Génération d'objets combinatoires

Durée : 3 heures

On rappelle que $[\![1,n]\!]$ désigne l'ensemble des entiers strictement positifs et inférieurs ou égaux à n.

Dans tout ce problème, on considère un entier $n \in \mathbb{N}^*$, et on note :

- $\mathscr{P}_n$ l'ensemble des parties de $[\![1,n]\!]$;
- $\mathfrak{S}_n$ l'ensemble des bijections de $[\![1,n]\!]$ dans lui-même.

Le but de chacune des deux parties de ce problème est de présenter des algorithmes élémentaires qui génèrent la liste des éléments de chacun de ces deux ensembles. Ces deux parties sont indépendantes.

Dans les algorithmes qui vous seront demandés en Caml, on ne tiendra pas compte d'un éventuel dépassement de capacité lié à la représentation en complément à deux des entiers.

Partie I. Génération des parties d'un ensemble

Dans cette partie, on cherche à engendrer la liste des éléments de $\mathscr{P}_n$.

Un élément de $\mathscr{P}_n$ sera représenté en Caml par le type *int list*, l'ordre dans lequel apparaissant les éléments n'ayant pas d'importance. Ainsi, la partie $\{2,5,4\}$ sera aussi bien représentée par la liste `[5; 4; 2]` que par `[4; 2; 5]`. Une partie de $\mathscr{P}_n$ sera représentée par le type *int list list* ; par exemple, $\big\{\{1,4\},\{5\},\{3,4,7\}\big\}$ sera représentée (entre autre) par la liste `[[4; 3; 7]; [4; 1]; [5]]`.

Question 1. À une partie A de $[\![1,n]\!]$, on associe sa *fonction caractéristique* $\chi_A : [\![1,n]\!] \to \{0,1\}$ définie par :

$$\forall k \in [\![1,n]\!],\ \chi_A(k) = \begin{cases} 1 & \text{si } k \in A \\ 0 & \text{si } k \notin A \end{cases}.$$

a) Montrer, en exhibant l'application réciproque, que l'application $(A \mapsto \chi_A)$ réalise une bijection entre $\mathscr{P}_n$ et l'ensemble $\mathcal{F}([\![1,n]\!],\{0,1\})$ des applications de $[\![1,n]\!]$ vers $\{0,1\}$.

b) En déduire que l'application $\phi : \mathscr{P}_n \to [\![0,2^n-1]\!]$ définie par $\phi(A) = \sum_{k=1}^{n} \chi_A(k) 2^{k-1}$ est une bijection.

On dira que $\phi(A)$ est l'*ordre* de A ; on se propose dans un premier temps d'engendrer les éléments de $\mathscr{P}_n$ par ordre croissant.

Question 2. Étant donné un élément k de $[\![1,n]\!]$ et une partie $\mathscr{B}$ de $\mathscr{P}_n$ telle que pour tout $A \in \mathscr{B}$, $k \notin A$, on pose :

$$k \oplus \mathscr{B} = \Big\{ A \cup \{k\} \;\Big|\; A \in \mathscr{B} \Big\}$$

En d'autres termes, on ajoute l'élément k à toutes les parties $A \in \mathscr{P}_n$ qui appartiennent à $\mathscr{B}$.

Par exemple, $2 \oplus \big\{\emptyset, \{1,3\}, \{4\}\big\} = \big\{\{2\}, \{1,2,3\}, \{2,4\}\big\}$.

a) Écrire en Caml une fonction **`ajoute`** qui prend en arguments un entier k et une partie $\mathscr{B}$ de $\mathscr{P}_n$ dont aucun élément ne contient k, et qui retourne $k \oplus \mathscr{B}$.

```
ajoute: 'a -> 'a list list -> 'a list list
```

b) Comment obtenir $\mathscr{P}_n$ à partir de $\mathscr{P}_{n-1}$ et de l'opérateur $\oplus$?
En déduire une fonction **`parties1`** qui prend en argument un entier n et qui retourne la liste des éléments de $\mathscr{P}_n$ *rangée par ordre croissant*. On demande une preuve rigoureuse de ce dernier point.

```
parties1: int -> int list list
```

On rappelle que `@` est la forme infixe de l'opérateur de concaténation des listes.

On propose maintenant une autre manière d'engendrer les éléments de $\mathscr{P}_n$. On définit une suite d'éléments de $\mathscr{P}_n$ en posant $A_1 = \{n\}$, et en construisant A_p à partir de A_{p-1} de la manière suivante :

- si $A_{p-1} = \{k_1, \dots, k_j\}$ avec $j \geqslant 2$ et $k_1 < \cdots < k_j$, alors $A_p = \begin{cases} \{k_1 - 1, k_1, \dots, k_j\} & \text{si } k_1 > 1 \\ \{k_2 - 1, k_3, \dots, k_j\} & \text{si } k_1 = 1 \end{cases}$;
- si $A_{p-1} = \{k_1\}$ avec $k_1 > 1$, alors $A_p = \{k_1 - 1, k_1\}$;
- si $A_{p-1} = \{1\}$, alors $A_p = \emptyset$;
- si $A_{p-1} = \emptyset$, alors A_p n'est pas défini.

Question 3.

a) Calculer la suite $A_1, \dots, A_{16}$ lorsque $n = 4$.

b) En raisonnant par récurrence sur $n \in \mathbb{N}^*$, montrer que A_p est défini pour $p \in [\![1, 2^n]\!]$, et que :

$$\mathscr{P}_n = \left\{ A_p \;\middle|\; p \in [\![1, 2^n]\!] \right\}.$$

Question 4.

a) On suppose dans cette question les éléments de $\mathscr{P}_n$ représentés par des listes *ordonnées par ordre croissant*. Écrire en Caml une fonction **`suivant`** calculant l'élément A_p à partir de A_{p-1}.

```
suivant: int list -> int list
```

b) En déduire une fonction **`parties2`** qui prend en argument un entier n et qui retourne la liste des éléments de $\mathscr{P}_n$ en suivant le principe décrit ci-dessus.

```
parties2: int -> int list list
```

Partie II. Génération des permutations

Dans cette partie, on cherche à engendrer la liste des éléments de $\mathfrak{S}_n$.
Un élément σ de $\mathfrak{S}_n$ sera identifié au n-uplet $\langle \sigma(1), \dots, \sigma(n) \rangle$, et sera représenté par une liste. Par exemple, la permutation $\sigma \in \mathfrak{S}_3$ définie par $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 2$ sera identifiée au triplet $\langle 3, 1, 2 \rangle$ et représentée par la liste `[3; 1; 2]`.

Soit σ un élément de $\mathfrak{S}_n$. On dit que le couple $(i, j) \in [\![1, n]\!]^2$ est une *inversion* de σ lorsque $i < j$ et $\sigma(j) < \sigma(i)$. On notera $\mathrm{inv}(\sigma)$ le nombre d'inversions de σ.
Si $\sigma \in \mathfrak{S}_n$, on appelle *code de Lehmer* de σ le n-uplet $c(\sigma) = \big(c_1(\sigma), \dots, c_n(\sigma)\big) \in \mathbb{N}^n$ défini par :

$$\forall i \in [\![1, n]\!],\ c_i(\sigma) = \mathrm{card}\left\{ j \in [\![i+1, n]\!] \;\middle|\; \sigma^{-1}(j) < \sigma^{-1}(i) \right\}.$$

Question 5.

a) Expliquer comment calculer $c_i(\sigma)$ lorsque σ est décrit par la liste $\langle \sigma(1), \dots, \sigma(n) \rangle$, puis déterminer le code de Lehmer de la permutation $\sigma = \langle 2, 1, 4, 5, 3 \rangle$.

b) Comment obtenir le nombre d'inversions d'une permutation à partir de son code de Lehmer ?

On considère le produit cartésien suivant :

$$K_n = [\![0, n-1]\!] \times [\![0, n-2]\!] \times \cdots \times [\![0, 1]\!] \times \{0\} \qquad \text{(c'est une partie de } \mathbb{N}^n\text{)}.$$

À un élément $(\gamma_1, \dots, \gamma_n)$ de K_n, on associe les listes d'entiers $\ell_n, \ell_{n-1}, \dots, \ell_1$ définies par :

$$\ell_n = \langle n \rangle ;$$
$$\text{si } \ell_{k+1} = \langle a_1, \dots, a_{n-k} \rangle, \text{ alors } \ell_k = \langle a_1, \dots, a_{\gamma_k}, k, a_{\gamma_k+1}, \dots, a_{n-k} \rangle.$$

Autrement dit, l'entier k est inséré après l'élément de rang γ_k.

Question 6.

a) Calculer les listes $\ell_5, \ell_4, \dots, \ell_1$ associées à $\gamma = (4, 2, 1, 1, 0)$.

b) Plus généralement, montrer que ℓ_1 est une permutation, et que $c(\ell_1) = (\gamma_1, \dots, \gamma_n)$.

c) En déduire que l'application c réalise une bijection entre $\mathfrak{S}_n$ et K_n.

Question 7.

a) Écrire une fonction CAML **insere** qui prend en paramètres un élément a, un entier j et une liste ℓ et qui retourne la liste obtenue en insérant a après l'élément de rang j de ℓ.

```
insere: 'a -> int -> 'a list -> 'a list
```

En cas d'échec on pourra par exemple déclencher une exception à l'aide de la commande `failwith "insere"`.

b) En déduire une fonction **genere** calculant la liste ℓ_1 définie à la question précédente à partir du n-uplet $(c_1,\dots,c_n)$ de K_n. On représentera c par un vecteur.

```
genere: int vect -> int list
```

Question 8. Soit $n \in \mathbb{N}^*$. On cherche à montrer que tout entier p de l'intervalle $[\![0, n!-1]\!]$ peut s'écrire de manière unique sous la forme :

$$p = \sum_{k=0}^{n-1} d_k k! \qquad \text{avec} \qquad \forall k \in [\![0, n-1]\!], \quad d_k \in [\![0, k]\!].$$

a) Montrer que pour tout $j \in \mathbb{N}^*$, $\displaystyle\sum_{k=0}^{j} k.k! = (j+1)! - 1$.

b) Soit $p \in [\![1, n!-1]\!]$, et j l'unique entier de $[\![0, n-1]\!]$ tel que $j! \leqslant p \leqslant (j+1)! - 1$.
Montrer qu'on a nécessairement $d_{j+1} = \cdots = d_{n-1} = 0$, puis que $d_j = \left\lfloor \dfrac{p}{j!} \right\rfloor$.

c) En déduire l'existence et l'unicité de la décomposition demandée.

Question 9. Les notations restent celles de la question précédente.

a) Rédiger une fonction **factorielle** qui a un entier associe sa factorielle.

```
factorielle: int -> int
```

b) On considère la suite $u_0, u_1, \dots, u_n$ définie par $u_0 = p$ et $u_j = \left\lfloor \dfrac{u_{j-1}}{j} \right\rfloor$ pour $j \in [\![1, n-1]\!]$.

Montrer que pour tout $j \in [\![0, n-1]\!]$, $d_j = u_j \mod (j+1)$, et en déduire une fonction **lehmer** qui prend en arguments deux entiers n et p et retourne le tableau $[|d_{n-1}; d_{n-2}; \cdots; d_1; d_0|]$.

```
lehmer: int -> int -> int vect
```

Question 10. L'*ordre* d'une permutation $\sigma \in \mathfrak{S}_n$ est l'entier $p = \displaystyle\sum_{k=0}^{n-1} c_{n-k}(\sigma) k!$.

Rédiger une fonction **permutations** qui prend en argument un entier n et retourne la liste de tous les éléments de $\mathfrak{S}_n$ rangée par ordre croissant.

```
permutations: int -> int list list
```

♦ ♦
♦