Travaux pratiques 2+

Puces ADN

MPII : Ocaml

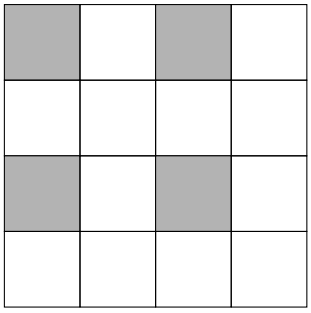
I Introduction

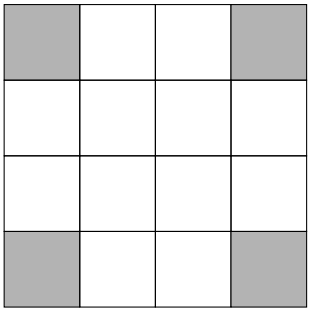
Une puce à ADN d'ordre n se compose d'un substrat carré (d'environ 1 cm de c^oté), décomposé en 4^n zones carrées de même taille ; dans chacune de ces zones, on implante un grand nombre de molécules d'ADN mono-brin identiques, de longueur n. Chacune de ces molécules est une suite de bases choisis dans l'ensemble $\{A, C, G, T\}$. Deux molécules implantées sur des zones différentes doivent différer par au moins une lettre (une base). La figure ci-dessous 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

La fabrication d'une telle puce requiert une succession de $4n$ étapes où l'on place successivement les 4 bases pour chaque niveau. On dépose une base sur la puces après avoir masqué les parties qui ne reçoivent pas la base concernée. La figure ci-dessous donne deux exemples de masques ; ce sont ceux associés à l'étape 5 (dépôt du deuxième A) pour chacune des deux dispositions présentées avant. Les zones transparentes du masque sont représentées en noir.





II Opérations sur les matrices

- On crée une matrice avec la fonction

```
Array.make_matrix : int -> int -> 'a -> 'a array array
```

- Le nombre de lignes, $nl(m)$, de m est `Array.length m`.
- La ligne d'indice i ($0 \leq i < nl(m)$) de m est `m.(i)`, c'est un tableau.
- Le nombre de colonnes, $nc(m)$, de m est `Array.length m.(0)`.
- Le terme d'indices i et j de m est `m.(i).(j)` ; $0 \leq i < nl(m)$), $0 \leq j < nc(m)$)

Exercice 1 - Copie
Écrire une fonction qui, à partir d'une matrice m, construit une nouvelle matrice p, de mêmes dimensions et de même contenu que m.

```
copy_matrix : 'a array array -> 'a array array
```

Exercice 2 - Application d'une fonction
Écrire une fonction qui, à partir d'une fonction f et d'une matrice m, construit une nouvelle matrice p, de mêmes dimensions que m et vérifiant $p_{i,j} = f(m_{i,j})$.

```
map : ('a -> 'b) -> 'a array array -> 'b array array
```

Exercice 3 - Recopie partielle
Écrire une fonction qui, à partir de 2 matrices m et p et de deux indices dx et dx, recopie la matrice m dans p en la décalant de d_x cases vers la droite (les colonnes) et de d_y cases vers le bas (les lignes).

```
blit : 'a array array -> 'a array array -> int -> int -> unit
```

Cela n'est possivle que si les tailles des matrices vérifient $nl(m)+dy \leq nl(p)$ et $nc(m)+dx \leq nc(p)$ mais il n'est pas demandé de le vérifier.

Exercice 4 - Symétrie
Écrire une fonction qui, à partir d'une matrice m, construit une nouvelle matrice p, de mêmes dimensions que m, déduite de m par symétrie horizontale : on échange les lignes 2 à 2 depuis les bords vers le centre

```
sym_h : 'a array array -> 'a array array
```

Écrire de même la symétrie verticale, `sym_y`.

III Implantation fractale

Dans cette implantation, les puces sont définies par une structure F_n définie par récurrence.

$$F_1 = \begin{array}{|c|c|}\hline A & C \\ \hline G & T \\ \hline \end{array} \qquad F_n = \begin{array}{|c|c|}\hline AF_{n-1} & CF_{n-1} \\ \hline GF_{n-1} & TF_{n-1} \\ \hline \end{array}$$

La notation XF_k désigne une puce d'ordre k, dans laquelle on a ajouté une lettre X en tête de élément de F_k.
On modélisera les suites de bases par des chaînes de caractères composées de `"A"`, `"C"`, `"G"` et `"T"`.

- On concatène deux chaînes par l'opérateur `^` : `"bon" ^ "jour"` donne `"bonjour"`.
- `ch.[i]` donne le caractère d'indice i (on commence par 0) de la chaîne `ch` : on notera que c'est un caractère, encadré de guillemets simples (`'x'`), et non une chaîne de longueur 1.

Exercice 5 - Construction de la puce
Écrire une fonction qui construit la puce d'ordre n.

```
chip_f : int -> string array array
```

Exercice 6 - Construction du masque
Écrire une fonction qui construit le masque associé au caractère b à la k-ième étape à partir de la matrice m définissant une puce. k varie entre 1 et n.

```
masque : char -> int -> string array array
                     -> bool array array
```

Exercice 7 - Dessin du masque
Écrire une fonction qui dessine le masque entré comme paramètre, par exemple en imprimant des caractères `*` pour les trous.

```
dessin : bool array array -> unit
```

IV Implantation de Gray

Dans l'implantation précédente le nombre de "trous" dans les masques est maximal : il y a 4^k trous dans le masque pour la k-ième étape et pour chaque base. Pour espérer diminuer ce nombre de trous on propose l'implantation suivante

$$G_1 = F_1 = \begin{array}{|c|c|}\hline A & C \\ \hline G & T \\ \hline \end{array} \qquad G_n = \begin{array}{|c|c|}\hline AG_{n-1} & C\overleftrightarrow{G_{n-1}} \\ \hline GF_{n-1}\updownarrow & T\overleftrightarrow{G_{n-1}}\updownarrow \\ \hline \end{array}$$

$\overleftrightarrow{M}$ désigne la matrice obtenue par symétrie verticale de M.
$M\updownarrow$désigne la matrice obtenue par symétrie horizontale de M.

Exercice 8 - Construction de la puce
Écrire une fonction qui construit la puce d'ordre n avec l'implantation de Gray.

```
chip_g : int -> string array array
```

Exercice 9 - Nombre de trous
Y-a-t-il moins de trous ?

Solutions

Solution de l'exercice 1 - Copie

Le type de la matrice copiée est celui de la matrice originale d'où l'initialisation avec `m.(0).(0)`.

```
let copy_matrix m =
   let nl = Array.length m in
   let nc = Array.length m.(0) in
   let p = Array.make_matrix nl nc m.(0).(0) in
   for i = 0 to (nl - 1) do
      for j = 0 to (nc - 1) do
         p.(i).(j) <- m.(i).(j) done done;
   p;;
```

Solution de l'exercice 2 - Application d'une fonction

```
let map f m =
   let nl = Array.length m in
   let nc = Array.length m.(0) in
   let p = Array.make_matrix nl nc (f m.(0).(0)) in
   for i = 0 to (nl - 1) do
      for j = 0 to (nc - 1) do
         p.(i).(j) <- f m.(i).(j) done done;
   p;;
```

Solution de l'exercice 3 - Recopie partielle

Attention : `dx` est un décalage de colonnes et `dy` un décalage de lignes.
La fonction modifie un de ses paramètres, elle ne renvoie rien (en fait elle renvoie `()`).

```
let blit m p dx dy =
   let nl = Array.length m in
   let nc = Array.length m.(0) in
   for i = 0 to (nl - 1) do
      for j = 0 to (nc - 1) do
         p.(i + dy).(j + dx) <- m.(i).(j) done done;;
```

Solution de l'exercice 4 - Symétrie

```
let sym_h m =
   let nl = Array.length m in
   let nc = Array.length m.(0) in
   let p = Array.make_matrix nl nc m.(0).(0) in
   for i = 0 to (nl - 1) do
      for j = 0 to (nc - 1) do
         p.(i).(j) <- f m.(nl - 1 - i).(j) done done;
   p;;
```

```
let sym_v m =
   let nl = Array.length m in
   let nc = Array.length m.(0) in
   let p = Array.make_matrix nl nc m.(0).(0) in
   for i = 0 to (nl - 1) do
      for j = 0 to (nc - 1) do
         p.(i).(j) <- f m.( i).(nc - 1 - j) done done;
   p;;
```

Solution de l'exercice 5 - Construction de la puce

À définition par récurrence, algorithme récursif.

```
let rec chip_f n =
   match n with
   |1 -> [|[|"A"; "C"|]; [|"G"; "T"|]|]
   |n -> let m = chip_f (n-1) in
         let d = Array.length m in
         let p = Array.make_matrix (2*d) (2*d) "" in
         blit (map (fun ch -> "A"^ch) m) p 0 0;
         blit (map (fun ch -> "C"^ch) m) p d 0;
         blit (map (fun ch -> "G"^ch) m) p 0 d;
         blit (map (fun ch -> "T"^ch) m) p d d;
         p;;
```

Solution de l'exercice 6 - Construction du masque

On peut supposer que les matrices sont carrées : `nl = nc`.

```
let masque car k m =
   let n = Array.length m in
   let p = Array.make_matrix n n false in
   for i = 0 to (n - 1) do
      for j = 0 to (n - 1) do
         p.(i).(j) <- m.(i).(j).[k-1] = car done done;
   p;;
```

Solution de l'exercice 7 - Dessin du masque

```
let dessin m =
   let n = Array.length m in
   print_newline();
   for i = 0 to (n - 1) do
      for j = 0 to (n - 1) do
         if m.(i).(j)
         then print_string "*"
         else print_string "." done;
      print_newline () done;;;
```

Solution de l'exercice 8 - Construction de la puce

Solution de l'exercice 9 - Nombre de trous