

Mini-Projet d'Algorithmique

VALETTE Manon – BEN KIRANE Malik

Novembre 2016

1 Partie théorique

Partie 1

Question 1.1 On peut énumérer tous les sous-ensembles C_k de paires (x_i, y_j) pour deux séquences $X = (x_i)_{1 \leq i \leq d}$ et $Y = (y_i)_{1 \leq i \leq d}$ et tester si se sont des alignements. Il y a d^2 paires possibles, soit la paires est dans C_k , soit elle ne l'est pas, ce qui nous donne 2^{d^2} sous-ensembles C_k tester.

Question 1.2 Soit M un alignement de X et Y . Supposons que $(x_m, y_n) \notin M$ et qu'il existe $i \leq m, j \leq n$ tels que $(x_i, y_n) \in M$ et $(x_m, y_j) \in M$ (i.e. x_m et y_n apparaissent dans M). Nécessairement $i < m$ et $j < n$ puisque x_m et y_n apparaissent au plus une fois. De plus comme il n'y a pas de croisement dans M et que $i < m, n < j$. Contradiction.

Notation Par abus, pour un alignement M de deux séquences $X = (x_i)_{0 \leq i \leq m}$ et $Y = (y_j)_{0 \leq j \leq n}$, on notera $x_k \in M$ lorsqu'il existe j , tel que $(x_k, y_j) \in M$, symétriquement on pourra aussi écrire $y_l \in M$, et on déduit les négations respectives.

Question 1.3 Les trois cas de figures suivants se déduisent de la réflexion précédente :

- $x_m \notin M$
- $y_n \notin M$
- $(x_m, y_n) \in M$

Notation Pour un alignement M de deux séquences $X = (x_i)_{1 \leq i \leq m}$ et $Y = (y_i)_{1 \leq i \leq n}$. On note $M_{i,j}$ le sous-alignement $\{(x_k, y_l) \in M \mid k \leq i, l \leq j\}$. On convient que $M = M_{m,n}$. Il est évident que $M_{i,j}$ est un alignement de ses sous-séquences correspondantes i.e. $(x_k)_{k \leq i}$ et $(y_l)_{l \leq j}$.

Question 1.4 Considérons M^* un alignement de coût minimal des séquences $(x_i)_{1 \leq i \leq m}$ et $(y_j)_{1 \leq j \leq n}$. Un tel alignement existe puisque l'ensemble des alignements pour une séquence donnée est fini.

1. lorsque $(x_m, y_n) \in M^*$

$$F(m, n) = f(M^*) = \underbrace{\sum_{(x_i, y_j) \in M_{m-1, n-1}^*} \delta_{x_i y_j} + \sum_{x_i \notin M_{m-1, n-1}^*} \delta_{gap} + \sum_{y_j \notin M_{m-1, n-1}^*} \delta_{gap} + \delta_{x_m y_n}}_{f(M_{m-1, n-1}^*)}$$

Or $M_{m-1, n-1}^*$ est de coût minimal pour ses sous-séquences coresspondantes. Donc :

$$F(m, n) = F(m-1, n-1) + \delta_{x_m y_n}.$$

2. lorsque $x_m \notin M^*$, $M_{m-1,n}^*$ est optimal pour ses sous-séquences correspondantes. Donc :

$$F(m, n) = F(m-1, n) + \delta_{gap}$$

3. lorsque $y_n \notin M^*$, de même $M_{m,n-1}^*$ est optimal et :

$$F(m, n) = F(m, n-1) + \delta_{gap}$$

Question 1.5 Afin de d'optimiser le coût d'un alignement il suffit de prendre la plus petite valeur des trois cas de figures. Soit pour $i \geq 1, j \geq 1$:

$$F(i, j) = \min \left\{ F(i-1, j-1) + \delta_{x_i y_j}, F(i-1, j) + \delta_{gap}, F(i, j-1) + \delta_{gap} \right\}$$

Question 1.6 Soit $i \in \{1 \dots m\}$. Tout alignement $M_{i,0}$ est vide. Donc,

$$F(i, 0) = \underbrace{\sum_{(x_i, y_j) \in M_{i,0}^*} \delta_{x_i y_j} + \sum_{y_j \notin M_{i,0}^*} \delta_{gap} + \sum_{x_i \notin M_{i,0}^*} \delta_{gap}}_{=0} = i \delta_{gap}.$$

Par symétrie, $F(0, j) = j \delta_{gap}$ pour tout $j \in \{1 \dots n\}$.

Question 1.7 On convient que l'on dispose de la primitive MIN renvoyant la valeur minimale d'un n-uplet. On convient aussi d'une primitive **creerMatrice**(N,M,V) qui initialise une matrice de taille $N \times M$ à valeur unique V et dont les indices sont compris entre [0,0] et [N-1,M-1].

L'approche, ici, est de type programmation dynamique : MEMO-COUT1 ne calcul $C[i, j]$ ($F(i, j)$) que s'il n'a pas été calculé précédemment. COUT1 appel MEMO-COUT1 $m \times n$ fois soit une complexité temporelle en $\Theta(mn)$. Cette mémorisation nous donne une complexité spatiale en $\Theta(mn)$.

La matrice P correspond aux pénalités de correspondances $\delta_{x_i y_j}$ et **d-gap** à la pénalité δ_{gap} .

MEMO-COUT1(C : Matrice(m,n), P : Matrice(m,n), d-gap, i, j)

SI $C[i, j] < 0$ ALORS

RETOURNER MIN(MEMO-COUT1(C, i-1, j-1) + P[i, j],
MEMO-COUT1(C, i-1, j) + d-gap,
MEMO-COUT1(C, i, j-1) + d-gap)

SINON RETOURNER $C[i, j]$

COUT1(P : Matrice(m,n), d-gap)

C <- creerMatrice(m+1,n+1,-1)

$C[0,0] \leftarrow 0$

POUR i = 1..m FAIRE

$C[i,0] \leftarrow i * d\text{-gap}$

POUR j = 1..n FAIRE

$C[0,j] \leftarrow j * d\text{-gap}$

POUR i = 1..m FAIRE

POUR j = 1..n FAIRE

$C[i, j] = \text{MEMO-COUT1}(C, P, d\text{-gap}, i, j)$

RETOURNER $C[m, n]$

Question 1.8 On dispose des $C[i, j]$: matrice globale des coûts des sous-alignements optimaux $M_{i,j}^*$. La liste de paire M , initialisée à la liste vide au début de l'algorithme est construite récursivement par REC-SOL1. Elle correspond à un alignement optimal pour la fonction coût f .

P et $d\text{-gap}$ ont la même sémantique qu'à la question 1.7.

```

REC-SOL1(M : liste de paires, i, j, P, d-gap)
  SI i = 0 OU j = 0 ALORS TERMINER
  SI C[i,j] = C[i-1,j-1] + P[i,j] ALORS
    M.append((i,j))
    REC-SOL1(M, i-1, j-1, P, d-gap)
  SINON SI C[i,j] = C[i-1,j] + d-gap ALORS
    REC-SOL1(M, i-1, j, P, d-gap)
  SINON REC-SOL1(M, i, j-1, P, d-gap)

```

```

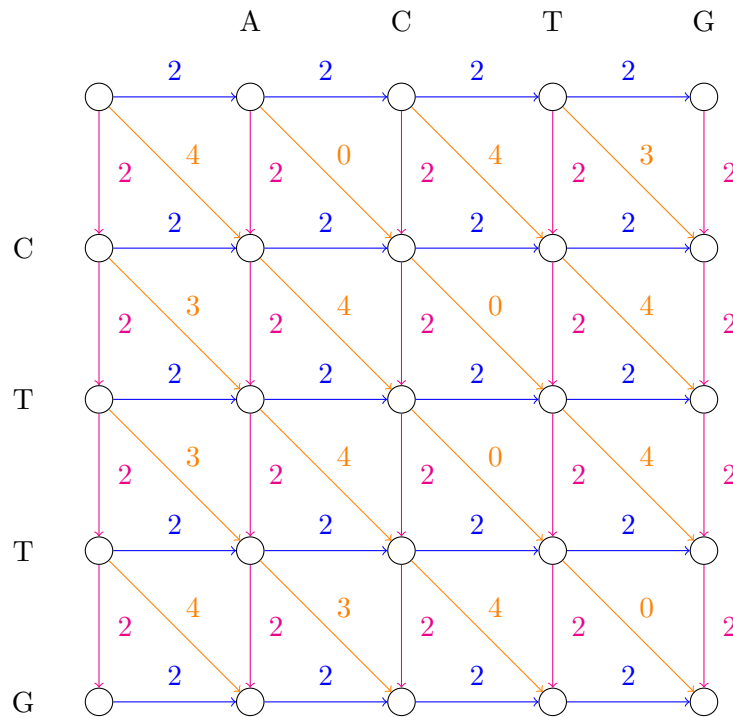
SOL1(m, n, P, d-gap)
  M <- creerListeVide()
  REC-SOL1(M, m, n, P, d-gap)
  RETOURNER M

```

REC-SOL1(m, n) à une complexité temporelle en $O(\max(m, n))$. Globalement cela nous donne toujours une complexité en $\Theta(mn)$. La complexité spatiale globale ne varie pas non plus ($\Theta(mn)$) puisque M occupe asymptotiquement au plus $O(m + n)$ espace.

Partie 2

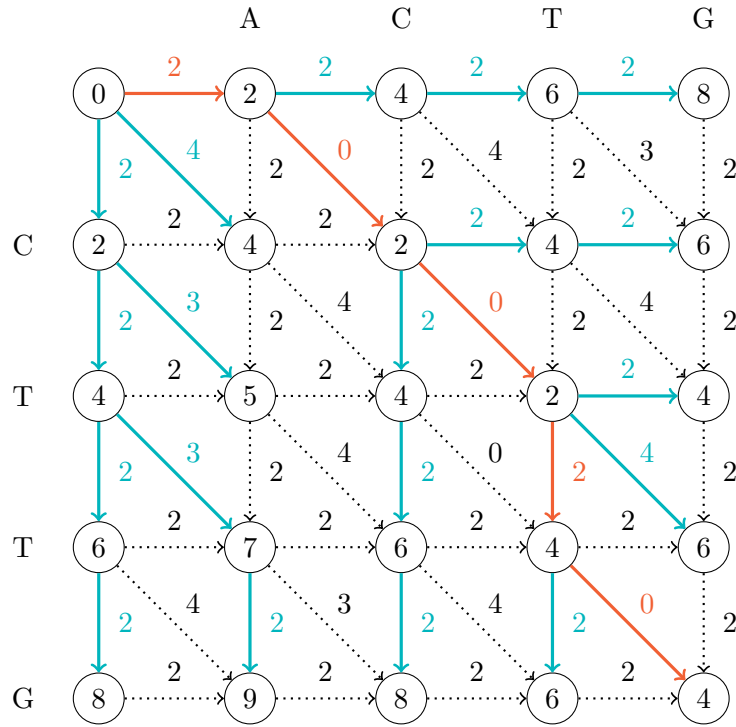
Question 2.1 Ci-dessous le graphe correspondant à l'exemple proposé.



Question 2.2 (facultatif)

Question 2.3 Pour déterminer un plus court chemin de $(0,0)$ à (m,n) dans G_{XY} , on peut utiliser l'algorithme de Dijkstra, dont la complexité est en $O((N + M) \log N)$ (pour un graphe à N sommets et M arcs).

L'arborescence des plus courts chemins obtenue avec l'algorithme de *Dijkstra* appliqué au graphe obtenu à la question 2.1 est représentée sur la figure ci-après par les arcs colorés, dont ceux en rouge constituent un plus court chemin de $(0,0)$ à (m,n) .



L'alignement optimal correspondant est donc :

–	A	C	T	–	G
–	–	C	T	T	G

Question 2.4 La complexité spatiale ne varie pas de celle proposée par l'algorithme de la partie 1, soit $\Theta(nm)$, l'algorithme de *Dijkstra* nous donne une arborescence des chemins de coûts minimum en $O(nm \log(nm))$ pour notre problème ($N = nm$ et $M = (n - 1)(m - 1) + n(m - 1) + m(n - 1)$ avec les notations de la question 2.3), soit une moins bonne complexité temporelle qu'à la partie 1.

Partie 3

Question 3.1 Si on compare deux séquences de longueurs d identiques (pire cas), les algorithmes des parties précédentes ont besoins à une constante prêt (1 octet) de d^2 espace mémoire que l'on note T_{mem} , soit $d = \sqrt{T_{mem}}$. Le tableau ci-dessous nous donne quelques applications numériques.

$T_{mem} =$	8Go	16Go	32Go
$d \approx$	89K	126K	179K

Question 3.2 On convient que l'on dispose d'un algorithme $P(i, j)$, cachant une structure de données spécifique, donnant les pénalités de correspondances $\delta_{x_i y_j}$ tel qu'en complexité spatiale, il ne dépend que de la taille des alphabets utilisés pour coder les séquences $X = (x_i)_{0 \leq i \leq m}$ et $Y = (y_j)_{0 \leq j \leq n}$ que l'on souhaite comparer. Entre autre on ne stocke pas les pénalités de correspondances nulles.

Pour COUT2, on est toujours sur une approche type programmation dynamique : le calcul de $F(i, j)$ par l'algorithme auxiliaire MEMO-COUT2 ne se fait que s'il n'a pas déjà été fait, l'argument l qui est passée en argument indique laquelle des deux lignes $i - 1$ ($l=0$) ou i ($l=1$ par exemple) est nécessaire pour le calcul de $F(i, j)$.

L'algorithme auxiliaire MAJ-MEMO, auquel on passe en argument des références aux tableaux de mémoïsation, fait la transition avec la paire de lignes suivante.

```

MAJ-MEMO(T0, T1, i)
  T0 <- T1
  T1[0] <- i * d-gap
  POUR j = 1..n FAIRE
    T1[j] = -1

MEMO-COUT2(T0, T1, l, i, j)
  SI l = 0 ALORS RETOURNER T0[j]
  SI T1[j] < 0 ALORS
    RETOURNER MIN(MEMO-COUT2(T0, T1, 0, i, j-1) + P(i, j),
                  MEMO-COUT2(T0, T1, 1, i, j-1) + d-gap,
                  MEMO-COUT2(T0, T1, 0, i, j) + d-gap)
  SINON RETOURNER T1[j]

COUT2(i, j)
  T0[0..j] : tableau
  T1[0..j] : tableau
  T0[0] = 0
  T1[0] = d-gap
  POUR l = 1..j FAIRE
    T0[l] = l * d-gap
  POUR l = 1..i FAIRE
    T1[l] = -1
  POUR k = 1..i FAIRE
    POUR l = 1..j FAIRE
      MEMO-COUT2(T0, T1, 1, k, l)
    MAJ-MEMO(T0, T1, k+1)
  RETOURNER T1[j]
```

La complexité temporelle de $\text{COUT2}(i, j)$ est la même que celle de $\text{COUT1}(i, j)$ i.e. $\Theta(ij)$. Cependant, si on note $cs(P(i, j))$ la complexité spatiale de P , la complexité spatiale de COUT2 est $\Theta(j + cs(P(i, j)))$. En supposant que $P(i, j)$ est asymptotiquement majorée par σ^2 avec σ la taille de l'alphabet utilisé pour encoder nos séquences, ce qui est une hypothèse convenable avec l'hypothèse faite sur la complexité spatiale de P , $\text{COUT2}(i, j)$ est en $O(\sigma^2 + j)$.

Question 3.3 COUT2 suit le même principe que COUT1 en translatant le problème au sous graphe induit par le sous-ensemble de sommets $\{(k, l) \mid i \leq k \leq m \text{ et } j \leq l \leq n\}$.

```

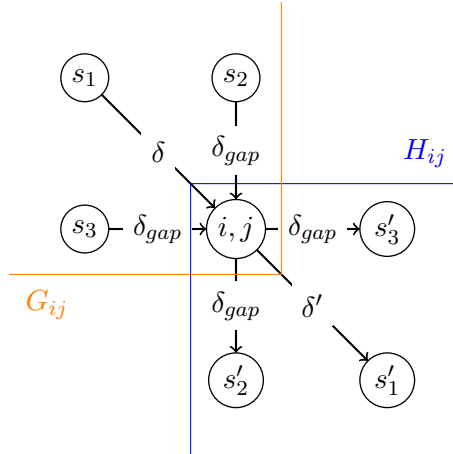
COUT2BIS(i, j, m, n)
  T0[0..n-j] : tableau
  T1[0..n-j] : tableau
  T0[0] <- 0
  T1[0] <- d-gap
  POUR l = 1..n-j FAIRE
    T1[l] <- l * d-gap
  POUR k = 1..m-i FAIRE
    POUR l = 1..n-j FAIRE
      MEMO-COUT2(T0, T1, 1, k, l)
    MAJ-MEMO(T0, T1, k)
  RETOURNER T1[n-j]

```

Notations

- Pour $1 \leq i \leq m-1$ et $1 \leq j \leq n-1$ et deux séquences X et Y , on note G_{ij} (resp. H_{ij}) le sous-graphes de G_{XY} induit par le sous-ensemble de sommets $\{(k, l) \mid k \leq i \text{ et } l \leq j\}$ (resp. $\{(k, l) \mid k \geq i \text{ et } l \geq j\}$). On remarque que l'on a $G_{ij} \cap H_{ij} = (\{(i, j)\}, \{\})$.
- Pour un graphe $G = (S, A)$ orienté, valué et $s_o, s \in S$, on note $d_{G_{s_o}}(s)$ le coût du plus court chemin de s_o à s , et c_G la fonction valuation de G .
- Pour simplifier les notations on pose $d_{G_{XY}} = d_{G_{XY}(0,0)}$, $d_{G_{ij}} = d_{G_{ij}(0,0)}$ et $d_{H_{ij}} = d_{H_{ij}(i,j)}$. On remarque que $g(i, j) = d_{G_{i,j}}((i, j))$ et $h(i, j) = d_{H_{i,j}}((m, n))$.

Question 3.4 Les cas ou $(i, j) = (0, 0)$ ou (m, n) sont triviaux. Soit $1 \leq i \leq m-1$ et $1 \leq j \leq n-1$ tels que qu'un plus court chemin de $(0, 0)$ à (m, n) dans G_{XY} passe par le sommet (i, j) . Il existe $(s_{g_0} = (0, 0), \dots, s_{g_l} = (i, j))$ un plus court chemin de $(0, 0)$ à (i, j) dans G_{ij} et $(s_{h_0} = (i, j), \dots, s_{h_l} = (m, n))$ un plus court chemin de (i, j) à (m, n) dans H_{ij} .



Les sommets s_1, s_2, s_3 , (resp. s'_1, s'_2, s'_3) représentés sur la figure ci-dessus sont les seuls prédecesseurs (resp. successeurs) de (i, j) dans G_{XY} . Si on note $((0, 0), \dots, s, (i, j), s', \dots, (m, n))$

un plus court chemin de $(0,0)$ à (m,n) passant par (i,j) dans G_{XY} , on a nécessairement $s \in \{s_1, s_2, s_3\}$ et $s' \in \{s'_1, s'_2, s'_3\}$.

On peut alors – en utilisant les notations introduites – facilement montrer par récurrence finie sur $k \in \{0 \dots l\}$ l'équation (1). Les équations (2) et (3) se déduisent alors simplement en interprétant les notations.

$$d_{G_{XY}}(m, n) = d_{G_{ij}}(i, j) + \sum_{k=0}^{l-1} c_{H_{ij}}(s_{h_k}, s_{h_{k+1}}) \quad (1)$$

$$= d_{G_{ij}}(i, j) + d_{H_{ij}}(m, n) \quad (2)$$

$$= g(i, j) + h(i, j) \quad (3)$$

Question 3.5 Listons les complexités spatiales de l'ensemble des structures de données misent en jeu par $\text{SOL2}(0,0,\mathbf{m},\mathbf{n})$.

— Structures intraséques :

$$\begin{array}{l|l} X2a, Y2b & \Theta(1) \\ Y2a, F2a & \Theta(n) \\ X2b, F2b & \Theta(m) \end{array}$$

- Les appels de SOL1 sur $X2a$ et $Y2a$ avec $F2a$ ont une complexité spatiale en $\Theta(n)$.
- Les appels de SOL1 sur $X2b$ et $Y2b$ avec $F2b$ ont une complexité spatiale en $\Theta(m)$.
- Un pire appel en termes de mémoire de COUT2 se fait pour $i = 0$ et $j = \lceil \frac{m-n}{2} \rceil$. Soit une complexité spatiale en $O(\max(m, n))$ (étape 1).
- De même pour la complexité spatiale de COUT2BIS .

La complexité spatiale de l'appel $\text{SOL2}(0,0,\mathbf{m},\mathbf{n})$ est donc bien en $O(m + n)$.