Extrait d'un DS de MP2I

Exercice 1 : Calcul d'alignements et de distance d'édition

Ce sujet porte sur un problème de génomique : l'alignement de séquences. D'un point de vue biologique, il s'agit de mesurer la similarité entre deux séquences d'ADN, que l'on voit simplement comme des suites de nucléotides. Cela permet, lorsqu'un nouveau génome est séquencé, de comparer ses gènes à ceux de génomes précédemment séquencés, et de repérer ainsi des ressemblances dues au fait que les deux espèces ont un ancêtre commun qui leur a transmis ce gène, même si ce gène a pu subir des mutations (évolutions) au cours du temps.

D'un point de vue informatique, les séquences de nucléotides sont vues comme des mots sur l'alphabet $\{\mathsf{A}, \mathsf{T}, \mathsf{G}, \mathsf{C}\}$ et l'on est ramené à deux problèmes d'algorithmique du texte : le calcul de la distance d'édition entre deux mots quelconques et la production d'un alignement réalisant cette distance. Pour chacun de ces problèmes, on s'intéresse d'abord à un algorithme naïf, puis à un algorithme de programmation dynamique. Enfin, on utilise la méthode diviser pour régner pour améliorer la complexité spatiale de ces algorithmes.

1. Le problème d'alignement de séquences

Dans cette partie on introduit d'abord des définitions générales en algorithmique du texte puis les notions d'alignement et de distance d'édition qui sont centrales pour ce projet.

Rappel. Alphabet, longueur, concaténation, mot vide. Un **alphabet** est un ensemble fini de caractères. Soit Σ un alphabet non vide. Un **mot** sur Σ est une suite finie de caractères de Σ. On notera Σ^* l'ensemble des mots sur Σ. Pour $x \in \Sigma^*$, la **longueur** du mot x, notée $|x|$, est la longueur de la suite. Si $|x| = n$, les caractères de x seront indexés par $[\![1, n]\!]$, le mot x s'écrit alors $x_1 x_2 \ \ldots \ x_n$. Soient x et y deux mots sur Σ de longueurs respectives n et m. La **concaténation** des mots x et y, notée $x \cdot y$ et y est le mot $x_1\, x_2 \ \ldots \ x_n\, y_1\, y_2 \ \ldots \ y_m$. Quel que soit l'alphabet, il existe un seul mot de longueur nulle appelé **le mot vide** et noté ε. Lorsque x est un mot de longueur n et $0 \leqslant i \leqslant n$, on note $x_{[\![1,i]\!]}$ le mot $x_1\ x_2\ \ldots\ x_i$. Aussi $x_{[\![1,0]\!]} = \varepsilon$, $x_{[\![1,1]\!]} = x_1$ et $x_{[\![1,n]\!]} = x$.

Définitions : Σ, $\overline{\Sigma}$, π. Dans le reste de cet exercice, on suppose que Σ ne contient pas le symbole spécial $-$, que l'on appellera un **gap**. On note alors $\overline{\Sigma}$ l'ensemble $\Sigma \cup \{-\}$. On définit de plus la fonction π de $\overline{\Sigma}^*$ dans Σ^* telle que $\pi(x)$ est le mot obtenu en supprimant tous les $-$ de x. Aussi si $\Sigma = \{a, b\}$, on a : $aababb \in \Sigma^*$ et $a-ba--a \in \overline{\Sigma}^*$ et $\pi(a-ba--a) = abaa$.

Alignement de deux mots. Soit $(x, y) \in \Sigma^* \times \Sigma^*$ deux mots, on dit alors que deux mots $(\bar{x}, \bar{y}) \in \overline{\Sigma}^* \times \overline{\Sigma}^*$ sont un **alignement** de x et y si :

$i)$ $\pi(\bar{x}) = x$;
$ii)$ $\pi(\bar{y}) = y$;
$iii)$ $|\bar{x}| = |\bar{y}|$;
$iv)$ $\forall i \in [\![1, |\bar{x}|]\!], \bar{x}_i \neq -$ ou $\bar{y}_i \neq -$.

En français, $(\bar{x}, \bar{y})$ est un alignement de (x, y) si et seulement si on obtient x (respectivement y) en supprimant tous les gaps dans $\bar{x}$ (respectivement dans $\bar{y}$), si $\bar{x}$ et $\bar{y}$ sont de même longueur et s'ils ne présentent pas de gap à la même position.

Pour $\Sigma = \{\mathsf{A}, \mathsf{T}, \mathsf{G}, \mathsf{C}\}$, $x =$ ATTGTA et $y =$ ATCTTA, voici plusieurs alignements de (x, y) :

$\bar{x}$: A T T G T A	$\bar{x}$: A T − T G T A	$\bar{x}$: A T T G T − A	$\bar{x}$: − − − − − − A T T G T A
$\bar{y}$: A T C T T A	$\bar{y}$: A T C T − T A	$\bar{y}$: A T − C T T A	$\bar{y}$: A T C T T A − − − − − −

On remarque que si $(\bar{x}, \bar{y})$ et $(\bar{u}, \bar{v})$ sont respectivement des alignements de (x, y) et (u, v), alors $(\bar{x} \cdot \bar{u}, \bar{y} \cdot \bar{v})$ est un alignement de $(x \cdot u, y \cdot v)$.

Q. 1 Si $x \in \Sigma^*$ est un mot de longueur n et $y \in \Sigma^*$ est un mot de longueur m, quelle est la longueur maximale d'un alignement de (x, y) ? **On n'attend pas une preuve du résultat, seulement une borne.**

Solution

$n+m$ est la taille maximale d'un alignement de (x, y).

Explications : *Soit $(\bar{x}, \bar{y})$ un alignement de (x, y). Notons l sa longueur, et k son nombre total de gaps. D'après (iii), $l = |\bar{y}| = |\bar{x}|$, donc $2\,l = |\bar{x}| + |\bar{y}|$. D'après (i) et (ii), les caractères apparaissant dans $\bar{x}$ et $\bar{y}$ hormis les gaps, sont exactement ceux de x et y, donc $|\bar{x}| + |\bar{y}| = |x| + |y| + k$. Enfin d'après (iv), chaque gap est aligné avec une lettre de x ou y donc $k \leqslant |x| + |y|$. On a finalement $2\,l \leqslant 2(|x| + |y|)$, soit $l \leqslant n + m$.*
Ainsi $n+m$ un majorant de la taille d'un alignement de (x, y), et puisque c'est aussi la longueur de l'alignement proposé ci-après, c'est la longueur maximale d'un alignement de (x, y).

$\bar{x}$: $x_1\, x_2\, \ldots\, x_n\, -\, -\, -\, \ldots\, -$
$\bar{y}$: $-\, -\, \ldots\, -\, y_1\, y_2\, y_3\, \ldots\, y_m$

2. Coût d'un alignement et distance d'édition

Opérations d'édition Un alignement $(\bar{x}, \bar{y})$ de (x, y) peut aussi être vu comme décrivant une manière de transformer le mot x en y en utilisant les trois opérations suivantes (appelées opérations d'édition) :

- l'**insertion**, qui consiste à insérer une lettre de y dans x, est encodée par un gap dans $\bar{x}$
- la **suppression**, qui consiste à supprimer une lettre du mot x, est encodée par un gap dans $\bar{y}$
- la **substitution**, qui consiste à changer une lettre de x en une lettre de y, est encodée par deux lettres différentes à la même position dans $\bar{x}$ et $\bar{y}$

Dans l'exemple précédent, le troisième alignement indique que l'on peut transformer x en y en : **supprimant** le deuxième T, **substituant** le G en C, **insérant** un T après le dernier T.
De ce point de vue, les différents alignements proposés dans l'exemple ne se valent pas. Le 4e encode une transformation qui nécessite 5 insertions et 5 suppressions, tandis qu'une insertion et une suppression suffisent, comme le montre le 2e alignement. Le 4e alignement aura donc un coût plus élevé que le 2e. Mais pour savoir ce qui est le plus coûteux entre deux substitutions (1er alignement) et une substitution + une insertion + une suppression (3e alignement), il faut fixer un coût pour chacune de ces opérations.

Coût d'un alignement. On fixe donc $c_{\mathsf{i}} \in \mathbb{R}^+$ (resp. $c_{\mathsf{d}} \in \mathbb{R}^+$) le coût d'une insertion (resp. d'une suppression), et pour tout couple de lettres différentes $(a,b) \in \Sigma^2$, $c_{\mathsf{s}}(a,b) \in \mathbb{R}^+$ le coût de la substitution de b à a (on remplace a par b). Pour des raisons pratiques, on s'autorise à écrire $c_{\mathsf{s}}(a,b)$ même si $a = b$, c'est-à-dire même si l'opération de substitution de a par b revient à ne rien faire, et ne coûte donc rien. On pose donc $\forall a \in \Sigma,\ c_{\mathsf{s}}(a,a) = 0$. Soit $(\bar{x}, \bar{y})$ un alignement de $(x,y) \in \Sigma^* \times \Sigma^*$ de longueur l. Le **coût** de l'alignement $(\bar{x}, \bar{y})$, noté $C(\bar{x}, \bar{y})$ est défini comme suit :

$$C(\bar{x},\bar{y}) = \sum_{k=1}^{l} c(\bar{x}_k, \bar{y}_k) \quad \text{où} \quad \forall (a,b) \in \overline{\Sigma}^2 \setminus \{(-,-)\},\ c(a,b) = \begin{cases} c_{\mathsf{i}} & \text{si } a = - \\ c_{\mathsf{d}} & \text{si } b = - \\ c_{\mathsf{s}}(a,b) & \text{sinon} \end{cases}$$

Puisqu'il n'y a qu'un nombre fini d'alignements de deux mots, on peut parler du coût minimal d'un alignement de deux mots, et ainsi définir la distance d'édition.

Distance d'édition. Soit $(x,y) \in \Sigma^* \times \Sigma^*$. La **distance d'édition de x à y**, alors définie par :

$$d(x,y) = \min \left\{ C(\bar{x},\bar{y}) \mid (\bar{x},\bar{y}) \text{ est un alignement de } (x,y) \right\}$$

Les problèmes qui nous intéressent. Sous le nom "alignement de séquences" se cachent en réalité deux problèmes : le calcul de la distance d'édition et la production d'un alignement de coût minimal. Afin d'éviter toute ambiguïté dans la suite on définit donc ces deux problèmes ci-dessous.

DIST	entrée :	x et y deux mots sur Σ
	sortie :	$d(x,y)$
ALI	entrée :	x et y deux mots sur Σ
	sortie :	$(\bar{x},\bar{y})$ un alignement de (x,y) tel que $C(\bar{x},\bar{y}) = d(x,y)$

3. Algorithmes naïfs

Dans cette sous-partie, on envisage de résoudre les deux problèmes en énumérant tous les alignements possibles entre deux mots.

Q. 2 Étant donné $x \in \Sigma^*$ un mot de longueur n, combien y a-t-il de mots $\bar{x}$ obtenus en ajoutant à x exactement k gaps ? Autrement dit combien y a-t-il de mots $\bar{x} \in \overline{\Sigma}^*$ tels que $|\bar{x}| = n+k$ et $\pi(\bar{x}) = x$?

Solution

Il y en a $\binom{n+k}{k}$ puisqu'il suffit pour définir un tel mot $\bar{x}$ de préciser les k positions des gaps parmi les $n+k$ caractères en tout.

Q. 3 Étant donné $\bar{x} \in \overline{\Sigma}^*$ de longueur $n+k$ ayant exactement k gaps et un mot $y \in \Sigma^*$ de longueur $m \in [\![k, n+k]\!]$, combien y a-t-il de mots $\bar{y}$ obtenus en ajoutant à y des gaps jusqu'à atteindre la longueur de $\bar{x}$ et en prenant soin que ceux-ci ne soient pas aux mêmes positions que les gaps dans $\bar{x}$? Autrement dit combien y a-t-il de mots $\bar{y} \in \overline{\Sigma}^*$ tels que $|\bar{y}| = |\bar{x}|$, $\pi(\bar{y}) = y$ et $\forall i \in [\![1, |\bar{x}|]\!], \bar{y}_i \neq -$ ou $\bar{x}_i \neq -$?

Solution

Posons $k' = n+k-m$. C'est le nombre de gap à insérer dans y. On a $k' \in [\![0, n]\!]$. Comme précédemment un mot $\bar{y}$ tel que $\pi(\bar{y}) = y$ est entièrement défini par la donnée des positions où figurent des gaps. Il y a a priori $n+k$ positions dans $\bar{y}$, mais la contrainte de ne pas placer de gaps aux mêmes positions que les gaps dans $\bar{x}$ laisse seulement n positions possibles pour les gaps dans $\bar{y}$. Il y donc a $\binom{n}{k'}$ mots $\bar{y}$ qui conviennent.

Q. 4 En déduire le nombre d'alignements possibles d'un couple de mots (x, y) de longueurs respectives n et m, en supposant que $n \geqslant m$. On ne demande pas de simplifier l'expression obtenue.

Solution

On raisonne sur le nombre de gaps qui peuvent apparaître dans $\bar{x}$ si $(\bar{x}, \bar{y})$ est un alignement de (x, y). Il peut y en avoir au minimum 0 et jusqu'à m d'après la question **??**. De plus grâce aux questions 2 et 3, on sait que pour $k \in [\![0, m]\!]$, il y a $\binom{n+k}{k}$ $\bar{x}$ ayant k gaps possibles, et que pour chaque $\bar{x}$, il y a $\binom{n}{n+k-m}$ $\bar{y}$ possibles. Il y a donc $\binom{n+k}{k}\binom{n}{n+k-m}$ alignements $(\bar{x}, \bar{y})$ avec k gaps dans $\bar{x}$.
En tout, il y a donc $\sum_{k=0}^{m} \binom{n+k}{k}\binom{n}{n+k-m}$ alignements possibles.

4. Mise en équation récursive du/des problème(s)

On s'efforce ici de briser récursivement le problème en sous-problèmes. Étant donné deux mots x et y on définit donc :

$$\mathsf{Align}(x, y) = \left\{(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \text{ est un alignement de } (x, y)\right\}$$

En raisonnant sur les dernières lettres des alignements on obtient la relation suivante.

$$\begin{aligned}\mathsf{Align}(x, y) = &\{(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \text{ est un alignement de } (x, y) \text{ et } \bar{x}_{|\bar{x}|} = -\}\\ \cup &\{(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \text{ est un alignement de } (x, y) \text{ et } \bar{y}_{|\bar{y}|} = -\}\\ \cup &\{(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \text{ est un alignement de } (x, y) \text{ et } \bar{x}_{|\bar{x}|} \neq - \text{ et } \bar{y}_{|\bar{y}|} \neq -\}\end{aligned}$$

Lorsque x est un mot non vide de Σ^*, on note $\tilde{x}$ le sous mot $x_{[\![1,|x|-1]\!]}$.

Q. 5 Démontrer que pour tout couple de mots $(x, y) \in \Sigma^* \times \Sigma^*$ on a :

$$\{(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{x}_{|\bar{x}|} = -\} = \{(\bar{u} \cdot -, \bar{v} \cdot y_{|y|}) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(x, \widetilde{y})\}$$

Solution

Notons $n = |x|$ et $m = |y|$

$\subseteq$ Soit $(\bar{x}, \bar{y})$ est un alignement de longueur l de (x, y) tel que $\bar{x}_l = -$. On a $\bar{y}_l \neq -$ par iv), et puisque $\pi(\bar{y}) = y$ par ii), on a $y_m = \bar{y}_l$. Finalement :

— $x = \pi(\bar{x})$ par ii), or $\pi(\bar{x}) = \pi(\widetilde{\bar{x}})$ car $\bar{x}_l = -$, donc $\pi(\widetilde{\bar{x}} = x$

— $\widetilde{y} \cdot y_m = y = \pi(\bar{y})$ par ii), or $\pi(\bar{y}) = \pi(\widetilde{\bar{y}} \cdot \bar{y}_l) = \pi(\widetilde{\bar{y}}) \cdot \pi(\bar{y}_l) = \pi(\widetilde{\bar{y}}) \cdot y_m$ donc $\pi(\widetilde{\bar{y}}) = \widetilde{y}$

— $|\widetilde{\bar{x}}| = l - 1 = |\widetilde{\bar{y}}|$

— $\forall i \in [\![1, l-1]\!], \widetilde{\bar{x}}_i = \bar{x}_i$ et $\bar{y}_i = \widetilde{\bar{y}}_i$

Finalement $(\widetilde{\bar{x}}, \widetilde{\bar{y}})$ est un alignement de longueur $l - 1$ de $(x, \widetilde{y})$.

$\supseteq$ Réciproquement soit $\bar{x} = \bar{u} \cdot -$ et $\bar{y} = \bar{v} \cdot y_m$ avec $(\bar{u}, \bar{v}) \in \mathsf{Align}(x, \widetilde{y})$, on a alors :

— $\pi(\bar{x}) = \pi(\bar{u} \cdot -) = \pi(\bar{u}) \cdot \pi(-) = \pi(\bar{u}) = x$ par ii)
— $\pi(\bar{y}) = \pi(\bar{v} \cdot y_m) = \pi(\bar{v}) \cdot \pi(y_m) = \widetilde{y} \cdot y_m = y$
— $|\widetilde{\bar{x}}| = 1 + |\widetilde{\bar{u}}| = 1 + |\widetilde{\bar{v}}| = |\widetilde{\bar{y}}|$
— $\forall i \in [\![1, |\bar{x}| - 1]\!], \bar{x}_i = \bar{u}_i$ et $\bar{v}_i = \widetilde{\bar{y}}_i$ de plus $\bar{y}_{|\bar{y}|} \neq -$.

On admettra de même que :
— $\{(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{y}_{|\bar{y}|} = -\} = \{(\bar{u} \cdot x_{|x|}, \bar{v} \cdot -) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(\widetilde{x}, y)\}$
— $\{(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{x}_{|\bar{x}|} \neq - \text{ et } \bar{y}_{|\bar{y}|} \neq -\} = \{(\bar{u} \cdot x_{|x|}, \bar{v} \cdot y_{|y|}) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(\widetilde{x}, \widetilde{y})\}$

Q. 6 En déduire **rigoureusement** que pour tout couple de mots $(x, y) \in \Sigma^* \times \Sigma^*$ on a :

$$d(x,y) = \begin{cases} 0 & \text{si } x = \varepsilon \text{ et } y = \varepsilon \quad \star \\ c_{\mathsf{i}} + d(x, \widetilde{y}) & \text{si } x = \varepsilon \text{ et } y \neq \varepsilon \\ c_{\mathsf{d}} + d(\widetilde{x}, y) & \text{si } x \neq \varepsilon \text{ et } y = \varepsilon \\ \min(c_{\mathsf{i}} + d(x, \widetilde{y}), c_{\mathsf{d}} + d(\widetilde{x}, y), c_{\mathsf{s}}(x_{|x|}, y_{|y|}) + d(\widetilde{x}, \widetilde{y})) & \text{sinon} \quad \star \end{cases}$$

On se contentera de démontrer les cas $\star$ et on donnera l'intuition des autres cas.

Solution

— Si $x = y = \varepsilon$, alors $d(x, y) = 0$ car le seul alignement de $(\varepsilon, \varepsilon)$ est $(\varepsilon, \varepsilon)$ (notamment parce que d'après la première question la longueur d'un tel alignement est au plus $|\varepsilon| + |\varepsilon| = 0$)
— Sinon :

$$\begin{aligned} d(x,y) &= \min\{C(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y)\} \\ &= \min(\{C(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{x}_{|\bar{x}|} = -\} \\ &\quad \cup \{C(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{y}_{|\bar{y}|} = -\} \\ &\quad \cup \{C(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{x}_{|\bar{x}|} \neq - \text{ et } \bar{y}_{|\bar{y}|} \neq -\}) \end{aligned}$$

— Si $x = \varepsilon$ alors $\{C(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{x}_{|\bar{x}|} \neq - \text{ et } \bar{y}_{|\bar{y}|} \neq -\} = \emptyset$ et $\{C(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{y}_{|\bar{y}|} = -\} = \emptyset$ donc :

$$\begin{aligned} d(x,y) &= \min\{C(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{x}_{|\bar{x}|} = -\} \\ &= \min\{C(\bar{u} \cdot -, \bar{v} \cdot y_{|y|}) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(x, \widetilde{y})\} \\ &= c_{\mathsf{i}} + \min\{C(\bar{u}, \bar{v}) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(x, \widetilde{y})\} \\ &= c_{\mathsf{i}} + d(x, \widetilde{y}) \end{aligned}$$

— Si $x \neq \varepsilon$ et $y \neq \varepsilon$ alors

$$\begin{aligned} d(x,y) &= \min(\{C(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{x}_{|\bar{x}|} = -\} \\ &\qquad \cup \{C(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{y}_{|\bar{y}|} = -\} \\ &\qquad \cup \{C(\bar{x}, \bar{y}) \mid (\bar{x}, \bar{y}) \in \mathsf{Align}(x, y) \text{ et } \bar{x}_{|\bar{x}|} \neq - \text{ et } \bar{y}_{|\bar{y}|} \neq -\}) \\ &= \min(\min\{C(\bar{u} \cdot -, \bar{v} \cdot y_{|y|}) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(x, \widetilde{y})\}, \\ &\qquad \min\{C(\bar{u} \cdot x_{|x|}, \bar{v} \cdot -) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(\widetilde{x}, y)\}, \\ &\qquad \min\{C(\bar{u} \cdot x_{|x|}, \bar{v} \cdot y_{|y|}) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(\widetilde{x}, \widetilde{y})\}) \\ &= \min(c_{\mathsf{i}} + \min\{C(\bar{u}, \bar{v}) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(x, \widetilde{y})\}, \\ &\qquad c_{\mathsf{d}} + \min\{C(\bar{u}, \bar{v}) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(\widetilde{x}, y)\}, \\ &\qquad c_{\mathsf{s}}(x_{|x|}, y_{|y|}) + \min\{C(\bar{u}, \bar{v}) \mid (\bar{u}, \bar{v}) \in \mathsf{Align}(\widetilde{x}, \widetilde{y})\}) \\ &= \min(c_{\mathsf{i}} + d(x, \widetilde{y}), c_{\mathsf{d}} + d(\widetilde{x}, y), c_{\mathsf{s}}(x_{|x|}, y_{|y|}) + d(\widetilde{x}, \widetilde{y})) \end{aligned}$$

Q. 7 Supposons avoir implanté récursivement la fonction d en utilisant les équations de la question précédente. On note alors $c_{n,m}$ le nombre d'appels de fonction induit par de telles équations sur des mots d'entrées x et y de taille n et m (on admet que le nombre d'appels dépend uniquement de la taille des mots et non pas de leurs lettres).

a) Donner une définition récursive de $(c_{n,m})_{(n,m)\in\mathbb{N}^2}$.

b) Montrer que $\forall n \in \mathbb{N}, c_{n,n} \geqslant 3^n$.

Solution

On remarque tout d'abord que :

$$\begin{aligned} c_{0,0} &= 1 \\ c_{0,m} &= 1 + c_{0,m-1} && \text{pour } m \in \mathbb{N}^* \\ c_{n,0} &= 1 + c_{n-1,0} && \text{pour } n \in \mathbb{N}^* \\ c_{n,m} &= 1 + c_{n,m-1} + c_{n-1,m} + c_{n-1,m-1} && \text{pour } n \neq 0 \text{ et } m \neq 0 \end{aligned}$$

On montre alors le résultat suivant pour $(n,m) \in \mathbb{N}^2$: $P_{n,m} : c_{n,m} \geqslant 3^{\min(n,m)}$ par induction le long de l'ordre lexicographique (par exemple).

— $c_{0,0} = 1 = 3^{\min(0,0)}$

— Si $n = 0$ alors $c_{0,m} = 1 + c_{0,m-1} \geqslant 1 = 3^{\min(0,m)}$

— Si $m = 0$ alors $c_{n,0} = 1 + c_{n-1,0} \geqslant 1 = 3^{\min(n,0)}$

— Soit $(n,m) \in \mathbb{N}^2$ tel que $n \neq 0$ et $m \neq 0$. On suppose que $P_{n',m'}$ est vraie pour tout (n',m') inférieur à (n,m) pour l'ordre lexicographique. On a donc en particulier $P_{n-1,m}$, $P_{n,m-1}$ et $P_{n-1,m-1}$. En remarquant que $\min(n-1,m) \geqslant \min(n,m) - 1$, et que $\min(n-1,m-1) = \min(n,m) - 1$, $P_{n-1,m}$ donne $c_{n-1,m} \geqslant 3^{\min(n,m)}$ On a donc $c_{n,m} = 1 + c_{n,m-1} + c_{n-1,m} + c_{n-1,m-1} \geqslant 1 + 3^{\min(n,m)-1} + 3^{\min(n,m)-1} + 3^{\min(n,m)-1} \geqslant 3^{\min(n,m)}$

Par induction forte on en déduit que $\forall (n,m) \in \mathbb{N}^2$, $c_{n,m} \geqslant 3^{\min(n,m)}$, et en particulier $\forall n \in \mathbb{N}$, $c_{n,n} \geqslant 3^n$.

Q. 8 Mettre en avant, informellement, les éléments clés justifiant de la pertinence d'un algorithme de programmation dynamique pour le calcul de $d(x,y)$.

Solution

Une équation récursive calculant un nombre $n \times m$ de valeurs en un temps au moins exponentiel, dû à de nombreux recalculs.

5. Programmation dynamique en C

Pour les questions qui suivent, on considère fixé $(x,y) \in \Sigma^* \times \Sigma^*$ un couple de mots de longueurs respectives n et m, et $(\bar{x},\bar{y})$ un alignement de (x,y) de longueur l. De plus, pour $i \in [\![0,n]\!]$ et $j \in [\![0,m]\!]$, on introduit la notation suivante :

$$\mathsf{D}(i,j) = d(x_{[\![1,i]\!]}, y_{[\![1,j]\!]})$$

Q. 9 Donner la définition récursive de $\mathsf{D}(i,j)$ induite par les résultats des questions précédentes.

Structure matricielle en C. Afin d'implanter en C et par programmation dynamique le calcul de $\mathsf{D}(i,j)$, nous devons nous munir d'une structure de matrice. On définit donc le type suivant :

```
23 struct matrix_c {
24   int  largeur;
25   int  hauteur;
26   int* contenu;
27 };
28 typedef struct matrix_c* matrix;
```

Dans une telle structure, le champ `largeur` représente le nombre de colonnes, le champ `hauteur` représente le nombre de lignes et le champ `contenu`, qui contient les lignes de la matrice mises bout à bout, enregistre les coefficients. Aussi si l'on représente une matrice $(M_{i,j})_{i\in[\![0,h-1]\!],j\in[\![0,l-1]\!]}$ au moyen d'une telle structure, `hauteur` vaut h, `largeur` vaut l, le tableau d'entiers `contenu` est de longueur `hauteur`$\times$`largeur` et ses l premières cases contiennent (dans cet ordre) $M_{0,0}$, $M_{0,1}$, …, $M_{0,l-1}$, les l cases suivantes contiennent (dans cet ordre) $M_{1,0}$, $M_{1,1}$, …, $M_{1,l-1}$, ….

Q. 10 Définir les trois fonctions :

- `matrix matrix_alloc(int h, int l, int init)`, permettant l'allocation d'une matrice de hauteur `h`, de largeur `l` et dont tout les coefficients sont initialisés à `init` ;
- `int matrix_get(matrix m, int i, int j)`, renvoyant le coefficient d'indice `i`, `j` de `m` ;
- `void matrix_set(matrix m, int i, int j, int val)`, permettant de modifier le coefficient d'indices `i`, `j` de la matrice `m` à la valeur `val`.

Solution

```
30 matrix matrix_alloc(int n, int m, int init) {
31   matrix res = (matrix)malloc(sizeof(struct matrix_c));
32   int *content = (int *)malloc(n * m * sizeof(int));
33   for (int i = 0; i < n * m; i++) {
34     content[i] = init;
35   }
36   res->largeur = m;
37   res->hauteur = n;
38   res->contenu = content;
39   return res;
40 }
41 int matrix_get(matrix m, int i, int j) {
42 return m->contenu[i * m->largeur + j];
43 }
44
45 void matrix_set(matrix m, int i, int j, int val) {
46   m->contenu[i * m->largeur + j] = val;
47 }
```

Q. 11 Proposer une fonction C, `int* dist(char* x, char* y)` prenant en arguments deux chaînes de caractères `x` et `y`, et retournant la matrice des $(\mathsf{D}(i,j))_{(i,j)\in[\![0,n]\!]\times[\![0,m]\!]}$ où n est la longueur de `x` et m est la longueur de `y`. On pourra supposer définies :

- deux constantes `int c_i` et `int c_d` représentant respectivement c_{i} et c_{d} ;
- une fonction `int c_s (char a, char b)` permettant le calcul de $c_{\mathsf{s}}(\mathsf{a},\mathsf{b})$;

- une fonction `int min3(int a, int b, int c);` calculant le minimum de 3 valeurs entières.

On rappelle que la fonction `int strlen(char* s)` permet de calculer la longueur de la chaîne de caractères `s`.

Solution

```
160 matrix dist_bis(char* x, char* y) {
161   int n = strlen(x);
162   int m = strlen(y);
163   matrix res = matrix_alloc(n+1, m+1, 0);
164   // ainsi on a déjà res(0,0)=0
165   // première ligne
166   for (int j = 1; j <= n ; j ++) {
167     matrix_set(res, 0, j, j*c_d);
168   }
169   // première colonne
170   for (int i = 1; i <= n ; i ++) {
171     matrix_set(res, i, 0, i*c_i);
172   }
173   //le coeur du tableau
174   for (int i = 1; i <= n ; i ++) {
175     for (int j = 1; j <= m; j ++) {
176       int val_i = matrix_get(res, i, j-1) + c_i; //insertion
177       int val_d = matrix_get(res, i-1, j) + c_d; //deletion
178       int val_s = matrix_get(res, i-1, j-1) + c_s(x[i-1], y[j-1]);//sub
179       matrix_set(res, i, j, min3(val_i,val_d, val_s));
180     }
181   }
182   return res;
183 }
```

Q. 12 Donner un ordre de grandeur, au moyen d'un $\mathcal{O}$ le plus précis possible, du coût en espace et en temps de la fonction `dist`.

Solution

En espace $\mathcal{O}(nm)$ en temps $\mathcal{O}(nm)$.

Dans le reste de cet exercice les implantations ne sont plus demandées en C, on vous demandera uniquement de fournir des algorithmes en pseudo-code de bonne qualité.

Q. 13 Proposer un algorithme en pseudo-code, prenant en arguments deux mots x et y ainsi que la matrice $(\mathrm{D}(i,j))_{(i,j)\in[\![0,n]\!]\times[\![0,m]\!]}$ (celle calculée par `dist`) qui calcule un alignement optimal $(\bar{x},\bar{y})$ de (x,y).

Solution

Intuitivement il s'agit de "remonter les flèches" dans le tableau D, en partant de la case (n,m) et jusqu'à $(0,0)$, la "flèche" indiquant l'opération (insertion, deletion ou suppression) qui a été retenue lors du calcul de D. On propose ci-dessous un code en C, même si on attendait du pseudo-code. La structure de cet algorithme ressemble à celle de l'interclassement de deux piles (ou listes)

triées, comme dans le tri fusion. AEF : à développer ? .

```c
void affiche_align_opti(char* x, char* y, matrix D){
  int n = strlen(x);
  int m = strlen(y);
  char* xb = (char*) malloc ((n+m+1)*sizeof(char)); // x barre
  char* yb = (char*) malloc ((n+m+1)*sizeof(char)); // y barre
  int k = n+m; //curseur dans xb et yb,
  xb[k]= '\0';
  yb[k]= '\0';
  k--;
  int i = n;
  int j = m;
  while( i>0 && j>0){
    if (matrix_get(D,i,j) == (matrix_get(D,i-1,j) + c_i) ){//cas insertion
      xb[k] = x[i-1];
      yb[k] = '_';
      i = i-1;
    }
    else if (matrix_get(D,i,j) == (matrix_get(D,i,j-1) + c_d)){//deletion
      xb[k] = '_';
      yb[k] = y[j-1];
      j = j-1;
    }
    else{//cas d'une substitution
      xb[k] = x[i-1];
      yb[k] = y[j-1];
      i = i-1;
      j = j-1;
    }
    k--;
  }
  while(i>0){ //néc j=0, "y est vide, il reste que des insertions"
    xb[k] = x[i-1];
    yb[k] = '_';
    k--;
    i = i-1;
  }
  while(j>0){ //néc i=0, "y est vide, il reste que des deletions"
    xb[k] = '_';
    yb[k] = y[j-1];
    k--;
    j = j-1;
  }
  printf("xb : %s\nyb : %s\n", xb+k+1, yb+k+1);
}
```

6. Amélioration spatiale

Q. 14 Proposer un algorithme prenant en arguments deux chaînes de caractères `x` et `y`, et calculant la distance d'édition de `x` à `y`, votre algorithme devra avoir une complexité temporelle de l'ordre de celle `dist`, mais ayant une complexité spatiale de l'ordre de $\Theta(m)$ où m est la taille de la chaîne de caractères `y`.

Solution

Algorithme 1 : Dist en espace mémoire réduit

Entrée : x et y deux mots
Sortie : la valeur minimale d'un alignement de (x, y)

$n \leftarrow |x|$;
$m \leftarrow |y|$;
$T_1 \leftarrow$ tableau de $m+1$ entiers, indicé par $[\![0, m]\!]$;
$T_2 \leftarrow$ tableau de $m+1$ entiers, indicé par $[\![0, m]\!]$;
pour $j \in [\![1, m]\!]$ **faire**
    $T_1[j] \leftarrow c_d * j$;
pour $i \in [\![1, m]\!]$ **faire**
    $T_2[0] \leftarrow c_i * i$;
    pour $j \in [\![1, m]\!]$ **faire**
        $v_{ins} \leftarrow T_1[j] + c_i$ $i-1, j$;
        $v_{del} \leftarrow T_2[j] + c_d$ $i, j-1$;
        $v_{sub} \leftarrow T_1[j-1] + c_s(x_i, y_j)$ $i-1, j-1$;
        $T_2[j] \leftarrow \min(v_{ins}, v_{del}, v_{sub})$;
    Échanger T_1 et T_2 ;
retourner $T_1[m]$

Nous avons donc amélioré la complexité spatiale du calcul de la distance d'édition, toutefois cette amélioration a un coût : il n'est plus possible de reconstruire l'alignement optimal, seulement d'obtenir sa valeur. Dans les sections suivantes on essaie de retrouver le meilleur des deux mondes : un calcul de l'alignement optimal avec une complexité mémoire en $\Theta(m)$ et une complexité temporelle en $\Theta(nm)$.

7. Une approche "diviser-pour-régner"

On va utiliser ici la méthode diviser-pour-régner. Grosso modo, on va décomposer respectivement x en $x^1 \cdot x^2$ et y en $y^1 \cdot y^2$, puis calculer récursivement $(\bar{s}, \bar{t})$ un alignement optimal de (x^1, y^1) et $(\bar{u}, \bar{v})$ un alignement optimal de (x^2, y^2), et enfin les concaténer, c'est-à-dire qu'on retournera $(\bar{s} \cdot \bar{u}, \bar{t} \cdot \bar{v})$, On ne procédera à ce découpage de x et y que lorsque $|x| \geqslant 2$ et $|y| \geqslant 1$, les autres cas seront à traiter directement, sans appels récursifs.

Q. 15 Montrer qu'il existe $(\overline{s}, \overline{t})$ un alignement optimal de (x^1, y^1) et $(\overline{u}, \overline{v})$ un alignement optimal de (x^2, y^2) tels que $(\overline{s} \cdot \overline{u}, \overline{t} \cdot \overline{v})$ n'est pas un alignement optimal de $(x^1 \cdot x^2, y^1 \cdot y^2)$.

Solution

Par exemple avec $x =$ `bateaux`, décomposé en $x^1 =$ `bat` et $x^2 =$ `eau` et $y =$ `eaux`, décomposé en $y^1 =$ `ea` et $y^2 =$ `ux`, on a d'une part $(\overline{s}, \overline{t}) =$ (`_bat`, `e_a_`) un alignement optimal de (x^1, y^1) de coût 9, et d'autre part $(\overline{u}, \overline{v}) =$ (`eau_`, `__ux`) un alignement optimal de (x^2, y^2) de coût 9,

pourtant $(\overline{s}\cdot\overline{u},\overline{t}\cdot\overline{v})$ n'est pas un alignement optimal de (x,y) car il est de coût $9+9=18$ alors que (bateau_, ___eaux) est de coût 12.

La réponse précédente justifie qu'on ne peut se contenter d'une approche "diviser-pour-régner" classique, où l'on couperait les deux chaînes en leur milieu.

Coupure. Soient x et y deux mots sur Σ de longueurs respectives $n \geqslant 2$ et $m \geqslant 1$. Soit $i \in [\![1,n]\!]$ et $j \in [\![1,m]\!]$ on dit que j est une **coupure** de (x,y) associée à i si et seulement s'il existe $(\bar{s},\bar{t})$ un alignement de $(x_{[\![1,i]\!]}, y_{[\![1,j]\!]})$ et $(\bar{u},\bar{v})$ un alignement de $(x_{[\![i+1,n]\!]}, y_{[\![j+1,m]\!]})$ tels que $(\bar{s}\cdot\bar{u},\bar{t}\cdot\bar{v})$ est un alignement minimal de (x,y).

On suppose fournie, et utilisable dans vos algorithmes une fonction coupure qui à partir d'un couple de mots (x,y) de longueurs respectives $n \geqslant 2$ et $m \geqslant 1$, renvoie un indice j^* qui est une coupure de (x,y) associée à $i^* = \left\lfloor \frac{|x|}{2} \right\rfloor$. On suppose de plus que cette fonction a une complexité algorithme pire cas majorée par $K_0\, nm$ (pour une certaine constante $K_0 \in \mathbb{R}^{+,*}$) sur deux entrées x et y de tailles respectives n et m.

On se concentre dans cette partie sur le cœur récursif de l'algorithme d'alignement par diviser-pour-régner, on suppose donc connue et utilisable une fonction base telle que base(x,y) calcule un alignement optimal de deux mots x et y lorsque $|x| \leqslant 1$ en une complexité majorée par $K_1\,|y|$.

Q. 16 Définir alors un algorithme alignementDpr calculant un alignement optimal de (x,y) par une approche "diviser-pour-régner".

Solution

Algorithme 2 : alignementDpr

Entrée : x et y deux mots
Sortie : un alignement optimal de (x,y)

```
si |x| ⩽ 2 alors
    retourner Base(x, y)
sinon
    i* ← ⌊|x|/2⌋ ;
    j* ← coupure(x, y) ;
    (s̄, t̄) ← alignementDpr(x_[1,i*], y_[1,j*]) ;
    (ū, v̄) ← alignementDpr(x_[i*+1,|x|], y_[j*+1,|y|]) ;
    retourner (s̄ · ū, t̄ · v̄)
```

Q. 17 Soit $C_{n,m}$ la suite donnant la complexité pire cas de l'algorithme alignementDpr sur des entrées de taille n et m.

- Montrer qu'il existe une constante K telle que $\forall p \in \mathbb{N}, \forall m \in NN, C_{2^p,m} \leqslant K 2^{p+1} m$.
- En admettant la croissance de la suite $(C_{n,m})_{(n,m)\in\mathbb{N}^2}$, donner une majoration de $(C_{n,m})_{(n,m)\in\mathbb{N}^2}$ au moyen d'un $\mathcal{O}$.

Solution

Notons $v_{p,m}$ la suite $C_{2^p,m}$ et remarquons que :

— $v_{p,m} \leqslant K2^p m + v_{p-1,g} + v_{p-1,d}$ pour des g et d tels que $g + d = m$. Or par hypothèse d'induction : $v_{p-1,g} \leqslant K2^p g$ et $v_{p-1,s} \leqslant K2^p d$ donc : $v_{p,m} \leqslant K2^p m + K2^p g + K2^p d = K2^p m + K2^p(g + d) = K2^p m + K2^p m = K2^{p+1} m$.

On choisit K convenablement pour que cela passe en 0. On a donc une complexité en $\mathcal{O}(nm)$ et un coût mémoire qui est celui de coupure

On remarque que la complexité spatiale de alignementDpr est essentiellement due à celle de coupure. Dans la fin de cet exercice on s'intéresse donc à une définition de la fonction coupure de complexité spatiale plus faible que $\Theta(nm)$, et de complexité temporelle en $\mathcal{O}(nm)$.

8. Le problème de la coupure

Pour $i \in [\![\frac{|x|}{2}, n]\!]$ et pour $j \in [\![0, m]\!]$, on définit $\mathcal{C}(i, j)$ comme suit.

$$\mathcal{C}(i,j) = \left\{ k \in [\![0,m]\!] \;\middle|\; k \text{ est une coupure de } (x_{[\![1,i]\!]}, y_{[\![1,j]\!]}) \text{ associée à } \left\lfloor \frac{|x|}{2} \right\rfloor \right\}$$

On cherche donc à calculer un élément de $\mathcal{C}(n, m)$. On remarque que $\forall j \in [\![1, m]\!],\ j \in \mathcal{C}(\lfloor \frac{|x|}{2} \rfloor, j)$.

Q. 18 Soit $i \in [\![\frac{|x|}{2} + 1, n]\!]$. Soit $j \in [\![0, m]\!]$. Monter que si $j > 0$ et $\mathsf{D}(i, j) = \mathsf{D}(i, j-1) + c_{\mathsf{i}}$, alors $\mathcal{C}(i, j-1) \subseteq \mathcal{C}(i, j)$

Solution

Soit $j \in [\![0, m]\!]$. On suppose que $\mathsf{D}(i,j) = \mathsf{D}(i,j-1) + c_{\mathsf{i}}$.
Cela signifie que pour tout $(\bar{p}, \bar{q}) \in \mathsf{Al}(i, j-1)$, alors $(\bar{p} \cdot -, \bar{q} \cdot y_j) \in \mathsf{Al}(i, j)$.

Soit $k \in \mathcal{C}(i, j-1)$. Par définition, il existe

- $(\bar{s}, \bar{t})$ un alignement optimal de $(x_{[\![1,i^*]\!]}, y_{[\![1,k]\!]})$
- $(\bar{u}, \bar{v})$ un alignement optimal de $(x_{[\![i^*+1,i]\!]}, y_{[\![k+1,j-1]\!]})$

tels que $(\bar{s} \cdot \bar{u}, \bar{t} \cdot \bar{v})$ est un alignement optimal de $(x_{[\![1,i]\!]}, y_{[\![1,j-1]\!]})$,
donc tels que $\left(\bar{s} \cdot (\bar{u} \cdot -), \bar{t} \cdot (\bar{v} \cdot y_j)\right)$ est un alignement optimal de $(x_{[\![1,i]\!]}, y_{[\![1,j]\!]})$.

Il suffit donc de montrer que $\left(\bar{u} \cdot -, \bar{v} \cdot y_j\right)$ est un alignement optimal de $(x_{[\![i^*+1,i]\!]}, y_{[\![k+1,j]\!]})$ pour retrouver la définition de k est une coupure pour $(x_{[\![1,i]\!]}, y_{[\![1,j]\!]})$.

S'il existait $(\bar{a}, \bar{b})$ un alignement de $(x_{[\![i^*+1,i]\!]}, y_{[\![k+1,j]\!]})$ tel que $C(\bar{a}, \bar{b}) < C(\bar{u} \cdot -, \bar{v} \cdot y_j)$,
alors on aurait $C(\bar{s} \cdot \bar{a}, \bar{t} \cdot \bar{b}) < C(\bar{s} \cdot (\bar{u} \cdot -), \bar{t} \cdot (\bar{v} \cdot y_j))$, autrement dit $(\bar{s} \cdot \bar{a}, \bar{t} \cdot \bar{b})$ serait un alignement de $(x_{[\![1,i]\!]}, y_{[\![1,j]\!]})$ de coût strictement moindre que l'optimum, ce qui est absurde.

On a en fait de même les deux affirmations suivantes.

- Si $\mathsf{D}(i,j) = \mathsf{D}(i-1,j) + c_{\mathsf{d}}$, alors $\mathcal{C}(i-1, j) \subseteq \mathcal{C}(i, j)$
- Si $\mathsf{D}(i,j) = \mathsf{D}(i-1,j-1) + c_{\mathsf{s}}(x_i, y_j)$ alors $\mathcal{C}(i-1, j-1) \subseteq \mathcal{C}(i, j)$

Q. 19 Conclure, en expliquant comment calculer coupure(x, y) avec une complexité temporelle pire cas en $\Theta(nm)$ et une complexité mémoire de l'ordre de $\Theta(m)$.