Département STPI Lundi 20 Juin 2011

I3 - Algorithmique

Durée : *3h*
Documents autorisés : **AUCUN** (calculatrice comprise) [1]

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
  triFusionRecursif(t,1,nb)
fin
procédure triFusionRecursif **(E/S** t : **Tableau[**1..MAX**] d'Entier** ,**E** d,f :**Naturel)**
debut
  si d<f **alors**
    triFusionRecursif(t,d,(d+f) div 2)
    triFusionRecursif(t,((d+f) div 2)+1,f)
    fusionnner(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 $O(nlog_2(n))$.

Solution proposée :
Cf. le cours

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 occurence 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 occurence du nombre 5 :

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

Solution proposée :
fonction rechercheDichotomique (t : **Tableau[**1..MAX**] d'Entier** ; n : **Naturel** ; element : **Entier) : Naturel**
  Déclaration a,b,m : **Naturel**
debut

1. Sauf les dictionnaires pour les étudiants non francophones

```
    a ← 1
    b ← n
    m ← (a + b) div 2
    tant que a < b faire
        si t[b]=e alors
            a ← b
        sinon
            si t[m] ≤ element alors
                a ← m
            sinon
                b ← m-1
            finsi
            m ← (a + b) div 2
        finsi
    fintantque
    retourner b
fin
```

2.2 Recherche dichotomique récursive

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

Solution proposée :

```
fonction rechercheDichotomiqueR (t : Tableau[1..MAX] d'Entier ; a,b : Naturel ; element : Entier) : Naturel
    Déclaration  m : Naturel
debut
    si a<b alors
        m ← (a + b) div 2
        si t[b]=e alors
            retourner rechercheDichotomiqueR(t,b,b,element)
        sinon
            si t[m] ≤ element alors
                retourner rechercheDichotomiqueR(t,m,b,element)
            sinon
                retourner rechercheDichotomiqueR(t,a,m-1,element)
            finsi
        finsi
    sinon
        retourner b
    finsi
fin
```

3 Problème (13 points)

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

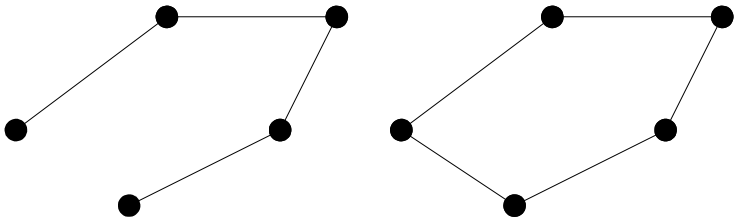
- **fonction** point2D (x,y : **Reel**) **:** Point2D
- **fonction** obtenirX (p : Point2D) **: Reel**
- **fonction** obtenirY (p : Point2D) **: Reel**
- **fonction** distanceEuclidienne (p1,p2 : Point2D) **: Reel**
- **procédure** translater (**E/S** p : Point2D,**E** vecteur : Point2D)

– **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 *polyline*) 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 polylignes composées de 5 points.



(a) polyligne ouverte (b) polyligne fermée

FIGURE 1 – Deux polylignes

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 fermé, on considère qu'il 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 supression 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).

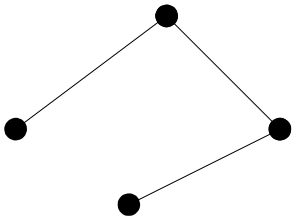


FIGURE 2 – Supression d'un point

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 ;
– connaitre le nombre de points d'un polyligne ;
– obtenir le ième point d'une polyligne ;

- insérer le ième point d'une polyligne ;
- supprimer le ième point d'une polyligne (on suppose qu'elle a au moins 3 points) ;
- calculer la longueur d'un polyligne ;
- translater une polyligne ;
- faire une rotation d'une polyligne.

Solution proposée :

- **fonction** polyligne (pt1,pt2 : Point2D, estFermee : **Booleen) :** Polyligne
- **fonction** estFermee (pl ; Polyligne) **: Booleen**
- **procédure** fermer **(E/S** pl : Polyligne**)**
- **procédure** ouvrir **(E/S** pl : Polyligne**)**
- **fonction** nbPoints (pl ; Polyligne**) : Naturel**
- **fonction** iemePoint (pl ; Polyligne, position : **Naturel) :** Point2D
- **procédure** ajouterPoint **(E/S** pl : Polyligne,**E** pt : Point2D, position : **Naturel)**
- **procédure** supprimer **(E/S** pl : Polyligne,**E** position : **Naturel)**
- **fonction** longueur (pl : Polyligne**) : Reel**
- **procédure** translater **(E/S** pl : Polyligne,**E** vecteur : Point2D**)**
- **procédure** faireRotation **(E/S** pl : Polyligne,**E** centre : Point2D, angleEnRadian : **Reel)**

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 ;
- translater une polyligne.

Solution proposée :

fonction polyligne (pt1,pt2 : Point2D, estFermee : **Booleen) :** Polyligne
  Déclaration resultat : Polyligne
debut
  resultat.nbPts ← 2
  resultat.lesPts[1] ← pt1
  resultat.lesPts[2] ← pt2
  resultat.estFermee ← Faux
  retourner resultat
fin
procédure ouvrir **(E/S** pl : Polyligne**)**
debut
  pl.estFermee ← Faux
fin
procédure translater **(E/S** pl : Polyligne,**E** vecteur : Point2D**)**
  Déclaration i : **Naturel**
debut
  pour i ←1 **à** nbPoints(pl) **faire**
    trasnlater(pl.lesPts[i],vecteur)
  finpour

fin

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 $360°$ 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 $|\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5| = 360$.

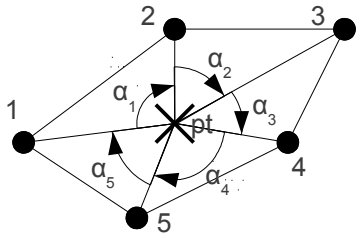


FIGURE 3 – Point à l'intérieur d'une polyligne

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

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

Solution proposée :

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

Déclaration i : **Naturel**
sommeAngle : **Reel**

debut
sommeAngle ← 0
pour i ←1 **à** nbPoints(p)-1 **pas de** 0 **faire**
sommeAngle ← sommeAngle+angle(pt,iemePoint(p,i),iemePoint(p,i+1))
finpour
sommeAngle ← sommeAngle+angle(pt,iemePoint(p,nbPoints(p)),iemePoint(p,1))
retourner sommeAngle=360 ou sommeAngle=-360
fin

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 :

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

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 :

$$S \approx 3 \times 4,25 \times \frac{28}{38} = 9,39 \text{ cm}^2$$

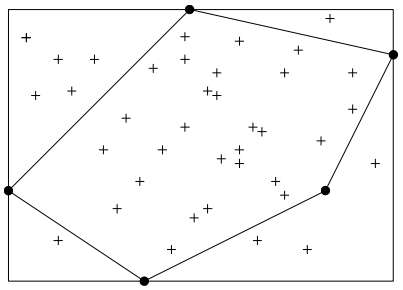
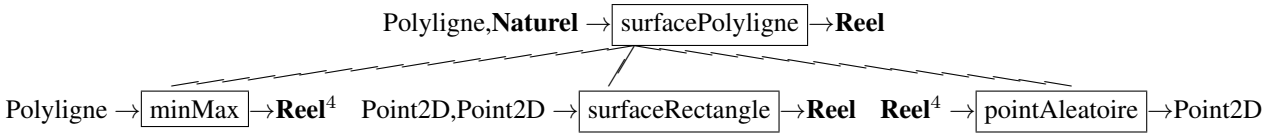


FIGURE 4 – Calcul de la surface d'une polyligne par la méthode de Monté-Carlo

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

– **fonction** reelAleatoire (borneMin,bornneMax : **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.
   Solution proposée :



2. Donnez les signatures des procédures et fonctions de votre analyse descendante.
   Solution proposée :
   – **fonction** surfacePolyligne (p : Polyligne, nbPoints : **Naturel) : Reel**
   – **procédure** minMax **(E** p : Polyligne,**S** xMin,xMax,yMin,yMax :**Reel)**
   – **fonction** surfaceRectangle (ptBG,ptHD : Point2D**) : Reel**
   – **fonction** pointAleatoire (borneMinX,borneMaxY,borneMinY,borneMaxY : **Reel) :** Point2D

3. Donnez l'algorithme de l'opération principale (au sommet de votre analyse descendante).
   Solution proposée :

```
fonction surfacePolyligne (p : Polyligne, nbPoints : Naturel) : Reel
debut
    minMax(p,xMin,xMax,yMin,yMax)
    surface ← surfaceRectangle(point2D(xMin,yMin),point2D(xMax,yMax))
    nbDans ← 0
    pour i ←1 à nbPoints faire
        si estALinterieur(p,pointAleatoire(xMin,xMax,yMin,yMax)) alors
            nbDans ← nbDans+1
        finsi
    finpour
```

    retourner surface*nbDans/nbPoints
fin