ÉCOLE POLYTECHNIQUE

ÉCOLE SUPÉRIEURE DE PHYSIQUE ET CHIMIE INDUSTRIELLES

CONCOURS 2002 **FILIÈRE MP - OPTION SCIENCES INDUSTRIELLES**
FILIÈRE PC

ÉPREUVE FACULTATIVE D'INFORMATIQUE

(Durée : 2 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.
On attachera une grande importance à la clarté, à la précision, à la concision de la rédaction.

⋆⋆⋆

On cherche à calculer le gain maximum possible à la Bourse sur une action pendant une période de n jours, en ne faisant qu'une opération d'achat et de vente d'une seule action. On suppose que les cours quotidiens de cette action sont enregistrés dans un tableau d'entiers naturels ($a_i \in \mathbf{N}$) de n éléments ($0 \leqslant i < n$).

On définit l'*amplitude* de la variation du cours comme la valeur absolue du maximum de la variation de ce cours pendant la période observée, c'est-à-dire la quantité suivante :

$$\textit{amplitude} = \max_{0 \leqslant i \leqslant j < n} |a_j - a_i| = \max_{0 \leqslant i < n} a_i - \min_{0 \leqslant i < n} a_i$$

Le temps d'exécution $T(f)$ d'une fonction f de la variable a est le nombre d'opérations élémentaires (addition, soustraction, multiplication, division, affectation) nécessaire au calcul de $f(a)$. Lorsque ce temps d'exécution dépend d'un paramètre n, il sera noté $T_n(f)$. On dit que la fonction f s'exécute :

- en temps linéaire par rapport au paramètre n, s'il existe $K > 0$ tel que pour tout n, $T_n(f) \leqslant Kn$;
- en temps quadratique par rapport au paramètre n, s'il existe $K > 0$ tel que pour tout n, $T_n(f) \leqslant Kn^2$.

Question 1 Écrire une fonction amplitude(a) qui retourne comme résultat l'amplitude de la variation du cours représenté par le tableau a. Donner un ordre de grandeur du temps d'exécution de cette fonction en fonction de n.

Le *gain maximum* est le gain maximum possible sur la période observée, c'est-à-dire la quantité suivante :

$$\textit{gain} = \max_{0 \leqslant i \leqslant j < n} (a_j - a_i)$$

Question 2 Donner un exemple où l'amplitude est différente du gain maximum. Que représente l'amplitude en terme de gain ou de perte ?

Question 3 En suivant textuellement la définition du gain, écrire une fonction gain(a) qui retourne, en temps quadratique (par rapport à n), le gain maximal possible sur le cours représenté par le tableau a.

Question 4 Modifier la fonction précédente pour aussi imprimer les deux dates i et j d'achat et de vente de l'action permettant d'obtenir le gain maximum sur le tableau a (avec $j - i$ minimum).

Pour tout i ($0 \leqslant i < n$) définissons le gain courant maximum $gainCourant_i$ comme le gain maximum possible obtenu en vendant son action au temps i, c'est-à-dire :

$$gainCourant_i = \max_{0 \leqslant k \leqslant i} (a_i - a_k)$$

Question 5 En calculant progressivement le gain courant maximum, écrire une fonction gain1(a) qui retourne, en temps linéaire (par rapport à n), le gain maximum possible sur le cours représenté par le tableau a.

Question 6 Modifier la fonction précédente pour aussi imprimer les deux dates i et j d'achat et de vente de l'action permettant d'obtenir le gain maximum sur le tableau a (avec $j - i$ minimum).

On considère maintenant la possibilité de faire séquentiellement deux transactions pendant la période observée, c'est-à-dire de considérer deux dates d'achat i et i', et deux dates de vente j et j' telles que $0 \leqslant i \leqslant j \leqslant i' \leqslant j' < n$.

Question 7 Écrire la fonction gain2(a) qui retourne, en temps quadratique (par rapport à n), le gain maximum possible en faisant deux transactions sur le cours de l'action représenté par le tableau a.

Question 8 Modifier la fonction précédente pour aussi imprimer les quatres dates i, j, et i', j' d'achats et de ventes de l'action permettant d'obtenir le gain maximum sur le tableau a (avec $j - i$ et $j' - i'$ minimum).

* *
*

Épreuve facultative d'Informatique, filières MP et PC

Rapport de M. Marc Pouzet, correcteur.

De l'épreuve

C'était la première édition de cette épreuve spécifique à l'École Polytechnique. L'épreuve était facultative et seules les copies des candidats admissibles ont été corrigées.

Résultats

Cette épreuve était commune aux filières MP et PC. La moyenne des candidats de la filière MP est 12.9, avec un écart type de 4.7. La moyenne des candidats de la filière PC est 12.4 avec un écart type de 4.9. La répartition par filière est la suivante :

	$0 \leq N < 4$	$4 \leq N < 8$	$8 \leq N < 12$	$12 \leq N < 16$	$16 \leq N < 20$
PC	8%	10%	14%	47%	21%
MP	8%	5%	18%	45%	24%

L'écart type important vient du grand nombre de très bonnes copies se détachant largement de la moyenne (au dessus de 16/20). Ceci s'explique certainement par la présence de candidats ayant une grande pratique de la programmation.

Le langage de programmation Maple a été utilisé par la grande majorité des candidats. Les autres langages ont été principalement Pascal et Java. Quelques candidats ont choisi Caml, C et C++. Trois copies ont été composées en utilisant un pseudo-langage « algorithmique » ou un pseudo-langage « basic » de calculatrice. La répartition est la suivante :

	Maple	Pascal	Java	Autres
PC	95 %	5 %	0 %	1 %
MP	86 %	2 %	10 %	2 %

Comme il s'agissait de la première édition de cette épreuve, le jury a décidé de corriger malgré tout les copies écrites dans un langage ne faisant

pas partie de la liste des langages autorisés. Une telle tolérance pourrait ne pas être reconduite pour les éditions ultérieures du concours.

Les coefficients des questions ont été ajustés par filière. Comme l'indique le tableau ci-dessous, les coefficients étaient plus élevés en PC qu'en MP.

Question	1	2	3	4	5	6	7	8
MP	4	2	4	3	5	3	5	3
PC	6	2	6	3	6	3	5	3

Evaluation

Cette épreuve consistait à écrire un certain nombre de fonctions de parcours d'un tableau de valeurs entières. Les structures de données et de contrôle nécessaires se résumaient aux tableaux, boucles `for`, conditionnelles, séquence, définition et utilisation de fonctions. L'utilisation de la récursivité n'était pas indispensable dans cette épreuve.

Un certain nombre de candidats ont choisi d'utiliser des opérations spécifiques à Maple (listes, opérateur `seq`). L'utilisation de ces primitives nécessite une bonne connaissance de Maple et elles ont été utilisées le plus souvent de manière approximative (problèmes de type, de bornes) ce qui a fait perdre des points à ces candidats. Dans ce type d'épreuve, il est conseillé de s'en tenir à des structures de données et de contrôle élémentaires (boucles `for`, tableaux). De plus, elles se prêtent mieux à un calcul de complexité.

Toutes les questions (exceptée la question 2) demandaient d'écrire du code. Le code de chaque question étant court, la correction commence par la lecture du code. S'il est juste, je ne tiens pas trop compte des commentaires. Sinon, je les lis attentivement pour comprendre l'idée du candidat. Toutefois, le candidat a déjà perdu une bonne partie de ses points. La situation peut être sauvée partiellement lorsque le candidat a eu l'idée de formuler un invariant de boucle sous forme de suites récurrentes.

Lors de la lecture des programmes, les petites erreurs de syntaxe (oubli d'un `end`, d'une marque de fin de boucle) n'ont pas été prises en compte dès lors qu'il n'y a pas d'ambiguïté possible et qu'il semble qu'elles seraient immédiatement corrigées lors d'une programmation sur machine. En revanche, les erreurs de type et une absence d'initialisation des variables ont été systématiquement sanctionnées.

Question 1. La plupart des candidats ont répondu correctement en donnant un algorithme de complexité linéaire (calculant la valeur minimale et maximale en un seul parcours ou à partir de deux fonctions `min` et `max` de calcul de la valeur minimale et maximale d'un tableau).

S'agissant d'une question facile, les algorithmes compliqués (e.g. quadratiques) et l'utilisation de variables non initialisées ont été sanctionnés.

Question 2. Cette question se composait de deux petites questions de valeurs identiques. La plupart des candidats y ont répondu correctement. Ont été considérées comme fausses, les réponses qui traduisaient une erreur d'interprétation (donner un exemple de gain négatif, cours contenant des valeurs négatives).

Question 3. Cette question consistait à écrire deux boucles imbriquées en suivant la définition et a été traitée correctement par la plupart des candidats.

Cette question étant facile, j'ai pénalisé fortement les erreurs dans les bornes des indices (faisant varier `i` et `j` de 1 à n) donnant lieu à des programmes incorrects. Une absence d'initialisation ou une solution compliquée (allouant un tableau intermédiaire des gains partiels suivi d'un calcul de son maximum) ont également été pénalisées.

Il fallait justifier la complexité quadratique de l'algorithme. Dans le cas de l'écriture de deux boucles imbriquées (où i et j vérifient $1 \leq i \leq n$ et $i \leq j \leq n$) il suffisait de montrer que $T_n(\texttt{gain}) = \Sigma_{i=1}^{n}\Sigma_{j=i}^{n}1 = n.(n+1)/2$.

Question 4. La réponse à cette question devait être donnée en reprenant la solution précédente et en y ajoutant des instructions permettant d'afficher les indices i et j. La minimisation de la différence $(j-i)$ rendait cette question plus difficile que la précédente.

Une solution parfaite ne minimisant pas $j - i$ recueillait la moitié des points seulement.

Certains candidats n'ont pas lu l'énoncé avec une attention suffisante. Ainsi, il était demandé d'*imprimer* les dates i et j et non pas de retourner un triplet de valeurs (v, i, j) où v représente la valeur du gain. Il est dommage de perdre ainsi la totalité des points de cette question.

Plusieurs candidats ont choisi de répondre à cette question en annotant

(avec de la couleur ou un signe particulier) le programme donné dans la question précédente et en indiquant les insertions à faire dans le programme. Ce type de réponse, souvent ambiguë et dont la lecture est difficile est fortement déconseillé. De plus, il ne peut se justifier ici dans la mesure où le programme de la question précédente était très court (moins de cinq lignes).

Question 5. Cette question constituait la première difficulté de l'épreuve et a *bloqué* de nombreux candidats. On constate que les candidats ayant réussi cette question ont obtenu la moyenne à l'épreuve. Elle a été globalement beaucoup mieux réussie par les candidats de la filière MP que par les candidats de la filière PC.

Les solutions quadratiques ont été très fortement pénalisées, ne recueillant pas ou peu de points. Ainsi, plusieurs candidats ont d'abord écrit une fonction de calcul du gain courant (de complexité linéaire) puis écrit une fonction principale de parcours des éléments du tableau et faisant appel à la fonction précédente.

L'objectif de cette question était d'écrire une seule boucle calculant progressivement le gain courant maximum. Une bonne manière d'arriver à la solution consistait à décrire le gain courant (`gc`), le maximum du gain courant (`gm`) et la valeur minimum (`min`) du tableau a sous forme de suites récurrentes.

$$\begin{array}{lcllcl} \texttt{gc}_n & = & \texttt{a}_n - \texttt{min}_n & \texttt{gc}_0 & = & 0 \\ \texttt{gm}_n & = & max(\texttt{gc}_n, \texttt{gm}_{n-1}) & \texttt{gm}_0 & = & 0 \\ \texttt{min}_n & = & min(\texttt{a}_n, \texttt{min}_{n-1}) & \texttt{min}_0 & = & \texttt{a}_0 \end{array}$$

Peu de candidats ont eu l'idée d'écrire le gain maximal sous forme d'une suite récurrente avant de passer à la programmation. C'est dommage car la plupart des candidats ayant suivi cette voie ont proposé un algorithme correct et ont recueilli tous les points.

Cette question s'étant révélée difficile, j'ai peu pénalisé les solutions linéaires compliquées (faisant, par exemple, intervenir un tableau intermédiaire pour stocker la suite des gains courant). De même, les petites erreurs de syntaxe ont été peu pénalisées.

Question 6. Cette question était difficile également. Les solutions affichant des valeurs correctes pour i et j mais ne minimisant pas $j - i$ ont reçu les deux-tiers des points.

Là encore, certains candidats ont écrit un programme n'imprimant pas i et j mais retournant un triplet (v, i, j). Cette erreur de lecture de l'énoncé a été sanctionnée également.

Question 7. Cette question s'est révélée plus facile que les deux questions précédentes. Nous avons privilégié ici les solutions modulaires, faisant soit un appel à la fonction `gain1` définie précédemment, soit en redéfinissant une fonction `gain1` calculant le gain maximum entre deux indices donnés en paramètre.

Les solutions non quadratiques (en $O(n^3)$ ou en $O(n^4)$) ont reçu le tiers des points.

Il était possible de répondre à cette question sans avoir réussi la question précédente à condition d'utiliser correctement l'appel à `gain1`.

Question 8. Là encore, il fallait privilégier une solution modulaire utilisant si possible des appels à la fonction définie dans la question 6.