BIN702 - Série d'exercices sur l'alignement multiple

Manuel Lafond

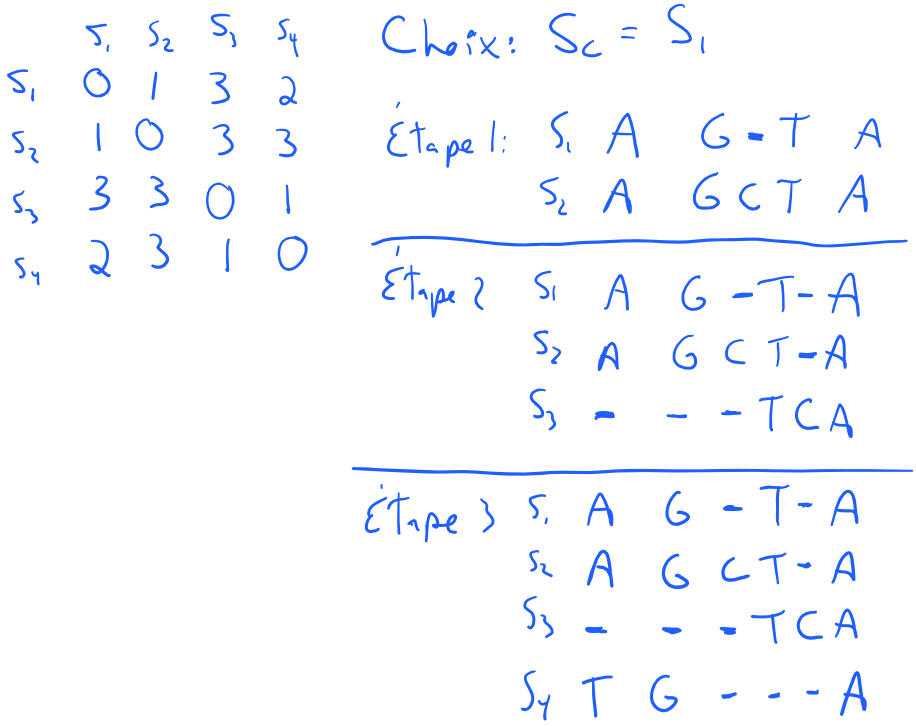
Exercice 1: Soient les séquences

$$S_1 = AGTA$$
$$S_2 = AGCTA$$
$$S_3 = TCA$$
$$S_4 = TGA$$

a. Considérez l'algorithme de centre-étoile vu en classe. On suppose qu'on choisit la séquence centrale qui minimise la somme des distances Levenshtein avec les autres, et on utilise la version minimisation du coût SP. On suppose un coût de 1 pour toute mutation et tout caractère aligné avec un gap (un gap avec un gap a un coût de 0). Quelle serait une séquence centrale possible ? Donnez le résultat de l'algorithme selon la séquence centrale choisie.

Solution. La figure suivante illustre la sélection de $S_c = S_1$ selon la somme de ses distances dans la matrice (on suppose ici que je ne fais aucune erreur !).

La construction de l'alignement centré à S_1 est aussi illustré.

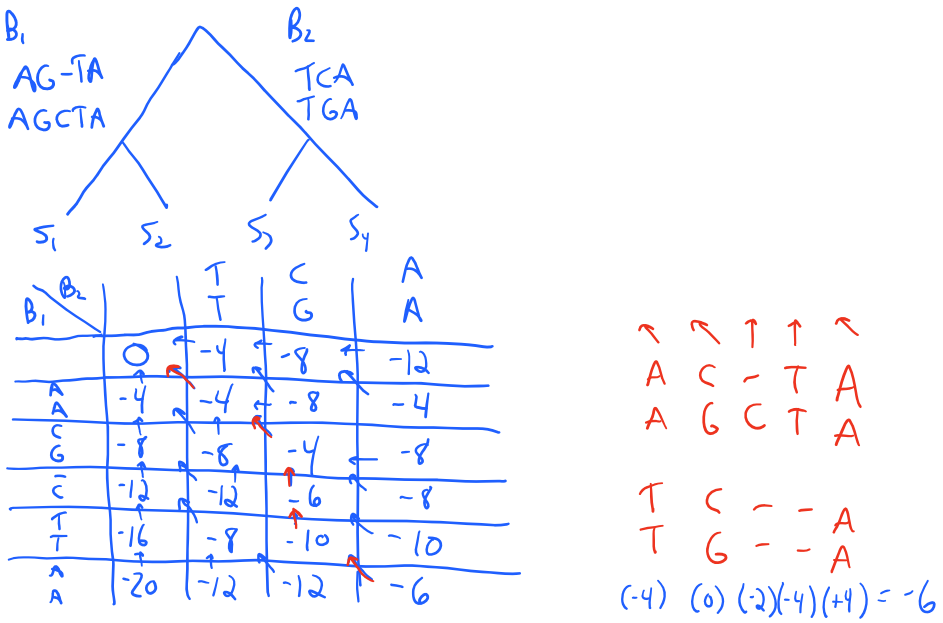


□

b. Considérez l'algorithme de Clustal pour maximiser le score SP, avec un score de $+1$ pour un match, -1 pour une mutation ou un caractère aligné avec un gap, et 0 pour un gap avec un gap. Supposez que vous avez l'arbre guide dans lequel S_1 et S_2 ont un parent commun, et S_3 et S_4 ont un autre parent commun (et ces deux parents sont joints par la racine).

   Donnez l'alignement multiple retourné par Clustal suite au parcours de l'arbre guide. Pour bien faire cet exercice, vous devriez construire la table de programmation dynamique pour combiner les deux alignements du 2ème niveau de l'arbre guide.

   Solution. La table de programmation dynamique pour le niveau de la racine est donné dans la figure suivante. Un alignement donné par Clustal est donné à droite.



□

Exercice 2: Écrivez les conditions initiales et la récurrence servant à calculer l'alignement multiple optimal entre 3 séquences S_1, S_2 et S_3.

Solution. Soit M la matrice de scores. On suppose $M[-,-] = 0$. Soit $V[i,j,k]$ le score d'alignement optimal entre $S_1[1..i], S_2[1..j]$ et $S_3[1..k]$.
On a $V[0,0,0] = 0$. On définit $V[i,j,k] = -\infty$ si on a $i < 0, j < 0$ ou $k < 0$. Pour $i,j,k \geq 0$ tels que $i+j+k > 0$, on doit énumérer toutes les façons d'avoir la dernière colonne de l'alignement multiple, en excluant le cas où il y a un gap partout.
Ceci mène à l'horreur suivante.

$$V[i,j,k] = \max \begin{cases} V[i-1,j,k] + M[S_1[i],-] + M[S_1[i],-] \\ V[i,j-1,k] + M[S_2[j],-] + M[S_2[j],-] \\ V[i,j,k-1] + M[S_3[k],-] + M[S_3[k],-] \\ V[i-1,j-1,k] + M[S_1[i],S_2[j]] + M[S_1[i],-] + M[S_2[j],-] \\ V[i-1,j,k-1] + M[S_1[i],S_3[k]] + M[S_1[i],-] + M[S_3[k],-] \\ V[i,j-1,k-1] + M[S_2[j],S_3[k]] + M[S_2[j],-] + M[S_3[k],-] \\ V[i-1,j-1,k-1] + M[S_1[i],S_2[j]] + M[S_1[i],S_3[k]] + M[S_2[j],S_3[k]] \end{cases}$$

□

Exercice 3: Quelle est la complexité de l'algorithme exact vu en classe qui calcule l'alignement exact optimal entre m séquences ? Vous pouvez supposer que toutes les séquences ont une longueur de n.
Notez qu'en classe, j'ai donné une borne de $n^m 2^m$. C'est presque ça, mais il manquait un petit détail dans cette analyse.

Solution. Si on a un ensemble de séquences $S_1, \ldots, S_m$, on doit calculer l'entrée $V[i_1, \ldots, i_m]$ pour chaque combinaison de
$i_1 \in \{1, \ldots, |S_1|\}, i_2 \in \{1, 2, \ldots, |S_2|\}, \ldots, i_m \in \{1, \ldots, |S_m|\}$. Le nombre d'entrées à calculer est $O(|S_1||S_2| \ldots |S_m|)$.
Pour une entrée spécifique, il faut boucler sur chaque vecteur de m bits comme discuté en classe. Il y en a $O(2^m)$ à énumérer. Pour chaque vecteur de m bits, il faut calculer le score induit par la dernière colonne correspondant à ce choix de bits. Ceci demande de calculer le score SP de cette colonne, ce qui demande de faire $O(m^2)$ additions (car il y a $O(m^2)$ paires de caractères à additionner). Donc, chaque entrée prend un temps $O(2^m m^2)$.
La complexité totale est $O(2^m m^2 |S_1||S_2| \ldots |S_m|)$.

□

Exercice 4: Dans l'algorithme de Clustal, nos arbres guide étaient toujours binaires. Supposons que l'arbre est plutôt ternaire, i.e. chaque noeud a trois enfants. Comment peut-on modifier Clustal pour supporter un arbre guide ternaire ?

Solution. Lorsqu'on atteint un noeud interne, on doit combiner trois alignments multiples en un seul. Il suffit d'utiliser la même technique vue en classe (convertir les colonnes en caractères de l'alphabet), mais d'utiliser l'algorithme exact pour alignment de trois séquences. □

Exercice 5 (Difficile)**:** Dans cette question, nous allons montrer que l'approche centre-étoile est une 2-approximation. Cette question est complèment optionnelle et ne vous préparera pas réellement à l'examen final.
Soit $S_1, \ldots, S_m$ un ensemble de m séquences. On veut minimiser le coût SP, qui est la somme des distances Levenshtein sur toutes les paires de séquences alignées. Soit A l'alignement multiple retourné par l'algorithme centre-étoile, et soit A^* l'alignement multiple optimal. Montrez que

$$cout(A) \leq 2cout(A^*)$$

où $cout(A)$ est le coût SP de l'alignement A (même chose pour $cout(A^*)$).
Notez que la distance Levenshtein satisfait l'inégalité triangulaire. C'est-à-dire, si $D(S_i, S_j)$ représente la distance entre S_i et S_j, alors pour S_i, S_j, S_k on a $D(S_i, S_k) \leq D(S_i, S_k) + D(S_k, S_i)$.

Plus précisément, si S_c est la séquence centrale, vous pouvez montrer que et que

$$cout(A) \leq m \sum_{i=1}^{m} D(S_c, S_i) \qquad \text{et} \qquad cout(A^*) \geq \frac{m}{2} \sum_{i=1}^{m} D(S_c, S_i)$$

Pour la première inégalité, vous pouvez utiliser le fait que grâce à l'inégalité triangulaire, on a $dist_A(S_i, S_j) \leq dist_A(S_i, S_c) + dist_A(S_c, S_j) = D(S_c, S_i) + D(S_c, S_j)$.

Solution. On va d'abord montrer que $cout(A) \leq m \sum_{i=1}^{m} D(S_c, S_i)$.
On a

$$
\begin{aligned}
cout(A) &= \sum_{i=1}^{m} \sum_{j=i+1}^{m} dist_A(S_i, S_j) \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} dist_A(S_i, S_j) \\
&\leq \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} (D(S_c, S_i) + D(S_c, S_j)) && \text{selon l'inégalité triangulaire} \\
&= \frac{1}{2} \left(\sum_{i=1}^{m} \sum_{j=1}^{m} D(S_c, S_i) + \sum_{i=1}^{m} \sum_{j=1}^{m} D(S_c, S_j) \right) \\
&= \frac{1}{2} \left(m \sum_{i=1}^{m} D(S_c, S_i) + m \sum_{j=1}^{m} D(S_c, S_j) \right) \\
&= \frac{1}{2} \left(2m \sum_{i=1}^{m} D(S_c, S_i) \right) \\
&= m \sum_{i=1}^{m} D(S_c, S_i)
\end{aligned}
$$

Pour l'autre inégalité, on a

$$
\begin{aligned}
cout(A^*) &= \sum_{i=1}^{m} \sum_{j=i+1}^{m} dist_{A^*}(S_i, S_j) \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} dist_{A^*}(S_i, S_j) \\
&\geq \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} D(S_i, S_j) && \text{car } D(S_i, S_j) \text{ est le minimum possible} \\
&\geq \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} D(S_c, S_j) && \text{par notre choix de } S_c \\
&= \frac{1}{2} m \sum_{j=1}^{m} D(S_c, S_j)
\end{aligned}
$$

□