ÉCOLE POLYTECHNIQUE

ÉCOLE SUPÉRIEURE DE PHYSIQUE ET CHIMIE INDUSTRIELLES

CONCOURS 2003 **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.

⋆⋆⋆

Avertissement On attachera une grande importance à la clarté, à la précision, à la concision de la rédaction.

L'enclos du robot

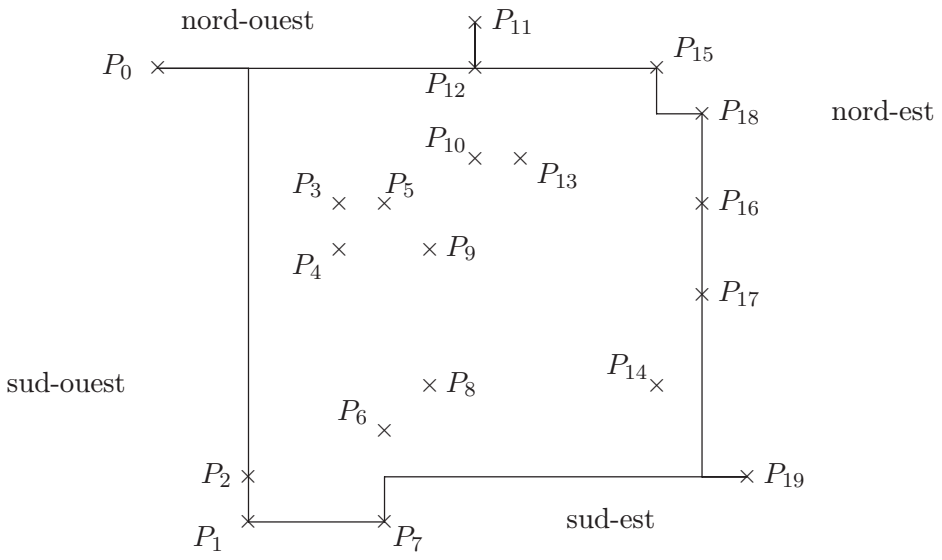
Frontière Sud-Ouest

Un robot rend visite à n points P_i $(0 \leq i < n)$ de coordonnées (a_i, b_i) $(a_i \in \mathbf{N}, b_i \in \mathbf{N}, n \geq 0)$. Le robot ne fait que des déplacements parallèles à l'axe des abscisses ou à l'axe des ordonnées. Ses déplacements sont toujours de longueur minimale entre deux points. Toutefois le robot n'est pas très fiable. On veut délimiter l'espace minimal nécessaire pour ses déplacements en tendant une ficelle autour du périmètre strictement nécessaire pour les déplacements du robot.

L'*intérieur Manhattan* des n points est l'ensemble des points de coordonnées (x, y) vérifiant les quatre conditions suivantes :

$\exists i.\ 0 \leq i < n$ et $a_i \leq x$ et $b_i \leq y$ (sud-ouest)
$\exists i.\ 0 \leq i < n$ et $a_i \geq x$ et $b_i \leq y$ (sud-est)
$\exists i.\ 0 \leq i < n$ et $a_i \geq x$ et $b_i \geq y$ (nord-est)
$\exists i.\ 0 \leq i < n$ et $a_i \leq x$ et $b_i \geq y$ (nord-ouest)

L'enveloppe Manhattan est un polygone dont l'intérieur est l'intérieur Manhattan de ces n points. Par exemple sur les 20 points de la figure suivante, c'est le polygone suivant :



On se propose de calculer les points définissant l'enveloppe Manhattan dans un premier temps, puis de tracer cette enveloppe dans un deuxième temps.

Question 1 Écrire la fonction *sudouest* qui retourne le résultat vrai si et seulement si le point de coordonnées (x_1, y_1) est au sudouest du point de coordonnée (x_2, y_2), c'est-à-dire si $x_1 \leq x_2$ et $y_1 \leq y_2$. Écrire de même les fonctions *nordouest*, *sudest*, *nordest*. (Dans les langages de programmation où les valeurs booléennes n'existent pas, on rendra l'entier 0 pour la valeur *faux* et 1 pour la valeur *vrai*)

Nous décomposons le calcul de l'enveloppe en quatre fonctions : la première calcule les points définissant la partie sud-ouest de l'enveloppe, la deuxième calcule les points définissant la partie nord-ouest, la troisième et quatrième font de même sur les parties sud-est et nord-est.

On suppose les coordonnées des n points P_i rangés dans deux tableaux a et b d'entiers contenant l'abscisse a_i et l'ordonnée b_i du point P_i pour tout i $(0 \leq i < n)$. En outre, on suppose les points rangés par ordre d'abscisses croissantes, c'est-à-dire $a_i \leq a_j$ pour $0 \leq i < j < n$.

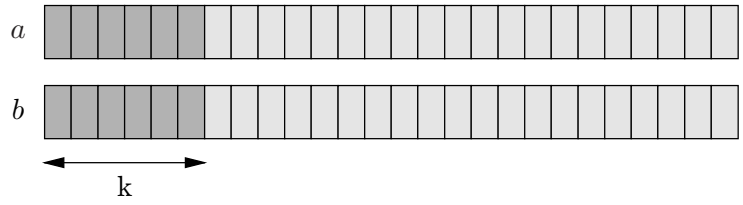
Question 2 Écrire une fonction *echange* prenant comme arguments les tableaux a et b, les indices i et j et qui échange, dans chacun des tableaux a et b, les valeurs contenues aux indices i et j.

Les points définissant la partie sud-ouest de l'enveloppe sont les points P_i tels que : $a_j \leq a_i$ et $b_j \leq b_i$ implique $a_j = a_i$ et $b_j = b_i$ pour tout j $(0 \leq j < n)$.

Question 3 Dans l'exemple précédent, donner parmi les 20 points P_i, les points définissant la partie sud-ouest de l'enveloppe.

Question 4 Écrire une fonction *frontiereSO* prenant en argument les tableaux a et b et retournant le nombre k de points définissant la partie sud-ouest de l'enveloppe Manhattan des n

points de coordonnées a_i et b_i. On modifiera les tableaux a et b pour qu'ils contiennent dans leur k premières places les coordonnées des points définissant la partie sud-ouest de l'enveloppe, rangées en ordre d'abscisses croissantes.



Après avoir exécuté la fonction précédente, on suppose qu'une fonction range dans deux tableaux *aSO*, *bSO* les coordonnées des points précédemment trouvés, qui définissent la partie sud-ouest. Une variable globale *nSO* a pour valeur le nombre de ces points.

Tracé de l'enveloppe

Les points définissant la partie nord-ouest de l'enveloppe sont les points P_i tels que $a_j \leq a_i$ et $b_j \geq b_i$ implique $a_j = a_i$ et $b_j = b_i$ pour tout j $(0 \leq j < n)$.

Question 5 Donner les points définissant la partie nord-ouest de l'enveloppe sur l'exemple. Modifier la fonction précédente pour obtenir la fonction *frontiereNO* correspondante pour la partie nord-ouest de l'enveloppe.

Question 6 Écrire également les fonctions *frontiereSE* et *frontiereNE* correspondant aux parties sud-est et nord-est.

On suppose à présent que les coordonnées des points précédemment trouvés, qui définissent les partie sud-ouest, nord-ouest, sud-est et nord-est, sont rangées respectivement dans des tableaux *aSO*, *bSO*, *aNO*, *bNO*, *aSE*, *bSE*, *aNE*, *bNE*, et toujours classées par ordre d'abscisses croissantes. Soient *nSO*, *nNO*, *nSE*, *nNE* les nombres de ces points. On suppose également qu'il existe deux fonctions graphiques *moveTo* et *lineTo* telles que :

- *moveTo(x,y)* déplace le point courant au point (x, y),
- *lineTo(x,y)* trace un segment du point courant jusqu'au point (x, y). Après le tracé, le point courant devient le point de coordonnées (x, y).

Question 7 Écrire une fonction qui dessine l'enveloppe Manhattan. (Cette fonction utilise les 12 variables globales *aSO*, *bSO*, *aNO*, *bNO*, *aSE*, *bSE*, *aNE*, *bNE*, *nSO*, *nNO*, *nSE*, *nNE*.)

Question 8 L'enveloppe peut-elle produire un polygone croisé ? Si oui, donner une piste pour résoudre ou éviter ce problème. $O(1)$ opérations.

* *
*

Épreuve facultative d'Informatique, filières MP et PC

Rapport de M. Marc Pouzet, correcteur.

C'était la deuxième 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 12.6, avec un écart-type de 4.25. La moyenne des candidats de la filière PC est 11.9 avec un écart-type de 4.4. 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	3%	15%	36%	24%	20%
MP	3%	9%	33%	30%	23%

L'écart-type important vient du nombre important 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 très grande majorité des candidats. Les autres langages ont été principalement Pascal et Java. Quelques candidats ont choisi C et C++.

Les coefficients des questions ont été ajustés par filière comme l'indique le tableau ci-dessous.

Question	1	2	3	4	5	6	7	8
MP	4	2	2	6	2	2	6	5
PC	4.5	4.5	2	6	2	2	6	4

Évaluation

Cette épreuve consistait à écrire un certain nombre de fonctions de parcours de tableaux 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.

Plus encore cette année, 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, je conseillerais de s'en tenir à des structures de données et de contrôle élémentaires (boucles `for`, tableaux).

Toutes les questions (exceptée la **question 3**) 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.

Cette année, j'ai trouvé plusieurs copies où les programmes étaient rédigés sans aucune structuration (à la manière d'un texte). Ce type d'écriture est totalement illisible et doit être absolument évité. Il a été fortement sanctionné.

Question 1 La plupart des candidats ont répondu correctement à cette question. S'agissant d'une question facile, j'ai privilégié les solutions simples suivant fidèlement les définitions données.

Une solution compliquée (e.g, faisant intervenir des variables intermédiaires) ne recevait pas tous les points. Certains candidats ont confondu l'affichage (au moyen de la primitive `display` de Maple) et la valeur retournée par une fonction. Ce type d'erreur ainsi que les erreurs de type (e.g, fonction retournant une chaine de caractères) ont été plus fortement sanctionnées.

Question 2 Il s'agissait d'une question classique et la plupart des candidats y ont répondu correctement. La solution s'écrivait en trois lignes et devait modifier physiquement les tableaux a et b. Les solutions incomplètes ou inutilement compliquées ne recevaient pas tous les points.

Ainsi, il me semble qu'il faut éviter les astuces, mêmes classiques, telles que l'échange de valeurs en utilisant seulement des opérations arithmétiques dont l'intérêt algorithmique est limité.

Ont été considérées comme fausses, les fonctions qui retournaient deux nouveaux tableaux dans lesquels les valeurs d'indice i et j ont été échangées.

Question 3 Cette question a été traitée par l'essentiel des candidats et il suffisait de donner les points P_0 et P_1 pour recueillir tous les points. Les réponses incomplètes (e.g, P_0 seulement) ou désignant trop de points traduisaient une mauvaise compréhension de l'énoncé et ont été sanctionnées.

Question 4 L'énoncé ne spécifiant pas de complexité attendue, cette question pouvait être traitée en donnant un algorithme quadratique ou un algorithme linéaire. Les quelques copies proposant un algorithme linéaire – beaucoup plus difficile à trouver – ont été récompensées.

La solution quadratique consistait à écrire deux boucles imbriquées d'indice i et j pour déterminer si $P_i = (a_i, b_i)$ appartenait à la frontière sud-ouest. La solution juste la plus souvent proposée consistait à calculer pour chaque point $P_i = (a_i, b_i)$ le nombre cpt de points $P_j = (a_j, b_j)$ au sud-ouest de P_i correspondant au calcul :

$$cpt \quad = \quad \sum_{j=0}^{n-1} (si \texttt{ sud\_ouest } (p_j, p_i) \; alors \; 1 \; sinon \; 0)$$

Ainsi, P_i appartient à la frontière sud-ouest de l'enveloppe Manhattan lorsque $cpt = 1$. Le programme découle naturellement de la définition donnée ci-dessus en faisant varier i ($0 \leq i < n$) et en modifiant progressivement les tableaux a et b.

Peu de candidats ont l'idée d'écrire des invariants de boucle ou de définir le résultat des programmes sous forme de suites récurrentes avant de se lancer dans 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.

Les raisons principales pour ne pas avoir eu tous les points ont été un oubli d'initialisation du compteur, un échange des paramètres p_j et p_i (en écrivant `sud_ouest` (p_i, p_j), par exemple) ou une erreur simple de borne.

Une réponse partielle retournant seulement k sans modifier les tableaux a et b recevait la moitié des points seulement.

Question 5 Lorsque le candidat avait choisi une solution quadratique, la réponse à cette question se déduisait de la précédente en remplaçant l'appel à la fonction `sudouest` par un appel à la fonction `nordouest`.

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 s'est souvent révélé ambigüe et difficile à lire. Dans le cas de programmes courts, il est préférable de l'éviter. Je recommanderais plutôt d'avoir une écriture plus modulaire des programmes permettant de mettre en commun des portions de programme identique.

Question 6 Comme précédemment, la réponse à cette question se déduisait de la **question 4**. Ceci justifiait donc l'intérêt d'une écriture modulaire (faisant référence à la fonction `sudouest`) dans la **question 4**.

La qualité de la réponse à la **question 4** a donc conditionné fortement la note finale. On constate que les candidats ayant réussi cette question ont obtenu la moyenne à l'épreuve. Il était donc important d'y consacrer du temps.

Question 7 Cette question ne comprenait pas de difficulté particulière. Elle n'a cependant pas été bien traitée dans l'ensemble (moyenne de 1.5/5).

La réponse à cette question consistait à écrire quatre boucles permettant de dessiner respectivement les parties sud-ouest, sud-est, nord-est et nord-ouest de l'enveloppe Manhattan et les liaisons entre elles.

Les raisons principales pour ne pas obtenir tous les points ont souvent été une erreur dans l'affichage d'une frontière (e.g, affichage convexe plutot que concave), un oubli ou une erreur dans l'affichage d'une liaison.

Il fallait privilégier ici une solution simple affichant les frontières dans le sens trigonométrique (ou le sens inverse), en affichant les liaisons lors de ce parcours.

Question 8 Cette question ne demandait pas d'écrire du code. Elle a été très peu traitée dans l'ensemble.

De nombreux candidats se sont lancés dans des explications compliquées pour justifier que leur algorithme produisait (ou ne produisait pas) de polygone croisé sans penser à donner un exemple. Une réponse recevant tous les points se résumait à l'énoncé d'un exemple de quatre points dont le tracé produisait un polygone croisé. Compte tenu du faible nombre de réponse à cette question, le manque de justification a été peu sanctionné.