EPREUVE ECRITE D'INFORMATIQUE

ENS : PARIS – LYON - CACHAN

Durée : 4 heures ***Coefficients*** **: PARIS MP 5 / MPI 4**
LYON MPI 4
CACHAN MP 5 / MPI 5

MEMBRES DE JURYS : H. COMON, A. DARTE

1 Présentation générale

Cette épreuve, d'une durée de 4 heures, était proposée aux élèves candidats au concours informatique des Ecoles Normales Supérieures et aux élèves candidats au concours MP des ENS et ayant choisi l'option informatique.

Le sujet portait sur l'algorithmique des mots. Il comprenait quatre parties :

- La première partie demandait de proposer sans preuve des algorithmes naïfs pour quelques problèmes d'algorithmique simples : comparaison de sous-chaînes (question 1), recherche de motifs (question 3), plus longue sous-chaîne commune (question 4), répétitions maximales (question 5). Il s'agissait d'un « filtre » destiné à éviter les candidats qui choisissent de ne traiter aucune question d'algorithmique. De façon surprenante, cette partie a pleinement joué son rôle de filtre : les données statistiques de cette partie sont similaires à celles de la totalité de l'épreuve !
- La deuxième partie avait pour objet de mettre au point un algorithme plus performant et pour cela établissait quelques résultats simples de combinatoire des mots sur lesquels s'appuyer.
- La troisième partie, plus difficile, étudiait la construction d'une structure de données classique de la combinatoire des mots, l'arbre des suffixes, tout d'abord en donnant un algorithme de construction et en l'étudiant (questions 1,2,3,4), puis en étudiant plusieurs améliorations (questions 5 à 11) permettant d'effectuer le calcul de cette structure de données en temps linéaire.
- La dernière partie étudiait des applications de l'arbre des suffixes aux problèmes algorithmiques définis en première partie. Aucun candidat n'a abordé cette partie de manière significative.

Le sujet comportait de nombreuses questions algorithmiques. Le langage de description des algorithmes n'était pas précisé : il était au choix du candidat. Certains candidats semblent d'ailleurs s'interroger sur ce qu'ils doivent faire de ce choix.

Une partie importante des candidats a décidé d'écrire les algorithmes en CAML. C'est bien entendu accepté. Cependant, ce choix, pour les questions posées dans le problème, ne facilitait pas toujours la tâche. Il était en général plus simple ici d'utiliser des structures de contrôle standard (boucles Pour et Tant que) et des affectations. Il n'était pas demandé de les écrire en utilisant la syntaxe d'un langage de programmation précis.

Les correcteurs ont noté avec plaisir que la grande majorité des candidats maîtrise bien les notions de complexité en temps des algorithmes et les notations $O()$.

2 Barème et notations

Le sujet était long, mais comme toujours le barême a tenu compte de cette difficulté ; un candidat ayant répondu correctement aux deux premières parties et à la première question de la troisième partie obtenait la note maximale.

La première partie était notée sur 8.5 points, la deuxième partie sur 10 points. La première question de la troisième partie, qui, bien qu'assez simple, demande du soin et du temps, a été notée sur 3.5 points.

Les correcteurs pouvaient en outre accorder des bonifications (jusqu'à 1.5 points par copie) dans les cas de remarques pertinentes ou d'algorithmes (ou démonstrations) particulièrement bien soignés. À l'inverse, de grosses erreurs montrant une mauvaise compréhension de notions fondamentales pouvaient entraîner un malus (jusqu'à -1 point par copie).

Tous les candidats, à l'exception d'une dizaine, ont abordé les quatre premières questions de la première partie ainsi que la deuxième partie. Seulement 25% d'entre eux ont abordé la troisième partie. À partir de la question 5 de la partie 3, seulement 4 candidats ont fourni des réponses correctes.

3 Quelques erreurs courantes et leur sanction

I.1 Dans 25% des copies on trouve un algorithme qui utilise la comparaison de sous-chaîne ! Les primitives autorisées sont rappelées en introduction et il est bien sûr nécessaire de s'y limiter.

Plusieurs copies, dans cette question et dans les suivantes, utilisent des boucles Pour, ce qui est bienvenu. Cependant, il est malvenu de modifier les indices de boucles dans le corps de la boucle. Il faut en effet savoir que les bornes d'une boucle Pour sont supposées être évaluées à l'entrée dans la boucle (même si ce n'est pas le cas dans tous les langages de programmation). On ne peut donc pas modifier ces bornes au cours de son exécution ; si l'on souhaite le faire, c'est qu'il faut utiliser une boucle Tant que.

Certains algorithmes ne terminent pas (ou donnent des résultats incorrects) de ce simple fait.

Pour terminer sur les boucles Pour, il est conseillé d'éviter les échappements en milieu de boucle ("Retourner" le résultat en milieu de boucle). C'est possible dans certains langages de programmation, mais l'effet d'un tel échappement n'est pas si clair lors de l'exécution d'un programme. Dans le cas où c'est nécessaire, utiliser une boucle Tant que qui gère explicitement l'échappement.

I.2 A peine 25% des copies ont répondu correctement à cette question. Il suffisait pourtant de remarquer que l'ensemble des mots tels que $\mathcal{I}_t$ a au moins un élément s'écrit $\mathcal{A}^* \cdot c \cdot \mathcal{A}^*$ et donc est rationnel. Quant au complémentaire, il suffisait d'invoquer la clôture par complémentaire de l'ensemble des langages rationnels.

Bon nombre de candidats ont tenté de construire un automate pour répondre à cette question. C'est possible et simple de construire un automate non-déterministe, mais beaucoup plus compliqué de construire un automate déterministe. Les candidats qui se sont lancés dans cette voie, ou bien se sont trompés dans la construction de l'automate déterministe, ou bien ont complémenté un automate non-déterministe.

Enfin, dans une copie sur 4, on obtient la réponse que le complémentaire de $\mathcal{A}^* \cdot c \cdot \mathcal{A}^*$ n'est pas rationnel. Ceci a été sanctionné : mieux vaut ne rien répondre que répondre n'importe quoi.

Comme en 2001, cette question sur la rationalité est celle qui a donné lieu au plus grand nombre de réponses aussi péremptoires que fantaisistes (un langage fini n'est jamais rationnel (?!), un langage est rationnel si et seulement s'il est fini (?!), un langage rationnel est stable par concaténation (vu plusieurs fois), ...).

I.3 C'est la question la mieux réussie par les candidats. On peut regretter que souvent ceux-ci n'utilisent pas la première question ou qu'ils omettent l'étude de complexité.

I.4 On trouve de nombreuses erreurs dans les algorithmes. Par exemple, les problèmes avec les boucles Pour évoqués ci-dessus, et, à l'inverse, des boucles Tant que dans lesquels on oublie d'incrémenter l'indice (le test d'arrêt n'est jamais satisfait). On trouve aussi des variables non initialisées, ou des algorithmes qui ne considèrent que les préfixes de s...

Les correcteurs auraient apprécié dans cette question et la suivante (surtout dans le cas de solutions compliquées, fréquentes à la surprise des correcteurs) une courte description informelle en préambule.

Enfin, les analyses de complexité sont souvent absentes ou fantaisistes. Même si ce n'était pas explicite dans l'énoncé, une courte justification de la complexité était nécessaire. Se poser cette question permettait de s'assurer qu'on avait répondu à la question et aurait permis dans bien des cas de détecter des erreurs.

I.5 Cette question est celle qui comporte le plus d'erreurs parmi les questions abordées. A nouveau, il est conseillé de donner une courte description informelle des algorithmes. Il est aussi conseillé de bien lire l'énoncé : beaucoup de copies font référence à une autre notion de maximalité que celle de l'énoncé. Il y a aussi beaucoup d'accès hors bornes du mot. Pourtant, la question 1 permettait de s'affranchir de ces problèmes en utilisant systématiquement COMPARE_SOUS_CHAÎNE. Il suffisait ici d'utiliser naïvement 3 boucles imbriquées et cette fonction ; on obtenait ainsi un algorithme en $O(n^4)$. Certains candidats, en n'utilisant pas la fonction de la première question, ont proposé des algorithmes corrects en $O(n^3)$.

II.1 à II.3 Peu de remarques sur ces questions. Plusieurs candidats ont remarqué une erreur de notation dans II.3c. Il a été tenu compte de cette erreur dans la correction des copies. En particulier, les candidats ayant sauté cette question n'ont pas été pénalisés dans la question 4.

II.4 Il s'agissait ici de comprendre comment les questions précédentes permettent de calculer L_k, d_k, g_k par récurrence sur k. Les erreurs principales concernent l'initialisation. Certains candidats oublient aussi que les résultats de la question 3 ne sont établis que pour $k \geq 3$.

La difficulté essentielle de la question résidait dans la justification de la complexité. En effet, les candidats ayant produit un algorithme correct remarquent bien que le calcul de l'indice i de la question 2 (et celui de l'indice j de la question 3c) nécessitent un parcours d'une partie du mot. Il fallait ici analyser l'algorithme plus finement en montrant que chacune des lettres n'est testée pour comparaison qu'un nombre borné de fois. Cette partie de la question a été très rarement traitée.

II.5 Ceux qui l'ont abordée ont bien répondu en général.

III.1 La plupart des réponses sont incorrectes. Beaucoup de candidats calculent les premières itérations ($i = 2, 3$) et concluent à tort que l'arbre ne contient que la racine et ses fils directs, chacun étiquetés par un suffixe de `mississipi`. Il fallait ici être assez soigneux.

III.2 A partir de ce point, les questions n'ont été abordées que par très peu de candidats. Ici, il faut poser explicitement une récurrence et étudier tous les cas. On ne peut se contenter d'une argumentation informelle.

III.4 Plusieurs candidats n'ayant pas répondu aux premières questions de cette partie, ont prétendu répondre à celle-ci. C'est en général une mauvaise idée d'essayer de "grapiller" dans ce genre d'épreuve, d'autant qu'ici il était très difficile de justifier proprement la complexité sans avoir bien compris l'algorithme. Pour l'essentiel, il fallait justifier le fait que trouve s'exécute toujours en temps linéaire, c'est à dire que le nombre de noeuds sur un chemin est borné par la taille du mot de départ. Très peu de candidats le font proprement.

Rappelons encore une fois, comme dans le rapport 2002, qu'un candidat auquel il reste du temps en fin d'épreuve ferait mieux de revenir sur des questions du début de l'énoncé qu'il n'a pas ou mal traité que de chercher des questions faciles en fin d'énoncé : ce sont parfois des leurres parce qu'il y a une difficulté qui échappe au candidat ; même dans le cas contraire le grapillage n'est jamais récompensé par les correcteurs.

Cela ne s'adresse pas aux candidats qui choisissent de passer la fin d'une partie pour traiter la suivante, mais dans sa totalité ; on prévient toutefois ces candidats que les parties sont en général ordonnées de façon à offrir une certaine progression dans la difficulté.