RAPPORT DE L'ÉPREUVE PRATIQUE D'ALGORITHMIQUE ET DE PROGRAMMATION ULC INFO 2011

Écoles concernées : ÉNS, ÉNS de Cachan, ÉNS de Lyon
Coefficients : ÉNS 4, ÉNS de Cachan 5, ÉNS de Lyon 4
Membres du jury : Loris MARCHAL, Guillaume MELQUIOND, Frédéric TRONEL

Remarques générales à propos de l'épreuve

Organisation de l'épreuve et statistiques Comme les années précédentes, cette épreuve demandait aux candidats de de mettre en œuvre la chaîne complète de résolution d'un problème informatique : analyse des spécifications, choix des structures de données, analyse de la complexité de la solution retenue, programmation, et tests. En plus de cette partie consacrée à la programmation proprement dite, une présentation orale permettait aux candidats d'une part de démontrer leurs capacités à expliciter la méthodologie qu'ils avaient suivie, d'autre part d'aborder des questions qu'ils n'auraient pas eu le temps de programmer complètement.

Cette année le jury a examiné 113 candidats, répartis en 6 sessions dont l'organisation reste identique à celle adoptée les années précédentes. À savoir, les candidats sont admis dans la salle de composition par groupe de 3 toutes les demi-heures. Les candidats ont 10 minutes pour se familiariser avec l'environnement mis à leur disposition. Ils peuvent pendant cette période poser des questions aux surveillants de l'épreuve s'ils rencontrent des difficultés d'ordre pratique. Puis l'épreuve est préparée durant 3h30 sur machine. En plus des sauvegardes sur le disque dur de la machine, une clé USB est fournie aux candidats afin qu'ils puissent sauvegarder une seconde copie de leur travail sur cette clé. Cette préparation est suivie d'une interrogation orale. Le contenu de la clé USB n'est pas examiné durant l'interrogation orale. Au total l'épreuve dure donc 4h10 heures. Ceci permet de s'assurer que les derniers candidats entrés ne croiseront pas les premiers candidats sortis. Cette année le jury a établi un barème qui équilibre les poids des parties programmation et présentation orale des résultats. Les candidats sont encouragés à présenter à l'oral leurs idées quant à la résolution de questions qu'ils n'auraient pas eu le temps de préparer durant l'épreuve pratique.

Comme les années précédentes, les langages et environnements suivants ont été proposés aux candidats :

- PC sous Windows XP avec CAML Light, Pascal (Delphi), Maple ou Java.
- PC sous Debian/Linux avec CAML Light, Objective CAML, C/C++, Pascal ou Java.

Les candidats ont choisi les langages et environnements dans les proportions suivantes :

Environnements	Windows		Linux		
Proportions	66%		34%		
Langages	Caml Light	Pascal	Objective Caml	Caml Light	C/C++
Proportions	56%	10%	17%	13%	4%

Conseils pratiques à destination des candidats et de leurs préparateurs Les conseils qui suivent ne sont spécifiques à aucun des sujets proposés cette année. Ils se veulent d'ordre général et doivent permettre aux futurs candidats de mieux aborder l'épreuve pratique d'algorithmique ainsi qu'à leurs préparateurs de les aider dans cette tâche en cernant mieux les attentes du jury.

Avant tout chose, le jury tient à rappeler qu'il maintient dans son barème un équilibre entre les points attribués à la partie programmation, c'est-à-dire aux réponses numériques que le candidat doit remplir en annexe de son sujet et qui sont corrigées de manière binaire (pas de demi-point pour une réponse approximativement exacte) et la partie algorithmique (interrogation orale à suivre). Il est **impératif** que les candidats aient préparé sérieusement cette seconde partie afin de ne pas perdre de temps (et donc de précieux points) durant l'oral. Par ailleurs, rappelons que le but du jury est de faire en sorte que chaque candidat puisse gagner autant de points que possible sur l'épreuve. Il est donc hautement recommandé d'écouter les conseils prodigués par le jury lors du déroulement de la partie orale de l'épreuve et d'être réactif afin de les prendre rapidement en considération. Rien n'est plus agaçant pour un examinateur que de voir un candidat s'obstiner dans une voie sans issue alors que ceci lui a été suggéré à plusieurs reprises. Il est aussi conseillé de ne pas chercher à impressionner l'examinateur par des références à des algorithmes avancés (tels que les algorithmes de Strassen ou Knuth–Morris–Pratt) sauf à être absolument certain qu'ils sont à-propos. Enfin, il peut parfois être demandé de mettre au point pour de petites instances de certains problèmes, des algorithmes dont la complexité est visiblement exponentielle. Ceci ne doit pas arrêter les candidats. En effet, le jury qui a mis les différents sujets au point, a pris soin de tester pour plusieurs langages que le résultat était atteignable dans des temps de calcul raisonnable.

Remarques spécifiques à chacune des épreuves

Cache de pages de Web

Ce sujet visait l'étude de différentes stratégies pour la gestion d'un cache de pages (de taille identique). Les premières questions (4, 5 et 6) étaient relativement simples à aborder, et ont été traitées par la majorité des candidats. Pour les questions 4 et 5, on pouvait simplement maintenir triée la liste des pages dans le cache afin de toujours éliminer la page en première position. La question 6 demandait de calculer également le tableau des fréquences.

Dans la partie suivante, on cherchait la solution optimale lorsque la séquence de requêtes est connue. Programmer cette solution n'introduisait pas de difficulté majeure ; la moitié des candidats ont répondu à cette question. Les questions à l'oral visaient à comparer les performances d'un algorithme *online*, qui ne connaît pas le futur, à celle d'un algorithme *offline*, qui connaît toute la séquence de requêtes. Quelques étudiants ont été troublés par la construction au vol d'une séquence de requêtes par un adversaire, mais la plupart des candidats ont bien répondu aux questions orales.

La partie suivante concernait le réordonnancement partiel de la séquence de requêtes. On demandait d'abord de programmer une stratégie donnée, un peu plus complexe que les stratégies précédentes, ce qui a été fait par un quart des candidats. Les deux questions à l'oral suivantes concernaient l'algorithme optimal dans ce cas, et ont été traitées par quelques candidats seulement. Enfin, la dernière partie sur le chargement de pages avec coûts hétérogènes a été très peu abordée.

Dominos

La question 2 admettait trois solutions de complexités temporelles et spatiales différentes et les candidats étaient interrogés à leur propos. Pour éviter de prendre en compte l'aléatoire du tirage des dominos dans les complexités temporelles, il était dit d'entrer de jeu aux candidats lors de l'oral de considérer le nombre de dominos tirés.

La question 4 demandait d'énumérer tous les trains possibles (ce qui ne veut pas dire passer en revue toutes les permutations). La taille des instances et les u_0 étaient choisis de telle sorte qu'un algorithme simple puisse résoudre la question en moins d'une seconde. Les candidats ont souvent fait des programmes bien trop compliqués (par exemple une profondeur de n^2 appels récursifs) et ont donc été confrontés à la barrière de temps.

La question 5 ressemblait à un problème de programmation dynamique mais n'en était pas un : il n'y avait en effet qu'un nombre constant de possibilités à chaque étape (poser ou non le

domino courant). Il existait donc un algorithme simple et linéaire pour le résoudre. Les candidats ont parfois eu du mal à le trouver.

La question 6 par contre se rapprochait de la programmation dynamique. Les arrangements de chaque bloc devaient être traités de façon exhaustive pour repérer toutes les paires de faces reliées par des trains de dominos. Les résultats par bloc étaient ensuite combinés par programmation dynamique pour obtenir le résultat final. Cette question et la suivant n'ont été abordée que par un cinquième des candidats.

L'algorithme de la question 7 était donné dans l'énoncé ; il s'agissait donc d'un exercice de pure programmation qui était là pour les candidats les plus rapides.

Mikado

Le sujet demandait de manipuler les coordonnées sous forme de fractions rationnelles afin de pouvoir détecter quand plus de deux droites étaient concourantes ou quand plusieurs nœuds avaient la même abscisse (les deux situations se produisaient pour les u_0 proposés aux candidats.) Il n'y avait cependant pas besoin d'y apporter un soin particulier : les entiers ne risquaient pas de déborder si les formules étaient écrites de manière raisonnable.

Les candidats ont souvent été perturbés dans leurs calculs de complexité par la présence de N^2 segments. Par exemple, les trier demande $O(N^2 \log N)$ et non pas $O(N \log N)$.

Les questions 3, 4, 5 et 7 ne présentaient aucune difficulté particulière, contrairement à la question 6 qui pouvait se révéler complexe si de mauvaises structures de données étaient employées. La question 6 pouvait être traitée par la formule d'Euler sur les graphes planaires, mais c'était un marteau-pilon difficile à mettre en œuvre et ce n'était pas le but de la question. L'énoncé fournissait suffisamment d'indications pour pouvoir traiter le problème sans connaissances particulières sur les graphes (ce n'était pas l'objet du sujet).

Les questions 8 et 9 se traitaient de façon similaire par un balayage horizontal du problème. Un bon choix des structures de données permettait de traiter chaque tranche en $O(N)$. Un tiers des candidats ont obtenu un algorithme pour ces questions mais sans avoir le temps de l'implanter.

Arbres, préfixes et suffixes

Dans ce sujet, on se proposait d'étudier l'arbre des préfixes d'un ensemble de séquences de caractères, ainsi que l'arbre des suffixes d'une séquence, dans le but de concevoir des algorithmes de recherches de motifs dans des séquences. La principale difficulté du sujet était l'utilisation des arbres d'arité quatre (ou cinq dans les parties 4 et 5) ; en général, les candidats s'en sont bien sortis, mais le temps nécessaire à la programmation ne leur a pas permis d'aborder la totalité du sujet.

Les questions 3 à 6 demandaient de construire l'arbre des préfixes pour un ensemble de séquences donné, puis de les utiliser à bon escient. Elles ont été traitées correctement par la moitié des candidats. Certains n'ont pas utilisé l'arbre pour les questions 4 à 6, ce qui n'était pas trop préjudiciable pour la 4 (au prix d'une complexité plus élevée), mais beaucoup plus coûteux pour les suivantes.

Dans la deuxième partie, on construisait l'arbre préfixe des suffixes d'une séquence, avec un algorithme très similaire à la première partie. Pour la question 8, il suffisait de réaliser que l'algorithme développé à la question 6 permettait de répondre directement. Ces questions n'ont été traitées que par trois candidats, mais la moitié des candidats a correctement répondu aux questions orales correspondantes.

Enfin, la dernière partie concernait la recherche de motifs communs à deux séquences, et a été abordée par quatre candidats seulement.

Sac à dos

Ce sujet portait sur l'étude du problème du sac à dos et plus précisément à différentes variantes du problème. Les deux premières questions portaient classiquement sur la génération d'instances

pseudo-aléatoires du problème. La partie suivante s'intéressait au problème du sac à dos avec répétition où les objets sont disponibles en quantité aussi grande que voulue. Les questions 3 et 4 portaient sur la résolution de cette variante du problème pour de petites instances. Une recherche exhaustive dans l'espace des solutions possibles était attendue. Malgré l'intitulé du paragraphe ceci semble avoir perturbé un certain nombre de candidats qui ont cherché à trouver des solutions plus intelligentes.

La question 5 portait sur un problème de programmation dynamique permettant de s'attaquer à des instances de taille plus importante. Si les trois-quarts des candidats ont réussi à traiter la question à l'oral, seule une petite moitié des candidats a réussi à la programmer. La question 6 nécessitait d'avoir traité correctement la question précédente puisqu'il fallait ici reconstituer le contenu du sac à dos à partir de la solution optimale découverte par la programmation dynamique. À noter que certains candidats ont utilisé pour ces questions un algorithme glouton qui fonctionnait car les instances générées pseudo-aléatoirement n'était pas suffisamment pathologique (en effet, elles avaient été filtrées de manière à ce qu'il n'y ait qu'une seule et unique solution optimale).

La partie suivante s'intéressait à une version fractionnaire du sac à dos, dans laquelle les objets n'existent qu'en un seul et unique exemplaire, mais peuvent être sélectionnés de manière fractionnaire. Il suffisait de développer un algorithme glouton pour obtenir les réponses correctes à la question 7. Une moitié des candidats a réussi à répondre à cette question. La justification de l'optimalité de l'algorithme ainsi obtenu n'a été traitée correctement à l'oral que par un tiers seulement des candidats.

La partie suivante s'intéressait à des sac à dos sans répétition. La structure des questions suivait exactement le même schéma que pour la seconde partie (sac à dos avec répétition). À savoir une recherche exhaustive sur de petites instances pour les questions 8 et 9, puis un problème de programmation dynamique pour la question 10. Seuls 4 candidats ont abordé la programmation de ces questions. Elles ne présentaient cependant aucune difficulté pour les questions 8 et 9 pour peu que l'on ait traité correctement les questions 3 et 4. Là encore, il ne fallait pas se laisser décontenancer par l'approche exhaustive. Les instances avaient été choisies suffisamment petites pour que les calculs puissent se faire dans des temps très courts (inférieurs à une minute).

La dernière question ne devait être traitée qu'à l'oral. On s'intéressait au cas particulier où les poids des objets suivent une suite super-croissante. On pouvait dans ce cas développer un algorithme glouton du type de celui utilisé par exemple dans l'algorithme de chiffrement de Merckle-Helmann. Un seul candidat a traité cette question.

Librairie de calcul arithmétique en précision arbitraire

Il s'agissait de développer une librairie de calcul sur les entiers en précision arbitraire. Le cœur du sujet concernait l'algorithme de Karatsuba pour la multiplication de grands entiers.

Les deux premières questions ne présentaient aucune difficulté puisqu'il s'agissait de générer les instances du problème, à savoir des entiers de longueur arbitraire. Afin de vérifier la correction des résultats obtenus, on demandait de fournir un résumé des entiers obtenus sous la forme du nombre de chiffres 3, 5 et 7 apparaissant dans l'écriture décimale des entiers. Cela ne présentait là non plus aucun problème particulier puisqu'il était conseillé aux candidats de coder leurs entiers en base 10.

La question 3 était une question préparatoire au reste du sujet. Elle avait pour but d'implémenter une fonction de comparaison entre grands entiers.

La question 4 concernait l'implémentation de l'opération d'addition. La question 5 introduisait une opération supplémentaire d'addition modulaire (sans propagation de la retenue comme pour l'opération précédente). Cette question avait pour seul but de vérifier que les candidats avaient bien implémenté l'opération d'addition. En effet les résultats attendus correspondaient à la longueur de la plus longue propagation de retenue dans le calcul d'une addition classique (même s'il n'était pas requis de s'en rendre compte pour répondre correctement à la question). La question 6 s'intéressait à l'opération de soustraction (en valeur absolue) qui était la dernière opération requise pour implémenter la multiplication selon l'algorithme de Karatsuba.

La question 7 consistait à implémenter la multiplication de deux grands entiers par l'algorithme de Karatsuba. Afin de simplifier l'implémentation on demandait un algorithme s'appliquant à des entiers dont les longueurs sont des puissances de 2. L'algorithme attendu utilise le paradigme "diviser pour régner". Son principe était largement décrit dans le sujet. La principale difficulté résidait dans le fait que les différents entiers obtenus dans les calculs intermédiaires se recouvrent partiellement. Il fallait donc traiter proprement l'addition des ces différents entiers afin d'obtenir le résultat correct. Il y avaient deux questions orales associées à cet algorithme. Il fallait d'une part détailler la complexité de l'algorithme obtenu et d'autre part expliquer pourquoi une seconde stratégie de calcul théoriquement équivalente ne devait cependant pas être utilisée en pratique.

En ce qui concerne la complexité de nombreux candidats ont été trompés par les questions suivantes où l'on supposait avoir un algorithme dont la complexité était en $O(n^{\frac{3}{2}})$. D'autres candidats ont utilisé un théorème vu en cours afin de donner directement la réponse correcte à savoir $O(n^{\log_2 3}) \simeq O(n^{1.58})$. Le jury attendait des candidats qu'ils sachent redémontrer au tableau ce résultat.

En ce qui concerne la seconde méthode de calcul, la plupart des candidats ayant traité la question 7 ont su répondre à cette question, en voyant que la stratégie proposée pouvait conduire à des débordements.

Dans les questions 8, 9 et 10, on supposait disposer d'un algorithme de multiplication de deux entiers de longueurs n_1 et n_2 et dont la complexité est en $O(\max(n_1, n_2)^{\frac{3}{2}})$. On s'intéressait alors au coût total d'une chaîne de multiplications de grands entiers dont les tailles ne sont pas homogènes. Il est clair que dans ce cas l'ordre dans lequel on effectue les multiplications peut faire varier le coût total dans des proportions considérables. Dans la question 8, il fallait calculer le coût lorsque les multiplications sont effectuées dans l'ordre d'apparition des entiers. La question 9 était une simple variation lorsque les multiplications sont effectuées dans l'ordre croissant des tailles des entiers. Enfin la question 10 était ouverte puisqu'il s'agissait pour les candidats de proposer une stratégie capable d'approcher voire de dépasser les coûts obtenus par une stratégie s'inspirant de celle utilisée à la question 9. En effet, les coûts à approcher étaient obtenus en multipliant à chaque étape les deux entiers dont les longueurs étaient minimales. La longueur du produit partiel ainsi obtenu était approximée par l'addition des longueurs des deux termes multipliés. Le procédé était alors appliqué itérativement jusqu'à épuisement des entiers à multiplier. Certains candidats ont pensé à l'algorithme de multiplication d'une chaîne de matrice par programmation dynamique. Cependant ici on ne pouvait pas appliquer cette stratégie car la multiplication des entiers est commutative. Il y a donc un nombre d'arbres à explorer qui est exponentiellement plus grand que dans le cas de la multiplication d'une chaîne de matrices.