

Option informatique en Sup MPSI

Devoir en temps limité

vendredi 2 avril 2004

On rappelle que $\llbracket 1, n \rrbracket$ désigne l'ensemble des entiers strictement positifs et inférieurs ou égaux à n . En particulier, on remarquera que $\llbracket 1, 0 \rrbracket = \emptyset$.

Dans tout ce problème, on considérera un entier $n \in \mathbb{N}$, et on notera \mathcal{P}_n l'ensemble des parties de $\llbracket 1, n \rrbracket$ et \mathfrak{S}_n l'ensemble des bijections de $\llbracket 1, n \rrbracket$ dans lui-même.

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

1 Génération des parties d'un ensemble

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

Un élément de \mathcal{P}_n sera représenté en Caml par le type `int list`, l'ordre dans lequel apparaissent 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 \mathcal{P}_n sera représentée par le type `int list list`; par exemple, $\{\{1, 4\}, \{5\}, \{3, 4, 7\}\}$ sera représentée (entre autre) par la liste `[[4;3;7]; [4;1]; [5]]`.

Question 1.

À une partie A de $\llbracket 1, n \rrbracket$, on associe sa *fonction caractéristique* $\chi_A : \llbracket 1, n \rrbracket \rightarrow \{0, 1\}$ définie par :

$$\forall k \in \llbracket 1, n \rrbracket, \chi_A(k) = \begin{cases} 1 & \text{si } k \in A \\ 0 & \text{si } k \notin A \end{cases}$$

1.a Montrer, en exhibant l'application réciproque, que l'application ($A \mapsto \chi_A$) est une bijection de \mathcal{P}_n vers l'ensemble $\mathcal{F}(\llbracket 1, n \rrbracket, \{0, 1\})$ des fonctions de $\llbracket 1, n \rrbracket$ vers $\{0, 1\}$.

1.b En déduire que l'application $\varphi : \begin{pmatrix} \mathcal{P}_n & \longrightarrow & \llbracket 0, 2^n - 1 \rrbracket \\ A & \longmapsto & \sum_{k=1}^n \chi_A(k) 2^{k-1} \end{pmatrix}$ est une bijection.

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

Question 2.

Étant donné un élément k de $\llbracket 1, n \rrbracket$ et une partie \mathcal{B} de \mathcal{P}_n telle que pour tout $A \in \mathcal{B}$, $k \notin A$, on pose :

$$k \oplus \mathcal{B} = \{A \cup \{k\} / A \in \mathcal{B}\}$$

(on ajoute l'élément k à toutes les parties de $\llbracket 1, n \rrbracket$ qui appartiennent à \mathcal{B}).

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

2.a Écrire la fonction Caml `ajoute` correspondante, de type `'a -> 'a list list -> 'a list list`.

2.b Comment obtenir \mathcal{P}_n à partir de \mathcal{P}_{n-1} et de l'opérateur \oplus ?

En déduire une fonction `parties1` de type `int -> int list list` énumérant les éléments de \mathcal{P}_n par ordre croissant. On justifiera ce dernier point avec soin.

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

On propose maintenant une autre manière d'engendrer les éléments de \mathcal{P}_n . On définit une suite d'éléments de \mathcal{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 $j \geq 2$ et $k_1 < \dots < k_j$, 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.

- 3.a** Calculer la suite A_1, \dots, A_{16} lorsque $n = 4$.
- 3.b** En raisonnant par récurrence sur $n \in \mathbb{N}^*$, montrer que A_p est défini pour $p \in \llbracket 1, 2^n \rrbracket$, et que :

$$\mathcal{P}_n = \{A_p / p \in \llbracket 1, 2^n \rrbracket\}.$$

Question 4.

- 4.a** Écrire une fonction suivant de type `int list -> int list` calculant l'élément A_p à partir de A_{p-1} .
- 4.b** On suppose donnée une fonction `deux_puissance` qui prend un entier $n \in \mathbb{N}$ et qui retourne 2^n (on ne vous demande pas de l'écrire). Définir une fonction `parties2` de type `int -> int list list` qui calcule la liste des éléments de \mathcal{P}_n suivant le principe décrit ci-dessus.

2 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 $(\sigma(1), \dots, \sigma(n))$, et sera représenté par une liste.

Soit σ un élément de \mathfrak{S}_n . On dit que le couple $(i, j) \in \llbracket 1, n \rrbracket^2$ est une *inversion* de σ lorsque $i < j$ et $\sigma(j) < \sigma(i)$. On notera $\text{inv}(\sigma)$ le nombre d'inversions de σ .

Si $\sigma \in \mathfrak{S}_n$, on appelle *code de Lehmer* de σ le n -uplet $c(\sigma) = (c_1(\sigma), \dots, c_n(\sigma)) \in \mathbb{N}^n$ défini par :

$$\forall i \in \llbracket 1, n \rrbracket, c_i(\sigma) = \text{card}\{j \in \llbracket i+1, n \rrbracket / \sigma^{-1}(j) < \sigma^{-1}(i)\}.$$

Question 5.

- 5.a** Expliquer comment calculer $c_i(\sigma)$ lorsque σ est décrit par la liste $(\sigma(1), \dots, \sigma(n))$. Déterminer le code de Lehmer de la permutation $(2, 1, 4, 5, 3)$.
- 5.b** Comment obtenir le nombre d'inversions d'une permutation à partir de son code de Lehmer ?

Question 6.

On note $K_n = \llbracket 0, n-1 \rrbracket \times \llbracket 0, n-2 \rrbracket \times \dots \times \llbracket 0, 1 \rrbracket \times \{0\}$ (c'est une partie de \mathbb{N}^n). 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 :

$$\begin{aligned} \ell_n &= (n) ; \\ \text{si } \ell_{k+1} &= (a_1, \dots, a_{n-k}), \text{ alors } \ell_k = (a_1, \dots, a_{\gamma_k}, k, a_{\gamma_k+1}, \dots, a_{n-k}) ; \end{aligned}$$

(k est inséré après l'élément de rang γ_k).

- 6.a** Calculer les listes ℓ_1, \dots, ℓ_5 associées à $\gamma = (4, 2, 1, 1, 0)$.
- 6.b** Plus généralement, montrer que ℓ_1 est une permutation, et que $c(\ell_1) = (\gamma_1, \dots, \gamma_n)$. En déduire que l'application c réalise une bijection entre \mathfrak{S}_n et K_n .

Question 7.

- 7.a** Écrire une fonction Caml `insère` de type `'a list -> 'a -> int -> 'a list` qui prend en paramètres une liste ℓ , un élément a et un entier j , et qui retourne la liste obtenue en insérant a après l'élément de

rang j de ℓ .

- 7.b** Rédiger alors une fonction `gènère` de type `int vect -> int list` calculant la liste ℓ_n définie à la question précédente à partir du n -uple (c_1, \dots, c_n) . Vous remarquerez que c sera représenté par un vecteur.

Question 8.

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

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

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

Soit j l'unique entier de $\llbracket 1, n-1 \rrbracket$ tel que $j! \leq p \leq (j+1)! - 1$.

Montrer qu'on a nécessairement $d_{j+1} = \dots = d_{n-1} = 0$, puis que $d_j = \left\lfloor \frac{p}{j!} \right\rfloor$.

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

Question 9.

On donne une fonction `factorielle` de type `int -> int` qui prend en paramètre un entier n et qui retourne $n!$, et une fonction `lehmer` de type `int -> int -> int vect` qui prend en paramètres un entier n et un entier p et qui retourne le vecteur $\llbracket d_{n-1}; \dots; d_1; 0 \rrbracket$ (il n'est pas demandé d'écrire ces fonctions). Rédiger alors une fonction `permutations` de type `int -> int list list` calculant tous les éléments de \mathfrak{S}_n .

