Correction TD 6 : Plus longue sous-séquence commune

Pascal Vanier

April 24, 2013

Question 1 Soient deux séquences $X_m = (x_1, \ldots, x_m)$ et $Y_n = (y_1, \ldots, y_n)$, on a alors plusieurs cas de figure :

- Si $x_m = y_n$, alors le problème revient à trouver la LCS de $X_{m-1} = (x_1, \ldots, x_{m-1})$ et $Y_{n-1} = (y_1, \ldots, y_{n-1})$.
- Sinon, la dernière lettre de la LCS ne sera **pas** : soit x_m, soit y_n, et le problème revient donc à choisir la plus longue des chaînes parmi $LCS(X_{m-1}, Y_n)$ et $LCS(X_m, Y_{n-1})$.

Notez que l'on aurait pu faire le même raisonnement sur les premières lettres au lieu des dernières, cela n'aurait rien changé.

Question 2 On peut établir la relation de récurrence suivante :

$$LCS\left(X_m, Y_n\right) = \begin{cases} 0 & \text{si } \ n = 0 \text{ ou } m = 0 \\ LCS\left(X_{m-1}, Y_{n-1}\right) + 1 & \text{si } x_m = y_n \\ \max\left(LCS\left(X_m, Y_{n-1}\right), LCS\left(X_{m-1}, Y_n\right)\right) & \text{si } x_m \neq y_n \end{cases}$$

Question 3 Comptons le nombre de sous-problèmes possibles pour deux chaines de longueur m et n : comme on a m séquences $X_1, \ldots, X_m$ et n séquences $Y_1, \ldots, Y_n$, on a au plus nm sous-problèmes.

Question 4 Traitons tout d'abord l'exemple, on commence donc par établir la table, vide au départ que l'on peut remplir ligne par ligne :

	.	C	A	T	G	T
.	0	0	0	0	0	0
A	0	0	1	1	1	1
C	0	1	1	1	1	1
G	0	1	1	1	2	2
C	0	1	1	1	2	2
T	0	1	1	2	2	3
G	0	1	1	2	3	3

La plus longue sous-séquence commune aux deux chaînes est donc de longueur 3. Donnons maintenant l'algorithme permettant de remplir la table :

```
Construire_table(chaine x, chaine y):
-- m := longueur(x), n := longueur(y)
-- table := nouvelle_table(m+1,n+1)
-- pour i=0 jusqu'à m faire
---- table[i] := 0
-- pour j=0 jusqu'à n faire
---- table[j] := 0
-- pour i=1 jusqu'à n faire
---- pour j=1 jusqu'à m faire
```

```
------ si x[i] == y[j] alors
-------- table[i,j] := table[i-1,j-1]+1
------ sinon
-------- table[i,j] := max(table[i-1,j],table[i,j-1])
-- retourner table
```

Cet algorithme est de complexité $\mathcal{O}(nm)$.

Question 5
À partir de la table précédente, il est maintenant facile de déduire une LCS en partant du coin en bas à droite et en cherchant un chemin vers le coin en haut à gauche :

```
LCS(tableau table, chaine x, chaine y)
-- m := longueur(x), n := longueur(y)
-- i := m, j:= n
-- lcs := []
-- tant que i!=0 et j!=0 faire
---- si x[i] == y[i] alors
------ lcs := x[i] @ lcs
------ i := i-1, j := j-1
---- sinon si table[i,j-1] > tablea[i-1,j]
------ j := j-1
---- sinon
------ i := i-1
-- retourner lcs
```

La complexité de cel algorithme est en $\mathcal{O}(n+m)$.

Question 6 A partir de la matrice de la question 4, on peut construire un graphe à nm sommets représentant les couples (i,j), avec $0 \leq i \leq m$ et $0 \leq j \leq n$. Ces couples représentent les sous-problèmes. On établit une arête entre le sommets (i,j) et $(i-1,j)$ (resp. $(i,j-1)$) lorsque `table[`i`,`j`]=table[`$i-1$`,`j`]` (resp. `table[`i`,`$j-1$`]`) et entre (i,j) et $(i-1,j-1)$ lorsque $x_i = x_j$. Trouver un (plus court) chemin entre (n,m) et $(0,0)$ revient à trouver la LCS.

Distance de Levenshtein

Question 7 La distance d'édition entre "baillonette" et "baïonnette" est de 4 : remplacer "ï" par "i", supprimer les deux "l" et ajouter un "n".

Question 8 On peut réécrire la formule de récurrence afin de calculer la distance de Levenshtein de la manière suivante :

$$LEV\,(X_m, Y_n) = \begin{cases} n & \text{si } \; m = 0 \\ m & \text{si } \; n = 0 \\ \min \begin{cases} LEV(X_{m-1}, Y_n) + 1 & \\ LEV(X_n, Y_{n-1}) + 1 & \\ LEV(X_{m-1}, Y_{n-1}) + 1 & \text{si } x_m \neq y_n \\ LEV(X_{m-1}, Y_{n-1}) & \text{sinon} \end{cases} & \end{cases}$$

La modification de l'algorithme est alors évidente, vu qu'une fois de plus une ligne ne dépend que de la précédente, comme avant.