TD d'algorithmique avancée

Corrigé du TD 5 : plus longue sous-séquence commune

Jean-Michel Dischler et Frédéric Vivien

Problématique

Une séquence est une suite finie de symboles pris dans un ensemble fini. Si $u = \langle a_1, a_2, ..., a_n \rangle$ est une séquence, où a_1, a_2, ..., a_n sont des lettres, l'entier n est la *longueur* de u. Une séquence $v = \langle b_1, b_2, ..., b_m \rangle$ est une *sous-séquence* de $u = \langle a_1, a_2, ..., a_n \rangle$ s'il existe des entiers i_1, i_2, ..., i_m ($1 \leq i_1 < i_2 < ... < i_m \leq n$) tels que $b_k = a_{i_k}$ pour $k \in [1, m]$. Par exemple, $v = \langle B, C, D, B \rangle$ est une sous-séquence de $u = \langle A, B, C, B, D, A, B \rangle$ correspondant à la suite d'indice $\langle 2, 3, 5, 7 \rangle$.

Une séquence w est une *sous-séquence commune* aux séquences u et v si w est une sous-séquence de u et de v. Une sous-séquence commune est *maximale* ou est une *plus longue sous-séquence* si elle est de longueur maximale. Par exemple : les séquences $\langle B, C, B, A \rangle$ et $\langle B, D, A, B \rangle$ sont des plus longues sous-séquences communes de $\langle A, B, C, B, D, A, B \rangle$ et de $\langle B, D, C, A, B, A \rangle$.

Résolution par programmation dynamique

1. On cherche à déterminer la longueur d'une sous-séquence commune maximale à $u = \langle a_1, a_2, ..., a_n \rangle$ et $v = \langle b_1, b_2, ..., b_m \rangle$. On note $L(i, j)$ la longueur d'une sous-séquence commune maximale à $\langle a_1, a_2, ..., a_i \rangle$ et $\langle b_1, b_2, ..., b_j \rangle$ ($0 \leq j \leq m$, $0 \leq i \leq n$). Donnez une récurrence définissant $L(i, j)$. *Indication* : on pourra distinguer les cas $a_i = b_j$ et $a_i \neq b_j$.

$$L(i,j) = \begin{cases} 0 & \text{si } i = 0 \text{ ou } j = 0, \\ 1 + L(i-1, j-1) & \text{sinon et si } a_i = b_j, \\ \max(L(i, j-1), L(i-1, j)) & \text{sinon.} \end{cases} \quad (1)$$

En effet, si $a_i \neq b_j$, une plus longue sous-séquence commune de $\langle a_1, a_2, ..., a_i \rangle$ et $\langle b_1, b_2, ..., b_j \rangle$ ne peut pas se terminer par une lettre c égale à a_i et à b_j, et donc une plus longue sous-séquence commune
- *soit ne se termine pas par a_i et elle est alors une plus longue sous-séquence commune des séquences $\langle a_1, a_2, ..., a_{i-1} \rangle$ et $\langle b_1, b_2, ..., b_j \rangle$, et est de longueur $L(i-1, j)$;*
- *soit ne se termine pas par b_j et elle est alors une plus longue sous-séquence commune des séquences $\langle a_1, a_2, ..., a_i \rangle$ et $\langle b_1, b_2, ..., b_{j-1} \rangle$, et est de longueur $L(i, j-1)$.*

Si, par contre, $a_i = b_j$, alors une plus longue sous-séquence commune de $\langle a_1, a_2, ..., a_i \rangle$ et $\langle b_1, b_2, ..., b_j \rangle$ se termine par $a_i = b_j$ (sinon on peut la rallonger par a_i) et son préfixe (la sous-séquence moins son dernier élément) est une plus longue sous-séquence commune de $\langle a_1, a_2, ..., a_{i-1} \rangle$ et $\langle b_1, b_2, ..., b_{j-1} \rangle$.

2. Écrivez alors un algorithme récursif calculant la longueur de la plus longue sous-séquence commune de deux séquences.

```
PLSSC-Récursif(A, i, B, j)
  si i = 0 ou j = 0
    alors renvoyer 0
    sinon si A[i] = B[j]
      alors renvoyer 1+ PLSSC-Récursif(A, i − 1, B, j − 1)
      sinon renvoyer max(PLSSC-Récursif(A, i − 1, B, j), PLSSC-Récursif(A, i, B, j − 1))
```

3. Montrez que cet algorithme est de complexité au moins exponentielle dans le cas où les deux séquences n'ont pas d'éléments en commun.

Dans ce cas, on se trouve toujours dans le troisième cas de l'équation 1 et la complexité est définie par la récurrence :

$$T(n,m) = \begin{cases} 0 & \text{si } n = 0 \text{ ou } m = 0, \\ T(n,m-1) + T(n-1,m) & \text{sinon.} \end{cases}$$

$$\begin{aligned} \text{D'où} \quad T(n,m) &= T(n,m-1) + T(n-1,m) \\ &= (T(n,m-2) + T(n-1,m-1)) + (T(n-1,m-1) + T(n-2,m-1)) \\ &\geq 2T(n-1,m-1). \end{aligned}$$

Donc $T(n,m) = \Omega(2^{\min(m,n)})$.

4. Écrivez alors un algorithme suivant le paradigme de la programmation dynamique et calculant la longueur de la plus longue sous-séquence commune de deux séquences.

PLSSC-ProgDyn(A, B)

$n \leftarrow$ *longueur*(A)
$m \leftarrow$ *longueur*(B)
pour $i \leftarrow 1$ **à** n **faire** $l[i,0] \leftarrow 0$
pour $j \leftarrow 1$ **à** m **faire** $l[0,j] \leftarrow 0$
pour $i \leftarrow 1$ **à** n **faire**
  pour $j \leftarrow 1$ **à** m **faire**
    si $A[i] = B[j]$
      alors $l[i,j] \leftarrow 1 + l[i-1,j-1]$
      sinon $l[i,j] \leftarrow \max(l[i-1,j], l[i,j-1])$
renvoyer $l[n,m]$

5. Quelle est la complexité de cet algorithme ?
La complexité est due aux deux boucles imbriquées : $T(n,m) = O(nm)$.

6. Modifiez l'algorithme précédent pour que l'on puisse en plus construire *une* plus longue sous-séquence commune et affichez une telle sous-séquence.

PLSSC-ProgDyn(A, B)

$n \leftarrow$ *longueur*(A)
$m \leftarrow$ *longueur*(B)
pour $i \leftarrow 1$ **à** n **faire** $l[i,0] \leftarrow 0$
pour $j \leftarrow 1$ **à** m **faire** $l[0,j] \leftarrow 0$
pour $i \leftarrow 1$ **à** n **faire**
  pour $j \leftarrow 1$ **à** m **faire**
    si $A[i] = B[j]$
      alors $l[i,j] \leftarrow 1 + l[i-1,j-1]$
        $s[i,j] \leftarrow$ « = »
      sinon si $l[i-1,j] > l[i,j-1])$
        alors $l[i,j] \leftarrow l[i-1,j]$
          $s[i,j] \leftarrow$ « ← »
        sinon $l[i,j] \leftarrow l[i,j-1]$
          $s[i,j] \leftarrow$ « → »
renvoyer $l[n,m]$ et s

Affichage-PLSC(s, A, i, j)

si $i = 0$ ou $j = 0$ **alors renvoyer**
si $s[i,j] =$ « = »
  alors Affichage-PLSC(s, A, $i-1$, $j-1$)
    affiche $A[i]$
  sinon si $s[i,j] =$ « ← » **alors** Affichage-PLSC(s, A, $i-1$, j)
    si $s[i,j] =$ « → » **alors** Affichage-PLSC(s, A, i, $j-1$)

On appelle initialement PLSSC-ProgDyn(A, B) puis Affichage-PLSC(s, A, n, m).