L246

SESSION 2003 - Second concours

INFORMATIQUE

Durée : 3 heures

Les calculatrices sont interdites.

Partitions et tableaux

Introduction

On étudie dans ce problème quelques résultats combinatoires simples liés aux partitions d'un entier et aux tableaux de Young. Il est recommandé de lire l'ensemble du sujet avant de commencer la rédaction. Les trois parties sont largement indépendantes.

Définitions et pseudo-programmes

- $\mathbb{N}$ désigne l'ensemble des entiers naturels, et $\mathbb{N}^*$ l'ensemble $\mathbb{N}$ privé de 0
- 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 $1 \leq i \leq k$ et $p_1 + p_2 + \ldots + p_k = n$; 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, avec la condition supplémentaire $p_1 \geq p_2 \geq \ldots \geq p_k$
- On définit l'ordre lexicographique inverse, noté $\rhd$, sur les compositions (et les partitions) d'un entier : si $a = (p_1,p_2,\ldots,p_k)$ et $a' = (p'_1,p'_2,\ldots,p'_{k'})$ sont deux compositions de n, on pose $a \rhd a'$ si et seulement s'il existe un entier $i \leq \min(k,k')$ tel que $p_j = p'_j$ pour $1 \leq j \leq i-1$ et $p_i > p'_i$

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 $b \rhd d \rhd c$.

Pour les questions qui demandent l'écriture d'un pseudo-programme : il s'agit d'exprimer votre algorithme dans un langage de votre choix, avec les structures de données (tableaux ou listes chaînées) et de contrôle (boucles, conditionnelles, ...) classiques.

Partie 1. Compositions

Question 1.1.

1. Donner toutes les compositions de $n = 5$ dans l'ordre $\rhd$
2. Combien y-a-t-il de compositions de n de longueur k ?
3. Combien y-a-t-il de compositions de n ?

Question 1.2.

1. Etant donnée $a = (p_1,p_2,\ldots,p_k)$ une composition de n qui n'est pas la dernière pour l'ordre $\rhd$, déterminer (en justifiant le résultat) la composition suivante pour l'ordre $\rhd$
2. Ecrire un pseudo-programme qui génère la composition suivante, pour l'ordre $\rhd$, d'une composition de n donnée ; préciser les structures de données utilisées

3. Quelle serait la complexité, en ordre de grandeur, d'un pseudo-programme qui génère toutes les compositions de n dans l'ordre $\triangleright$ (dire quelles opérations significatives sont prises en compte)?

Partie 2. Partitions

Question 2.1.

1. Donner toutes les partitions de $n = 7$ dans l'ordre $\triangleright$
2. Etant donnée $a = (p_1,p_2,\ldots,p_k)$ une partition de n qui n'est pas la dernière pour l'ordre $\triangleright$, déterminer (en justifiant le résultat) la partition suivante pour l'ordre $\triangleright$
3. Ecrire un pseudo-programme qui génère la partition suivante, pour l'ordre $\triangleright$, d'une partition de n donnée. Quelle est la complexité, en ordre de grandeur, de votre pseudo-programme?

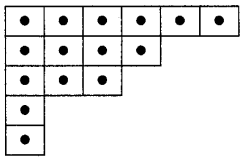
Question 2.2.

1. Caractériser la partition de n de longueur k qui est la dernière des partitions de n de longueur k pour l'ordre $\triangleright$ (i.e. telle qu'il n'existe pas de partition suivante de longueur k)
2. Etant donnée $a = (p_1,p_2,\ldots,p_k)$ une partition de n de longueur k, et qui n'est pas la dernière des partitions de n de longueur k pour l'ordre $\triangleright$, déterminer (en justifiant le résultat)) la partition de n longueur k qui est la suivante de a pour l'ordre $\triangleright$

Partie 3. Tableaux de Young

A toute partition $a = (p_1,p_2,\ldots,p_k)$ de n, on associe un diagramme $D(a)$ de k lignes, où la ligne i comporte p_i cases pour $1 \leq i \leq k$. Noter que $D(a)$ comprend n cases au total. Ainsi pour la partition $a = (6,4,3,1,1)$ de $n = 15$,

D(a) =



On obtient un tableau de Young (on dira simplement "Y-tableau" dans la suite) en remplissant les n cases de $D(a)$ par n 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; n est la taille du tableau. Voici un Y-tableau pour la partition précédente:

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

On va étudier l'insertion, puis la suppression, d'éléments dans un Y-tableau. Pour insérer un élément x dans un Y-tableau 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 continue 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 le Y-tableau précédent, on obtient le Y-tableau

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

Question 3.1.

1. Expliquer pourquoi on obtient bien un Y-tableau de taille $n+1$ après l'insertion d'un élément dans un Y-tableau de taille n
2. Ecrire un pseudo-programme qui réalise l'insertion d'un élément dans un Y-tableau de taille n

Question 3.2.

1. Proposer un algorithme pour supprimer un élément d'un Y-tableau de taille n
2. Ecrire un pseudo-programme qui réalise cet algorithme

Question 3.3.

1. Proposer un algorithme pour supprimer le plus petit élément d'un Y-tableau de taille n
2. Peut-on imaginer un algorithme de tri basé sur les Y-tableaux?

Question 3.4.

Dans cette dernière question, on s'intéresse au nombre $f(p_1,p_2,\ldots,p_k)$ de Y-tableaux de taille n, de forme $(p_1,p_2,\ldots,p_k)$ avec $p_1 + p_2 + \ldots + p_k = n$, et comprenant tous les entiers entre 1 et n. On posera $f(p_1,p_2,\ldots,p_k) = 0$ si on n'a pas $p_1 \geq p_2 \geq \ldots \geq p_k \geq 0$, et $f(p_1,p_2,\ldots,p_k,0) = f(p_1,p_2,\ldots,p_k)$.

1. Montrer la récurrence :

$$f(p_1,p_2,\ldots,p_k) = f(p_1-1,p_2,\ldots,p_k) + f(p_1,p_2-1,\ldots,p_k) + \ldots + f(p_1,p_2,\ldots,p_k-1)$$

si $p_1 \geq p_2 \geq \ldots \geq p_k \geq 1$

2. Soit $\Delta(x_1,x_2,\ldots,x_k) = \det \begin{vmatrix} x_1^{k-1} & x_2^{k-1} & \ldots & x_k^{k-1} \\ \vdots & & & \vdots \\ x_1^2 & x_2^2 & & x_k^2 \\ x_1 & x_2 & & x_k \\ 1 & 1 & \ldots & 1 \end{vmatrix} = \Pi_{1\leq i<j\leq k}(x_i - x_j)$ (on admettra la valeur de Δ, appelé déterminant de Vandermonde). Soit g la fonction de $k+1$ variables définie par

$g(x_1,x_2,\ldots,x_k,y) = x_1.\Delta(x_1+y,x_2,\ldots,x_k)+x_2.\Delta(x_1,x_2+y,\ldots,x_k)+\ldots+x_k.\Delta(x_1,x_2,\ldots,x_k+y).$

Montrer que

$$g(x_1,x_2,\ldots,x_k,y) = (x_1 + x_2 + \ldots + x_k + \frac{k(k-1)y}{2}).\Delta(x_1,x_2,\ldots,x_k)$$

3. En déduire par récurrence la formule :

$$f(p_1,p_2,\ldots,p_k) = \frac{\Delta(p_1+k-1,p_2+k-2,\ldots,p_k)n!}{(p_1+k-1)!(p_2+k-2)!\ldots p_k!}$$

pour $p_1 + k - 1 \geq p_2 + k - 2 \geq \ldots \geq p_k$