Compositions et partitions d'un entier

Durée : libre

Définitions et notations

Une *composition* d'un entier $n \in \mathbb{N}^*$ est un k-uplet $a = (p_1, p_2, \ldots, p_k)$ avec $p_i \in \mathbb{N}^*$ pour $1 \leqslant i \leqslant k$ et $p_1 + p_2 + \cdots + p_k = n$; l'entier k est la *longueur* de la composition.

Une *partition* d'un entier $n \in \mathbb{N}^*$ est une composition $a = (p_1, p_2, \ldots, p_k)$ de n vérifiant la condition supplémentaire $p_1 \geqslant p_2 \geqslant \cdots \geqslant p_k$.

On définit l'ordre lexicographique inverse, noté $\succ$, sur les compositions (et les partitions) d'un entier : si $a = (p_1, p_2, \ldots, p_k)$ et $b = (q_1, q_2, \ldots, q_\ell)$ sont deux compositions de n, on convient que $a \succ b$ si et seulement s'il existe un entier $i \leqslant \min(k, \ell)$ tel que $p_j = q_j$ pour $1 \leqslant j \leqslant i-1$ et $p_i > q_i$.
Par exemple pour $n = 5$, $a = (1,2,1,1)$ est une composition mais pas une partition ; $b = (3,1,1)$, $c = (2,1,1,1)$ et $d = (2,2,1)$ sont des partitions, avec $b \succ d \succ c$.

Dans la suite du problème, on choisira de représenter en Caml une composition (et une partition) $a = (p_1, p_2, \ldots, p_k)$ par la liste (ordonnée en sens inverse) `[pk; ... ; p2; p1]`. Par exemple, la composition $a = (1,2,1,1)$ de 5 sera représentée par la liste `[1; 1; 2; 1]`.

Partie I. Compositions

Question 1.

a) Donner toutes les compositions de $n = 5$ dans l'ordre $\succ$.

b) Combien y-a-t-il de compositions de n de longueur k ?

c) Combien y-a-t-il de compositions de n ?

Question 2.

a) Étant donnée une composition $a = (p_1, p_2, \ldots, p_k)$ de n qui n'est pas la dernière pour l'ordre $\succ$, déterminer la composition suivante pour l'ordre $\succ$.

b) Rédiger une fonction `scinde` de type *int list –> int * (int list)* qui prend pour argument une liste ℓ et renvoie un couple (n, ℓ') où n est le nombre de 1 qui débutent la liste ℓ et ℓ' la liste des éléments qui suivent cette succession de 1. Par exemple, `scinde [1; 1; 2; 1]` devra renvoyer le couple `(2, [2; 1])`.

c) En déduire une fonction `composition_suivante` de type *int list –> int list* qui prend en argument la représentation d'une composition et renvoie la représentation de la composition suivante pour l'ordre $\succ$ si celle-ci existe, et la liste vide dans le cas contraire.

d) Définir alors une fonction `compositions` de type *int –> int list list* qui prend pour argument un entier n et qui renvoie la liste de toutes les représentations des compositions de l'entier n, dans un ordre quelconque. Chaque composition devra apparaître une et une seule fois dans cette liste.

Partie II. Partitions

Question 3. Donner toutes les partitions de $n = 7$ dans l'ordre $\succ$.

Question 4.

a) Étant donnée une partition $a = (p_1, p_2, \ldots, p_k)$ de n qui n'est pas la dernière pour l'ordre $\succ$, déterminer la partition suivante pour l'ordre $\succ$.

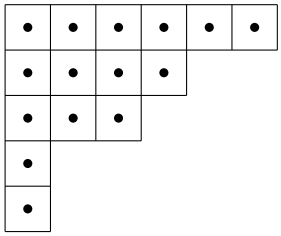
b) Rédiger une fonction `ajoute` de type *int –> 'a –> 'a list –> 'a list* qui prend en argument un élément x, un entier n et une liste ℓ et qui renvoie la liste ℓ à qui a été ajouté n fois l'élément x en tête de liste. Par exemple, `ajoute 3 1 [2; 3; 4]` devra renvoyer la liste `[1; 1; 1; 2; 3; 4]`.

c) En déduire une fonction `partition_suivante` de type *int list –> int list* qui prend en argument la représentation d'une partition et renvoie la représentation de la partition suivante pour l'ordre $\succ$ si celle-ci existe, et la liste vide dans le cas contraire.

d) Définir alors une fonction **`partitions`** de type *int –> int list list* qui prend pour argument un entier n et qui renvoie la liste de toutes les représentations des partitions de l'entier n, dans un ordre quelconque. Chaque partition devra apparaître une et une seule fois dans cette liste.

Partie III. Tableaux de Young

À toute partition $a = (p_1, p_2, \ldots, p_k)$ de n, on associe un diagramme $\mathrm{D}(a)$ de k lignes, où la ligne i comporte p_i cases pour $1 \leqslant i \leqslant k$. Noter que $\mathrm{D}(a)$ comprend n cases au total. Ainsi, pour la partition $a = (6,4,3,1,1)$ de $n = 15$, $\mathrm{D}(a)$ est égal au diagramme représenté ci-dessous à gauche :



1	3	5	9	12	16
2	6	10	15		
4	13	14			
11					
17					

On obtient un tableau de Young de taille n en remplissant les n cases de $\mathrm{D}(a)$ par des entiers distincts, avec la contrainte que tout élément d'une case est inférieur aux éléments des cases voisines à droite et en-dessous, si ceux-ci existent. Ci-dessus à droite figure un exemple de tableau de Young associé à la partition précédente.

Les tableaux de Young seront représentés en Caml par le type *int list list*. Par exemple, le tableau de Young ci-dessus sera représenté par la liste de listes suivante :

```
t = [ [1; 3; 5; 9; 12; 16]; [2; 6; 10; 15]; [4; 13; 14]; [11]; [17] ]
```

Question 5.

a) Rédiger une fonction **`taille`** de type *int list list –> int* qui prend en argument la représentation d'un tableau de Young et qui renvoie sa taille. Par exemple, **`taille t`** devra renvoyer l'entier 15.

b) Rédiger une fonction **`forme`** de type *int list list –> int list* qui prend en argument la représentation d'un tableau de Young et renvoie la représentation de la partition associée à son diagramme. Par exemple, **`forme t`** devra renvoyer la liste **`[1; 1; 3; 4; 6]`**.

Pour insérer un élément x dans un tableau de Young de taille n (qui ne contient pas déjà x), on utilise l'algorithme décrit informellement comme suit :

- si x est supérieur à tous les éléments de la première ligne, on crée une nouvelle case dans cette ligne contenant x ;
- sinon, soit y le plus petit élément de la première ligne supérieur à x ; on remplace y par x et on poursuit l'algorithme en insérant y dans la deuxième ligne ;
- l'algorithme se termine quand le dernier élément déplacé a été inséré, au besoin en créant une nouvelle ligne.

Ainsi, en insérant $x = 8$ dans la tableau de Young précédent, on obtient le tableau de Young suivant :

1	3	5	8	12	16
2	6	9	15		
4	10	14			
11	13				
17					

Question 6.

a) Expliquer pourquoi on obtient bien un tableau de Young de taille $n+1$ après l'insertion d'un élément dans un tableau de Young de taille n.

b) Rédiger une fonction **`insere`** de type *int –> int list list –> int list list* qui prend pour arguments un entier x et la représentation d'un tableau de Young t ne contenant pas x et qui renvoie la représentation du tableau de Young obtenu par insertion de x dans t.

c) Décrire de manière informelle l'algorithme inverse, qui supprime un élément présent dans un tableau de Young. La fonction Caml correspondante n'est pas demandée.