Option Informatique en Sup MPSI
TP : algorithmique des puces à ADN

1 Vecteurs en Caml

▶ En Caml, un *vecteur* est une structure de donnée linéaire (comme une liste), avec accès direct à chaque élément (donc : pas comme une liste). Les éléments sont tous du même type, et sont numérotés de 0 à $n-1$ si n est la longueur du vecteur. Le fragment de programm suivant éclaircira ceci :

```
let v = [|2;-3;7;0;666|] ;;
v.(2) <- 5 * v.(1) ;;
print_int v.(2) ;;
```

Les résultats affichés par Caml sont :

```
v : int vect = [|2; -3; 7; 0; 666|]
- : unit = ()
-15- : unit = ()
```

▶ Quelques fonctions utiles :

```
make_vect : int -> 'a -> 'a vect
vect_length : 'a vect -> int
list_of_vect : 'a vect -> 'a list
vect_of_list : 'a list -> 'a vect
```

▶ L'utilisation des vecteurs (valeurs *mutables*) fait sortir du cadre de la programmation fonctionnelle pure. On tolère cet écart, dans la mesure où les programmes écrits dans ce style restent totalement limpides (voir l'exemple dans la section 2).

2 Programmation : opérations sur les matrices

▶ Nous aurons besoin de manipuler des matrices ; le court programme suivant montre comment définir une première matrice a, modifier un des coefficients de a, et enfin construire la transposée b de a :

```
let a = make_matrix 2 3 0 ;; a.(1).(0) <- 1 ;;
let transpose a =
 let nl = vect_length a and nc = vect_length a.(0) in
 let a' = make_matrix nc nl a.(0).(0) in
 for i = 0 to nc - 1 do
  for j = 0 to nl - 1 do
   a'.(i).(j) <- a.(j).(i)
  done
 done ;
 a' ;;
let b = transpose a ;;
```

Vous noterez que `a` et `b` sont de type `int vect vect`.

Question 1 • Rédigez en Caml une fonction :

```
copy_matrix : 'a vect vect -> 'a vect vect
```

spécifiée comme suit : `copy_matrix m` construit une nouvelle matrice p, de mêmes dimensions et de même contenu que m.

Question 2 • Rédigez en Caml une fonction :

```
do_matrix : ('a -> 'b) -> 'a vect vect -> 'b vect vect
```

spécifiée comme suit : `do_matrix f m` construit une nouvelle matrice p, de mêmes dimensions que m, et vérifiant $p_{i,j} = f(m_{i,j})$ quels que soient i et j.

Question 3 • Rédigez en Caml une fonction :

```
blit_matrix : 'a vect vect -> 'a vect vect -> int -> int -> ()
```

spécifiée comme suit : `blit_matrix p q di dj` recopie la matrice p dans la matrice q, la copie étant décalée de di cases vers la droite et dj cases vers le bas. Ceci ne peut se faire que si $nl(p) + dj \leqslant nl(q)$ et $nc(p) + di \leqslant nc(q)$, où $nl(m)$ est le nombre de lignes et $nc(m)$ le nombre de colonnes de la matrice m.

Question 4 • Rédigez en Caml une fonction :

```
sym_x : 'a vect vect -> 'a vect vect
```

spécifiée comme suit : `sym_x m` construit une nouvelle matrice p, de mêmes dimensions que m, déduite de m par symétrie autour d'un axe horizontal. Rédigez de même la fonction `sym_y`.

3 Puces à ADN

▶ Une *puce à ADN* d'ordre n se compose d'un substrat carré (d'environ 1 cm de côté), décomposé en 4^n zones carrées de même taille ; danas chacune de ces zones, on implante un grand nombre de molécules d'ADN monobrin identiques, de longueur n. Chacune de ces molécules est un mot sur l'alphabet $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$. Deux molécules implantées sur des zones différentes doivent différer par au moins une lettre. La figure 1 présente deux dispositions différentes pour une puce à ADN d'ordre 2.

AA	AC	CA	CC
AG	AT	CG	CT
GA	GC	TA	TC
GG	GT	TG	TT

AA	AC	CC	CA
AG	AT	CT	CG
GG	GT	TT	TG
GA	GC	TC	TA

Figure 1: implantation fractale (à gauche) et avec un code de Gray (à droite)

▶ La fabrication d'une telle puce requiert une succession de $4n$ étapes, que l'on peut représenter par le mot $m = (\mathsf{ACGT})^n$. Initialement, on dépose sur la puce des molécules dotées d'une base qui va adhérer au substrat, et d'un groupe protecteur photolabile. Lors de la k-ième étape, on pose un *masque* sur la puce et on l'expose aux ultra-violets : ceci élimine les groupes protecteur des zones exposées ; on enlève le masque et on déverse sur la puce une solution contenant la base m_k, à laquelle est attaché un groupe protecteur. La figure 2 donne une représentation (très approximative).

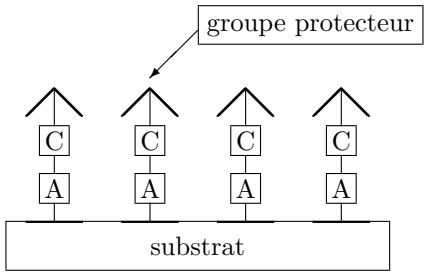

Figure 2: puce à ADN

▶ La figure 3 donne deux exemples de masques ; ce sont ceux associés à l'étape 5 (dépôt du deuxième A) pour chacune des deux dispositions présentées à la figure 1. Remarque : curieusement, les zones transparentes du masque sont représentées en noir !

▶ Avec l'implantation fractale, les puces sont définies par les deux dessins ci-dessous ; la notation $\mathsf{X}\mathsf{F}_n$ désigne une puce d'ordre n, dans laquelle on a ajouté une lettre X en tête de chaque mot.

Figure 3: masques pour l'étape $\mathsf{A}-2$

$$\mathsf{F}_1 = \begin{array}{|c|c|}\hline \mathsf{A} & \mathsf{C} \\ \hline \mathsf{G} & \mathsf{T} \\ \hline \end{array} \qquad \mathsf{F}_{n+1} = \begin{array}{|c|c|}\hline \mathsf{A}\,\mathsf{F}_n & \mathsf{C}\,\mathsf{F}_n \\ \hline \mathsf{G}\,\mathsf{F}_n & \mathsf{T}\,\mathsf{F}_n \\ \hline \end{array}$$

▶ À la frontière entre les parties cachées et les parties exposées, se produisent des phénomènes nuisibles (diffraction). On veut donc minimiser la longueur totale des bords des masques. Par exemple, les longueurs de bord des deux masques de la figure 3 sont respectivement 12 et 8 (on ne compte pas ce qui est ≪au bord≫ de la puce).

▶ Pour ce faire, nous disposerons les 4^n mots selon un *code de Gray bidimensionnel* noté G_n et défini par les équations suivantes :

$$\mathsf{G}_1 = \begin{array}{|c|c|}\hline \mathsf{A} & \mathsf{C} \\ \hline \mathsf{G} & \mathsf{T} \\ \hline \end{array} \qquad \mathsf{G}_{n+1} = \begin{array}{|c|c|}\hline \mathsf{A}\,\mathsf{G}_n & \mathsf{C}\,\overrightarrow{\mathsf{G}_n} \\ \hline \mathsf{G}\,\mathsf{G}_n\downarrow & \mathsf{T}\,\overrightarrow{\mathsf{G}_n\downarrow} \\ \hline \end{array}$$

La notation $\overrightarrow{\mathsf{G}_n}$ (respectivement : $\mathsf{G}_n\downarrow$) signifie que G_n a subi une symétrie autour de son axe vertical (respectivement : horizontal). La signification de $\overrightarrow{\mathsf{G}_n\downarrow}$ est claire !

Question 5 Notons $C_{gray}(n)$ la longueur de bord totale des $4n$ masques utilisés pour une puce d'ordre n avec la méthode de Gray ; ainsi, $C_{gray}(1) = 8$ et $C_{gray}(2) = 48$. Donnez l'expression exacte de $C_{gray}(n)$ en fonction de n.

Question 6 ⋆⋆ Donnez de même l'expression exacte, en fonction de n, de la longueur de bord totale $C_{fractale}(n)$ des $4n$ masques utilisés pour une puce d'ordre n avec la méthode ≪fractale≫.

4 Programmation : puces à ADN

▶ Nous utiliserons le type `base` défini comme suit :

```
type base = A | C | G | T ;;
```

Question 7 • Rédigez en Caml une fonction :

```
make_fractal_chip : int -> base list vect vect
```

spécifiée comme suit : `make_fractal_chip n` construit la puce d'ordre n, avec l'implantation ≪fractale≫.

Question 8 • Rédigez en Caml une fonction :

```
make_gray_chip : int -> base list vect vect
```

spécifiée comme suit : `make_gray_chip n` construit la puce d'ordre n, avec le code de Gray bidimensionnel.

Question 9 • Rédigez en Caml une fonction :

```
masque : base -> int -> base list vect vect -> bool vect vect
```

spécifiée comme suit : `masque b k m` construit une matrice p de même ordre que m, et décrivant le masque associé à l'étape (b, k).

Question 10 • Rédigez en Caml une fonction :

```
dessine_masque : bool vect vect -> unit
```

spécifiée comme suit : `dessine_masque m` dessine le masque m.

Question 11 • Rédigez en Caml une fonction :

```
mask_border_length : bool vect vect -> int
```

spécifiée comme suit : `mask_border_length p` calcule la longueur de bord du masque p.

Question 12 • Rédigez en Caml une fonction :

```
total_border_length : base list vect vect -> int
```

spécifiée comme suit : `total_border_length c` calcule la longueur de bord totale de la puce c.

FIN