STPI Jeudi 25 Juin 2009

I3 - Algorithmique

Durée : *3h00*
Documents autorisés : **AUCUN** (calculatrice comprise)

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 Distance de Hamming (5 points)

1.1 Distance de Hamming entre deux mots

La distance de Hamming entre deux chaînes de caractères, appelées « mots », de même longueur est égale au nombre de caractères, à la même position, qui diffère. Par exemple la distance de Hamming entre "rose" et "ruse" est de 1, alors que la distance de Hamming entre "110110" et "000101" est de 4.

Proposer l'algorithme de la fonction **itérative** suivante qui permet de calculer la distance de Hamming de deux mots, non vides, que l'on sait de même longueur[1] :
- **fonction** distanceHammingDeuxMots **(**mot1,mot2 : **Chaine de caracteres) : Naturel**

1.2 Distance de Hamming d'un langage

Un langage est un ensemble de mots. La distance de Hamming d'un langage est égale au minimum des distances de Hamming entre deux mots de ce langage différents deux à deux.

Proposer le corps de la fonction **itérative** suivante qui permet de calculer la distance de Hamming d'un langage d'au moins 2 mots qui possède uniquement des mots de même longueur et tous différents deux à deux[1] :
- **fonction** distanceHammingLangage **(**leLangage : **Tableau[**1..MAX**] de Chaine de caracteres** , nbMots : **Naturel) : Naturel**

2 Somme des chiffres d'un naturel (3 points)

Proposer l'algorithme de la fonction **récursive** suivante qui permet de calculer la somme des chiffres d'un naturel. Par exemple $sommeChiffres(524)$ retournera la valeur 11.
- **fonction** sommeChiffres **(**n : **Naturel) : Naturel**

3 Un petit dessin (3 points)

3.1 Rappels et prérequis

Nous avons vu en cours qu'un écran graphique d'ordinateur (représenté par une variable de type `Ecran`) est un quadrillage de pixels. Nous avons vu aussi que l'on pouvait obtenir ou modifier la couleur d'un pixel à l'aide des deux procédures suivantes :
- **procédure** fixerCouleurPixel **(E/S** e : Ecran,**E** x,y : **Naturel**, c : Couleur**)**
- **fonction** obtenirCouleurPixel **(**e : Ecran,x,y : **Naturel) :** Couleur

Pour le type `Ecran`, on considère avoir aussi la procédure suivante :
- **procédure** segment **(E/S** e : Ecran,**E** x1,y1,x2,y2 : **Naturel**, c : Couleur**)**

[1] Cela signifie que vous n'avez pas à faire ce(s) test(s) dans votre algorithme

– qui permet dessiner sur l'écran un segment de droite de couleur c entre les pixels de coordonnées $(x1, y1)$ et $(x2, y2)$

Enfin, on considère avoir en plus les procédures mathématiques suivantes :

– **procédure** rotation **(E** xc,yc,x1,y1 : Naturel, angle : **Reel**, **S** x2,y2 : Naturel**)**
  – qui permet de calculer l'image, de coordonnées $(x2, y2)$, d'un pixel, de coordonnées $(x1, y1)$, par une rotation de centre de coordonnées (xc, yc) et d'angle $angle$ (exprimé en degrés). Pour rappel, l'image M' d'un point M par une rotation de centre C et de rayon α est défini par $\widehat{MCM'} = \alpha$ et $\|\overrightarrow{CM'}\| = \|\overrightarrow{CM}\|$.

– **procédure** homothetie **(E** xc,yc,x1,y1 : Naturel, k : **Reel**, **S** x2,y2 : Naturel**)**
  – qui permet de calculer l'image, de coordonnées $(x2, y2)$, d'un pixel, de coordonnées $(x1, y1)$, par une homothétie de centre de coordonnées (xc, yc) et de rapport k. Pour rappel, l'image M' d'un point M par une homothétie de centre C et de rapport k est défini par $\overrightarrow{CM'} = k\overrightarrow{CM}$.

Par exemple l'exécution de la procédure `exemple` (donnée uniquement à titre d'illustration) avec les paramètres effectifs $e, 100, 100, c$ donne la figure 1.

procédure exemple **(E/S** e : Ecran, **E** x,y : Naturel, c : Couleur**)**

Déclaration x3,y3,x4,y4 : **Naturel**

debut
segment(e,x,y,x+300,y,c)
rotation(x+300,y,x+300+283,y,45,x3,y3)
homothetie(x3,y3,x3+200,y3,0.5,x4,y4)
segment(e,x+300,y,x3,y3,c)
segment(e,x3,y3,x4,y4,c)
fin

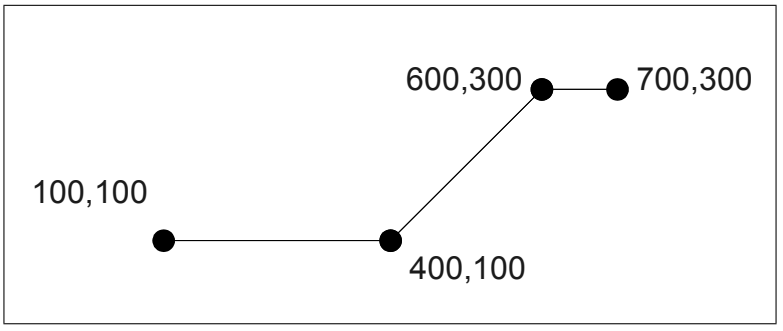

FIG. 1 – Exemple d'utilisation des procédures `segment`, `rotation` et `homothétie` (les coordonnées des points sont données à titre indicatif)

3.2 Procédure mystère

Soit la procédure `mystere` suivante :

procédure mystere **(E/S** e : Ecran, **E** x1,y1,x2,y2,n : Naturel, c : Couleur**)**

Déclaration x3,y3,x4,y4,x5,y5 : **Naturel**

debut
si n=0 **alors**
segment(e,x1,y1,x2,y2,c)
sinon
homothetie(x1,y1,x2,y2,0.33,x3,y3)
homothetie(x1,y1,x2,y2,0.66,x5,y5)
rotation(x3,y3,x5,y5,60,x4,y4)
mystere(e,x1,y1,x3,y3,n-1,c)

```
        mystere(e,x3,y3,x4,y4,n-1,c)
        mystere(e,x4,y4,x5,y5,n-1,c)
        mystere(e,x5,y5,x2,y2,n-1,c)
    finsi
fin
```

Dessiner l'exécution de cette procédure avec les paramètres effectifs $e, 0, 0, 1000, 0, 2, c$.

4 Traitement du Signal (9 points)

Un « signal » est une suite numérique (des réels). Les opérations habituelles sur un signal sont :
- initialiser un signal avec n fois la même valeur (n de type Naturel) ;
- obtenir le nombre de valeurs d'un signal ;
- obtenir la ième valeur d'un signal (avec $0 \leq i \leq n$) ;
- fixer la ième valeur d'un signal (avec $0 \leq i \leq n$) ;
- ajouter une nouvelle valeur à la fin d'un signal (si le signal avait au préalable n valeurs, il en possède ensuite $n + 1$).

4.1 Conception préliminaire (1,5 point)

On considère posséder le type `Signal`. Proposer les signatures des fonctions et procédures, correspondant aux opérations précédentes permettant de manipuler un signal.

4.2 Conception détaillée (2 points)

On décide de représenter le type `Signal` de la façon suivante :

```
Type Signal = Structure
    donnees : Tableau[1..MAX] de Reel
    nbDonnees : Naturel
finstructure
```

Donner les algorithmes des fonctions et procédures de la conception préliminaire.

4.3 Utilisation du type `Signal`

Pour rappel, dans cette partie nous sommes **utilisateur** du type Signal. Pour manipuler un signal, nous n'utilisons donc que les fonctions et procédures proposées par la conception préliminaire.

4.3.1 Min et max (1,5 points)

Proposer l'algorithme d'une procédure `calculerMinMax` qui permettra d'obtenir la plus petite et la plus grande valeur d'un signal (que l'on suppose possédé au moins une valeur).

4.3.2 Lissage de courbe (4 points)

L'objectif ici est de développer un « filtre non causal », c'est-à-dire une fonction qui lisse un signal en utilisant une fenêtre glissante pour moyenner les valeurs : la i^{eme} valeur du nouveau signal est la moyenne des valeurs du signal source se trouvant à l'intérieur d'une fenêtre glissante centrée en i (Cf. figure 2). Pour les premières et dernières valeurs, seules les valeurs dans la fenêtre sont prises en compte.

Après avoir fait une analyse descendante du problème, proposer l'algorithme de la fonction `filtreNonCausal` avec la signature suivante sachant que le nombre de valeur d'un signal est obligatoirement supérieur à 1 et que la taille de la fenêtre est obligatoirement impaire[1] :

- **fonction** filtreNonCausal **(**signalNonLisse : Signal, tailleFenetre : **Naturel) :** Signal

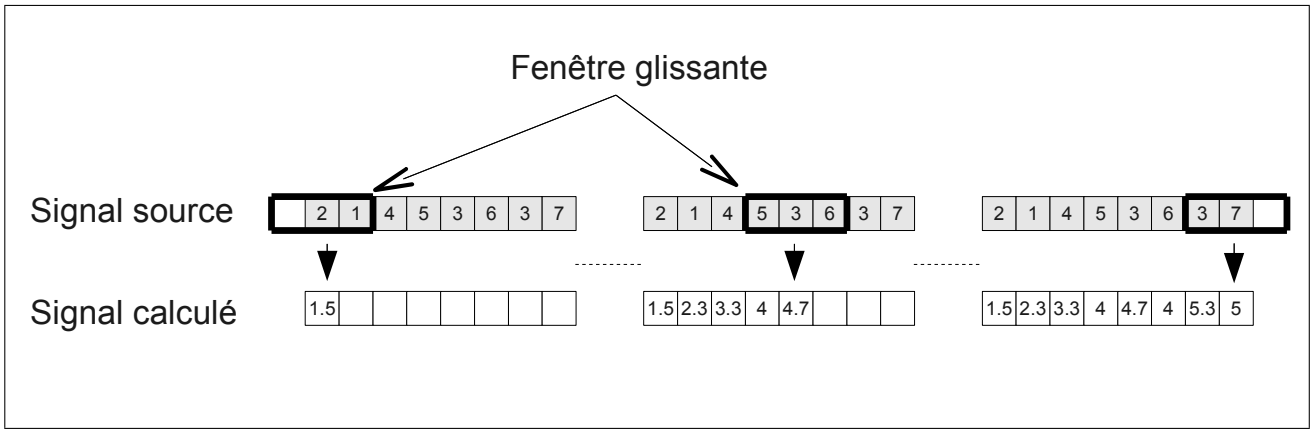

FIG. 2 – Lissage d'un signal avec une fenêtre de taille 3