Rapport du Jury du Concours 2010
Épreuve Pratique d'Algorithmique et de Programmation (EPAP)

Loris Marchal, Guillaume Melquion, Frédéric Tronel

21 juin 2011

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'il avait suivie, mais aussi d'aborder des questions qu'ils n'auraient pas eu le temps de traiter complètement.

Cette année le jury a examiné 116 candidats, répartis en 5 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. L'épreuve est préparée durant 3h30 sur machine, suivie d'une présentation d'une demi-heure devant un jury. Au total l'épreuve dure donc 4 heures. Ceci permet de s'assurer du fait que les derniers candidats entrés ne croiseront pas les premiers candidats sortis. Cette année le jury a pris le parti d'établir un barème favorisant légèrement la partie résultats *pratiques* par rapport à la présentation orale des résultats (environ 1.5 fois plus de points pour la première partie par rapport à la seconde). Ceci ne doit cependant pas dissuader les candidats de présenter à l'oral leurs idées quant à la résolution de question qu'ils n'auraient pas eu le temps de préparer durant l'épreuve pratique.

Comme les années précédentes, les langages et environnements suivants 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 :

Langages	CAML Light	CAML Light	Objective CAML	Pascal	C/C++	Maple
Environnement	Windows	Linux	Linux	Windows	Linux	Windows
Proportion	50%	18%	15%	10%	6%	1%

Conseils pratiques à destination des candidats et de leurs préparateurs Pour commencer, le jury souhaiterait souligner qu'il a particulièrement apprécié l'enthousiasme et le dynamisme des candidats. Nous tenons cependant à formuler quelques critiques et conseils pratiques aux futurs candidats ainsi qu'à leurs préparateurs. Nous espérons que ceux-ci pourront leurs être utiles.

Tout d'abord et comme il est souvent d'usage, nous rappelons aux candidats qu'il est nécessaire de bien lire l'ensemble du sujet avant de commencer l'épreuve. Certaines questions peuvent être traitées assez aisément pour peu que l'on ait pris le temps de bien lire l'énoncé et d'y réfléchir un peu. Bien souvent ces questions dépendent de questions plus complexes à traiter. Dans ce

cas, et même si le candidat n'est pas parvenu à traiter la question complexe dont dépend une question plus simple, il est toujours de bon ton de montrer à l'examinateur lors de l'oral que l'on a bien compris la logique qui sous-tend l'enchaînement des questions du sujet. Ceci peut rapporter quelques points précieux.

Même si cette épreuve se veut pratique, il est nécessaire de passer du temps à préparer l'épreuve orale qui lui est associée. Presqu'une moitié des points lui sont réservés. Il est donc utile d'avoir réfléchi à la complexité des algorithmes que le candidat aura implémenté. Le manque de réflexion préalable de certains candidats leur fait dire des énormités. C'est ainsi que certains algorithmes visiblement exponentiels peuvent devenir miraculeusement linéaires. Par ailleurs, il est parfois préférable d'avoir réfléchi sérieusement aux algorithmes à implémenter pour répondre aux différentes questions que de s'acharner sur l'implémentation d'une question qui résisterait aux efforts de mise au point. Le jury est bien conscient que la longueur de l'épreuve ne permet pas toujours d'arriver à une implémentation pleinement fonctionnelle du premier coup, et que la fatigue aidant il est parfois difficile de détecter les erreurs subtiles qui ne manquent pas d'apparaître lorsqu'il s'agit de programmer complètement un algorithme. Dans ce cas, le jury conseille ceci :

- soit il s'agit des toutes premières questions de l'épreuve. Dans ce cas les candidats sont encouragés à les implémenter et à en vérifier la correction à l'aide des exemples pour lesquels la réponse attendue est donnée par le sujet. Le jury s'attend à ce que les candidats soient capables de mettre au point leurs algorithmes sur de petites instances pour lesquels les calculs peuvent être menés à la main (par exemple).
- S'il s'agit de questions plus avancées dans le sujet, les candidats sont invités à les préparer pour un traitement uniquement à l'oral. Rien ne sert de s'acharner sur une question qui va consommer l'ensemble du temps de préparation et qui ne permettra pas au candidat de prouver sa juste valeur durant l'épreuve orale.

Outres ces remarques d'ordre général, le jury a constaté quelques points plus liés aux connaissances des candidats.

Par exemple, les candidats maîtrisent en général parfaitement les arbres binaires, mais nombre d'entre eux éprouvent des difficultés à implémenter des arbres d'arité quelconque.

La plupart des sujets nécessitent de précalculer certaines valeurs et de les stocker (dans un tableau) afin de pouvoir retrouver ces valeurs en temps constant. C'est souvent le cas pour la suite aléatoire initiale. Ceci n'est pas forcément indiqué dans le sujet mais les candidats doivent avoir le réflexe d'y penser par eux-mêmes.

Certains candidats semblent penser que seuls les algorithmes récursifs trouveront grâce aux yeux des examinateurs. Ceci est évidemment faux. Le jury attend des algorithmes qui répondent correctement aux questions posées dans le sujet. Parfois un algorithme récursif est particulièrement élégant, alors qu'à d'autres moments un algorithme purement itératif sera plus simple à expliquer et mettre en œuvre.

Finalement, les examinateurs ont constaté que les candidats qui avaient choisi Pascal comme langage de programmation ont très souvent rencontré les pires difficultés pour mettre en œuvre les algorithmes demandés par les sujets. Ceci s'explique par le fait que le compilateur Pascal fourni dans l'environnement mis à disposition des candidats possède de nombreuses restrictions qui le rendent peu utilisable. En conséquence de quoi nous déconseillons son utilisation.

Remarques spécifiques à chacune des épreuves

Gestion d'un entrepôt Ce sujet visait l'étude et l'optimisation d'un processus décrit sous la forme d'un arbre de tâches, sous l'angle de la minimisation de l'espace de stockage des résultats temporaires. Après quelques questions directes sur la création des arbres et leur caractéristiques (questions 2 et 3), des stratégies d'ordonnancement simples était étudiées. La question 4 se concentrait sur un parcours en profondeur, alors que la question 5 demandait un algorithme glouton, qui traitait à chaque instant le nœud optimal au sens d'une métrique donnée. La partie suivante s'intéressait au cas homogène, lorsque tous les résultats temporaires sont de taille unité. Une question à l'oral devait mettre les candidats sur la voie de la stratégie optimale, qui pouvait ensuite

être implantée à l'aide d'un parcours en profondeur dans la question 6. Les parties suivantes recherchaient la stratégie d'ordonnancement optimale, d'abord sur un type particulier d'arbres (les harpons), puis sur les arbres quelconques.

On attendait a priori des bons candidats qu'ils traitent les questions 2 à 6 et abordent les parties suivantes. Seuls un tiers des candidats y sont parvenus. Il semble que quelques candidats n'aient pas su comment coder des arbres dont l'arité n'était pas fixée à l'avance. Les questions 4 et 5 ont donc été particulièrement discriminantes. En revanche, 5 candidats (sur 26) ont pu aborder la partie 4.2 sur les harpons, et réfléchir à des stratégies de résolution.

Satisfiabilité Ce sujet portait sur l'étude de la satisfiabilité de formules propositionnelles en forme normale conjonctive. Les questions 2 à 6 portaient sur la création des formules et leur simplification par propagation des clauses unitaires. Les formules simplifiées étaient ensuite résolues par un arbre binaire de décision (question 7). La fin du sujet était indépendante de ce qui précédait. Elle portait sur le cas particulier des clauses à deux littéraux et leur résolution par recherche de chemins (questions 8 et 9) et de composantes fortement connexes (questions 10 à 12).

Les questions 2 à 6 et 8 étaient de simples parcours de listes/tableaux. La question 7 demandait un parcours d'arbre. Les questions 9 et 10 un parcours de graphe en profondeur. Les questions 2, 4 et 8 demandaient de correctement lire l'énoncé, sous peine de calculer d'autres valeurs que celles demandées.

A priori, les bons candidats devaient atteindre la question 9. En pratique, une dizaine de candidats ont traité la question à l'oral dont six en détail ; trois d'entre eux l'ont implantée. Plus généralement, la moitié des candidats ont programmé la question 7. La question 8 a été implantée par une dizaine de candidats. Les questions 11 et 12 quant à elles ont peu ou pas été abordées.

Les questions discriminantes pour ce sujet ont été les questions 7 à 9.

Ruée vers l'or Ce sujet étudiait le partitionnement de tableaux en une ou deux dimensions, afin d'obtenir des partitions dont le poids est équilibré. La plus grande partie du sujet (questions 2 à 9) concernait le partitionnement à une dimension, et la fin (questions 10 à 11) étudiait le cas à deux dimensions.

Les premières questions (questions 4 et 5) demandaient aux candidats de coder une stratégie de découpage récursive. Les questions suivantes étudiaient le partitionnement optimal, c'est-à-dire celui dans lequel la plus grosse partition à un poids minimal. Pour obtenir le partitionnement optimal, on demandait d'abord un algorithme simple, de complexité quadratique dans la taille du tableau, puis on cherchait à réduire sa complexité temporelle, à l'aide d'algorithmes de plus en plus évolués. Le premier algorithme demandé (question 6) s'appuyait sur une relation de récurrence, donnée dans le sujet, pour définir la valeur optimale du partitionnement. Il s'agissait alors d'utiliser la programmation dynamique pour obtenir la valeur voulue. À l'aide de questions orales, on guidait alors les candidats vers une simplification de cette relation de récurrence, afin de limiter la portée de la boucle calculant le minimum dans la relation de récurrence. Ces remarques devaient permettre d'obtenir un algorithme de complexité linéaire dans la taille du tableau (question 7). D'autres optimisation étaient proposées dans les questions 8 et 9 afin d'obtenir un algorithme de partitionnement de complexité logarithmique dans la taille du tableau.

La taille des instances sur lesquelles les candidats devaient tester leurs algorithmes était choisie de telle sorte que seul un algorithme de la complexité voulue (ou inférieure) puisse donner une réponse en un temps raisonnable.

Enfin, les dernières questions (10 et 11) s'intéressaient au cas bi-dimensionnel, d'abord avec une stratégie simple fondée sur le partitionnement mono-dimensionnel, puis à l'aide d'une relation de récurrence à trouver, et à utiliser avec un algorithme de programmation dynamique.

La moitié des candidats a traité la question 6, les bons candidats ont terminés la partie mono-dimensionnel, et un candidat a même terminé le sujet. Une difficulté de ce sujet était la gestion correcte des indices lorsqu'on parcourt le tableau pour le découper. Une autre difficulté était de reconnaître que la relation de récurrence donnée pour la question 6 ne devait pas s'implanter telle qu'elle, sous peine de recalculer beaucoup de fois les mêmes résultats. Quelques candidats ont eu ce

problème, ils ont parfois pu se corriger à l'oral. Enfin, on attendait que les candidats connaissent la complexité d'une recherche dichotomique dans un tableau trié, même s'ils implantaient un algorithme plus simple (par exemple à la question 4). De nombreux candidats se sont contentés d'une complexité linéaire, malgré nos questions insistantes, probablement du fait que ce tableau (les sommes partielles) n'était pas explicitement décrit comme trié.

Malgré les indications données dans l'énoncé, un candidat n'a pas pré-calculé et stocké les valeurs des sommes partielles, ce qui a très vite rendu la complexité des algorithmes prohibitive.

Recherche de motifs Ce sujet portait sur le comptage d'occurrence d'un motif dans un texte à base d'algorithmes du type Boyer-Moore, c'est-à-dire des algorithmes qui n'ont pas besoin de lire toutes les lettres du texte. Les différentes approches étaient chacune traitées en deux temps : précalcul des tables (questions 5, 7, 10 et 12) puis mise en œuvre des algorithmes (questions 6, 8, 9, 11 et 12). Les questions 1 à 4 servaient à poser le décor et à fournir des moyens de valider les résultats suivants.

Le précalcul des tables posaient des questions d'algorithmique qui ont été abordées à l'oral. La mise en œuvre des algorithmes ne demandait par contre qu'une lecture attentive de l'énoncé et un peu de rigueur dans la programmation. Exceptées les 12 et 13, toutes les questions pouvaient être implantées par des algorithmes itératifs.

A priori, les bons candidats devaient atteindre la question 11. En pratique, nombre de candidats ont été bloqués par la notion d'automate et n'ont pas touché aux questions 10 et 11 ; le sujet ne demandait pourtant aucune connaissance particulière. La moitié des candidats ont implanté la question 6, un quart la question 9. Peu de candidats ont su trouver un algorithme quadratique pour la question 7 alors que la plupart ont su trouver un algorithme linéaire pour la 5.

Les questions discriminantes pour ce sujet ont été l'implantation des questions 6, 8 et 9 et l'oral de la question 7.

Pyraminx Ce sujet portait sur la résolution du Pyraminx, un casse-tête similaire au célèbre Rubik's cube, mais se présentant sous la forme d'une pyramide. La combinatoire du Pyraminx est bien plus faible que celle de son homologue hongrois. Ceci permet d'envisager sa résolution de manière exhaustive par une machine en un nombre minimal de mouvements. Ce sujet comportait un grand nombre de pages (20 pages) dont les 12 premières étaient consacrées à des notations permettant de manipuler formellement les configurations, ainsi que les mouvements pouvant être appliqués au Pyraminx. Ces notations introduisaient entre autres, deux types de configuration pour décrire l'état du Pyraminx. L'une était qualifiée de complète et l'autre de compacte. La configuration compacte tirait parti des symétries du casse-tête afin de réduire la quantité d'information nécessaire à la description d'un état du jeu. Elle permettait donc de diminuer la combinatoire du jeu. Cette première difficulté passée, les candidats pouvaient commencer à programmer les premières questions du sujet.

Les deux premières questions permettaient de mélanger de manière pseudo-aléatoire le Pyraminx afin de l'amener dans une configuration où les couleurs des quatre faces ne sont plus uniformes. Il n'y avait aucune difficulté, mais ces questions permettaient aux candidats de vérifier qu'ils avaient bien compris et assimilé les différentes notations introduites par le sujet.

La question 3 était consacrée au calcul d'une bijection entre les permutations et les entiers. Les permutations en jeu opéraient sur un ensemble suffisamment petit pour qu'il ne soit pas nécessaire d'avoir recours à des calculs en précision arbitraire. Ceci permettait de répondre à la question 4 qui avait pour but d'associer un unique entier à une configuration compacte. La question 5 était consacrée au calcul inverse, à savoir la détermination d'une configuration compacte à partir de l'entier qui lui a été associé à la question 4. Ceci complétait donc la détermination d'une bijection entre les configurations compactes et les entiers.

La question 6 était indépendante du reste du sujet, elle permettait de retrouver la configuration complète associée à une configuration compacte.

La question 7 était consacrée à la construction de toutes les configurations atteignables depuis l'état initial en k pas. Il fallait pour cela construire un graphe des configurations atteignables

depuis l'état initial, en utilisant la bijection construite à la question 5 pour déterminer de manière rapide si un état avait déjà été rencontré.

La question 8 était quant à elle consacrée au calcul d'une plus courte suite de mouvements permettant de replacer le Pyraminx dans son état initial à partir d'une configuration mélangée. Il convenait de parcourir le graphe créé à la question précédente en suivant les arcs dans l'ordre inverse.

De manière générale la longueur du sujet a rendu difficile le traitement des questions. Aucun candidat n'a pu aborder les questions 6,7 et 8 en terme de programmation sur machine. Les bons candidats ont par contre été capables de traiter ces questions à l'oral.