

Option Informatique en Spé MP et MP*

Programmation Caml : permutations

Placez au début de votre programme un commentaire indiquant votre nom, votre classe, la date, et le sujet du T.P. Rédigez les réponses aux questions autres que celles de pure programmation sur une feuille de papier séparée : vous n'êtes pas ici pour faire preuve de vos éventuels talents de dactylographe.

► Nous nous proposons d'écrire un ensemble de fonctions pour manipuler les permutations de l'intervalle discret $\llbracket 1, n \rrbracket$. Soit s une telle permutation ; nous la représenterons par la liste des images par s des éléments de $\llbracket 1, n \rrbracket$, dans l'ordre «naturel» d'énumération. Par exemple, avec $n = 4$, la permutation s définie par $s(1) = 3$, $s(2) = 2$, $s(3) = 4$ et $s(4) = 1$ sera représentée par la liste `[3;2;4;1]`.

► Le générateur aléatoire de Caml `random__int` est une fonction de type `int -> int`, spécifiée comme suit : `random__int n` est une variable aléatoire uniformément répartie dans l'intervalle discret $\llbracket 0, n - 1 \rrbracket$. Deux appels consécutifs de `random__int` sont indépendants, au moins en apparence.

Question 1 • Rédigez en Caml une fonction :

```
extrait : 'a list -> ('a * 'a list)
```

spécifiée comme suit : `extrait l` rend un couple (t, q) où t est un élément de l choisi au hasard, et q est la liste déduite de l par suppression de cet élément. Vous lèverez une exception si l est la liste vide.

Question 2 • Rédigez en Caml une fonction :

```
crée_permutation : int -> int list
```

spécifiée comme suit : `crée_permutation n` rend une liste de longueur n , représentant une permutation «aléatoire» de $\llbracket 1, n \rrbracket$. Indication : utilisez la fonction `extrait`. Vous lèverez une exception si $n < 0$.

Question 3 • Rédigez en Caml une fonction :

```
image : int -> int list -> int
```

spécifiée comme suit : si l est une liste de longueur n représentant une permutation s de $\llbracket 1, n \rrbracket$, alors `image i l` rendra l'image de $s(i)$. Vous lèverez une exception si i n'est pas dans l'intervalle $\llbracket 1, n \rrbracket$.

Question 4 • Rédigez en Caml une fonction :

```
uniq : 'a list -> bool
```

spécifiée comme suit : `uniq l` indique si la liste l est «sans doublon». Précisons ceci : notant $n = |l|$ et $l = (\ell_1, \dots, \ell_n)$, `uniq l` rend la valeur `false` ssi il existe dans $\llbracket 1, n \rrbracket$ deux indices i et j distincts tels que $\ell_i = \ell_j$. Indication : utilisez la fonction `mem` de la bibliothèque Caml.

Question 5 • Utilisez `uniq` pour rédiger en Caml une fonction :

```
vérifie_permutation : int list -> bool
```

spécifiée comme suit : `vérifie_permutation l` vérifie si la liste l , de longueur $n \geq 1$, représente une permutation de $\llbracket 1, n \rrbracket$. Par exemple :

- `vérifie_permutation [3;2;4;1]` rendra `true` ;
- `vérifie_permutation [3;2;4;2]` et `vérifie_permutation [3;5;1]` rendront `false`.

Question 6 • Calculez le nombre de *Cons* (c'est-à-dire le nombre d'emplois de l'opérateur `::`) effectués lorsque vous appliquez `vérifie_permutation` à une liste de longueur n .

Question 7 • Avec la fonction `sort` de la bibliothèque `sort`, proposez une version de `vérifie_permutation` de coût $\mathcal{O}(n \ln n)$.

► Nous allons maintenant rédiger une version de `vérifie_permutation` dont le coût sera $\mathcal{O}(n)$.

Question 8 • Deux raisons peuvent faire qu'une liste l d'entiers de longueur n ne représente pas une permutation de $\llbracket 1, n \rrbracket$:

- la présence d'éléments n'appartenant pas à cet intervalle ;

- l'absence d'éléments appartenant à cet intervalle.

Expliquez comment utiliser un `int vect` de longueur n correctement initialisé, pour tester si ℓ tombe dans l'un de ces cas de figure.

Question 9 • Montrez qu'en fait un `bool vect` de longueur n suffit.

Question 10 • Mettez en œuvre l'idée précédente.

► À la réflexion, un vecteur de longueur n semble plus commode pour représenter une permutation de $\llbracket 1, n \rrbracket$. C'est ce que nous ferons désormais. Toutes les fonctions que vous rédigerez doivent avoir un coût linéaire en la longueur des arguments, et lèveront une exception lorsque leur(s) argument(s) est (sont) incohérent(s).

Question 11 • Révisez la bibliothèque `vect` :

- création d'un vecteur ;
- accès à une composante ;
- modification d'une composante.

Question 12 • Avec cette représentation, comment s'écrit la fonction `image` ?

Question 13 • Rédigez en Caml une fonction :

```
cycle : int -> int vect
```

spécifiée comme suit : `cycle n` construit le vecteur qui représente la permutation v de $\llbracket 1, n \rrbracket$ définie par $v(i) = i + 1$ pour $1 \leq i < n$ et $v(n) = 1$.

Question 14 • Rédigez en Caml une fonction :

```
compte_transpositions : int vect -> int
```

spécifiée comme suit : `compte_transpositions v` rend le nombre de transpositions de la permutation v représentée par le vecteur v .

Question 15 • Rédigez en Caml une fonction :

```
compose_permutations : int vect -> int vect -> int vect
```

spécifiée comme suit : `compose_permutations v w` construit le vecteur qui représente la permutation $v \circ w$, où v et w sont les permutations représentées par les vecteurs v et w .

Question 16 • Rédigez en Caml une fonction :

```
inverse_permutation : int vect -> int vect
```

spécifiée comme suit : `inverse_permutation v` construit le vecteur qui représente la permutation v^{-1} , où v est la permutation représentée par le vecteur v .

Question 17 • Rédigez en Caml une fonction :

```
ordre : int vect -> int
```

spécifiée comme suit : `ordre v` calcule l'*ordre* de la permutation v représentée par le vecteur v , c'est-à-dire le plus petit exposant $k \geq 1$ tel que v^k soit l'identité de $\llbracket 1, n \rrbracket$.

Question 18 • Rédigez en Caml une fonction :

```
ordre_moyen : int -> int -> int
```

spécifiée comme suit : `ordre_moyen n p` calcule la moyenne des ordres de p permutations de $\llbracket 1, n \rrbracket$ construites au hasard (avec la fonction `crée_permutation` ou une fonction qui s'en déduit).

FIN