

MINI-PROJET

Alignement de séquences

2.1 Quelques définitions générales en algorithmique du texte

Question 1

$$(i) \Pi(\bar{x}) = x \text{ et } \Pi(\bar{u}) = u \quad \text{par définition 2.2}$$

$$\Pi(\bar{x}, \bar{v}) = \Pi(\bar{x}) \cdot \Pi(\bar{v})$$

$$= x \cdot v$$

$$(ii) \Pi(\bar{y}) = y \text{ et } \Pi(\bar{v}) = v \quad \text{par définition 2.2}$$

$$\Pi(\bar{y}, \bar{v}) = \Pi(\bar{y}) \cdot \Pi(\bar{v})$$

$$= y \cdot v$$

$$(iii) |\bar{x}| = |\bar{y}| \text{ et } |\bar{u}| = |\bar{v}| \quad \text{par définition 2.2.}$$

Par conséquent $|\bar{x}| + |\bar{u}| = |\bar{y}| + |\bar{v}|$

$$(iv) \forall i \in [1, \dots, |\bar{x}|], x_i \neq - \text{ ou } \bar{x}_i \neq -$$

$$\forall j \in [1, \dots, |\bar{v}|], v_j \neq - \text{ ou } \bar{v}_j \neq -$$

D'après comme d'après (iii) $|\bar{x} \cdot \bar{u}| = |\bar{y} \cdot \bar{v}|$ avec $|\bar{x}| = |\bar{y}|$

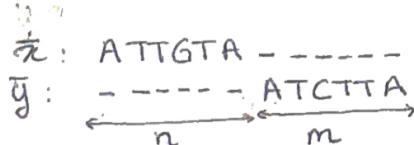
On peut dire

$$\forall i \in [1, \dots, |\bar{x} \cdot \bar{u}|], (\bar{x} \cdot \bar{u})_i \neq - \text{ ou } (\bar{y} \cdot \bar{v})_i \neq -.$$

Question 2.

La longueur maximale d'un alignement est $n + m$

Exemple : $x = \text{ATTGTA}$ et $y = \text{ATCTTA}$



3.1 Méthode naïve par énumération

Question 3

On cherche à savoir combien il y a de succès tel que succès = nombre de gaps sur un mot de longueur $n+k$.

On pose donc :

$$\binom{n+k}{n+k} = \binom{n+k}{k} = \frac{(n+k)!}{k!((n+k)-k)!} = \frac{(n+k)!}{k! n!}$$

Question 4

- Selon la définition si (\bar{x}, \bar{y}) est un alignement, alors

$$|\bar{x}| = |\bar{y}|$$

$$\text{Donc, } n+k = m+k'$$

$$\text{Donc, } k' = (n+k) - m$$

- On recherche le nombre de gaps k' à ajouter à y sachant que nous connaissons le nombre k de \bar{x} . On sait également qu'il ne peut pas y avoir deux gaps au même endroit.

Donc cela nous ramène à

$$\binom{k'}{m+k'-k} = \frac{(m+k'-k)!}{(k')!(m-k)!} = \frac{n!}{(n+k-m)!(m-k)!}$$

- Aussi le nombre d'alignements possibles pour (x, y) est la somme du nombre de combinaisons de x multiplié par le nombre de combinaisons de y en fonction de x

$$\sum_{k=0}^m \left(\binom{k}{n+k} \times \binom{k'}{m+k'-k} \right)$$

- Pour $|x|=15$ et $|y|=10$, on trouve 258199265 combinaisons!

Question 5

Comme vu à la question 4,
Un algorithme naïf ~~consisterait à faire~~ à faire $\sum_{k=0}^n \left(\binom{k}{n+k} \times \binom{k'}{m+k'-k} \right)$
~~calcul~~. Comme k varie de 0 à n ,
et $n \geq m$, on peut conclure que, un algorithme
naïf est de complexité temporelle en $O(n!)$

question 6

Il faut des tableaux à une dimension pour stocker l'alignement de x et y . D'algorithme à une complexité spatiale en $O(n)$.

3.2 Programmation dynamique

3.2.1 Calcul de la distance d'édition par programmation dynamique

Question 7

- Si $\bar{u}_e = -$, alors $\bar{v}_e \neq -$ donc \bar{v}_e est une lettre soit $\bar{v}_e = y_j$.
- Si $\bar{v}_e = -$, alors $\bar{u}_e \neq -$ donc \bar{u}_e est une lettre soit $\bar{u}_e = x_i$.
- Si $\bar{u}_e \neq -$ et $\bar{v}_e \neq -$, alors \bar{u}_e et \bar{v}_e sont des lettres soit $\bar{u}_e = x_i$ et $\bar{v}_e = y_j$.

Question 8

- Si $\bar{u}_e = -$
alors $C(x, y) = C(\bar{u}_{[1, \dots, i-1]}, \bar{v}_{[1, \dots, i-1]}) + c_{\text{ins}}$
- Si $\bar{v}_e = -$
alors $C(x, y) = C(\bar{u}_{[1, \dots, i-1]}, \bar{v}_{[1, \dots, i-1]}) + c_{\text{del}}$
- Si $\bar{u}_e = -$ et $\bar{v}_e = -$
alors $C(x, y) = C(\bar{u}_{[1, \dots, i-1]}, \bar{v}_{[1, \dots, i-1]}) + c_{\text{sub}}(x_i, y_j)$

Question 9

$D(i, j)$ peut prendre 3 valeurs différentes :

- $D(i-1, j) + c_{\text{ins}}$
- $D(i, j-1) + c_{\text{del}}$
- $D(i-1, j-1) + c_{\text{sub}}(x_i, y_j)$

$D(i, j)$ calcule le coup optimal d'un alignement de x_1, \dots, x_n et y_1, \dots, y_m .
Donc pour que le coup reste optimal il faut $D(i, j)$ soit égale à la plus petite de ces 3 valeurs.
Par conséquent,

$$D(i, j) = \min \begin{cases} D(i-1, j) + c_{\text{ins}} \\ D(i, j-1) + c_{\text{del}} \\ D(i-1, j-1) + c_{\text{sub}}(x_i, y_j) \end{cases}$$

Question 10

$$D(0, 0) = d(x_0, y_0) = d(\epsilon, \epsilon) = 0.$$

Question 11

$$D(0, j) = j \times c_{\text{ins}} \quad \forall j \in \mathbb{N}, \text{ car cela correspond à } j \text{ insertions.}$$
$$D(i, 0) = i \times c_{\text{del}} \quad \forall i \in \mathbb{N}, \text{ car cela correspond à } i \text{ suppressions.}$$

Question 12

DIST_1

Entrée: x et y deux mots

Sortie $d(x, y)$ la distance d'édition entre x et y .

$$n \leftarrow |x|$$

$$m \leftarrow |y|$$

$T \leftarrow \text{Creation_nouveau_tableau}(n, m)$

$$T[0][0] \leftarrow 0$$

Pour i allant de 1 à n faire

$$\quad \quad T[i][0] \leftarrow i * \text{cdel}$$

Pour j allant de 1 à m faire

$$\quad \quad T[0][j] \leftarrow j * \text{cins}$$

Pour i allant de 1 à n faire

Pour j allant de 1 à m faire.

$$\quad \quad T[i][j] \leftarrow \min(T[i-1][j] + \text{cins}, T[i][j-1] + \text{cdel}, T[i-1][j-1] + \text{csub}(x_i, y_j))$$

Retourner $T[n][m]$

Question 13

On construit un tableau de taille $n \times m$ donc la complexité spatiale est en $O(n \times m)$

Question 14

Pour deux séquences de longueur n et m , le nombre de comparaisons est $O(n \times m)$.

La complexité de DIST_1 est quadratique, donc la complexité temporelle est en $O(n^2)$.

3.2.2 Calcul d'un alignement optimal par programmation dynamique.

Question 15

Si $D(i, j) = D(i, j-1) + \text{ias}$, cela veut dire que pour avoir la distance d'édition minimale $d(x_{[1, \dots, i]}, y_{[1, \dots, j]})$ il faut faire une insertion. L'insertion est encodé par un gap dans \bar{x} .
D'où (\bar{s}, \bar{t}, y_j) est bien un alignement de $(x_{[1, \dots, i]}, y_{[1, \dots, j]})$

Question 46

SOL-1

Entrée : (x, y) un couple de mot

T un tableau contenant les valeurs de D

Sortie : (\bar{x}, \bar{y}) un alignement minimal de (x, y)

$\bar{x} \leftarrow \text{carré.chaine_vide}$

$\bar{y} \leftarrow \text{carré.chaine_vide}$

$i \leftarrow |x|$

$j \leftarrow |y|$

Tant que $i := 0$ et $j := 0$ faire.

Si $t[i][j] - t[i-1][j] = cDel$

$\bar{x} \leftarrow \bar{x} + x_{i-1}$

$\bar{y} \leftarrow \bar{y} + '-'$

Si $t[i][j] - t[i][j-1] = cIns$

$\bar{x} \leftarrow \bar{x} + '-'$

$\bar{y} \leftarrow \bar{y} + y_{j-1}$

Si $t[i][j] - t[i-1][j-1] = cSub(x_{i-1}, y_{j-1})$

$\bar{x} \leftarrow \bar{x} + x_{i-1}$

$\bar{y} \leftarrow \bar{y} + y_{j-1}$

$i \leftarrow i-1$

$j \leftarrow j-1$

$\oplus i = i-1$

$\bar{x} \leftarrow \text{chaine_inverse}(\bar{x})$

$\bar{y} \leftarrow \text{chaine_inverse}(\bar{y})$

Retourner (\bar{x}, \bar{y})

Question 47

En combinant les algorithmes DIST-1 et SOL-1, on fait la création du tableau par les deux séquences x et y , la recherche du chemin optimal dans la matrice, ainsi que la construction de l'alignement.
On obtient une complexité en $O(n * m)$

Question 18

On créer une matrice de taille $n * m$...
L'algorithme qui combine DIST-1 et SOL-1 est de complexité spatiale $O(n * m)$.

3.3 Amélioration de la complexité spatiale du calcul de la distance.

Question 19

Nous avons vu dans les questions précédentes que pour remplir une case (i, j) , nous avons besoin de la case $(i-1, j-1)$, $(i-1, j)$ et de la case $(i, j-1)$. Une fois les valeurs de la ligne $i-1$ calculée, toutes les lignes antérieures ne sont donc plus d'utilité si l'on veut seulement recuperer la distance d'édition.

Question 20

DIST_2

Entrée: x et y deux mots

Sortie: $d(x, y)$ la distance d'édition entre x et y .

$n \leftarrow |x|$

$m \leftarrow |y|$

$\text{vide} \leftarrow \text{creation_nouveau_tableau}(m)$

$\text{tab} \leftarrow \text{creation_nouveau_tableau}(n, m)$

$\text{tab}[0][0] = 0$

pour j allant de 1 à $m+1$ faire:

$\text{tab}[0][j] \leftarrow j * c_{\text{Ins}}$

pour i_1 allant de 1 à $n+1$ faire:

$\text{tab}[1][i_1] \leftarrow i_1 * c_{\text{Del}}$

 pour j_1 allant de 1 à $m+1$ faire:

$\text{tab}[1][j_1] = \min(\text{tab}[0][j_1] + c_{\text{Ins}}, \text{tab}[1][j_1-1] + c_{\text{Del}},$
 $\quad + \text{tab}[0][j_1-1] + c_{\text{Sub}}(x[i_1-1], y[j_1-1]))$

 pour j_2 allant de 0 à $m+1$ faire:

$\text{tab}[0][j_2] = \text{tab}[1][j_2]$

$\text{tab}[1] = \text{vide}$

Retourner tab

3.4 Amélioration de la complexité spatiale du calcul d'un alignement optimal par la méthode "diviser pour régner"

Question 21.

mot-gaps

Entrée: k un entier naturel

Sortie: mot un mot constitué de k gaps.

pour i allant de 1 à k :

$\text{mot} \leftarrow \text{mot} + \text{''-''}$

Rehausser mot.

Garrison 22

align - lettre - mot

Entrée : x et y deux mots
Sortie : (\bar{x}, \bar{y}) le meilleur alignement

Si $x = y$ fane:

L Rehauer (x, y)

si x est une lettre de y faire :

$i \leftarrow y.\text{index}(x)$

$\bar{x} \leftarrow \text{not_gap}(\text{`y'})$

$$\bar{x}[i] \leftarrow x$$

RETOURNER (\bar{x}, y)

sinan

soit paire-concordante (x) est une lettre de y faire

```
i <- y.index(palae.concordance(x))
```

$\bar{x} \leftarrow \text{mot-gap}'(iy)$

$\pi[i] \leftarrow \infty$

returner (\bar{x}, y)

Sinan :

$$\bar{\pi} \leftarrow \text{met-gap}(|y|)$$

$$\bar{x}[0] \leftarrow x$$

retourner (\bar{x}, y)

10

Question 23

• Cherchons un alignement optimal (\bar{s}, \bar{t}) de (BAL, RD) .

x_1 : B A L est le meilleur car de cout 13

le Old Town

est le meilleur car de tout 13

- Cherchons un alignement optimal (\bar{u}, \bar{v}) de (LON, ND)

x_2 : LON- est le meilleur car de tout g

$$\Delta j_m$$

Or si on concatene les deux alignements on obtient un alignement $(\bar{s}, \bar{u}, \bar{t}, \bar{v})$ de cout $13 + 9 = 22$.

Cet alignement n'est pas optimal puisqu'il existe un alignement de coût inférieur

B A L L O N -
- - - R O N D
3 3 3 5 0 0 M

Question 24

SOL - 2

Entrée : (x, y) un couple de mot
Sortie : (\bar{x}, \bar{y}) un alignement minimal

```

soit  $|x| = l$  et  $|y| = m$  faire :
[...]
    retourner align - mot - lettr -  $(x, y)$ 
si  $|y| = 0$  faire :
[...]
    retourner  $(x, motgap(n))$ 
sinon :
     $i \leftarrow |x| / 2$ 
     $j \leftarrow couple(x, y)$ 
     $(\bar{x}_1, \bar{y}_1) = SOL-2(x[:i], y[:j])$ 
     $(\bar{x}_2, \bar{y}_2) = SOL-2(x[i:], y[j:])$ 

```

Question 25

Couper

Entrée : 2 mots x, y
Sortie :

$n \leftarrow |x|$

$m \leftarrow |y|$

$\text{vide} \leftarrow \text{creation_nouveau_tableau}(m)$

$\text{tab} \leftarrow \text{creation_nouveau_tableau}(n, m)$

$\text{tab}[0][0] \leftarrow 0$

$\text{tabI} \leftarrow \text{creation_nouveau_tableau}(m, m)$

pour j allant de 1 à $m+1$ faire :

[...]
 $\text{tab}[0][j] \leftarrow j * cIns$

pour $i \in \mathbb{N}$ allant de 1 à $n+1$ faire :

$\text{tab}[i][0] \leftarrow i * cDel$

$\text{tabI}[i][0] \leftarrow 0$

pour $j \in \mathbb{N}$ allant de 1 à $m+1$ faire :

$\text{tab}[i][j] \leftarrow \min(\text{tab}[i-1][j] + cIns, \text{tab}[i-1][j-1] + cDel,$
 $\text{tab}[i-1][j-1] + cSub(x[i-1], y[j-1]))$

si $i = n/2$ faire :

[...]
 $\text{tabI}[i][j] = j$

si $i > n/2$ faire :

[...]
 $\text{tabI}[i][j] = \text{tab}[i][j] - \text{tab}[i][j-1] - cDel$

[...]
 $\text{tabI}[i][j] = \text{tabI}[i-1][j-1]$

[...]
 $\text{tabI}[i][j] = \text{tab}[i][j] - \text{tab}[i-1][j] - cIns$

[...]
 $\text{tabI}[i][j] = \text{tabI}[i-1][j]$

[...]
 $\text{tabI}[i][j] = \text{tab}[i][j] - \text{tab}[i-1][j-1] - cSub(x[i-1], y[j-1])$

[...]
 $\text{tabI}[i][j] = \text{tabI}[i-1][j-1]$

pour j_2 allant de 0 à $m+1$ faire
 $\text{tab}[0][j_2] \leftarrow \text{tab}[1][j_2]$
 si $i_1 \geq n/2$ faire
 $\quad \text{tabI}[0][j_2] \leftarrow \text{tabI}[1][j_2]$
 $\text{tab}[1] = \text{vide}$
 $\text{tabI}[1] = \text{vide}$

Retourner $\text{tabI}[0][m]$

Question 26

On garde seulement deux lignes des tableaux, on a une complexité spatiale en $O(m)$

Question 27

De la même manière la complexité spatiale de sol-2 est en $O(m)$.

Question 28

ous pourra calculer les valeurs des cases de tab et de tabI. La complexité temporelle est en $O(n^2 m)$.

Question 29

L'étude des courbes B et D nous montre que en améliorant la complexité spatiale, on perd en complexité temporelle.