INFORMATIQUE

Partie I - Polygones simples

Dans toute cette partie, le plan est muni d'un repère orthonormé fixé. On identifiera un vecteur et le couple de ses coordonnées dans ce repère.

Notes de programmation. On dispose des types `point, vecteur, segment` et `polygone.` Un `point` (ou un `vecteur`) est défini par deux entiers, un `segment` est défini par deux points et un `polygone` est une liste de points.

Remarque : dans tout le problème un segment, resp. un polygone, est toujours supposé formé de points distincts.

En Caml :

```
type point = { x:int ; y:int };;
type vecteur = { xv:int ; yv:int };;
type segment = { a:point ; b:point };;
type polygone == point list;;
```

On dispose par ailleurs de constructeurs de vecteurs et de segments à partir des points les définissant ;

```
let CreerVecteur p q = { xv = q.x-p.x ; yv = q.y-p.y};;
let CreerSegment p q = { a = p ; b = q };;
```

Les fonctions dont on demandera le code devront avoir pour signatures :

```
determinant : vecteur -> vecteur -> int = <fun>
produit_scalaire : vecteur -> vecteur -> int = <fun>
direct : point -> point -> point -> int = <fun>
meme_cote : segment -> point -> point -> int = <fun>
appartient : segment -> point -> int = <fun>
intersecte : segment -> segment -> int = <fun>
simplifie : polygone -> polygone = <fun>
en_dehors : polygone -> point = <fun>
interieur : polygone -> point -> int = <fun>
```

Pour simplifier la présentation Caml/Pascal, la notation fonctionnelle `f(u,v)` *est utilisée dans l'énoncé à la place de* `f u v` *en Caml.*

En Pascal :

```
type
  point=record x,y:integer end;
  vecteur=record xv,yv: integer end;
```

Filière MP

```
segment=record a,b:point end;
polygone=^cellule;
cellule=record contenu:point; suivant:polygone end;
```

Les fonctions/procédures dont on demandera le code auront pour entêtes :

```
function determinant(u,v:vecteur):integer;
function produit_scalaire(u,v:vecteur):integer;
function direct(a,b,c:point):integer;
function meme_cote(I:segment;p,q:point):integer;
function appartient(I:segment; p:point):integer;
function intersecte(I,J:segment):integer;
function simplifie (P:polygone):polygone;
procedure en_dehors(P:polygone; var res:point);
function interieur(P:polygone;pt:point):integer;
```

I.A - Intersection

Remarque : géométriquement, le segment $[a, b]$ est l'ensemble des points de la droite (a, b) situés entre a et b ; on rappelle que les segments considérés sont non réduits à un point.

I.A.1) Écrire des fonctions `determinant` et `produit_scalaire` telles que `determinant(u,v)` retourne le déterminant de la matrice formée par les vecteurs u et v, et `produit_scalaire(u,v)` retourne le produit scalaire des vecteurs u et v.

I.A.2) *Définition : étant donnés trois points a, b, c, le triangle (a, b, c) est* direct *si une mesure de l'angle $(\overrightarrow{ab}, \overrightarrow{ac})$ est dans l'intervalle $]0,\pi[$*, indirect *si une mesure de l'angle $(\overrightarrow{ab}, \overrightarrow{ac})$ est dans l'intervalle $]-\pi, 0[$*, aplati *si les trois points a, b et c sont alignés.*

Montrer que le triangle (a, b, c) est direct si et seulement si $\det(\overrightarrow{ab}, \overrightarrow{ac}) > 0$.

I.A.3) Écrire une fonction `direct` telle que `direct(a,b,c)` retourne 1 si le triangle (a, b, c) est direct, -1 s'il est indirect et 0 s'il est aplati.

I.A.4) Écrire une fonction `meme_cote` telle que `meme_cote(I,p,q)` retourne 1 si les points p et q sont du même côté de la droite D portée par le segment I, -1 s'ils ne sont pas du même côté, et 0 si p ou q est sur D.
Illustrer la démarche à l'aide de figures bien choisies.

I.A.5) Écrire, *en la justifiant*, une fonction `appartient` telle que `appartient(I,p)` retourne 1 si le point p appartient au segment I et -1 sinon.

I.A.6) Écrire une fonction `intersecte` telle que `intersecte(I,J)` retourne 1 si les segments I et J ont une intersection non vide et -1 sinon.
Utiliser et justifier à l'aide de figures le fait que, sauf cas particuliers à expliciter, I et J ont une intersection non vide lorsque les extrémités de I ne sont pas du même côté de J et que conjointement les extrémités de J ne sont pas du même côté de I.

I.B - Intérieur / Extérieur

Dans cette partie, n est un entier strictement supérieur à 2, $P = p_1 p_2 \ldots p_n$ est un polygone ; les points $p_1, \ldots, p_n$ sont les sommets du polygone ; les segments $[p_1, p_2], \ldots, [p_{n-1}, p_n], [p_n, p_1]$ en sont les arêtes. Par convention on pose $p_0 = p_n$ et $p_{i+kn} = p_i$ pour tout entier naturel k.

Géométriquement, un polygone est, par définition, l'ensemble des points de ses arêtes ; ainsi on dit qu'un point a est sur le polygone P s'il appartient à l'une de ses arêtes. On dit qu'un segment S coupe le polygone P si ce segment à une intersection non vide avec l'une des arêtes de P.

De plus, dans ce problème, le polygone P sera toujours supposé *simple*, ce qui signifie que deux arêtes quelconques non consécutives sont toujours disjointes et que deux arêtes consécutives ont un seul point commun.

Résultat admis 1. Un polygone simple P sépare le plan en deux composantes connexes, ce qui signifie que le complémentaire de P dans le plan comporte deux composantes connexes : l'*intérieur* de P est par définition la composante connexe bornée et l'autre, non bornée, est l'*extérieur* de P.

Résultat admis 2. Soient P un polygone simple, a et b deux points du plan qui ne sont pas sur P. Si le segment $[a, b]$ ne coupe pas P alors a et b sont dans la même composante connexe.

Résultat admis 3. Soient P un polygone simple, a et b deux points du plan qui ne sont pas sur P. Si le segment $[a, b]$ coupe P en un seul point qui n'est pas un sommet de P, alors a et b ne sont pas dans la même composante connexe.

I.B.1) Écrire une fonction `simplifie` telle que `simplifie(P)` supprime du polygone P les points p_i tels que p_{i-1}, p_i, p_{i+1} sont alignés. Évaluer sa complexité.
Dans la suite, le polygone P sera supposé sous sa forme simplifiée, i.e. il n'existe pas de sommet p_i tel que p_{i-1}, p_i, p_{i+1} sont alignés.

I.B.2) **Caml :** écrire une fonction `en_dehors` telle que `en_dehors P` retourne un point appartenant à la composante extérieure de P. Le temps d'exécution (dans le pire des cas) doit être linéaire en le nombre de points de P.

Pascal : écrire une procédure `en_dehors` telle que `en_dehors(P,pt)` place dans la variable `pt` un point appartenant à la composante extérieure de P. Le temps d'exécution (dans le pire des cas) doit être linéaire en le nombre de points de P.

I.B.3) Soit s un point à l'extérieur de P. En supposant que le segment $[q, s]$ ne contient aucun des sommets de P, dire dans quelle composante connexe se situe q, en fonction du nombre d'intersections de $[q, s]$ avec P.

I.B.4) En déduire une fonction `interieur` telle que `interieur(P,q)` renvoie 1 si q est dans la composante intérieure de P, -1 si q est dans la composante extérieure, et 0 si q est sur P. Le nombre d'opérations (dans le pire des cas) de la fonction doit être linéaire en le nombre de sommets de P.
On fera ici l'hypothèse que le point s à l'extérieur de P est tel qu'aucun sommet de P n'est sur le segment $[q, s]$.

I.B.5) Adapter cette fonction pour traiter tous les cas d'intersections possibles et donner alors le nombre d'opérations (dans le pire des cas).
Indication : on déplacera habilement le point extérieur s.

I.B.6) Soient deux points a et b qui ne sont pas sur P et tels que $[a, b]$ coupe P en un seul point qui est l'un de ses sommets p_i. Caractériser, par une condition portant sur p_{i-1}, p_{i+1} et $[a, b]$, l'appartenance des points a et b à la même composante connexe. *Justifier à l'aide de figures bien choisies.*

I.B.7) Soient deux points a et b qui ne sont pas sur P et tels que l'intersection de $[a, b]$ et de P est une arête $[p_i, p_{i+1}]$ de P. Caractériser, par une condition portant sur p_{i-1}, p_{i+2} et $[a, b]$, l'appartenance des points a et b à la même composante connexe. *Justifier à l'aide de figures bien choisies.*

I.B.8) Réécrire la fonction `interieur` de la question I.B.5 avec un nombre d'opérations (dans le pire des cas) linéaire par rapport à n.

Partie II - Complexité de communication

Lorsque E est un ensemble fini, $|E|$ désigne son cardinal. Lorsque x est un réel, $\lceil x \rceil$ (resp. $\lfloor x \rfloor$) désigne sa partie entière supérieure (resp. inférieure), c'est-à-dire l'unique entier vérifiant $x \leq \lceil x \rceil < x + 1$ (resp. $x - 1 < \lfloor x \rfloor \leq x$).

Soit une application $f : X \times Y \to Z$. Pour $y \in Y$ fixé, f^y désigne l'application partielle $f^y \left\{ \begin{array}{l} X \to Z \\ x \mapsto f(x, y) \end{array} \right.$

et pour $x \in X$ fixé, f_x l'application partielle $f_x \left\{ \begin{array}{l} Y \to Z \\ y \mapsto f(x, y) \end{array} \right.$

II.A - Communication à sens unique

Dans cette partie on modélise le scénario suivant. Soient X, Y et Z trois ensembles finis arbitraires et $f : X \times Y \to Z$ une fonction donnée. Deux personnes, Alice et Bob, veulent calculer $f(x, y)$ pour des valeurs $x \in X$ et $y \in Y$. La difficulté est que seule Alice connaît x et seul Bob connaît y. L'objectif est de déterminer l'information minimale qu'Alice doit envoyer à Bob pour que ce dernier puisse calculer $f(x, y)$.

Un *protocole à sens unique* $\mathscr{P}$ *calculant* f est un couple de fonctions (g_0, g_1), avec $g_0 : X \to \{0,1\}^*$ et $g_1 : \{0,1\}^* \times Y \to Z$, tel que $g_1(g_0(x), y) = f(x, y)$, pour tout $x \in X$ et $y \in Y$, $\{0,1\}^*$ désignant l'ensemble des mots sur l'alphabet $\{0,1\}$.

Autrement dit, $g_0(x)$ est l'information qu'Alice envoie à Bob, et $g_1(m, y)$ est la valeur calculée par Bob ayant reçu le message m d'Alice.

Le *coût sur l'entrée* (x, y) d'un protocole à sens unique $\mathscr{P}$ est la taille en bits de $g_0(x)$, c'est-à-dire la longueur de ce mot. *Le coût* d'un protocole à sens unique est le maximum de ses coûts sur toutes les entrées possibles. La *complexité de communication à sens unique de* f, notée $D^1(f)$, est le minimum des coûts des protocoles à sens unique calculant f.

Note sur la rédaction. Dans cette partie, la description d'un protocole se fera par la donnée **explicite** des fonctions g_0 et g_1 correspondantes.

II.A.1) Exemples - Propriétés simples

a) Calculer $D^1(f)$ lorsque f est une fonction constante.

b) Trouver, pour f donnée, un protocole dont le coût est $\lceil \log_2 |X| \rceil$. Que peut-on en déduire sur $D^1(f)$?

c) Soient $X = \{1,\dots,n\}$, p un entier tel que $1 \le p \le n-1$, et f une fonction de période p par rapport à la première variable, *i.e.* $f(x + p, y) = f(x, y)$ pour tout $(x,y) \in X \times Y$ tel que $x + p \le n$. Montrer que $D^1(f) \le \lceil \log p \rceil$.

d) X et Y sont ici l'ensemble des parties de $\{1,\dots,n\}$, et, pour $x, y \subset \{1,\dots,n\}$, $f(x, y)$ est le plus grand élément de $x \cup y$. Montrer que $D^1(f) \le \lceil \log_2 n \rceil$. Comparer à la majoration de la question II.A.1- b.

e) Montrer que pour chaque $y \in Y$ fixé, $D^1(f) \ge \left\lceil \log_2 \left| \mathrm{Im}(f^y) \right| \right\rceil$, avec $\mathrm{Im}(f^y)$ l'ensemble des valeurs prises par f^y.

f) Calculer $D^1(f)$, f désignant la fonction de la question II.A.1-d.

II.A.2) Calcul exact de $D^1(f)$

On suppose que $X = \{1,\dots,p\}$ et $Y = \{1,\dots,q\}$, pour deux entiers p et q strictement positifs. La matrice de communication de $f : X \times Y \to Z$ est un tableau à p lignes et q colonnes, noté M_f, dont les lignes sont indexées par X et les colonnes

par Y, et définie par $(M_f)_{x,y} = f(x,y)$, pour $x \in X$ et $y \in Y$. Soit t le nombre de lignes distinctes de M_f. On va montrer l'égalité $D^1(f) = \lceil \log_2 t \rceil$.

a) Donner un protocole calculant f dont le coût est $\lceil \log_2 t \rceil$.

b) Montrer que le protocole précédent est optimal, *i.e.* tout protocole calculant f a un coût supérieur ou égal à $\lceil \log_2 t \rceil$.

c) Application : Alice et Bob détiennent chacun un mot de taille n sur l'alphabet $\{0,1\}$. Quelle information minimale Alice doit-elle fournir à Bob pour que celui-ci puisse décider de façon certaine si les deux mots sont égaux ?

On commencera par modéliser précisément cette situation en terme de protocole de communication à sens unique.

Donner un protocole à sens unique dont le coût atteint cette borne.

Lorsque X et Y sont quelconques, l'égalité $D^1(f) = \lceil \log_2 t \rceil$ est maintenue en prenant pour t le nombre d'applications partielles $(f_x)_{x \in X}$ distinctes. En effet, chaque fonction f_x correspond à une ligne de la matrice dans le cas précédent.

II.B - Communication avec aller-retour

Nous considérons maintenant des communications générales où les deux participants peuvent interagir plusieurs fois entre eux. De plus, ils veulent maintenant calculer **tous les deux** la valeur de la fonction. Soient X, Y et Z trois ensembles finis arbitraires et $f : X \times Y \to Z$ une fonction donnée. Étant donnés $x \in X$, détenu par Alice et $y \in Y$, détenu par Bob, le but est de calculer $f(x,y)$ en s'échangeant le minimum de bits. Un protocole (calculant f) est maintenant divisé en étapes. À chaque étape, le protocole désigne une personne, indifféremment Alice ou Bob. La personne désignée envoie alors un bit à l'autre personne. Ce bit ne dépend que des bits échangés précédemment et de l'entrée détenue par la personne désignée. Lorsque le protocole termine, alors les deux participants sont en mesure de calculer $f(x,y)$ d'après les bits échangés et la valeur de x (respectivement y).

Plus formellement : un *protocole* $\mathscr{P}$ de domaine $X \times Y$ et à valeurs dans $\mathbb{Z}$ est un arbre binaire étiqueté comme suit. Chaque nœud interne est étiqueté soit « Alice » soit « Bob », chaque feuille est étiquetée par un élément $z \in \mathbb{Z}$, et enfin les arêtes par des sous-ensembles de X ou de Y avec la contrainte suivante. Si un nœud interne est étiqueté « Alice » (respectivement « Bob ») alors ses arêtes gauche et droite sont respectivement étiquetées par des sous-ensembles disjoints E_g et E_d de X (respectivement de Y).

La *valeur* du protocole $\mathscr{P}$ sur l'entrée (x,y), avec $x \in X$ et $y \in Y$, est l'étiquette de la feuille atteinte en partant de la racine et en parcourant l'arbre suivant le chemin décrit ci-après. A chaque nœud interne étiqueté « Alice » (respectivement « Bob ») le parcours suit le fils gauche si $x \in E_g$ (respectivement $y \in E_g$) et

le fils droit dans l'autre cas, *i.e.* $x \in E_d$ (respectivement $y \in E_d$). Pour indiquer le sens du parcours, Alice (respectivement Bob) envoie à Bob (respectivement Alice) le bit 0 pour le fils gauche et 1 pour le fils droit. Le protocole sera toujours supposé bien défini de sorte qu'à chaque nœud, x ou y selon le cas, appartient à la réunion $E_g \cup E_d$.

Le *coût sur l'entrée* (x, y) d'un protocole $\mathscr{P}$ est la longueur totale du chemin sur l'entrée (x, y), soit donc la profondeur de la feuille atteinte.

Le *coût* d'un protocole est le maximum de ses coûts sur toutes les entrées possibles, soit encore la hauteur de l'arbre $\mathscr{P}$.

Un protocole calcule f si sa valeur est $f(x, y)$ sur chaque entrée (x, y).

La *complexité de communication* de f, notée $D(f)$, est le minimum des coûts des protocoles calculant f.

Note sur la rédaction. Dans cette partie, les protocoles seront souvent décrits sous *forme compacte* où chaque série de bits émis consécutivement par le même participant est concentré sur la même ligne.

Exemple. Soient $X = \{0,1\}^3$, $Y = \{1, 2, 3\}$ et $Z = \{0,1\}$. Posons $f((x_1, x_2, x_3), y) = x_y$. Suivent un protocole pour f ainsi que sa forme compacte.

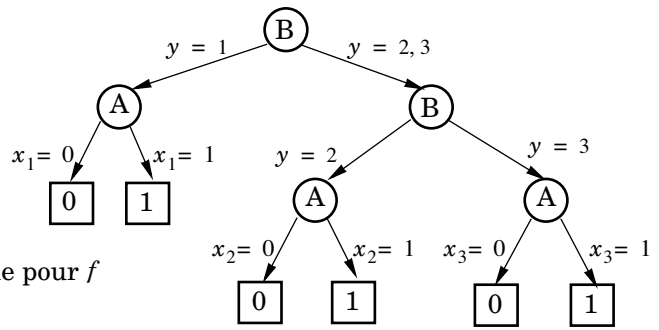

Figure 1 : Protocole pour f

1. Si $y = 1$ Bob envoie à Alice le bit $b = 0$ sinon le bit $b = 1$.
2. Si $b = 0$ alors Alice renvoie x_1 à Bob
   et le protocole termine et renvoie $f(x, y) = x_1$.
3. Si $y = 2$ Bob envoie à Alice le bit $c = 0$, sinon le bit $c = 1$.
4. Alice envoie le bit x_{2+c} à Bob.
5. Le protocole termine et renvoie $f(x, y) = x_{2+c}$.

Figure 2 : Forme compacte du protocole pour f.

II.B.1) **Quelques exemples**

a) Voici la forme compacte d'un protocole où $X = Y = Z = \{0, 1, 2, 3\}$.

1. Alice calcule $a = 0$ si $x \in \{0,1\}$, et $a = 1$ sinon.
   Alice envoie a à Bob.
2. Si $a = 0$ et $y \in \{2, 3\}$ Bob renvoie $b = 1$.
   Si $a = 0$ et $y \in \{0,1\}$ Bob renvoie $b = 0$.
   Si $a = 1$ et $y \in \{0,1\}$ Bob renvoie $b = 1$.
   Si $a = 1$ et $y \in \{2, 3\}$ Bob renvoie $b = 0$.
3. Si $b = 1$ le protocole termine et renvoie $f(x, y) = 1 - a$.
   Sinon Alice renvoie $c = x - 2a$.
4. Bob renvoie $d = 1$ si $c < y - 2a$, et $d = 0$ sinon.
5. Le protocole termine et renvoie $f(x, y) = d$.

Représenter l'arbre binaire associé à ce protocole, préciser la fonction f calculée par le protocole, et donner le coût du protocole.

b) $X = \{0,1\}^n$, $Y = \{1, \ldots, n\}$, $Z = \{0,1\}$ et $f(x, y) = x_y$ où x_y représente le y-ième bit de x. Montrer l'inégalité : $D(f) \leq \lceil \log_2 n \rceil + 1$.

c) $X = \{1, \ldots, n\} \times \{0,1\}^n$, $Y = \{1, \ldots, n\}^n$, $Z = \{0,1\}$ et $f((x,u), y) = u_i$ avec $i = y_x$, x-ième bit de y. Montrer l'inégalité : $D(f) \leq 2\lceil \log_2 n \rceil + 1$.

II.B.2) **Comparaison avec la communication à sens unique**

Le but des questions suivantes est de montrer l'inégalité suivante et son optimalité :

$$\lfloor \log_2 D^1(f) \rfloor + 1 \leq D(f) \leq D^1(f) + \lceil \log_2 |\mathrm{Im}\,(f)| \rceil .$$

a) Montrer l'inégalité : $D(f) \leq D^1(f) + \lceil \log_2 |\mathrm{Im}\,(f)| \rceil$.

b) Montrer l'inégalité : $D(f) \geq \lceil \log_2 |\mathrm{Im}\,(f)| \rceil$, et construire une fonction $f_0 : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ satisfaisant $D(f_0) = D^1(f_0) + \lceil \log_2 |\mathrm{Im}\,(f_0)| \rceil$.

c) Montrer l'inégalité : $D^1(f) \leq 2^{D(f)} - 1$ et en déduire que $\lfloor \log_2 D^1(f) \rfloor + 1 \leq D(f)$.

d) Montrer que la fonction de la question II.B.1-b) satisfait (pour n bien choisi mais arbitrairement grand) l'égalité : $\lfloor \log_2 D^1(f) \rfloor + 1 = D(f)$.

••• FIN •••