Composition d'Informatique (4 heures), Filière MP

Rapport de MM. Jean-Christophe Filliâtre et Jean-Pierre Tillich, correcteurs.

L'épreuve

Il s'agissait dans ce problème d'étudier la structure de corde, une structure de données pour la représentation efficace de grandes chaînes de caractères. Dans la **partie I**, il était demandé de programmer quelques fonctions élémentaires sur des mots, représentés ici par des listes d'entiers. La **partie II** introduisait la structure de corde proprement dite, à savoir un arbre binaire dont les feuilles sont des mots. Les candidats devaient alors programmer les opérations essentielles sur les cordes (longueur, concaténation, extraction d'un caractère et d'une sous-corde). Les parties **III** et **IV** étaient consacrées au problème de l'équilibrage d'une corde, c'est-à-dire à la réorganisation de l'arbre qu'elle constitue dans le but de réduire le coût moyen d'accès aux caractères. La partie **III** proposait un premier algorithme d'équilibrage et il était demandé de le programmer, ainsi que de montrer qu'il permet d'obtenir un coût moyen logarithmique en la longueur de la corde. La partie **IV** introduisait enfin un algorithme d'équilibrage optimal. Le candidat devait programmer une partie de cet algorithme.

Remarques générales

Les notes des candidats français se répartissent selon les tableaux suivants :

$0 \leqslant N < 4$	29	3,9 %
$4 \leqslant N < 8$	130	17,6 %
$8 \leqslant N < 12$	279	37,8 %
$12 \leqslant N < 16$	223	30,2 %
$16 \leqslant N \leqslant 20$	78	10,6 %
Total	739	100 %
Nombre de copies : 739		
Note moyenne : 10,97		
Écart-type : 3,85		

Parmi les candidats français, 7,58% ont choisi d'écrire leur programme en langage Pascal. La moyenne des copies Pascal est de 8,99, soit presque deux points en dessous de la moyenne générale. Même si de très bonnes copies ont été écrites en Pascal, la concision du langage Caml semble être un avantage.

Presque toutes les fonctions demandées pouvaient être écrites en moins de 10 lignes.

Les candidats doivent être conscients du fait qu'une réponse longue doit être expliquée en détail et que très souvent un programme très long contient un grand nombre d'erreurs.

Pour obtenir la note maximale, il était nécessaire de traiter le problème en entier.

Commentaire détaillé

Pour chaque question, sont indiqués entre crochets le pourcentage de candidats ayant obtenu au moins la moitié des points correspondants et le pourcentage de candidats ayant obtenu la note 0 (que la question ait été traitée ou non).

Partie I

Les questions de cette première partie étaient faciles mais demandaient de traiter avec soin les conditions d'arrêt.

Question 1 [99% - 1%]. Cette question a été correctement traitée dans l'immense majorité des cas.

Question 2 [95% - 5%]. À nouveau peu de problèmes ont été rencontrés par les candidats.

Question 3 [79% - 21%]. Un nombre signicatif de copies a été pénalisé pour ne pas avoir correctement traité la condition d'arrêt. Ce phénomène est amplifié lorsqu'une solution plus compliquée que nécessaire est choisie (par exemple en calculant d'abord un préfixe renversé puis en l'inversant). Par ailleurs, les solutions quadratiques plutôt que linéaires ont été sanctionnées.

Question 4 [86% - 14%]. Les remarques de la question précédente s'appliquent également ici. Les candidats ayant fait le choix d'une programmation itérative ont toutefois mieux traité cette question que la précédente, cette fonction s'écrivant naturellement à l'aide d'une boucle.

Partie II

Dans cette partie, beaucoup de réponses inexactes sont liées soit à la non-utilisation de l'invariant sur les cordes, soit à la création de cordes ne respectant pas cet invariant.

Question 5 [83% - 16%]. Cette fonction peut être réalisée avec une complexité constante. Des solutions de complexité linéaire utilisant la fonction `longueurMot` ont été pénalisées. Il convenait également de ne pas oublier le cas de la corde vide.

Question 6 [43% - 9%]. Cette fonction nécessitait de considérer avec soin le cas du mot vide, afin que l'invariant sur les cordes soit respecté. Comme dans le cas précédent, cette fonction pouvait être réalisée avec une complexité constante ; des solutions linéaires, consistant par exemple à construire un arbre binaire en forme de peigne avec les lettres du mot, ont été sanctionnées.

Question 7 [73% - 14%]. Ici aussi cette fonction pouvait être réalisée avec une complexité constante. La plupart des erreurs sont dues au fait que la corde créée ne respecte pas l'invariant.

Question 8 [91% - 9%]. Cette question a été bien traitée dans l'ensemble, mis à part quelques imprécisions concernant les tests ou les paramètres de l'appel récursif.

Question 9 [55% - 41%]. Cette question est apparue plus diffcile que les précédentes, la première diffculté étant de bien considérer les trois cas de figure (la sous-corde est entièrement contenue dans le sous-arbre gauche, dans le sous-arbre droit, ou chevauche les deux) et la seconde de traiter avec soin les nombreux paramètres des appels récursifs.

Partie III

Question 10 [68% - 32%]. Cette question avait pour but de familiariser le candidat avec l'algorithme proposé, ce qui pouvait se révéler précieux, notamment pour la question 14. Malheureusement, une partie non négligeable des candidats ne l'a pas traitée.

Question 11 [95% - 4%]. Cette question a été bien traitée dans l'ensemble. Il convenait évidemment d'écrire une solution de complexité linéaire. Des solutions de complexité exponentielle utilisant une écriture récursive de la fonction de Fibonacci ont été pénalisées.

Question 12 [63% - 34%]. Ce qui a pénalisé les candidats dans cette question a été d'une part l'absence de traitement du dépassement de la capacité de la file, et d'autre part l'oubli de l'affectation à `Vide` d'une case de la file lorsque nécessaire.

Question 13 [33% - 58%]. Cette question s'est révélée l'une des plus difficiles. La difficulté principale de la question était de comprendre que, lorsqu'il y a effectivement concaténation de deux cordes non vides, alors le résultat sera nécessairement placé plus loin dans la file, en vertu des propriétés des nombres de Fibonacci. D'autre part, certains candidats ont modifié l'algorithme proposé de manière à forcer l'invariant, à l'aide d'un test supplémentaire sur la hauteur (à la question 12), et n'ont pas expliqué pourquoi l'insertion était effectivement réalisée dans tous les cas de figure.

Question 14 [47% - 42%]. Cette question s'est également révélée difficile, car il y avait plusieurs possibilités de ne pas respecter l'algorithme proposé : soit en construisant une corde qui ne représentait pas le bon mot (par inversion des sous-cordes), soit en construisant une corde n'ayant pas la bonne hauteur (en concaténant les éléments de la file dans l'ordre inverse).

Question 15 [6% - 52%]. De très nombreux candidats ont incorrectement répondu à cette question en raison d'une incompréhension de l'algorithme à la question 14. L'autre difficulté était de correctement rédiger la récurrence établissant la première inégalité demandée. La seconde inégalité a été établie par un grand nombre de candidats.

Partie IV

Cette partie proposait un algorithme complexe, dont seule une partie devait être réalisée par les candidats. Certains candidats ont apparemment lu le sujet trop rapidement et proposé une solution pour la fonction phase 1, ce qui n'était pas demandé.

Question 16 [62% - 23%]. Une partie non négligeable des candidats a choisi d'utiliser une fonction auxiliaire renvoyant la liste des feuilles d'une corde. Cette solution était acceptée si elle était réalisée avec une complexité linéaire, ce qui était rarement le cas (la réalisation la plus courante consistant à utiliser la concaténation de listes, résultant en une complexité quadratique). Il existait cependant une solution directe de complexité linéaire.

Question 17 [61% - 37%]. Certaines solutions qui ont été pénalisées consistaient à utiliser une fonction de complexité non linéaire pour calculer la profondeur d'une feuille dans une corde. D'autres solutions incorrectes utilisaient à tort la fonction `initialiserQ` sur la corde c_1, sans prendre garde au fait que l'information contenue dans le tableau q concernant la corde c était alors perdue.

Question 18 [12% - 84%]. C'était peut-être la question la plus difficile du sujet, mais il existait néanmoins de nombreuses solutions : une solution courte et élégante de complexité

linéaire et utilisant la récursivité ; une autre consistant à rechercher les deux premières feuilles de profondeur maximale et à les concaténer, avant de réitérer le processus ; une solution similaire mais consistant à rechercher les deux premières feuilles consécutives de même profondeur ; ou encore une solution consistant à insérer successivement chaque feuille le plus à gauche dans un arbre initialement vide. Cette dernière solution a été tentée par de nombreux candidats, mais s'est révélée difficile à mettre en œuvre et n'a été traitée correctement qu'une seule fois.

Question 19 [64% - 35%]. Cette question était relativement simple. La seule difficulté consistait à ne pas oublier le cas particulier de la corde vide.