

# I3 - Algorithmique

Durée : 3h

Documents autorisés : **AUCUN** (calculatrice comprise)<sup>1</sup>

## Remarques :

- Veuillez lire attentivement les questions avant de répondre.
- Le barème donné est un barème indicatif qui pourra évoluer lors de la correction.
- Rendez une copie propre.
- N'utilisez pas de crayon à papier.

## 1 Questions de cours (2 points)

Pour rappel, l'algorithme du tri par fusion est :

**procédure** triFusion (E/S t : Tableau[1..MAX] d'Entier ,E nb :Naturel)

**debut**

triFusionRecuratif(t,1,nb)

**fin**

**procédure** triFusionRecuratif (E/S t : Tableau[1..MAX] d'Entier ,E d,f :Naturel)

**debut**

**si** d<f **alors**

triFusionRecuratif(t,d,(d+f) div 2)

triFusionRecuratif(t,((d+f) div 2)+1,f)

fusionner(t,d,(d+f) div 2,f)

**finsi**

**fin**

Sachant que la complexité de la procédure *fusionner* est en  $O(n)$  démontrez que la complexité de la procédure *triFusion* est en  $O(n \log_2(n))$ .

## 2 Exercice de TD (5 points)

### 2.1 Recherche dichotomique itérative

Donnez l'algorithme de la fonction suivante qui retourne l'indice de la dernière occurrence d'un entier  $e$  que l'on sait présent dans le tableau  $t$  possédant  $nb$  entiers significatifs triés en ordre croissant :

- **fonction** indiceDerniereOccurence (t : Tableau[1..MAX] d'Entier , nb : Naturel, e : Entier) : Naturel

Testez votre algorithme sur le tableau suivant en cherchant l'indice de la dernière occurrence du nombre 5 :

|   |   |   |   |   |   |    |    |    |
|---|---|---|---|---|---|----|----|----|
| 1 | 5 | 5 | 5 | 7 | 9 | 10 | 10 | 12 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7  | 8  | 9  |

### 2.2 Recherche dichotomique récursive

Faites de même, mais cette fois de manière récursive. Votre récursivité est terminale ou non terminale ? Justifiez.

1. Sauf les dictionnaires pour les étudiants non francophones

### 3 Problème (13 points)

Soit le type `Point2D` utilisable à l'aide des fonctions et procédures suivantes :

- **fonction** `point2D (x,y : Reel) : Point2D`
- **fonction** `obtenirX (p : Point2D) : Reel`
- **fonction** `obtenirY (p : Point2D) : Reel`
- **fonction** `distanceEuclidienne (p1,p2 : Point2D) : Reel`
- **procédure** `translater (E/S p : Point2D,E vecteur : Point2D)`
- **procédure** `faireRotation (E/S p : Point2D,E centre : Point2D, angle : Reel)`

#### 3.1 Polyligne

« Une ligne polygonale, ou ligne brisée (on utilise aussi parfois polyligne par traduction de l'anglais *poly-line*) est une figure géométrique formée d'une suite de segments, la seconde extrémité de chacun d'entre eux étant la première du suivant.[...] Un polygone est une ligne polygonale fermée. » (Wikipédia)

La figure 1 présente deux polygones composées de 5 points.

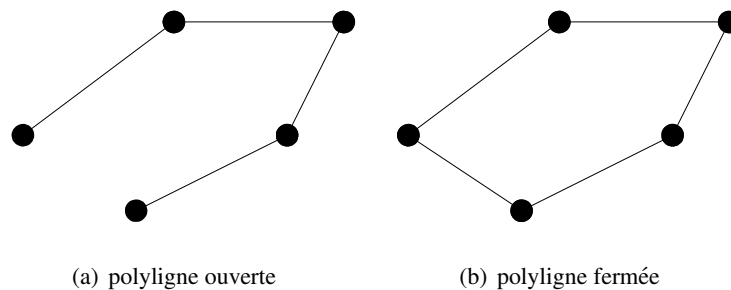


FIGURE 1 – Deux polygones

De cette définition nous pouvons faire les constats suivants :

- Une polyligne est constituée d'au moins deux points ;
- On peut obtenir le nombre de points d'une polyligne ;
- Une polyligne est ouverte ou fermée (qu'elle soit ouverte ou fermée ne change pas le nombre de points : dans le cas où elle est fermée, on considère qu'il y a une ligne entre le dernier et le premier point) ;
- On peut insérer, supprimer des points à une polyligne (par exemple la figure 2 présente la suppression du troisième point de la polyligne ouverte de la figure 1) ;
- On peut parcourir les points d'une polyligne ;
- On peut effectuer des transformations géométriques (translation, rotation, etc.) ;
- On peut calculer des propriétés d'une polyligne (par exemple sa longueur totale).

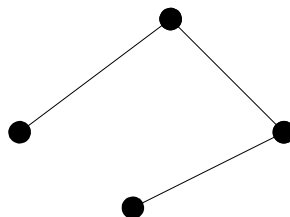


FIGURE 2 – Suppression d'un point

## 3.2 Conception préliminaire

Proposez la signature des fonctions et procédures correspondant aux opérations suivantes :

- créer une polyligne ouverte à partir de deux `Point2D` ;
- savoir si une polyligne est fermée ;
- ouvrir une polyligne ;
- fermer une polyligne ;
- connaître le nombre de points d’une polyligne ;
- obtenir le  $i$ ème point d’une polyligne ;
- insérer le  $i$ ème point d’une polyligne ;
- supprimer le  $i$ ème point d’une polyligne (on suppose qu’elle a au moins 3 points) ;
- calculer la longueur d’une polyligne ;
- traduire une polyligne ;
- faire une rotation d’une polyligne.

## 3.3 Conception détaillée

On propose de représenter le type `Polyligne` de la façon suivante :

**Type Polyligne = Structure**

lesPts : **Tableau**[1..MAX] de `Point2D`

nbPts : **Naturel**

estFermee : **Booleen**

**finstructure**

Proposez les fonctions et procédures correspondant aux opérations suivantes :

- créer une polyligne ouverte à partir de deux `Point2D` ;
- ouvrir une polyligne ;
- traduire une polyligne.

## 3.4 Utilisation d’une polyligne

Dans cette partie, nous sommes utilisateur du type `Polyligne` et nous respectons le principe d’encapsulation.

### 3.4.1 Point à l’intérieur

Nous supposons posséder la fonction suivante qui permet de calculer l’angle orienté en degré formé par les segments  $(ptCentre, pt1)$  et  $(ptCentre, pt2)$  :

- **fonction** `angle (ptCentre, pt1, pt2 : Point2D) : Reel`

Il est possible de savoir si un point  $pt$  est à l’intérieur ou à l’extérieur d’une polyligne fermée en calculant la somme des angles orientés formés par les segments issus de  $pt$  vers les points consécutifs de la polyligne. En effet si cette somme en valeur absolue est égale à  $360^\circ$  alors le point  $pt$  est à l’intérieur de la polyligne, sinon il est à l’extérieur.

Par exemple, sur la figure 3, on peut savoir algorithmiquement que  $pt$  est à l’intérieur de la polyligne car  $|\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5| = 360$ .

Proposez le code de la fonction suivante en supposant que  $p$  est bien une polyligne fermée :

- **fonction** `estALinterieur (p : Polyligne ; pt : Point2D) : Booleen`

### 3.4.2 Surface d’une polyligne par la méthode de monté-carlo

Une des façons d’approximer la surface d’une polyligne est d’utiliser la méthode de Monté-Carlo. Le principe de cette méthode est de « calculer une valeur numérique en utilisant des procédés aléatoires, c’est-à-dire des techniques probabilistes » (Wikipédia). Dans le cas du calcul d’une surface, il suffit de tirer au hasard des points qui sont à l’intérieur du plus petit rectangle contenant la polyligne. La surface  $S$  de la polyligne pourra alors être approximée par la formule suivante :

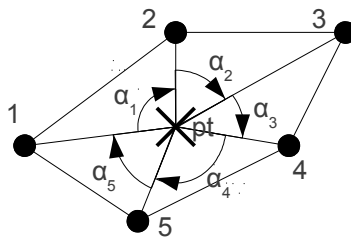


FIGURE 3 – Point à l'intérieur d'une polyligne

$$S \approx \text{SurfaceDuRectangle} \times \frac{\text{Nb points dans la polyligne}}{\text{Nb points total}}$$

Par exemple, sur la figure 4, en supposant que le rectangle fasse 3 cm de hauteur et 4,25 cm de largeur, et qu'il y a 28 points sur 39 qui sont à l'intérieur de la polyligne, sa surface  $S$  peut être approximée par :

$$S \approx 3 \times 4,25 \times \frac{28}{39} = 9,39 \text{ cm}^2$$

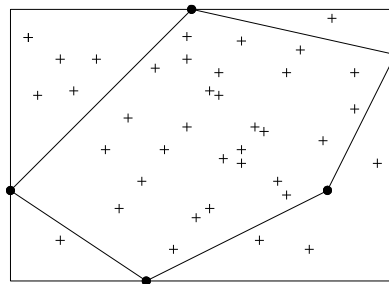


FIGURE 4 – Calcul de la surface d'une polyligne par la méthode de Monté-Carlo

On suppose posséder la fonction suivante qui permet d'obtenir un réel aléatoire entre une borne minimum et une borne maximum :

– **fonction** reelAleatoire (borneMin, borneMax : **Reel**) : **Reel**

1. Proposez l'analyse descendante pour le calcul d'une surface d'une polyligne à l'aide de la méthode de Monté-Carlo.
2. Donnez les signatures des procédures et fonctions de votre analyse descendante.
3. Donnez l'algorithme de l'opération principale (au sommet de votre analyse descendante).