TP8 : recherche de motifs et programmation dynamique

AP3 Algorithmique et programmation — L2 mathématiques

1 décembre 2016

Exercice 1 (Plus long sous-suite commune)**.**

(a) Soit Σ un alphabet fini. Une *sous-suite commune* à deux mots $u, v \in \Sigma^*$ est une suite $w \in \Sigma^*$, tel qu'il existe deux suites croissantes d'indices $i_1 < i_2 < \ldots < i_{|w|}$ et $j_1 < j_2 < \ldots < j_{|w|}$ vérifiant

$$w_k = u_{i_k} = v_{j_k} \text{ pour } 1 \leqslant k \leqslant |w| \,.$$

Le problème de *la plus longue sous-suite commune* (LCS) consiste à déterminer une sous-suite de longueur maximale commune à u et v.

Par la suite, A sera un tableau à deux dimensions indexé par $\{0, \ldots, \text{len(u)}\} \times \{0, \ldots, \text{len(v)}\}$ et $A[i, j]$ désignera la longueur de la plus longue sous-suite commune à $u[1, \ldots, i]$ et $v[1, \ldots, j]$.

On peut prouver (cf cours) que le tableau A vérifie la propriété suivante :

Si $A[0, j] = A[i, 0] = 0$ alors pour $0 \leqslant i < n$ et $0 \leqslant j < m$:

$$A[i+1, j+1] = \max \begin{cases} A[i, j] + \mathbf{1}_{u[i+1]=v[j+1]} \\ A[i+1, j] \\ A[i, j+1] \end{cases}$$

Écrire une fonction python `TabDyn(u,v)` renvoyant un tableau A comme ci-dessus i.e. tel que, pour tout $i \leqslant n$ et $j \leqslant m$, $A[i, j]$ contient la longueur de la plus longue sous-suite commune à $u[1, \ldots, i]$ et $v[1, \ldots, j]$.

(b) Écrire une fonction python `LCS(u,v)` renvoyant une plus longue sous-suite commune à u et v.

(c) Déterminer la complexité de cet algorithme.

Exercice 2 (Produit de matrices)**.** On rappelle que le produit de matrices est associatif, c'est-à-dire que $(AB)C = A(BC)$. Pour faire le produit de matrices $A_1 A_2 \ldots A_n$, il existe de nombreuses façons de parenthéser correctement cette expression. Le but de cet exercice est de trouver le parenthésage optimal d'un produit de matrices $A_1 A_2 \ldots A_n$ minimisant le nombre de multiplications scalaires.

(a) Combien de multiplications d'entiers faut-il faire pour calculer

$$(3) \times ((4) \times ((2) \times (6 \quad 7 \quad 8 \quad 9))) \quad ?$$

Et

$$(((3) \times (4)) \times (2)) \times (6 \quad 7 \quad 8 \quad 9) \quad ?$$

(b) À partir de maintenant, on suppose avoir n matrices $A_1, \dots, A_n$ multipliables, c'est-à-dire que : A_1 a dimension $p_0 \times p_1$, A_2 a dimension $p_1 \times p_2$, etc. jusque A_n qui a dimension $p_{n-1} \times p_n$. Remarquer que les coefficients des A_i importent peu et que seul les p_i sont nécessaires pour résoudre le problème du parenthésage optimal.

(c) Montrer que le nombre de parenthésages de $A_1 A_2 \dots A_n$ est supérieur ou égal à 2^{n-2}. Est-il efficace de tester tous les parenthésages possibles ?

(d) Pour calculer le parenthésage optimal, on va utiliser la méthode de programmation dynamique.

Notons $m_{i,j}$ ($1 \leq i \leqslant j \leqslant n$) le nombre minimal de multiplications scalaires à effectuer pour calculer $A_i \dots A_j$. Démontrer que les $m_{i,j}$ sont définies par les relations suivantes :

$$m_{i,j} = \begin{cases} 0 & \text{si } i = j \\ \min_{i \leqslant k < j} \left(m_{i,k} + m_{k+1,j} + p_{i-1} p_k p_j \right) & \text{sinon} \end{cases}$$

(e) Écrire une fonction `matrixmult(p)` prenant en paramètre la liste p des dimensions p_i, $0 \leqslant i \leqslant n$ et renvoyant le nombre minimal de multiplications scalaires à effectuer pour calculer le produit $A_1 \dots A_n$.

(f) Modifier la fonction précédente en une fonction `matrixmult_ext(p)` prenant le même paramètre et renvoyant le couple $(m_{1,n}, s)$ où $m_{1,n}$ est comme avant et s est la chaîne de caractères comportant le parenthésage des A_i. Par exemple, pour les matrices de la question 2.(a), la fonction doit retourner :

```
(6, '((((A1)A2)A3)A4)')
```