Plus long facteur commun et plus long sous-mot commun

Notions abordées

- facteur, sous-mot, préfixe, suffixe
- programmation dynamique
- pseudo-code d'algorithmes
- invariant et preuves par invariant

Exercice 1 : Plus long sous-mot/facteur commun

Q. 1 Rappeler les définitions de **facteur** d'un mot et **sous-mot** d'un mot. Donner un exemple illustrant la différence entre les deux.

Solution

On dit que y est un **sous-mot** de x ssi il existe φ une fonction strictement croissante de $[\![1, |y|]\!]$ dans $[\![1, |x|]\!]$ telle que $y = x_{\varphi(1)}x_{\varphi(2)}\dots x_{\varphi(m)}$.

On dit que y est un **facteur** de x ssi il existe deux mots u et v sur Σ tels que $x = u{\cdot}y{\cdot}v$.

jrd est un sous-mot de $jardin$ mais pas un facteur.

Q. 2 Définir le problème du plus long sous-mot commun.

Solution

$$\text{PLSMC}\begin{cases}\textbf{Entrée :}\ x\in\Sigma^\star,\ y\in\Sigma^\star\\ \textbf{Sortie}\ \ \textbf{:}\ \max\{\,|u|\,|\,u\in\Sigma^\star, u \text{ est sous-mot de } x \text{ et de } y\}\end{cases}$$

Q. 3 Définir le problème du plus long facteur commun.

Solution

$$\text{PLFC}\begin{cases}\textbf{Entrée :}\ x\in\Sigma^\star,\ y\in\Sigma^\star\\ \textbf{Sortie}\ \ \textbf{:}\ \max\{\,|u|\,|\,u\in\Sigma^\star, u \text{ est facteur de } x \text{ et de } y\}\end{cases}$$

Q. 4 Expliquer comment résoudrele problème du plus long sous-mot commun par programmation dynamique. On attend :

i. la définition d'une famille de valeurs bien choisie, qui se calcule récursivement et permet de résoudre le problème ;
ii. la preuve que la relation de récurrence annoncée est correcte ;
iii. le pseudo-code d'un algorithme qui calcule la valeur optimale, muni d'un invariant ;
iv. le pseudo-code d'un algorithme qui calcule une solution optimale, muni d'un invariant.

En bonus on pourra donner la preuve de correction de ce deuxième algorithme.

Solution

Soit $(x,y)\in\Sigma^\star\times\Sigma^\star$ une instance de PLSMC. On note $n=|x|$ et $m=|x|$.

On pose, pour tout $(i,j)\in[\![0,n]\!]\times[\![0,m]\!]$:

$$\mathcal{E}_{i,j}\overset{\text{déf}}{=}\{\,u\,|\,u\in\Sigma^\star, u \text{ est sous-mot de } x_1\dots x_i \text{ et de } y_1\dots y_j\}$$
$$B_{i,j}\overset{\text{déf}}{=}\max\{|u|\,|\,u\in\mathcal{E}_{i,j}\}$$

La valeur optimale (*i.e.* la solution de PLSMC pour l'instance (x,y)) est alors donnée par $B_{n,m}$.

Le calcul des valeurs $B_{i,j}$ peut se faire grâce aux relations suivantes.

$$\forall i\in[\![0,n]\!], B_{i,0}=0 \qquad \text{car } y_1\dots y_0=\varepsilon \text{ n'a que } \varepsilon \text{ comme sous-mot, et } |\varepsilon|=0$$
$$\forall j\in[\![0,m]\!], B_{0,j}=0 \qquad \text{idem}$$
$$\forall (i,j)\in[\![1,n]\!]\times[\![1,m]\!], B_{i,j}=\begin{cases}1+B_{i-1,j-1} & \text{si } x_i=y_j\\ \max(B_{i-1,j},B_{i,j-1}) & \text{sinon}\end{cases} \qquad (\bigstar)$$

Justifions la relation de récurrence ($\bigstar$) par disjonction de cas. Soit (i,j)$\in[\![1,n]\!]\times[\![1,m]\!]$.

- Cas $x_i=y_j$.
  - Si $v\in\mathcal{E}_{i-1,j-1}$, alors $v\cdot x_i\in\mathcal{E}_{i,j}$, donc $B_{i,j}\geqslant|v\cdot x_i|=1+|v|$. Cette relation pour $v^*\in\mathcal{E}_{i-1,j-1}$ optimal, donne en particulier $B_{i,j}\geqslant 1+B_{i-1,j-1}$.
  - Si $u^*\in\mathcal{E}_{i,j}$ est optimal, nécéssairement $|u^*|\geqslant 1$ car $x_i\in\mathcal{E}_{i,j}$. On peut donc parler de la dernière lettre de u^* et décomposer $u^*=v\cdot z$ avec $v\in\Sigma^\star$ et $z\in\Sigma$. Si $z\neq x_i$, alors u^* serait sous-mot de $x_1\dots x_{i-1}$ et de $y_1\dots y_{j-1}$, et alors $u^*\cdot x_i=u^*\cdot y_j$ serait un sous-mot de $x_1\dots x_i$ et de $y_1\dots y_j$ de longueur $|u^*|+1$ soit $B_{i,j}+1$. ABSURDE.
    Ainsi $u^*=v\cdot x_i$ et $v\in\mathcal{E}_{i-1,j-1}$, donc $B_{i,j}=|u^*|=|v|+1\leqslant B_{i-1,j-1}+1$.

  Par double inégalité on a bien $B_{i,j}=1+B_{i-1,j-1}$ dans ce cas.
- Cas $x_i\neq y_j$.
  - Puisqu'un sous-mot d'un sous-mot est un sous-mot, $\mathcal{E}_{i,j-1}\subseteq\mathcal{E}_{i,j}$ donc $B_{i,j}\geqslant B_{i,j-1}$. De même $B_{i,j}\geqslant B_{i-1,j}$, d'où $B_{i,j}\geqslant\max(B_{i-1,j},B_{i,j-1})$.
  - Soit $u^*\in\mathcal{E}_{i,j}$ optimal. Si $u^*=\varepsilon$, $u^*\in\mathcal{E}_{i,j-1}$, donc $B_{i,j}=|u^*|\leqslant B_{i,j-1}\leqslant\max(B_{i-1,j},B_{i,j-1})$.
    Sinon, $u^*=v\cdot z$ avec $v\in\Sigma^\star$ et $z\in\Sigma$. Puisque $x_i\neq y_j$, on a $z\neq x_i$ ou $z\neq y_j$.
    Si $z\neq x_i$, alors $u^*\in\mathcal{E}_{i-1,j}$, et donc $B_{i,j}=|u^*|\leqslant B_{i-1,j}\leqslant\max(B_{i-1,j},B_{i,j-1})$.
    De même, si $z\neq y_j$ on a $B_{i,j}=|u^*|\leqslant B_{i,j-1}\leqslant\max(B_{i-1,j},B_{i,j-1})$.
    Ainsi on a bien $B_{i,j}\leqslant\max(B_{i-1,j},B_{i,j-1})$ dès lors que $x_i\neq y_j$.

  Par double inégalité on a bien $B_{i,j}=\max(B_{i-1,j},B_{i,j-1})$ dans ce cas.

On en déduit l'algorithme de programmation dynamique suivant.

Algorithme 1 : Algorithme pour PLSMC (valeur/solution optimale)

Entrée : Deux mots de longueurs respectives n et m : $x = x_1 \dots x_n$ et $y = y_1 \dots y_m$
Sortie : La longueur d'un plus long sous-mot commun à x et y ou un tel sous-mot

```
1  B ← tableau d'entiers indexé par [[0,n]] × [[0,m]] ;
2  M ← tableau de couples d'entiers indexé par [[0,n]] × [[0,m]] ;
3  pour i = 0 à n faire
4      B[i][0] ← 0 ;
5      M[i][0] ← (i − 1, 0) ;
6  pour j = 0 à m faire
7      B[0][j] ← 0 ;
8      M[0][j] ← (0, j − 1) ;
9  pour i = 1 à n faire
10     pour j = 1 à m faire
11         si x_i = y_j alors
12             B[i][j] ← 1 + B[i − 1][j − 1] ;
13             M[i][j] ← (i − 1, j − 1) ;
14         sinon
15             si B[i − 1][j] > B[i][j − 1] alors
16                 B[i][j] ← B[i − 1][j] ;
17                 M[i][j] ← (i − 1, j) ;
18             sinon
19                 B[i][j] ← B[i][j − 1] ;
20                 M[i][j] ← (i, j − 1) ;
21 retourner B[n][m] ;
22 (i, j) ← (n, m) ;
23 u ← ε ;
24 tant que i > 0 et j > 0 faire
25     si M[i][j] = (i − 1, j − 1) alors
26         u ← x_i · u
27     (i, j) ← M[i][j] ;
28 retourner u ;
```

Justifions la correction de l'algorithme 1 Les trois premières boucles remplissent le tableau B avec les valeurs de B grâce aux relations précédemment démontrées, autrement dit à la ligne 21 on a $\forall (i,j) \in [\![0,n]\!] \times [\![0,m]\!], \mathsf{B}[i][j] = B_{i,j}$. Ainsi la version de l'algorithme qui calcule la valeur est correct.

Pour montrer que l'algorithme qui calcule une solution est correct, on s'appuye sur l'invariant suivant pour la boucle ligne 24.

$$u \text{ est sous-mot de } x_{i+1} \dots x_n \text{ et de } y_{j+1} \dots y_m \text{ et } |u| = \mathsf{B}[n][m] - \mathsf{B}[i][j]$$

- Initialement $(i,j) = (n,m)$ donc $x_{i+1} \dots x_n = \varepsilon$ et de $y_{j+1} \dots y_m = \varepsilon$, ainsi $u = \varepsilon$ est bien un sous-mot de ces deux mots. De plus, on a alors $\mathsf{B}[n][m] - \mathsf{B}[i][j] = \mathsf{B}[n][m] - \mathsf{B}[n][m]$ et $|u| = |\varepsilon| = 0$. L'invariant proposé est donc vrai avant la boucle ligne 24.

- Supposons l'invariant vérifié à l'entrée d'un tour de boucle, montrons qu'il l'est toujours en sortie de ce tour. On note i^a (resp. j^a et u^a) la valeur de la variable i (resp. j et u) à l'entrée du tour de boucle. On note i^b (resp. j^b et u^b) les valeurs en sortie.
  - Si $\mathsf{M}[i^a][j^a] = (i^a - 1, j^a - 1)$, alors on a $i^b = i^a - 1$ et $j^b = j^a - 1$ d'après la l.27, et $u^b = x_{i^a} \cdot u^a$ d'après la l.**??**. De plus, vu comment est construit M entre les lignes 3 et 21, on a nécessairement $x_{i^a} = y_{j^a}$ et $\mathsf{B}[i^a][j^a] = 1 + \mathsf{B}[i^a - 1][j^a - 1]$.
    On a alors

$$\begin{aligned} |u^b| &= 1 + |u^a| \\ &= 1 + (\mathsf{B}[n][m] - \mathsf{B}[i^a][j^a]) \qquad \text{par hypothèse} \\ &= \cancel{1} + \mathsf{B}[n][m] - \left(\cancel{1} + \mathsf{B}[i^a - 1][j^a - 1]\right) \\ &= \mathsf{B}[n][m] - \mathsf{B}[\underbrace{i^a - 1}_{i^b}][\underbrace{j^a - 1}_{j^b}] \end{aligned}$$

    et comme u^a est sous-mot de $x_{i^a+1} \dots x_n$ par hypothèse, $u^b = x_{i^a} \cdot u^a$ est sous-mot de $x_{i^a} \dots x_n$ soit de $x_{i^b+1} \dots x_n$. De même $u^b = y_{j^a} \cdot u^a$ est sous-mot de $y_{j^a} \dots y_m$ soit de $y_{j^b+1} \dots y_m$. Ainsi les valeurs i^b,j^b et u^b vérifient bien l'invariant dans ce cas.
  - Sinon, on a $u^b = u^a$, or par hypothèse u^a est un sous-mot de $x_{i^a+1} \dots x_n$ et $y_{j^a+1} \dots y_m$, donc u^b aussi, et a fortiori u^b est sous-mot de $x_{i^a} \dots x_n$ et $y_{j^a} \dots y_m$.
    Vu comment est construit M entre les lignes 3 et 21 on a deux sous-cas possibles.
    - Si $\mathsf{M}[i^a][j^a] = (i^a - 1, j^a)$, on a aussi $\mathsf{B}[i^a][j^a] = \mathsf{B}[i^a - 1][j^a]$ par construction, et d'après la ligne 27, $i^b = i^a - 1$, $j^b = j^a$.
    - Si $\mathsf{M}[i^a][j^a] = (i^a, j^a - 1)$, on a aussi $\mathsf{B}[i^a][j^a] = \mathsf{B}[i^a][j^a - 1]$ par construction, et d'après la ligne 27, $i^b = i^a$, $j^b = j^a - 1$.

    Dans les deux sous-cas $\mathsf{B}[n][m] - \mathsf{B}[i^b][j^b] = \mathsf{B}[n][m] - \mathsf{B}[i^a][j^a]$, or par hypothèse $\mathsf{B}[n][m] - \mathsf{B}[i^a][j^a] = |u^a|$, et comme $u^b = u^a$ on a bien $\mathsf{B}[n][m] - \mathsf{B}[i^b][j^b] = |u^b|$.
    Ainsi les valeurs i^b,j^b et u^b vérifient bien l'invariant dans ce cas.

En sortant de la boucle ligne 24, on a $i = 0$ et $j = 0$, l'invariant nous assure donc que :

— u est sous-mot de $x_{0+1} \dots x_n$ et de $y_{0+1} \dots y_m$, soit sous mot de x et y ;,

— $|u| = \mathsf{B}[n][m] - \mathsf{B}[0][0]$, or on sait que $\mathsf{B}[n][m] = B_{n,m}$ depuis la ligne 21 et $\mathsf{B}[0][0] = 0$ (*Cf.* ligne 4), donc $|u| = B_{n,m}$.

Ces deux points assurent que u, le mot renvoyé, est bien un plus long sous-mot commun à x et y.

Q. 5 Expliquer comment résoudre le problème du plus long facteur commun par programmation dynamique. On attend :

i. la définition d'une famille de valeurs bien choisie, qui se calcule récursivement et permet de résoudre le problème ;
ii. la preuve que la relation de récurrence annoncée est correcte ;
iii. le pseudo-code d'un algorithme qui calcule la valeur optimale, muni d'un invariant ;
iv. le pseudo-code d'un algorithme qui calcule une solution optimale, muni d'un invariant.

Solution

Soit $(x, y) \in \Sigma^\star \times \Sigma^\star$ une instance de PLFC. On note $n = |x|$ et $m = |x|$.

On pose, pour tout $(i,j) \in [\![0,n]\!] \times [\![0,m]\!]$:

$$\mathcal{S}_{i,j} \overset{\text{déf}}{=} \{\, u \mid u \in \Sigma^\star, u \text{ est suffixe de } x_1 \dots x_i \text{ et de } y_1 \dots y_j \}$$
$$A_{i,j} \overset{\text{déf}}{=} \max\{|u| \mid u \in \mathcal{S}_{i,j}\}$$

La valeur optimale (*i.e.* la solution de PLFC pour l'instance (x,y)) est alors donnée par $\max\{A_{i,j} \mid (i,j) \in [\![0,n]\!] \times [\![0,m]\!]\}$.
En effet, un facteur est toujours un suffixe d'un préfixe, ainsi un facteur de x est un suffixe de $x_1 \dots x_i$ pour un certain $i \in [\![0,n]\!]$, et de même un facteur de y est un suffixe de $y_1 \dots y_j$ pour un certain $j \in [\![0,m]\!]$, donc on a

$$\begin{aligned}
\text{PLFC}(x,y) &= \max\left\{ |u| \,\middle|\, u \in \Sigma^\star, u \text{ est facteur de } x \text{ et de } y \right\} \\
&= \max\left\{ |u| \,\middle|\, u \in \Sigma^\star, \exists (i,j) \in [\![0,n]\!] \times [\![0,m]\!], u \text{ est suffixe de } x_1 \dots x_i \text{ et de } y_1 \dots y_j \right\} \\
&= \max\left\{ |u| \,\middle|\, u \in \bigcup_{(i,j) \in [\![0,n]\!] \times [\![0,m]\!]} \mathcal{S}_{i,j} \right\} \\
&= \max_{(i,j) \in [\![0,n]\!] \times [\![0,m]\!]} \max\left\{ |u| \,\middle|\, u \in \mathcal{S}_{i,j} \right\} \\
&= \max_{(i,j) \in [\![0,n]\!] \times [\![0,m]\!]} A_{i,j}
\end{aligned}$$

Le calcul des valeurs $A_{i,j}$ peut se faire grâce aux relations suivantes.

$$\forall i \in [\![0,n]\!], A_{i,0} = 0 \qquad \text{car } y_1 \dots y_0 = \varepsilon \text{ n'a que } \varepsilon \text{ comme suffixe, et } |\varepsilon| = 0$$
$$\forall j \in [\![0,m]\!], A_{0,j} = 0 \qquad \text{idem}$$
$$\forall (i,j) \in [\![1,n]\!] \times [\![1,m]\!], A_{i,j} = \begin{cases} 1 + A_{i-1,j-1} & \text{si } x_i = y_j \\ 0 & \text{sinon} \end{cases} \qquad (\bigstar)$$

Justifions la relation de récurrence ($\bigstar$) par disjonction de cas. Soit (i,j) $\in [\![1,n]\!] \times [\![1,m]\!]$.

- Si $x_i \neq y_j$ le seul suffixe commun à $x_1 \dots x_i$ et $y_1 \dots y_j$ est ε, car un suffixe ayant au moins une lettre aurait pour dernière lettre x_i en tant que suffixe de $x_1 \dots x_i$ et pour dernière lettre y_j en tant que suffixe de $y_1 \dots y_j$, on aurait alors $x_i = y_j$. ABSURDE.
- Si $x_i = y_j$ on a alors :

$$\begin{aligned}
\mathcal{S}_{i,j} &= \{\varepsilon\} \cup \left\{ v \cdot x_i \,\middle|\, v \in \Sigma^\star, \begin{bmatrix} v \cdot x_i \text{ est suffixe de } x_1 \dots x_{i-1} x_i \\ v \cdot y_j \text{ est suffixe de } y_1 \dots y_{j-1} y_j \end{bmatrix} \right\} \\
&= \{\varepsilon\} \cup \left\{ v \cdot x_i \,\middle|\, v \in \Sigma^\star, \begin{bmatrix} v \text{ est suffixe de } x_1 \dots x_{i-1} \\ v \text{ est suffixe de } y_1 \dots y_{j-1} \end{bmatrix} \right\} \\
&= \{\varepsilon\} \cup \mathcal{S}_{i-1,j-1} \cdot \{x_i\}
\end{aligned}$$

$$\begin{aligned}
\text{donc } A_{i,j} &= \max\left(|\varepsilon|,\ \max\left\{ \underbrace{|v \cdot x_i|}_{|v|+1} \mid v \in \mathcal{S}_{i-1,j-1} \right\} \right) \\
&= \max\left(0,\ 1 + \max\{|v| \mid v \in \mathcal{S}_{i-1,j-1}\}\right) \\
&= \max\left(0,\ \underbrace{1 + A_{i-1,j-1}}_{\geqslant 1}\right) \\
&= 1 + A_{i-1,j-1}
\end{aligned}$$

On peut aussi justifier le cas $x_i = y_j$ par double inégalité comme suit.

Soit $u^* \in \mathcal{S}_{i,j}$ tel que $|u^*| = A_{i,j}$. On sait que $u^* \neq \varepsilon$ car le mot x_i est un suffixe de $x_1 \dots x_i$ et de $y_1 \dots y_j$, strictement plus long que ε. Ainsi u^* s'écrit $v \cdot z$ avec $v \in \Sigma^\star$ et $z \in \Sigma$. Puisque u^* est suffixe de $x_1 \dots x_i$ (resp. de $y_1 \dots y_j$), on a $z = x_i$ (resp. $z = y_j$) et v est un suffixe de $x_1 \dots x_{i-1}$ (resp. de $y_1 \dots y_{j-1}$). Ainsi $|v| \leqslant A_{i-1,j-1}$, or $|v| = |u^*| - 1 = A_{i,j} - 1$, donc $A_{i,j} \leqslant 1 + A_{i-1,j-1}$.

Réciproquement, si on considère un mot $v \in \mathcal{S}_{i-1,j-1}$ tel que $|v| = A_{i-1,j-1}$, le mot $v \cdot x_i$, qui s'écrit aussi $v \cdot y_j$, est un suffixe de $x_1 \dots x_{i-1} \cdot x_i$ et de $y_1 \dots y_{j-1} \cdot y_j$, de longueur $1 + A_{i-1,j-1}$. Cela assure que $1 + A_{i-1,j-1} \leqslant A_{i,j}$.

Par double inégalité, on a bien $A_{i,j} = 1 + A_{i-1,j-1}$ dans le cas où $x_i = y_j$.

On en déduit l'algorithme de programmation dynamique suivant.

Algorithme 2 : Algorithme pour PLFC (valeur/solution optimale)

Entrée : Deux mots de longueurs respectives n et m : $x = x_1 \dots x_n$ et $y = y_1 \dots y_m$
Sortie : La longueur d'un plus long facteur commun à x et y ou un tel facteur

```
1  A ← tableau d'entiers indexé par ⟦0, n⟧ × ⟦0, m⟧ ;
2  pour i = 0 à n faire
3      A[i][0] ← 0 ;
4  pour j = 0 à m faire
5      A[0][j] ← 0 ;
6  pour i = 1 à n faire
7      pour j = 1 à m faire
8          si x_i = y_j alors
9              A[i][j] ← 1 + A[i − 1][j − 1] ;
10         sinon
11             A[i][j] ← 0 ;
12 max ← 0 ;
13 i_max ← 0 ;
14 pour i = 1 à n faire
15     pour j = 1 à m faire
16         si A[i][j] > max alors
17             max ← A[i][j] ;
18             i_max ← i ;
19 retourner max ;
20 f ← i_max ; d ← (f − max) + 1 ;
21 retourner x_d ... x_f ;
```

Éléments pour établir un algorithme de programmation dynamique

- → identifier une famille de "sous-problèmes"[a] qu'il est intéressant de résoudre ;
- → identifier quels paramètres permettent de caractériser ces sous-problèmes relativement à l'instance de départ ;
- → définir une quantité paramétrée qui représente la valeur optimale du sous-problème défini par ces paramètres ;
- → justifier que cette famille de quantités permet de résoudre le problème initial ;
- → dire comment calculer les cas de base, *i.e.* les quantités pour les paramètres minimaux ;
- → dire comment calculer l'une de ces quantités à partir des valeurs pour des paramètres plus petits ;
- → si on opte pour de l'impératif, fournir le pseudo-code qui explique quelle taille de tableau alouer pour stocker ces quantités, dans quel ordre on va remplir le tableau, comment on va ensuite récupérér la valeur optimale ;
- → dans le cas où seule la valeur optimale nous intéresse, il est souvent possible de modifier l'algorithme précédent en vue de réduire sa consommation mémoire ;
- → dans le cas où une solution optimale doit aussi être fournie, on complète l'algorithme précédent en ajoutant le plus souvent l'entegistrement d'information pertinente lors du remplissage du tableau, et en ajoutant une phase de reconstruction de la solution qui remonte dans le tableau à partir d'une case de valeur optimale.

a. On appelle ici, comme c'est l'usage, famille de sous-problèmes une famille d'instances, instances du problème initial ou parfois d'une variante de celui-ci.