Composition Hors Classement d'Informatique

Jean-Jacques Lévy Yves Robert

20 décembre 2000 – 2h

Avertissement On attachera une grande importance à la clarté, à la précision, à la concision de la rédaction. Les sections A et B sont indépendantes.

A. Décompositions d'entiers

Soit $n \geq 0$ un entier fixé. Une décomposition de n est un k-uplet d'entiers $(a_1,a_2,\ldots,a_k)$ tel que $a_i \geq 1$ pour $1 \leq i \leq k$ et $a_1 + a_2 + \ldots + a_k = n$. On ordonne les décompositions additives de n par l'ordre lexicographique. Par exemple si $n = 10$,

$$(2,5,1,1,1) < (2,5,2,1) < (3,7) < (4,1,1,1,1,2).$$

Formellement, l'ordre lexicographique (l'ordre du dictionnaire) est défini de la manière suivante: $(a_1,a_2,\ldots a_k) < (b_1,b_2,\ldots b_\ell)$ ssi $a_i = b_i$ pour $1 \leq i \leq m$ pour un m tel que $0 \leq m \leq k$ et soit $m = k < \ell$, soit $a_{m+1} < b_{m+1}$. Noter que cet ordre est total.

On représente les décompositions sous forme de listes chaînées contenant successivement les valeurs $a_1,a_2,\ldots,a_k$.

Question 1 Préciser la structure de donnée sous forme d'une classe `ListeDecomp` en Java. Ecrire une fonction `static boolean estDecomp(ListeDecomp d, int n)` qui vérifie si la liste d représente bien une décomposition de n.

Question 2 Donner toutes les décompositions du nombre 5 dans l'ordre lexicographique. Combien de chiffres sont modifiés dans une décomposition pour obtenir la suivante?

Question 3 Ecrire la fonction `static boolean estMax(ListeDecomp d, int n)` qui vérifie si une liste d est une décomposition de n sans successeur dans l'ordre lexicographique.

Question 4 Etant donnée une décomposition d de n qui n'est pas la plus grande, écrire la fonction `static ListeDecomp suivante(ListeDecomp d)` qui renvoie la décomposition suivante de d (dans l'ordre lexicographique).

B. Partage équilibré de périmètre

Soit un polygone dont la suite des longueurs des côtés $\langle a_0,a_1,a_2,\ldots,a_{n-1}\rangle$ est stockée dans un tableau unidimensionnel d'entiers de taille $n \geq 2$. Soit une paire d'indices (i,j), avec $0 \leq i < n$ et $0 \leq j < n$; ces indices déterminent deux portions de périmètres, l'une de longueur $\sum_{k=i}^{j-1} a_k$ et l'autre de longueur $\sum_{k=j}^{i-1} a_k$, où tous les indices sont pris modulo n. On cherche la (en fait, une) paire d'indices qui minimise la valeur absolue de la différence des deux portions de périmètre qu'ils déterminent, i.e.

$$\min_{0 \leq i,j < n} \Delta(i,j) \text{ avec } \Delta(i,j) = \left| \sum_{k=i}^{j-1} a_k \; - \; \sum_{k=j}^{i-1} a_k \right|$$

Le polygone est représenté par un tableau `int[ ] a` donnant les longueurs de tous ses cotés.

Nous allons construire des solutions de plus en plus efficaces. Le coût d'une solution est défini comme le nombre d'opérations élémentaires (addition d'entiers, accès à un tableau, etc) qu'elle effectue. (Rappel: en Java, l'expression x modulo n s'écrit `x % n`.)

Question 5 Quel serait l'ordre de grandeur $O(n^x)$ du coût de la méthode suivante, trop naïve pour que nous l'écrivions en Java: pour toutes les paires (i,j), calculer la différence $\Delta(i,j)$ et prendre le minimum?

Question 6 Ecrire une fonction `static int perimetre(int[] a)` qui renvoie le périmètre total du polygone décrit par a.

Question 7 Pour un indice i donné, soit $d(i)$ le premier indice tel que $\sum_{k=i}^{d(i)} a_k > p/2$, où p est le périmètre (et les indices sont toujours pris modulo n).

a) Montrer que l'indice j tel que $\Delta(i,j) = \min_{0 \leq j < n} \Delta(i,j)$ est égal soit à $d(i)$ soit à $d(i)+1$.

b) Ecrire une fonction `static void partage1(int[] a)` qui calcule et imprime la solution optimale $(i,j,\Delta(i,j))$. On veut que le coût de la fonction `partage1(a)` soit en $O(n^2)$.

Question 8 On demande maintenant d'écrire une fonction `static void partage2(int[] a)` similaire à la fonction précédente mais dont le coût est en $O(n)$, i.e. linéaire en n. Il est impératif de justifier le principe de votre algorithme avant d'écrire la fonction.