématique Tome 1/2 - René Cori, Daniel Lascar - pas en BU
Typo assez ancienne, pas toujours agréable à lire.

— [*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.

2 Dev

— L'inclusion de requêtes conjonctives est NP-Complet - ([Abi]) - 928, 932
Réduction originale que le jury n'aura pas forcément l'occasion d'entendre souvent dans la leçon 928. Attention cependant, certains détails sont laissés au lecteurs dans [Abi]

— Minimisation de Nérode - ([Car], *Sec 1.7 p.49*) - 909

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

— Interpolation - ([Da], [Go]) - 918
Selon le système de preuve.

— Elimination des quantificateurs dans la théorie des ordres linéaires - (?) - 914,924

— Automate des occurrences - ([Cor], *p.886*) - 907,909,927
Tiens bien en 15 min, mais attention à bien maitriser les petits calculs. Extension possible vers KMP.

— Théorème de Kleene - ([Car], *Thm 1.59 p.36*) - 907,909,923
*Tout faire est ambitieux, mais cela passe si on prend les constructions les plus basiques (*Thompson *et* McNaughton-Yamada*). Si on fait par contre* Antimirov*, cela peut suffire.*

— Complétude d'un système de preuve - ([Da], [Go]) - 918
Faire la complétude du système présenté. Il y a plusieurs preuve pour chaque système.

— 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*.*

— 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.

— Problèmes indécidables pour les grammaires algébriques. - ([Car]) - 914,923
Ambigüité, universalité. Insister sur la réductio, pas sur la notion de grammaire.

— Une fonction Turing calculable est μ-recursive. - ([Wol], [Car]) - 912,913
Preuve précise mais non pédagogique dans le Cori, claire mais non précise dans le Wolper, précise mais pas finie dans le Carton ...

— Correction et complétude du système d'Armstrong - ([Ull]) - 932
Développement très simple et parfaitement dans le thème. Savoir le présenter sur un exemple. Attention, il faut supposer qu'il existe au moins deux éléments dans le domaine. Par ailleurs, il arrive tard dans le plan donc bien être sûr de pouvoir le mettre dans les 3 pages.

— Preuve de correction d'un algorithme - ([Cor], [Bea]) - 927, ?
Dijkstra, KMP, unification

— Équivalence entre deux variantes des machines de Turing - ([Sip], p136) - 912,913

— Calcul de la sémantique d'un ou plusieurs programmes - (?) - 930
Dans l'idéal, on montre que deux programmes sont équivalents, via des calculs de sémantiques, potentiellement dans des sémantiques différentes.

— Un exemple d'analyse d'un langage jouet - ([Aho], *Ch4.4 ou Ch 4.9.1 p298*) - 923
Le langage d'une calculatrice minimaliste est un classique. L'analyse d'une grammaire LL(1) est plus difficle. Ne pas hésiter à faire des dessins, avec des arbres d'interprétation possible.

— Indécidabilité de l'arithmétique de Peano - (?) - 914,924
Passer par l'encodage des fonctions calculables. Assez long si on fait tout.

— Équivalence des sémantiques opérationnelle et dénotationnelle - ([Win], *p. 61*) - 930
On peut aussi faire l'équivalence entre petit-pas et grand-pas, mais moins intéréssante.

— 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.

— Décidabilité de l'arithmétique de Presburger - ([Car], *Thm 3.63 p.164*) - 909,914,924
Idée générale simple, mais attention aux détails. Réfléchir au codage, à sa sémantique, et à la complexité globale de la construction.

— Montre que X est ou n'est pas récursive primitive - ([Cori], [Car]) - 912
X pour Ackermann*, avec une preuve un peu compliqué dans le Cori, ou X = isPrime dans le Carton. Attention,* Ackermann *est casse-geule, il y a plein de lemmes auxilliaires souvent oubliés (e.g, monotonicté de la fonction considérée).*

— Preuve de la factorielle en Hoare - ([Win], *ch6.6 p.93*) - 927
Peut être l'occasion de parler des problèmes d'automatisation.

— Borne inférieure sur la compléxité d'un tri par comparaison - ([Cor]) - 903,926
Peut être un peu court.

— Calcul de la distance d'édition - ([Cro2]) - 907,931
Bien prendre un coût de 1 *pour chaque opération, quitte à généraliser si le jury pose une question.*

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

— Complétude de la résolution propositionnelle - ([Gou] ?) - 916

— Correction des algorithmes de Prim et Kruskal - (Cor, *p. ? ?*) - 925,927
La preuve du Cormen est générale pour ce type d'algorithme.

— Tri bitonique - ([Cor]) - 903
Attention, disponible uniquement dans la seconde édition

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

— Lemme de l'étoile et variantes - ([Car], *Sec 1.9 p.61*) - 909

— Plus longue sous séquence commune - ([Cor], *Ch15,4 p341*) - 907,931
Preuve et exemple. Avoir en tête des applications (séquence d'ADN, commande diff)

— Algorithme de Cocke-Kasami-Younger - ([Hopcroft, Ullman], *Ch7.4.4, p298*) - 923
Uniquement esquissé dans le Carton. Maitriser la mise en forme normale de Chomsky*.*

— Automate d'Aho-Corasick - ([Cro2], [Bea]) - 907,909,921
Généralise KMP à plusieurs motifs. [Bea] donne bien mieux les intuitions que [Cro2]

— 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.

— Equivalence entre fonctions récursives et fonctions représentables - ([Kri], *p. 24*) - 912,929
Il suffit de faire le sens recursif est représentable, mais il faut savoir justifier facilement de l'autre direction. Être bien au clair sur le problème des fonctions partielles et des problèmes de terminaison. Savoir exprimer le lambda terme correspondant à une fonction recursive, e.g l'addition, à partir du résultat.

— $2SAT$ est NL-Complet (ou algo en temps linéaire selon les leçons) - ([Car], [Pap], [Cor] (pour les algo de graphe)) - 915,916,925

— 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.

— Indécidabilité de l'arrêt et applications à quelques problèmes indécidables - ([Sip], p159,172) - 912,913
Ne faire Rice *que si on sait bien le faire, qu'on le comprend, et qu'on sait l'appliquer.*