

## TD n°5

### Programmation dynamique

#### 1 Mise en page et impression d'un texte

On dispose d'un fichier texte sans passage a la ligne (il s'agit d'un seul paragraphe) que l'on veut imprimer sur une feuille de taille donnee. Chaque caractere imprime occupe la même largeur, y compris les espaces, comme sur une machine a ecrire mecanique. Le probleme est de decouper le texte en lignes, pour minimiser les espaces qui restent a la fin de chaque ligne.

Formellement, le texte est une suite de  $n$  mots de longueurs  $l_1, l_2, \dots, l_n$ , mesurees en nombre de caracteres. Chaque ligne contient  $M$  caracteres, blancs compris. Si une ligne contient les mots  $i$  a  $j$  (inclus), ou  $i \leq j$ , et qu'on laisse exactement un espace entre deux mots, le nombre de caracteres blancs a la fin de la ligne est

$$M - j + i - \sum_{k=i}^j l_k$$

On veut minimiser la somme, sur toutes les lignes hormis la derniere, des cubes des nombres de caracteres blancs presents a la fin de la ligne. On dira alors que la presentation du texte est « equilibree ».

**Exercice 1** Donner la solution optimale pour des lignes de longueur 17 avec le texte suivant :

La programmation dynamique est une des bases algorithmiques

**Exercice 2** Montrez que l'algorithme naïf (glouton) echoue.

**Exercice 3** Donner un algorithme de programmation dynamique qui resout le probleme.

**Exercice 4** Donner la complexite en temps et en espace de l'algorithme precedent.

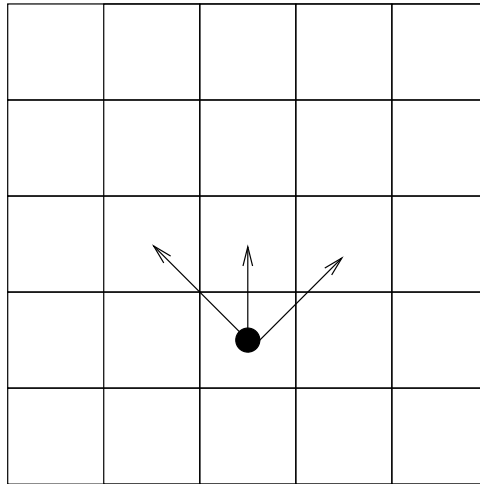
#### 2 Un jeu de dames...

On dispose d'un damier de dimension  $n \times n$ . A chaque case  $(i, j)$  est affecte un cout  $c_{i,j}$ . On veut deplacer un pion sur le damier depuis n'importe quelle case de la rangee du bas ( $i = 1$ ) vers n'importe quelle case de la rangee du haut ( $i = n$ ). A chaque fois que l'on passe par une case, on paye le coût de cette case. Le but est donc de trouver le chemin de coût minimal, sachant que le pion situe sur la case  $(i, j)$  ne peut aller en un coup que sur les cases  $(i + 1, j - 1)$ ,  $(i + 1, j)$  ou  $(i + 1, j + 1)$ . Comme sur le dessin.

**Exercice 5** Trouvez une relation de recurrence pour calculer ce qu'il coûte d'atteindre la case  $(i, j)$ . Comment gerer les "eets de bords" de maniere pratique pour que cette relation reste vraie même pour  $j = 1$  ou  $j = n$  ?

**Exercice 6** En deduire un algorithme pour resoudre le probleme. L'algorithme doit chercher non seulement le coût minimal mais aussi le chemin a suivre.

**Exercice 7** Quelle est la complexite de cet algorithme ?



### 3 Plus longue sous-séquence commune

On considère deux séquences (textes formés de lettres)  $S$  et  $T$ . Leurs longueurs sont notées respectivement  $m$  et  $n$ . Une **sous-séquence** d'un texte est obtenue en enlevant certaines lettres du texte. Par exemple **algorithmique** a comme sous-séquence **gothique**. Une sous-séquence commune à  $S$  et  $T$  est une séquence qui est sous-séquence des deux à la fois. On s'intéresse à trouver la plus longue sous-séquence commune (PLSSC), c'est-à-dire celle avec le plus de lettres.

Pour cela on construit un tableau  $LPLSSC[i][j]$  (avec  $1 \leq i \leq m$  et  $1 \leq j \leq n$ ) tel que  $LPLSSC[i][j]$  est la **longueur** de la plus longue sous-séquence

- { commune à  $S$  et  $T$
- { terminant avant la  $i$ ème lettre de  $S$  (inclusivement)
- { terminant avant la  $j$ ème lettre de  $T$  (inclusivement)

Par exemple si  $S = \text{algorithmique}$  et  $T = \text{programmation}$  on a  $LPLSSC[4][3] = 1$  (o est PLSSC de **algo** et **prog**) et  $LPLSSC[4][3] = 1$  (o est PLSSC de **algo** et **prog**) et  $LPLSSC[13][13] = 4$  avec **orti**

**Exercice 8** Notons  $Q = Q[1] \dots Q[k]$  une plus longue sous-séquence commune à  $S$  et  $T$ . Montrer que

- { Si  $T[m] = S[n]$  alors  $Q[k] = S[m] = T[n]$  et  $Q[1..k-1]$  est la plus longue sous-séquence commune à  $S[1..m-1]$  et à  $T[1..n-1]$ ;
- { Si  $S[m] \neq T[n]$  et  $Q[k] \neq S[m]$  alors  $Q$  est une plus longue sous-séquence commune à  $S[1..m-1]$  et  $T$ ;
- { Si  $S[m] \neq T[n]$  et  $Q[k] \neq T[n]$  alors  $Q$  est une plus longue sous-séquence commune à  $S$  et  $T[1..n-1]$

**Exercice 9** En déduire la formule de récurrence générale qui permet de calculer  $LPLSSC[i][j]$  en fonction de  $LPLSSC[i-1][j-1]$ ,  $LPLSSC[i][j-1]$  et  $LPLSSC[i-1][j]$ . Ne pas oublier de donner son initialisation).

**Exercice 10** Écrire le programme de programmation dynamique qui, étant donné deux séquences, calcule la longueur de d'une plus longue sous-séquence commune.

**Exercice 11** Comment retrouver *une* longue(s) sous-séquence commune en mémorisant des informations complémentaires?

## 4 Plus longue sous-séquence croissante

Les objets sur lesquels on travaille maintenant sont des sequences de nombres entiers. Il y a au moins deux nombres par sequence. Par exemple, une telle sequence (de longueur 9) est :  $T = (11, 5, 2, 8, 7, 3, 1, 6, 4)$ . On note de maniere generale  $S = (s_1, \dots, s_i, \dots, s_n)$  une sequence de longueur  $n > 1$ .

On appelle **sous-séquence croissante** (SSC) de  $S$  une sous-sequence dont :

- { les elements sont pris de gauche a droite dans  $S$  ;
- { les elements croissent strictement de gauche a droite.

On appelle **intervalle croissant** (IC) une sous-sequence croissante sans trou (contigue)

Par exemple, (2,3,4) et (1,6) sont des SSC de  $T = (11, 5, 2, 8, 7, 3, 1, 6, 4)$  et (1,6) est de plus un IC.

Le but de ces exercices est de trouver la longueur des plus longs IC et SSC d'une sequence quelconque  $S$ , notes respectivement  $LIC(S)$  et  $LSSC(S)$ .

Par exemple, les plus longs IC de  $T$  sont (2, 8) et (1, 6), donc  $LIC(T) = 2$ . Les plus longues SSC de  $T$  sont (2, 3, 4) et (2, 3, 6), donc  $LSSC(T) = 3$ .

L'operation elementaire de mesure de la complexite est la comparaison.

**Exercice 12** Montrer que, pour toute sequence  $S$ , on a :  $LSSC(S) \geq LIC(S)$ .

**Exercice 13** Donner le principe d'un algorithme en  $O(n)$  pour calculer  $LIC(S)$ .

**Exercice 14** Pourquoi un algorithme comme le precedent ne permet-il pas de calculer  $LSSC(S)$  avec un algorithme en  $O(n)$  ?

**Exercice 15** Pour calculer  $LSSC(S)$ , on va remplir un tableau a deux lignes et  $n$  colonnes. La premiere ligne contient  $s_i$  (l'element de rang  $i$  de  $S$ ), la seconde  $l_i$ , qui est par definition la longueur de la plus longue sous-sequence de  $S$  dont le dernier element est  $s_i$ . Dans notre exemple :

$i$	1	2	3	4	5	6	7	8	9
$s_i$	11	5	2	8	7	3	1	6	4
$l_i$	1	1	1	2	2	2	1	3	3

Montrer et expliquer la relation de recurrence suivante. Quelle valeur faut-il donner au terme Max quand on n'a aucun  $j$  tel que  $0 < j < i$  pour lequel  $s_j < s_i$  ?

$$l_1 = 1$$

$$l_i = 1 + \text{Max}_{\substack{0 < j < i \\ s_j < s_i}} l_j$$

**Exercice 16** Ecrire le programme qui calcule  $LSSC(S)$  pour toute sequence  $S$  de longueur  $n$ . Quel est son ordre de grandeur de complexite ?