Optimisation Combinatoire - Travaux Dirigés n^o 4

— *Maîtrise d'Informatique* —

Programmation dynamique

La *programmation dynamique*, comme la méthode diviser pour régner, résout les problèmes en combinant les solutions des sous-problèmes. Dans la méthode diviser pour régner, les sous-problèmes sont résolus de façon récursive et la méthode est donc adaptée lorsque les sous-problèmes sont indépendants. La programmation dynamique est en revanche applicable lorsque les sous-problèmes ne sont pas indépendants, c'est-à-dire lorsqu'ils partagent des sous-sous-problèmes.

1 Plus longue sous-chaîne commune et variantes

Une sous-chaîne d'une chaîne de caractères est une chaîne qui peut être obtenue en effaçant certains caractères de la chaîne initiale. Plus formellement, une chaîne $u = b_1 \dots b_m$ est dite *sous-chaîne* d'une chaîne $w = a_1 \dots a_k$ s'il existe une suite $i_1, \dots, i_m$ de m positions dans w telle que $b_j = a_{i_j}$ pour tout $1 \leq j \leq m$.

Voici un exemple avec $n = 9$ et $k = 4$. Les quatre positions sont données par $i_1 = 2$, $i_2 = 4$, $i_3 = 5$ et $i_4 = 8$. La chaîne $u = babb$ est donc une sous-chaîne de $w = abaabbabb$. On aurait pu prendre $i_1 = 6$, $i_2 = 7$, $i_3 = 8$ et $i_4 = 9$.

$$\begin{array}{ccccccccccc}
 & & & i_1 & & i_2 & i_4 & & & i_4 & \\
 & & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
w & = & a & b & a & a & b & b & a & b & b \\
u & = & & b & & a & b & & & b &
\end{array}$$

Étant données deux chaînes v et w, le problème est de calculer la plus longue chaîne u qui soit simultanément sous-chaîne de v et de w. On suppose que v et w sont de longueur m et n. Pour $0 \leq i \leq m$ et $0 \leq j \leq n$, on définit $m_{i,j}$ comme la longueur de la plus longue sous-chaîne commune au préfixe de longueur i de v et au préfixe de longueur j de w. Par convention, on pose $m_{0,j} = 0$ et $m_{i,0} = 0$.

▶ **Exercice 1** Exprimer $m_{i,j}$ en fonction des trois quantités $m_{i-1,j-1}$, $m_{i,j-1}$ et $m_{i-1,j}$. En déduire un algorithme pour calculer les mi, j. Comment retrouver la plus longue sous-chaîne commune à v et w à partir des valeurs de $m_{i,j}$?

Un problème classique en bio-informatique est de trouver une mesure de similarité entre deux séquences. Pour cela, on définit un alignement de deux séquences v et w comme une paire (v', w') de deux séquences de même longueur sur l'alphabet augmenté du symbole '$-$' telle qu'on retrouve v et w en effaçant les symboles '$-$' de v' et w'.

Par exemple si $v = AGCGTC$ et $w = GATCTC$, un alignement peut être

$$\begin{array}{cccccccccc}
v' & = & A & G & - & - & C & G & T & C \\
w' & = & - & G & A & T & C & - & T & C
\end{array}$$

À chaque alignement est associé un *score* calculé de la manière suivante. On commence par choisir les scores σ des paires de lettres en prenant par exemple

$$\sigma(x, x) = +2 \qquad \sigma(x, y) = -1 \qquad \sigma(x, -) = -2$$

Si m' est la longueur commune à v' et w', le score de l'alignement est alors la somme $\sum_{i=1}^{m'} \sigma(v'_i, w'_i)$. La mesure de similarité entre v et w est alors le score maximal d'un alignement entre des chaînes v' et w'.

▶ **Exercice 2** Donner une méthode pour calculer la mesure de similarité de deux chaînes.

2 Choix d'activités

On reprend le problème du choix d'activités (cf. feuille de Td n°3). On suppose que les activités ont été triées par ordre d'horaire de fin croissant :

$$f_1 \leq f_2 \leq \cdots \leq f_n.$$

On note m_i la taille du plus grand ensemble d'activités mutuellement compatibles parmi les activités $\{1, 2, \ldots, i\}$.

▶ **Exercice 3** Donner un algorithme de programmation dynamique pour le calcul itératif des m_i.

3 Problème du havresac

On dispose de n objets pour remplir un *havresac*. Chaque objet i a un poids w_i et une valeur v_i et le havresac a une capacité maximale de W kilos. Il s'agit de remplir le havresac avec des objets avec une valeur totale maximale mais sans pour autant dépasser la capacité du havresac.

Soient k et w des entiers tels que $k \leq n$ et $w \leq W$. On considère le problème $P_{k,w}$ du havresac où on ne prend en compte que les k premiers objets et où on limite la taille du havresac à w kilos.

▶ **Exercice 4** Trouver une relation entre les $P_{k,w}$. En déduire un algorithme de programmation dynamique permettant de les calculer. Quelle est la complexité de l'algorithme ?

4 Produits de matrices

▶ **Exercice 5** Soit n un entier positif et soient P, Q et R trois matrices de taille $2 \times n$, $n \times 2$ et $2 \times n$. Pour calculer le produit PQR, on calcule d'abord le produit PQ puis on le multiplie avec R. Quel est le nombre de multiplications effectuées ? Quel est ce nombre si l'on calcule d'abord le produit QR puis le produit avec P.

Soient n matrices $M_1, \ldots, M_n$. On note l_i et c_i le nombre de lignes et de colonnes de la matrice M_i pour $1 \leq i \leq n$. On suppose que les dimensions des matrices sont compatibles pour que le produit $M_1 \ldots M_n$ existe. Ceci signifie que $c_i = l_{i+1}$ pour $1 \leq i \leq n - 1$.

Le problème est de trouver un parenthésage du produit $M_1 \ldots M_n$ qui minimise le nombre de multiplications effectuées au cours du calcul. Pour $1 \leq i \leq j \leq n$, on note $w_{i,j}$ le nombre minimal de multiplications nécessaires au calcul du produit partiel $M_i \ldots M_j$. Par convention, on pose $w_{i,j} = 0$ si $i = j$.

▶ **Exercice 6** Donne une méthode de programmation dynamique pour calculer les valeurs $w_{i,j}$. Quelle est la complexité de cette méthode ? Quelle serait la complexité d'un parcours de tous les parenthésages possibles.

5 Sous-suites d'une suite d'entiers

▶ **Exercice 7** Soit T un tableau contenant des entiers relatifs. Donner un algorithme permettant de calculer la somme maximale d'une suite d'éléments consécutifs de T.

▶ **Exercice 8** Soit T un tableau contenant des entiers relatifs. Donner un algorithme permettant de calculer la longueur maximale d'une sous-suite croissante d'éléments (non nécessairement consécutifs) de T.