Leçon 931 - Schémas algorithmiques. Exemples et applications.

4 décembre 2019

1 Extraits du Rapport

Rapport de jury 2018,2019

Cette leçon permet au candidat de présenter différents schémas algorithmiques, en particulier « diviser pour régner », programmation dynamique et approche gloutonne. Le candidat pourra choisir de se concentrer plus particulièrement sur un ou deux de ces paradigmes. Le jury attend du candidat qu'il illustre sa leçon par des exemples variés, touchant des domaines différents et qu'il puisse discuter les intérêts et limites respectifs des méthodes. 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 temps comme en espace.

Rapport de jury 2017 - rapport concernant la 902, diviser pour regner

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 de jury 2017 - rapport concernant la 907, programmation dynamique

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.

2 Cœur de la leçon

- Approche gloutonne, diviser pour régner, programmation dynamique.
- Exemples (les plus classiques étant : algorithme de Dijkstra, tri fusion, plus longue sous-séquence commune).

3 À savoir

- Des exemples moins classiques (Prim ou Kruskal, Hopcroft ou Karatsuba ou recherche de paires de points les plus proches ou FFT, multiplication de plusieurs matrices ou CYK,...)
- Approche ascendante et descendante de la programmation dynamique.
- Calcul de complexité par formule de récurrence.
- Complexité en espace.
- Influence du schéma algorithmique sur les preuves de correction et de complexité.

4 Ouvertures possibles

— Des exemples de plus en plus compliqués.

— Problèmes d'optimisation et algorithmes approchés ?

5 Conseils au candidat

— Présenter les trois schémas et un unique exemple simple pour chacun risque de faire un plan peu intéressant, il faut probablement creuser l'un des schémas.

— On ne doit pas voir un catalogue de schémas et d'algorithme. Il faut absolument justifier les enchaînements, et les forces et faiblesses relative entre les différents paradigmes.

— Il faut maîtriser les ouvertures liés aux différents algorithmes présentés (prouver la correction, prouver la complexité, détails d'implémentation, est-ce le meilleur algorithme pour faire cela ?,...).

6 Questions classiques

— Donner un exemple d'algorithme glouton et d'entrée sur laquelle la solution optimale est atteinte ? Une sur laquelle elle n'est pas atteinte ?

— Quelle est la différence entre diviser pour régner et la programmation dynamique ?

— Quelle est le lien entre la programmation dynamique et la mémoisation ? Quelles en sont les différences ?

— Avantages et désavantages de l'approche gloutonne ? De diviser pour régner, de la programmation dynamique ?

— Exemple de problème ne vérifiant pas la propriété de sous structure optimal ?

— Certaines structure de données sont-elles plus pertinentes que d'autres pour la mémoisation ?

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.

— [*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.

8 Dev

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

++ 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.

\+ Correction des algorithmes de Prim et Kruskal - (Cor, *p. ? ?*) - 925,927
La preuve du Cormen est générale pour ce type d'algorithme.
Bien insister sur le caractère glouton dans cette leçon.