

INFORMATIQUE

DURÉE : 5 heures

Les deux problèmes sont obligatoires

PROBLÈME 1

Le but du problème est de trouver un algorithme décidant si un point du plan est à l'intérieur d'un polygone simple.

Définitions

Nous nous plaçons dans un plan affine euclidien orienté. Ce plan est muni d'un repère orthonormé.

Soit $P = (p_1, \dots, p_n)$ une suite de n ($n \geq 3$) points distincts (appelés *sommets*) telle que trois quelconques de ces points ne sont pas alignés. Le *contour polygonal fermé* C d'une suite P est la réunion des segments de droite fermés joignant deux sommets successifs de S : $C = [p_1, p_2] \cup \dots \cup [p_n, p_1]$. Le contour polygonal fermé C est *simple* si deux segments non consécutifs ont une intersection vide : $[p_i, p_{i+1}] \cap [p_j, p_{j+1}] = \emptyset$ pour $1 \leq i, j \leq n$ et $i \neq j-1, j, j+1$, (les indices i et j étant pris modulo n). Dans ce cas, le contour C (C est la *frontière*) sépare le plan en deux régions. La région bornée par C est appelée *l'intérieur de P* , la région non bornée est appelée *l'extérieur de P* . On appelle *polygone simple*, noté encore $P = (p_1, \dots, p_n)$, la réunion de la frontière C et de l'intérieur de P . Le plan étant orienté, on suppose que pour un observateur qui parcourt le contour de P dans le sens croissant des indices, l'intérieur de P est à sa gauche.

Le *temps d'exécution* $t(A)$ d'un algorithme A est mesuré par son nombre d'opérations arithmétiques élémentaires telles que soustractions, additions ou multiplications (on fait abstraction de toutes les autres opérations comme les comparaisons ou les structures de contrôle). On dit que A *s'exécute en temps* $f(n)$, noté $O(f(n))$, si son nombre d'opérations élémentaires est majoré par $kf(n)$ où k est une constante, f une fonction et n la taille des données (ici n est le nombre de sommets d'un polygone).

1. Soit P un polygone simple à n sommets.
 - 1.1. Montrer qu'on peut déterminer si un point m est l'intérieur de P .
 Indication : compter le nombre de points d'intersection d'une demi-droite d'origine m avec le contour de P en détaillant le décompte des intersections.
 - 1.2. Écrire un algorithme, déterminant si m est à l'intérieur de P , qui s'exécute en un temps proportionnel au nombre n de côtés de P (on dira : en temps $O(n)$).
2. Un polygone simple $P = (p_1, \dots, p_n)$ est *convexe* si tous ses angles $(p_i p_{i+1}, p_{i+1} p_{i+2})$, pour $i = 1, \dots, n$ ($i+1$ et $i+2$ sont modulo n) sont de mesure strictement inférieure à π .
 - 2.1. Soient A, B et C trois points du plan. Donner un algorithme s'exécutant en temps constant qui décide si A est à gauche de la droite (BC) orientée de B vers C .
 - 2.2. Donner un algorithme qui, prenant en entrée un point m et un polygone simple convexe P à n sommets, décide si m est à l'intérieur de P en temps $O(n)$.
 - 2.3. Optimiser l'algorithme de la question 2.2. pour obtenir un temps d'exécution en $O(\log n)$.

Tournez la page S.V.P.

PROBLÈME 2

Le but du problème est d'étudier un algorithme de discrétisation d'une droite pour la représenter sur un écran graphique.

Définitions

Un *alphabet* est un ensemble fini de symboles, appelés *lettres*, tous distincts. On appelle *mot*, sur un alphabet A , une suite finie d'éléments de A . La *concaténation* de deux mots sur un alphabet A est l'opération binaire qui consiste à mettre bout à bout ces deux mots : si $f = f_1 f_2 \dots f_m$ et $g = g_1 g_2 \dots g_n$ (avec $f_i \in A$ pour $1 \leq i \leq m$ et $g_j \in A$ pour $1 \leq j \leq n$), alors la concaténation de f et g , notée fg , est le mot $fg = f_1 f_2 \dots f_m g_1 g_2 \dots g_n$.

La suite vide, appelée *mot vide*, est l'élément neutre pour la concaténation et est notée ϵ .

L'ensemble des mots sur A est noté A^* . Un *langage* sur A est une partie de A^* . La *longueur* d'un mot $f \in A^*$ est le nombre de lettres le composant et est notée $|f|$. La longueur du mot vide est donc 0. Le nombre de fois où une lettre $a \in A$ apparaît dans un mot $f \in A^*$ est noté $|f|_a$.

Un mot $f \in A^*$ qui n'est pas le mot vide est un *facteur* d'un mot $a \in A^*$ s'il existe deux mots $f_1, f_2 \in A^*$ tels que $w = f_1 f f_2$. Si $f_1 = \epsilon$ (resp. $f_2 = \epsilon$), alors f est un *facteur gauche* (resp. *facteur droit*) de w .

Un *morphisme* h de A^* dans B^* est une application h telle que $h(uv) = h(u)h(v)$ pour tous les mots u et v et $h(\epsilon) = \epsilon$. Le morphisme h est, en fait, complètement défini par sa restriction de A vers B^* car pour tout mot $w = w_1 w_2 \dots w_n$ de A^* , on a : $h(w_1 w_2 \dots w_n) = h(w_1)h(w_2) \dots h(w_n)$ avec $w_i \in A$ pour $i = 1, \dots, n$.

Considérons, pour tout entier $n \geq 2$, l'ensemble $\Delta(n)$ constitué par les droites d'équation $y = \delta(x)$ où

$\delta(x) = \frac{p}{q}x + \frac{r}{q}$, les entiers p, q et r vérifiant :

(H1) $0 < p < q \leq n$;

(H2) $0 \leq r < q \leq n$;

(H3) $(p, q) = 1$ (p et q sont premiers entre eux).

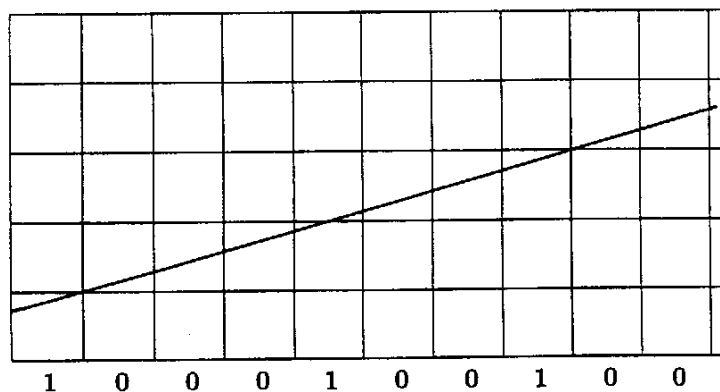
Soit $\Delta = \bigcup_{n \geq 2} \Delta(n)$. On identifiera une droite d'équation $y = \delta(x)$ avec la fonction δ : on notera donc $\delta \in \Delta(n)$.

On considère l'application $\Psi : \Delta \rightarrow \{0, 1\}^*$ associant à une droite $\delta \in \Delta(n)$ le mot $\Psi(\delta) = f_1 f_2 \dots f_n$ ($f_k \in \{0, 1\}$ pour $1 \leq k \leq n$) défini de la manière suivante :

Pour tout k , si δ coupe une droite horizontale d'équation $y = c$ où c est un entier ($c \in \mathbb{N}$) sur la bande verticale ayant pour projection l'intervalle $]k-1, k]$, alors $f_k = 1$, sinon $f_k = 0$.

Exemple. La figure 1 donne un exemple de « codage » pour $n = 10$ de la droite de $\Delta(10)$ d'équation

$$y = \delta(x) = \frac{2}{7}x + \frac{5}{7}.$$



$$\Psi(\delta) = 1000100100$$

Figure 1

Les algorithmes seront écrits dans le langage Pascal, ou dans un langage algorithmique proche de Pascal ou encore dans le langage Fortran. Dans ce dernier cas, les pointeurs seront des indices dans des tableaux.

1. Montrer que les lettres du mot $\Psi(\delta) = f_1 f_2 \dots f_n$ peuvent se définir à l'aide de la fonction δ et de la fonction partie entière (la partie entière de x est notée $[x]$).
2. Pour $n = 3$, donner $\Psi(\delta)$ et $\Psi(\lambda)$ pour les droites d'équations $y = \delta(x) = \frac{1}{2}x$ et $y = \lambda(x) = \frac{1}{3}x + \frac{1}{3}$.

Soit D l'image de Δ par $\Psi : D = \Psi(\Delta)$.

Considérons le morphisme Φ de $\{0, 1\}^*$ dans $\{0, 1\}^*$ qui échange les lettres, c'est-à-dire tel que $\Phi(0) = 1$ et $\Phi(1) = 0$.

3. Montrer que si $f \in D$ alors $\Phi(f) \in D$.

Idée : considérer les droites $y = \delta(x) = \frac{p}{q}x + \frac{r}{q}$ et $y = \lambda(x) = \frac{q-p}{q}x + \frac{q-r-1}{q}$.

4. Soit $f \in \{0, 1\}^*$. On note $fact(i, f)$ le nombre de facteurs de f , de longueur i , deux à deux différents. On peut montrer que f appartient à D si et seulement si, pour $1 \leq i \leq |f|$, $fact(i, f)$ est inférieur ou égal à $i + 1$. En utilisant cette propriété, donner l'algorithme de la fonction **test1(tab, n)** qui renvoie **1** si le mot **f** de longueur **n** contenu dans le tableau **tab** appartient à D et **0** sinon.

Soit $f \in \{0, 1\}^*$. Nous dirons que f vérifie la condition (C) si et seulement si :

$$(C) \quad \begin{cases} \text{pour toute factorisation } f = f_1 u f_2 v \text{ avec } |u| = |v|, \text{ on a} \\ -1 \leq |u|_1 - |v|_1 \leq 1. \end{cases}$$

On considère le langage L constitué de tous les mots autres que 0^m et 1^m ($m \geq 0$) et satisfaisant la condition (C).

5. En utilisant la définition de L , donner l'algorithme de la fonction **test2(tab, n)** qui renvoie **1** si le mot **f** de longueur **n** contenu dans le tableau **tab** appartient à L et **0** sinon.
6. Soit f un mot de D tel que $f = ugv$ avec $|g| = n$ et $|g|_1 = m$. Montrer que toute droite $y = \delta(x)$ appartenant à $\Delta(|f|)$, telle que $\Psi(\delta) = f$, a une pente α vérifiant les inégalités suivantes :

$$\frac{m-1}{n} < \alpha < \frac{m+1}{n}.$$

7. Montrer que D est inclus dans L .
8. Soit $f \in L$ et commençant par la lettre 0. Montrer qu'il existe un entier $m \geq 0$ tel que le mot f s'écrive $f = 0u_1 0u_2 0 \dots 0u_n 0u_{n+1}$ avec :
 - (E1) pour tout i , $1 \leq i \leq n$, $u_i = 1^m$ ou $u_i = 1^{m+1}$, et
 - (E2) $u_{n+1} = 1^s$ avec $0 \leq s \leq m + 1$.

Considérons alors le mot réduit de f , noté $r(f)$, obtenu en effectuant les substitutions suivantes à partir de la factorisation de la question 8 ($f \in L$ et commence par un zéro) :

- (R1) à chaque facteur $0u_i$ ($1 \leq i \leq n$) est substitué la lettre 0 si $|u_i| = m$ et 1 si $|u_i| = m + 1$,
- (R2) au facteur $0u_{n+1}$ est substitué 1 si $|u_{n+1}| = m + 1$ et le mot vide si $|u_{n+1}| < m + 1$.

Tournez la page S.V.P.

9. Soit f un mot débutant par 0 et autre que $(01^m)^n 01^s$ ($0 \leq s \leq m$). Montrer que f appartient à L si et seulement si le mot réduit de f appartient à L .

10. Montrer que L est inclus dans D .

Idée :

considérer les droites $y = \delta(x) = \frac{p}{q}x + \frac{r}{q}$ et $y = \lambda(x) = \frac{p + qm}{p + q(m + 1)}x + \frac{r}{p + q(m + 1)}$.

11. En utilisant la caractérisation de la question 9, donner l'algorithme de la fonction **test3(tab,n)** qui renvoie 1 si le mot f de longueur n contenu dans le tableau **tab** appartient à D et 0 sinon.

12. Soit $g = 01$ l'image par Ψ de la droite $\delta_0(x) = \frac{1}{2}x$ ($\delta_0 \in \Delta(2)$). Considérons le morphisme θ qui substitue à 0 le mot 01 et à 1 le mot 011. Quelles droites codent le mot $\theta(g) = 01011$ et le mot $\theta^2(g)$? Montrer que $\theta^n(g)$ est un mot de longueur F_{2n+2} obtenu à partir de la droite d'équation $y = \delta_n(x)$ où $\delta_n(x) = \frac{F_{2n+1}}{F_{2n+2}}x$, la suite F_n étant la suite de Fibonacci $F_0 = 1, F_1 = 1$ et $F_{n+2} = F_{n+1} + F_n$.

13. Soient f un mot de D et θ_m ($m \geq 0$) le morphisme défini par $\theta_m(0) = 01^m$ et $\theta_m(1) = 01^{m+1}$. Montrer que les pentes des droites codées par les mots $f_n = \theta_m^n(f)$ convergent vers

$$\alpha_m = \frac{-m + \sqrt{m(m+4)}}{2} \text{ lorsque } n \rightarrow \infty.$$