ÉPREUVE PRATIQUE D'ALGORITHMIQUE ET DE PROGRAMMATION
ENS : PARIS — LYON

Coefficient **: 4**

MEMBRES DE JURYS : L. BOUGÉ, Y. ROBERT

L'objectif de cette nouvelle épreuve est d'évaluer les capacités des candidats à mettre en œuvre de manière cohérente la chaîne *complète* de résolution d'un problème informatique : analyse de la spécification abstraite d'un problème, conception d'un algorithme, évaluation de son coût, programmation sur machine dans l'un des langages proposés, exécution sur un jeu de valeurs tests, discussion des résultats obtenus.

Il s'agit donc d'une épreuve transversale, complémentaire des épreuves écrites, plutôt centrées sur les aspects formels et mathématiques, et des épreuves orales, plutôt centrées sur les aspects algorithmiques. Cette nouvelle épreuve permet aux candidats de mettre en valeur leur maîtrise d'un environnement informatique et d'un langage de programmation au service d'une démarche scientifique de résolution de problème et d'évaluation des diverses stratégies possibles. L'étalement des notes obtenues montre que cette épreuve a bien rempli son rôle, en faisant émerger un lot de candidats particulièrement brillants.

Le jury a examiné 66 candidats (que des garçons !), répartis en 5 sessions de 4 heures chacune. Les candidats avaient le choix entre plusieurs plates-formes de programmation :
– PC sous Windows 98, avec Caml Light 0.74 et Maple 7 ;
– PC sous Linux/KDE, avec Caml Light et OCaml 3.04 (sous XEmacs/Tuareg), C (sous KDevelop 2.0 ou XEmacs, compilateur `gcc`) et Pascal (XEmacs, compilateur `gpc`).

Les environnements Caml utilisés sont disponibles librement sur le serveur INRIA `http://caml.inria.fr/`. Les compilateurs `gcc` et `gpc` sont disponibles librement à partir du serveur `http://www.gnu.org/`. L'éditeur XEmacs est disponible librement sur le server `http://www.xemacs.org/`.

Il était demandé aux candidats de choisir l'une des plates-formes et l'un des langages par avance. Les trois quarts des candidats ont choisi de programmer en Caml Light sous Windows. Seulement deux candidats ont utilisé Maple. Le reste des candidats ont choisi de composer sous Linux/KDE, et se sont répartis de manière équilibrée entre Caml Light, OCaml, Pascal et C.

Les sujets ont été préparés selon plusieurs exigences.
– L'épreuve devait être évaluable *uniquement* à partir des copies rendues par les candidats (réponses aux questions et résultats numériques finaux). Les disquettes rendues par les candidats n'ont été utilisées qu'à titre exceptionnel. Il n'a *pas été tenu compte* de la structuration des programmes, ou même de la possibilité de rejouer leur exécution après l'épreuve.
– Les problèmes posés ne devaient pas faire intervenir de notion conceptuelle complexe pour pouvoir être abordés par tous les candidats. Nous nous sommes volontairement restreints à des parcours de tableaux ou de matrices.
– Les solutions devaient être programmables dans l'ensemble des langages proposés. Ceci excluait en particulier tout sujet reposant explicitement sur des structures de données dynamiques comme des listes ou des arbres.
– Les résultats devaient être reproductibles sur l'ensemble des plates-formes disponibles. Ceci

excluait l'usage des générateurs aléatoires ou des bibliothèques mathématiques spécifiques aux environnements. Les calculs flottant ont aussi été utilisés avec la plus grande prudence pour ne pas dépendre des erreurs d'arrondi.
- Les problèmes posés devaient, si possible, être abordables par *plusieurs algorithmes*, avec des degrés de raffinement (et donc de coût) échelonnés. L'objectif est que l'algorithmique ne soit pas un obstacle majeur pour les candidats, mais que les meilleurs candidats puissent trouver matière à exprimer librement leur talent. Typiquement, il existait toujours un algorithme simple mais coûteux (par exemple, $O(n^4)$), mais aussi des algorithmes plus élaborés moins coûteux (par exemple, $O(n^2)$). La réponse à certaines questions par un algorithme simpliste (mais correct) pouvait pousser les plates-formes à leurs limites matérielles (taille des entiers, espace mémoire, etc.) ou conduire à des temps de calcul déraisonnables dans ce cadre (plusieurs minutes, voire des heures !)
- Des efforts ont été faits pour permettre aux candidats de *tester* leurs algorithmes et leurs implémentations sur des versions réduites des problèmes proposés, afin de pouvoir vérifier la correction de leurs résultats avant de passer à l'échelle. Par exemple, il était suggéré d'appliquer la méthode d'abord à une chaîne de longueur 10 avant de considérer une chaîne de longueur 10 000. Malheureusement, certains candidats n'ont pas compris l'intérêt de ces préliminaires : ils se sont lancés "tête baissée" dans des calculs gigantesques... mais faux.

On trouvera en annexe à ce rapport quelques sujets proposés (mais pas les solutions !)

Une remarque générale du jury est la suivante. La plupart des candidats ont été capables de proposer un algorithme pour résoudre les problèmes posés, et d'en évaluer grossièrement la complexité, ce qui est bien. Par contre, la mise en œuvre *concrète* des algorithmes proposés a souvent été maladroite, et source de nombreuses erreurs. L'évaluation de l'épreuve étant essentiellement fondée sur la correction numérique des résultats, ce point ne peut être négligé.

La plupart des erreurs sont dues à l'incapacité des candidats de *réfléchir* quelques minutes crayon en main avant de se précipiter sur le clavier de leur PC. Cette hâte est souvent fatale. L'un des problèmes demandait par exemple de parcourir des matrices binaires (dont les éléments sont des 0 ou des 1) pour détecter la présence d'un motif particulier. Tous les candidats (ou presque !) réussissent à compter correctement le nombre de 1 dans une matrice binaire carrée de taille $n \times n$. Par contre, beaucoup peinent à trouver le nombre de carrés de taille $p \times p$ composés uniquement de 1 dans une telle matrice, même par la méthode la plus simple : pour chaque position (i, j) de la matrice, tester si tous les élément du carré $[i..(i + p - 1), j..(j + p - 1)]$ sont des 1 (on peut évidemment faire beaucoup plus efficace). En fait, la plupart des candidats se trompent dans les indices des boucles externes sur i et j. Caml ou Maple détectant les débordements de tableaux, il y a peu d'erreurs par excès. Par contre, il y a de nombreuses erreurs par défaut, les carrés du bord de la matrice n'étant pas complètement explorés. Nous pourrions multiplier de tels exemples !

Il ne s'agit pas là d'un manque de préparation ou de talent. Il s'agit d'un *manque de méthode scientifique* pour la conception et la mise en œuvre des algorithmes. Nous sommes par exemple désolés du faible nombre de candidats qui s'aident de croquis, alors que cela aurait été d'une grande aide dans le cas ci-dessus. De même, quelques minutes de réflexion sur papier auraient souvent suffi à éviter des heures de débogage pénible et surtout bien incertain. Enfin, tester le programme sur une petite sous-matrice, par exemple avec $n = 10$ et $p = 3$, aurait certainement permis de détecter la plupart des erreurs d'indices...

Références

Certains exercices présentés ci-après ont été inspiré du livre suivant :

Alain Darte et Serge Vaudenay, *Algorithmique et optimisation, exercices corrigés*, Collection Sciences Sup, Dunod, 2001. ISBN : 2100056433. `http://www.dunod.com/`.

Il s'agit des exercices suivants :

Algorithmes d'ordonnancement : chapitre 2.5 ;

Voyageur de commerce : chapitre 2.6.

Remerciements

La mise en place de cette nouvelle épreuve n'aurait pas été possible sans le soutien de nombreuses personnes que nous tenons à remercier chaleureusement ici : Hervé Brunet, Vincent Danjean, Alain Darte, Louis Granboulan, Arnaud Legrand, Jean-Louis Moisy, Eric Patinaud, Loïs Taulelle.