RAPPORT DE L'ÉPREUVE PRATIQUE D'ALGORITHMIQUE ET DE PROGRAMMATION ULC INFO 2012

Écoles concernées : Cachan, Lyon, Paris
Coefficients : Cachan 5, Lyon 4, Paris 4
Membres du jury : Pierre-Alain FOUQUE, Loris MARCHAL, Guillaume MELQUIOND

Remarques générales à propos de l'épreuve

Organisation de l'épreuve et statistiques Comme les années précédentes, cette épreuve demandait aux candidats 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 leur capacité 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 126 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. Le nombre de candidats est limité chaque jour de telle sorte que les derniers candidats entrés ne croiseront pas les premiers candidats sortis.

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

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

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

Environnements	Windows		Linux		
Proportions	67%		33%		
Langages	Caml Light	Pascal / Maple	Objective Caml	Caml Light	C/C++
Proportions	66%	1%	20%	6%	7%

Conseils pratiques à destination des candidats et de leurs préparateurs Avant toute 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. 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 programmer durant l'épreuve pratique.

Qui plus est, la partie orale de l'épreuve peut porter sur n'importe quel point de la partie écrite et pas seulement sur ceux explicitement marqués comme tels dans le sujet. En particulier, le jury attend des candidats qu'ils puissent expliquer la façon dont ils ont obtenu les réponses aux questions et qu'ils aient une idée de la complexité de leur approche. Certains candidats (heureusement peu nombreux) arrivent à l'épreuve orale en ayant complètement oublié ce qu'ils ont bien pu programmer deux heures plus tôt ; n'ayant pas non plus noté sur leurs brouillons de quoi se rafraîchir la mémoire, ils se retrouvent donc grandement handicapés pour la suite de l'épreuve.

Remarques spécifiques à chacune des épreuves

Repliements d'ARN

Ce sujet s'intéressait à la structure secondaire des molécules d'ARN. Le modèle retenu pour le sujet était une version simplifiée du modèle utilisé en bio-informatique pour ne pas noyer l'aspect algorithmique sous des détails physico-chimiques. Les questions 1 et 2 permettaient aux candidats de s'assurer qu'ils avaient bien compris les définitions. La question 3 s'apparentait à une recherche de motifs répétés dans une chaîne. Il n'était pas demandé aux candidats de faire mieux que l'algorithme naïf de complexité cubique.

Les questions 4, 5, 6 s'intéressaient aux propriétés d'un repliement particulier défini à l'aide d'un ordre lexicographique. La difficulté principale était ici de construire ce repliement. Malheureusement, certains candidats n'ont pas su déchiffrer correctement ce qu'était un ordre lexicographique et non donc pas pu faire ces trois questions.

Le dernier bloc de questions (7, 8, 9) était un exercice de type programmation dynamique. Le sujet contenait une indication sur la façon de découper le problème en sous-problèmes avant la question 7. Les candidats qui se sont penchés sur cette question ont parfois proposé des relations de récurrence beaucoup plus compliquées que nécessaire. Peu de candidats ont eu le réflexe de stocker les valeurs pour ne pas les recalculer, et même quand la suggestion leur en était faite à l'oral, ils n'ont pas su proposer des méthodes simples comme la mémoïsation. Remarque : la question 7 était plus simple que ce qu'on appelle traditionnellement de la programmation dynamique puisque toutes les valeurs calculées devaient être conservées et il n'y avait donc pas d'optimisation en espace possible. Les questions 8 et 9 utilisaient exactement le même algorithme de complexité cubique que la question 7, mais avec des relations de récurrence de subtilité croissante.

Variance et intervalles

Ce sujet s'intéressait au calcul de la variance et au produit de matrices quand les entrées ne sont pas connues exactement mais sont données par des intervalles de valeurs réelles. Cette épreuve s'appuyait sur une part non négligeable de propriétés arithmétiques, mais tous les résultats étaient donnés explicitement dans le sujet et les candidats devaient simplement les justifier à l'oral.

Les questions 2 et 3 ne présentaient pas de difficultés. Certains candidats ont cependant utilisé des tableaux bien plus grands que nécessaire pour résoudre la question 2 (128000 au lieu de 400).

La question 4 portait sur le calcul de la variance maximale et nécessitait de faire une analyse exhaustive dont la combinatoire était exponentielle. L'indication qui précédait cette question expliquait comment restreindre le nombre de possibilités pour éviter l'explosion du nombre de cas. Malheureusement, nombre de candidats ont supposé que l'indication, malgré sa position dans le sujet, s'appliquait à la question 5 et non pas 4 et n'ont donc pas pu attaquer l'instance de la question 4.c.

Les questions 5 et 6 permettaient de réduire plus encore le nombre de possibilités pour que l'explosion combinatoire (le problème de la variance maximale est de toutes façons NP-complet) n'empêche pas d'attaquer des instances plus grosses. Le cas particulier de la question 5 suggérait l'utilisation d'un algorithme de découpage en sous-problèmes pour traiter le cas général de la

question 6. Cependant, certains candidats ont plutôt proposé un algorithme de raffinement pour la question 6, ce qui permettait de traiter les instances 6.a et 6.b seulement.

La question 7 portait quant à elle sur la variance minimale. Il y avait à nouveau un découpage en sous-problèmes à effectuer. La complexité temporelle ne posait aucune difficulté pratique cette fois-ci. Aussi bien pour la question 6 que pour la 7, les questions orales ont tourné autour des découpages optimaux du point de vue de la complexité.

La dernière partie du sujet était plus simple et portait sur le produit de matrices d'intervalles. Les questions 8 et 9 servaient à vérifier que les candidats avaient bien compris les principes de bases de l'arithmétique d'intervalles décrits dans l'énoncer et comment les implanter. La question 10 était un petit puzzle consistant à trouver une formule pour le carré de matrice qui soit optimale vis-à-vis de l'arithmétique d'intervalles. Il n'y avait aucune difficulté de programmation pour ces trois dernières questions.

Réseaux d'interconnexion

Ce sujet s'intéressait à la construction de réseaux d'interconnexion de points du plan. Après avoir brièvement vérifié que les points générés ne comptait pas de points multiples (question 2), on s'intéressait d'abord à un cas particulier simple dans le cas rectilinéaire (question 3). On étudiait ensuite l'algorithme de Prim, détaillé dans le sujet, ainsi que le degré maximal des réseaux construits par cet algorithme (questions 4 et 5). L'introduction de points relais venait alors compliquer le problème, avec la recherche de solution optimale pour de petits ensembles de points relais (question 6), puis avec une adaptation de l'algorithme de Prim (question 7). Les deux dernières questions étudiaient des caractéristiques plus avancées des réseaux produits à la question 5.

La quasi-totalité des candidats s'est cantonnée pour la programmation aux questions 1 à 5. Quelques rares candidats ont remarqué à la question 3 que l'abscisse recherchée n'était pas toujours la médiane (cas de plusieurs points de même ordonnée), alors que les valeurs numériques données sur la fiche réponse type ne prenaient pas en compte ce cas particulier. Bien entendu, nous avons accepté les deux réponses. La question 4 demandait simplement de retranscrire l'algorithme proposé, ce qui a était fait par une large majorité de candidats. À la question 5, il suffisait d'enrichir le code précédent pour se souvenir du degré de chaque nœud. La question 6, programmée correctement par un seul candidat, requérait une recherche exhaustive du sous-ensemble de points relais, pour pouvoir appliquer l'algorithme précédent. La moitié des candidats a tout de même proposé cet algorithme à l'oral. La question 7 demandait une gestion attentive des cas particuliers, mais n'introduisait pas de difficultés algorithmiques. La question 8 a été traitée par 3 candidats, souvent en utilisant des algorithmes de graphes lourds (en $O(n^4)$), alors que le problème était beaucoup plus simple ici. Il était possible de résoudre les questions 8 et 9 en s'apercevant que le réseau construit était un arbre et en utilisant des parcours d'arbres. Cependant, l'utilisation d'arbres par les candidats n'était pas attendue pour répondre à ces questions : il était possible de les résoudre en enrichissant l'algorithme de Prim de la question 5.

Cartographie d'arbres

Ce sujet s'intéressait à deux problèmes algorithmiques liés aux arbres : le routage et la pagination (présentée comme une cartographie). Le premier problème (routage), le plus simple, était abordé dans les questions 3 et 4, qui pouvaient toutes deux se résoudre par des parcours d'arbres en profondeur. La deuxième partie s'intéressait à la pagination d'arbres et son optimisation pour minimiser certaines métriques (coûts d'accès moyen et maximal aux feuilles). Après avoir étudié une pagination simple (question 5), on guidait les candidats vers des méthodes plus complexes. La question 6 étudiait le coût maximum et guidait les candidats vers une solution récursive étudiant à la fois le coût maximum et la taille de la page contenant la racine. La question 7 proposait de résoudre le problème du coût moyen ; la solution était là encore récursive, mais en utilisant plus de résultats intermédiaires : un peu à la façon de la programmation dynamique, il fallait calculer un tableau pour chaque nœud de l'arbre. Les dernières questions élaboraient sur cette solution en réutilisant les résultats de la question 6.

Malheureusement, seuls deux candidats ont dépassé en programmation la question 6, les autres se sont cantonnés aux simples questions de routage. Certains candidats ont eu les plus grandes difficultés à concevoir et à mettre en œuvre des algorithmes de parcours d'arbres un peu élaborés (question 4 et 5), ce qui est assez décevant. La plupart des candidats qui ont abordé ces questions ont proposé des algorithmes de complexité élevée, alors qu'il n'était pas très difficile d'aboutir à des algorithmes linéaires. L'interrogation orale a permis d'aborder les premières questions de la partie 3 avec la plupart des candidats.

Chaînes d'addition

Ce sujet s'intéressait au calcul d'une chaîne d'addition minimale pour calculer efficacement des exponentiations. La question 2 demandait de calculer la représentation binaire d'entiers et la question 3 proposait d'utiliser ces représentations pour calculer la chaîne d'addition en utilisant la méthode binaire d'exponentiation rapide. À la question 4, un algorithme récursif était suggéré en utilisant la méthode du plus petit facteur premier et la méthode de la question 5 concernait le calcul de l'arbre des puissances de Knuth. Il était possible de le calculer étage par étage en se souvenant pour chaque noeud de son père car il suffit de savoir remonter dans l'arbre. Les questions 6 et 7 considéraient de calculer en base 2^k en utilisant du précalcul. Les questions 8 et 9 suggéraient de considérer des chaînes sur l'alphabet $\{-1, 0, 1\}$ pour diminuer le nombre de bits à 1 dans les écritures binaires.

Les premières questions ne posaient pas de difficulté de programmation. La majorité des candidats ont bien programmé la question 4. Seule la question 5 présentait quelques problèmes de programmation et beaucoup des candidats qui l'ont abordé ont programmé un arbre d'arité arbitraire. Malheureusement, peu de candidats ont abordé les questions 6 et 7. Les questions 8 et 9 ont été faites à l'oral et ne présentaient pas de difficulté de programmation.

Le lièvre et la tortue

Ce sujet s'intéressait à différentes stratégies pour détecter un cycle dans les itérations d'une fonction aléatoire. La question 2 demandait d'utiliser de la mémoire. Les candidats ont bien souvent stocké les itérés de la fonction au lieu de stocker les indices dans la case x_i. La question 3 suggérait de calculer des ensembles pour simuler la programmation d'une table de hachage. Certains candidats ont compris et ont programmé une table de hachage dès la question 2, ce qui n'était pas demandé. La partie 3.1 demandait de programmer l'algorithme de Floyd, le calcul de la longueur de la pré-période et du cycle et du calcul de collision. Certains candidats n'ont pas vu comment calculer la pré-période et la collision à partir de l'équation donnée. La partie 3.2 consistait à modifier l'algorithme précédent en utilisant une nouvelle suite grâce à l'algorithme de Brent et la partie 3.3 son application à la factorisation. Malheureusement, très peu de candidats ont utilisé un calcul de pgcd dans l'algorithme de Brent. La partie 4 proposait de programmer l'algorithme de Nivasch en utilisant une ou plusieurs piles suivant la variante.

Beaucoup de candidats ont très bien réussi cette épreuve qui ne présentait pas beaucoup de difficulté de programmation mais demandait de bien comprendre les diverses stratégies. Les résultats de la question 3 présentaient une petite erreur (la première valeur était la pré-période et non i_0 qui correspondait à la somme de la longueur de la pré-période et de la période) et peu de candidats ont correctement programmé cette question. La programmation des parties 3.1 et 3.2 ne posait pas de problème particulier et ces parties ont été bien traitées par les candidats. La majorité des candidats n'a pas abordé la partie 4.