Université de Sherbrooke - Département d'informatique
BIN702 - Algorithmes pour la bioinformatique (Automne 2023)

EXAMEN FINAL

Enseignant : Manuel Lafond Date : le 12 décembre 2023

- Cet examen d'une durée de 3 h 30 est individuel. **Je porterai une attention particulière au plagiat.**
- Vous devez remettre votre examen via turnin : `https://turnin.dinf.usherbrooke.ca/`.

  Vous devez remettre un fichier pdf contenant vos réponses (aucune forme spécifique n'est exigée).
- Toute forme de documentation est permise.
- L'examen est sur un total de 107 points. Si votre note dépasse 100, elle sera ramenée à 100. En d'autres termes, la note de votre examen intra sera $\min(P, 100)$, où P est le nombre de points accumulés dans l'examen.
- Vous pouvez **utiliser les résultats démontrés** en classe, dans les notes de cours et dans les exercices sans justification.
- Votre pseudo-code peut être de haut niveau, par exemple en phrases, dans la mesure où les étapes importantes sont claires et leur complexité justifiée de façon appropriée. Le **maximum de points** sera atteint si le code atteint une **complexité en temps optimale**.

QUESTION 1 : alignement multiple (15 points)

a. (10 points) Dans cette question, on considère la distance d'édition (donc pénalité de 1 pour une mutation ou une insertion/suppression). Soit les séquences:

$$S_1 = TTC \quad S_2 = TATC \quad S_3 = ATT \quad S_4 = TC$$

Un bienfaiteur vous fournit l'alignement global optimal entre chaque paire de séquences :

S_1	T-TC	S_1	-TTC	S_1	TTC	S_2	TATC	S_2	TATC	S_3	ATT
S_2	TATC	S_3	ATT-	S_4	-TC	S_3	-ATT	S_4	--TC	S_4	-TC

Donnez le résultat de l'alignement **centre-étoile** sur ces séquences. Vous devez clairement indiquer votre choix de séquence centrale, puis donner l'alignement résultant.

b. (5 points) Dans **l'algorithme exact** d'alignement multiple vu en classe, nous devions remplir une table de programmation dynamique à plusieurs dimensions. La récurrence permettant de calculer cette table considérait le maximum sur un nombre exponentiel de cas possibles. La représentation succincte de la récurrence était :

$$V[i_1, i_2, ..., i_n] = \max_{(b_1,b_2,...,b_n) \in \underbrace{\{0,1\}^n}_{vecteur\_bits}} (V[i_1 - b_1, i_2 - b_2, ..., i_n - b_n] + \text{score}(derniere\_colonne))$$

Décrivez brièvement ce que représente chacun des cas évalués dans la maximisation et pourquoi il y en a un nombre exponentiel.

QUESTION 2 : phylogénétique et méthodes par distances (13 points)

a. (9 points) Considérez la distance ultra-métrique D suivante:

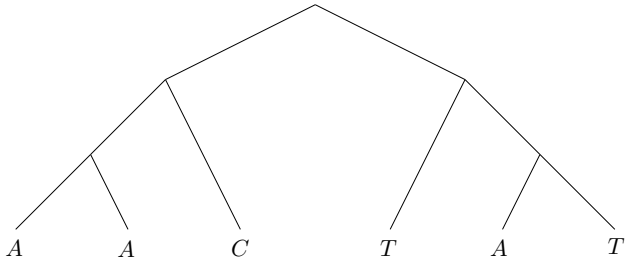
	A	B	C	D
A	0	6	4	10
B	6	0	6	10
C	4	6	0	10
D	10	10	10	0

Donnez l'arbre UPGMA ainsi que les longueurs d'arêtes satisfaisant les conditions d'ultra-métricité.

b. (4 points) Soit D, une distance ultra-métrique. Est-ce que D est nécessairement additive? Justifiez votre réponse.

QUESTION 3 : phylogénétique par caractères (15 points)

a. (7 points) Considérez l'arbre suivant, avec un caractère associé à chaque feuille.



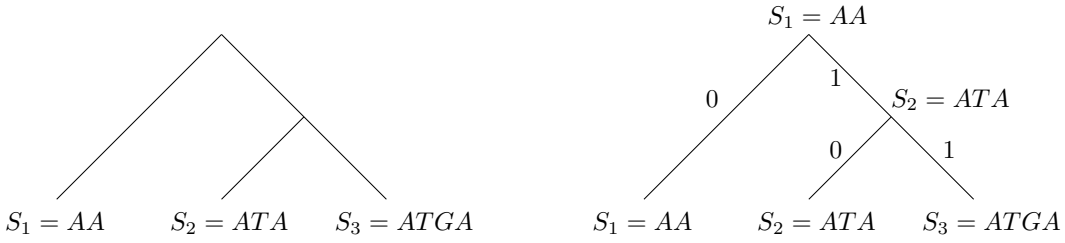
Donnez le résultat de l'algorithme vu en classe permettant de minimiser les changements aux branches. Vous devez donner les caractères candidats pour chaque noeud, puis donner un étiquetage optimal.

b. (8 points) Soit $S = \{S_1, \ldots, S_n\}$ un ensemble de séquences, possiblement de longueurs différentes. Soit T un arbre binaire dans lequel chaque feuille v est assignée à une séquence distincte $s(v) \in S$. Une *assignation* de T consiste à assigner une séquence $s(v)$ à chaque noeud interne v de T, avec **la contrainte que** $s(v)$ **doit être dans** S. Le coût d'une assignation s est

$$\sum_{uv \in E(T)} dist(s(u), s(v))$$

où *dist* est la distance Levenshtein.

Montrez comment on peut trouver une assignation de coût minimum en temps polynomial sur un arbre donné. Comme élément de réponse, vous pouvez donner une récurrence de programmation dynamique avec conditions initiales, ou bien un algorithme.



La figure ci-haut illustre un exemple d'instance. Gauche: une entrée avec $n = 3$ feuilles. Droite: une sortie possible dans laquelle chaque noeud interne est assigné à une des séquences de l'entrée. Les nombre représentent les coûts sur les branches.

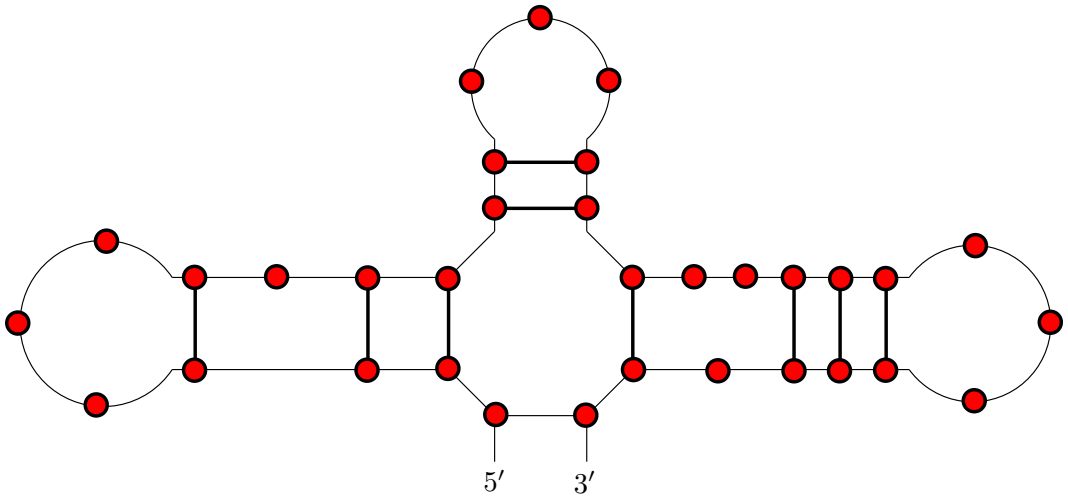
Suggestion. J'utiliserais une table $V[v, S_i]$, qui représente le coût minimum du sous-arbre enraciné en v avec la contrainte d'assigner S_i à v.

QUESTION 4 : structures secondaires d'ARN I (15 points)

a. (6 points) Considérez l'algorithme qui minimise la somme en énergie des sous-structures de l'ARN, où les énergies sont données par des fonctions arbitraires. Supposons, pour simplifier, que chaque sous-structure a un coût fixe (peu importe sa longueur) et que vous avez les fonctions d'énergie suivantes:

Tige-boucle: $eT(i,j) = 2$	Empilement: $eE(i,j) = -10$
Boucle interne: $eB(i,i',j',j) = 4$	Multi-boucle: $eM(i,i_1,\ldots,j_k,j) = 6$

Considérez la structure apparaissant sur la figure (les cercles représentent des nucléotides). Donnez le nombre d'occurrences de chaque sous-structure, puis donnez son énergie libre totale.



b. (9 points) Dans la structure secondaire d'une séquence S d'ARN, deux nucléotides consécutifs ne devraient pas partager d'appariement car ils sont déjà liés chimiquement. En d'autres termes, un nucléotide $S[i]$ ne devrait jamais être apparié avec le nucléotide suivant $S[i+1]$.

Étant donné une séquence d'ARN S et une liste P d'appariements possibles, donnez un algorithme en temps polynomial qui calcule le nombre maximum d'appariements avec la condition que si (i,j) sont appariés, alors $j > i+1$.

Comme élément de réponse, il est suffisant de donner une récurrence de programmation dynamique (avec conditions initiales).

QUESTION 5 : distance de réarrangements I (13 points)

Soit les deux génomes signés suivants.

$$G_1 = a^+ b^+ c^+ d^+ e^+$$
$$G_2 = a^- b^+ d^- c^- e^+$$

a. Exprimez ces deux génomes avec la notation des extrémités de gènes (avec les indices d pour "début" et f pour "fin").

b. Donnez la liste des adjacences de ces deux génomes.

c. Dessinez le graphe d'adjacences de ces deux génomes.

d. Donnez la distance DCJ de G_1 à G_2, avec justification.

Note : vous n'avez pas à donner une séquence d'opérations DCJ.

Vous devez choisir **deux (2) questions** parmi les questions 6-7-8-9. Si vous me remettez plus de deux réponses, je corrigerai seulement les deux premières. Indiquez clairement vos choix.

QUESTION 6 : un Clustal un peu plus rapide (18 points)

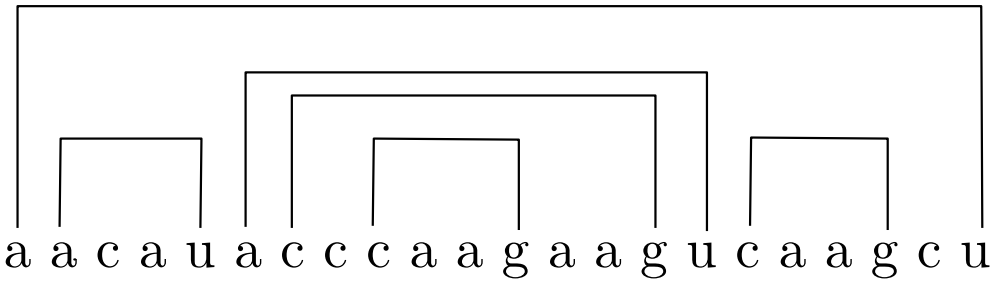
Rappelons que lorsque Clustal parcourt son arbre guide, il doit combiner deux alignements multiples $\mathcal{A}_1$ et $\mathcal{A}_2$ à chaque noeud visité. On suppose que $\mathcal{A}_1$ et $\mathcal{A}_2$ ont chacun n rangées.
On peut combiner ces alignements en interprétant chaque colonne de $\mathcal{A}_1$ et $\mathcal{A}_2$ comme un caractère distinct. Pour une colonne B_i de $\mathcal{A}_1$ et une colonne D_j de $\mathcal{A}_2$, on définit $M[B_i, D_j]$ comme le score la somme des paires (*sum-of-pairs*) de B_i et D_j, où le score de deux caractères a et b est donné par $M[a, b]$. Ensuite, on exécute l'alignement global ordinaire sur les colonnes en utilisant M comme matrice de scores.

a. (12 points) Calculer naïvement un seul $M[B_i, D_j]$ peut prendre un temps $O(n^2)$. Montrez que si l'alphabet est constant, donc $|\Sigma| \in O(1)$, on peut calculer $M[B_i, D_j]$ en temps $O(n)$.

b. (6 points) Supposez maintenant que chaque entrée $M[B_i, D_j]$ se calcule en temps $O(n)$. Supposez aussi que $\mathcal{A}_1$ et $\mathcal{A}_2$ ont chacun m colonnes. Donnez la complexité totale pour combiner $\mathcal{A}_1$ et $\mathcal{A}_2$ en utilisant la procédure décrite ci-haut. Votre analyse doit considérer le temps pour convertir les colonnes en caractères, calculer leurs scores, puis exécuter l'alignement global.

QUESTION 7 : structures secondaires d'ARN II (18 points)

a. (7 points) Considérez la séquence d'ARN ci-dessous avec ses appariements inférés.



Dessinez l'arbre correspondant à ces appariements, puis dessinez la structure d'ARN correspondante. Il n'est pas nécessaire de dessiner chaque nucléotide, mais les appariements et les sous-structures de la molécule doivent apparaître.

b. (11 points) Soit S une séquence d'ARN et P un ensemble d'appariements possibles. Une *tige-boucle* est définie comme un appariement (i, j) avec aucun autre appariement à l'intérieur. Ceci veut dire que pour tout i', j' tels que $i < i' < j' < j$, (i', j') ne sont pas appariés.

La longueur d'une tige-boucle (i, j) est $j - i$. On voudrait limiter la longueur des tige-boucles à 3. Donnez un algorithme permettant de calculer le nombre maximum d'appariements possibles dans S, avec la restriction que chaque tige-boucle a une longueur de 3 ou moins (retournez $-\infty$ si ce n'est pas possible).

Vous pouvez donner votre réponse dans le format que vous souhaitez (récurrences, pseudo-code, texte), mais soyez clairs.

QUESTION 8 : phylogénie parfaite (18 points)

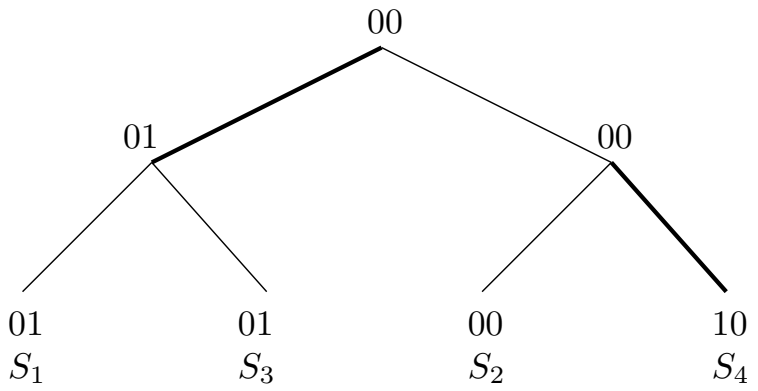
Soit T un arbre avec chaque noeud v étiqueté par une séquence $s(v)$, tel que toutes les séquences ont la même longueur. On suppose que nous travaillons avec l'alphabet $\Sigma = \{0, 1\}$. On dit qu'un changement est requis sur une branche uv à une position i si $s(u)$ et $s(v)$ diffèrent à la position i. La figure ci-bas illustre un arbre étiqueté avec un seul changement par position (la branche gauche en gras est un changement à la position 2, la branche droite en gras à la position 1).

a. (9 points) Supposons que l'on a en entrée des séquences $S_1, \ldots, S_n$ de longueur 1. Par exemple, on pourrait avoir $S_1 = 0, S_2 = 1, S_3 = 1, S_4 = 0$. Montrez qu'il existe toujours un arbre avec les séquences S_i aux feuilles tel qu'une seule branche (ou moins) requière un changement.

b. (9 points) Supposons que l'on a en entrée un arbre T avec étiquettes $s(v)$ pour chaque noeud v. Donnez un algorithme qui détermine si, pour chaque position i, il y a une branche ou moins qui requière un changement à la position i. Donnez ensuite la complexité de votre algorithme, avec justification.

   Dans l'exemple ci-bas, votre algorithme devrait retourner que oui, puisque chaque position ne nécessite qu'un changement.

$$S_1 = 01 \quad S_2 = 00 \quad S_3 = 01 \quad S_4 = 10$$



QUESTION 9 : distance de réarrangements II (18 points)

a. (10 points) Soit A et B deux chromosomes non-signés, donc deux séquences de caractères ordinaires. Rappelons qu'une *délétion* supprime un segment contigu d'une séquence.

Donnez un algorithme qui calcule la distance de délétion de A à B, c'est-à-dire le nombre minimum de délétions à appliquer à A pour obtenir B (ou ∞ si ce n'est pas possible). Donnez ensuite la complexité de votre algorithme, avec justification.

b. (8 points) Considérez les chromosomes $A = a^+b^+c^+$ et $B = b^-c^-a^+$. Montrez comment transformer A en B avec un nombre minimum d'opérations DCJ (vous devez donner le détail des opérations DCJ à effectuer).