ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Sections d'Informatique et de Systèmes de Communication

Série d'exercices 6 31 Oct. 2011

1. Multiplication de plusieurs matrices

Supposons que nous avons 4 matrices aux formats suivants:

Matrice	Format
A_1	5×10
A_2	10×3
A_3	3×12
A_4	12×5

Utiliser un algorithme de programmation dynamique approprié vu en cours pour calculer la disposition de parenthèses de l'expression $A_1 \cdots A_4$ qui minimise le nombre de multiplications nécessaires.

2. Algorithmes

Considérons le problème suivant: étant donné une suite triée (croissante) d'éléments de $\mathbb{N}$ et un nombre $x \in \mathbb{N}$, déterminer s'il existe dans la suite deux éléments distincts dont la somme vaut x.

Par exemple pour l'input $\big((1,2,2,3,3,10,11),\ 6\big)$ l'output serait *vrai*, alors que pour l'input $\big((1,1,1,2,5,6,6,7),\ 4\big)$ l'output serait *faux*.

a) Quel est le temps de parcours de l'algorithme naïf pour ce problème?

b) Supposons que la suite est stockée dans un tableau a de taille n. Donner (en pseudo code) un algorithme qui résout ce problème en temps $O(n)$.

3. Sac à dos 0/1

a) Trouver une solution optimale (i.e. objets et valeur optimale) au problème du sac à dos 0/1 suivant:

$W = 20,$

objet	A	B	C	D	E
valeur	22	15	8	15	15
poids	11	7	6	7	3

b) La solution optimale dans le problème ci-dessus est-elle unique? Si ce n'est pas le cas, donner toutes les solutions optimales. Peut-on adapter l'algorithme du point a) pour trouver toutes les solutions optimales? Comment faire?

c) Montrer que si on a un problème de sac à dos avec n objets, alors il peut y avoir au plus 2^n solutions optimales.

d) Montrer que pour tout n pair, il y a un choix de W, des v_i et des w_i tel qu'il y a au moins $2^{n/2}$ solutions optimales.

4. La suite de Fibonacci

Le but de cet exercice est de développer un algorithme rapide pour calculer la suite de Fibonacci en utilisant des techniques vues dans le cours et aux exercises.

Rappel: La suite de Fibonacci est définie par la récursion suivante:

$$F_1 = 1, \qquad F_2 = 1, \qquad F_n = F_{n-1} + F_{n-2} \ \ \forall n \geq 2.$$

a) Considérer l'algorithme standard pour calculer la suite de Fibonacci:

Call: FIB(n)
Input: n: L'indice de l'élément de la suite à calculer.
Output: La valeur de F_n.

$a_1 \leftarrow 1$
$a_2 \leftarrow 1$
while $n \geq 3$ **do**
  $a_3 \leftarrow a_2 + a_1$
  $a_1 \leftarrow a_2$
  $a_2 \leftarrow a_3$
  $n \leftarrow n - 1$
return a_2

Calculer l'ordre du temps de parcours de cet algorithme (en fonction de n) en supposant que l'addition de deux nombres à N bits se fait en $O(N)$. (*Indication:* Utiliser directement le fait que $F_n \approx \varphi^n$.)

b) Considérer la suite vectorielle suivante:

$$\vec{a}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \vec{a}_n = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \vec{a}_{n-1} \ \ \forall n \geq 1.$$

(i) Montrer que pour tout $n \in \mathbb{N}$, on a

$$\vec{a}_n = \begin{pmatrix} F_{n+1} \\ F_{n+2} \end{pmatrix}.$$

(ii) Donner une formule fermée pour la suite $(\vec{a}_n)$.

c) En utilisant la suite vectorielle ci-dessus en combinaison avec des algorithmes rapides pour calculer des puissances de nombres déduire un algorithme rapide pour calculer un élément de la suite de Fibonacci. Calculer l'ordre du temps de parcours de cet algorithme (en fonction de n) en supposant qu'une multiplication d'un nombre de N bits coute $O(N \log(N))$.

d) Est-il possible de trouver un algorithme qui calcule la suite de Fibonacci en $o(n)$? Justifiez votre réponse.